Linearity in Process Languages

Mikkel Nygaard
BRICS[*], University of Aarhus
nygaard@brics.dk

Glynn Winskel
Computer Laboratory, University of Cambridge
Glynn.Winskel@cl.cam.ac.uk

Abstract

The meaning and mathematical consequences of linearity (managing without a presumed ability to copy) are studied for a path-based model of processes which is also a model of affine-linear logic. This connection yields an affine-linear language for processes, automatically respecting open-map bisimulation, in which a range of process operations can be expressed. An operational semantics is provided for the tensor fragment of the language. Different ways to make assemblies of processes lead to different choices of exponential, some of which respect bisimulation.

1. Path-based models of processes

In distributed computation it can be hard or impossible for a process to copy a process, while it is generally easy for a process to ignore another process. For this reason an operation on processes associated with distributed computation often has the following property: a computation path of the process arising from the application of the operation to an input process has resulted from at most one computation path of the input process. As we will see, this property expresses that the operation is an affine linear map within a model of linear logic, a fact which carries many consequences.

This article presents a model of processes based on computation paths and so can make precise the sense in which many process operations are associated with linear maps, investigates the consequences of linearity for the important equivalence of bisimulation, and delineates the boundaries of linearity with respect to one, fairly broad, mathematical model, in which processes are represented as presheaves.

Of course some process operations do involve sending and copying code, which can give rise to maps which are not linear. The presheaf model exposes how different degrees of copying lead to different kinds of nonlinear maps, some respecting bisimulation, others not.

Consider processes which, like those of CCS [17] and CSP [10], can perform simple atomic actions, one at a time, among which might 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 [10] 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. This data, 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" of its paths.

To illustrate the idea, suppose that actions are drawn from some alphabet L, and consider processes whose computation paths have the shape of strings of actions, so members of L^*. The substring ordering $\leq$ makes L^* a partial order, and so a category with an arrow from s to t precisely when $s \leq t$. A presheaf over L^* is a functor from the opposite category $(L^*)^{\mathrm{op}}$, where all the arrows are reversed, to the category of sets and functions $\mathbf{Set}$. 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 $s \leq t$, the function $X(s,t) : X(t) \to X(s)$ tells how paths of shape t restrict to subpaths of shape s.

Suppose that we replace the category of sets used in the definition of presheaves by the simple subcategory $\mathbf{2}$, consisting of the empty set, $\emptyset$, and the singleton set, $\mathbf{1}$, with the only non-identity map being $\emptyset \subseteq \mathbf{1}$. A functor X from $(L^*)^{\mathrm{op}}$ to $\mathbf{2}$ is the same as a monotonic function from the reverse order $(L^*)^{\mathrm{op}}$ to the order $\mathbf{2}$, so that if $s \leq t$ then $X(t) \leq X(s)$. When thinking of X as representing a process, $X(s) = \mathbf{1}$ means that the process can perform a path of shape s while $X(s) = \emptyset$ means that it can't. If $X(t) = \mathbf{1}$ and $s \leq t$, then $X(s) = \mathbf{1}$. The functor X is a characteristic

---

[*] Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation.

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 actions to occur concurrently in a computation path, and for individual actions to have a more complicated structure. Later on, processes will be allocated types; the type of a process will specify the kind of computation path shapes it can perform.

2. Processes as presheaves

Let $\mathbb{P}$ be a small category. The category of *presheaves over* $\mathbb{P}$, written $\widehat{\mathbb{P}}$, is the category $[\mathbb{P}^{\mathrm{op}}, \mathbf{Set}]$ with objects the functors from $\mathbb{P}^{\mathrm{op}}$ to the category of sets, and maps the natural transformations between them. In our applications, $\mathbb{P}$ is thought of as consisting of computation-path shapes where a map $e : p \to q$ expresses how p is extended to q.

A presheaf category has all limits and colimits given pointwise, at a particular object, by the corresponding limits or colimits of sets. In particular, a presheaf category has all sums (coproducts) 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 $\varnothing$ with empty contribution at each $p \in \mathbb{P}$. In process terms, a sum of presheaves represents a nondeterministic sum of processes.

2.1. A linear category of presheaf models

A category of presheaves, $\widehat{\mathbb{P}}$, is accompanied by the *Yoneda embedding*, a functor $y_{\mathbb{P}} : \mathbb{P} \to \widehat{\mathbb{P}}$, which fully and faithfully embeds $\mathbb{P}$ in $\widehat{\mathbb{P}}$. For every object p of $\mathbb{P}$, the Yoneda embedding yields $y_{\mathbb{P}}(p) = \mathbb{P}(-, p)$. Presheaves isomorphic to images of objects of $\mathbb{P}$ under the Yoneda embedding are called *representables*.

Via the Yoneda embedding we can regard $\mathbb{P}$ essentially as a full subcategory of $\widehat{\mathbb{P}}$. Moreover, $\widehat{\mathbb{P}}$ is characterized (up to equivalence of categories) as the free colimit completion of $\mathbb{P}$. In other words, $y_{\mathbb{P}}$ satisfies the universal property that for any functor $F : \mathbb{P} \to \mathcal{E}$, where $\mathcal{E}$ is a category with all colimits, there is a colimit-preserving functor $G : \widehat{\mathbb{P}} \to \mathcal{E}$, determined to within isomorphism, such that $F \cong G \circ y_{\mathbb{P}}$—see *e.g.* [15], p. 43:

$$\begin{tikzcd}
\mathbb{P} \arrow[r, "y_{\mathbb{P}}"] \arrow[dr, "\cong", "F"'] & \widehat{\mathbb{P}} \arrow[d, "G"] \\
 & \mathcal{E}
\end{tikzcd}$$

The proof rests on the fact that any presheaf is a colimit of representables. We will use an "inner product" notation and describe G above as $F \cdot -$.

In particular, we can take $\mathcal{E}$ to be a presheaf category $\widehat{\mathbb{Q}}$. As the universal property suggests, colimit-preserving functors between presheaf categories are useful. Define the category $\mathbf{Lin}$ to consist of small categories $\mathbb{P}, \mathbb{Q}, \ldots$ with maps $G : \mathbb{P} \to \mathbb{Q}$ the colimit-preserving functors $\widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$.

By the universal property, colimit-preserving functors $G : \widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$ correspond to within isomorphism to functors $F : \mathbb{P} \to \widehat{\mathbb{Q}}$, and such functors are in 1-1 correspondence with profunctors $\bar{F} : \mathbb{P} \nrightarrow \mathbb{Q}$. Recall that the category of profunctors from $\mathbb{P}$ to $\mathbb{Q}$, written $\mathbf{Prof}(\mathbb{P}, \mathbb{Q})$, is the functor category $[\mathbb{P} \times \mathbb{Q}^{\mathrm{op}}, \mathbf{Set}]$, which clearly equals the category of presheaves $\widehat{\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}}$, and is isomorphic to the functor category $[\mathbb{P}, \widehat{\mathbb{Q}}]$. We thus have the chain of equivalences:

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

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 $\mathbb{P}^{\mathrm{op}}$; clearly presheaves over $\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}$ correspond to presheaves over $(\mathbb{Q}^{\mathrm{op}})^{\mathrm{op}} \times \mathbb{P}^{\mathrm{op}}$, showing how maps $G : \mathbb{P} \to \mathbb{Q}$ correspond to maps $G^{\perp} : \mathbb{Q}^{\mathrm{op}} \to \mathbb{P}^{\mathrm{op}}$ in $\mathbf{Lin}$. The tensor product of $\mathbb{P}$ and $\mathbb{Q}$ is given by the product of small categories $\mathbb{P} \times \mathbb{Q}$ and the function space from $\mathbb{P}$ to $\mathbb{Q}$ by $\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}$. On objects $\mathbb{P}$ and $\mathbb{Q}$, products (written $\mathbb{P} \,\&\, \mathbb{Q}$) and coproducts are both given by $\mathbb{P} + \mathbb{Q}$, the disjoint juxtaposition of $\mathbb{P}$ and $\mathbb{Q}$. As for the exponential ! of linear logic, there's room for choice—see Section 7.

Linear maps preserve colimits. The colimit of the empty diagram is $\varnothing$, to be thought of as a *nil* process, which is unable to perform any computation path. So linear maps, unlike many process operations, always send the nil process to the nil process. We could extend to maps from $!\mathbb{P}$ to $\mathbb{Q}$, for objects $\mathbb{P}$ and $\mathbb{Q}$ in $\mathbf{Lin}$, but by the properties of the exponential, this would allow arbitrary copying of the argument process. All we often need is to allow maps to ignore their arguments and this can be got much more cheaply, by moving to a model of affine linear logic.

2.2. An affine-linear category of presheaf models

Many operations associated with process languages do not preserve sums, so arbitrary colimits. Prefixing operations only preserve connected colimits, that is, colimits of nonempty connected diagrams. Prefixing operations derive from the functor $\lfloor - \rfloor : \widehat{\mathbb{P}} \to \widehat{\mathbb{P}_{\perp}}$. The category $\mathbb{P}_{\perp}$ comprises $\mathbb{P}$ together with a new initial object $\perp$. The functor $\lfloor - \rfloor$ adjoins a "root" to a presheaf X in $\widehat{\mathbb{P}}$ in the sense that $\lfloor X \rfloor(p)$ is a copy of $X(p)$ for any p in $\mathbb{P}$, while $\lfloor X \rfloor(\perp)$ is a singleton set $\{*\}$, the new root being $*$; the restriction maps are extended so that restriction to $\perp$ sends elements to $*$. A map from X to Y in $\widehat{\mathbb{P}}$ is sent to its obvious extension from $\lfloor X \rfloor$ to $\lfloor Y \rfloor$ in $\widehat{\mathbb{P}_\perp}$. Presheaves that to within isomorphism can be obtained as images under $\lfloor - \rfloor$ are called *rooted* [12].

Proposition 2.1 *Any presheaf Y in $\widehat{\mathbb{P}_\perp}$ has a decomposition as a sum of rooted presheaves $Y \cong \Sigma_{i \in Y(\perp)} \lfloor Y_i \rfloor$, where, for $i \in Y(\perp)$, the presheaf Y_i in $\widehat{\mathbb{P}}$ is, to within isomorphism, given as $Y_i(p) = \{x \in Y(p) \mid (Y!_p)x = i\}$ where $!_p$ is the unique arrow $p \to \perp$ in $\mathbb{P}_\perp$.*

Intuitively, thinking of presheaves as processes, the presheaves Y_i, where $i \in Y(\perp)$, in the decomposition of Y are those processes that Y can become after performing the initial action $\perp$.

The *strict Yoneda embedding* $j_{\mathbb{P}} : \mathbb{P}_\perp \to \widehat{\mathbb{P}}$, sends $\perp$ to $\varnothing$ and elsewhere acts like $y_{\mathbb{P}}$. The presheaf category $\widehat{\mathbb{P}}$ with $j_{\mathbb{P}}$ is a free connected-colimit completion of $\mathbb{P}_\perp$. Together they satisfy the universal property that for any functor $F : \mathbb{P}_\perp \to \mathcal{E}$, where $\mathcal{E}$ is a category with all connected colimits, there is a connected-colimit preserving functor $F^\dagger : \widehat{\mathbb{P}} \to \mathcal{E}$, determined to within isomorphism, such that $F \cong F^\dagger \circ j_{\mathbb{P}}$:

$$\begin{tikzcd}
\mathbb{P}_\perp \arrow[r, "j_{\mathbb{P}}"] \arrow[dr, "F"'] & \widehat{\mathbb{P}} \arrow[d, "F^\dagger"] \arrow[dl, phantom, "\cong"] \\
& \mathcal{E}
\end{tikzcd}$$

The central observation on which the proof relies is that any presheaf over $\mathbb{P}$ is a connected colimit of representables (images under $y_{\mathbb{P}}$) together with $\varnothing$, the empty presheaf.

The universal property suggests the importance of connected-colimit preserving functors. Define $\mathbf{Aff}$ to be the category consisting of small categories $\mathbb{P}, \mathbb{Q}, \ldots$, with maps $G : \mathbb{P} \to \mathbb{Q}$ the connected-colimit preserving functors $\widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$. It has $\mathbf{Lin}$ in which the maps preserve arbitrary colimits as a subcategory, one which shares the same objects. We can easily characterise those maps in $\mathbf{Aff}$ which are in $\mathbf{Lin}$:

Proposition 2.2 *Suppose $G : \widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$ is a functor which preserves connected colimits. The following properties are equivalent: (i) G preserves all colimits; (ii) G preserves all coproducts (sums); (iii) G is strict, i.e., $G(\varnothing) = \varnothing$.*

Because $\widehat{\mathbb{P}}$ is the free connected-colimit completion of $\mathbb{P}_\perp$, we obtain the equivalence $\mathbf{Aff}(\mathbb{P}, \mathbb{Q}) \simeq [\mathbb{P}_\perp, \widehat{\mathbb{Q}}]$, and consequently the equivalence $\mathbf{Aff}(\mathbb{P}, \mathbb{Q}) \simeq \mathbf{Lin}(\mathbb{P}_\perp, \mathbb{Q})$. The equivalence is part of an adjunction between $\mathbf{Aff}$ and $\mathbf{Lin}$ regarded as 2-categories, in which the 2-cells are natural transformations. We can easily extend lifting to a 2-functor $(-)_\perp : \mathbf{Aff} \to \mathbf{Lin}$; for $G : \mathbb{P} \to \mathbb{Q}$ in $\mathbf{Aff}$, the functor $G_\perp : \mathbb{P}_\perp \to \mathbb{Q}_\perp$ in $\mathbf{Lin}$ takes $Y \in \widehat{\mathbb{P}_\perp}$ with decomposition $\Sigma_{i \in Y(\perp)} \lfloor Y_i \rfloor$ to $G_\perp(Y) = \Sigma_{i \in Y(\perp)} \lfloor G(Y_i) \rfloor$. Lifting restricts to a 2-comonad on $\mathbf{Lin}$ with $\mathbf{Aff}$ as its coKleisli category. The comonad $(-)_\perp$ has turned the model of linear logic $\mathbf{Lin}$ into a model $\mathbf{Aff}$ of affine linear logic (where the tensor unit is terminal).

2.3. Bisimulation

Bisimulation between presheaves is derived from the notion of open map between presheaves [11, 12].

A morphism $h : X \to Y$, between presheaves X, Y over $\mathbb{P}_\perp$, is *open* iff for all morphisms $e : p \to q$ in $\mathbb{P}_\perp$, any commuting square (on the left below) can be split into two commuting triangles (on the right):

$$\begin{tikzcd}
j_{\mathbb{P}}(p) \arrow[r, "x"] \arrow[d, "j_{\mathbb{P}}(e)"'] & X \arrow[d, "h"] & & j_{\mathbb{P}}(p) \arrow[r, "x"] \arrow[d, "j_{\mathbb{P}}(e)"'] & X \arrow[d, "h"] \\
j_{\mathbb{P}}(q) \arrow[r, "y"'] & Y & & j_{\mathbb{P}}(q) \arrow[r, "y"'] \arrow[ur, dotted, "z"] & Y
\end{tikzcd}$$

That the square commutes means that the path $h \circ x$ in Y can be extended via e to a path y in Y. That the two triangles commute means that the path x can be extended via e to a path z in X which matches y.

Open maps are a generalisation of functional bisimulations, or zig-zag morphisms, known from transition systems [12]. Presheaves in $\widehat{\mathbb{P}}$ are *bisimilar* iff there is a span of open maps between them.[1]

The preservation of connected colimits by a functor between presheaf categories is sufficient to ensure that it preserves open maps and bisimulation.

Proposition 2.3 *[8] Let $G : \widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$ be any connected-colimit preserving functor between presheaf categories. Then G preserves open maps and open-map bisimulation.*

3. Constructions for path orders

The constructions in $\mathbf{Aff}$ below form the basis of a denotational semantics of the affine-linear process language presented in Section 4. The types and open terms of that language will be interpreted, respectively, as objects and arrows of $\mathbf{Aff}$. Actually, only the full subcategory of *path orders*, small partial order categories, is needed, and we'll simplify the discussion accordingly, treating denotations of types as though they were just partial orders. (In fact, all the constructions can be imitated for a domain analogue of presheaf categories, in which $\mathbf{Set}$ is replaced by $\mathbf{2}$ and processes denote trace sets—see Section 1 and [19].)

---

[1] We have chosen here to develop the definition of open map from the strict Yoneda embedding rather than the Yoneda embedding. Maps between presheaves are open w.r.t. strict Yoneda iff they are surjective and open w.r.t. Yoneda.

3.1. Sums and fixed points

Each path order $\mathbb{P}$ is associated with (nondeterministic) sum operations, a map $\Sigma : \boldsymbol{\&}_{i \in I} \mathbb{P} \rightarrow \mathbb{P}$ in $\mathbf{Aff}$ taking a tuple $\langle X_i \rangle_{i \in I}$, to the sum (coproduct) $\Sigma_{i \in I} X_i$ in $\widehat{\mathbb{P}}$. The empty sum yields $\varnothing \in \widehat{\mathbb{P}}$. Finite sums, of size k, are typically written as $X_1 + \cdots + X_k$.

For path orders $\mathbb{P}$ and $\mathbb{Q}$, the category $\mathbf{Aff}(\mathbb{P}, \mathbb{Q})$, being equivalent to $\widehat{(\mathbb{P}_\perp)^{\mathrm{op}} \times \mathbb{Q}}$, has all colimits and in particular all ω-colimits. Any operation $G : \mathbf{Aff}(\mathbb{P}, \mathbb{Q}) \rightarrow \mathbf{Aff}(\mathbb{P}, \mathbb{Q})$ which preserves connected colimits will have a fixed point $\mathit{fix}\, G : \mathbb{P} \rightarrow \mathbb{Q}$, a map in $\mathbf{Aff}$. We will build up the denotation of fixed points out of composition in $\mathbf{Aff}$. The composition $G \circ F$ of maps F in $\mathbf{Aff}(\mathbb{P}, \mathbb{Q})$ and G in $\mathbf{Aff}(\mathbb{Q}, \mathbb{R})$, being got as the application $G(F(-))$, preserves connected colimits in the argument F and colimits in G.

3.2. Tensor

The tensor product $\mathbb{P} \otimes \mathbb{Q}$ of path orders $\mathbb{P}, \mathbb{Q}$ is given by the set $(\mathbb{P}_\perp \times \mathbb{Q}_\perp) \setminus \{(\perp, \perp)\}$, ordered coordinatewise.

Let $F : \mathbb{P} \rightarrow \mathbb{P}'$ and $G : \mathbb{Q} \rightarrow \mathbb{Q}'$. We define $F \otimes G : \mathbb{P} \otimes \mathbb{Q} \rightarrow \mathbb{P}' \otimes \mathbb{Q}'$ as the extension (*cf.* Section 2.2) $H^\dagger$ of a functor $H : (\mathbb{P} \otimes \mathbb{Q})_\perp \rightarrow \widehat{\mathbb{P}' \otimes \mathbb{Q}'}$. Notice that $(\mathbb{P} \otimes \mathbb{Q})_\perp \cong \mathbb{P}_\perp \times \mathbb{Q}_\perp$ and so we can define $H : \mathbb{P}_\perp \times \mathbb{Q}_\perp \rightarrow \widehat{\mathbb{P}' \otimes \mathbb{Q}'}$ by taking

$$(H(p, q))(p', q') = \lfloor F(j_{\mathbb{P}} p) \rfloor (p') \times \lfloor G(j_{\mathbb{Q}} q) \rfloor (q')$$

for $p \in \mathbb{P}_\perp$, $q \in \mathbb{Q}_\perp$ and $(p', q') \in \mathbb{P}' \otimes \mathbb{Q}'$.

The unit for tensor is the empty path order $\mathbb{O}$. Objects $X \in \widehat{\mathbb{P}}$ correspond to maps $\tilde{X} : \mathbb{O} \rightarrow \mathbb{P}$ sending $\varnothing$ to X. Given $X \in \widehat{\mathbb{P}}$ and $Y \in \widehat{\mathbb{Q}}$ we define $X \otimes Y \in \widehat{\mathbb{P} \otimes \mathbb{Q}}$ to be the element pointed to by $\tilde{X} \otimes \tilde{Y} : \mathbb{O} \rightarrow \mathbb{P} \otimes \mathbb{Q}$.

3.3. Function space

The function space of path orders $\mathbb{P} \multimap \mathbb{Q}$ is given by the product of partial orders $(\mathbb{P}_\perp)^{\mathrm{op}} \times \mathbb{Q}$.

We have the following chain of isomorphisms:

$$\begin{aligned} \mathbb{P} \otimes \mathbb{Q} \multimap \mathbb{R} &= ((\mathbb{P} \otimes \mathbb{Q})_\perp)^{\mathrm{op}} \times \mathbb{R} \\ &\cong (\mathbb{P}_\perp)^{\mathrm{op}} \times (\mathbb{Q}_\perp)^{\mathrm{op}} \times \mathbb{R} \cong \mathbb{P} \multimap (\mathbb{Q} \multimap \mathbb{R}) \ . \end{aligned}$$

So that $\widehat{\mathbb{P} \otimes \mathbb{Q} \multimap \mathbb{R}} \cong \widehat{\mathbb{P} \multimap (\mathbb{Q} \multimap \mathbb{R})}$. Thus there is a 1-1 correspondence curry from maps $\mathbb{P} \otimes \mathbb{Q} \rightarrow \mathbb{R}$ to maps $\mathbb{P} \rightarrow (\mathbb{Q} \multimap \mathbb{R})$ in $\mathbf{Aff}$; its inverse is called $\mathit{uncurry}$. We obtain application, $\mathit{app} : (\mathbb{P} \multimap \mathbb{Q}) \otimes \mathbb{P} \rightarrow \mathbb{Q}$, as $\mathit{uncurry}(1_{\mathbb{P} \multimap \mathbb{Q}})$.

We shall write $t\ u$ for the application of t of type $\mathbb{P} \multimap \mathbb{Q}$ to u of type $\mathbb{P}$. The ability to curry justifies the formation of terms $\lambda x.t$ of type $\mathbb{P} \multimap \mathbb{Q}$ by lambda abstraction where t of type $\mathbb{Q}$ is a term with free variable x of type $\mathbb{P}$.

3.4. Products

The product of path orders $\mathbb{P} \mathbin{\&} \mathbb{Q}$ is given by the disjoint union of $\mathbb{P}$ and $\mathbb{Q}$. An object of $\widehat{\mathbb{P} \mathbin{\&} \mathbb{Q}}$ can be identified with a pair (X, Y), with $X \in \widehat{\mathbb{P}}$ and $Y \in \widehat{\mathbb{Q}}$, which provides the projections $\pi_1 : \mathbb{P} \mathbin{\&} \mathbb{Q} \rightarrow \mathbb{P}$ and $\pi_2 : \mathbb{P} \mathbin{\&} \mathbb{Q} \rightarrow \mathbb{Q}$. More general, not just binary, products $\boldsymbol{\&}_{i \in I} \mathbb{P}_i$ with projections π_j, for $j \in I$, are defined similarly. From the universal property of products, a collection of maps $F_i : \mathbb{P} \rightarrow \mathbb{P}_i$, for $i \in I$, can be tupled together to form a unique map $\langle F_i \rangle_{i \in I} : \mathbb{P} \rightarrow \boldsymbol{\&}_{i \in I} \mathbb{P}_i$ with the property that $\pi_j \circ \langle F_i \rangle_{i \in I} = F_j$ for all $j \in I$. The empty product is given by $\mathbb{O}$ and as the terminal object is associated with unique maps $!_{\mathbb{P}} : \mathbb{P} \rightarrow \mathbb{O}$, constantly $\varnothing$, for any path order $\mathbb{P}$.

Because there are empty presheaves we can define maps in $\mathbf{Lin}$ from products to tensors of path orders. For instance, in the binary case, $\sigma : \mathbb{P} \mathbin{\&} \mathbb{Q} \rightarrow \mathbb{P} \otimes \mathbb{Q}$ in $\mathbf{Lin}$ is specified by $(X, Y) \mapsto (X \otimes \varnothing) + (\varnothing \otimes Y)$. The composition of such a map with the diagonal map of the product, *viz.*

$$\delta_{\mathbb{P}} : \mathbb{P} \xrightarrow{\mathit{diag}} \mathbb{P} \mathbin{\&} \mathbb{P} \xrightarrow{\sigma} \mathbb{P} \otimes \mathbb{P}$$

takes X to $(X \otimes \varnothing) + (\varnothing \otimes X)$ and gives a weak form of diagonal map. Analogously, one can define general weak diagonal maps $\delta_{\mathbb{P}^k} : \mathbb{P} \rightarrow \mathbb{P} \otimes \cdots \otimes \mathbb{P}$ in $\mathbf{Lin}$ from $\mathbb{P}$ to k copies of $\mathbb{P}$ tensored together. Weak diagonal maps allow the same argument to be used in several different, though incompatible, ways.

3.5. Prefixed sums

The category $\mathbf{Aff}$ does not have coproducts (since all constant functors are maps of $\mathbf{Aff}$ there can be no initial object, so empty coproduct). However, we can build a useful sum in $\mathbf{Aff}$ with the help of the coproduct of $\mathbf{Lin}$ and lifting. Let $\mathbb{P}_\alpha$, for $\alpha \in A$, be a family of path orders. As their prefixed sum, $\Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha$, we take the disjoint union of the path orders $\mathbb{P}_{\alpha \perp}$, over the underlying set $\bigcup_{\alpha \in A} \{\alpha\} \times \mathbb{P}_{\alpha \perp}$; the latter path order forms a coproduct in $\mathbf{Lin}$ with the obvious injections $\mathit{in}_\beta : \mathbb{P}_{\beta_\perp} \rightarrow \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha$, for $\beta \in A$. The injections $\beta : \mathbb{P}_\beta \rightarrow \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha$ in $\mathbf{Aff}$, for $\beta \in A$, are defined to be the compositions $\beta = \mathit{in}_\beta \lfloor - \rfloor$. Finite prefixed sums are written $\alpha_1 \mathbb{P}_1 + \cdots + \alpha_k \mathbb{P}_k$.

This construction is not a coproduct in $\mathbf{Aff}$. However, it does satisfy a weaker property analogous to the universal property of a coproduct. Suppose $F_\alpha : \mathbb{P}_\alpha \rightarrow \mathbb{Q}$ are maps in $\mathbf{Aff}$ for all $\alpha \in A$. Then, there is a mediating map $F : \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha \rightarrow \mathbb{Q}$ in $\mathbf{Lin}$ determined to within isomorphism such that $F \circ \alpha = F_\alpha$ for all $\alpha \in A$.

Suppose that the family of maps $F_\alpha : \mathbb{P}_\alpha \rightarrow \mathbb{Q}$, with $\alpha \in A$, has the property that each F_α is constantly $\varnothing$ whenever $\alpha \in A$ is different from β and that F_β is $H : \mathbb{P}_\beta \rightarrow \mathbb{Q}$. Write $H_{@\beta} : \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha \rightarrow \mathbb{Q}$ for a choice of mediating map in $\mathbf{Lin}$.

If a term t of type $\mathbb{Q}$ with free variable x of type $\mathbb{P}_\beta$ denotes $H : \mathbb{P}_\beta \to \mathbb{Q}$ in **Aff** and u is of type $\Sigma_{\alpha\in A}\alpha\mathbb{P}_\alpha$, then we shall write $[u > \beta\ x \Rightarrow t]$ for $H_{@\beta}(u)$. This construction "tests" or matches u against the pattern $\beta\ x$ and passes the results of successful matches for x on to t; the possibly multiple results of successful matches are then summed together.

Because prefixed sum is not a coproduct we do not have that tensor distributes over prefixed sum. However there is a map $dist : \mathbb{Q} \otimes \Sigma_{\alpha\in A}\alpha\mathbb{P}_\alpha \to \Sigma_{\alpha\in A}\alpha(\mathbb{Q}\otimes\mathbb{P}_\alpha)$ in **Aff**, expressing a form of distributivity, given as the extension $H^\dagger$ of the functor

$$\begin{array}{l} H : \mathbb{Q}_\perp \times (\Sigma_{\alpha\in A}\alpha\mathbb{P}_\alpha)_\perp \to \Sigma_{\alpha\in A}\widehat{\alpha(\mathbb{Q}\otimes}\mathbb{P}_\alpha) \\ H(q,(\alpha,p)) = y_{\Sigma_{\alpha\in A}\alpha(\mathbb{Q}\otimes\mathbb{P}_\alpha)}(\alpha,(q,p)),\ H(q,\perp) = \varnothing\ . \end{array}$$

3.6. Recursive type definitions

In modelling a process language like CCS but where processes are passed at channels $a \in A$, we require the "least" path orders that satisfy the following equations:

$$\mathbb{P} = \tau\mathbb{P} + \Sigma_a a!\mathbb{C} + \Sigma_a a?\mathbb{F} \quad \mathbb{C} = \mathbb{P}\otimes\mathbb{P} \quad \mathbb{F} = \mathbb{P} \multimap \mathbb{P}\ .$$

We can solve such recursive equations for path orders by several techniques, ranging from the sophisticated method of [7], providing inductive and coinductive characterisations, to simple methods essentially based on inductive definitions. Paralleling [14], path orders under the order

$$\begin{array}{lll} \mathbb{P} \trianglelefteq \mathbb{Q} & \iff & \mathbb{P} \subseteq \mathbb{Q} \quad \text{and} \\ & & (\forall p,p' \in \mathbb{P}.\, p \le_{\mathbb{P}} p' \iff p \le_{\mathbb{Q}} p') \end{array}$$

form a (large) cpo with respect to which all the constructions we have just seen can be made Scott continuous. Solutions to equations like those above are then obtained as (simultaneous) least fixed points.

4. An affine-linear language for processes

Assume that path orders are presented using the constructions with the following syntax:

$$\begin{array}{rl} \mathbb{T} ::= & \mathbb{O} \mid \mathbb{T}_1 \otimes \mathbb{T}_2 \mid \mathbb{T}_1 \multimap \mathbb{T}_2 \mid \mathbb{T}_1 \,\&\, \mathbb{T}_2 \mid \Sigma_{\alpha\in A}\alpha\mathbb{T}_\alpha \mid \\ & P \mid \mu_j P_1,\ldots,P_k.(\mathbb{T}_1,\ldots,\mathbb{T}_k) \end{array}$$

All the constructions have been met earlier with the exception of the notation for recursively defined path orders. P is drawn from a set of variables used in the recursive definition of path orders; $\mu_j P_1,\ldots,P_k.(\mathbb{T}_1,\ldots,\mathbb{T}_k)$ stands for the j-component ($1 \le j \le k$) of the least solution to the defining equations $P_1 = \mathbb{T}_1, \ldots, P_k = \mathbb{T}_k$, in which the expressions $\mathbb{T}_1,\ldots,\mathbb{T}_k$ may contain the P_j's. We'll write $\mu\vec{P}.\vec{\mathbb{T}}$ as an abbreviation for the k-tuple with j-component $\mu_j P_1,\ldots,P_k.(\mathbb{T}_1,\ldots,\mathbb{T}_k)$, and confuse a closed expression for a path order with the path order itself.

The operations of Section 3 form the basis of a syntax of terms (see Figure 1) which will be subject to typing and linearity constraints. The language is similar to that in [1], being based on a form of pattern matching. Accordingly, variables like x in the match $[u > \alpha\ x \Rightarrow t]$ are binding occurrences and bind later occurrences of the variable in the body t. We shall take for granted an understanding of free and bound variables, and substitution on raw terms.

Let $\mathbb{P}_1,\ldots,\mathbb{P}_k,\mathbb{Q}$ be closed expressions for path orders and let the variables $x_1,\ldots,x_k$ be distinct. A syntactic judgement $x_1 : \mathbb{P}_1,\ldots,x_k : \mathbb{P}_k \vdash t : \mathbb{Q}$ stands for a map

$$[\![x_1 : \mathbb{P}_1,\ldots,x_k : \mathbb{P}_k \vdash t : \mathbb{Q}]\!] : \mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k \to \mathbb{Q}$$

in **Aff**. We shall typically write Γ, or Δ, for an environment list $x_1 : \mathbb{P}_1,\ldots,x_k : \mathbb{P}_k$ and most often abbreviate the denotation to $\mathbb{P}_1\otimes\cdots\otimes\mathbb{P}_k \xrightarrow{t} \mathbb{Q}$, or even $\Gamma \xrightarrow{t} \mathbb{Q}$. When the environment list is empty, the corresponding tensor product is the empty path order $\mathbb{O}$.

An affine-linear language will restrict copying and so substitutions of a common term into distinct variables. The counterpart in the model is the absence of a suitable diagonal map from objects $\mathbb{P}$ to $\mathbb{P}\otimes\mathbb{P}$. Consider a term $t(x,y)$, with its free variables x and y shown explicitly, for which

$$x : \mathbb{P}, y : \mathbb{P} \vdash t(x,y) : \mathbb{Q}\ ,$$

corresponding to a map $\mathbb{P}\otimes\mathbb{P} \xrightarrow{t(x,y)} \mathbb{Q}$ in **Aff**. This does not generally entail that $x : \mathbb{P} \vdash t(x,x) : \mathbb{Q}$—there may not be a corresponding map in **Aff**, for example if $t(x,y) = x \otimes y$. There is however a condition on how the variables x and y occur in t which ensures that the judgement $x : \mathbb{P} \vdash t(x,x) : \mathbb{Q}$ holds and that it denotes the map in **Aff** obtained as the composition

$$\mathbb{P} \xrightarrow{\delta_{\mathbb{P}}} \mathbb{P}\otimes\mathbb{P} \xrightarrow{t(x,y)} \mathbb{Q}$$

—using the weak diagonal map seen earlier in Section 3.4. Syntactically, this is assured if the variables x and y are *not crossed* in t according to the following definition:

Definition 4.1 Let t be a raw term. Say a set of variables V is *crossed* in t iff there are subterms of t of the form tensor $t_1 \otimes t_2$, application $(t_1\ t_2)$, or match $[t_1 > p \Rightarrow t_2]$, for which t has free occurrences of variables from V appearing in both t_1 and t_2.

The term-formation rules for the affine language are listed in Figure 2 alongside their interpretations as constructors on morphisms, taking the morphisms denoted by the premises to that denoted by the conclusion (*cf.* [2]).

Proposition 4.2 *Suppose* $\Gamma, x : \mathbb{P} \vdash t : \mathbb{Q}$*. The set* $\{x\}$ *is not crossed in* t*.*

$$
\begin{array}{lll}
t, u, \ldots ::= & x, y, z, \ldots \mid \varnothing \mid \Sigma_{i \in I} t_i \mid & \text{(Variables and sums)} \\
 & \mathit{rec}\, x.t \mid \lambda x.t \mid t\ u \mid & \text{(Recursion, abstraction, and application)} \\
 & \alpha\ t \mid [u > \alpha\ x \Rightarrow t] \mid & \text{(Injections and match for prefixed sums)} \\
 & (t, u) \mid [u > (x, -) \Rightarrow t] \mid [u > (-, x) \Rightarrow t] \mid & \text{(Pairing and match for products)} \\
 & t \otimes u \mid [u > x \otimes y \Rightarrow t] & \text{(Tensor operation and match)}
\end{array}
$$

Figure 1. Raw syntax

Term formation rules

Interpretations

Structural rules:

$$\frac{}{x : \mathbb{P} \vdash x : \mathbb{P}} \quad \text{and} \quad \frac{\Delta \vdash t : \mathbb{P}}{\Gamma, \Delta \vdash t : \mathbb{P}}$$

$$\frac{\Gamma, x : \mathbb{P}, y : \mathbb{Q}, \Delta \vdash t : \mathbb{R}}{\Gamma, y : \mathbb{Q}, x : \mathbb{P}, \Delta \vdash t : \mathbb{R}}$$

$$\frac{}{\mathbb{P} \xrightarrow{1_{\mathbb{P}}} \mathbb{P}} \quad \text{and} \quad \frac{\Delta \xrightarrow{t} \mathbb{P}}{\Gamma \otimes \Delta \xrightarrow{!_\Gamma \otimes t} \mathbb{O} \otimes \mathbb{P} \cong \mathbb{P}}$$

$$\frac{\Gamma \otimes \mathbb{P} \otimes \mathbb{Q} \otimes \Delta \xrightarrow{t} \mathbb{R}}{\Gamma \otimes \mathbb{Q} \otimes \mathbb{P} \otimes \Delta \cong \Gamma \otimes \mathbb{P} \otimes \mathbb{Q} \otimes \Delta \xrightarrow{t} \mathbb{R}}$$

Recursive path orders:

$$\frac{\Gamma \vdash t : \mathbb{T}_j[\mu \vec{P}.\vec{\mathbb{T}}/\vec{P}]}{\Gamma \vdash t : \mu_j \vec{P}.\vec{\mathbb{T}}} \quad \text{and} \quad \frac{\Gamma \vdash t : \mu_j \vec{P}.\vec{\mathbb{T}}}{\Gamma \vdash t : \mathbb{T}_j[\mu \vec{P}.\vec{\mathbb{T}}/\vec{P}]}$$

The premise and conclusion are interpreted as the same map because $\mu_j \vec{P}.\vec{\mathbb{T}}$ and $\mathbb{T}_j[\mu \vec{P}.\vec{\mathbb{T}}/\vec{P}]$ denote equal path orders.

Sums of terms:

$$\frac{}{\Gamma \vdash \varnothing : \mathbb{P}} \quad \text{and} \quad \frac{\Gamma \vdash t_i : \mathbb{P} \quad \text{for all } i \in I}{\Gamma \vdash \Sigma_{i \in I} t_i : \mathbb{P}}$$

$$\frac{}{\Gamma \xrightarrow{\varnothing} \mathbb{P}} \quad \text{and} \quad \frac{\Gamma \xrightarrow{t_i} \mathbb{P} \quad \text{for all } i \in I}{\Gamma \xrightarrow{\langle t_i \rangle_{i \in I}} \mathbin{\&}_{i \in I} \mathbb{P} \xrightarrow{\Sigma} \mathbb{P}}$$

Recursive definition:

$$\frac{\Gamma, x : \mathbb{P} \vdash t : \mathbb{P} \quad \{y, x\} \text{ not crossed in } t \text{ for all } y \text{ in } \Gamma}{\Gamma \vdash \mathit{rec}\, x.t : \mathbb{P}}$$

$$\frac{\Gamma \otimes \mathbb{P} \xrightarrow{t} \mathbb{P}}{\Gamma \xrightarrow{\mathit{fix}\, G} \mathbb{P}}$$

—where for $F : \Gamma \to \mathbb{P}$ the map $G(F) : \Gamma \to \mathbb{P}$ is the composition $\Gamma \xrightarrow{\delta_\Gamma} \Gamma \otimes \Gamma \xrightarrow{1_\Gamma \otimes F} \Gamma \otimes \mathbb{P} \xrightarrow{t} \mathbb{P}$.

Abstraction and application:

$$\frac{\Gamma, x : \mathbb{P} \vdash t : \mathbb{Q}}{\Gamma \vdash \lambda x.t : \mathbb{P} \multimap \mathbb{Q}} \quad \text{and} \quad \frac{\Gamma \vdash t : \mathbb{P} \multimap \mathbb{Q} \quad \Delta \vdash u : \mathbb{P}}{\Gamma, \Delta \vdash t\ u : \mathbb{Q}}$$

$$\frac{\Gamma \otimes \mathbb{P} \xrightarrow{t} \mathbb{Q}}{\Gamma \xrightarrow{\mathit{curry}\, t} \mathbb{P} \multimap \mathbb{Q}} \quad \text{and} \quad \frac{\Gamma \xrightarrow{t} \mathbb{P} \multimap \mathbb{Q} \quad \Delta \xrightarrow{u} \mathbb{P}}{\Gamma \otimes \Delta \xrightarrow{\mathit{app} \circ (t \otimes u)} \mathbb{Q}}$$

Injections and match for prefixed sums:

$$\frac{\Gamma \vdash t : \mathbb{P}_\beta \text{, where } \beta \in A}{\Gamma \vdash \beta\ t : \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha}$$

$$\frac{\Gamma, x : \mathbb{P}_\beta \vdash t : \mathbb{Q}\text{, where } \beta \in A \quad \Delta \vdash u : \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha}{\Gamma, \Delta \vdash [u > \beta\ x \Rightarrow t] : \mathbb{Q}}$$

$$\frac{\Gamma \xrightarrow{t} \mathbb{P}_\beta \text{, where } \beta \in A}{\Gamma \xrightarrow{t} \mathbb{P}_\beta \xrightarrow{\beta} \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha}$$

$$\frac{\Gamma \otimes \mathbb{P}_\beta \xrightarrow{t} \mathbb{Q}\text{, where } \beta \in A \quad \Delta \xrightarrow{u} \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha}{\Gamma \otimes \Delta \xrightarrow{1_\Gamma \otimes u} \Gamma \otimes \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha \xrightarrow{t_{@\beta} \circ \mathit{dist}} \mathbb{Q}}$$

Pairing and match for products (symmetric rule for right projection omitted):

$$\frac{\Gamma \vdash t : \mathbb{P} \quad \Gamma \vdash u : \mathbb{Q}}{\Gamma \vdash (t, u) : \mathbb{P} \mathbin{\&} \mathbb{Q}}$$

$$\frac{\Gamma, x : \mathbb{P} \vdash t : \mathbb{R} \quad \Delta \vdash u : \mathbb{P} \mathbin{\&} \mathbb{Q}}{\Gamma, \Delta \vdash [u > (x, -) \Rightarrow t] : \mathbb{R}}$$

$$\frac{\Gamma \xrightarrow{t} \mathbb{P} \quad \Gamma \xrightarrow{u} \mathbb{Q}}{\Gamma \xrightarrow{\langle t, u \rangle} \mathbb{P} \mathbin{\&} \mathbb{Q}}$$

$$\frac{\Gamma \otimes \mathbb{P} \xrightarrow{t} \mathbb{R} \quad \Delta \xrightarrow{u} \mathbb{P} \mathbin{\&} \mathbb{Q}}{\Gamma \otimes \Delta \xrightarrow{1_\Gamma \otimes (\pi_1 \circ u)} \Gamma \otimes \mathbb{P} \xrightarrow{t} \mathbb{R}}$$

Tensor operation and match:

$$\frac{\Gamma \vdash t : \mathbb{P} \quad \Delta \vdash u : \mathbb{Q}}{\Gamma, \Delta \vdash t \otimes u : \mathbb{P} \otimes \mathbb{Q}}$$

$$\frac{\Gamma, x : \mathbb{P}, y : \mathbb{Q} \vdash t : \mathbb{R} \quad \Delta \vdash u : \mathbb{P} \otimes \mathbb{Q}}{\Gamma, \Delta \vdash [u > x \otimes y \Rightarrow t] : \mathbb{R}}$$

$$\frac{\Gamma \xrightarrow{t} \mathbb{P} \quad \Delta \xrightarrow{u} \mathbb{Q}}{\Gamma \otimes \Delta \xrightarrow{t \otimes u} \mathbb{P} \otimes \mathbb{Q}}$$

$$\frac{\Gamma \otimes \mathbb{P} \otimes \mathbb{Q} \xrightarrow{t} \mathbb{R} \quad \Delta \xrightarrow{u} \mathbb{P} \otimes \mathbb{Q}}{\Gamma \otimes \Delta \xrightarrow{1_\Gamma \otimes u} \Gamma \otimes \mathbb{P} \otimes \mathbb{Q} \xrightarrow{t} \mathbb{R}}$$

Figure 2. Term formation rules and interpretations

$$\begin{aligned}
[u > x \Rightarrow t] &\equiv (\lambda x.t)\ u && \\
[u > \alpha\, p \Rightarrow t] &\equiv [u > \alpha\, x \Rightarrow [x > p \Rightarrow t]] && \text{for a fresh variable } x \\
[u > (p,-) \Rightarrow t] &\equiv [u > (x,-) \Rightarrow [x > p \Rightarrow t]] && \text{for a fresh variable } x \\
[u > p \otimes q \Rightarrow t] &\equiv [u > x \otimes y \Rightarrow [x > p \Rightarrow [y > q \Rightarrow t]]] && \text{for fresh variables } x, y
\end{aligned}$$

Figure 3. General patterns

Exploiting the naturality of the various operations used in the semantic definitions, we can prove a general substitution lemma.

Lemma 4.3 (Substitution) *Suppose*

$$\Gamma, x_1 : \mathbb{P}, \ldots, x_k : \mathbb{P} \vdash t : \mathbb{Q}$$

and that the set of variables $\{x_1, \ldots, x_k\}$ *is not crossed in* t. *Suppose* $\Delta \vdash u : \mathbb{P}$ *where the variables of* Γ *and* Δ *are disjoint. Then,* $\Gamma, \Delta \vdash t[u/x_1, \ldots, u/x_k] : \mathbb{Q}$ *and, (suppressing the types for brevity)*

$$[\![t[u/x_1, \ldots, u/x_k]]\!] \cong [\![t]\!] \circ (1_\Gamma \otimes (\delta_{\mathbb{P}_k} \circ [\![u]\!]))\ .$$

Note that in the case where $k = 1$, the lemma specialises to $[\![t[u/x]]\!] \cong [\![t]\!] \circ (1_\Gamma \otimes [\![u]\!])$. A particular consequence is that linear application amounts to substitution:

Lemma 4.4 *If* $\Gamma \vdash (\lambda x.t)\ u : \mathbb{Q}$, *then* $\Gamma \vdash t[u/x] : \mathbb{Q}$ *and* $[\![\Gamma \vdash (\lambda x.t)\ u : \mathbb{Q}]\!] \cong [\![\Gamma \vdash t[u/x] : \mathbb{Q}]\!]$.

Similarly, we have the expected result for recursion:

Lemma 4.5 *If* $\Gamma \vdash \mathit{rec}\, x.t : \mathbb{P}$, *then* $\Gamma \vdash t[\mathit{rec}\, x.t/x] : \mathbb{P}$ *and* $[\![\Gamma \vdash \mathit{rec}\, x.t : \mathbb{P}]\!] \cong [\![\Gamma \vdash t[\mathit{rec}\, x.t/x] : \mathbb{P}]\!]$.

The next lemma follows directly from the universal properties of prefixed sum (the last property because the mediating map is in **Lin**):

Lemma 4.6 *Properties of prefix match:*

$$\begin{aligned}
(i) &\quad [\![\Gamma \vdash [\alpha\, u > \alpha\, x \Rightarrow t] : \mathbb{Q}]\!] \cong [\![\Gamma \vdash t[u/x] : \mathbb{Q}]\!] \\
(ii) &\quad [\![\Gamma \vdash [\alpha\, u > \beta\, x \Rightarrow t] : \mathbb{Q}]\!] \cong \varnothing \text{ if } \alpha \neq \beta \\
(iii) &\quad [\![\Gamma \vdash [\Sigma_{i \in I} u_i > \alpha\, x \Rightarrow t] : \mathbb{Q}]\!] \\
&\qquad \cong [\![\Gamma \vdash \Sigma_{i \in I} [u_i > \alpha\, x \Rightarrow t] : \mathbb{Q}]\!]
\end{aligned}$$

General patterns It will be convenient for the examples of the next Section to allow general patterns according to

$$p ::= x \mid \alpha\, p \mid (p,-) \mid (-,p) \mid p \otimes q$$

—with p being well-formed if all its variables are distinct. A match $[u > p \Rightarrow t]$ is understood inductively as an abbreviation for a term in the metalanguage, according to Figure 3—the right projection $(-,p)$ is omitted. We write $[u_1 > p_1, \ldots, u_k > p_k \Rightarrow t]$ as an abbreviation of $[u_1 > p_1 \Rightarrow [\cdots [u_k > p_k \Rightarrow t] \cdots]]$.

5. Examples

The affine-linear language is remarkably expressive, as the following examples show. Having denotations in **Aff**, all operations expressible in the language will automatically preserve open-map bisimulation.

5.1. CCS

As in CCS, assume a set of labels A, a complementation operation producing $\bar{a}$ from a label a, with $\bar{\bar{a}} = a$, and a distinct label τ. We can specify the path order $\mathbb{P}$ as

$$\mathbb{P} = \tau\mathbb{P} + \Sigma_{a \in A} a\mathbb{P} + \Sigma_{a \in A} \bar{a}\mathbb{P}\ .$$

Let α range over $A \cup \{\tau\}$. The CCS parallel composition can be defined as the following term Par of type $\mathbb{P} \multimap (\mathbb{P} \multimap \mathbb{P})$:

$$\begin{aligned}
\mathit{rec}\, P.\lambda x.\lambda y.&\Sigma_\alpha [x > \alpha\, x' \Rightarrow \alpha\ (P\ x'\ y)] + \\
&\Sigma_\alpha [y > \alpha\, y' \Rightarrow \alpha\ (P\ x\ y')] + \\
&\Sigma_{a \in A} [x > a\, x', y > \bar{a}\, y' \Rightarrow \tau\ (P\ x'\ y')]\ .
\end{aligned}$$

The other CCS operations are easy to encode. Here, open-map bisimulation coincides with strong bisimulation. We can recover the expansion law for general reasons. Write $X|Y$ for $\mathit{Par}\ X\ Y$ with X, Y terms of type $\mathbb{P}$. Suppose $X = \Sigma_\alpha \Sigma_{i \in I(\alpha)} \alpha\, X_i$ and $Y = \Sigma_\alpha \Sigma_{j \in J(\alpha)} \alpha\, Y_j$. Using Lemmas 4.5 and 4.4, then Lemma 4.6, we get

$$\begin{aligned}
X|Y \cong\ &\Sigma_\alpha [X > \alpha\, x' \Rightarrow \alpha\ (x'|Y)] + \\
&\Sigma_\alpha [Y > \alpha\, y' \Rightarrow \alpha\ (X|y')] + \\
&\Sigma_{a \in A} [X > a\, x', Y > \bar{a}\, y' \Rightarrow \tau\ (x'|y')] \\
\cong\ &\Sigma_\alpha \Sigma_{i \in I(\alpha)} \alpha\ (X_i|Y) + \Sigma_\alpha \Sigma_{j \in J(\alpha)} \alpha\ (X|Y_j) + \\
&\Sigma_{a \in A} \Sigma_{i \in I(a), j \in J(\bar{a})} \tau\ (X_i|Y_j)\ .
\end{aligned}$$

In similar ways it is easy to express CSP and any parallel composition given by a synchronisation algebra [20] in the affine-linear language.

5.2. A linear higher-order process language

Recall the path orders for processes, concretions and abstractions for a higher-order language in Section 3.6. We are chiefly interested in the parallel composition of processes, $\mathit{Par}_{\mathbb{P},\mathbb{P}}$, this time of the uncurried type $\mathbb{P} \otimes \mathbb{P} \multimap \mathbb{P}$ for

Processes in parallel with processes:

$$\begin{aligned} P|Q = {} & \Sigma_\alpha[P > \alpha\ x \Rightarrow \alpha\ (x|Q)] + \Sigma_\alpha[Q > \alpha\ y \Rightarrow \alpha\ (P|y)] + \\ & \Sigma_n[P > \mathit{open}\ n!\ x, Q > \mathit{open}\ n?\ y \Rightarrow \tau\ (x|y)] + \\ & \Sigma_n[P > \mathit{open}\ n?\ x, Q > \mathit{open}\ n!\ y \Rightarrow \tau\ (x|y)] + \\ & \Sigma_n[P > \mathit{mvin}\ n?\ f, Q > \mathit{mvin}\ n!\ (s \otimes r) \Rightarrow \tau\ ((f\ s)|r)] + \\ & \Sigma_n[P > \mathit{mvin}\ n!\ (s \otimes r), Q > \mathit{mvin}\ n?\ f \Rightarrow \tau\ (r|(f\ s))] \end{aligned}$$

Abstractions in parallel with concretions: $F|C = [C > s \otimes r \Rightarrow (F\ s)|r]$

Abstractions in parallel with processes: $F|P = \lambda x.((F\ x)|P)$

Concretions in parallel with processes: $C|P = [C > s \otimes r \Rightarrow s \otimes (r|P)]$

Figure 4. Parallel composition for mobile ambients

clarity. But parallel composition is really a family of mutually dependent operations also including components such as $Par_{\mathbb{F},\mathbb{C}}$ of type $\mathbb{F} \otimes \mathbb{C} \multimap \mathbb{P}$ to say how abstractions compose in parallel with concretions *etc.* All these components can be tupled together in a product using &, and parallel composition defined as a simultaneous recursive definition whose component at $\mathbb{P} \otimes \mathbb{P} \multimap \mathbb{P}$ satisfies

$$\begin{aligned} P|Q = {} & \Sigma_\alpha[P > \alpha\ x \Rightarrow \alpha\ (x|Q)] + \\ & \Sigma_\alpha[Q > \alpha\ y \Rightarrow \alpha\ (P|y)] + \\ & \Sigma_a[P > a?\ f, Q > a!\ (s \otimes r) \Rightarrow \tau\ ((f\ s)|r)] + \\ & \Sigma_a[P > a!\ (s \otimes r), Q > a?\ f \Rightarrow \tau\ (r|(f\ s))]\ , \end{aligned}$$

where, *e.g.*, $P|Q$ abbreviates $Par_{\mathbb{P},\mathbb{P}}\ (P \otimes Q)$. In the summations $a \in A$ and α ranges over $a!, a?, \tau$.

5.3. Mobile ambients with public names

We can translate the Ambient Calculus with public names [4] into the affine-linear language, following similar lines to the linear process-passing language above. Assume a fixed set of ambient names $n, m, \ldots \in N$. Following [5], the syntax of ambients is extended beyond just processes (P) to include concretions (C) and abstractions (F):

$$\begin{aligned} P ::= {} & \varnothing \mid P|P \mid \mathit{rep}\ P \mid n[P] \mid \mathit{in}\ n\ P \mid \mathit{out}\ n\ P \mid \\ & \mathit{open}\ n!\ P \mid \tau\ P \mid \mathit{mvin}\ n!\ C \mid \mathit{mvout}\ n!\ C \mid \\ & \mathit{open}\ n?\ P \mid \mathit{mvin}\ n?\ F \mid x \\ C ::= {} & P \otimes P \qquad F\ ::=\ \lambda x.P \end{aligned}$$

The notation for actions departs a little from that of [5]. Here some actions are marked with ! and others with ?—active (or inceptive) actions are marked by ! and passive (or receptive) actions by ?. We say actions α and β are *complementary* iff one has the form *open* $n!$ or *mvin* $n!$ while the other is *open* $n?$ or *mvin* $n?$ respectively. Complementary actions can synchronise together to form a τ-action. We adopt a slightly different notation for concretions ($P \otimes R$ instead of $\langle P \rangle R$) and abstractions ($\lambda x.P$ instead of $(x)P$) to make their translation into the affine-linear language clear.

The usual conventions are adopted for variables. Terms are assumed to be *linear*, in that a variable appears on at most one side of any parallel compositions within the term, and subterms of the form *rep* P are closed. A replication *rep* P is intended to behave as $P|$ *rep* P so readily possesses a recursive definition in the affine-linear language.

Suitable path orders for ambients are given recursively by (n ranges over N):

$$\begin{aligned} \mathbb{P} = {} & \tau\mathbb{P} + \Sigma_n \mathit{in}\ n\mathbb{P} + \Sigma_n \mathit{out}\ n\mathbb{P} + \Sigma_n \mathit{open}\ n!\mathbb{P} + \\ & \Sigma_n \mathit{mvin}\ n!\mathbb{C} + \Sigma_n \mathit{mvout}\ n!\mathbb{C} + \Sigma_n \mathit{open}\ n?\mathbb{P} + \\ & \Sigma_n \mathit{mvin}\ n?\mathbb{F} \\ \mathbb{C} = {} & \mathbb{P} \otimes \mathbb{P} \qquad \mathbb{F}\ =\ \mathbb{P} \multimap \mathbb{P} \end{aligned}$$

The eight components of the prefixed sum in the equation for $\mathbb{P}$ correspond to eight forms of ambient actions: τ, *in* n, *out* n, *open* $n!$, *mvin* $n!$, *mvout* $n!$, *open* $n?$, and *mvin* $n?$. We obtain the prefixing operations as injections into the appropriate component of the prefixed sum $\mathbb{P}$.

Parallel composition is really a family of operations, one of which is a binary operation between processes but where in addition there are parallel compositions of abstractions with concretions, and even abstractions with processes and concretions with processes. The family of operations

$$\begin{aligned} &(-|-) : \mathbb{F} \otimes \mathbb{C} \multimap \mathbb{P}, \quad (-|-) : \mathbb{C} \otimes \mathbb{F} \multimap \mathbb{P}, \\ &(-|-) : \mathbb{F} \otimes \mathbb{P} \multimap \mathbb{F}, \quad (-|-) : \mathbb{P} \otimes \mathbb{F} \multimap \mathbb{F}, \\ &(-|-) : \mathbb{C} \otimes \mathbb{P} \multimap \mathbb{C}, \quad (-|-) : \mathbb{P} \otimes \mathbb{C} \multimap \mathbb{C} \end{aligned}$$

are defined in a simultaneous recursive definition, see Figure 4 (the remaining cases are given symmetrically). Presheaves X, Y over $\mathbb{P}$ will have decompositions into rooted components, and by Lemma 4.6, their parallel composition satisfies the obvious expansion law.

Ambient creation can be defined recursively in the affine-linear language:

$$\begin{aligned} m[P] = {} & [P > \tau\ x \Rightarrow \tau\ m[x]] + \\ & \Sigma_n[P > \mathit{in}\ n\ x \Rightarrow \mathit{mvin}\ n!\ (m[x] \otimes \varnothing)] + \\ & \Sigma_n[P > \mathit{out}\ n\ x \Rightarrow \mathit{mvout}\ n!\ (m[x] \otimes \varnothing)] + \\ & [P > \mathit{mvout}\ m!\ (s \otimes r) \Rightarrow \tau\ (s|m[r])] + \\ & \mathit{open}\ m?\ P + \mathit{mvin}\ m?\ \lambda y.m[P|y]\ . \end{aligned}$$

The denotations of ambients are determined by their capabilities: an ambient $m[P]$ can perform the internal (τ)

actions of P, enter a parallel ambient ($\mathit{mvin}\ n!$) if called upon to do so by an $\mathit{in}\ n$-action of P, exit an ambient n ($\mathit{mvout}\ n!$) if P so requests through an $\mathit{out}\ n$-action, be exited if P so requests through an $\mathit{mvout}\ m!$-action, be opened ($\mathit{open}\ m?$), or be entered by an ambient ($\mathit{mvin}\ m?$); initial actions of other forms are restricted away. Ambient creation is at least as complicated as parallel composition. This should not be surprising given that ambient creation corresponds intuitively to putting a process behind (so in parallel with) a wall or membrane which if unopened mediates in the communications the process can do, converting some actions to others and restricting some others away. The tree-containment structure of ambients is captured in the chain of $\mathit{open}\ m?$'s that they can perform.

By the properties of prefix-match (Lemma 4.6), there is an expansion theorem for ambient creation. For X with decomposition $X = \Sigma_\alpha \Sigma_{i \in X(\alpha)} \alpha\ X_i$, where α ranges over atomic actions of ambients,

$$\begin{aligned} m[X] \cong\ & \Sigma_{i \in X(\tau)} \tau\ m[X_i] + \\ & \Sigma_n \Sigma_{j \in X(\mathit{in}\ n)} \mathit{mvin}\ n!\ (m[X_j] \otimes \varnothing) + \\ & \Sigma_n \Sigma_{k \in X(\mathit{out}\ n)} \mathit{mvout}\ n!\ (m[X_k] \otimes \varnothing) + \\ & \Sigma_{l \in X(\mathit{mvout}\ m!)} [X_l > s \otimes r \Rightarrow \tau\ (s | m[r])] + \\ & \mathit{open}\ m?X + \mathit{mvin}\ m?(\lambda y. m[X|y])\ . \end{aligned}$$

5.4. Nondeterministic dataflow

The affine-linear language allows us to define processes of the kind encountered in treatments of nondeterministic dataflow. Define $\mathbb{P}$ recursively so that $\mathbb{P} = a\mathbb{P} + b\mathbb{P}$, consisting of streams (or sequences) of a's and b's.

The recursively defined process $A : \mathbb{P} \multimap \mathbb{P}$ selects and outputs a's while ignoring all b's:

$$A = \lambda x.\ [x > a\ x' \Rightarrow a\ (A\ x')] + [x > b\ x' \Rightarrow A\ x']$$

The recursively defined process $F : \mathbb{P} \otimes \mathbb{P}$ produces two identical, parallel streams of a's and b's as output:

$$F = [F > z_1 \otimes z_2 \Rightarrow (a\ z_1) \otimes (a\ z_2) + (b\ z_1) \otimes (b\ z_2)]$$

The recursively defined process $S : \mathbb{P} \multimap (\mathbb{P} \otimes \mathbb{P})$ separates a stream of a's and b's into two streams, the first consisting solely of a's and the second solely of b's:

$$\begin{aligned} S = \lambda x.\ & [x > a\ x', (S\ x') > z_1 \otimes z_2 \Rightarrow (a\ z_1) \otimes z_2] + \\ & [x > b\ x', (S\ x') > z_1 \otimes z_2 \Rightarrow z_1 \otimes (b\ z_2)] \end{aligned}$$

A subcategory of **Aff** supports a "trace operation" to represent processes with feedback loops (see [9]). The trace operation is, however, not definable in the present affine-linear language. It can be shown that if the trace operation of [9] were definable in the presheaf semantics, we could obtain (replacing **Set** by **2**) a compositional relational semantics of nondeterministic dataflow with feedback, shown impossible by Brock&Ackerman [3].

6. Operational semantics

We now consider the *tensor fragment* of the affine-linear language, the fragment obtained by leaving out product (for brevity) and function space (in progress).

We employ the language of general patterns of Section 4. It can be given semantics using judgements of the form $x_1 : \mathbb{P}_1, \ldots, x_k : \mathbb{P}_k \Vdash p : \mathbb{P}$ with the x_i distinct, interpreted as functors $(\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k)_\perp \xrightarrow{a} \mathbb{P}_\perp$ according to Figure 5. (The rule identifying a recursive type with its unfolding is omitted.) As there are no rules for weakening or exchange, the environment Π for which $\Pi \Vdash p : \mathbb{P}$ is given uniquely by p and $\mathbb{P}$. We'll only need the sub-language $p ::= \alpha\ x \mid p \otimes x \mid x \otimes p$. Here, the interpretations map into non-$\perp$ elements and, accordingly, we may view $\llbracket \Pi \Vdash p : \mathbb{P} \rrbracket$ as a functor into $\mathbb{P}$ rather than $\mathbb{P}_\perp$. We obtain *atomic paths* as images $\llbracket \Pi \Vdash p : \mathbb{P} \rrbracket(\perp) \in \mathbb{P}$ for which we'll write

$$a ::= \alpha \perp \mid a \otimes \perp \mid \perp \otimes a\ .$$

Conversely, for each a we may recover a pattern p_a (unique up to variable names) by replacing all $\perp$'s of a with distinct variables. We let atomic paths stand for patterns and write $\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k \Vdash a : \mathbb{P}$ if $x_1 : \mathbb{P}_1, \ldots, x_k : \mathbb{P}_k \Vdash p_a : \mathbb{P}$. Notice that $(\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k)_\perp$, uniquely given by a and $\mathbb{P}$, is isomorphic to the path order above $a \in \mathbb{P}$. We'll write α for $\alpha \perp$.

Given $\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k \Vdash a : \mathbb{P}$ and $\vdash t : \mathbb{P}$, we may restrict $\llbracket \vdash t : \mathbb{P} \rrbracket$ to $(\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k)_\perp$ by applying the map $a^* : \mathbb{P} \to (\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k)_\perp$ of **Lin** got by composition as $X \mapsto X(a-)$. (More generally, we may compose a^* with the maps of **Aff** interpreting open terms, so making sense of (semantic) expressions like $a^* t$ when $\mathbb{P}' \Vdash a : \mathbb{P}$ and $\Gamma \vdash t : \mathbb{P}$.) Identifying t with its denotation, we see that $a^* t$ is a presheaf over a lifted path order and therefore has a decomposition as a sum of rooted presheaves by Proposition 2.1. It turns out that each rooted component has the form $\lfloor t' \rfloor$ for a denotation t' of a term of type $\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k$. Judgements $t \xrightarrow{a} t'$ in the operational semantics will express that $\lfloor t' \rfloor$ is a rooted component of $a^* t$. In fact, derivations of transitions $t \xrightarrow{a} t'$ will be in 1-1 correspondence with components of the decomposition of $a^* t$.

The operational semantics is informed by isomorphisms saying how to find the rooted components of $a^* t$. As an example we have the isomorphisms on the left below, suggesting the rules on the right:

$$\beta^*(\alpha\ t) \cong \begin{cases} \lfloor t \rfloor & \text{if } \alpha = \beta \\ \varnothing & \text{if } \alpha \neq \beta \end{cases} \qquad \frac{}{\alpha\ t \xrightarrow{\alpha} t}$$

$$a^* \Sigma_{i \in I} t_i \cong \Sigma_{i \in I} a^* t_i \qquad \frac{t_j \xrightarrow{a} t'}{\Sigma_{i \in I} t_i \xrightarrow{a} t'}\, j \in I$$

These rules are for closed terms. In the semantics

$$a^*[u > x \otimes y \Rightarrow t] \cong [u > x \otimes y \Rightarrow a^* t]\ ,$$

Typing	Interpretation
$\dfrac{}{x : \mathbb{P} \Vdash x : \mathbb{P}}$	$\dfrac{}{\mathbb{P}_\perp \xrightarrow{1} \mathbb{P}_\perp}$
$\dfrac{\Pi \Vdash p : \mathbb{P}_\alpha \quad \alpha \in A}{\Pi \Vdash \alpha\, p : \Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha}$	$\dfrac{\Pi_\perp \xrightarrow{p} \mathbb{P}_{\alpha\perp} \quad \alpha \in A}{\Pi_\perp \xrightarrow{\lfloor in_\alpha\, p \rfloor} (\Sigma_{\alpha \in A} \alpha \mathbb{P}_\alpha)_\perp}$
$\dfrac{\Pi \Vdash p : \mathbb{P} \quad \Lambda \Vdash q : \mathbb{Q}}{\Pi, \Lambda \Vdash p \otimes q : \mathbb{P} \otimes \mathbb{Q}}$	$\dfrac{\Pi_\perp \xrightarrow{p} \mathbb{P}_\perp \quad \Lambda_\perp \xrightarrow{q} \mathbb{Q}_\perp}{(\Pi \otimes \Lambda)_\perp \xrightarrow{p \times q} (\mathbb{P} \otimes \mathbb{Q})_\perp}$

Figure 5. Semantics of patterns

which suggests that we let t (an open term) take an a-transition in the "environment" $u > x \otimes y$. Syntactically, environments e are lists of such matches. An environment "exports" a set of variables; the empty environment exports the empty set, while $e, u > x \otimes y$ exports what e exports, except the free variables of u, plus x and y. We may formalise this using a judgement $e \vdash x_1 : \mathbb{P}_1, \ldots, x_k : \mathbb{P}_k$ (with the x_i distinct) which denotes the same presheaf over $\mathbb{P}_1 \otimes \cdots \otimes \mathbb{P}_k$ as $[e \Rightarrow x_1 \otimes \cdots \otimes x_k]$. A term t in environment e will be written $e \Mapsto t$. For $e \vdash \Gamma, \Delta$ and $\Gamma \vdash t : \mathbb{P}$, we give the judgement $\vdash e \Mapsto t : \mathbb{P}; \Delta$ the same denotation as the term $[e \Rightarrow t \otimes x_1 \otimes \cdots \otimes x_k]$ where the x_i are now the variables exported by e but not free in t.

Incorporating such environments, the operational rules are shown in Figure 6. The left rules (v_l) and (t_l) have the obvious right counterparts. In (tm) x and y are implicitly renamed to avoid overshadowing of exported variables in the environment $e, u > x \otimes y$. The rules are well-typed:

Lemma 6.1 *Assume* $\vdash e \Mapsto t : \mathbb{P}; \Delta$. $e \Mapsto t \xrightarrow{a} e' \Mapsto t'$ *implies* $\mathbb{P}' \Vdash a : \mathbb{P}$ *and* $\vdash e' \Mapsto t' : \mathbb{P}'; \Delta$.

An ordinal size measure can be defined on terms in environments in such a way that transitions are accompanied by a decrease in size if we replace general recursion by finite unfoldings. Building on this fact, we can prove the main result below which says that rooted components correspond to derivations. It is proved by well-founded induction using an order based on the size measure. The induction hypothesis says that for $\vdash e \Mapsto t : \mathbb{P}; \Delta$ and $\mathbb{P}' \Vdash a : \mathbb{P}$ with $\vec{x} \equiv x_1 \otimes \cdots \otimes x_k$ any subset of the variables in Δ, we have $(a \otimes \perp)^*[e \Rightarrow t \otimes \vec{x}] \cong \Sigma_d \lfloor [e' \Rightarrow t' \otimes \vec{x}] \rfloor$ where d ranges over derivations with conclusion of the form $e \Mapsto t \xrightarrow{a} e' \Mapsto t'$.

Theorem 6.2 *Assume* $\vdash t : \mathbb{P}$ *and* $\mathbb{P}' \Vdash a : \mathbb{P}$. *Then* $a^* t \cong \Sigma_d \lfloor t' \rfloor$ *where the sum is over all derivations* d *with conclusion* $\Mapsto t \xrightarrow{a} \Mapsto t'$.

As the operational semantics stands it gives an interleaving model of the tensor fragment. We have been able to represent the definable presheaves of the tensor fragment as event structures, a representation in which the tensor operation denotes the simple parallel composition of event structures got by juxtaposition; elements of definable presheaves correspond to finite configurations of the event-structure representation, with restriction in the presheaf matched by restriction to a subconfiguration in the event structure [18]. We are currently working on exhibiting the operational semantics as a transition system with independence [20] in order to strengthen the correspondence with the presheaf semantics begun in Theorem 6.2.

$$(\mathsf{r})\,\frac{e \Mapsto t[rec\, x.t/x] \xrightarrow{a} e' \Mapsto t'}{e \Mapsto rec\, x.t \xrightarrow{a} e' \Mapsto t'}$$

$$(\mathsf{v}_l)\,\frac{e_1 \Mapsto u \xrightarrow{a \otimes \perp} e_1' \Mapsto u'}{e_1, u > x \otimes y, e_2 \Mapsto x \xrightarrow{a} e_1', u' > x \otimes y, e_2 \Mapsto x}$$

$$(\mathsf{s})\,\frac{e \Mapsto t_j \xrightarrow{a} e' \Mapsto t'}{e \Mapsto \Sigma_{i \in I} t_i \xrightarrow{a} e' \Mapsto t'}\; j \in I$$

$$(\mathsf{p})\,\frac{}{e \Mapsto \alpha\, t \xrightarrow{\alpha} e \Mapsto t}$$

$$(\mathsf{pm})\,\frac{e \Mapsto u \xrightarrow{\alpha} e' \Mapsto u' \quad e' \Mapsto t[u'/x] \xrightarrow{a} e'' \Mapsto t'}{e \Mapsto [u > \alpha\, x \Rightarrow t] \xrightarrow{a} e'' \Mapsto t'}$$

$$(\mathsf{t}_l)\,\frac{e \Mapsto t \xrightarrow{a} e' \Mapsto t'}{e \Mapsto t \otimes u \xrightarrow{a \otimes \perp} e' \Mapsto t' \otimes u}$$

$$(\mathsf{tm})\,\frac{e, u > x \otimes y \Mapsto t \xrightarrow{a} e', u' > x \otimes y \Mapsto t'}{e \Mapsto [u > x \otimes y \Rightarrow t] \xrightarrow{a} e' \Mapsto [u' > x \otimes y \Rightarrow t']}$$

Figure 6. Operational rules

7. Nonlinearity

Of course code can be copied, and this may lead to maps which are not linear. According to the discipline of linear logic, nonlinear maps from $\mathbb{P}$ to $\mathbb{Q}$ are introduced as linear maps from $!\mathbb{P}$ to $\mathbb{Q}$—the exponential ! applied to $\mathbb{P}$ allows arguments from $\mathbb{P}$ to be copied and discarded freely.

In the domain model of linear logic (see Section 3) $!\mathbb{P}$ can be taken to be the finite-join completion of $\mathbb{P}$. Then, the nonlinear maps in the coKleisli category of ! correspond to Scott continuous functions. A close analogue for presheaf models is to interpret $!\mathbb{P}$ as the finite-colimit completion of $\mathbb{P}$. Note that now $!\mathbb{P}$ is a category, and no longer just a partial order. With this understanding of $!\mathbb{P}$, it can be shown that $\widehat{\mathbb{P}}$ with the inclusion functor $!\mathbb{P} \to \widehat{\mathbb{P}}$ is the free filtered-colimit completion of $!\mathbb{P}$—see [13]. It follows that maps $!\mathbb{P} \to \mathbb{Q}$ in **Lin** correspond, to within isomorphism, to continuous (*i.e.*, filtered-colimit preserving) functors $\widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$. But, unfortunately, continuous functors from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$ need not send open maps to open maps. This raises the question of whether other choices of exponential fit in better with bisimulation.

Bear in mind the intuition that objects of $\mathbb{P}$ correspond to the shapes of computation path a process, represented as a presheaf in $\widehat{\mathbb{P}}$, might perform. An object of $!\mathbb{P}$ should represent a computation path of an assembly of processes each with computation-path shapes in $\mathbb{P}$—the assembly of processes can then be a 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 of processes 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 sub-paths.

This intuition suggests exploring other less liberal ways of copying, *e.g.* without being able to copy after some initial run. We will discover candidates for exponentials $!\mathbb{P}$ based on computation-path shapes of simple assemblies of processes, ones built out of indexed families. We start with an example.

7.1. An example

Since both sum and tensor preserve open maps, the functor copy taking a presheaf X over $\mathbb{P}$ to the presheaf

$$\begin{array}{lr} & \mathit{copy}(X) = 1 + X + X^2 + \cdots + X^k + \cdots \\ \text{over} & !\mathbb{P} = \mathbb{1} + \mathbb{P} + \mathbb{P}^2 + \cdots + \mathbb{P}^k + \cdots \end{array}$$

will preserve open maps. Here the superscripts abbreviate repeated applications of tensor in $\mathbf{Lin}$. So $\mathbb{P}^k$ is the product of k copies of the partial order $\mathbb{P}$, in which the objects are k-tuples of objects of $\mathbb{P}$—in particular, $\mathbb{1}$ is the partial order consisting solely of the empty tuple called 1 above. The presheaf X^k comprises k copies of X tensored together, so that $X^k\langle p_1, \ldots, p_k\rangle = X(p_1) \times \cdots \times X(p_k)$.

By supplying "coefficients" we can obtain various non-linear maps. An appropriate form of polynomial is given by a functor $F : !\mathbb{P} \to \widehat{\mathbb{Q}}$, which splits up into a family of functors $F_k : \mathbb{P}^k \to \widehat{\mathbb{Q}}$, for $k \in \omega$. We can extend F to a functor $F[-] = F \cdot \mathit{copy}(-) : \widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$. For $X \in \widehat{\mathbb{P}}$,

$$F[X] = F_0 + F_1 \cdot X + F_2 \cdot X^2 + \cdots + F_k \cdot X^k + \cdots$$

Because $F \cdot -$ is colimit-preserving it preserves open maps. So does copy. Hence $F[-]$ preserves open maps.

Note, that the original polynomial F is not determined to within isomorphism by the functor $F[-]$ it induces. (We can only hope for such uniqueness if we restrict to polynomials which are symmetric, *i.e.*, such that $F_k \cong F_k \circ \pi$ for all permutations π of the k arguments.)

We write $\mathbf{Poly}(\mathbb{P}, \mathbb{Q})$ for the functor category $[!\mathbb{P}, \widehat{\mathbb{Q}}]$ of polynomials from $\mathbb{P}$ to $\mathbb{Q}$. In order to compose polynomials $F \in \mathbf{Poly}(\mathbb{P}, \mathbb{Q})$ and $G \in \mathbf{Poly}(\mathbb{Q}, \mathbb{R})$ we first define $F^! \in \mathbf{Poly}(\mathbb{P}, !\mathbb{Q})$ by taking $F^!\langle p_1, \ldots, p_n\rangle\langle q_1, \ldots, q_k\rangle$ to be

$$\Sigma_{\mu\langle s_1,\ldots,s_k\rangle=\langle p_1,\ldots,p_n\rangle} F s_1 q_1 \times \cdots \times F s_k q_k \ ,$$

when $\langle p_1, \ldots, p_n\rangle \in !\mathbb{P}$ and $\langle q_1, \ldots, q_k\rangle \in !\mathbb{Q}$. The operation $\mu : !!\mathbb{P} \to \mathbb{P}$ flattens, by concatenation, a tuple $\langle s_1, \ldots, s_k\rangle$ of tuples $s_r = \langle s_{r1}, \ldots, s_{rm_r}\rangle$, for $1 \leq r \leq k$, down to a tuple

$$\mu\langle s_1, \ldots, s_k\rangle = \langle s_{11}, \ldots, s_{1m_1}, \ldots, s_{k1}, \ldots, s_{km_k}\rangle \ .$$

So, the sum is indexed by all ways to partition $\langle p_1, \ldots, p_n\rangle$ into tuples $\langle s_1, \ldots, s_k\rangle$. Now, we can define the composition of polynomials to be $G \circ F = G \cdot (F^! -) \in \mathbf{Poly}(\mathbb{P}, \mathbb{R})$. At $\langle p_1, \ldots, p_n\rangle$ in $!\mathbb{P}$, $G \circ F\langle p_1, \ldots, p_n\rangle$ equals

$$\Sigma_{\mu\langle s_1,\ldots,s_k\rangle=\langle p_1,\ldots,p_n\rangle} G_k \cdot F s_1 \times \cdots \times F s_k \ ,$$

with $F s_1 \times \cdots \times F s_k\langle q_1, \ldots, q_k\rangle = F s_1 q_1 \times \cdots \times F s_k q_k$. This composition is only defined to within isomorphism; polynomials form a bicategory $\mathbf{Poly}$, rather than a category.

Note that $!\mathbb{O} = \mathbb{1}$. In the special case where $F : !\mathbb{O} \to \widehat{\mathbb{Q}}$, so that F merely points to a presheaf X in $\widehat{\mathbb{Q}}$, the composition $G \circ F$ of a polynomial $G : !\mathbb{Q} \to \widehat{\mathbb{R}}$ with the polynomial F is isomorphic to $G[X]$. So certainly compositions of this form preserve open maps and bisimulation.

More generally, the polynomials in $\mathbf{Poly}(\mathbb{P}, \mathbb{Q})$ and $\mathbf{Poly}(\mathbb{Q}, \mathbb{R})$ correspond to presheaves over $(!\mathbb{P})^{\mathrm{op}} \times \mathbb{Q}$ and $(!\mathbb{Q})^{\mathrm{op}} \times \mathbb{R}$, respectively. So under this correspondence polynomials are related by open maps and bisimulation. It can be shown that the composition of polynomials in general preserves open maps between polynomials.

However, the present interpretation of ! fails as a candidate for the exponential of linear logic because $\mathbf{Poly}$ is not cartesian-closed in any reasonable sense. There is an isomorphism of categories $\mathbf{Poly}(\mathbb{R}, \mathbb{P} \,\&\, \mathbb{Q}) \cong \mathbf{Poly}(\mathbb{R}, \mathbb{P}) \times \mathbf{Poly}(\mathbb{R}, \mathbb{Q})$, natural in $\mathbb{R}$ in $\mathbf{Lin}$, showing the sense in which $\mathbb{P} \,\&\, \mathbb{Q}$, given by juxtaposition, remains a product in the bicategory of polynomials. There is also clearly an isomorphism of functor categories $[!\mathbb{P} \times !\mathbb{Q}, \widehat{\mathbb{R}}] \cong [!\mathbb{P}, \widehat{(!\mathbb{Q})^{\mathrm{op}} \times \mathbb{R}}]$. But, in general, $!(\mathbb{P} \,\&\, \mathbb{Q})$ and $!\mathbb{P} \times !\mathbb{Q}$ are not isomorphic, so that $(!\mathbb{Q})^{\mathrm{op}} \times \mathbb{R}$ is not a function space for the polynomials with respect to $- \,\&\, -$. The difficulty boils down to a lack of symmetry in the current definition of $!\mathbb{P}$, where tuples like $\langle p_1, \ldots, p_k\rangle$ and its permutations $\langle p_{\pi(1)}, \ldots, p_{\pi(k)}\rangle$ are not necessarily related by any maps. Nor for that matter, are there any maps from a tuple $\langle p_1, \ldots, p_k\rangle$ to a larger tuple $\langle p_1, \ldots, p_k, \ldots, p_m\rangle$, even though intuitively the larger tuple would be a path of a larger assembly of processes, so arguably an extension of the smaller tuple in which further copies have been invoked.

To allow different kinds of polynomial, polynomials which can take account of the symmetry there exists between different copies and also permit further copies to be invoked as needed, we broaden the picture.

7.2. General polynomials

The example suggests that we take assemblies of processes to be families where we can reindex copies, precisely how being prescribed in $\mathbb{U}$, a subcategory of sets in which the maps are the possible reindexings. A $\mathbb{U}$-family of a category $\mathcal{A}$ comprises $\langle A_i \rangle_{i \in I}$ where $i \in I$, with I an object of $\mathbb{U}$, index objects A_i in $\mathcal{A}$. A map of families $(f, e) : \langle A_i \rangle_{i \in I} \to \langle A'_j \rangle_{j \in J}$ consists of a reindexing function $f : I \to J$ in $\mathbb{U}$ and $e = \langle e_i \rangle_{i \in I}$, a family of maps $e_i : A_i \to A'_{f(i)}$ in $\mathcal{A}$. With the obvious composition we obtain $\mathcal{F}_{\mathbb{U}}(\mathcal{A})$, the category of $\mathbb{U}$-families.

Imitating the example, we define the category of polynomials $\mathbf{Poly}_{\mathbb{U}}(\mathbb{P}, \mathbb{Q})$, from $\mathbb{P}$ to $\mathbb{Q}$, to be the functor category $[\mathcal{F}_{\mathbb{U}}(\mathbb{P}), \mathbb{Q}]$. Under sufficient conditions, that $\mathbb{U}$ is small, has a singleton and dependent sum (a functor $\Sigma : \mathcal{F}_{\mathbb{U}}(\mathbb{U}) \to \mathbb{U}$ collapsing any family of sets in $\mathbb{U}$ to a set in $\mathbb{U}$), we can compose polynomials in the manner of the coKleisli construction. For this we need a "distributive law" converting a family of presheaves into a presheaf over families of paths. It can be shown that provided all the maps in $\mathbb{U}$ (the possible reindexings) are injective, composition of polynomials preserves open maps and bisimulation. If $\mathbb{U}$ contains the empty set, we can specialise composition, as in the example, to obtain a functor $F[-] : \widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$ from F in $\mathbf{Poly}_{\mathbb{U}}(\mathbb{P}, \mathbb{Q})$.

The example is now seen as the special case in which $\mathbb{U}$ consists of subsets, possibly empty, of positive natural numbers $\{1, \ldots, n\}$ with identities as the only maps. In the special case in which $\mathbb{U}$ is the full subcategory of $\mathbf{Set}$ consisting of the empty set and a singleton, polynomials amount to functors $\mathbb{P}_{\perp} \to \widehat{\mathbb{Q}}$ so to maps in $\mathbf{Aff}$. If we take $\mathbb{U}$ to be $\mathbb{I}$ (finite sets with injections), or $\mathbb{B}$ (finite sets with bijections), we can repair an inadequacy in the example; then, $\mathcal{F}_{\mathbb{U}}(\mathbb{P} \,\&\, \mathbb{Q})$ and $\mathcal{F}_{\mathbb{U}}(\mathbb{P}) \times \mathcal{F}_{\mathbb{U}}(\mathbb{Q})$ are isomorphic, so that we obtain a function space for the polynomials with respect to the product $- \,\&\, -$. Both $\mathcal{F}_{\mathbb{I}}$ and $\mathcal{F}_{\mathbb{B}}$ are good candidates for the exponential !—they also behave well w.r.t. bisimulation.

There is a fly in the ointment however. The complete mathematical story, in which one would see the polynomials as maps in a coKleisli construction, uses bicategories and at least pseudo (co)monads on a biequivalent 2-category. At present this theory, even the definitions, are not sufficiently developed (though remedial work has started with Martin Hyland and John Power).

Acknowledgements A good deal of the background for this work was developed with Gian Luca Cattani for his PhD [6]. Discussions with Martin Hyland and John Power have played a crucial role in the work on nonlinearity.

References

[1] S. Abramsky. Computational interpretation of linear logic. Tech. Report 90/20, Dept. of Computing, Imperial College, 1990.

[2] T. Braüner. *An Axiomatic Approach to Adequacy*. BRICS Dissertation Series DS-96-4, 1996.

[3] J. Brock and W. Ackerman. Scenarios: A model of nondeterminate computation. In *Proc. of Formalization of Programming Concepts*. LNCS 107, 1981.

[4] L. Cardelli and A. D. Gordon. Anytime, Anywhere. Modal Logics for Mobile Ambients. In *Proc. POPL'00*.

[5] L. Cardelli and A. D. Gordon. A Commitment Relation for the Ambient Calculus. Note, 2000.

[6] G. L. Cattani. *Presheaf Models for Concurrency*. BRICS Dissertation Series DS-99-1, 1999.

[7] G. L. Cattani, M. Fiore, and G. Winskel. A Theory of Recursive Domains with Applications to Concurrency. In *Proc. of LICS '98*.

[8] G. L. Cattani, A. J. Power, and G. Winskel. A categorical axiomatics for bisimulation. In *Proc. of CONCUR'98*, LNCS 1466, 1998.

[9] T. Hildebrandt, P. Panangaden, and G. Winskel. Relational semantics of nondeterministic dataflow. In *Proc. of CONCUR'98*, LNCS 1466, 1998.

[10] C. A. R. Hoare. A model for communicating sequential processes. *Tech. Report PRG-22, University of Oxford Computing Lab.*, 1981.

[11] A. Joyal and I. Moerdijk. A completeness theorem for open maps. *Annals of Pure and Applied Logic*, 70:51–86, 1994.

[12] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127:164–185, 1996.

[13] G. M. Kelly. *Basic concepts of enriched category theory*. London Math. Soc. Lecture Note Series 64, CUP, 1982.

[14] K. G. Larsen and G. Winskel. Using information systems to solve recursive domain equations effectively. LNCS 173, 1984.

[15] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic*. Springer-Verlag, 1992.

[16] A. R. G. Milner. *A Calculus of Communicating Systems*. LNCS 92, 1980.

[17] A. R. G. Milner. *Communication and concurrency*. Prentice Hall, 1989.

[18] M. Nygaard. Towards an Operational Understanding of Presheaf Models. Progress report, University of Aarhus, 2001.

[19] G. Winskel. A Linear Metalanguage for Concurrency. In *Proc. AMAST'98*, LNCS 1548, 1998.

[20] G. Winskel and M. Nielsen. Models for Concurrency. In *Handbook of Logic in Computer Science, Volume 4*, OUP, 1995