Nominal Domain Theory for Concurrency

David Turner and Glynn Winskel

University of Cambridge Computer Laboratory

Abstract. This paper investigates a methodology of using FM (Fraenkel-Mostowski) sets, and the ideas of nominal set theory, to adjoin name generation to a semantic theory. By developing a domain theory for concurrency within FM sets the domain theory inherits types and operations for name generation, essentially without disturbing its original higher-order features. The original domain theory had a metalanguage HOPLA (Higher Order Process Language) and accordingly this expands to a metalanguage, Nominal HOPLA, with name generation (closely related to an earlier language new-HOPLA). Nominal HOPLA possesses an operational and denotational semantics which are related via soundness and adequacy results, again carried out within FM sets.

Introduction

Fraenkel-Mostowski (FM) set theory provided an early example of a set theory violating the Axiom of Choice (AC). It did this by building a set theory around a basic set of finitely permutable atoms $\mathbb{A}$. Functions had to respect the permutability of atoms, which was sufficient to disallow functions required to fulfill AC. Atoms share the same properties as names in computer science. Most often the precise nature of names is unimportant; what matters is their ability to identify and their distinctness. For this reason FM set theory has begun to play a foundational role in computer science, especially in syntax, making formal previously informal and often inaccurate assumptions about, for example, the freshness of variables in substitution[2, 3]. This paper turns FM set theory to the problem of adjoining names and name generation to a semantic theory, a domain theory for concurrency.

At heart what makes FM set theory important for treating names are adjunctions associated with new-name abstraction. The simplest and best-known adjunction, implicit in [3], is for the category of nominal sets (those FM sets which remain invariant under all finite permutations of names). Its right adjoint δ constructs a form of function space consisting of 'new-name abstractions'. Closely related though less well-known are the adjunctions in FM sets on which this paper hinges. Here the associated functors can only be defined locally w.r.t. the sets of names involved.

Importantly, aside from these name features, FM set theory behaves much like more familiar set theories such as ZF, which is invaluable in transferring developments in a name-free setting into FM sets. For us it will mean that a

path-based domain theory for concurrency can be *systematically* extended with name generation by working within FM set theory.

In the domain theory a process denotes a set of paths in a path order, specifying the type of computations it can do. Path sets provide a fully-abstract denotational semantics for the higher order process language HOPLA[4]. HOPLA was extended with name generation to a language new-HOPLA, able to express for example the pi-Calculus, Higher-Order pi-Calculus and mobile ambients[8]. But providing a denotational semantics was problematic. With the then standard way to adjoin name generation to a category of domains, by moving to a functor category, indexing both processes and their types by the current set of names, it became difficult to show that enough function spaces existed (there is an error in [6]). These problems are obviated by working within FM set theory. The way is open to developing more complicated semantics, such as that based on presheaves over path categories, within FM sets.[1]

1 FM Sets

We provide a brief introduction to Fraenkel-Mostowski (FM) sets[2, 3].

Fix an infinite set of names (or 'atoms') written $\mathbb{A}$. A *finite permutation* of $\mathbb{A}$ is a permutation σ of $\mathbb{A}$ such that $\sigma a \neq a$ for only finitely many $a \in \mathbb{A}$. The collection of all finite permutations of $\mathbb{A}$ forms a group. The group is generated by all transpositions (ab) which swap a name a and a name b.

Imagine building a hierarchy of sets as in ZF, but starting from $\mathbb{A}$ rather than the empty set. The permutation action on the collection of atoms induces a permutation action $\cdot$ on the hierarchy of elements by $\in$-recursion, giving rise to a notion of support. A set $s \subseteq \mathbb{A}$ *supports* the element x if for any finite permutation σ such that $\sigma a = a$ for all $a \in s$ it is also the case that $\sigma \cdot x = x$. If x has a finite support then it has a smallest finite support, written $\mathrm{supp}(x)$. The FM sets are defined to be those elements with hereditarily finite support.

The collection of all FM sets and finitely-supported functions forms a category **FMSet** which has subcategories $\mathbf{FMSet}_s$ comprising sets and functions all of whose supports are contained in the finite set of names s. The subcategory **NSet** ($= \mathbf{FMSet}_\emptyset$) of *nominal* sets consists of those FM sets and functions with empty support.

FM sets allow the usual operations of set theory, though with the proviso that elements must always have finite support. In addition there are important operations associated with names. The binary predicate $x \# y$ expresses that two FM sets x and y have disjoint supports. If $f : \mathbb{A} \to X$ is a finitely-supported function and X is a FM set then $\mathbf{fresh}\, a\, \mathbf{in}\, fa$ denotes the unique $x \in X$ such that $fa = x$ for any $a \in \mathbb{A}$ such that $a \# \langle f, fa \rangle$ as long as such an $a \in \mathbb{A}$ exists. When $X = \{\top, \bot\}$ then $f : \mathbb{A} \to X$ is a predicate on $\mathbb{A}$ and $\mathbf{fresh}\, a\, \mathbf{in}\, fa$ coincides with $\mathit{N}a.fa$ where N is the 'new' quantifier of Pitts and Gabbay. This

[1] This paper summarises Turner's PhD thesis[5], where all proofs and a fuller set of references can be found; we apologise for the paucity of references forced here.

permits the definition of the α-equivalence relation $\sim_\alpha$ between pairs $\langle x, a\rangle$ where x is an FM set and a is a name, by setting

$$\langle x_1, a_1\rangle \sim_\alpha \langle x_2, a_2\rangle \text{ iff } \mathcal{N}\, b.\ (a_1 b)\cdot x_1 = (a_2 b)\cdot x_2.$$

The α-equivalence class $\{\langle a, x\rangle\}_{\sim_\alpha}$ is an FM set written $[a].x$. Note that $\mathrm{supp}([a].x) = \mathrm{supp}(x) \setminus \{a\}$ so that a is 'bound' in $[a].x$. The operation of *concretion* acts so $([a].x)@b =_{\mathrm{def}} (ab)\cdot x$ provided $[a].x \# b$. We write $\mathcal{A}bs(\mathbb{A})$ for the class of α-equivalence classes.

1.1 Name Generation in Nominal Sets

Defining

$$X \otimes Y =_{\mathrm{def}} \{\langle x, y\rangle \mid x \# y\}$$

gives a tensor $\otimes$ on **NSet**. Provided X is a nominal set, α-equivalence $\sim_\alpha$ restricts to an equivalence relation on $X \times \mathbb{A}$. The quotient $(X \times \mathbb{A})/\!\sim_\alpha$ is written δX. For example,

$$\delta\mathbb{A} = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].a \mid a \in \mathbb{A}\} \dot\cup \{\mathbf{fresh}\, a\, \mathbf{in}\, [a].a\} \cong \mathbb{A} \dot\cup \{*\}.$$

The operation δ is the object part of a right adjoint to $(-)\otimes\mathbb{A}$; the counit is given by concretion @. The right adjoint δ constructs a form of function space: for a nominal set X, the nominal set δX consists of 'new-name abstractions' x' which applied to a fresh name a yield $x'@a$ in X. New-name abstractions in δX capture the effect of new-name generation, albeit in a rather subtle way.

1.2 Name Generation in FM Sets

Unfortunately $(-)\otimes\mathbb{A}$ is no longer a functor on the larger category **FMSet**. If names are to appear explicitly in our syntax, in operations and types (the case for Nominal HOPLA—though not new-HOPLA[2]) we are led outside **NSet**, and name generation requires an alternative to the adjunction $(-)\otimes\mathbb{A} \dashv \delta$.

Turner[5] exhibits a suitable adjunction in FM sets given by the situation

$$(-)^{\#a} : \mathbf{FMSet}_s \leftrightarrows \mathbf{FMSet}_{s\dot\cup\{a\}} : \delta_a$$

now local to a finite set of names s with $a \in \mathbb{A}\setminus s$. The left adjoint $(-)^{\#a}$ is defined on objects by $X^{\#a} =_{\mathrm{def}} \{x \in X \mid a \# x\}$ and on arrows by restriction. The right adjoint δ_a can be described as a subset of α-equivalence classes x': on objects

$$\delta_a X =_{\mathrm{def}} \{x' \in \mathcal{A}bs(\mathbb{A}) \mid \mathcal{N}\, b.\, x'@b \in (ab)\cdot X\},$$

and if $f : X \to Y$ is an arrow of $\mathbf{FMSet}_{s\dot\cup\{a\}}$ and $x' \in \delta_a X$ then $\delta_a f(x') =_{\mathrm{def}} \mathbf{fresh}\, b\, \mathbf{in}\, [b].\big(((ab)\cdot f)(x'@b)\big)$. The unit has components $\xi_X : x \mapsto \mathbf{fresh}\, b\, \mathbf{in}\, [b].x$

[2] A parallel to this paper showing how nominal sets **NSet** are sufficient to produce an adequate denotational semantics for new-HOPLA is underway.

and the counit, $\zeta_X : x' \mapsto x'@a$. Notice that if X has empty support then X is a nominal set and $\delta_a X = \delta X$. In particular $\delta_a \mathbb{A} = \delta\mathbb{A} \cong \mathbb{A} \dot\cup \{*\}$. Also if $s' \subseteq s$ it follows that s' and $\mathbb{A} \setminus s'$ are both objects of $\mathbf{FMSet}_s$. In this case $\delta_a s' = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].c \mid c \in s'\} \cong s'$ via the isomorphism above, and $\delta_a(\mathbb{A} \setminus s') = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].c \mid c \in \mathbb{A} \setminus s'\} \dot\cup \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].b\} \cong (\mathbb{A} \setminus s') \dot\cup \{*\}$.

1.3 Name Generation in FM Preorders

We will see that the adjunction for name generation can be imported into other structures, of which preorders are the simplest. An **FM-preorder** is defined, as usual, to comprise $\langle \mathbb{P}, \leq_{\mathbb{P}} \rangle$ where $\mathbb{P}$ and $\leq_{\mathbb{P}}$ are both FM sets such that $\leq_{\mathbb{P}}$ is a reflexive and transitive binary relation on $\mathbb{P}$. The $\in$-recursive nature of the permutation action on FM sets gives rise to a permutation action on FM-preorders, where $\sigma \cdot \mathbb{P} = \{\sigma \cdot p \mid p \in \mathbb{P}\}$ and $p \leq_{\mathbb{P}} p'$ if and only if $\sigma \cdot p \leq_{\sigma\cdot\mathbb{P}} \sigma \cdot p'$. Functions in **FMSet** must be finitely-supported so we define the category **FMPre** to consist of FM-preorders and finitely-supported monotone functions (again the standard definition). For s a finite set of names, $\mathbf{FMPre}_s$ is the subcategory of **FMPre** consisting of only those objects and arrows which are supported by s.

FM-preorders inherit mechanisms for name generation directly from those in FM sets. Let s be a finite set of names and $a \notin s$. For $\langle \mathbb{P}, \leq_{\mathbb{P}} \rangle$ an object of $\mathbf{FMPre}_s$, define

$$\langle \mathbb{P}, \leq_{\mathbb{P}} \rangle^{\#a} = \langle \mathbb{P}^{\#a}, \leq_{\mathbb{P}^{\#a}} \rangle$$

ordered by $\leq_{\mathbb{P}}^{\#a}$ the restriction of $\leq_{\mathbb{P}}$. For $\langle \mathbb{P}, \leq_{\mathbb{P}} \rangle$ an object of $\mathbf{FMPre}_{s\dot\cup\{a\}}$, define $\delta_a \langle \mathbb{P}, \leq_{\mathbb{P}} \rangle = \langle \delta_a\mathbb{P}, \leq_{\delta_a\mathbb{P}} \rangle$ where for p'_1 and p'_2 elements of $\delta_a\mathbb{P}$

$$p'_1 \leq_{\delta_a\mathbb{P}} p'_2 \Leftrightarrow_{\mathrm{def}} \mathsf{N}\, b.\, p'_1@b \leq_{(ab)\cdot\mathbb{P}} p'_2@b.$$

Taking their action on maps to be that of the corresponding functors on **FMSet**, we obtain a functor $(-)^{\#a} : \mathbf{FMPre}_s \to \mathbf{FMPre}_{s\dot\cup\{a\}}$ and its right adjoint δ_a. The adjunction shares the same unit ξ and counit ζ as those for FM-sets.

2 A Linear Category of FM Domains

The development of the domain theory in FM-sets here is substantially the same as an earlier domain theory developed in traditional set theory [4]. The one extra constraint here is that all sets (so subsets and functions) must be finitely-supported.

The objects of the *linear* category **FMLin** are FM-preorders $\mathbb{P}$, thought of as consisting of computation paths with the preorder $p \leq p'$ expressing how a path p extends to a path p'. A path order $\mathbb{P}$ determines a domain $\widehat{\mathbb{P}}$, that of its *path sets*, finitely-supported down-closed sets w.r.t. $\leq_{\mathbb{P}}$, ordered by inclusion. The arrows of **FMLin**, *linear* maps, from $\mathbb{P}$ to $\mathbb{Q}$ are finitely-supported functions from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$ which preserve joins of finitely-supported subsets. The category **FMLin** is monoidal-closed with a tensor given by the product $\mathbb{P} \times \mathbb{Q}$ of FM-preorders

and a corresponding function space by $\mathbb{P}^{op} \times \mathbb{Q}$. The category has all biproducts (where the objects are given by disjoint juxtaposition of preorders) which serve as both products and coproducts.

In fact, the category **FMLin** will have enough structure to form a model of Girard's (classical) linear logic[1]. As usual, one can move to more liberal maps through the use of a suitable comonad (an exponential of linear logic often written !). Here, $!\mathbb{P}$, for an FM-preorder $\mathbb{P}$, will (essentially) consist of the *isolated* elements of the domain $\widehat{\mathbb{P}}$ under inclusion—$!\mathbb{P}$ can be thought of as consisting of compound paths, associated with several runs. The coKleisli category of ! consists of FM-preorders which consist of *continuous* functions between the domains of path sets. However, in the regime of FM sets, we will have to exercise some care in choosing what 'continuous', and also 'isolated', are to mean if fundamental operations of name generation are to be continuous.

2.1 Name Generation in FMLin

FMLin inherits name generation from **FMPre**. Let $s \subseteq_{\text{fin}} \mathbb{A}$ and $a \in \mathbb{A} \setminus s$. There is a name-generation adjunction

$$(-)^{\#a+} \dashv \delta_a^+ : \mathbf{FMLin}_s \leftrightarrows \mathbf{FMLin}_{s \cup \{a\}}.$$

Here $\mathbf{FMLin}_s$ is the subcategory of **FMLin** whose objects and arrows are all supported by s. The key laws are isomorphisms

$$\phi_{\mathbb{P}} : \widehat{\mathbb{P}^{\#a}} \cong \widehat{\mathbb{P}}^{\#a} \quad \text{and} \quad \theta_{\mathbb{Q}} : \delta_a \widehat{\mathbb{Q}} \cong \widehat{\delta_a \mathbb{Q}}$$

natural in $\mathbb{P}$ in $\mathbf{FMPre}_s$ and $\mathbb{Q}$ in $\mathbf{FMPre}_{s \cup \{a\}}$. The isomorphisms and inverses are given concretely as follows:

$$\begin{array}{lll} \phi_{\mathbb{P}}(x) =_{\text{def}} \{p \in x \mid a \# p\} & \text{and} & \phi_{\mathbb{P}}^{-1}(x) =_{\text{def}} x \cup \bigcup_{b \# x, \mathbb{P}} (ab) \cdot x \\ \theta_{\mathbb{Q}}^{(a)}(y') =_{\text{def}} \{q' \mid \text{Ⱨ}\, b.\, q' @ b \in y' @ b\} & \text{and} & \theta_{\mathbb{Q}}^{-1}(y) =_{\text{def}} \mathbf{fresh}\, b\, \mathbf{in}\, [b].\{q \mid [b].q \in y\}. \end{array}$$

Define the functor $(-)^{\#a+} : \mathbf{FMLin}_s \to \mathbf{FMLin}_{s \cup \{a\}}$ to act as $(-)^{\#a}$ on objects and take $f : \mathbb{P} \underset{\text{L}}{\to} \mathbb{Q}$ to $\phi_{\mathbb{Q}} \circ f^{\#a} \circ \phi_{\mathbb{P}}^{-1} : \mathbb{P}^{\#a} \to \mathbb{Q}^{\#a}$, and δ_a^+ similarly.

In [5] it is shown that these functors are well-defined, and that the composite bijection

$$\mathbf{FMLin}_{s \cup \{a\}}(\mathbb{P}^{\#a}, \mathbb{Q}) \cong \mathbf{FMPre}_{s \cup \{a\}}(\mathbb{P}^{\#a}, \widehat{\mathbb{Q}}) \cong \mathbf{FMPre}_s(\mathbb{P}, \delta_a \widehat{\mathbb{Q}}) \cong \mathbf{FMPre}_s(\mathbb{P}, \widehat{\delta_a \mathbb{Q}}) \cong \mathbf{FMLin}_s(\mathbb{P}, \delta_a \mathbb{Q}),$$

got via the isomorphism $\theta_{\mathbb{Q}}$, extends to an adjunction with unit $\widehat{\xi}$ and counit $\widehat{\zeta}$.

3 Continuity in FM Domains

Linear maps are too restrictive to give a semantics for concurrent processes. In[4] the solution was to turn from linear to continuous maps, which preserve only directed joins, via a suitable comonad on **FMLin**. But this is not appropriate in the FM setting: the desired semantics for name generation is not directed-join continuous!

3.1 Continuity and name generation

To see this, we consider a term construction $\mathtt{new}\, a\,.\, t$ inspired by new-HOPLA[8]. Imagine that t denotes a process whose actions lie within the set of names $\mathbb{A}$; so its denotation $[\![t]\!]$ is an element of $\hat{\mathbb{A}}$. By definition the term $\mathtt{new}\, a\,.\, t$ denotes an element of $\widehat{\delta\mathbb{A}}$; its denotation $[\![\mathtt{new}\, a\,.\, t]\!]$ is given as $\theta_{\mathbb{A}}([a].[\![t]\!])$, where $\theta_{\mathbb{A}} : \delta_a(\widehat{\mathbb{A}}) \cong \widehat{\delta_a\mathbb{A}}$ is the isomorphism described in the previous section. The term $\mathtt{new}\, a\,.\, t$ denotes a process with actions of the form $[b].c$ and $[c].c$ from $\delta_a\mathbb{A}$.

Consider now an open term $\mathtt{new}\, a\,.\,(-)$. Substitution into $\mathtt{new}\, a\,.\,(-)$ replaces a with a name a' fresh w.r.t. the argument being substituted, if necessary. Consequently, the substitution of $\mathbb{A}$, with empty support, results in denotation $\theta_{\mathbb{A}}([a].\mathbb{A})$ which can be shown to contain $[a].a$. Whereas, the substitution of $s \subseteq_{\mathrm{fin}} \mathbb{A}$, results in denotation $\theta_{\mathbb{A}}([a'].s)$, with $a' \notin s$, a denotation which cannot contain $[a].a$. As $\mathbb{A} = \bigcup_{s \subseteq_{\mathrm{fin}} \mathbb{A}} s$ is a directed join, this shows that $\mathtt{new}\, a\,.\,(-)$ does not yield a directed-join continuous function.

3.2 FM-Continuity

It makes little difference to classical domain theory whether one uses increasing (ordinal-indexed) sequences or directed sets, because the Axiom of Choice (AC) can be used to move between the two. However, AC does not hold in the theory of FM sets, and this equivalence breaks down. A particular difference is that in any sequence in FM set theory with support s each element of the sequence must also have support s; this 'uniformity' of support does not hold for directed sets in general.

Definition 1. *An FM set X has **uniform support** s if every element $x \in X$ is supported by s. An **FM-directed** set is a directed set with uniform support. If $\mathbb{P}$, $\mathbb{Q}$ are FM-preorders, say that a function $f : \widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$ is **FM-continuous** if it is finitely-supported and preserves joins of FM-directed sets. (Note FM-linear maps are FM-continuous.)*

If X has uniform support then it can be wellordered within FM set theory: AC gives an (external) wellordering and the uniformity ensures that this wellordering is itself finitely-supported. Approximation by FM-directed sets and approximation by (ordinal-indexed) sequences are equivalent in FM set theory.

Returning to the example of $\mathtt{new}\, a\,.\,(-)$, notice that the directed set $\{s \mid s \subseteq_{\mathrm{fin}} \mathbb{A}\}$ does not have a uniform support. Let $X \subseteq \widehat{\mathbb{A}}$ be directed with uniform support s. Then every $x \in X$ is either a subset of s or a superset of $\mathbb{A} \setminus s$, so X is finite. Since X is also directed it contains a maximum element. As a direct consequence, $\mathtt{new}\, a\,.\,(-)$ is FM-continuous.

3.3 FM-Isolated elements

We investigate the structure of isolated elements of domains $\widehat{\mathbb{P}}$, for $\mathbb{P}$ an FM-preorder, with respect to FM-directed sets.

Definition 2. *An element $P \in \widehat{\mathbb{P}}$ is **FM-isolated** (or simply **isolated**) iff for all FM-directed sets $X \subseteq \widehat{\mathbb{P}}$, if $P \subseteq \bigcup X$ then there exists $x \in X$ such that $P \subseteq x$.*

For example, every element of $\widehat{\mathbb{A}}$ is isolated, because any FM-directed subset of $\widehat{\mathbb{A}}$ contains a maximum element (see above). More generally,

Definition 3. *For $\mathbb{P}$ a FM-preorder, F a finite subset of $\mathbb{P}$ and s a finite set of names containing* $\mathrm{supp}(\mathbb{P})$*, define $\langle F\rangle_s =_{\mathrm{def}} \bigcup_{\sigma \# s} \sigma \cdot F$; write $\langle F\rangle_{s\downarrow}$ for the down-closure of $\langle F\rangle_s$.*

Every $x \in \widehat{\mathbb{A}}$ is of this form: either x is finite and hence $x = \langle x\rangle_{\mathrm{supp}(x)}$ or else x is cofinite and hence $x = \langle\{a\}\rangle_{\mathrm{supp}(x)}$ for any $a \in x$. In general:

Lemma 1. *If $F \subseteq_{\mathrm{fin}} \mathbb{P}$ and s is a finite set of names that supports $\mathbb{P}$ then $\langle F\rangle_{s\downarrow}$ is isolated in $\widehat{\mathbb{P}}$. Conversely, if $P \in \widehat{\mathbb{P}}$ is isolated and* $\mathrm{supp}(P, \mathbb{P}) \subseteq s$ *then there exists $F \subseteq_{\mathrm{fin}} \mathbb{P}$ such that $P = \langle F\rangle_{s\downarrow}$.*

3.4 The Category FMCts

Let **FMCts** be the category with objects FM-preorders and arrows from $\mathbb{P}$ to $\mathbb{Q}$ the FM-continuous functions from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$.

We can characterise FM-continuous maps in terms of FM-linear maps whose source is under an exponential !. It is sensible to define $!\mathbb{P}$ as comprising the FM-isolated elements of $\widehat{\mathbb{P}}$ ordered by inclusion. However, with an eye to defining recursive types, we instead define $!\mathbb{P}$ to be the equivalent FM-preorder with elements $\langle F\rangle_s$ where $F \subseteq_{\mathrm{fin}} \mathbb{P}$ and s supports $\mathbb{P}$; its order is given by taking $P \leq_{!\mathbb{P}} P'$ whenever $\forall p \in P \exists p' \in P'.\, p \leq_{\mathbb{P}} p'$.

Each $\widehat{\mathbb{P}}$ is the free FM-directed-join completion of $!\mathbb{P}$. (The order $\widehat{\mathbb{P}}$ is algebraic with respect to approximation by FM-directed sets.) It follows that ! extends to functor making an adjunction $\mathbf{FMLin}(!\mathbb{P}, \mathbb{Q}) \cong \mathbf{FMCts}(\mathbb{P}, \mathbb{Q})$, where the inclusion is right adjoint to the !. Its unit $\eta_{\mathbb{P}} : \mathbb{P} \underset{\mathbf{C}}{\to} !\mathbb{P}$ is given concretely by $\eta_{\mathbb{P}} X = \{P \in !\mathbb{P} \mid P \subseteq X\}$. The adjunction satisfies the conditions Benton *et al* proposed for a model of linear logic[1].

3.5 Name Generation in FMCts

We inherit adjunctions

$$(-)^{\#a++} \dashv \delta_a^{++} : \mathbf{FMCts}_s \leftrightarrows \mathbf{FMCts}_{s \cup \{a\}}$$

supporting name generation in **FMCts** from the adjunctions $(-)^{\#a+} \dashv \delta_a^{+}$ on the linear categories. Here $s \subseteq_{\mathrm{fin}} \mathbb{A}$ and $a \in \mathbb{A} \setminus s$ and $\mathbf{FMCts}_s$ is the subcategory of **FMCts** supported by s. In detail, $(-)^{\#a++}$ and δ_a^{++} act respectively as $(-)^{\#a}$ and δ_a on objects. The arrow $f : \mathbb{P} \underset{\mathbf{C}}{\to} \mathbb{Q}$ of $\mathbf{FMCts}_s$ is taken to the composite $f^{\#a++} =_{\mathrm{def}} \phi_{\mathbb{Q}} \circ f^{\#a} \circ \phi_{\mathbb{P}}^{-1}$ and the arrow $g : \mathbb{P} \underset{\mathbf{C}}{\to} \mathbb{Q}$ of $\mathbf{FMCts}_{s \cup \{a\}}$ is taken to

$\delta_a^{++} g =_{\text{def}} \theta_{\mathbb{Q}} \circ \delta_a g \circ \theta_{\mathbb{P}}^{-1}$. These definitions coincide with those of $(-)^{\#a+}$ and δ_a^+ on linear arrows.

Via an isomorphism $!((-)^{\#a}) \cong (!(-))^{\#a}$, analogous to ϕ^{-1} of section 2.1, we obtain as a composite the bijection

$$\begin{aligned}\mathbf{FMCts}_{s\cup\{a\}}(\mathbb{P}^{\#a}, \mathbb{Q}) &\cong \mathbf{FMLin}_{s\cup\{a\}}(!(\mathbb{P}^{\#a}), \mathbb{Q}) \cong \mathbf{FMLin}_{s\cup\{a\}}((!\mathbb{P})^{\#a}, \mathbb{Q}) \\ &\cong \mathbf{FMLin}_s(!\mathbb{P}, \delta_a\mathbb{Q}) \cong \mathbf{FMCts}_s(\mathbb{P}, \delta_a\mathbb{Q}),\end{aligned}$$

of the adjunction $(-)^{\#a++} \dashv \delta_a^{++}$, with unit $\widehat{\xi}$ and counit $\widehat{\zeta}$ —see [5].

The machinery of freshness, the functors $(-)^{\#a}$ and the isomorphisms $\phi_{\mathbb{P}} : \widehat{\mathbb{P}^{\#a}} \to \widehat{\mathbb{P}^{\#a}}$, can be extended to model freshness with respect to a finite set of names s. This is used to capture 'freshness assumptions' in the type system: a variable of type $\mathbb{P}^{\#s}$ insists that it receives input that is fresh for s, and a term of type $\mathbb{P}^{\#s}$ avoids the names in s in its evaluation. Concretely, $\mathbb{P}^{\#s} = \{p \in \mathbb{P} \mid p \# s\}$ with order given by the restriction of the order on $\mathbb{P}$, while $\phi_{\mathbb{P}}^{(s)} x = \{p \in x \mid p \# s\}$, for $x \in \widehat{\mathbb{P}^{\#s}}$.

4 Nominal HOPLA

Nominal HOPLA is an expressive calculus for higher-order processes with nondeterminism and name-binding. It can be seen as a straightforward extension of HOPLA with terms of the form $\mathtt{new}\, a.t$ and $t\mathtt{[}a\mathtt{]}$ which arise directly from the adjunction $(-)^{\#a++} \dashv \delta_a^{++}$. Its syntax is defined in FM sets.

4.1 Syntax

Fix a set of term variables $\mathtt{x}, \mathtt{y}, \ldots$ and a set of type variables $P, \ldots$, each invariant under the permutation action. Types are given by the grammar

$$\mathbb{P}, \mathbb{Q} ::= P \mid !\mathbb{P} \mid \mathbb{Q} \to \mathbb{P} \mid \delta\mathbb{P} \mid \bigoplus_{\ell \in L} \mathbb{P}_\ell \mid \mu_j \boldsymbol{P}.\, \mathbb{P}\,,$$

where P is a type variable, $\boldsymbol{P}$ is a list of type variables, and $\mu_j \boldsymbol{P}.\, \mathbb{P}$ binds $\boldsymbol{P}$, and a nominal set L is used to index components of a sum type (a biproduct in **FMLin**).

A closed type is a type with no free variables, and in the following, closed types are normally simply called 'types'.

Terms and actions are given by mutually recursive grammars. **Terms** are given by the following grammar, where $\mathtt{x}$ ranges over variables, a ranges over names, s over finite sets of names, p over actions, ℓ over labels and $\mathbb{P}$ over types.

$$\begin{aligned} t, u ::= \quad & \mathtt{x} \mid \mathtt{rec}\,\mathtt{x}.t \mid \textstyle\sum_{i\in I} t_i \mid \mathtt{!}t \mid \mathtt{[}u \mathtt{>} p\mathtt{(x:}\mathbb{P} \mathtt{\#}\ s\mathtt{)\ =>}\ t\mathtt{]} \mid \\ & \lambda\mathtt{x}.t \mid t\mathtt{(}u\mathtt{:}\mathbb{P}\mathtt{)} \mid \mathtt{new}\, a.t \mid t\mathtt{[}a\mathtt{]} \mid \ell\mathtt{:}t \mid \pi_\ell t \mid \mathtt{abs}\, t \mid \mathtt{rep}\, t \end{aligned}$$

The forms $\mathtt{rec}\,\mathtt{x}.t$, $\mathtt{[}u \mathtt{>} p\mathtt{(x:}\mathbb{P} \mathtt{\#}\ s\mathtt{)\ =>}\ t\mathtt{]}$ and $\lambda\mathtt{x}.t$ all bind $\mathtt{x}$ in t, and the set of free variables of t is defined in the usual way. The form $\mathtt{new}\, a.t$ binds the name a in t. In a nondeterministic sum the mapping $i \mapsto t_i$ is a finitely supported function from a nominal set I. Write $\mathtt{nil}$ for the empty sum.

Actions play a central role in the operational semantics of Nominal HOPLA—section 4.3. The grammar of actions, labelling the transitions in the operational semantics, is given as follows where t ranges over closed terms, a ranges over names and ℓ over labels.

$$p ::= \texttt{!} \mid \ell\texttt{:}p \mid t \mapsto p \mid \texttt{abs}\ p \mid \texttt{new}\, a.\, p$$

The form $\texttt{new}\, a.\, p$ binds the name a in p.

Actions and terms form nominal sets where the permutation actions are given by the obvious structural recursion.

Substitution Substitution $t[v/y]$ of term v for variable y in a term t is defined as usual. The substitution is capture-avoiding in both variables and names, in the sense that for substitution into a term of the forms $\texttt{rec x.}t$, $\texttt{[}u\ \texttt{>}\ p\texttt{(x:}\mathbb{P}\ \texttt{\#}\ s\texttt{)}\ \texttt{=>}\ t\texttt{]}$ and $\lambda\texttt{x.}t$ the variable $\texttt{x}$ is assumed not to be free in v, and for substitution into a term of the form $\texttt{new}\, a\texttt{.}t$ the name a is chosen to be fresh for v.

4.2 Typing Rules

For Terms An environment $\Gamma = \texttt{x}_1 : {\mathbb{P}_1}^{\#s_1}, \ldots, \texttt{x}_n : {\mathbb{P}_n}^{\#s_n}$ where $\texttt{x}_1, \cdots, \texttt{x}_n$ are distinct variables, $\mathbb{P}_1, \cdots, \mathbb{P}_n$ are types and $s_1, \cdots, s_n$ are finite sets of names. The intended meaning of $\texttt{x} : \mathbb{P}^{\#s}$ is that the variable $\texttt{x}$ takes values of type $\mathbb{P}$ that are assumed to be fresh for s.

Terms of Nominal HOPLA are typed with judgements of the form $\Gamma \vdash_s t : \mathbb{P}$, where Γ is an environment, s is a finite set of names, t is a term and $\mathbb{P}$ is a type. The type $\mathbb{P}$ describes the actions that the term may perform. The environment Γ records types and freshness assumptions for the variables of t. The set s represents the 'current' set of names.

Structural rules. Weakening: the environment may be extended with extra variables. *Exchange:* two variables in the environment may be exchanged. *Contraction:* a pair of variables (with equal types) may be replaced by a single variable. In addition to these standard rules are two rules associated with names:

Fresh-Weakening. It is possible to impose extra freshness assumptions on a variable.

$$\frac{\Gamma, \texttt{x} : \mathbb{Q}^{\#s''} \vdash_s t : \mathbb{P}}{\Gamma, \texttt{x} : \mathbb{Q}^{\#s'} \vdash_s t : \mathbb{P}}\ (s'' \subseteq s' \subseteq s)$$

Support-Weakening. It is possible to extend the 'current' set s of names.

$$\frac{\Gamma \vdash_{s'} t : \mathbb{P}}{\Gamma \vdash_s t : \mathbb{P}}\ (s' \subseteq s)$$

Variable. A bare variable is typed by the environment in the obvious fashion.

$$\frac{-}{\texttt{x} : \mathbb{P}^{\#\varnothing} \vdash_\varnothing \texttt{x} : \mathbb{P}}$$

Prefix. The term constructor `!` takes a term t to a term $!t$ that intuitively may perform an anonymous action `!` and resume as t. The possible action `!` is recorded in the type.

$$\frac{\Gamma \vdash_s t : \mathbb{P}}{\Gamma \vdash_s\ !t : !\mathbb{P}}$$

Match. A term of the form $[u$ `>` q`(x:`$\mathbb{Q}'$ `#` s'`)` `=>` $t]$ intuitively matches the output of u against the action q and feeds the resumption of u into the variable `x` in t. If `x` has some freshness assumptions imposed on it then u and q must satisfy those assumptions. The side condition that $s'' \subseteq s \setminus s'$ is assumed.

$$\frac{\Gamma, \mathtt{x} : \mathbb{Q}'^{\#s'} \vdash_s t : \mathbb{P} \quad \Lambda \vdash_{s''} u : \mathbb{Q} \quad \vdash_{s''} \mathbb{Q} : q : \mathbb{Q}'}{\Gamma, \Lambda^{\#s'} \vdash_s\ [u\ \mathtt{>}\ q\mathtt{(x:}\mathbb{Q}'\ \mathtt{\#}\ s'\mathtt{)}\ \mathtt{=>}\ t] : \mathbb{P}}$$

Recursion. A term of the form $\mathtt{rec\,x.}t$ intuitively acts as its unfolding $t[\mathtt{rec\,x.}t/\mathtt{x}]$, so that `x` must be of the same type as t.

$$\frac{\Gamma, \mathtt{x} : \mathbb{P}^{\#\varnothing} \vdash_s t : \mathbb{P}}{\Gamma \vdash_s \mathtt{rec\,x.}t : \mathbb{P}}$$

Function Abstraction and Application. A term t of type $\mathbb{P}$ may be abstracted with respect to the free variable `x` of type $\mathbb{Q}$ to leave a term $\lambda\,\mathtt{x.}t$ of type $\mathbb{Q} \to \mathbb{P}$ that can in turn be applied to a term of type $\mathbb{Q}$ in the usual fashion.

$$\frac{\Gamma, \mathtt{x} : \mathbb{Q}^{\#\varnothing} \vdash_s t : \mathbb{P}}{\Gamma \vdash_s \lambda\,\mathtt{x.}t : \mathbb{Q} \to \mathbb{P}} \qquad \frac{\Gamma \vdash_s t : \mathbb{Q} \to \mathbb{P} \quad \Lambda \vdash_s u : \mathbb{Q}}{\Gamma, \Lambda \vdash_s t\mathtt{(}u\mathtt{:}\mathbb{Q}\mathtt{)} : \mathbb{P}}$$

Labelling and Label Projection. The actions of a term t may be 'tagged' with a label ℓ_0 by forming the term $\ell_0\mathtt{:}t$. The effect of the term former π_{ℓ_0} is that terms of the form $\pi_{\ell_0}t$ can perform only the actions of t that are tagged by the label ℓ_0. In both of these rules the support of ℓ_0 must be contained in s.

$$\frac{\Gamma \vdash_s t : \mathbb{P}_{\ell_0}}{\Gamma \vdash_s\ \ell_0\mathtt{:}t : \bigoplus_{\ell \in L} \mathbb{P}_\ell} \qquad \frac{\Gamma \vdash_s t : \bigoplus_{\ell \in L} \mathbb{P}_\ell}{\Gamma \vdash_s\ \pi_{\ell_0} t : \mathbb{P}_{\ell_0}}$$

Nondeterministic Sum. A term $\sum_{i \in I} t_i$ makes a nondeterministic choice amongst its components and behaves as the chosen component. The mapping $i \mapsto \Gamma \vdash_{s_i} t_i : \mathbb{P}$ must be supported by s.

$$\frac{\Gamma \vdash_{s_i} t_i : \mathbb{P} \qquad \text{each } i \in I}{\Gamma \vdash_s \sum_{i \in I} t_i : \mathbb{P}}$$

Recursive Type Folding and Unfolding. As the recursively-defined type $\mu_j \boldsymbol{P}.\ \mathbb{P}$ is isomorphic (and not equal) to its unfolding $\mathbb{P}_j[\mu \boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}]$ it is necessary to record any uses of the isomorphism $\mathbf{abs} = \mathbf{rep}^{-1}$ in the syntax of the term.

$$\frac{\Gamma \vdash_s t : \mathbb{P}_j[\mu \boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}]}{\Gamma \vdash_s \mathtt{abs}\,t : \mu_j \boldsymbol{P}.\ \mathbb{P}} \qquad \frac{\Gamma \vdash_s t : \mu_j \boldsymbol{P}.\ \mathbb{P}}{\Gamma \vdash_s \mathtt{rep}\,t : \mathbb{P}_j[\mu \boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}]}$$

Name Abstraction and Application. The only alteration to the syntax of terms over that of conventional HOPLA is the following pair of term formers. Intuitively the term $\mathtt{new}\, a\,.\, t$ can perform the same actions as t with the name a bound, whereas the term $t\,[a]$ takes the outputs of t, which contain a bound name since t is of type $\delta\mathbb{P}$, and instantiates that name as a. In both cases the side-condition $a \notin s$ is assumed.

$$\frac{\Gamma^{\#a} \vdash_{s \cup \{a\}} t : \mathbb{P}}{\Gamma \vdash_s \mathtt{new}\, a\,.\, t : \delta\mathbb{P}} \qquad \frac{\Gamma \vdash_s t : \delta\mathbb{P}}{\Gamma^{\#a} \vdash_{s \cup \{a\}} t\,[a] : \mathbb{P}}$$

For Actions Actions are typed by judgements of the form $\vdash_s \mathbb{P} : p : \mathbb{P}'$ where s is a finite set of names and $\mathbb{P}$ and $\mathbb{P}'$ are types. Intuitively, a term of type $\mathbb{P}$ may perform an action p and resume as a term of type $\mathbb{P}'$.

$$\frac{\vdash_{s'} \mathbb{P} : p : \mathbb{P}'}{\vdash_s \mathbb{P} : p : \mathbb{P}'}\ (s' \subseteq s) \qquad \frac{-}{\vdash_\varnothing\, !\mathbb{P} : \mathtt{!} : \mathbb{P}} \qquad \frac{\vdash_s \mathbb{P} : p : \mathbb{P}' \quad \vdash_s u : \mathbb{Q}}{\vdash_s \mathbb{Q} \to \mathbb{P} : u \mapsto p : \mathbb{P}'}$$

$$\frac{\vdash_s \mathbb{P}_{\ell_0} : p : \mathbb{P}'}{\vdash_s \bigoplus_{\ell \in L} \mathbb{P}_\ell : \ell_0 : p : \mathbb{P}'} \qquad \frac{\vdash_s \mathbb{P}_j[\mu \boldsymbol{P}.\, \mathbb{P}/\boldsymbol{P}] : p : \mathbb{P}'}{\vdash_s \mu_j \boldsymbol{P}.\, \mathbb{P} : \mathtt{abs}\ p : \mathbb{P}'} \qquad \frac{\vdash_{s \cup \{a\}} \mathbb{P} : p : \mathbb{P}'}{\vdash_s \delta\mathbb{P} : \mathtt{new}\, a.\, p : \delta\mathbb{P}'}$$

Substitution respects the type system of Nominal HOPLA, as long as freshness assumptions are themselves respected.

Lemma 2 (Syntactic Substitution Lemma). *Suppose that t and v satisfy $\Gamma, \mathtt{y} : \mathbb{R}^{\#r} \vdash_s t : \mathbb{P}$ and $\Delta \vdash_{s_1} v : \mathbb{R}$ where $s_1 \cap r = \varnothing$ and the variables in Γ are distinct from those in Δ. Then $\Gamma, \Delta^{\#r} \vdash_{s \cup s_1} t[v/\mathtt{y}] : \mathbb{P}$.*

4.3 Operational Semantics

Nominal HOPLA is given an operational semantics in the style of a labelled transition system. That a term t such that $\vdash t : \mathbb{P}$ may perform an action p such that $\vdash \mathbb{P} : p : \mathbb{P}'$ and resume as the term t' is written $\mathbb{P} : t \xrightarrow{p} t'$. The operational semantics of closed, well-typed terms are defined below.

$$\frac{\mathbb{P} : t[\mathtt{rec}\, \mathtt{x}\,.\, t/\mathtt{x}] \xrightarrow{p} t'}{\mathbb{P} : \mathtt{rec}\, \mathtt{x}\,.\, t \xrightarrow{p} t'} \qquad \frac{\mathbb{P} : t_{i_0} \xrightarrow{p} t'}{\mathbb{P} : \sum_{i \in I} t_i \xrightarrow{p} t'}$$

$$\frac{-}{!\mathbb{P} : !t \xrightarrow{!} t} \qquad \frac{\mathbb{P} : t[u'/\mathtt{x}] \xrightarrow{p} t' \ \ \mathbb{Q} : u \xrightarrow{q} u' \ \vdash \mathbb{Q} : q : \mathbb{Q}'}{\mathbb{P} : [u > q(\mathtt{x}{:}\mathbb{Q}'\ \#\ s') \Rightarrow t] \xrightarrow{p} t'}$$

$$\frac{\mathbb{P} : t \xrightarrow{p} t'}{\delta\mathbb{P} : \mathtt{new}\, a\,.\, t \xrightarrow{\mathtt{new}\, a.\, p} \mathtt{new}\, a\,.\, t'} \qquad \frac{\delta\mathbb{P} : t \xrightarrow{\mathtt{new}\, a.\, p} \mathtt{new}\, a\,.\, t'}{\mathbb{P} : t\,[a] \xrightarrow{p} t'}$$

$$\frac{\mathbb{P} : t[u/\mathtt{x}] \xrightarrow{p} t'}{\mathbb{Q} \to \mathbb{P} : \lambda\, \mathtt{x}\,.\, t \xrightarrow{u \mapsto p} t'} \qquad \frac{\mathbb{Q} \to \mathbb{P} : t \xrightarrow{u \mapsto p} t'}{\mathbb{P} : t(u{:}\mathbb{Q}) \xrightarrow{p} t'}$$

$$\frac{\mathbb{P}_{\ell_0} : t \xrightarrow{p} t'}{\bigoplus_{\ell\in L}\mathbb{P}_\ell : \ell_0{:}t \xrightarrow{\ell_0:p} t'} \qquad \frac{\bigoplus_{\ell\in L}\mathbb{P}_\ell : t \xrightarrow{\ell_0:p} t'}{\mathbb{P}_{\ell_0} : \pi_{\ell_0} t \xrightarrow{p} t'}$$

$$\frac{\mathbb{P}_j[\mu\boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}] : t \xrightarrow{p} t'}{\mu_j\boldsymbol{P}.\ \mathbb{P} : \mathtt{abs}\, t \xrightarrow{\mathtt{abs}\ p} t'} \qquad \frac{\mu_j\boldsymbol{P}.\ \mathbb{P} : t \xrightarrow{\mathtt{abs}\ p} t'}{\mathbb{P}_j[\mu\boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}] : \mathtt{rep}\, t \xrightarrow{p} t'}$$

The following lemma demonstrates that the operational semantics given above interacts well with the type system described above.

Lemma 3. *If* $\mathbb{P} : t \xrightarrow{p} t'$ *then* $\vdash t : \mathbb{P}$ *and there exists a unique* $\mathbb{P}'$ *such that the judgement* $\vdash \mathbb{P} : p : \mathbb{P}'$ *holds; furthermore* $\vdash t' : \mathbb{P}'$.

4.4 Denotational Semantics

Types and Environments A closed type denotes a nominal preorder (an FM-preorder with empty support). We will specify the preorder inductively by rules saying what paths belong to types (judgements $p : \mathbb{P}$) and what the preorder is on them (judgements $p \leq_{\mathbb{P}} p'$). (The method is inspired by [7].) The language of paths is given by

$$p ::= Q \mid Q \mapsto p \mid \ell{:}p \mid \mathtt{abs}\ p \mid \mathtt{new}\, a.\, p,$$

where Q is a set of paths of the form $\langle\{p_1, \ldots, p_n\}\rangle_s$, ℓ is a label and a is a name.

$$\frac{p_1 : \mathbb{P} \quad \ldots \quad p_n : \mathbb{P}}{\langle\{p_1, \ldots, p_n\}\rangle_s : !\mathbb{P}} \qquad \frac{Q : !\mathbb{Q} \quad p : \mathbb{P}}{Q \mapsto p : \mathbb{Q} \to \mathbb{P}}$$

$$\frac{p : \mathbb{P}_{\ell_0}}{\ell_0{:}p : \bigoplus_{\ell\in L}\mathbb{P}_\ell}\ (\ell_0 \in L) \qquad \frac{p : \mathbb{P}_j[\mu\boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}]}{\mathtt{abs}\ p : \mu_j\boldsymbol{P}.\ \mathbb{P}} \qquad \frac{p : \mathbb{P}}{\mathtt{new}\, a.\, p : \delta\mathbb{P}}$$

where the ordering $\leq_{\mathbb{P}}$ of paths of type $\mathbb{P}$ is given recursively as follows.

$$\frac{P \preceq_{\mathbb{P}} P'}{P \leq_{!\mathbb{P}} P'} \qquad \frac{Q' \leq_{!\mathbb{Q}} Q \quad p \leq_{\mathbb{P}} p'}{Q \mapsto p \leq_{\mathbb{Q}\to\mathbb{P}} Q' \mapsto p'}$$

$$\frac{p \leq_{\mathbb{P}_{\ell_0}} p'}{\ell_0{:}p \leq_{\bigoplus_{\ell\in L}\mathbb{P}_\ell} \ell_0{:}p'} \qquad \frac{p \leq_{\mathbb{P}_j[\mu\boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}]} p'}{\mathtt{abs}\ p \leq_{\mu_j\boldsymbol{P}.\ \mathbb{P}} \mathtt{abs}\ p'} \qquad \frac{p \leq_{\mathbb{P}} p'}{\mathtt{new}\, a.\, p \leq_{\delta\mathbb{P}} \mathtt{new}\, a.\, p'}$$

Here, $P \preceq_{\mathbb{P}} P'$ means that for all $p \in P$ there exists $p' \in P'$ such that $p \leq_{\mathbb{P}} p'$. It is straightforward to show that these definitions construct path orders that are nominal preorders and hence objects of $\mathbf{FMPre}_{\varnothing}$. As in HOPLA, in a recursively-defined type $\mu_j\boldsymbol{P}.\ \mathbb{P}$ each path is of the form $\mathtt{abs}\ p$ which means there is an isomorphism $\mathbf{rep} : \mu_j\boldsymbol{P}.\ \mathbb{P} \cong \mathbb{P}_j[\mu\boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}] : \mathbf{abs}$, where $\mathbf{abs}(p) =_{\mathrm{def}} \mathtt{abs}\ p$ and $\mathbf{rep}(\mathtt{abs}\ p) =_{\mathrm{def}} p$.

An environment $\mathtt{x}_1 : \mathbb{P}_1{}^{\#s_1}, \ldots, \mathtt{x}_n : \mathbb{P}_n{}^{\#s_n}$ (with freshness constraints contained in s_0) denotes an object $\mathbb{P}_1^{\#s_1} \,\&\, \cdots \,\&\, \mathbb{P}_n^{\#s_n}$. Notice that such an object is isomorphic to $\widehat{\mathbb{P}_1^{\#s_1}} \times \cdots \times \widehat{\mathbb{P}_n^{\#s_n}}$ via the isomorphisms $\phi^{(s)}$ and m, and it will be convenient to use a 'tuple' notation for environments in the following.

Terms and Actions Typing judgements $\Gamma \vdash_s t : \mathbb{P}$ denote arrows

$$[\![\Gamma \vdash_s t : \mathbb{P}]\!] : [\![\Gamma]\!] \underset{\mathbf{C}}{\to} \mathbb{P}$$

in $\mathbf{FMCts}_s$. The denotation of a typing judgement is built by recursion on the derivation of the typing judgement, making use of the various universal constructions available in FM domains. Typing judgements $\vdash_s \mathbb{P} : p : \mathbb{P}'$ denote arrows

$$[\![\vdash_s \mathbb{P} : p : \mathbb{P}']\!] : \mathbb{P} \underset{\mathbf{C}}{\to} !\mathbb{P}'$$

in $\mathbf{FMCts}_s$ by recursion on the structure of p as shown below. Intuitively the arrow $[\![\vdash_s \mathbb{P} : p : \mathbb{P}']\!]$ matches its input against the action p and returns a collection of possible resumptions after performing p. When the types are clear we abbreviate $[\![\Gamma \vdash_s t : \mathbb{P}]\!]$ and $[\![\vdash_s \mathbb{P} : p : \mathbb{P}']\!]$ to $[\![t]\!]$ and $[\![p]\!]$.

Prefixing and Matching The adjunction $\mathbf{FMLin}(!\mathbb{P}, \mathbb{Q}) \cong \mathbf{FMCts}(\mathbb{P}, \mathbb{Q})$ gives the semantics for an anonymous prefix action, written `!`. The unit η acts as a constructor for this action, taking a term t to the prefixed term `!`t as follows.

Definition 4. *Suppose that* $\Gamma \vdash_s \texttt{!}t : !\mathbb{P}$ *is derived from* $\Gamma \vdash_s t : \mathbb{P}$. *Let* $\gamma \in \widehat{[\![\Gamma]\!]}$ *and* $P \in !\mathbb{P}$. *Then* $P \in [\![\Gamma \vdash_s \texttt{!}t : !\mathbb{P}]\!]\langle\gamma\rangle$ *iff* $P \subseteq [\![\Gamma \vdash_s t : \mathbb{P}]\!]\langle\gamma\rangle$.

The denotation of the judgement $\vdash_\varnothing !\mathbb{P} : \texttt{!} : \mathbb{P}$ is simply the identity map. The counit ϵ acts as a destructor for the `!` action, intuitively 'matching' a `!` action in the output of a term u and passing the resumption after performing the `!` to a term t.

Definition 5. *Suppose that* $\gamma \in \widehat{[\![\Gamma]\!]}$ *and* $\lambda \in \widehat{[\![\Lambda^{\#s'}]\!]}$ *and* $p \in \mathbb{P}$. *Then* $p \in [\![\Gamma, \Lambda^{\#s'} \vdash_s \texttt{[}u \texttt{ > } q\texttt{(x:}\mathbb{Q}'\texttt{ \# } s'\texttt{) => } t\texttt{]} : \mathbb{P}]\!]\langle\gamma, \lambda\rangle$ *iff there exists* $Q \in !\mathbb{Q}'$ *such that* $p \in [\![t]\!]\langle\gamma, Q\rangle$, $Q \in \big([\![q]\!] \circ [\![u]\!]\big)\langle\lambda\rangle$ *and* $Q \# s'$.

Names and Binding The adjunction $(-)^{\#a++} \dashv \delta_a^{++}$ gives rise to the denotational semantics for terms of the form $\texttt{new}\, a\texttt{.}t$ and $t\texttt{[}a\texttt{]}$. Concrete definitions of these operations are given here.

Definition 6. *Suppose* $\Gamma \vdash_s \texttt{new}\, a\texttt{.}t : \delta\mathbb{P}$ *is derived from* $\Gamma^{\#a} \vdash_{s \cup \{a\}} t : \mathbb{P}$ *where* $a \notin s$. *Let* $\gamma \in \widehat{[\![\Gamma]\!]}$, *let* b *be a fresh name and let* $p \in \mathbb{P}$. *Then,* $\texttt{new}\, b.\, p \in [\![\Gamma \vdash_s \texttt{new}\, a\texttt{.}t : \delta\mathbb{P}]\!]\langle\gamma\rangle$ *iff* $(ab) \cdot p \in [\![\Gamma^{\#a} \vdash_{s \cup \{a\}} t : \mathbb{P}]\!]\langle\gamma\rangle$.

Definition 7. *Suppose* $\Gamma^{\#a} \vdash_{s \cup \{a\}} t\texttt{[}a\texttt{]} : \mathbb{P}$ *derives from* $\Gamma \vdash_s t : \delta\mathbb{P}$ *where* $a \notin s$. *Let* $\gamma \in \widehat{[\![\Gamma^{\#a}]\!]}$ *and let* $p \in \mathbb{P}$. *Then,* $\texttt{new}\, a.\, p \in [\![\Gamma \vdash_s t : \delta\mathbb{P}]\!]\langle\gamma\rangle$ *iff* $p \in [\![\Gamma^{\#a} \vdash_{s \cup \{a\}} t\texttt{[}a\texttt{]} : \mathbb{P}]\!]\langle\gamma\rangle$.

The *structural rules* simply adjust the types of the denotations without substantially altering their semantics. They make use of the cartesian structure of each $\mathbf{FMCts}_s$; weakening corresponds to projection, for example. The semantics

of the first new structural rule (fresh-weakening) comes from the obvious inclusion $(-)^{\#a} \Rightarrow (-)$ combined with the isomorphism ϕ, and the second new structural rule (support-weakening) from the inclusion $\mathbf{FMCts}_{s'} \hookrightarrow \mathbf{FMCts}_s$. The denotational semantics of the remaining constructs follows that of HOPLA in [4] very closely. The semantics of *higher-order* processes arises from the cartesian-closed structure of $\mathbf{FMCts}_s$. The semantics of *labelled* processes is based on the biproducts in the linear category; injection into the biproduct corresponds to tagging the outputs of a process with a particular label, and projection to matching against a label. Via cartesian closure a hom-set of $\mathbf{FMCts}_s$ inherits a partial order by inclusion, which in particular has all joins of ω-chains. This provides a standard semantics to *recursion* in the language. The semantics of *nondeterministic sums* is given by union.

Substitution amounts to composition of denotations. However, care must be taken to ensure that all the relevant freshness assumptions are satisfied.

Lemma 4 (Semantic Substitution Lemma). *Suppose that* $\Gamma, \mathtt{y} : \mathbb{R}^{\#r} \vdash_s t : \mathbb{P}$ *and* $\Delta \vdash_{s_1} v : \mathbb{R}$ *where* $s_1 \cap r = \varnothing$ *and the variables in* Γ *are distinct from those in* Δ*. Then*

$$[\![\Gamma, \Delta^{\#r} \vdash_{s \cup s_1} t[v/\mathtt{y}] : \mathbb{P}]\!] = [\![\Gamma, \mathtt{y} : \mathbb{R}^{\#r} \vdash_s t : \mathbb{P}]\!] \circ \big(\mathbf{1}_\Gamma \,\&\, [\![\Delta \vdash_{s_1} v : \mathbb{R}]\!]^{\#r++}\big).$$

5 Soundness and Adequacy

The possibility of observing an action p of a process f is caught by a judgement $\mathbb{P} : t \xrightarrow{p} t'$. In fact the match operator reduces these general observations to observations of just `!` actions, because to observe the action p in the term t is the same as to observe a `!` action in the term `[`t `>` p`(x:`$\mathbb{P}$ `#` s`) => !nil]`.

Lemma 5 (Soundness). *If* $!\mathbb{P} : t \xrightarrow{!} t'$ *and* s *is a finite set of names such that* $\mathrm{supp}(t, t') \subseteq s$ *then* $[\![\vdash_s \, !t' : !\mathbb{P}]\!] \subseteq [\![\vdash_s t : !\mathbb{P}]\!]$.

Define a logical relation $X \trianglelefteq_{\mathbb{P}} t$ between subsets $X \subseteq \mathbb{P}$ and terms such that $\vdash t : \mathbb{P}$ by way of an auxiliary relation $p \in_{\mathbb{P}} t$ between paths $p \in \mathbb{P}$ and terms such that $\vdash t : \mathbb{P}$ as shown in 5. The intuition behind the statement that $p \in_{\mathbb{P}} t$ is that p is a computation path of type $\mathbb{P}$ that the process t may perform. Its definition is by recursion on the structure of paths.

$$\begin{aligned}
X \trianglelefteq_{\mathbb{P}} t &\iff \forall p \in X.\ p \in_{\mathbb{P}} t \\
P \in_{!\mathbb{P}} t &\iff \exists t'.\ !\mathbb{P} : t \xrightarrow{!} t' \text{ and } P \trianglelefteq_{\mathbb{P}} t' \\
Q \mapsto p \in_{\mathbb{Q} \to \mathbb{P}} t &\iff \forall u.\ (Q \trianglelefteq_{\mathbb{Q}} u \Rightarrow p \in_{\mathbb{P}} t\,\mathtt{(}u\mathtt{:}\mathbb{Q}\mathtt{)}) \\
\mathtt{new}\, a.\, p \in_{\delta\mathbb{P}} t &\iff \text{И}\, a.\ p \in_{\mathbb{P}} t\,\mathtt{[}a\mathtt{]} \\
\ell_0 : p \in_{\bigoplus_{\ell \in L} \mathbb{P}_\ell} t &\iff p \in_{\mathbb{P}_{\ell_0}} \pi_{\ell_0} t \\
\mathtt{abs}\, p \in_{\mu_j \boldsymbol{P}.\ \mathbb{P}} t &\iff p \in_{\mathbb{P}_j[\mu \boldsymbol{P}.\ \mathbb{P}/\boldsymbol{P}]} \mathtt{rep}\, t
\end{aligned}$$

This relation can be used to demonstrate that if a path p appears — semantically — in the denotation $[\![t]\!]$ then the term t can — operationally — perform the path p.

Lemma 6. *Suppose* $\Gamma \vdash_s t : \mathbb{P}$ *where* $\Gamma = \mathtt{x}_1 : {\mathbb{P}_1}^{\#s_1}, \ldots, \mathtt{x}_n : {\mathbb{P}_n}^{\#s_n}$. *For each* $i \in \{1, \ldots, n\}$ *let* $\gamma_i \in \widehat{\mathbb{P}_i}^{\#s_i}$ *and let* v_i *be a closed term such that* $\vdash_{s \setminus s_i} v_i : \mathbb{P}_i$ *and* $\gamma_i \trianglelefteq_{\mathbb{P}_i} v_i$. *Then*

$$[\![\Gamma \vdash_s t : \mathbb{P}]\!]\langle \gamma_1, \ldots, \gamma_n \rangle_\Gamma \trianglelefteq_{\mathbb{P}} t[v]$$

where $t[v]$ *is the term obtained by simultaneously substituting each* $\mathtt{x}_i$ *with* v_i.

Lemma 7. *If* $\vdash_s \mathbb{P} : p : \mathbb{P}'$ *and* $X \trianglelefteq_{\mathbb{P}} t$ *and* $P \in [\![p]\!]X$ *then there exists* t' *such that* $\mathbb{P} : t \xrightarrow{p} t'$ *and* $P \trianglelefteq_{\mathbb{P}'} t'$.

We obtain the main theorem of this paper, namely the adequacy of the denotational semantics of Nominal HOPLA with respect to observations of ! actions.

Theorem 1 (Adequacy). $[\![\vdash t : !\mathbb{P}]\!] \neq \varnothing$ *if and only if there exists* t' *such that* $!\mathbb{P} : t \xrightarrow{!} t'$.

Nominal HOPLA subsumes new-HOPLA and inherits its expressiveness. What of full abstraction? Names introduce new subtleties. The obstacle to full abstraction and a tentative proposal to achieve it are described in [5].

References

1. N. Benton, G. Bierman, V. de Paiva, and M. Hyland. Linear lambda-calculus and categorical models revisited. In E. Borgër, G. Jagër, K. H. Bunïng, S. Martini, and M. Richter, (Eds.), *Proceedings of the Sixth Workshop on Computer Science Logic*, pages 61–84. Springer Lecture Notes in Computer Science, vol. 702, 1993.
2. M. J. Gabbay. *A Theory of Inductive Definitions with Alpha-Equivalence*. PhD thesis, Cambridge University, 2001.
3. M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.
4. M. Nygaard and G. Winskel. Domain theory for concurrency. *Theor. Comput. Sci.*, 316(1-3):153–190, 2004.
5. D. C. Turner. *Nominal Domain Theory for Concurrency*. PhD thesis, Cambridge University, Submitted 2008. Submitted version available from `http://www.cl.cam.ac.uk/~dct25/`
6. G. Winskel. Name generation and linearity. In *LICS '05: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, pages 301–310, Washington, DC, USA, 2005. IEEE Computer Society.
7. G. Winskel and K. G. Larsen. Using information systems to solve recursive domain equations effectively. Extended abstract: Springer Lecture Notes in Computer Science, vol. 173. Full version: Technical Report UCAM-CL-TR-51, University of Cambridge, Computer Laboratory, 1984.
8. G. Winskel and F. Zappa Nardelli. new-HOPLA: a higher-order process language with name generation. In *Proc. of 3rd IFIP TCS*, pages 521–534, 2004.