Nominal Domain Theory for Concurrency

David Turner
Computer Laboratory, University of Cambridge

Glynn Winskel
Computer Laboratory, University of Cambridge

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 this expands to Nominal HOPLA, with name generation (closely related to an earlier language new-HOPLA, whose denotational semantics has been problematic). Nominal HOPLA possesses an operational and denotational semantics which are related via soundness and adequacy results, again carried out within FM sets.

1 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 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 can be used to model names in computer science, because commonly the precise nature of a name is unimportant; what matters is its ability to identify and its distinctness from other names. 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 — specifically a domain theory for concurrency.

At heart what makes FM set theory important for treating names are adjunctions associated with new-name abstraction. These are defined fully in appendix A. The simplest and best-known adjunction 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. 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. Often the only change to the theory is to insist that all constructions, particularly those derived from powersets and function spaces, consist only of finitely-supported elements. For us it will mean that a path-based domain theory for concurrency can be *systematically* extended with name generation by working within FM sets.

The idea behind this semantic model is that a process denotes a set of paths in a path order giving the type of computations it can do. Paths 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 π-Calculus, Higher-Order π-Calculus and mobile ambients[7], but giving it 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 far from clear that enough function spaces existed. 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.

We assume some familiarity with nominal set theory and the broader universe of FM sets. See appendix A for a brief introduction. This paper is an abbreviated version of Turner's PhD thesis[5].

Domain theory from path sets To set the scene we give a quick review of the domain theory for processes based on path sets[4]. The objects of the category $\mathbf{Lin}$ are preorders $\mathbb{P}$ 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*, lower (*i.e.* down-closed) sets w.r.t. $\leq_{\mathbb{P}}$, ordered by inclusion. The arrows of $\mathbf{Lin}$, *linear* maps, from $\mathbb{P}$ to $\mathbb{Q}$ are join-preserving functions from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$. The category $\mathbf{Lin}$ is

monoidal-closed with a tensor given the product $\mathbb{P} \times \mathbb{Q}$ of path orders and a corresponding function space by $\mathbb{P}^{op} \times \mathbb{Q}$. $\mathbf{Lin}$ has enough structure to form a model of Girard's classical linear logic [1]. The exponential $!\mathbb{P}$ consists of finite elements of $\widehat{\mathbb{P}}$ under inclusion—$!\mathbb{P}$ can be thought of as consisting of compound paths associated with several runs. Its coKleisli category consists of path orders with continuous functions between the domains of path sets.

2 FM Domain Theory

A similar story unfolds within FM set theory, but it is complicated by a number of factors. Firstly, everything in sight must be finitely-supported: linear maps need only preserve finitely-supported joins, for example. Secondly, if the domain theory is to make use of the new name abstraction of FM sets then the key adjunctions $(-)^{\#a} \dashv \delta_a$ must be woven into the tale. In particular these adjunctions must mesh properly with the central constructions of the domain theory, such as that of taking lower sets of a preorder. Thirdly, and perhaps most surprisingly, the appropriate notion of 'continuous' will turn out to not be simply the preservation of directed joins — even finitely-supported directed joins — but of directed joins with more stringent constraints on supports.

2.1 FM Preorders

Definition 1. *The FM analogue of the concept of a preorder is a **FM-preorder**, which is a pair $\langle \mathbb{P}, \leq_{\mathbb{P}} \rangle$ 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 $\sigma \cdot p \leq_{\sigma \cdot \mathbb{P}} \sigma \cdot p'$.

Definition 2. *The finitely-supported function $f : \mathbb{P} \to \mathbb{Q}$ between FM-preorders is **monotone** if for all $p' \leq_{\mathbb{P}} p \in \mathbb{P}$ it is the case that $fp' \leq_{\mathbb{P}} fp$.*

Definition 3. *The collection of FM-preorders and finitely-supported monotone functions forms a category $\mathbf{FMPre}$. For any finite set of names s let $\mathbf{FMPre}_s$ be the subcategory of $\mathbf{FMPre}$ 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.

Definition 4. *Let $a \notin s$. If $\mathbb{P}$ is an object of $\mathbf{FMPre}_s$ then define*

$$\mathbb{P}^{\#a} =_{\text{def}} \{p \in \mathbb{P} \mid a \# p\}$$

ordered by the restriction of $\leq_{\mathbb{P}}$, and if $f : \mathbb{P} \to \mathbb{Q}$ is an arrow of $\mathbf{FMPre}_s$ then for all $p \in \mathbb{P}^{\#a}$ define

$$f^{\#a}(p) =_{\text{def}} f(p).$$

This is a functor $(-)^{\#a} : \mathbf{FMPre}_s \to \mathbf{FMPre}_{s \cup \{a\}}$ which has a right adjoint δ_a as follows.

Definition 5. *Let $a \notin s$. If $\mathbb{P}$ is an object of $\mathbf{FMPre}_{s \cup \{a\}}$ then define the α-equivalence relation $\sim_\alpha$ on $\mathbb{P} \times \mathbb{A}$ by having $\langle p_1, a_1 \rangle \sim_\alpha \langle p_2, a_2 \rangle$ iff $\mathsf{N}\, b.\, (a_1 b) \cdot p_1 = (a_2 b) \cdot p_2$ and define the FM-preorder*

$$\delta_a \mathbb{P} =_{\text{def}} \{p' \in (\mathbb{P} \times \mathbb{A})/{\sim_\alpha} \mid \mathsf{N}\, b.\, p'@b \in (ab) \cdot \mathbb{P}\},$$

where if p'_1 and p'_2 are elements of $\delta_a \mathbb{P}$ then

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

If $f : \mathbb{P} \to \mathbb{Q}$ is a finitely-supported monotone function then for all $p' \in \delta_a \mathbb{P}$ define

$$\delta_a f(p') =_{\text{def}} \mathbf{fresh}\, b\, \mathbf{in}\, [b].\big((ab) \cdot f\big)(p'@b).$$

The unit ξ and counit ζ of the adjunction $(-)^{\#a} \dashv \delta_a$ are

$$\xi_{\mathbb{P}}(p) =_{\text{def}} \mathbf{fresh}\, b\, \mathbf{in}\, [b].p \quad \text{and} \quad \zeta_{\mathbb{P}}(p') =_{\text{def}} p'@a.$$

3 Nondeterministic Domains in FM

A process is to denote the collection of computation paths that it may perform, with the added constraint that each process can only access finitely many names — the denotations of processes must be finitely supported. Apart from this additional constraint the development of categories of FM-linear maps follows closely that of the category $\mathbf{Lin}$ of path orders and linear maps.

If $\mathbb{P}$ is any FM-preorder define

$$\widehat{\mathbb{P}} =_{\text{def}} \{x_\downarrow \mid x \subseteq_{\text{fs}} \mathbb{P}\},$$

ordered by inclusion, where $x_\downarrow$ is the lower set generated by x. Such a poset is an FM-complete join-semilattice in the sense that every finitely-supported subset of $\widehat{\mathbb{P}}$ has a join in $\widehat{\mathbb{P}}$ given by its union. The order $\widehat{\mathbb{P}}$ contains elements of the form $\{p\}_\downarrow$ for each $p \in \mathbb{P}$ which comprise its (completely) prime elements. (The order $\widehat{\mathbb{P}}$ is prime algebraic in FM set theory.) Finally, $\widehat{\mathbb{P}}$, with $\{\cdot\}_\downarrow : \mathbb{P} \to \widehat{\mathbb{P}}$, can be characterised abstractly as the free finitely-supported join completion of $\mathbb{P}$. It follows that finitely-supported monotone maps $\mathbb{P} \to \widehat{\mathbb{Q}}$ are in bijective correspondence with finitely-supported finitely-supported-join-preserving maps $\widehat{\mathbb{P}} \to \widehat{\mathbb{Q}}$. Such maps are 'FM-linear' or simply 'linear'. In fact, this bijective correspondence is natural in $\mathbb{P}$ and $\mathbb{Q}$ and therefore an adjunction

$$\mathbf{FMPre}(\mathbb{P}, \widehat{\mathbb{Q}}) \cong \mathbf{FMLin}(\mathbb{P}, \mathbb{Q})$$

where $\mathbf{FMLin}$ is the category with the same objects as $\mathbf{FMPre}$ and whose arrows are FM-linear maps.

3.1 The Structure of FMLin

Like **Lin** the category **FMLin** has enough structure to form a model of classical linear logic[1].

Under the correspondence

$$\mathbf{FMLin}(\mathbb{P},\mathbb{Q}) \cong \widehat{\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}}$$

the ordering on $\widehat{\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}}$ gives rise to an ordering (by pointwise inclusion) on $\mathbf{FMLin}(\mathbb{P},\mathbb{Q})$, and joins in $\widehat{\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}}$ give rise to joins (by pointwise union) in $\mathbf{FMLin}(\mathbb{P},\mathbb{Q})$.

Since left adjoints preserve coproducts, the disjoint union of the orders $\mathbb{P}_1$ and $\mathbb{P}_2$ forms the binary coproduct $\mathbb{P}_1 + \mathbb{P}_2$ in **FMLin**. Furthermore this coproduct is also a binary product $\mathbb{P}_1 \,\&\, \mathbb{P}_2$: the projections are defined by $\mathbf{out}_1 =_{\mathrm{def}} [\mathbf{1}_{\mathbb{P}_1}, \varnothing]$ and $\mathbf{out}_2 =_{\mathrm{def}} [\varnothing, \mathbf{1}_{\mathbb{P}_2}]$. The object $\mathbb{P}_1 + \mathbb{P}_2$ is a binary biproduct. More generally, if $(\mathbb{P}_\ell)_{\ell \in L}$ is any collection of FM-preorders where the mapping $\ell \mapsto \mathbb{P}_\ell$ is supported by s then the object $\bigoplus_{\ell \in L} \mathbb{P}_\ell$, defined as the disjoint union of the $\mathbb{P}_\ell$, is a biproduct, supported by s.

A tensor product on **FMLin** can be defined as the product $\mathbb{P}_1 \times \mathbb{P}_2$ of the underlying preorders. As $\mathbf{FMLin}(\mathbb{P} \times \mathbb{Q}, \mathbb{R}) \cong \mathbf{FMLin}(\mathbb{P}, \mathbb{Q}^{\mathrm{op}} \times \mathbb{R})$, naturally in $\mathbb{P}$ and $\mathbb{R}$, we have that **FMLin** is closed with respect to the $\times$ tensor.

3.2 Name Generation in FMLin

FMLin inherits name generation from **FMPre**. Let $s \subseteq_{\mathrm{fin}} \mathbb{A}$ 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:

$$\begin{array}{rcl} \phi_{\mathbb{P}}(x) & =_{\mathrm{def}} & \{p \in x \mid a \# p\} \\ \text{and } \phi_{\mathbb{P}}^{-1}(x) & =_{\mathrm{def}} & x \cup \bigcup_{b \# x, \mathbb{P}} (ab) \cdot x \end{array}$$

$$\begin{array}{rcl} \theta_{\mathbb{Q}}(y') & =_{\mathrm{def}} & \{q' \mid \text{Ⱨ}\, b.\, q'@b \in y'@b\} \\ \text{and } \theta_{\mathbb{Q}}^{-1}(y) & =_{\mathrm{def}} & \mathbf{fresh}\, b\, \mathbf{in}\, [b].\{q \mid [b].q \in y\}. \end{array}$$

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

$$\widehat{\mathbb{P}^{\#a}} \xrightarrow{\phi_{\mathbb{P}}^{-1}} \widehat{\mathbb{P}}^{\#a} \xrightarrow{f^{\#a}} \widehat{\mathbb{Q}}^{\#a} \xrightarrow{\phi_{\mathbb{Q}}} \widehat{\mathbb{Q}^{\#a}}\ .$$

Using θ define $\delta_a^+ : \mathbf{FMLin}_{s \cup \{a\}} \to \mathbf{FMLin}_s$ similarly.

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

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

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

4 Continuity in FM Domains

Linear maps are too restrictive to give a semantics for concurrent processes. With HOPLA[4] the solution was to turn from linear to continuous maps, which preserve only directed joins. But this is not appropriate in the nominal setting: the desired semantics for name generation is not directed-join continuous.

4.1 Continuity and name generation

To see this, we consider a term construction $\mathtt{new}\, a\, \mathtt{.}\, t$ inspired by new-HOPLA[7]. 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 $\mathtt{new}\, a\, \mathtt{.}\, t$ denotes an element of $\widehat{\delta\mathbb{A}}$; its denotation $[\![\mathtt{new}\, a\, \mathtt{.}\, t]\!]$ 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 $\mathtt{new}\, a\, \mathtt{.}\, t$ denotes a process with actions of the form $[b].c$ and $[c].c$ from $\delta_a \mathbb{A}$.

Consider now an open term $\mathtt{new}\, a\, \mathtt{.}\, (-)$. Substitution into $\mathtt{new}\, a\, \mathtt{.}\, (-)$ 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 $\theta_A([a].A)$ which can be shown to contain $[a].a$. However, the substitution of $s \subseteq_{\mathrm{fin}} \mathbb{A}$ results in denotation $\theta_A([a'].s)$, 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 $\mathtt{new}\, a\, \mathtt{.}\, (-)$ does not yield a directed-join continuous function.

4.2 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 6. *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 preserve joins of FM-directed sets.*

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 $\mathtt{new}\, a\,.(-)$, first 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 maximal element x. So $\mathtt{new}\, a\,.(-)$ is FM-continuous.

4.3 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 7. *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. To see this, let $x \in \widehat{\mathbb{A}}$ be such that $x \subseteq \bigcup X$ where $X \subseteq \widehat{\mathbb{A}}$ is directed and uniformly supported by s, then every element of X is either a subset of s or a superset of $\mathbb{A} \setminus s$. Therefore X is finite, and since it is also directed it contains a maximal element x' and hence $x \subseteq x'$. More generally,

Definition 8. *If $\mathbb{P}$ is a FM-preorder, F a finite subset of $\mathbb{P}$ and s a finite set of names that contains* $\mathrm{supp}(\mathbb{P})$ *then define*

$$\langle F \rangle_s =_{\mathrm{def}} \bigcup_{\sigma \# s} \sigma \cdot F.$$

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 fact, the elements of the form $\langle F \rangle_s$ are precisely the isolated elements of $\widehat{\mathbb{P}}$:

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 $P = \langle F \rangle_{s\downarrow}$.*

4.4 The Category FMCts

Let $\mathbf{FMCts}$ be the category with objects FM-preorders and arrows from $\mathbb{P}$ to $\mathbb{Q}$ are FM-continuous functions from $\widehat{\mathbb{P}}$ to $\widehat{\mathbb{Q}}$. Note $\mathbf{FMLin}$ is a subcategory of $\mathbf{FMCts}$.

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. 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 $\forall p \in P \exists p' \in P'.\, p \leq_{\mathbb{P}} p'$. Write $i_{\mathbb{P}}$ for the map $i_{\mathbb{P}} : !\mathbb{P} \to \widehat{\mathbb{P}}$ given by $i_{\mathbb{P}} P =_{\mathrm{def}} P_{\downarrow}$.

Each $\widehat{\mathbb{P}}$, with $i_{\mathbb{P}}$, 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 correspondence

$$\mathbf{FMCts}(\mathbb{P}, \mathbb{Q}) \cong \widehat{!\mathbb{P}^{\mathrm{op}} \times \mathbb{Q}}$$

enriches hom-sets in $\mathbf{FMCts}$ with a partial order structure given by pointwise inclusion, and joins given by pointwise union. Generally, if $(f_i)_{i \in I}$ is a collection of continuous maps $\mathbb{P} \underset{\mathbf{C}}{\to} \mathbb{Q}$ where the mapping $i \mapsto f_i$ is supported by s then their pointwise union $\sum_{i \in I} f_i$ is an FM-continuous map supported by s.

Since right adjoints preserve products, $\mathbf{FMCts}$ has finite products given by the disjoint union of the underlying preorders as in $\mathbf{FMLin}$. The product of the objects $\mathbb{P}_1$ and $\mathbb{P}_2$ is written as $\mathbb{P}_1 \,\&\, \mathbb{P}_2$. If $\mathbb{P}_1$ and $\mathbb{P}_2$ are objects of $\mathbf{FMPre}$ then there exists an isomorphism $m_{\mathbb{P}_1, \mathbb{P}_2} : \widehat{\mathbb{P}_1} \times \widehat{\mathbb{P}_2} \cong \widehat{\mathbb{P}_1 \,\&\, \mathbb{P}_2}$ where if $x_i \in \widehat{\mathbb{P}_i}$ for $i \in \{1, 2\}$ then

$$m_{\mathbb{P}_1, \mathbb{P}_2} \langle x_1, x_2 \rangle =_{\mathrm{def}} x_1 \uplus x_2.$$

The general biproduct $\bigoplus_{\ell \in L} \mathbb{P}_\ell$ of $\mathbf{FMLin}$ becomes just a product in $\mathbf{FMCts}$.

If $\mathbb{P}$ and $\mathbb{Q}$ are objects of $\mathbf{FMCts}$ then there is an isomorphism $m^{!}_{\mathbb{P}, \mathbb{Q}} : !\mathbb{P} \times !\mathbb{Q} \cong !(\mathbb{P} \,\&\, \mathbb{Q})$ which maps a pair $\langle P, Q \rangle$ to the union $P \uplus Q$. Because $\mathbf{FMLin}$ is monoidal closed, so that each $(-) \times \mathbb{Q}$ has a right adjoint $\mathbb{Q} \multimap (-)$, with the natural isomorphism $m^{!}$ we derive that $\mathbb{Q} \to (-) =_{\mathrm{def}} !\mathbb{Q} \multimap (-)$ is right adjoint to $(-) \,\&\, \mathbb{Q}$, so that $\mathbf{FMCts}$ is cartesian closed.

There is a map $n_{\mathbb{P}, \mathbb{Q}} : \widehat{\mathbb{P}} \times \widehat{\mathbb{Q}} \to \widehat{\mathbb{P} \times \mathbb{Q}}$ defined by

$$n_{\mathbb{P}, \mathbb{Q}} \langle x, y \rangle =_{\mathrm{def}} \{\langle p, q \rangle \in \mathbb{P} \times \mathbb{Q} \mid p \in x \text{ and } q \in y\}.$$

The composition $\widehat{m^{!}_{\mathbb{P}, \mathbb{Q}}} \circ n_{!\mathbb{P}, !\mathbb{Q}} \circ m^{-1}_{!\mathbb{P}, !\mathbb{Q}} \circ (\eta_{\mathbb{P}} \,\&\, \mathbf{1}_{!\mathbb{Q}})$ defines a natural strength map $S_{\mathbb{P}, \mathbb{Q}}$ for the monad $(!, \eta, \epsilon_!)$ on $\mathbf{FMCts}$. Concretely, if $x \in \widehat{\mathbb{P}}$ and $Y \in \widehat{!\mathbb{Q}}$ then

$$S_{\mathbb{P}, \mathbb{Q}}(x \uplus Y) = \{P \uplus Q \mid P \subseteq x, P \in !\mathbb{P} \text{ and } Q \in Y\}.$$

4.5 Name Generation in FMCts

We inherit adjunctions

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

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 $\mathbf{FMLin}_s$ is the subcategory of **FMLin** whose objects and arrows are all 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 $\mathbf{FMCts}_{s \dot\cup \{a\}}$ is taken to $\delta_a^{++} g =_{\mathrm{def}} \theta_{\mathbb{Q}} \circ \delta_a g \circ \theta_{\mathbb{P}}^{-1}$. 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 3.2, we obtain as a composite the bijection

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

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 $x \in \widehat{\mathbb{P}^{\#s}}$.

5 Nominal HOPLA

Nominal HOPLA is an expressive calculus for higher-order processes with nondeterminism and name-binding. Its development follows closely that of HOPLA (a Higher-Order Process LAnguage)[4] and is inspired by the language new-HOPLA[7].

In order to present Nominal HOPLA it is necessary to give the language an abstract syntax, and this syntax includes some binding operators such as the usual function abstraction $\lambda \mathtt{x}.t$ which binds free occurrences of the variable $\mathtt{x}$ in the term t. However, the structure of interest here is not the syntax of Nominal HOPLA but its semantics, and the binding of variables in its syntax is a distraction. To avoid confusion the binding of variables is treated in the usual informal fashion: bound variables are always distinct from the other variables in scope, and substitution silently avoids capturing free variables. In particular, if $\mathtt{x}$ is a variable and σ is a permutation of the set of atoms then $\sigma \cdot \mathtt{x} = \mathtt{x}$.

5.1 Syntax

Fix a set of term variables $\mathtt{x}, \mathtt{y}, \ldots$ and a set of type variables $P, \ldots$, each with a discrete permutation action. Also fix a set $\mathcal{L}$ of nominal label-sets, also with the discrete permutation action. Labels are written $\ell, \ell_0, \ldots \in L \in \mathcal{L}$.

5.1.1 Syntax of Types

Types are given by the grammar

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

where P is a type variable, $\vec{P}$ is a list of type variables, and $\mu_j \vec{P}.\, \vec{\mathbb{P}}$ binds $\vec{P}$. A closed type is a type with no free variables, and in the following, closed types are normally simply called 'types'. The permutation action on types is the discrete action.

5.1.2 Syntax of Environments

Environments are given by the grammar

$$\Gamma ::= () \mid \Gamma, \mathtt{x} : \mathbb{P}^{\#s}$$

where $\mathtt{x}$ ranges over variables, $\mathbb{P}$ ranges over types and s ranges over finite sets of names, and the variables in Γ are distinct from $\mathtt{x}$. The intended meaning of $\mathtt{x} : \mathbb{P}^{\#s}$ is that the variable $\mathtt{x}$ takes values of type $\mathbb{P}$ that are assumed to be fresh for s. The set of environments may be equipped with a permutation action which simply permutes the freshness assumptions.

5.1.3 Syntax of Terms

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 (see 5.1.4), ℓ over labels and $\mathbb{P}$ over types.

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

The forms $\mathtt{rec\,x.}\,t$, $\mathtt{[}u \mathtt{>} p\mathtt{(x:}\mathbb{P} \mathtt{\#}\, s\mathtt{)\ =>}\ t\mathtt{]}$ 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 $\mathtt{new}\,a\mathtt{.}\,t$ binds the name a in t.

The nondeterministic sum may be over an infinite set I, but there are constraints to ensure that it behaves properly:

the mapping $i \mapsto t_i$ is a finitely supported function from a nominal set I to the set of terms, and is such that there exists a finite set X of variables such that for all i the free variables of t_i are contained in X. Write `nil` for the inactive term $\sum_{i \in \varnothing} t_i$.

5.1.4 Syntax of Actions

The operational semantics of Nominal HOPLA, as defined in section 5.4, is given in the style of a labelled transition system. 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 the same way that a is bound in the term $\texttt{new}\, a\texttt{.}\, t$.

Actions and terms form nominal sets where the permutation action is given by the obvious structural recursion.

5.1.5 Substitution

The substitution $t[v/\texttt{y}]$ of a term v for the variable `y` in a term t is defined by recursion on t in the usual fashion. Substitution is capture-avoiding in both names and variables, in the sense that for substitution into a term of the forms

$$\texttt{rec x.}\, t \qquad \texttt{[}\, u \texttt{ > } p\texttt{(x:}\mathbb{P}\texttt{ \# }s\texttt{) => }\, t\texttt{]} \qquad \lambda\texttt{x.}\, t$$

the variable `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.

5.2 Typing Rules

5.2.1 Typing Rules for Terms

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.

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

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

Weakening. The environment may be extended with extra variables.

$$\frac{\Gamma \vdash_s t : \mathbb{P}}{\Gamma, \texttt{x} : \mathbb{Q}^{\#\varnothing} \vdash_s t : \mathbb{P}}$$

Exchange. Two variables in the environment may be exchanged.

$$\frac{\Gamma, \texttt{x2} : \mathbb{Q}_2{}^{\#s_2}, \texttt{x1} : \mathbb{Q}_1{}^{\#s_1}, \Lambda \vdash_s t : \mathbb{P}}{\Gamma, \texttt{x1} : \mathbb{Q}_1{}^{\#s_1}, \texttt{x2} : \mathbb{Q}_2{}^{\#s_2}, \Lambda \vdash_s t : \mathbb{P}}$$

Contraction. It is possible to replace a pair of variables (with equal types) with a single variable.

$$\frac{\Gamma, \texttt{x1} : \mathbb{Q}^{\#s'}, \texttt{x2} : \mathbb{Q}^{\#s'} \vdash_s t : \mathbb{P}}{\Gamma, \texttt{x1} : \mathbb{Q}^{\#s'} \vdash_s t[\texttt{x1/x2}] : \mathbb{P}}$$

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 (Terms). It is possible to extend the 'current' set s of names.

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

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

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

Match. A term of the form $\texttt{[}\, u \texttt{ > } q\texttt{(x:}\mathbb{Q}'\texttt{ \# }s'\texttt{) => }\, t\texttt{]}$ 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.

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

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

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

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 $\lambda\texttt{x.}\, t$ of type $\mathbb{Q} \to \mathbb{P}$ that can in turn be applied to a term of type $\mathbb{Q}$ in the usual fashion.

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

Labelling and Label Projection. The actions of a term t may be 'tagged' with a label ℓ_0 by forming the term $\ell_0 \mathbf{:} t$. 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.

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

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

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

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 $\mathtt{new}\, a \mathtt{.} t$ can perform the same actions as t with the name a bound, whereas the term $t\mathtt{[}a\mathtt{]}$ 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.

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

5.2.2 Typing Rules 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 this means that p is an action that terms of type $\mathbb{P}$ may perform and the resumption is of type $\mathbb{P}'$.

$$\frac{\vdash_{s'} \mathbb{P} : p : \mathbb{P}'}{\vdash_s \mathbb{P} : p : \mathbb{P}'}\ (s' \subseteq s) \qquad \frac{-}{\vdash_\varnothing !\mathbb{P} : \mathtt{!} : \mathbb{P}}$$

$$\frac{\vdash_s \mathbb{P} : p : \mathbb{P}' \quad \vdash_s u : \mathbb{Q}}{\vdash_s \mathbb{Q} \to \mathbb{P} : u \mapsto p : \mathbb{P}'} \qquad \frac{\vdash_s \mathbb{P}_{\ell_0} : p : \mathbb{P}'}{\vdash_s \bigoplus_{\ell \in L} \mathbb{P}_\ell : \ell_0 \mathbf{:} p : \mathbb{P}'}$$

$$\frac{\vdash_s \mathbb{P}_j[\mu \vec{P}.\ \vec{\mathbb{P}}/\vec{P}] : p : \mathbb{P}'}{\vdash_s \mu_j \vec{P}.\ \vec{\mathbb{P}} : \mathtt{abs}\ p : \mathbb{P}'} \qquad \frac{\vdash_{s \cup \{a\}} \mathbb{P} : p : \mathbb{P}'}{\vdash_s \delta\mathbb{P} : \mathtt{new}\, a.\, p : \delta\mathbb{P}'}$$

5.3 The Substitution Lemma

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 $\Delta \vdash_{s_1} v : \mathbb{P}$ where $s_1 \cap r = \varnothing$ and the variables in Γ are distinct from those in Δ. Then*

$$\Gamma, \Delta^{\#r} \vdash_{s \cup s_1} t[v/\mathtt{y}] : \mathbb{P}$$

5.4 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.

$$\frac{\mathbb{P} : t[\mathtt{rec\ x.}t/\mathtt{x}] \xrightarrow{\ p\ } t'}{\mathbb{P} : \mathtt{rec\ x.}t \xrightarrow{\ p\ } t'} \qquad \frac{-}{!\mathbb{P} : \mathtt{!}t \xrightarrow{\ \mathtt{!}\ } t}$$

$$\frac{\mathbb{P} : t[u'/\mathtt{x}] \xrightarrow{\ p\ } t' \quad \mathbb{Q} : u \xrightarrow{\ q\ } u' \quad \vdash \mathbb{Q} : q : \mathbb{Q}'}{\mathbb{P} : \mathtt{[}\, u \mathtt{>}\, q\mathtt{(x:}\mathbb{Q}'\ \mathtt{\#}\ s'\mathtt{)\ =>}\ t\mathtt{]} \xrightarrow{\ p\ } t'}$$

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

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

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

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

$$\frac{\mathbb{P} : t_{i_0} \xrightarrow{\ p\ } t'}{\mathbb{P} : \sum_{i \in I} t_i \xrightarrow{\ p\ } t'}$$

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 $\vdash t' : \mathbb{P}'$.*

5.5 Denotational Semantics

The denotational semantics of Nominal HOPLA arises directly from various universal constructions in $\mathbf{FMCts}$ discussed above. The design of the name-free process calculus HOPLA[4] was also guided by the principle of universal constructions, and Nominal HOPLA can be seen as a straightforward extension of HOPLA with terms of the form $\mathtt{new}\, a\,\mathtt{.}\, t$ and $t\,\mathtt{[}\,a\,\mathtt{]}$ which arise directly from the adjunction $(-)^{\#a++} \dashv \delta_a^{++}$.

5.5.1 Types and Environments

A closed type denotes the collection of paths of the appropriate type, ordered by extension. Such *path orders*, even recursively-defined ones, can be constructed inductively out of syntactic tokens by a method inspired by the use of information systems to solve recursive domain equations[6] as demonstrated here.

The denotation of the type $\mathbb{P}$ is given in terms of a language of paths, given by the grammar

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

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. Paths are typed by judgements of the form $p : \mathbb{P}$ according to the following rules.

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

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

$$\frac{p : \mathbb{P}}{\mathtt{new}\, a.\, p : \delta\mathbb{P}}$$

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

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

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

$$\frac{p \leq_{\mathbb{P}} p'}{\mathtt{new}\, a.\, p \leq_{\delta\mathbb{P}} \mathtt{new}\, a.\, p'}$$

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 $\mu_j\vec{P}.\,\vec{\mathbb{P}}$ each path is of the form $\mathtt{abs}\; p$ which means there is an isomorphism

$$\mathbf{rep} : \mu_j\vec{P}.\,\vec{\mathbb{P}} \cong \mathbb{P}_j[\mu\vec{P}.\,\vec{\mathbb{P}}/\vec{P}] : \mathbf{abs},$$

where $\mathbf{abs}(p) =_{\mathrm{def}} \mathtt{abs}\; p$ and $\mathbf{rep}(\mathtt{abs}\; p) =_{\mathrm{def}} p$.

Environments Γ (with freshness constraints contained in s_0) denote objects of $\mathbf{FMCts}_{s_0}$ by setting $[\![()]\!] =_{\mathrm{def}} \mathbb{O}$, the empty preorder, and $[\![\Gamma, \mathbf{x} : \mathbb{P}^{\#s}]\!] =_{\mathrm{def}} [\![\Gamma]\!] \,\&\, \mathbb{P}^{\#s}$. Notice that $\widehat{[\![\Gamma, \mathbf{x} : \mathbb{P}^{\#s}]\!]} \cong \widehat{[\![\Gamma]\!]} \times \widehat{\mathbb{P}}^{\#s}$ via the isomorphisms $\phi^{(s)}$ and m, so it is convenient to use a 'tuple' notation $\langle\gamma, p\rangle$ for elements of $\widehat{[\![\Gamma, \mathbf{x} : \mathbb{P}^{\#s}]\!]}$ in the following.

5.5.2 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 $\mathbf{FMCts}$. Intuitively, the arrow $[\![\Gamma \vdash_s t : \mathbb{P}]\!]$ receives some input in its free variables, as typed by Γ, and returns the set of paths that the term t can perform with the given input.

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.

If the type information is clear then $[\![\Gamma \vdash_s t : \mathbb{P}]\!]$ and $[\![\, \vdash_s \mathbb{P} : p : \mathbb{P}']\!]$ are abbreviated to $[\![t]\!]$ and $[\![p]\!]$ respectively.

Higher-Order Processes The cartesian-closed structure of $\mathbf{FMPre}_s$ gives rise to a semantics for higher-order processes as for the simply-typed λ-calculus. In slightly greater detail, abstraction of a variable of type $\mathbb{P}$ is simply given by transposition in the exponential adjunction $(-) \,\&\, \mathbb{P} \dashv \mathbb{P} \to (-)$, and function application is given by the counit of this adjunction.

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

Definition 9. *Suppose that* $\Gamma \vdash_s \mathtt{!}t : !\mathbb{P}$ *is derived from* $\Gamma \vdash_s t : \mathbb{P}$. *Let* $\gamma \in \widehat{[\![\Gamma]\!]}$ *and* $P \in !\mathbb{P}$. *Then*

$$P \in [\![\Gamma \vdash_s \mathtt{!}t : !\mathbb{P}]\!]\langle\gamma\rangle \text{ iff } P \subseteq [\![\Gamma \vdash_s t : \mathbb{P}]\!]\langle\gamma\rangle.$$

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

Definition 10. *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* $Q \# s'$.

Labelled Processes Biproducts $\bigoplus_{\ell\in L}\mathbb{P}_\ell$ give rise to a denotational semantics for labelling in exactly the same way as it did for HOPLA. Injection into the biproduct corresponds to tagging the outputs of a process with a particular label, and projection corresponds to matching against a particular label.

Recursion The hom-sets of $\mathbf{FMCts}_s$ are posets which have all joins of ω-chains. Let $f : \mathbb{Q} \,\&\, \mathbb{P} \underset{\mathrm{C}}{\to} \mathbb{P}$ be an arrow of $\mathbf{FMCts}_s$ and consider the continuous operation on hom-sets $f^* : \mathbf{FMCts}_s(\mathbb{Q},\mathbb{P}) \to \mathbf{FMCts}_s(\mathbb{Q},\mathbb{P})$ given by $f^*(g) = f \circ (\mathbf{1}_\mathbb{Q} \,\&\, g) \circ \Delta_\mathbb{Q}$. By Kleene's fixpoint theorem it has a least fixed point $\mathrm{fix}(f^*)$ defined as follows. Let $g_0 = \varnothing \in \mathbf{FMCts}_s([\![\Gamma]\!],\mathbb{P})$ and for each $n \in \omega$ define $g_{n+1} = f \circ (\mathbf{1}_\Gamma \,\&\, g_n) \circ \Delta_{[\![\Gamma]\!]}$, then $\mathrm{fix}(f^*)\langle\gamma\rangle = \bigcup_{n\in\omega} g_n\langle\gamma\rangle$. This fixed point gives a semantics for terms of the form $\texttt{rec x.}t$ as follows.

Definition 11. *If the judgement* $\Gamma \vdash_s \texttt{rec x.}t : \mathbb{P}$ *is derived from* $\Gamma, \mathtt{x} : \mathbb{P}^{\#\varnothing} \vdash_s t : \mathbb{P}$ *then define*

$$[\![\Gamma \vdash_s \texttt{rec x.}t : \mathbb{P}]\!] =_{\mathrm{def}} \mathrm{fix}([\![\Gamma, \mathtt{x} : \mathbb{P}^{\#\varnothing} \vdash_s t : \mathbb{P}]\!]^*).$$

Recursively-defined processes need recursively-defined types. The isomorphism $\mathbf{rep} : \mu_j\vec{P}.\,\vec{\mathbb{P}} \cong \mathbb{P}_j[\mu\vec{P}.\,\vec{\mathbb{P}}/\vec{P}] : \mathbf{abs}$, relating a recursive type to its unfolding, gives rise to the denotational semantics for terms of recursive types just as it does for HOPLA.

Nondeterminism Each hom-set $\mathbf{FMCts}_s([\![\Gamma]\!],\mathbb{P})$ can be seen as a subset of the complete FM-preorder $\widehat{![\![\Gamma]\!]^{\mathrm{op}} \times \mathbb{P}}$. Therefore, given a collection of arrows $f_i : [\![\Gamma]\!] \underset{\mathrm{C}}{\to} \mathbb{P}$ where the mapping $i \mapsto f_i$ is supported by s, the join $\sum_{i\in I} f_i$ of the f_i (in $\widehat{![\![\Gamma]\!]^{\mathrm{op}} \times \mathbb{P}}$) is an arrow of $\mathbf{FMCts}_s$. This join is given by pointwise union and can be used to give a denotational semantics to nondeterministic sums as follows.

Definition 12. *Suppose that* $\Gamma \vdash_s \sum_{i\in I} t_i : \mathbb{P}$ *is derived from the collection of judgements* $\Gamma \vdash_{s_i} t_i : \mathbb{P}$ *for each* $i \in I$. *Let* $\gamma \in \widehat{[\![\Gamma]\!]}$. *Then*

$$[\![\Gamma \vdash_s \textstyle\sum_{i\in I} t_i : \mathbb{P}]\!]\langle\gamma\rangle = \bigcup_{i\in I}\big([\![\Gamma \vdash_{s_i} t_i : \mathbb{P}]\!]\langle\gamma\rangle\big).$$

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 $t\texttt{[}a\texttt{]}$: $[\![\texttt{new}\,a\texttt{.}t]\!]$ is given by one transposition of $[\![t]\!]$ in the adjunction, whereas $[\![t\texttt{[}a\texttt{]}]\!]$ arises from the other transposition. Concrete descriptions of these operations are given here.

Definition 13. *Suppose* $\Gamma \vdash_s \texttt{new}\,a\texttt{.}t : \delta\mathbb{P}$ *is derived from* $\Gamma^{\#a} \vdash_{s\dot\cup\{a\}} t : \mathbb{P}$ *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* $(ab)\cdot p \in [\![\Gamma^{\#a} \vdash_{s\dot\cup\{a\}} t : \mathbb{P}]\!]\langle\gamma\rangle$.

Definition 14. *Suppose* $\Gamma^{\#a} \vdash_{s\dot\cup\{a\}} t\texttt{[}a\texttt{]} : \mathbb{P}$ *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* $p \in [\![\Gamma^{\#a} \vdash_{s\dot\cup\{a\}} t\texttt{[}a\texttt{]} : \mathbb{P}]\!]\langle\gamma\rangle$.

Structural Rules The denotational semantics associated with the usual structural rules 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) gets its semantics from the inclusion $\mathbf{FMCts}_{s'} \hookrightarrow \mathbf{FMCts}_s$. These rules simply adjust the types of the denotations without substantially altering their semantics.

5.5.3 Substitution as Composition

Substitution effectively amounts to composition of denotations, as the following lemma shows. 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*

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

6 Soundness and Adequacy

The operational semantics gives rise to a notion of observation that can be made about a process: it is possible to observe an action $\vdash \mathbb{P} : p : \mathbb{P}'$ by deriving a judgement of the form $\mathbb{P} : t \xrightarrow{p} t'$. 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 $\texttt{[}t \texttt{ > } p\texttt{(x:}\mathbb{P}\texttt{ \# } s\texttt{) => !nil]}$.

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*

$$[\![\vdash_s \texttt{!}t' : !\mathbb{P}]\!] \subseteq [\![\vdash_s t : !\mathbb{P}]\!].$$

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 6. 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.

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

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* $P \trianglelefteq_{\mathbb{P}'} t'$.

It is now possible to show the main theorem of this paper, namely the adequacy of the given 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* $!\mathbb{P} : t \xrightarrow{\texttt{!}} t'$.

Obstacles 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, editors, *Proceedings of the Sixth Workshop on Computer Science Logic*, pages 61–84. Springer Verlag, 1993.

[2] M. J. Gabbay. *A Theory of Inductive Definitions with Alpha-Equivalence*. PhD thesis, Cambridge University, 2001. Supervised by Andrew Pitts.

[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, 2008. Supervised by Glynn Winskel.

[6] G. Winskel and K. G. Larsen. Using information systems to solve recursive domain equations effectively. Technical Report UCAM-CL-TR-51, University of Cambridge, Computer Laboratory, 1984.

[7] 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.

A Nominal and FM Set Theory

This appendix comprises a brief introduction to the theories of nominal and Fraenkel-Mostowski (FM) sets. More comprehensive expositions are available elsewhere[2, 3].

A.1 Nominal Set Theory

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, written $\mathcal{G}$.

A $\mathcal{G}$-set is a set X together with a $\mathcal{G}$ action, written as an infix $\cdot_X$ or simply $\cdot$, on X. A set $s \subseteq \mathbb{A}$ *supports* the element x of the $\mathcal{G}$-set X if for any $\sigma \in \mathcal{G}$ such that $\sigma a = a$ for all $a \in s$ it is also the case that $\sigma \cdot x = x$. If $x \in X$ has a finite support then it has a unique smallest finite support, written $\mathrm{supp}(x)$ and referred to as *the* support of x. A nominal set is a $\mathcal{G}$-set all of whose elements have finite support. For example, $\mathbb{A}$ is a nominal set with the permutation action $\sigma \cdot a =_{\mathrm{def}} \sigma a$ for each $a \in \mathbb{A}$, which means that $\mathrm{supp}(a) = \{a\}$. The collection of all finitely-supported subsets of a nominal set X is a natural notion of the *nominal powerset* $\mathcal{P}_{\mathrm{fs}}X$ of X, where the permutation action on $\mathcal{P}_{\mathrm{fs}}X$ is given by $\sigma \cdot A =_{\mathrm{def}} \{\sigma \cdot x \mid x \in A\}$. Similarly the natural permutation action on functions $X \to Y$ is given by $\sigma \cdot f =_{\mathrm{def}} \lambda x.\sigma \cdot (f(\sigma^{-1} \cdot x))$, and the finitely-supported functions $X \to_{\mathrm{fs}} Y$ form a *nominal function space*. Functions with empty support are called *equivariant* and the collection of nominal sets and equivarant functions forms a category $\mathbf{NSet}$ which is a boolean topos.

A.2 FM Set Theory

FM sets are built in a hierarchy $\mathcal{V}_{\mathrm{FM}}$ rather like that of ZF sets, with the following differences. Firstly, the starting point includes the contents of $\mathbb{A}$ as well as the empty set. The permutation action on this collection of atoms gives rise to a permutation action on all of $\mathcal{V}_{\mathrm{FM}}$ by $\in$-recursion, which gives rise to a notion of support for arbitrary elements of $\mathcal{V}_{\mathrm{FM}}$. The iterative process of the construction of $\mathcal{V}_{\mathrm{FM}}$ then continues in such a way as to only include elements that have hereditarily finite supports. Thus the collection of all FM sets behaves as a 'large' nominal set, and any FM set with empty support is a nominal set. The collection of all FM sets and finitely-supported functions forms a category $\mathbf{FMSet}$ which has subcategories $\mathbf{FMSet}_s$ comprising sets and functions all of whose supports are contained in the finite set of names s.

A.3 Name Generation in Nominal Sets

The binary predicate $\#$ is used to state that two FM sets (or two elements of a nominal set) have disjoint supports. Defining $X \otimes Y =_{\mathrm{def}} \{\langle x, y\rangle \mid x \# y\}$ gives a tensor $\otimes$ on $\mathbf{NSet}$. Also, 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. If $X = \{\top, \bot\}$ then $f : \mathbb{A} \to X$ is a predicate on $\mathbb{A}$ and $\mathbf{fresh}\, a\, \mathbf{in}\, fa$ coincides with $\mathsf{И}\, a.\ fa$ where $\mathsf{И}$ is the 'new' quantifier of Pitts and Gabbay.

This permits the definition of the α-equivalence relation $\sim_\alpha$ on $X \times \mathbb{A}$ by setting $\langle x_1, a_1\rangle \sim_\alpha \langle x_2, a_2\rangle$ if and only if $\mathsf{И}\, b.\ (a_1 b) \cdot x_1 = (a_2 b) \cdot x_2$. The quotient by α-equivalence $(X \times \mathbb{A})/\!\sim_\alpha$ may be written δX, and the α-equivalence class containing the pair $\langle a, x\rangle$ is written $[a].x$. Notice that $\mathrm{supp}([a].x) = \mathrm{supp}(x) \setminus \{a\}$ so that a is bound in $[a].x$. For example, $\delta\mathbb{A} = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].a \mid a \in \mathbb{A}\} \,\dot\cup\, \{\mathbf{fresh}\, a\, \mathbf{in}\, [a].a\} \cong \mathbb{A} \,\dot\cup\, \{*\}$ where in effect $*$ is a newly generated name. Thus δ captures the idea of name generation. The operation δ is the object part of a right adjoint to $(-) \otimes \mathbb{A}$; the unit is $x \mapsto \mathbf{fresh}\, a\, \mathbf{in}\, [a].x$ and the counit is given by *concretion*: $([a].x)@b =_{\mathrm{def}} (ab) \cdot x$ for freshly chosen b.

A.4 Name Generation in FM Sets

Turner[5] demonstrates that there is an analogous adjunction to $(-) \otimes \mathbb{A} \dashv \delta$ in FM sets given by the situation

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

where $s \subseteq_{\mathrm{fin}} \mathbb{A}$ and $a \in \mathbb{A} \setminus s$. 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 given in terms of α-equivalence classes $[a].x$ which are defined in the universe of FM sets just as in small nominal sets above: on objects $\delta_a X =_{\mathrm{def}} \{x' \mid \mathsf{И}\, b.\, x'@b \in (ab) \cdot X\}$, 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 is $x \mapsto \mathbf{fresh}\, b\, \mathbf{in}\, [b].x$ as above and the counit is $x' \mapsto x'@a$. Notice that if X has empty support then X is a nominal set and $\delta_a X = \delta X$; similarly if f is an equivariant function. In particular $\delta_a\mathbb{A} = \delta\mathbb{A} \cong \mathbb{A} \,\dot\cup\, \{*\}$. Also if $s' \subseteq s$ it follows that s', $s' \,\dot\cup\, \{a\}$, $\mathbb{A} \setminus s'$ and $\mathbb{A} \setminus s' \setminus \{a\}$ are all objects of $\mathbf{FMSet}_{s \dot\cup \{a\}}$. In this case $\delta_a s' = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].c \mid c \in s'\} \cong s'$ via the isomorphism above, and $\delta_a(s' \,\dot\cup\, \{a\}) = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].c \mid c \in s'\} \,\dot\cup\, \{[a].a\} \cong s' \,\dot\cup\, \{*\}$. Also $\delta_a(\mathbb{A} \setminus s') = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].c \mid c \in \mathbb{A} \setminus s'\} \,\dot\cup\, \{[a].a\} \cong (\mathbb{A} \setminus s') \,\dot\cup\, \{*\}$ and $\delta_a(\mathbb{A} \setminus s' \setminus \{a\}) = \{\mathbf{fresh}\, b\, \mathbf{in}\, [b].c \mid c \in \mathbb{A} \setminus s'\} \cong \mathbb{A} \setminus s'$. Summarising, if $a \in X$ then there is a new name $* \in \delta_a X$. This indicates that δ_a captures the FM-analogue of the idea of name generation.