MURDOCH J. GABBAY*

THE π-CALCULUS IN FM

ABSTRACT: FM (Fraenkel Mostowski) techniques are an approach to metaprogramming on syntax in the presence of binding. We develop novel FM theory and with it develop theory of π-calculus syntax and its operational semantics. Technicalities of name binding and also of name generation in transitions are smoothly handled.

1 INTRODUCTION

Fraenkel-Mostowski (FM) techniques were introduced in [Gabbay, 2000; Gabbay and Pitts, 2001]. They were developed to allow us to reason inductively about syntax with binding. Consider a de Bruijn-style datatype of λ-terms:

$$\Lambda_{db} \stackrel{\text{def}}{=} \text{Var } \mathbf{of}\ \mathbb{N} + \text{App } \mathbf{of}\ \Lambda_{db} \times \Lambda_{db} + \text{Lam } \mathbf{of}\ \Lambda_{db}. \tag{1}$$

Here α-equivalence is equality and this is good but the inductive principle is 'twisted':

$$\begin{array}{l} \big(\forall i.\ \phi(\text{Var}(i)) \quad \wedge \\ \ \forall t_1, t_2.\ \phi(t_1) \wedge \phi(t_2) \rightarrow \phi(\text{App}(t_1, t_2)) \quad \wedge \\ \ \forall t.\ \phi(t) \rightarrow \phi(\text{Lam}(t))\big) \\ \ \rightarrow \forall t.\ \phi(t) \end{array} \tag{2}$$

In the clause $\phi(t) \rightarrow \phi(\text{Lam}(t))$ the subterm t of $\text{Lam}(t)$ is not what we would normally consider a subterm of the λ-term corresponding to $\text{Lam}(t)$.

FM techniques allow α-equivalence to be logical equality, and simultaneously deliver natural induction schemes, as we shall see in this paper for the theory of the π-calculus.

Process calculi abound and many of them involve term-formers for locality, encryption, and so on. These might interact with name-generation and scope extrusion in complicated ways and involve side-conditions on bound and free names. It would be interesting and useful if we could show that FM techniques could simplify these presentations, and perhaps also their proofs.

This paper starts with a brief account of basic FM theory which expands into quite a detailed account of some novel theory which we shall find useful: some improved proofs and definitions of known constructions like abstraction types, and new material on what we call abstractive functions, which turn out to be quite a powerful unifying tool in FM.

*M.J.Gabbay, mjg1003@cl.cam.ac.uk, Computer Laboratory, Cambridge University, UK

Fairouz Kamareddine (eds.),
Thirty Five Years of Automating Mathematics *1–24.*

We then take a simple π-calculus (without matching or replication) and indicate how to specify its syntax, operational semantics, and some standard operational and syntactic equivalence relations on it. We outline details of the proofs and see that the entire development can be made very close to normal informal practice, only that thanks to FM the treatment of names and binding in syntax *and* name-generation in operational semantics, is completely rigorous.

We conclude with a brief survey of other techniques, and outline possible future research.

2 $\mathbb{A}$-PERMUTATION

Hypothesise some set of **atoms** $a, b, c, \ldots \in \mathbb{A}$, which we shall use throughout this document to represent object-level variable names. Also write $\mathcal{P}_{fin}(\mathbb{A})$ for the set of finite subsets of $\mathbb{A}$. Now consider a name-carrying datatype of λ-terms:

$$\Lambda = \mathrm{Var}\ \mathbf{of}\ \mathbb{A} + \mathrm{App}\ \mathbf{of}\ \Lambda \times \Lambda + \mathrm{Lam}\ \mathbf{of}\ \mathbb{A} \times \Lambda \tag{3}$$

Capture-avoiding name for name substitution might be defined as:

$$\begin{array}{lll} [b/a]\mathrm{Var}(n) & = \mathrm{Var}([b/a](n)) & \\ [b/a]\mathrm{App}(t_1, t_2) & = \mathrm{App}([b/a]t_1, [b/a]t_2) & \\ [b/a]\mathrm{Lam}(n, t) & = \mathrm{Lam}(n, [b/a]t) & a, b \neq n \\ [b/a]\mathrm{Lam}(a, t) & = \mathrm{Lam}(a, t) & \\ [b/a]\mathrm{Lam}(b, t) & = \mathrm{Lam}(n, [b/a][n/b]t) & \\ & \qquad n = gsym(FV(t) \cup \{a, b\}) & \end{array} \tag{4}$$

Here $gsym(S)$ takes $S \in \mathcal{P}_{fin}(\mathbb{A})$ and returns some $x \notin S$. The notation $[b/a](n)$ denotes the function $[b/a] : \mathbb{A} \to \mathbb{A}$ acting on n, where $[b/a]n = n$ for $n \neq a$ and $[b/a](a) = b$.

This is *almost* a function which distributes through the syntax of a term and acts on the atoms inside it, only we must take account of the fact that $\mathrm{Lam}(n, t)$ abstracts n in t.

Now write $(a\ b) : \mathbb{A} \to \mathbb{A}$ for the function such that $(a\ b)(a) = b$, $(a\ b)(b) = a$, and $(a\ b)(n) = n$ for $n \neq a, b$, and call this a **transposition**. Using it we can define a form of capture-avoiding name for name substitution more simply:

$$\begin{array}{ll} (b\ a) \cdot \mathrm{Var}(n) & = \mathrm{Var}((b\ a)(n)) \\ (b\ a) \cdot \mathrm{App}(t_1, t_2) & = \mathrm{App}((b\ a) \cdot t_1, (b\ a) \cdot t_2) \\ (b\ a) \cdot \mathrm{Lam}(n, t) & = \mathrm{Lam}((b\ a)(n), (b\ a) \cdot t). \end{array} \tag{5}$$

LEMMA 1. *For $t \in \Lambda$ and $b, a : \mathbb{A}$, if $b \notin FV(t)$ then $[b/a]t =_\alpha (b\ a).t$.*

Here we write α-equivalence as $=_\alpha$.

Proof. By induction on term size using the observation that for $c, d \notin FV(t)$ (but possibly occurring bound in t), $(c\ d) \cdot t =_\alpha t$. ■

For example:

$$(c\ d) \cdot \mathrm{Lam}(d, \mathrm{Lam}(c, \mathrm{Var}(c))) = \mathrm{Lam}(c, \mathrm{Lam}(d, \mathrm{Var}(d)))$$

and $\mathrm{Lam}(c, \mathrm{Lam}(d, \mathrm{Var}(d))) =_\alpha \mathrm{Lam}(d, \mathrm{Lam}(c, \mathrm{Var}(c)))$.

DEFINITION 2. Write $\pi, \kappa \in \mathbb{A}_\Pi$ for the subgroup of functions generated by $(a\ b)$. Write **Id** for the identity $\lambda a : \mathbb{A}.a$ and $\circ$ for the group composition, which is functional composition.

Let an FM **set** be a pair $(X, \cdot)$ of a set X with an $\mathbb{A}_\Pi$ permutation action $\cdot : \mathbb{A}_\Pi \times X \to X$ (written infix) satisfying

$$\forall x : X.\ \exists S \in \mathcal{P}_{fin}(\mathbb{A}).\ \forall a, b \notin S.\ (a\ b) \cdot x = x. \tag{6}$$

We shall tend to write just X for the FM set $(X, \cdot)$. We call (6) the **finite support property**, we shall see why below.

Write FM-Sets for the **category of** FM **sets** with objects FM sets X, Y and arrows functions $f : X \to Y$ such that $\forall a, b.\ f((a\ b) \cdot x) = (a\ b) \cdot f(x)$.

A permutation action is a function $\mathbb{A}_\Pi \times X \to X$ which we write infix $\pi \cdot x$, satisfying $\mathbf{Id} \cdot x = x$ and $\pi \cdot \pi' \cdot x = \pi \circ \pi' \cdot x$.

DEFINITION 3. Say $x \in X$ is **equivariant** when $(a\ b) \cdot x = x$ for all a, b.

Thus x is equivariant when we can take $S = \emptyset$ in (6) above.

$\mathbb{A}$ is an FM set with the natural action $(a\ b) \cdot n = (a\ b)(n)$. Any ordinary set, such as $\mathbb{B} = \{\top, \bot\}$ boolean truth values, $\mathbb{N}$ the natural numbers, or $\mathbb{R}$ the reals, can be made an FM set with the trivial action $(a\ b) \cdot x = x$ for all x, thus such that every element is equivariant.

Given FM sets X and Y the permutation action naturally extends to functions $f : X \to Y$ by conjugation, so

$$((a\ b) \cdot f)(x) = (a\ b) \cdot \big(f((a\ b) \cdot x)\big). \tag{7}$$

It is not necessarily the case that such a function satisfies (6). For example a choice function $\mathcal{P}_{fin}(\mathbb{A}) \to \mathbb{A}$ cannot, and it is interesting to verify this (see [Gabbay and Pitts, 2001, Remark 4.6] for a discussion). Note that equivariant functions are precisely the arrows of FM-Sets.

2.1 Set-theoretic model of FM-Sets

We obtain a concrete model of FM-Sets as follows. We construct a set-theoretic cumulative hierarchy starting with atoms $a \in \mathbb{A}$ as base elements: in the standard notation $V_0 = \mathbb{A}$.

Given V_i we construct V_{i+1} by collecting the subsets $U \subseteq V_i$ that satisfy (6). This is a model of ZF set theory with atoms ($\mathbb{A}$ in this case), along with the extra axiom (6), see [Gabbay, 2000; Gabbay and Pitts, 2001] for a full development.

The permutation action is given by the pointwise action on the elements: formally $\pi \cdot a = \pi(a)$ and $\pi \cdot x = \{\pi \cdot y \mid y \in x\}$ (this is clearly a permutation action).

Functions f are implemented in sets as graphs $\{\langle x, f(x)\rangle\}$. Applying a permutation according to this action, $(a\ b) \cdot f = \{\langle (a\ b) \cdot x, (a\ b) \cdot f(x)\rangle\}$, and translating that back into functional notation this is precisely (7).

Recall that the action on a set is pointwise on its elements. Thus an equivariant set X is one such that $X = \{(a\ b) \cdot x \mid x \in X\}$. Equivariant sets can be seen as objects in FM-Sets and arrows as equivariant function-sets between them.

THEOREM 4. *The construction above gives a model of* FM-Sets.

COROLLARY 5. FM-Sets *is a boolean topos.*

We see why permutation is better-behaved than atom-for-atom substitution $[b/a]$. If we try giving that a similar inductive interpretation on the set-universe, so $[b/a]n = [b/a](x)$ for $x \in \mathbb{A}$ and $[b/a]x = \{[b/a]y \mid y \in x\}$ for $x \notin \mathbb{A}$, things do not work out smoothly. For example $f = \{\langle c, c\rangle \mid c \in \mathbb{A}\}$ denotes the identity $\lambda x : \mathbb{A}.x$ and $(b\ a) \cdot f = f$. However we see that $[b/a]f = \{\langle c, c\rangle \mid c \in \mathbb{A} \setminus \{a\}\}$ is not equal to f and is no longer even a total function-set from $\mathbb{A}$ to $\mathbb{A}$.

2.2 Support and apartness

DEFINITION 6. For $S \in \mathcal{P}_{fin}(\mathbb{A})$ write $Fix(S)$ for the set of π fixing S pointwise:

$$\pi \in Fix(S) \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \forall n \in S.\ \pi(n) = n.$$

Say π **fixes** $x : X$ when $\pi \cdot x = x$. Say $S \subseteq \mathbb{A}$ **supports** $x : X$ when if π fixes pointwise S then $\pi \cdot x = x$:

$$(S\ \textit{supports}\ x) \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \forall \pi \in Fix(S).\ \pi \cdot x = x.$$

Note that $Fix(S)$ is a group; this will be useful later. (6) says in this terminology that **every** x **has a finite supporting set of atoms**.

THEOREM 7. *Any $x : X$ has a unique minimal supporting set of atoms. Write it $S(x)$ and call it the* **support** *of x.*

Proof. Construct $S(x)$ as $\bigcap \{S \in \mathcal{P}_{fin}(\mathbb{A}) \mid S\ \textit{supports}\ x\}$. The lemma below proves this supports x. ■

LEMMA 8. *If S and S' support x then so does $S \cap S'$.*

Proof. Suppose κ fixes $S \cap S'$. We must show $\kappa \cdot x = x$. Choose some injection $\iota : S' \setminus S \hookrightarrow \mathbb{A} \setminus (S \cup S')$ and make it into a bijection π by letting $\pi(\iota a) = a$ for a in the image of $S' \setminus S$ (note that $\pi \circ \pi = \mathbf{Id}$). Since $\pi \in Fix(S)$ we know $\pi \cdot x = x$. Observe also that $\pi \circ \kappa \circ \pi$ fixes S', so $\pi \circ \kappa \circ \pi \cdot x = x$. It follows by group algebra that $\kappa \cdot x = x$. ■

In other works $S(x)$ is written '**Supp**(x)'. We prefer a more compact notation here for typographical reasons.

DEFINITION 9. For $x : X$ and $y : Y$ write

$$x\#y \quad \stackrel{\text{def}}{\Leftrightarrow} \quad S(x) \cap S(y) = \emptyset.$$

We read this as x **is (𝔸-)apart from** y.

We can verify by calculation that $S(a \in \mathbb{A}) = \{\mathbb{A}\}$. Two useful results follow: $a\#b$ if and only if $a \neq b$, and if $a, b\#x$ then $(a\ b) \cdot x = x$.

$S(S \in \mathcal{P}_{fin}(\mathbb{A})) = S$. Write $L \in \mathcal{P}_{cofin}(\mathbb{A})$ for the set of **cofinite sets** of atoms. L is cofinite when $\mathbb{A} \setminus L$ is finite. $S(L) = \mathbb{A} \setminus L$. If $U \subseteq \mathbb{A}$ is neither finite nor cofinite then $S(U) = \mathbb{A}$ so in FM-Sets $\mathcal{P}(\mathbb{A}) = \mathcal{P}_{fin}(\mathbb{A}) \cup \mathcal{P}_{cofin}(\mathbb{A})$.

If P is a predicate $\mathbb{A} \to \mathbb{B}$ we can write $\mathsf{И}a.\ P(a)$ for $P \in \mathcal{P}_{cofin}(\mathbb{A})$, this is the FM **И-quantifier** (pronounced 'New'), familiar from [Gabbay and Pitts, 2001, p.8]. The observation on powersets becomes the some/any property of FM atoms: $\neg\mathsf{И}a.\ P(a) \Leftrightarrow \mathsf{И}a.\ \neg P(a)$. We shall see $\mathsf{И}$ as an instance of a very general construction in the next subsection.

Recall that arrows in FM-Sets are equivariant functions f on underlying sets, so $f(\pi \cdot x) = \pi \cdot f(x)$. It is then easy to show that $S(fx) \subseteq S(x)$. Again, it is interesting to verify these facts. We shall use them often.

2.3 Abstractive functions and abstractions

DEFINITION 10. For $a : \mathbb{A}$ and $y : Y$ define

$$[a]y \quad \stackrel{\text{def}}{=} \quad \bigcap \left\{ U \subseteq \mathbb{A} \times Y \;\middle|\; \langle a, y\rangle \in U \wedge S(U) \subseteq S(y) \setminus \{a\} \right\}. \qquad (8)$$

At least one such U always exists since $U = \mathbb{A} \times Y$ is equivariant. By calculation $S(a) = \{a\}$ so the last part of this formula could be read as $S(y) \setminus S(a)$. (8) is an instance of a general construction $[x]y$ whose definition is identical except that we write $S(y) \setminus S(x)$.

LEMMA 11. *For a, y, and $[a]y$ as above, $[a]y$ is precisely the orbit E of $\langle a, y\rangle$ under permutations $\pi \in F \stackrel{\text{def}}{=} Fix(S(y) \setminus \{a\})$.*

Proof. Suppose $\pi \in F$. Then $\pi \cdot E = \{\langle \pi \cdot a', \pi \cdot y' \rangle \mid \langle a', y' \rangle \in E\}$. But $\langle a', y' \rangle \in E$ precisely when $a' = \pi' \cdot a$ and $y' = \pi' \cdot y$ for $\pi' \in F$, and since F is a group $\pi \cdot E = E$. Therefore $[a]y \subseteq E$.

By construction $\langle a, y \rangle \in [a]y$. Also $\pi \cdot [a]y = [a]y$ for all $\pi \in F$, so $\langle \pi(a), \pi \cdot y \rangle \in [a]y$ for all these π. Therefore $E \subseteq [a]y$. ■

As a corollary, $S([a]y) = S(y) \setminus \{a\}$.

DEFINITION 12. Define **$\mathbb{A}$-abstraction** by

$$[\mathbb{A}]Y \stackrel{\text{def}}{=} \{[a]y \mid a : \mathbb{A}, y : Y\}. \tag{9}$$

See [Gabbay and Pitts, 2001, Section 5]. The canonical map $abs : \mathbb{A} \times Y \to [\mathbb{A}]Y$ 'binds' a (or more properly the support of a) in y. abs is otherwise injective:

LEMMA 13. *If* $[a]y = [a]y'$ *then* $y = y'$.

Proof. By easy calculation. ■

COROLLARY 14. $abs : \mathbb{A} \times Y \to [\mathbb{A}]Y$ *is universal amongst* $F : \mathbb{A} \times Y \to Z$ *such that* $\forall a, y.\ a \# F(a, y)$.

Thus $[a]y$ 'y with a bound'. Since a is bound—abs is not injective and for any $b \# y$, $(a\ b) \cdot [a]y = [a]y$—we introduce a nameless notation $\hat{y}$ for an arbitrary element of $[\mathbb{A}]Y$.

Abstraction types with their universal properties are an example of an abstractive function.

2.4 Abstractive functions

Any $f : X \to Y$ induces an inverse map $f^* : Y \to \mathcal{P}(X)$

$$f^* : y : Y \longmapsto \{x : X \mid f(x) = y\}.$$

$f^*(y)$ partitions into orbits under $Fix(S(y))$ (as well as any other subgroup of $\mathbb{A}_\Pi$). Write $O_f(y)$ for this particular set.

Each orbit O has a **breadth** $|S(z \in O)|$ the cardinality of $S(z)$ of a representative $z \in O$. Write this $|O|$.

DEFINITION 15. Suppose f is such that for all y there is a unique orbit M of greatest breadth:

$$\exists M \in O_f(y).\ \forall O \in O_f(y).\ |O| \geq |M| \implies O = M$$

Write $M_f(y)$ for this orbit when it exists and say f has **orbits of maximal breadth**.

The set $\bigcup_y M_f(y)$ is an FM set, write it just M_f.

DEFINITION 16 (Abstractive function). Say $f : X \to Y$ is **Barendregt abstractive** or just **abstractive** when it has orbits of maximal breadth. When $O_f(y)$ is always a singleton set, so orbits are maximal because they are unique, say f is **purely abstractive**.

An abstractive $f{:}X \to Y$ induces a purely abstractive restriction $f{:}M_f \to Y$. By construction, if f is surjective, so is its restriction to M_f.

1. For $f = \lambda x.*{:}\mathbb{A} \to \mathbf{1}$ the set $O_f(*)$ has just one orbit, $\mathbb{A}$, with breadth 1. f is purely abstractive.
2. For $f = \lambda x.* : \mathbb{A}^2 \to \mathbf{1}$ the set $O_f(*)$ has two orbits, $\{\langle a, b\rangle \mid a \neq b\}$ with breadth 2 and $\{\langle a, a\rangle \mid a : \mathbb{A} \cong \mathbb{A}\}$ with breadth 1. f is Barendregt abstractive.
3. For $f = \lambda x.* : \mathcal{P}_{fin}(\mathbb{A}) \to \mathbf{1}$ the set $O_f(*)$ is isomorphic to $\mathbb{N}$. There is no orbit of maximal breadth and f is not abstractive.
4. For $f = \lambda x.\mathbb{A} + \mathbb{A} : \mathbf{1}$ the set $O_f(*)$ has two orbits of equal breadth. f is not abstractive.
5. For $f = abs : \mathbb{A} \times Y \to [\mathbb{A}]Y$ the set $O_f([a]y)$ is the orbit E described in Lemma 11. *abs* is purely abstractive.
6. For $f = \pi_2 : \mathbb{A} \times Y \to Y$ (second projection) $O_f(y)$ has one orbit of maximal breadth $\{\langle a, y\rangle \mid a\#y\}$. f is Barendregt abstractive.
7. Recall from (3) the datatype Λ. The quotient by α-equivalence, write it $\alpha : t \mapsto [t]_{=_\alpha} : \Lambda \to \Lambda/{=_\alpha}$, is Barendregt abstractive. M_f is the set of Barendregt representative terms.

Abstractive functions give an abstract account of the Barendregt variable convention (amongst other things), because the orbit of maximal breadth consists of x representing y with the most possible different names for those 'bound' by the function. Abstractive functions are also related to FreshML abstraction-patterns. The interpreter generates fresh names for atoms in (possibly nested) abstraction-patterns.

If we read these example $f{:}X \to Y$ 'backwards' as 'maps' $Y \to M_f$, they do the following respectively: choose a canonical atom, choose a canonical pair of distinct atoms, choose a canonical largest finite set of atoms (there is none), choose a canonical atom in a disjoint sum (there is none), choose a fresh atom for y, choose a Barendregt representative for t. Theorem 21 will make this intuition formal.

LEMMA 17. *If $f : X \to Y$ and $g : X' \to Y'$ have orbits of maximal breadth then so do $f + g : X + X' \to Y + Y'$ and $f \times g : X \times X' \to Y \times Y'$ (with the obvious definitions).*

If $f : X \to Y$ and $g : Y \to Z$ have orbits of maximal breadth so does $g \circ f : X \to Z$.

Proof.(Sketch) The case of disjoint sum is very easy. For the case of products, $M_{f \times g}\langle y_1, y_2\rangle$ is the $Fix(S(\langle y_1, y_2\rangle))$-orbit of $\langle z_1, z_2\rangle \in M_f(y_1) \times M_g(y_2)$ where z_1 and z_2 are chosen such that $S(z_1) \setminus S(y_1)$ and $S(z_2) \setminus S(y_2)$ are disjoint (which makes the cardinality of $S(z_1) \cap S(z_2)$ minimal, details omitted).

The case of functional composition is also easy. $M_{g \circ f}(z)$ is equal to $M_g(u)$ for a representative $u \in M_f(z)$. ■

COROLLARY 18. *If $f{:}X \to Y$ and $g{:}X' \to Y'$ are (Barendregt) abstractive then $f+g{:}X+X' \to Y+Y'$ is (Barendregt) abstractive and $f \times g{:}X \times X' \to Y \times Y'$ is Barendregt abstractive. If $f{:}X \to Y$ and $g{:}Y \to Z$ are (Barendregt) abstractive then so is $g \circ f : X \to Z$.*

Proof. Using Lemma 17. ■

DEFINITION 19. For $f : X \to Y$ abstractive and surjective and $F : X \to Z$, say F **factors through f on orbits of maximal support** when there is some $h : Y \to Z$ such that for all $x \in M_f$, $F(x) = hf(y)$.

$$\begin{array}{ccc} x \in M_f & \xrightarrow{F} & Fx = hy \\ \Big\downarrow f & \nearrow_{h} & \\ y = fx. & & \end{array} \tag{10}$$

DEFINITION 20. For f and F as in the previous definition, write $F \leq_{ab} f$ when $\forall x \in M_f.\ S(Fx) \subseteq S(fx)$.

THEOREM 21. *For f and F as above, $F \leq_{ab} f$ if and only if F factors through f on orbits of maximal support.*

Proof. Suppose for all $x \in M_f$, $Fx = h \circ f(x)$. Then $S(Fx) = S(hf(x)) \subseteq S(fx)$.

Conversely suppose $F \leq_{ab} f$ and $x \in M_f$. Write $y = fx$ and let hy be Fx. If $Fx' = y$ then $x' = \pi{\cdot}x$ for some $\pi \in Fix(S(y))$ and $Fx' = F\pi{\cdot}x = \pi{\cdot}(Fx)$. Now $S(Fx) \subseteq S(y)$ by assumption so $\pi \cdot (Fx) = Fx$. So h is well-defined. ■

As a corollary, purely abstractive surjective $f : X \to Y$ universal amongst maps $F : X \to Z$ such that $F \leq_{ab} f$. If f is Barendregt abstractive and surjective, it is universal 'on orbits of maximal breadth'.

- For $abs : \mathbb{A} \times X \to [\mathbb{A}]X$ the condition $F \leq_{ab} abs$ is $S(F(a,x)) \subseteq S(x) \setminus S(a)$. We recover Corollary 14. Compare with [Gabbay and Pitts, 2001, p.15, Lemma 6.3].

- For $\pi_2 : \mathbb{A} \times X \to X$ the condition $F \leq_{ab} f$ is $a\#F(a,x)$. Write $И a.\ F(a,x)$ for the unique value of $F(a,x)$ when $a\#x$ (for fixed x). This is a functional version of the $И$ quantifier mentioned in the last subsection.

  When $Z = \mathbb{B}$ the condition $a\#F(a,x)$ is always satisfied since $F(a,x) = \top$ or $F(a,x) = \bot$, and we obtain the $И$-quantifier mentioned in §2.3. (The X makes parameters explicit.)

  Now suppose that $F = P : \mathbb{A} \times X \to \mathbb{B}$. The universality property on orbits of maximal breadth then gives $И a.\ P(a,x)$ if and only if $\forall a.\ a\#x \Rightarrow P(a,x)$, a known commutativity property between $И$ and $\forall$ which we shall find useful.

- Recall that for Λ defined in (3) the quotient $\alpha : \Lambda \to \Lambda/{=_\alpha}$ is Barendregt abstractive. Therefore $F : \Lambda \to Z$ respects α-equivalence on Barendregt representative terms t precisely when for all such t and $a \in bn(t)$, $a\#F(t)$.

DEFINITION 22. When $f : X \to Y$ is abstractive and $F \leq_{ab} f$, write the h we construct above as $И_f F : Y \to Z$.

We write $(И_f F)(y)$ as $Иy{=}f(x).\ Fx$, this is the unique value of F at x such that $f(x) = y$ and x is a Barendregt representative in the sense discussed above.

We call this the **generalised $И$ quantifier**.

3 THE π-CALCULUS

3.1 An inductive datatype of π-calculus terms up to binding

Define an inductive FM datatype of π-calculus terms up to $=_\alpha$:

$$\begin{array}{rcl} \Pi & \stackrel{\text{def}}{=} & 1 + \Pi + \mathbb{A}^2 \times \Pi + \mathbb{A} \times [\mathbb{A}]\Pi + \Pi^2 + [\mathbb{A}]\Pi \\ & & P ::= 0 \mid \overline{x}yP \mid x\hat{P} \mid (P \mid P) \mid \nu\hat{P} \end{array} \tag{11}$$

To construct a term of the form $\nu\hat{P}$ it suffices to provide $y : \mathbb{A}$, $P : \Pi$, and apply ν to the abstraction $[y]P$. Recall from after Corollary 14 that we can write such an abstraction namelessly as $\hat{P}$.

Π is inductively defined and has a primitive recursion scheme. For any Z, given

$$f_0 : Z \qquad f_\tau : \Pi \to Z \qquad f_o : \mathbb{A}^2 \times \Pi \to Z \qquad f_i : \mathbb{A} \times [\mathbb{A}]\Pi \to Z$$
$$f_p : \Pi^2 \to Z \qquad f_\nu : [\mathbb{A}]\Pi \to Z$$

there is a unique $f : \Pi \to Z$ such that all of the following hold:

$$f(0) = f_0 \quad f(\tau.P) = f_\tau P \quad f(\overline{x}y.P) = f_o(x, y, P) \quad f(x\hat{P}) = f_i(x, \hat{P})$$
$$f(P \mid Q) = f_p(P, Q) \quad f(\nu\hat{P}) = f_\nu(\hat{P}). \quad (12)$$

However abstraction $abs : \mathbb{A} \times \Pi \to [\mathbb{A}]P$ has special properties formalised in its being an abstractive function. We take advantage of this to write primitive recursive definitions *which look like standard ones*.

Recall the notations for renamings $[b/a]$ from §2.

DEFINITION 23. Analogously to $\mathbb{A}_\Pi$ write $\sigma \in \mathbb{A}_\Sigma \subseteq \mathbb{A}^\mathbb{A}$ for the monoid generated by renamings $[b/a]$, and write **Id** for the identity $\lambda x.x$.

By (12) we can define **name substitution** on Π by primitive recursion as follows:

$$0\sigma = 0 \quad (\tau.P)\sigma = \tau.P\sigma \quad (\overline{x}y.P)\sigma = \overline{x\sigma}y\sigma.P\sigma$$
$$(x\hat{P})\sigma = x\sigma\mathsf{M}\hat{P}{=}[n]P.\ x\sigma[n](P\sigma) \quad (13)$$
$$(P_1 \mid P_2)\sigma = P_1\sigma \mid P_2\sigma \qquad (\nu\hat{P})\sigma = \mathsf{M}\hat{P}{=}[n]P.\ \nu[n](P\sigma).$$

We also show an induction principle derived from primitive recursion. For a predicate ϕ on Π, if

For all $x, P, Q, \hat{P}$
$$\phi(0) \quad \phi(P) \to \phi(\tau.P) \quad \phi(P) \to \phi(\overline{x}y.P) \quad \phi(P) \wedge \phi(Q) \to \phi(P \mid Q)$$
$$\mathsf{M}\hat{P}{=}[y]P.\ \phi(P) \to \phi(x[y]P) \qquad \mathsf{M}\hat{P}{=}[y]P.\ \phi(P) \to \phi(\nu[y]P)$$
$$\text{hold, then} \quad \forall P.\ \phi(P). \quad (14)$$

A more usual induction scheme would have clauses for input and restriction as follows:

$$\forall x, y, P.\ \phi(P) \to \phi(x[y]P) \qquad \forall y, P.\ \phi(P) \to \phi(\nu[y]P)$$

These do not assume y fresh, and since this is the only case we care about in practice it can be useful to, in effect, build it into the datatype.

If $y\#\sigma$ and $y \neq x$ then

$$y\sigma = y \quad (\nu[y]P)\sigma = \nu[y](P\sigma), \quad \text{and} \quad (x[y]P)\sigma = x\sigma[y](P\sigma). \quad (15)$$

We read the second two clauses 'for free' off the appropriate clauses of (13) (input and restriction), because $y\#[y]P$ always holds. Concerning the first clause, we say of many FM results in this paper that they follow 'by (easy) calculation'. The following useful result follows by slightly involved calculation, for once we give it in full.

LEMMA 24. *$y\#\sigma$ if and only if y is neither in the domain nor the image of σ as a partial function on atoms.*

Proof. Suppose $y\#\sigma$. σ is finitely generated by renamings so cofinitely many atoms z are not in the domain or image of σ, or formally $\forall z'.\ \sigma z' = z \Rightarrow z = z'$.

By (6) also cofinitely many z have $z\#\sigma$, so choose one z such that both hold. Then $(y\ z)(\sigma y) = ((y\ z)\cdot\sigma)((y\ z)\cdot y) = \sigma z = z$, so $\sigma y = y$. Also suppose that $y = \sigma y'$ for some y'. Then $z = (y\ z)\cdot y = \sigma((y\ z)\cdot y')$. So $(y\ z)\cdot y' = z$ and therefore $y' = y$.

Conversely, suppose y is neither in the domain nor image of σ. Take fresh z not in the domain or image as before. Write $\tau = (z\ y)\cdot\sigma$. We show $\tau = \sigma$ by applying them to arbitrary z'.

$\tau z' = (z\ y)\cdot(\sigma(z\ y)\cdot z')$ by definition. $\tau z = (z\ y)\cdot\sigma y = z = \sigma z$. $\tau y = (z\ y)\cdot\sigma z = y = \sigma y$. Otherwise $\tau z' = (z\ y)\cdot\sigma z'$. We know $\sigma z' \neq z, y$, so this is just $\sigma z'$. ■

COROLLARY 25. *If $y\#\sigma$ and $y\#x$, then $y\#\sigma x$.*

We shall use these results often in our proofs.

We return briefly to (13): $(x\hat{P})\sigma = \mathit{I\!N}\hat{P}{=}[n]P.\ x\sigma[n](P\sigma)$ the clause for input can be equivalently expressed by

$$\forall x.\ \mathit{I\!N} n.\ \forall P.\ (x[n]P)\sigma = x\sigma[n]P\sigma. \tag{16}$$

(We can treat the clause for restriction similarly.) This presentation style is useful, we explore it in detail in the next subsection.

3.2 Operational semantics for Π

DEFINITION 26 (Actions). Define a datatype of actions by

$$Act \stackrel{\text{def}}{=} 1 + \mathbb{A}^2 + \mathbb{A}^2 \qquad \alpha, \mu ::= \tau \mid \overline{x}y \mid xy. \tag{17}$$

Write $\langle P, [z]\langle l, Q\rangle\rangle$ an element of $\Pi \times [\mathbb{A}](Act \times \Pi)$ as $P \xrightarrow{z.l} Q$.

DEFINITION 27. We define an **early and late transition relations**

$$\rightarrow \subseteq \Pi \times [\mathbb{A}](Act \times \Pi)$$

inductively by the rules of Figure 1.

$$\tau\textbf{:} \qquad \forall P.\ \text{И} z.\ \tau.P \overset{z.\tau}{\to} P$$

$$\textbf{Out:} \qquad \forall x, y, P.\ \text{И} z.\ \overline{x}y.P \overset{z.\overline{x}y}{\to} P$$

$$\textbf{In:} \qquad \forall x, n.\ \text{И} y.\ \forall P.\ x[y]P \overset{y.xn}{\to} P[n/y]$$

$$\textbf{Par1:} \qquad \forall P_1, P_2.\ \text{И} z.\ \forall \mu, Q_1.\ \frac{P_1 \overset{z.\mu}{\to} Q_1}{P_1 \mid P_2 \overset{z.\mu}{\to} Q_1 \mid P_2}$$

$$\textbf{Close1:} \qquad \forall P_1, P_2, x.\ \text{И} y, z.\ \forall Q_1, Q_2.\ \frac{P_1 \overset{y.\overline{x}y}{\to} Q_1 \quad P_2 \overset{z.xy}{\to} Q_2}{P_1 \mid P_2 \overset{z.\tau}{\to} \nu[y](Q_1 \mid Q_2)}$$

$$\textbf{Com1:} \qquad \forall P_1, P_2, x, y.\ \text{И} z.\ \frac{P_1 \overset{z.\overline{x}y}{\to} Q_1 \quad P_2 \overset{z.xy}{\to} Q_2}{P_1 \mid P_2 \overset{z.\tau}{\to} Q_1 \mid Q_2}$$

$$\textbf{Open:} \qquad \forall x.\ \text{И} y.\ \forall P, Q.\ \text{И} z.\ \frac{P \overset{z.\overline{x}y}{\to} Q}{\nu[y]P \overset{y.\overline{x}y}{\to} Q}$$

$$\textbf{Res:} \qquad \forall \mu, z.\ \text{И} y.\ \forall P, Q.\ \frac{P \overset{z.\mu}{\to} Q}{\nu[y]P \overset{z.\mu}{\to} \nu[y]Q}$$

(a) Early transitions

$$\textbf{In:} \qquad \forall x.\ \text{И} y.\ \forall P.\ x[y]P \overset{y.xy}{\to} P$$

$$\textbf{Close1:} \qquad \forall P_1, P_2, x.\ \text{И} y.\ \forall Q_1, Q_2.\ \frac{P_1 \overset{y.\overline{x}y}{\to} Q_1 \quad P_2 \overset{y.xy}{\to} Q_2}{P_1 \mid P_2 \overset{y.\tau}{\to} \nu[y](Q_1 \mid Q_2)}$$

$$\textbf{Com1:} \qquad \forall P_1, P_2, x, y, Q_1.\ \text{И} z.\ \forall Q_2.\ \frac{P_1 \overset{z.\overline{x}y}{\to} Q_1 \quad P_2 \overset{z.xz}{\to} Q_2}{P_1 \mid P_2 \overset{z.\tau}{\to} Q_1 \mid Q_2[z/y]}$$

(b) Late transitions as early, except for rules shown

We elide symmetric versions **(Par2)**, **(Close2)**, and **(Com2)**.

Figure 1. FM presentation of late and early π-calculus transitions.

A transition in our sense is an element $\langle P, [z]\langle \mu, Q\rangle\rangle$. We model bound output by literally outputting a bound name. For example the canonical 'output of a bound channel name' $\nu y.\overline{x}y \stackrel{\overline{x}(y)}{\rightarrow} 0$ which may be well-known to the reader [Milner *et al.*, 1992], but also $\nu y.\overline{x}y \stackrel{\overline{x}(y')}{\rightarrow} 0$ and $\nu y.\overline{x}y \stackrel{\overline{x}(z)}{\rightarrow} 0$, are represented by the single element $\langle \nu y.\overline{x}y, [y]\langle \overline{x}y, 0\rangle\rangle$ which we write as $\nu y.\overline{x}y \stackrel{[y]\overline{x}y}{\rightarrow} 0$. The restriction that $y \neq x$ is built into the the order of the quantifiers $\forall x.\ \mathit{И} y.\ \forall P, Q$ in **(Open)**, which we now discuss.

By the discussion on quantifiers at the end of the previous section we can expand the quantifiers $\forall x.\ \mathit{И} y.\ \forall P.\ \mathit{body}$ of the late **(In)** as $\forall x, y, P.\ y\#x \Rightarrow \mathit{body}$. We can expand the quantifiers $\forall x, n.\ \mathit{И} y.\ \forall P.\ \mathit{body}$ of the early **(In)** as $\forall x, y, P.\ y\#x, n \Rightarrow \mathit{body}$.

For a more complex example, the quantifiers $\forall x.\ \mathit{И} y.\ \forall P, Q.\ \mathit{И} z.\ \mathit{body}$ of **(Open)** expand as $\forall x, y, P, Q, z.\ y\#x \wedge z\#x, y, P, Q \Rightarrow \mathit{body}$. Now recall that $y\#x$ is equivalent to $y \neq x$ and $y\#P$ equivalent to $y \notin n(P)$. Here $n(P)$ is the free names of P, we write it just n because in the FM datatype there *are* no bound names.

Thus we see how the mix of $\forall$ and $\mathit{И}$ encodes side-conditions on free and bound names. The advantage of using $\mathit{И} z$ is that we can assume z fresh also for other parts of our context, for example a substitution σ, or perhaps a different process R. We now consider an example.

Recall the notation $\sigma : \mathbb{A}_\Sigma$ and the renaming action, defined in and after Definition 23. Recall from Lemma 24 that if $y\#\sigma$ then $y\sigma = y$. [Milner *et al.*, 1992, Lemma 3] in this framework and notation becomes:

LEMMA 28. *For* $\sigma : \mathbb{A}_\Sigma$

$$\forall P.\ \mathit{И} z.\ \forall \alpha, Q.\ P \stackrel{z.\alpha}{\rightarrow} Q \implies P\sigma \stackrel{z.(\alpha\sigma)}{\rightarrow} Q\sigma.$$

Here $\alpha\sigma$ *denotes the evident inductive action of* σ *on actions* α.

Proof. By induction on the transition using hypothesis

$$\phi(t) \stackrel{\text{def}}{=} \forall P.\ \mathit{И} z.\ \forall \alpha, Q.\ t = (P \stackrel{z.\alpha}{\rightarrow} Q) \implies P \stackrel{z.\alpha}{\rightarrow} Q \wedge P\sigma \stackrel{z.(\alpha\sigma)}{\rightarrow} Q\sigma.$$

The case **(Close1)**. Suppose P_1, P_2, x, y, Q_1, Q_2 are such that $y\#P_1, P_2, x,$ *and* σ. Suppose also $P_1 \stackrel{y.\overline{x}y}{\rightarrow} Q_1$ and $P_2 \stackrel{y.xy}{\rightarrow} Q_2$. By Lemma 24 $y\#P_1\sigma$, so by induction hypothesis $P_1\sigma \stackrel{y.\overline{x\sigma}y}{\rightarrow} Q_1\sigma$. Similarly for P_2. $y\#P_1\sigma, P_2\sigma, x\sigma$ so we apply **(Close1)** to these two new transitions.

The case **(Open)**. Suppose we have x, y, P, Q, z and $y\#x$ and $z\#x, y, P, Q$. Suppose $P \stackrel{z.\overline{x}y}{\rightarrow} Q$. By induction hypothesis $P\sigma \stackrel{z.\overline{x\sigma}y}{\rightarrow} Q\sigma$. Also we know that $y\#x\sigma$ and $z\#x\sigma, y\sigma, P\sigma, Q\sigma$, so we can apply **(Open)** to this new transition. ■

In the proof above we deconstruct a transition tr choosing suitably fresh names for the bound variables, and reconstruct it with σ applied, verifying that the appropriate freshness conditions hold. When we deconstruct we took advantage of the FM framework to choose y fresh also for σ, so that $y\sigma = y$ (capture avoidance for free).

Continuing to read [Milner *et al.*, 1992] we see that further results become trivial or easy. Lemma 4 of [Milner *et al.*, 1992] becomes equivariance of $\rightarrow$:

LEMMA 29. *For $\pi : \mathbb{A}_\Pi$, and for P, z, μ, and Q, with $z\#\pi, P$,*

$$P \stackrel{z.\mu}{\rightarrow} Q \implies \pi.P \stackrel{z.(\pi.\mu)}{\rightarrow} \pi.Q.$$

Lemma 5 part (a) and the second half of (b) for bound output of [Milner *et al.*, 1992] disappear because Π is already up to α-equivalence and because bound outputs really do bind the output name. The first half of Lemma 5 part (b) is subsumed in the FM version by Lemma 28.

We give appropriate definitions of bisimulation for the early and late calculi respectively. Early bisimulation in the early transition relation is the greatest symmetric relation closed under the rule

$$P \simeq Q \implies \mathit{M}z.\ \forall \alpha, P'.\ \left(P \stackrel{z.\alpha}{\rightarrow} P' \implies \exists Q'.\ Q \stackrel{z.\alpha}{\rightarrow} Q' \wedge P' \leq Q'\right) \quad (18)$$

Late bisimulation in the late transition system is the greatest symmetric relations such that if $P \simeq Q$ then

$$\begin{array}{rl} \forall z, P'.\ P \stackrel{z.\tau}{\rightarrow} P' & \implies \exists Q'.\ Q \stackrel{z.\tau}{\rightarrow} Q' \wedge P' \simeq Q' \\ \mathit{M}z.\ \forall x, y, P'.\ P \stackrel{z.\overline{x}y}{\rightarrow} P' & \implies \exists Q'.\ Q \stackrel{z.\overline{x}y}{\rightarrow} Q' \wedge P' \simeq Q' \\ \mathit{M}z.\ \forall x, P'.\ P \stackrel{z.xz}{\rightarrow} P' & \implies \exists Q'.\ \forall y.\ Q \stackrel{z.xz}{\rightarrow} Q' \wedge P'[y/z] \simeq Q'[y/z] \end{array} \quad (19)$$

We obtain an early version in the late transition system by swapping $\exists$ and $\forall$ in the last clause to $\forall y.\ \exists Q'$, in the way we would expect.

DEFINITION 30. For $P, Q \in \Pi$, we write $\stackrel{\bullet}{\simeq}$ for the symmetric relation defined by

$$P \stackrel{\bullet}{\simeq} Q \iff \forall \sigma : \mathbb{A}_\Sigma.\ P\sigma \simeq Q\sigma. \quad (20)$$

LEMMA 31. *$\simeq$ is preserved by all term-formers except for input. $\stackrel{\bullet}{\simeq}$ is preserved by all term-formers.*

Proof. By inductions on $\rightarrow$. We have checked the proofs in detail. They are as systematic as the proof of Lemma 28 and by simple induction. ■

3.3 Structural congruence

We can use FM to specify freshness constraints also for structural congruences. For example:

DEFINITION 32. Define a relation $\equiv$ on Π inductively as the smallest congruent equivalence relation closed additionally under the rules

$$\begin{array}{c} \forall P,Q,R.\quad P \mid 0 \equiv P \quad P \mid Q \equiv Q \mid P \quad (P \mid Q) \mid R \equiv P \mid (Q \mid R) \\ \text{И}y.\ \nu y.0 \equiv 0 \qquad \forall P.\ \text{И}y.\ \forall Q.\ P \mid \nu y.Q \equiv \nu y.(P \mid Q) \\ \text{И}y, y'.\ \forall P.\ \nu y \nu y'.P \equiv \nu y' \nu y.P \end{array} \tag{21}$$

We call this **structural congruence**.

(These rules are modified from [Parrow, 2001, Table 2, p.10].)

THEOREM 33. *Structural congruence implies early bisimilarity:* $P \equiv Q \Rightarrow P \simeq Q$.

Proof.(Sketch) By standard FM-style induction. Let $\Phi_\simeq : \mathcal{P}(\Pi \times \Pi) \to \mathcal{P}(\Pi \times \Pi)$ be the monotone operator on relations implicit in (18). Thus a relation R is a bisimulation when $R \subseteq \Phi_\simeq(R)$ and $\simeq \ \stackrel{\text{def}}{=}\ \bigcup \{R \mid R \subseteq \Phi_\simeq(R)\}$ is the largest bisimulation.

Then

- $R \stackrel{\text{def}}{=} \{\langle \nu y.0, 0\rangle\}$ satisfies $\Phi_\simeq(R) \subseteq \equiv$.
- $R \stackrel{\text{def}}{=} \{\langle \nu u.(P \mid Q), P \mid \nu u.Q\rangle \mid u\#P\}$ satisfies $\Phi_\simeq(R) \subseteq \equiv$.
- $R \stackrel{\text{def}}{=} \{\langle \nu y \nu y'.P, \nu y' \nu y.P\rangle \mid y'\#y\}$ satisfies $\Phi_\simeq(R) \subseteq \equiv$.
- The relations

$$R_0 = \{\langle P \mid 0\ ,\ P\rangle\}, \quad R_1 = \{\langle P \mid Q\ ,\ Q \mid P\rangle\}, \quad \text{and} \quad R_2 = \{\langle (P \mid Q) \mid R\ ,\ P \mid (Q \mid R)\rangle\}$$

  satisfy $\Phi_\simeq(R_i) \subseteq \equiv$ for $i = 0, 1, 2$.

Proofs by induction.

It is easy to prove by induction that for all $P, Q : \Pi$ and substitutions σ, if $P \equiv Q$ then $P\sigma \equiv Q\sigma$; И avoids variable capture as in the proof of Lemma 28.

Using all of these facts we can verify that $\equiv \subseteq \Phi_\simeq(\equiv)$ and hence $\equiv \subseteq \simeq$. ■

$$\tau\textbf{:} \qquad \tau.P \overset{\tau}{\to}' P$$

$$\textbf{Out:} \qquad \overline{x}y.P \overset{\overline{x}y}{\to}' P$$

$$\textbf{In:} \qquad xy.P \overset{xn}{\to}' P[n/y]$$

$$\textbf{Par1:} \quad \frac{P_1 \overset{\mu}{\to}' Q_1}{P_1 \mid P_2 \overset{\mu}{\to}' Q_1 \mid P_2}\ bn(\mu) \cap fn(P_2) = \emptyset$$

$$\textbf{Close1:} \quad \frac{P_1 \overset{\overline{x}(y)}{\to}' Q_1, P_2 \overset{xy}{\to}' Q_2}{P_1 \mid P_2 \overset{\tau}{\to}' \nu y.(Q_1 \mid Q_2)}\ y \notin fn(P_2)$$

$$\textbf{Com1:} \quad \frac{P_1 \overset{\overline{x}y}{\to}' Q_1, P_2 \overset{xy}{\to}' Q_2}{P_1 \mid P_2 \overset{\tau}{\to}' Q_1 \mid Q_2}$$

$$\textbf{Open:} \quad \frac{P \overset{\overline{x}y}{\to}' Q}{\nu y.P \overset{\overline{x}(y)}{\to}' Q}\ y \neq x$$

$$\textbf{Res:} \quad \frac{P \overset{\mu}{\to}' Q}{\nu y.P \overset{\mu}{\to}' \nu y.Q}\ y \notin n(\mu)$$

Symmetric versions **(Par2)**, **(Close2)**, and **(Com2)** are elided.

Figure 2. A conventional presentation of early π-calculus transitions.

4 EQUIVALENCE WITH CONVENTIONAL PRESENTATION

It remains to verify that the presentation of Π with its transition system $\to$ in §3 corresponds to the π-calculus as we know it from elsewhere.

Define a datatype of terms of the π-calculus not up to binding

$$\begin{array}{c} \Pi' \stackrel{\text{def}}{=} 1 + \Pi' + \mathbb{A}^2 \times \Pi' + \mathbb{A}^2 \times \Pi' + \Pi'^2 + \mathbb{A} \times \Pi' \\ P ::= 0 \mid \tau.P \mid \overline{x}y.P \mid xy.P \mid (P \mid P) \mid \nu n P \end{array} \tag{22}$$

Define a datatype of actions by

$$Act' \stackrel{\text{def}}{=} 1 + \mathbb{A}^2 + \mathbb{A}^2 + \mathbb{A}^2 \qquad \alpha, \mu ::= \tau \mid \overline{x}y \mid \overline{x}(y) \mid xy \tag{23}$$

DEFINITION 34 (Conventional early transition system for Π'). The transitions of Π' are written $P \overset{\mu}{\to}' Q$ and inductively defined in Figure 2.

Abstraction inductively induces a map $\eta : \Pi' \to \Pi$:

$$\eta 0 = 0 \quad \eta(\tau.P') = \tau.\eta P' \quad \eta(\overline{x}y.P') = \overline{x}y.\eta P' \quad \eta(xn.P') = x[n]\eta P' \quad \eta(P'_1 \mid P'_2) = \eta P'_1 \mid \eta P'_2 \quad \eta(\nu n.P') = \nu n.\eta P'. \tag{24}$$

LEMMA 35. *For $P' : \Pi'$, $y \notin fn(P') \iff y\#\eta(P')$.*

Proof. By induction on syntax for fixed y using inductive hypothesis

$$\phi(P') \stackrel{\text{def}}{\Leftrightarrow} y \notin fn(P') \leftrightarrow y\#\eta(P').$$

We consider just the case of restriction. Suppose $\phi(P')$. We verify $\phi(\nu n.P')$.

If $n \neq y$ then $y \notin fn(P')$ precisely when $y \notin fn(\nu n.P')$. Also, $y\#P'$ precisely when $y\#\nu[n]\eta(P')$. Since $\nu[n]\eta(P') = \eta(\nu n.P')$ the result follows.

If $n = y$ then $y \notin fn(\nu y.P')$ and also $y\#\nu[y]\eta(P')$. Since $\nu[n]\eta(P') = \eta(\nu n.P')$ the result follows. ■

LEMMA 36. *$\eta(-)$ is surjective. Its kernel is precisely α-equivalence on Π'.*

Proof. By inductions on syntax. ■

η is also Barendregt abstractive, by Corollary 18.

In the last section we mentioned that Π is an FM datatype and has no names of bound atoms. η and N are a formal sense in which we can give them names.

η extends pointwise to a map $\mathcal{P}(\Pi' \times \Pi') \to \mathcal{P}(\Pi \times \Pi)$ and hence to a map from a relation R on Π' to a relation, write it ηR, on Π. It also gives rise to a map η^{-1} from $\mathcal{P}(\Pi \times \Pi)$ to $\mathcal{P}(\Pi' \times \Pi')$, R maps to $\{\langle P', Q' \rangle \mid \eta P' \; R \; \eta Q'\}$.

THEOREM 37. *$\eta{\simeq} = {\simeq}$, where $\simeq$ is early bisimilarity or late bisimilarily on Π' and Π respectively.*

Proof. We show that $\eta{\simeq}$ is a bisimulation in the FM sense, and that $\eta^{-1}{\simeq}$ is a bisimulation in the traditional sense, for a notion of bisimulation as in (18) and (19). ■

THEOREM 38. *The transition relations $\to \subseteq \Pi \times \Pi$ and $\to' \subseteq \Pi' \times \Pi'$ are related as shown in Figure 3.*

Note that (26) refers only to Barendregt representatives. This is technically convenient for the proof. Once we have it we can take advantage of results on transitions for the standard name-carrying Π', such as [Milner *et al.*, 1992, Lemma 5], to strengthen it to all $P', Q' : \Pi'$.

The proof of Theorem 38 is in two parts corresponding to (25) and (26). The first part is by induction on $\to'$, the second by induction on $\to$. To save space we do not re-write out the induction hypotheses in complete formality.

For any $P', Q' : \Pi', x, y : \mathbb{A}$, write $P = \eta P', Q = \eta Q'$. Then

$$\begin{array}{rcll} P' \overset{\tau}{\rightarrow}{}' Q' & \Longrightarrow & P \overset{\tau}{\rightarrow} Q & \wedge \\ P' \overset{xy}{\rightarrow}{}' Q' & \Longrightarrow & P \overset{xy}{\rightarrow} Q & \wedge \\ P' \overset{\overline{x}y}{\rightarrow}{}' Q' & \Longrightarrow & P \overset{\overline{x}y}{\rightarrow} Q & \wedge \\ P' \overset{\overline{x}(y)}{\rightarrow}{}' Q' & \Longrightarrow & P \overset{y.\overline{x}y}{\rightarrow} Q. & \end{array} \tag{25}$$

For any $P, Q : \Pi, x, y, : \mathbb{A}$, choosing fresh names for the bound atoms in P and Q using $\mathsf{И}$;

$$\begin{array}{l} \mathsf{И}P = \eta P', Q = \eta Q'. \\ \qquad \begin{array}{rcll} P \overset{\tau}{\rightarrow} Q & \Longrightarrow & P' \overset{\tau}{\rightarrow}{}' Q' & \wedge \\ P \overset{xy}{\rightarrow} Q & \Longrightarrow & P' \overset{xy}{\rightarrow}{}' Q' & \wedge \\ P \overset{\overline{x}y}{\rightarrow} Q & \Longrightarrow & P' \overset{\overline{x}y}{\rightarrow}{}' Q' & \wedge \\ P \overset{y.\overline{x}y}{\rightarrow} Q & \Longrightarrow & P' \overset{\overline{x}(y)}{\rightarrow}{}' Q'. & \end{array} \end{array} \tag{26}$$

Figure 3. Equivalence of conventional and FM presentations.

Proof.(Of Theorem 38, first part, (25))
The case (In) of $\rightarrow'$. Suppose $xy.P \overset{xn}{\rightarrow'} P[n/y]$. We would like to apply **(In)** of $\rightarrow$ to $x, n, y, \eta(P)$, and some $z\#x, n, y, \eta(P)$. However to do so we must verify $y\#x, n$, and this does not hold. We draw on the library of results for the conventional π-calculus to deduce that for $y'\#x, n$, $xy'.P[y'/y] \overset{xn}{\rightarrow'} P[y'/y][n/y']$. $xy'.P[y'/y]$ is α-equivalent to $xy.P$ so both map to $x[y]\eta(P)$. We can now proceed.

The case (Par1) of $\rightarrow'$. We consider only the case $\mu = \overline{x}(y)$. Suppose $P_1 \overset{\overline{x}(y)}{\rightarrow'} Q_1$. Suppose also $y \notin fn(P_2)$ for some P_2. Then $P_1 \mid P_2 \overset{\overline{x}(y)}{\rightarrow'} Q_1 \mid P_2$. By the inductive hypothesis $\eta(P_1) \overset{y.\overline{x}y}{\rightarrow} \eta(Q_1)$. It remains to apply **(Par1)** to deduce $\eta(P_1) \mid \eta(P_2) \overset{y.\overline{x}y}{\rightarrow} \eta(Q_1) \mid \eta(P_2)$. We must verify $y\#\eta(P_1), \eta(P_2)$. We know $y\#\eta(P_2)$ since $y \notin fn(P_2)$. We know $y\#\eta(P_1)$ because $y \notin fn(P_1)$, this is inductively guaranteed in the conventional theory by the assumption that P_1 can output y as a bound name.

The case (Open) of $\rightarrow'$. Suppose $P \overset{\overline{x}y}{\rightarrow'} Q$ and $y \neq x$. By inductive hypothesis we have $\eta(P) \overset{z.\overline{x}y}{\rightarrow} \eta(Q)$ for $z\#P, x, y, Q$. We deduce $\nu y.P \overset{\overline{x}(y)}{\rightarrow'} Q$. We wish to apply **(Open)** of $\rightarrow$ to $x, y, \eta(P), \eta(Q), z$ to deduce $\nu[y]\eta(P) \overset{y.\overline{x}y}{\rightarrow} \eta(Q)$. The freshness condition $y\#x$ reduces to $y \neq x$, which is just the side-condition. ■

Proof.(Of Theorem 38, second part, (26))
The case (In) of $\rightarrow$. Suppose x, n, y, P, z are such that $y\#x, n$ and $z\#x, n, y, P$. We apply **(In)** to deduce $x[y]P \overset{z.xn}{\rightarrow} P[n/y]$. Now choose distinct names fresh for x, n, y, P, z and for the bound atoms in P, to obtain a Barendregt representative $P' : \Pi'$ of P, so $\eta(P') = P$. We wish to apply **(In)** of $\rightarrow'$ to x, y, P', n. We can do this, there are no side conditions.

The remaining issue is whether, for example, $\eta(xy.P') = x[y]P$ and $\eta(P'[n/y]) = P[n/y]$. These follow from appropriate freshness hypotheses. For example, the names for bound variables in P' were chosen fresh for a context which included n and y, so accidental capture is impossible.

The case (Par1) of $\rightarrow$. We consider only the case of a bound output. Suppose we have P_1, P_2, y, Q_1, Q_2 with $y\#P_1, P_2$, and $P_1 \overset{y.\overline{x}y}{\rightarrow} Q_1$. By inductive hypothesis for fresh choices of bound names $\eta(P_1') = P_1, \eta(P_2') = P_2, \eta(Q_1') = Q_1$ we have $P_1' \overset{\overline{x}(y)}{\rightarrow'} P_2'$. The side-condition $y \notin fn(P_2')$ follows because $fn(P_2') = S(P_2)$ and $y\#P_2$, so we may apply **(Par1)** of $\rightarrow'$.

The case (Open) of $\rightarrow$. Suppose x, y, P, Q, z have $y\#x$ and $z\#x, y, P, Q$, and suppose $P \overset{z.\overline{x}y}{\rightarrow} Q$. By inductive hypothesis for fresh choices of bound names $\eta(P') = P, \eta(Q') = Q$ we have $P' \overset{\overline{x}y}{\rightarrow'} Q'$. The side-condition $y \neq x$

follows from $y\#x$. ■

5 CONCLUSIONS

FM techniques were developed to inductively represent syntax-with-binding, so that we can reason and program on it structurally. These ideas have been considered mathematically and logically [Gabbay and Pitts, 2001; Urban *et al.*, 2003; Pitts, 2001] and also implemented as a dialect of ML [FreshML, 2003; Pitts and Gabbay, 2000; Shinwell *et al.*, 2003]—yet until now FM literature has for simplicity only really considered the syntax of elementary λ-calculi.

The π-calculus is a natural choice for a more complex study. Its binding is more complicated and includes scope extrusion, name passing, and name-generation (also called dynamic allocation).

We have seen how standard structural specifications of transition systems translate quite directly to FM, with the equivalence being in essence just the quotient map by binding which we would hope for. We have seen how side-conditions on free and bound names are taken care of by judiciously mixing $\forall$ and $\mathsf{И}$ quantifiers.

In the course of this research we discovered the notion of an abstractive function and the generalised $\mathsf{И}$ quantifier described in §2.4. We needed it first to express Theorem 38, but then realising its generality we devoted some space to developing it.

In the syntax-free mathematical context of FM sets, abstractive functions and generalised $\mathsf{И}$ provide a unifying account of (syntax-free mathematical generalisations of) the Barendregt variable naming convention, FreshML pattern-matching, and the $\mathsf{И}$ quantifier.

Related work. FM techniques are related to presheaf semantics for syntax, see for example [Fiore *et al.*, 1999; Fiore and Turi, 2001]. The presheaves used there are $Set^{\mathbb{I}}$ and $Set^{\mathbb{F}}$ ($\mathbb{I}$ the category of finite sets and injections, $\mathbb{F}$ that of finite sets and all maps).

Presheaves can be viewed as sets with 'actions' at 'stages', so the two presheaf categories above make injective and possibly non-injective renamings respectively intensional features of the underlying universe. An FM set X gives a presheaf mapping $S \in \mathcal{P}_{fin}(\mathbb{A})$ to $\{x \in X \mid S(x) \subseteq S\}$. It is a special one though, such that every element appears at a unique minimal stage: its support, this is Theorem 7. In this way FM maintains a 'first-order set-based' presentation which we feel is particularly clear.

Approaches based on Higher-Order Abstract Syntax (HOAS) can suffer the following three problems. First, the function spaces can be too large. For this reason the Theory of Contexts has no ι unique choice [Bucalo *et al.*, 2001]. Second, function spaces can destroy inductive structure, so there can be issues with deriving inductive principles for the datatypes. Thirdly,

these problems overcome, if we need to generate names we may need to index all our predicates and relations by an explicit set of known names (called X or A, say) see for example [Bruni *et al.*, 2002; Honsell *et al.*, 2001, Fig.2] and [Cattani and Sewell, 2000, Fig.1]. This is a programming analogue of the the presheaf semantics mentioned above. Other more first-order set-based presentations may do this too, for example [Cattani and Sewell, 2000]. These indexes can be cleanly presented, but it would still be better if they were not there at all.

Two advantages of HOAS are that they can be implemented in existing frameworks (wheras FM based on FM sets, cannot), and that name-passing can be modelled by function application. Name-passing in this paper is modelled by substitution defined inductively on syntax in (13).

We already discussed de Bruijn approaches in the Introduction. Their 'twisted' inductive principles can cause great technical difficulties. In studies by Hirschkoff [Hirschkoff, 1999; Hirschkoff, 1997] 75 per cent of the technical lemmas of a full formalisation of the π-calculus regarded 'straightening them out'.

Of all approaches to syntax, surely the benchmark is name-carrying syntax trees like the one in (3). The problems with capture-avoidance for such datatypes are well-known. This paper shows how FM approach treats names in the π-calculus in a way which is close to this elementary practice, but entirely rigorous. This will hopefully serve as a model and theoretical foundation for more complex examples.

We should mention that a lot of the problems with name-carrying syntax only appear if we insist on a purely structural approach. In [Martin and Gordon, 1993] pages 24 and 25 the authors are primarily interested in expressivity (of the spi-calculus). Accordingly they adopt a 'chemical' presentation after [Berry and Boudol, 1990]; transitions are defined on processes modulo scope extrusion and other structural congruence rules.

Even here, FM can help. We saw in §3.3 how it takes care of side-conditions for freshness in scope extrusion. Of course this is a simple observation, but for more complex calculi with various forms of binding, it can be useful both for presentation and theory.

Future work. The models of this paper are purely syntactic, and not semantic. It is a feature of FM abstraction that we may only choose a *fresh* name for the bound name. Wheras in HOAS abstraction is function abstraction, and name-passing can therefore be function application, here we use the renaming (inductively!) defined in Definition 23.

There is nothing wrong with this, but it does block an obvious abstract account of name-passing based on FM abstractions instead of functions. Other semantic models based on presheaves, such as [Fiore *et al.*, 1996; Cattani *et al.*, 1997], also use function spaces. [Fiore and Turi, 2001] makes the same observation of models of syntax in $Set^{\mathbb{I}}$ and goes on to provide a comprehensive categorical framework to account for name and value passing.

It might be interesting to carry out a similar programme using in an FM style. Even if this turns out to be equivalent to the presheaf material, which we doubt because our account would have a notion of support, it would be a different perspective and presentation.

We see the most immediate and concrete application for this work to History-Dependent (HD) Automata [Montanari and Pistore, 1997], which are a notion of model based only on syntax. An HD automaton is roughly speaking a graph of representative processes encoded in de Bruijn notation, with edges transitions annotated by permutations between the names to take care of reindexing and name-generation (the many flavours of HD automaton corresponding to different and differently-detailed implementations of this idea). This could be viewed as an FM construction and this paper is a first step towards constructing it.

A related approach to the same issue is θ-automata described in [Bruni *et al.*, 2002]. Roughly, these models are obtained by adding a meta-level restriction operator θ to the syntax of terms, which acts only at top level and restricts certain names in a process. Name-generation is modelled by generating a name, then restricting it. This concept is clearly related to the datatype $\rightarrow \subseteq \Pi \times [\mathbb{A}](Act \times \Pi)$ developed in this paper, with restriction replaced by FM abstraction. The concept of model in that work corresponds to that of an HD automaton, only implemented using terms up to α-equivalence (bound by θ) instead of a de Bruijn representation.

We plan to develop an FM based structure unifying these two approaches. Since FM now comes equipped with a programming language FreshML [FreshML, 2003], datatypes and inductive definitions should translate into programs.

ACKNOWLEDGEMENTS

My grateful thanks go to Marino Miculan, Furio Honsell, and Ugo Montanari. I also recognise the support of UK EPSRC grant GR/R07615 which partly funded this research.

BIBLIOGRAPHY

[Berry and Boudol, 1990] Gerard Berry and Gerard Boudol. The chemical abstract machine. In *Proceedings of the seventeenth annual ACM symposium on Principles of programming languages*, pages 81–94, New York, NY, USA, 1990. ACM Press.

[Bruni *et al.*, 2002] R. Bruni, F. Honsell, M. Lenisa, and M. Miculan. Modeling fresh names in pi-calculus using abstractions. Technical Report 21/2002, Dipartimento di Matematica e Informatica, University of Udine (Italy), April 2002.

[Bucalo *et al.*, 2001] A. Bucalo, M. Hofmann, F. Honsell, M. Miculan, and I. Scagnetto. Consistency of the theory of contexts, 2001.

[Cattani and Sewell, 2000] Gian Luca Cattani and Peter Sewell. Models for name-passing processes: Interleaving and causal. In *Logic in Computer Science*, pages 322–332, 2000.

[Cattani *et al.*, 1997] Gian Luca Cattani, Ian Stark, and Glynn Winskel. Presheaf models for the pi-calculus. In E[ugenio] Moggi and G[iuseppe] Rosolini, editors, *Proceedings of the 7th International Conference on Category Theory and Computer Science (Santa Margherita Ligure, Italy, September 4–6, 1997)*, volume 1290 of *LNCS*, pages 106–126. Springer, 1997.

[Fiore and Turi, 2001] Marcelo Fiore and Daniele Turi. Semantics of name and value passing. In *Proc. 16$^{\text{th}}$ LICS Conf.*, pages 93–104. IEEE, Computer Society Press, 2001.

[Fiore *et al.*, 1996] Marcelo Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully-abstract model for the π-calculus (extended abstract). In *Eleventh Annual Symposium on Logic in Computer Science (LICS) (New Brunswick, New Jersey)*, pages 43–54. IEEE, Computer Society Press, July 1996.

[Fiore *et al.*, 1999] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax and variable binding. In *14th Annual Symposium on Logic in Computer Science*, pages 193–202. IEEE Computer Society Press, Washington, 1999.

[FreshML, 2003] FreshML. FreshML homepage, 2003. `http://www.freshml.org`.

[Gabbay and Pitts, 2001] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[Gabbay, 2000] Murdoch J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, Cambridge, UK, 2000.

[Hirschkoff, 1997] Daniel Hirschkoff. A full formalization of pi-calculus theory in the Calculus of Constructions. In E. Gunter and A. Felty, editors, *Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'97)*, pages 153–169, Murray Hill, New Jersey, August 1997.

[Hirschkoff, 1999] Daniel Hirschkoff. *Mis en oeuvre de preuves de bisimulation*. PhD thesis, École Nationale des Ponts et des Chausées (ENPC), January 1999. In French.

[Honsell *et al.*, 2001] Furio Honsell, Marino Miculan, and Ivan Scagnetto. π-calculus in (co)inductive type theory. *Theoretical Computer Science*, 253(2):239–285, 2001.

[Martin and Gordon, 1993] M. Martin and D. Gordon. A calculus for cryptographic protocols : the spi calculus. Technical Report SRC-RR-149, Internationales Begegnungs und Forschungszentrum Schloss Dagstuhl, Germany, 1993.

[Milner *et al.*, 1992] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, September 1992.

[Montanari and Pistore, 1997] Ugo Montanari and Marco Pistore. An introduction to history dependent automata. In Andrew Gordon, Andrew Pitts, and Carolyn Talcott, editors, *Conference Record of the Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II, Stanford University, December 8–12, 1997)*, volume 10 of *ENTCS*. Elsevier Science Publishers, 1997.

[Parrow, 2001] Joachim Parrow. An introduction to the pi-calculus. In Jan Bergstra, Alban Ponse, and Scott Smolka, editors, *Handbook of Process Algebra*, pages 479–543. Elsevier Science, 2001.

[Pitts and Gabbay, 2000] A. M. Pitts and M. J. Gabbay. A metalanguage for programming with bound names modulo renaming. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction. 5th International Conference, MPC2000, Ponte de Lima, Portugal, July 2000. Proceedings*, volume 1837 of *Lecture Notes in Computer Science*, pages 230–255. Springer-Verlag, Heidelberg, 2000.

[Pitts, 2001] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 2001. To appear. (A preliminary version appeared in the *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software* (TACS 2001), LNCS 2215, Springer-Verlag, 2001, pp 219–242.).

[Shinwell *et al.*, 2003] M. R. Shinwell, A. M. Pitts, and M. J. Gabbay. FreshML: Programming with binders made simple. In *Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), Uppsala, Sweden*. ACM Press, August 2003.

[Urban *et al.*, 2003] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. In M. Baaz, editor, *Computer Science Logic and 8th Kurt Gödel Colloquium (CSL'03 & KGC), Vienna, Austria. Proccedings*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2003.