FM-HOL, A HIGHER-ORDER THEORY OF NAMES

MURDOCH J. GABBAY

Abstract. FM (Fraenkel-Mostowski) set theory techniques were developed to give good support to inductive reasoning on formal syntax in the presence of α-equivalence and variable binding. The original set theory has inspired a Higher-Order Logic (HOL) theory, FM-HOL, presented in this paper. FM-HOL has similar facilities for handling syntax-with-binding to the original set theory, but is mathematically more powerful, introduces several novel features, and is much better suited to machine automation. This paper concentrates on the mathematical aspect of FM-HOL, presenting it as an improved foundation in which to analyse syntax-with-binding.

1. Introduction

This paper is about the the problem of variable binding and α-equivalence. Consider, for example, unwanted variable capture in the λ-calculus: we do not expect $(\lambda x.\lambda y.x)\,y$ to evaluate to $\lambda y.y$, but to $\lambda y'.y$ for some 'fresh', 'local' y'. The underlying function is **capture-avoiding substitution** $[y/x]\lambda x.y$ which is expected to rename y to y'. Old versions of LISP displayed this phenomenon, when LISP macros with local variables can see them accidentally captured by the global context into which they were expanded.

So what is happening mathematically, and how are we to specify elementary functions like substitution and α-equivalence? One way is to take the type of variable names to be the natural numbers $\mathbb{N}$, and choose increasingly larger 'fresh' numbers as names. For example, we could define a type of terms for the λ-calculus by

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

and α-equivalence $=_\alpha$ on Λ by

$$\begin{array}{rll} t =_\alpha t' \stackrel{\text{def}}{=} & n = n' & \leftarrow t = \mathbf{Var}(n), t' = \mathbf{Var}(n') \\ & s_1 =_\alpha s_1', s_2 =_\alpha s_2' & \leftarrow t = \mathbf{App}(s_1, s_2), t' = \mathbf{App}(s_1', s_2') \\ & [u/n]s =_\alpha [u/n']s' & \leftarrow t = \mathbf{Lam}(n, s), t' = \mathbf{Lam}(n', s'), u > t, t' \end{array} \tag{2}$$

where '$u > t, t'$' denotes some u which is larger than any v occuring in t and t' and hence does not occur in t.

Remark 1.1. But there are problems. For example do we mean "*for all such* u" or "*for one such* u". The former choice is good for doing induction going down (where we want a strong hypothesis), the latter for doing induction going up (where we want a weak proof-obligation). There seems to be no happy compromise (see R5.6).

Here, at least, we can get away with $[u/n]s$ denoting simple textual substitution, because we have assumed u completely fresh for t, t'. Later on, when we define evaluation relations, we may have to consider full capture-substitution. The technical details of choosing fresh variable names only causes further problems. ◇

Remark 1.2 (Existing accounts)**.** All analyses agree in taking Λ of the form

$$\Lambda \stackrel{\text{def}}{=} \mathbf{Var\ of\ varset} + \mathbf{App\ of}\ \Lambda \times \Lambda + \mathbf{Lam\ of}\ (\mathbf{abstractions\ of}\ (\Lambda)). \tag{3}$$

Date: March 15, 2002.

Many thanks to Andrew Pitts and Michael Norrish for their comments. Thanks also to UK EPSRC grant GR/R07615 which partly funded this research.

De Bruijn indices typically model **varset** by $\mathbb{N}$ and **abstractions of** (Λ) by Λ itself. This works because we so to speak 'twist' the meaning of the variable symbols under the binder **Lam** so that 0 refers to the abstracted variable symbol, 1 refers to variable 0, 2 to 1, and so on. This leads to rather 'twisted' inductive definitions out of this datatype, as we inductively 'untwist' going under abstractions (incrementing and decrementing variable numbers according as we add or remove abstractions from above them, at a likely cost of $O(n)$ in the size of the term).

HOAS models abstractions either as **varset** $\to \Lambda$ or $\Lambda \to \Lambda$, where **varset** may be $\mathbb{N}$, or may be a new type. The technique tends to suffer technical difficulties, for further discussion see [3, paragraph 33.2].

There are many other approaches ranging from category theoretic constructions which work by moving into a more-or-less exotic foundational system (for just one example see [6]), to work by Pollack and others [9]. ◇

Remark 1.3 (FM and the literature)**.** FM is a new approach with a pleasantly elementary feel, and which seems close to informal practice. FM is not a particular theory, but a collection of principles which we try to realise in the context of a set theory, programming language, etc.

The original FM theory, which motivates most of the principles and definitions in this paper, was FM sets. FM stands for "Fraenkel-Mostowski", who invented permutation models of set theory to prove the independence of the axiom of choice, see [14]. Sixty years later Pitts and I hijacked the idea of set-theoretic atoms to be variable names (**varset** in (3)), and the Fraenkel-Mostowski atom-permutation on the set universe to rename variable names on sets denoting terms of syntax ($[u/n]s$ in (2)). See $\mathbb{A}$ and $\pi.x$ in R3.3, and §2.

Existing literature on FM techniques includes [3] (my thesis), [1] and [2] (twin papers which first presented FM set theories in refereed publications), [11] (a first-order logic by Pitts with elements of FM), and [12] (on a version of ML extended with constructs inspired by FM for manipulating syntax up to α-equivalence). The literature is growing. Besides this paper, [5] (a discussion of an implementation and testing of FM techniques in the machine environment Isabelle) has been submitted for publication. Work proceeds on the programming language and also principles of unification for logics with formulae describing binding, and tutorial papers are in preparation. ◇

Remark 1.4 (Contribution of this paper)**.** In this paper I discuss a recently-invented theory FM-HOL, a Higher-Order Logic (HOL) with elements of FM. It carries out a similar programme in HOL to that of the set theories [1] and [2]. But I see this paper as more than a change of foundations. We significantly extend FM with many novel definitions. For example the treatment of small sets of atoms in D4.2, the methods of construction of abstraction sets in §6, and the concept of 'nameable atoms' introduced in R7.8 and discussed in the surrounding text.

FM-HOL is also designed to be far better for computer implementation than FM sets. I argue the case for this in a description of an FM-HOL implementation in Isabelle [5]. I believe that this too is a very significant plus for FM-HOL and I refer the interested reader to that paper. ◇

2. Overview

Notation 2.1 (Types)**.** We write 't of type α' as t^α or $t : \alpha$. FM-HOL has no polymorphic types but we will want to refer to type-indexed constants such as $\lambda x^\alpha.\mathbf{Inl}(x) : Varies(\alpha)$. We call this a type-scheme and write it $\bigwedge \alpha.\ Varies(\alpha)$ or just $Varies(\alpha)$.

FM theories have an equality **type of atoms** $\mathbb{A}$ along with an action of permutations π of $\mathbb{A}$ at all types, written $\pi.x$ for some x^σ, see D3.1. $\mathbb{A}$ is the **varset** of (3), its elements are intended to represent variable symbols of an object language whose syntax is constructed

(as an inductive datatype) in FM-HOL. Thus the Λ of (1) can be implemented in FM as

$$\Lambda \stackrel{\text{def}}{=} \textbf{Var of } \mathbb{A} + \textbf{App of } \Lambda \times \Lambda + \textbf{Lam of } \mathbb{A} \times \Lambda. \tag{4}$$

The significance of the permutation action is a little more subtle. Consider a transposition $(a\ b)$ for $a, b : \mathbb{A}$ and its action on $\mathbf{Lam}f.\mathbf{Lam}x.f(x)$ in the Λ of (4). We shall prefer the informal λ notation $\lambda f.\lambda x.f(x)$ for readability. Observe that

$$(f\ x).\lambda f.\lambda x.f(x) = \lambda x.\lambda f.x(f) =_\alpha \lambda f.\lambda x.f(x). \tag{5}$$

Thus transposition provides a notion of **atom-swapping** which by being an isomorphism on $\mathbb{A}$ avoids some of the problems of $[f/x]\lambda f.\lambda x.f(x)$ when $[f/x]$ identifies f and x to $\lambda f.\lambda f.f(f)$. There is no loss in expressive power, e.g. in the third line of (2), $(u\ n)$ and $(u\ n')$ will do just as well as $[u/n]$ and $[u/n']$. There is a good case that atom-permutation is more elementary than capture-avoiding atom-substitution, and mathematically better-behaved than naïve atom-substitution. Cf. [2, Chapter 2].

Using this dedicated type of atoms $\mathbb{A}$ and its permutation action at all types, we find in §6 that we can define a type $\mathbb{A}.\alpha$ of **abstractions of** α of "elements x^α with one distinguished bound atom a". This plays the rôle of (**abstractions of** (Λ)) in (3) and allows us to define an inductive datatype of syntax-with-binding, e.g. an inductively defined type of λ-terms up to α-equivalence (see (26), (30)). The motivation for using $\mathbb{A}.\alpha$ is that it seems better-behaved than many of the alternatives, for example $\mathbb{A} \to \alpha$ in HOAS or α itself ('twisted', as discussed above) in the de Bruijn indeces approach.

Furthermore, $\mathbb{A}.\alpha$ comes with well-behaved logical and functional tools for reasoning and programming with types of syntax-with-binding, e.g. **the new quantifier** $\mathsf{И}$ whose excellent properties are described in §5. To my knowledge these tools are unique to the FM approach.

We finish the mathematical development in §7 with a discussion of axioms of definite and indefinite choice in HOL with FM, which are in some sense final elements in the development of a fully-fledged, implementable, theory.

For the reader unfamiliar with FM, this paper presents FM-HOL as a workable environment for manipulating syntax-with-binding. For the FM practitioner, the paper contains a new FM theory, as well as various new definitions and principles of proof which I have found useful and wish to showcase.

3. HOL$_{+\mathbf{lift}}$

Definition 3.1 (HOL$_{+\mathbf{lift}}$)**.** HOL$_{+\mathbf{lift}}$ is a classical higher-order logic with a type of atoms $\mathbb{A}$ and a (polymorphic) constant $\mathbf{lift} : \mathbb{A}^{Perm} \to \alpha \to \alpha$ (where $\mathbb{A}^{Perm}$ denotes the type of permutations of $\mathbb{A}$):

($\mathbb{A}$) $\quad \mathbb{A}$ **is a type**

(**lift**) $\quad \mathbf{lift} : \mathbb{A}^{Perm} \to \alpha \to \alpha$

For brevity write '$\mathbf{lift}_\alpha \pi x$' as '$\pi.x$'. Using the notation of N3.2, $\mathbb{A}$ and **lift** satisfy the following axioms:

(L$\circ$) $\quad \pi.\pi'.x = (\pi \circ \pi').x$

(L$\mathbb{A}$) $\quad \pi.a = \pi(a^{\mathbb{A}})$

(LE) $\quad (t\,\vec{x}) \;=\; \pi.\big(t\ (\pi^{-1}.\vec{x})\big)$ $\quad$ **when** t ***closed.***

Notation 3.2. A finite list $x_i, i = 1, \ldots, n$ is a ***vector*** written $\vec{x}$ or $(x_i)_i$. For $\pi \in \mathbb{A}^{Perm}$, $\pi.\vec{x}$ denotes π applied pointwise on each element of $\vec{x}$.

$\pi_1 \circ \pi_2$ denotes composition of permutations in $\mathbb{A}^{Perm}$.

t^α is ***closed*** when as syntax it contains no free variables. Thus $\top^{\mathbb{B}}$ and $\lambda x^{\mathbb{A}}.x$ are closed but $(\lambda x^{\mathbb{B}}.\top)(y^{\mathbb{B}})$ is not (though it β-reduces to $\top$). This is a property of syntax and not any underlying denotation.

It is convenient to write **Id** for the identity function $\lambda x:\alpha.x$. We use this notation frequently to denote the **Id** on $\mathbb{A}$, which is in $\mathbb{A}^{Perm}$ (e.g. R3.4).

Remark 3.3. So $\mathrm{HOL}_{+\mathbf{lift}}$ is a HOL with a distinguished type of atoms $\mathbb{A}$ and an atom-permutation action $\pi.x$ at all types (given by **lift**), just as discussed in R1.3 and §2. (L$\circ$) is a standard property of a 'permutation action', (L$\mathbb{A}$) says it acts at $\mathbb{A}$ in the reasonable way. Further standard properties of a permutation action, e.g. $\mathbf{Id}.x = x$ (N3.2), are considered in L3.5. ◇

Remark 3.4 (Equivariance)**.** (LE) is the interesting axiom. In general mathematical terms, the standard permutation action on an n-ary function f is $\pi.f = \lambda\vec{x}.\pi.f(\pi^{-1}.\vec{x})$ (cf. Item 4 of L3.5). (LE) can be read as asserting that $\mathbb{A}$-permutation has no action on functions $f = t$ specified by a *closed* term t and that in this sense the meta-language is blind to the identities of individual atoms.

In the light of Item 1 of L3.5 below, we now know—immediately, without any induction—that for example in (4),

$$(a\ b).\mathbf{App}(s_1, s_2) = \mathbf{App}((a\ b).s_1, (a\ b).s_2).$$

Such equalities are very useful, as is the fact that they are immediate and do not require inductive proof: the inductive action we see in equalities such as

$$[b/a]\mathbf{App}(s_1, s_2) = \mathbf{App}([b/a]s_1, [b/a]s_2)$$

are given to us 'for free' by the theory by virtue of its equivariance.

◇

Another significant feature of this theory is the fact that the permutation action is defined at all types, not just by induction on types of syntax. This allows us to permute atoms in elements that are not obviously syntactic. For example, we can rename atoms in elements of types of abstractions $\mathbb{A}.\alpha$ (first mentioned in §6). Just to look at the definition of elements of abstraction type, (21) and (22) in D6.1, it is by no means clear that $a.s$ is a syntactic entity in the same way as, say, $<s_1, s_2>$ (assuming s, s_1, s_2 are). Nevertheless, we can rename the atoms inside $a.x : \mathbb{A}.\alpha$ using the permutation action. And indeed, we *need* the permutation action in the proofs of §6 to prove the necessary results which show that $a.x$ *is* 'x with a bound', see R6.7.

Michael Norrish has suggested an alternative formulation of (LE). I do not discuss it here but it is important. See [5, {5.15}] and future publications.

Lemma 3.5 (Technical results)**.** *Various basic properties of the permutation action follow from D3.1:*

1. *Permutation distributes over meta-language functions:* $\pi.t(x_1, \ldots, x_n) = t(\pi.x_1, \ldots, \pi.x_n)$*, for t a closed term. Note in the case $n = 0$ we have $\pi.t = t$: the meta-language may not contain object-level variable symbols.*
2. $\pi.\pi^{-1}.x = x$ *(and* $\pi^{-1}.\pi.x = x$*).*
3. $\mathbf{Id}.x = x$*.*
4. *Permutations acts on functions in the standard way:* $\pi.(f\,x) = (\pi.f\,\pi.x)$*, and* $\pi.f = \lambda x.\pi.(f\,(\pi^{-1}.x))$*.*

Proof.
1. Direct from (LE), applying π to both sides.
2. From (LE) taking $t = \lambda x.x$.
3. Combining Item 2 with (L$\circ$). For **Id** see N3.2.
4. Again from (LE) taking $t = \lambda f, x.(f\,x)$ and using Item 2 to simplify.

□

We come to the second step of constructing our theory:

4. FM-HOL$_{-\iota}$

Notation 4.1 (Predicates and HOL sets)**.** We may call $P : \alpha \to \mathbb{B}$ a function into $\mathbb{B}$ a ***predicate***, or also a ***(HOL)-set***. Sets are usually written $X, Y, Z, \ldots : \alpha \to \mathbb{B}$ with

elements $x, y, z, \ldots : \alpha$. We borrow set-theoretic notation. E.g. for a HOL-set X, $x \in X$ denotes $(X\, x)$. We may write $\emptyset$ or $\bot$ for the 'empty' set $\lambda x^\alpha.\bot$, and α or $\top$ for $\lambda x^\alpha.\top$. We borrow other notation as convenient, e.g. $X \subseteq Y$.

Definition 4.2 (Small)**.** Define a predicate $\mathsf{S} : \mathcal{P}(\mathbb{A}) \to \mathbb{B}$ given by

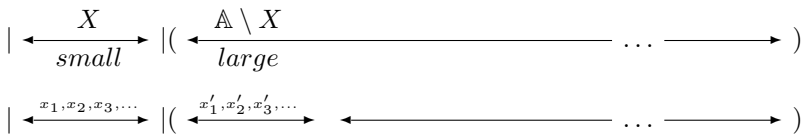
$$X \in \mathsf{S} \stackrel{\mathbf{def}}{\iff} (X \textbf{ well-orderable}) \wedge (\mathbb{A} \setminus X \textbf{ \textit{not} well-orderable}) \wedge (X \hookrightarrow \mathbb{A} \setminus X). \tag{6}$$

($X \hookrightarrow Y$ means "there is an injection from X to Y".)

We call X **small** when $X \in \mathsf{S}$ and **large** when $\mathbb{A} \setminus X \in \mathsf{S}$.

A picture might illustrate the point:

(7)



S is a theory of small sets of atoms (object-level variable names). X is 'small' when we can enumerate it, when $\mathbb{A} \setminus X$ is *not* enumerable, and also large enough to contain copies of X itself.

Remark 4.3 (Small=well-orderable)**.** A reader familiar with FM, perhaps from any one of [1], [2], or [4], will recognise this as a generalisation of the situation in FM set theory, where 'small'='finite'. Here the slogan is **'small'='well-orderable'**.

This principle, and the details of the definition of smallness (6), give the notion of smallness excellent properties which make possible the development in the rest of this paper. D4.2 is one of the contributions of this paper to FM theory as a whole. ◇

Lemma 4.4 (Closure properties)**.** S *is closed under finite unions and finite (nonempty) intersections. In fact,* S *is closed under unions indexed by small* $I \in \mathsf{S}$*. Proof from the properties of well-orderings. The slogan is* **"small unions of small sets are small"***.*

Definition 4.5 (Support)**.** For $X \subseteq \mathbb{A}$ and $\pi : \mathbb{A}^{Perm}$ a permutation on atoms write

$$\pi \textbf{ fixes } X \quad \textbf{for} \quad \text{"}\forall x \in X.\ \pi.x = x\text{"}. \tag{8}$$

For an arbitrary z^α, read

$$\text{"}\forall \pi : \mathbb{A}^{Perm}.\ \pi \textbf{ fixes } X \implies \pi.z = z\text{"} \quad \textbf{as} \quad \text{"}X \textbf{ supports } z\text{"}. \tag{9}$$

Using this notation we define $\mathbf{Supp} : \alpha \to \mathcal{P}(\mathbb{A})$ by

$$\mathbf{Supp}(z^\alpha) \stackrel{\text{def}}{=} \bigcap \{X \in \mathsf{S} \mid X \textbf{ supports } z\}. \tag{10}$$

$\mathbf{Supp}(z)$ is a notion of the "variable names inside z". Indeed,

Remark 4.6. For $X \in \mathsf{S}$, $\mathbf{Supp}(X) = X = \mathbf{Supp}(\mathbb{A} \setminus X)$. Also, for terms of a datatype such as $t : \Lambda$ as defined in (4), $\mathbf{Supp}(t)$ is precisely the atoms occuring in t. Proofs omitted. ◇

But in (10) we take an intersection over *small* X supporting z^σ. In the case $z = X \in \mathsf{S}$ or $z = t \in \Lambda$ it is clear that at least one such exists. But what of arbitrary z^σ? We introduce an axiom:

Definition 4.7 (FM-HOL$_{-\iota}$)**.** FM-HOL$_{-\iota}$ is HOL$_{+\mathbf{lift}}$ augmented with an axiom

$$\mathbf{Supp}(z) \in \mathsf{S}. \tag{11}$$

Lemma 4.8. *Every* z^σ *has some (not necessarily minimal) small supporting set. The slogan is* **"all elements have small support"***.*

Lemma 4.9. ***Supp***(z) *supports* z *and, by construction, is a minimal supporting set.*

Proof. Using the technical lemmas and definitions below (for a comment on the fact that we can prove them, see R4.3). □

Lemma 4.10 (Technical properties of S)**.**

1. *For any $X : \mathcal{P}(\mathbb{A})$, either $X = \boldsymbol{Supp}(X)$ or $\mathbb{A} \setminus X = \boldsymbol{Supp}(X)$.*
2. *For any $X : \mathcal{P}(\mathbb{A})$, $X \in \mathsf{S}$ iff $\mathbb{A} \setminus X \in \mathsf{S}$.*
3. *For all $X : \mathcal{P}(\mathbb{A})$, either $X \in \mathsf{S}$ or $\mathbb{A} \setminus X \in \mathsf{S}$, and not both.*
4. *$\boldsymbol{Supp}(\emptyset) = \emptyset = \boldsymbol{Supp}(\mathbb{A})$, so $\emptyset$ is small and $\mathbb{A}$ is large.*
5. *Hence $\mathbb{A}$ is infinite and cannot be well-ordered.*
6. *For $X \in \mathsf{S}$, $X \hookrightarrow \mathbb{A} \setminus X$.*

Proof.
1. By calculation.
2. Observing the diagram (7).
3. $\mathbf{Supp}(X) \in \mathsf{S}$ always by (11), use previous results in this list.
4. Observing $\mathbf{Supp}(\emptyset) = \emptyset$ and using the above.
5. That $\mathbb{A}$ is large was observed above. Large sets cannot be well-ordered by definition (cf. (7)).
6. Again, see (7).

□

Definition 4.11 ($\mathsf{И}$)**.** In view of L4.10 we write X is ***large*** when it is not small, and write the set of large sets of atoms $\mathsf{И}$ (cf. R4.16). Observe that $\mathcal{P}(\mathbb{A}) = \mathsf{S} \cup \mathsf{И}$ and recall from L4.10 that $X \in \mathsf{S} \iff \mathbb{A} \setminus X \in \mathsf{И}$.

Lemma 4.12. *Any small set X injects into any large set Z.*

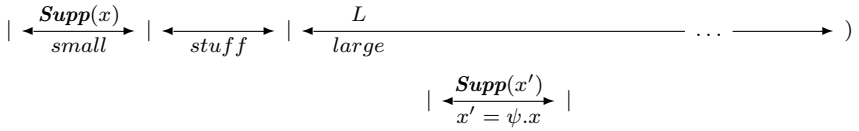
Proof. By L4.10 we know X and $\mathbb{A} \setminus Z$ are small. By L4.4, $X \cup (\mathbb{A} \setminus Z)$ is small. We now consult the picture (7) to inject $X \cup (\mathbb{A} \setminus Z)$ into its complement $(\mathbb{A} \setminus X) \cap Z$, and hence X into Z. □

The following technical result is highly significant. It brings together the properties of small sets of atoms (see (6) and (7)) with the permutation action to prove that for any x^σ we can 'shift' its support $\mathbf{Supp}(x) = U$ to a fresh U'.

Lemma 4.13 (Renaming Lemma)**.** *Recall that any x^σ has small support by* (11)*. From the proof of L4.12 we see that for any x^σ and any $L \in \mathsf{И}$ there is an x'^σ with $\boldsymbol{Supp}(x') \subseteq L$ and ψ such that $x \overset{\psi}{\leftrightarrow} x'$.*

Here $x \overset{\psi}{\leftrightarrow} x'$ means $\psi.x = x'$, $\psi.x' = x$, and ψ is the identity off $\boldsymbol{Supp}(x) \cup \boldsymbol{Supp}(x')$. Call ψ a **renaming permutation***. It "shifts the support of x to be (somewhere) inside L", and this result asserts that for any L, at least one such a ψ exists:*

(12)



This all works towards the following two results:

Lemma 4.14. *If X and Y support z then $X \cap Y$ supports z.*

Proof. The proof is technical but I include it because it has a certain beauty. To visualise it, see the bracketed comment at the end of this proof. If $X \subseteq Y$ or $Y \subseteq X$ then we are done. So let $U = X \setminus Y \neq \emptyset$. Consider some π fixing X and Y. We prove π fixes $X \cap Y$, which completes the proof.

Choose $L \in \mathsf{И}$ large and disjoint from $X, Y, \mathbf{Supp}(\pi)$. Let ψ be a renaming permutation for U and L, $U \overset{\psi}{\leftrightarrow} U'$ for $U' \subseteq L$ as discussed in L4.13.

Observe that ψ fixes Y. It follows by our assumption that Y supports z that $\psi.z = z$, whence

$$(\psi^{-1} \circ \pi \circ \psi).z = \pi.z.$$

However, $\psi^{-1} \circ \pi \circ \psi$ fixes X. So by our assumption that X supports z, we have $\pi.z = z$ as required. (I would draw a picture of this proof. I suggest a two-dimensional Venn diagram, with small sets X, Y represented by intersecting circles inside a large box $\mathbb{A}$.) □

Lemma 4.15. *Any descending chain of small sets $X_1 \supseteq X_2 \supseteq X_3 \ldots$ terminates. This property is inherited from the corresponding property of well-orderings (ordinals).*

Proof of L4.9. Suppose $\mathbf{Supp}(z)$ does not support z. We examine the definition (10) and using L4.14 deduce the existence of a non-terminating descending chain of supporting sets above $\mathbf{Supp}(z)$. This contradicts L4.15. □

Remark 4.16. We now continue to develop a theory of syntax-with-binding using $\mathbb{A}$ as object-level variable names. The reader may wonder why we called the large atoms $\mathsf{И}$ instead of, say, L for large. $\mathsf{И}$ stands for 'new' and gives rise to a novel quantifier whose theory we now develop: ◇

5. # and И

Recall from D4.11 we introduced the notation $\mathsf{И}$ for the set of large sets of atoms.

Definition 5.1. Write $\mathsf{И}\, P$ or $\mathsf{И}a.\ P(a)$ for "$P \in \mathsf{И}$".

$\mathsf{И}$ the quantifier has excellent properties:

Lemma 5.2. $\mathsf{И}$ *distributes over all structure of the HOL logic. That is:*

$$\mathsf{И}a.\ P(a) \wedge \mathsf{И}a.\ Q(a) \iff \mathsf{И}a.\ P(a) \wedge Q(a) \tag{13}$$

$$\mathsf{И}a.\ P(a) \vee \mathsf{И}a.\ Q(a) \iff \mathsf{И}a.\ P(a) \vee Q(a) \tag{14}$$

$$\mathsf{И}a.\ \neg P(a) \iff \neg \mathsf{И}a.\ P(a) \tag{15}$$

$$\mathsf{И}a.\ \top \tag{16}$$

$$\neg \mathsf{И}a.\ \bot \tag{17}$$

$$(\mathsf{И}a.\ P(a)) \wedge Q \iff \mathsf{И}a.\ (P(a) \wedge Q) \tag{18}$$

Proof. From L4.4 and L4.10 we can deduce that S and $\mathsf{И}$ are closed under pairwise meets and joins, $\emptyset \in \mathsf{S}$ and $\mathbb{A} \in \mathsf{И}$. This proves all formulae save the last. To prove that, observe that FM-HOL is classical so for any $Q^{\mathbb{B}}$ we know $Q = \top$ or $Q = \bot$, proceed by cases. □

The significance of L5.2 is that we may choose two fresh atoms in different proofs of $\mathsf{И}\, P$ and $\mathsf{И}\, Q$, and it does not matter *which* ones because we can always use L5.2 to deduce $\mathsf{И}a.\ P(a) \wedge Q(a)$. Similarly, in a proof if we wish to prove $\mathsf{И}\, Q$ from $\mathsf{И}\, P$ it suffices to prove $\mathsf{И}a.\ P(a) \to Q(a)$, that is, to 'choose a fresh atom' a and prove $P(a) \to Q(a)$.

Definition 5.3. Write

$$a \notin \mathbf{Supp}(x) \quad \textbf{as} \quad a\#x,$$

and read this as "a ***is fresh for*** x" or sometimes "a ***is not 'in'*** x".

Observe that a predicate $P{:}\mathbb{A} \to \mathbb{B}$ is just a set of atoms X, hence subject to L4.10. We can use this and our new definition D5.3 to turn D5.1 into directed intro- and elim-rules for $\mathsf{И}$:

Lemma 5.4 ($\mathsf{И}$ introduction)**.** $a\#P \wedge P(a) \implies \mathsf{И}a.\ P(a)$.

Proof. By L4.10 $P = \mathbf{Supp}(P)$ or $P = \mathbb{A} \setminus \mathbf{Supp}(P)$. Since $P(a)$ and $a \notin \mathbf{Supp}(P)$ we have $P = \mathbb{A} \setminus \mathbf{Supp}(P)$. Since $\mathbf{Supp}(P) \in \mathsf{S}$, we have $P \in \mathsf{И}$ as required. □

Lemma 5.5 ($\mathsf{И}$ elimination)**.** $(\mathsf{И}a.\ P(a)) \wedge a\#P \implies P(a)$.

Proof. We have assumed $P \in \mathsf{И}$ and $a \notin \mathbf{Supp}(P)$. Since $\mathbf{Supp}(P) \in \mathsf{S}$ we know $P = \mathbb{A} \setminus \mathbf{Supp}(P)$ so $P(a)$ as required. □

Remark 5.6. Thus И is a fully-fledged binder with well-defined intro- and elim-rules. Note how they pleasingly combine (the best) aspects of $\forall$ and $\exists$ rules. Recall R1.1. When doing induction, to prove $И a.\ P(a)$ it suffices to establish it for *one* fresh a, and conversely when we know $И a.\ P(a)$ we have $P(a)$ for *any* fresh a. In FM, we have the 'happy compromise' spoken of in R1.1. $\diamond$

We conclude by using И to rewrite (2), a definition of $=_\alpha$ on the Λ of (1), in FM style on Λ of (4):

$$\Lambda \stackrel{\text{def}}{=} \mathbf{Var\ of}\ \mathbb{A} + \mathbf{App\ of}\ \Lambda \times \Lambda + \mathbf{Lam\ of}\ \mathbb{A} \times \Lambda \tag{19}$$

$$\begin{array}{rll} t =_\alpha t' \stackrel{\text{def}}{=} & n = n' & \leftarrow t = \mathbf{Var}(n), t' = \mathbf{Var}(n') \\ & s_1 =_\alpha s_1', s_2 =_\alpha s_2' & \leftarrow t = \mathbf{App}(s_1, s_2), t' = \mathbf{App}(s_1', s_2') \\ & И u.\ (u\ n).s =_\alpha (u\ n').s' & \leftarrow t = \mathbf{Lam}(n, s), t' = \mathbf{Lam}(n', s') \end{array} \tag{20}$$

It remains however to *bind* atoms in syntax:

6. Construction of $a.x$

Definition 6.1. Define

$$a.x^\alpha \stackrel{\text{def}}{=} \bigcap \left\{ X \subseteq \mathbb{A} \times \alpha \;\middle|\; <a, x> \in X \wedge \mathbf{Supp}(X) \subseteq \mathbf{Supp}(x) \setminus \{a\} \right\}. \tag{21}$$

More generally, for types α and β, we may define

$$x^\alpha.y^\beta \stackrel{\text{def}}{=} \bigcap \left\{ X \subseteq \alpha \times \beta \;\middle|\; <x, y> \in X \wedge \mathbf{Supp}(X) \subseteq \mathbf{Supp}(y) \setminus \mathbf{Supp}(x) \right\}. \tag{22}$$

Definition 6.2. Define new type-constructors by

$$\mathbb{A}.\alpha \stackrel{\text{def}}{=} \left\{ a^{\mathbb{A}}.x^\alpha \right\} \qquad \textbf{and} \qquad \alpha.\beta \stackrel{\text{def}}{=} \left\{ x^\alpha.y^\beta \right\}. \tag{23}$$

The coincidence between the special case of $\alpha = \mathbb{A}$ is deliberate, they are the same type, proof omitted.[1]

Lemma 6.3. $\boldsymbol{Supp}(x.y) \subseteq \boldsymbol{Supp}(y) \setminus \boldsymbol{Supp}(x)$ *and in particular* $a\#a.x$. *The proof is a simple corollary of the technical lemma* $\boldsymbol{Supp}(\bigcap_i X_i) \subseteq \bigcup_i \boldsymbol{Supp}(X_i)$.

Lemma 6.4 (Alternative characterisation)**.** *An alternative characterisation of* $x^\alpha.y^\sigma$ *is as the equivalence-class* $[<x, y>]_\sim$ *where*

$$<x, y> \sim <x', y'> \stackrel{\text{def}}{=} \mathsf{R}x \xrightarrow{\phi} x'.\, y' = \phi.y. \tag{24}$$

R *binds only* ϕ, *read* $\mathsf{R}x \xrightarrow{\phi} x'.\, P(x, \phi)$ *as "rename* x *to* x' *(by* ϕ*) in* P*". It is defined by*

$$\mathsf{R}x \overset{\phi:\mathbb{A}^{Perm}}{\longrightarrow} x'.\, P(x, \phi) \stackrel{\text{def}}{=} \exists \phi : \mathbb{A}^{Perm}.\ P(x, \phi)\ \wedge\ x \overset{\phi}{\leftrightarrow} x'\ \wedge\ \boldsymbol{Supp}(x') \cap \boldsymbol{Supp}(P) \subseteq \boldsymbol{Supp}(x) \tag{25}$$

where $x \overset{\phi}{\leftrightarrow} x'$ *is defined in L4.13.*

Proof. We prove $\sim$ an equivalence relation. We then use a technique similar to that in the proof of L4.12 to find z, y' such that $<x, y> \sim <z, y'>$ and z has support completely fresh for the context, and use this to deduce $\mathbf{Supp}(x)$ is disjoint from $\mathbf{Supp}([<x, y>]_\sim)$. Clearly $<x, y> \in [<x, y>]_\sim$. From all this we can conclude $[<x, y>]_\sim = x.y$. $\square$

Lemma 6.5. $\boldsymbol{Supp}(x.y) = \boldsymbol{Supp}(y) \setminus \boldsymbol{Supp}(x)$. *Proof using L6.4.*

Lemma 6.6. *For* $x, y : \alpha$ *and a fixed* $a : \mathbb{A}$, $a.x = a.y \Leftrightarrow x = y$. *Proof omitted.*

Remark 6.7. L6.3, L6.4, and L6.5 tell an attractive story: abstraction $x.y$ simply forms the pair $<x, y>$, but also binds precisely those atoms which may occur free in x.

L6.6 completes the picture and tells us that in particular, $\mathbb{A}.\alpha = \left\{ a^{\mathbb{A}}.x^\alpha \right\}$ is the type of x^α with one distinguished bound atom a. $\diamond$

[1] Generalised abstraction was discovered independently by Pitts.

Remark 6.8. This morally justifies our usage of the following as a datatype of λ-terms up to α-equivalence:

$$\Lambda_\alpha \stackrel{\text{def}}{=} \mathbf{Var\ of}\ \mathbb{A} + \mathbf{App\ of}\ \Lambda_\alpha \times \Lambda_\alpha + \mathbf{Lam\ of}\ \mathbb{A}.\Lambda_\alpha. \tag{26}$$

Here the term $\lambda a.a$ is represented by $\mathbf{Lam}(a.\mathbf{Var}(a))$. $a.\mathbf{Var}(a)$ plays the rôle of $\mathbf{Lam}(\lambda a^{\mathbb{A}}.\mathbf{Var}(a))$ in (one flavour of) HOAS, $\mathbf{Lam}(\mathbf{Var}(0))$ in De Bruijn, or simply $\mathbf{Lam}{<}a, \mathbf{Var}(a){>}$ in a naïve representation.

The formal justification for the claim that Λ_α is λ-terms up to α-equivalence is by defining a function from Λ in (4) and (19) to (26)

$$\begin{array}{rcll} q : \Lambda \to \Lambda_\alpha & \stackrel{\text{def}}{=} & \mathbf{Var}(a) & \mapsto \mathbf{Var}(a) \\ & & \mathbf{App}(s_1, s_2) & \mapsto \mathbf{App}(q(s_1), q(s_2)) \\ & & \mathbf{Lam}(a, s) & \mapsto \mathbf{Lam}(a.(q(s))) \end{array} \tag{27}$$

and proving q surjective and that its kernel is $=_\alpha$ as defined in (20). Details omitted. $\diamond$

Thus we have a definition Λ_α of λ-terms Λ quotiented by $=_\alpha$, $\Lambda/=_\alpha$. The top-level structure of Λ_α is *not* a quotient, but an inductive datatype. Thus the definition of some standard function on Λ_α, e.g. capture-avoiding substitution, is purely inductive, but to construct it (see (30)) we need definite choice ι:

7. Axioms of choice and consistency

Lemma 7.1. HOL$_{+\textit{lift}}$ *and FM-HOL*$_{-\iota}$ *both have easy models in* FM *sets with types as sets and function types as sets of function-sets. Thus, these theories are consistent.*

Proof. Write the semantic mapping function $[\![-]\!]$. Then $[\![\mathbb{A}]\!]$ is the set of atoms $\mathbb{A}$, $[\![\mathbf{Tran}^{\mathbb{A}\to\mathbb{A}\to\alpha\to\alpha}]\!]$ is set-transposition restricted to the function-set of its action on $[\![\alpha]\!]$.

There is only one significant detail: types *do* have to be interpreted as *equivariant* sets ("$[\![\alpha]\!]$ is equivariant for all α", see R3.4) so that for all $a = [\![a^{\mathbb{A}}]\!], b = [\![b^{\mathbb{A}}]\!] \in \mathbb{A}$ and $x \in [\![\alpha]\!]$, $[\![\mathbf{Tran}]\!](a, b, x) \in [\![\alpha]\!]$ always. $\square$

Remark 7.2. In an automated environment it is extremely useful to mediate between predicates and functions. That is, if we construct a predicate $P(x, y)$ we want to be able to derive from it a function symbol $f_P(x)$ such that $P(x, f_P(x))$. No sugar—we want an actual closed term $f : (\alpha \to \alpha \to \mathbb{B}) \to (\alpha \to \mathbb{B})$, and we want to be able to prove $P(x, f(P)(x))$ in the logic.

The two usual candidates for constructing such an f are indefinite choice, also known as Hilbert's ϵ, and definite choice, written ι. ϵ is a polymorphic constant symbol $\epsilon : (\alpha \to \mathbb{B}) \to \alpha$ with axiom $(\exists x.\ P(x)) \Rightarrow P(\epsilon\, P)$. Given that, we take f to be $\lambda x.\epsilon y.\, P(x, y)$.

ι is a polymorphic constant symbol $\iota : (\alpha \to \mathbb{B}) \to \alpha$ such that $(\exists! x.\ P(x)) \Rightarrow P(\iota\, P)$. We can take f to be $\lambda x.\iota y.\, P(x, y)$, which lets us form a function-term from the term for its graph.

To me ι seems vital, because forming functions from graphs is common practice which an automated HOL should support. $\diamond$

Lemma 7.3. HOL$_{+\textit{lift}}$*, and therefore FM-HOL*$_{-\iota}$*, are inconsistent with Hilbert's* ϵ *R7.2. Cf. the corresponding* FM *sets phenomenon in* [2, Remark 4.6].

Proof. Consider two terms

$$a^{\mathbb{A}} \stackrel{\text{def}}{=} \epsilon x^{\mathbb{A}}.\, \top \quad \textbf{and} \quad b^{\mathbb{A}} \stackrel{\text{def}}{=} \epsilon x^{\mathbb{A}}.\, x \neq a.$$

Observe that both a and b are closed terms yet $(a\ b).a \neq a$. This contradicts (LE) for $t = a$ and $\pi = (a\ b)$. $\square$

However...

Lemma 7.4. *Call a type α **pure** when every x^α is equivariant (for all $a, b{:}\mathbb{A}$, $(a\ b).x = x$, see R3.4 and cf.* [2, Remark 4.2]*), standard examples are $\mathbb{B}$ and $\mathbb{N}$. If α and β are pure then so is $\alpha \to \beta$, proof omitted. Thus the type-class of pure types is closed under function-types. It is consistent to postulate Hilbert's $\epsilon : (\alpha \to \mathbb{B}) \to \alpha$ for α of that class, proof omitted.*

Thus in an automated environment with type-classes, such as Isabelle/HOL, we do not actually *lose* any theory, since the standard theories of real numbers, groups, and so on, are carried out in the pure types.

Definition 7.5 (FM-HOL)**.** As the name suggests, FM-HOL$_{-\iota}$ lacks ι. But it is consistent to add it, as we discuss below, and we call the theory obtained FM-HOL.

Now adding ι to FM-HOL$_{-\iota}$ leads to no obvious inconsistencies in the logic. However, there is a snag *proving* consistency using the set-theoretic models we used in L7.1 to prove consistency of FM-HOL$_{-\iota}$:

Remark 7.6. In a set-theoretic model of FM-HOL$_{-\iota}$ as discussed in L7.1, consider how to add $[\![\iota]\!]$ for each type α. This is a function-set in $([\![\alpha]\!] \to [\![\mathbb{B}]\!]) \to [\![\alpha]\!]$.

In fact FM sets has definite choice ι so we might think we could simply interpret the HOL-ι as the function-set obtained restricting sets-ι to $[\![\alpha]\!] \to [\![\mathbb{B}]\!]$.

But ι is typed as a total function and most $P^{\alpha\to\mathbb{B}}$ do not satisfy $\exists!x.\ P(x)$. What should $[\![\iota]\!]([\![P]\!])$ be for such a P? Sets-ι can default to some fixed set, e.g. $\emptyset$. $[\![\iota]\!]$ cannot, because the default value must be in $[\![\alpha]\!]$. Consider the case $\alpha = \mathbb{A}$. Whichever default atom $a \in [\![\mathbb{A}]\!]$ we choose for "bad P" the closed term $t \stackrel{\text{def}}{=} \iota x^{\mathbb{A}}.\ \top$ will not satisfy (LE) in the model, since for any $b \neq a$, $(b\ a).[\![t]\!] \neq [\![t]\!]$.

There are (at least) two solutions. Pitts suggests a domain-theoretic semantics so that $[\![\iota]\!]$ can default to $\bot$. Personally I miss the simplicity of a set model of our HOL and want to preserve it if I can. I think I know how to do so, and with honour too: ◇

Remark 7.7. Introduce a constant symbol $D : \mathsf{S}$ with axiom

$$\forall S : \mathsf{S}.\ S \hookrightarrow D. \tag{28}$$

(Recall the notation of D4.2: $S \hookrightarrow D$ means "S injects into D".)

Now consider a system FM-HOL given by the above along with a modified version of FM-HOL$_{-\iota}$ such that (LE) is taken to be

$$\pi \textbf{ fixes } D \implies (t\,\vec{x}) \;=\; \pi.\big(t\ (\pi^{-1}.\vec{x})\big) \quad \textbf{when } t \textbf{ closed.} \tag{29}$$

('π **fixes** D' defined in (8).) ◇

Remark 7.8 (Nameable atoms)**.** Thus D gives us a stock of "***nameable atoms***" for which equivariance (LE) need not apply, and such that we can consistently introduce into the meta-language closed terms (names) $c \in D$ for them. ◇

Remark 7.9. In this paper we introduce D to solve the technical problem of modelling ι also for badly-typed P (R7.2, R7.6), but it has independent motivations.

Suppose we want to represent in FM-HOL this common syntactic declaration: "Elements of *widget* are written $x, y, z, \ldots \in$ *widget*". Are not x, y, z really these 'nameable atoms'? Similarly for other sources of distinguished atoms, such as '`stderr`' ($\mathbb{A}$ is UNIX filehandles) or '`Bool`' ($\mathbb{A}$ is base types) and so on. We might introduce a separate type such as $\mathbb{N}$ and use a sum type $\mathbb{A}+\mathbb{N}$ for 'x, y, z, and $\ldots$', or 'system filehandles and user-defined ones', etc. But in automation this carries a significant penalty in efficiency and readability (note for experts: and of course if we look at the actual set-theoretic denotation, using $\mathbb{A} + \mathbb{N}$ just 'hacks' D in using the expressivity of the object-level).

An independent motivation for nameable atoms is as expressing atoms introduced by an external context or 'local theory declaration'. This is best explained by example. To implement a theory of groups in some machine system it is convenient to open a 'local theory', in which is declared a group G. We prove our theorems about this fixed but

arbitrary G, then close the theory. When we do so G is eliminated and our theorems, Thm say, are lifted over an extra hypothesis **"G a group"**, to become G **a group** $\Rightarrow Thm$.

This avoids the clutter of a standing hypotheses **"G a group"** in the proof-script developing the theory of groups and is nowadays standard practice in proving environments, which a FM-HOL implementation should accommodate (see for example [8] or [7, 'Sections', 1.1.1]). $\diamond$

Returning to the task of giving semantics to ι, we can construct an FM set theory with corresponding set D and set equivariance axiom (29), and allow $[\![\iota]\!]$ to default to some value with support entirely in D in the case of 'badly-typed' input $[\![P]\!]$. $\forall S \in \mathsf{S}.\ S \hookrightarrow D$ ensures D is at least as large as any set of names we can enumerate, so that we can always find a new default error-value no matter what is in the context.

Lemma 7.10. *FM-HOL with nameable atoms and unique choice ι, as described above, is consistent. Further details of the proof omitted.*

Using ι we can now give a sensible definition of substitution on Λ_α in (26), the type of λ-terms up to α-equivalence:

$$\begin{array}{lll} [u/n]\mathbf{Var}(n) & = u & \\ [u/n]\mathbf{Var}(a) & = \mathbf{Var}(a) & (a \neq n) \\ [u/n]\mathbf{App}(t_1, t_2) & = \mathbf{App}([u/n]t_1, [u/n]t_2) & \\ [u/n]\mathbf{Lam}(a.t) & = \mathbf{Lam}(a.[u/n]t) & \end{array} \tag{30}$$

—which is what we would expect the definition to look like in an FM programming language.

In the final clause of this definition, that for **Lam**, we have made some effort to present matters attractively, as a user of an implemented HOL datatypes package might expect to type in. The specification, expressed religiously in the FM language we have had the space to develop in this paper, is

$$[u/n]\mathbf{Lam}(t_*^{\mathbb{A}.\Lambda_\alpha}) = \iota z.\, \mathit{M}a.\ \exists t.\ \Big(t_* = a.t\ \wedge\ z = \mathbf{Lam}(a.[u/n]t)\Big).$$

This formula adds no particular mathematical context to the development: for new a there always exists t such that $t_* = a.t$, the underlying technical result supporting this is the renaming lemma L4.13, and it does not matter which a we use, cf. §5 and R5.6. We omit further discussion.

8. Implementing FM-HOL, Conclusions

An FM-HOL-like theory called Isabelle/HOL/FM has been implemented and is described in [5]. It is not quite an implementation of FM-HOL. It uses a HOL meta-language essentially identical to the FM-HOL$_{-\iota}$ of §4. However, it constructs an FM set theory in a type of individuals ('the set universe') `i`, rather than directly implementing FM structures (such as $a.x$, D6.1) in its own types. This was for reasons partly historical and partly practical. Concerning history, the implementation started off as FM sets, see the discussion at the start of §1. Concerning practicality, the inductive definitions package of Isabelle set theory (see [10]) seems better-suited to theoretical work. It has an emphasis on flexibility and doing everything inside the Isabelle theory which means if we hack the Isabelle theory (e.g. to include elements of FM) the datatypes package changes with it relatively painlessly. In contrast the inductive definitions package of Isabelle/HOL ([13, Ch.6]) seems aimed at the end-user. It is highly-engineered with lots of powerful ML code and slick user interfaces which assume certain things of the theory which FM is liable to make false.

At my suggestion Berghofer, Paulson, and Wenzel kindly reengineered the Isabelle/HOL system to make it independent of ϵ (see L7.3) except, apparently, for a hidden dependency in the datatypes package. Implementing Isabelle/FM-HOL now seems feasible. Perhaps I

should simply go right in, hack Isabelle/HOL to Isabelle/FM-HOL, and if some ML code breaks then so be it.

While writing, I have always had applications in mind: when I motivated a new constant symbol, or a particular way of doing things, it was usually with reference to implementation. And yet I have tried to show the beauty of this logic as an abstract entity in its own right. And this beauty charms me. I hope my reader has enjoyed the story too.

References

1. M. J. Gabbay and A. M. Pitts, *A new approach to abstract syntax involving binders*, 14th Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, Washington, 1999, pp. 214–224.
2. M. J. Gabbay and A. M. Pitts, *A new approach to abstract syntax with variable binding*, Formal Aspects of Computing **?** (2001), ?–?, Special issue in honour of Rod Burstall. To appear.
3. Murdoch J. Gabbay, *A theory of inductive definitions with alpha-equivalence*, Ph.D. thesis, Cambridge, UK, 2000.
4. ———, *FM-HOL, a higher-order theory of names*, 35 Years of Automath, Heriot-Watt University, Edinburgh, Scotland, April 2002, Submitted.
5. ———, *FM techniques in Isabelle*, TPHOLS, August 2002, Submitted.
6. M. Hofmann, *Semantical analysis of higher-order abstract syntax*, 14th Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, Washington, 1999, pp. 204–213.
7. Paulin-Mohring Huet, Kahn, *The COQ tutorial, v7.2*, http://pauillac.inria.fr/coq/doc/tutorial.html, LogiCal Project.
8. Florian Kammüller, Markus Wenzel, and Lawrence C. Paulson, *Locales: A sectioning concept for Isabelle*, LNCS, vol. 1690, TPHOLS99, Springer, 1999.
9. James McKinna and Robert Pollack, *Some lambda calculus and type theory formalized*, Journal of Automated Reasoning **23** (1999), no. 3-4, 373–409.
10. Lawrence C. Paulson, *(Co)Inductive definitions in ZF*, Isabelle System.
11. A. M. Pitts, *Nominal logic: A first order theory of names and binding*, Theoretical Aspects of Computer Software, 4th International Symposium, TACS 2001, Sendai, Japan, October 29-31, 2001, Proceedings (N. Kobayashi and B. C. Pierce, eds.), Lecture Notes in Computer Science, vol. 2215, Springer-Verlag, Berlin, 2001, pp. 219–242.
12. A. M. Pitts and M. J. Gabbay, *A metalanguage for programming with bound names modulo renaming*, Mathematics of Program Construction. 5th International Conference, MPC2000, Ponte de Lima, Portugal, July 2000. Proceedings (R. Backhouse and J. N. Oliveira, eds.), Lecture Notes in Computer Science, vol. 1837, Springer-Verlag, Heidelberg, 2000, pp. 230–255.
13. Larry Paulson Tobias Nipkow, *Isabelle HOL: The tutorial (draft)*, February 2001, To be published by Springer.
14. J. Truss, *Permutations and the axiom of choice*, Automorphisms of first order structures (H.D.Macpherson R.Kaye, ed.), OUP, 1994, pp. 131–152.

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