数理解析研究所講究録 1318 巻 2003 年 46-47

Nominal Sets, Equivariance Reasoning, and Variable Binding

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

1 Equivariance reasoning

FM techniques are a methodology of thinking about syntax and in particular about syntax with binding. They take their name from the original FM (Fraenkel-Mostowski) theory of sets presented in my thesis [4]. For this document we use a different way of presenting FM techniques based on the idea of a **Nominal Set**, which is a set equipped with certain algebraic properties (just as a group, ring, or field, is a set equipped with certain properties), see Definition 1.2. First we need background machinery:

Definition 1.1. Fix a countably infinite **set of atoms** $\mathbb{A}$. Write typical elements $a, b, c, \ldots \in \mathbb{A}$. For $a, b \in \mathbb{A}$ write $(a\ b)$ for the function $\mathbb{A} \to \mathbb{A}$ such that $a \mapsto b$ and $b \mapsto a$ and $n \mapsto n$ for all other atoms $n \neq a, b$. This is a bijection with inverse itself, write $P_{\mathbb{A}}$ for the group generated by $(a\ b)$ for all $a, b \in \mathbb{A}$ under functional composition $\circ$. Write typical elements $\pi, \pi' \in P_{\mathbb{A}}$ and $\mathbf{Id} \in P_{\mathbb{A}}$ for the identity.

Definition 1.2. A **Nominal Set** is a pair $\langle X, \cdot\, \rangle$ of an underlying set X and **permutation action** $\cdot$ (written infix) of type $P_{\mathbb{A}} \times X \to X$ and satisfying the usual axioms, namely $\pi \cdot (\pi' \cdot x) = \pi \circ \pi' \cdot x$ and $\mathbf{Id} \cdot x = x$. The permutation action also satisfies a finiteness condition omitted here.[1]

The point is that finite labelled trees, and hence the standard model of syntax but also a natural model of *proofs* as trees, are Nominal Sets: the permutation action is given pointwise by the action on the labels. Thus for example natural numbers $\mathbb{N}$ satisfy a trivial permutation action given by $\pi \cdot n = n$ always. A datatype of trees for terms of the λ-calculus (using $\mathbb{A}$ as variable names)

$$\Lambda \cong \mathbb{A} + \Lambda \times \Lambda + \mathbb{A} \times \Lambda \tag{1}$$

is also a Nominal Set with the permutation action given pointwise by the action on the atoms labelling the tree. Furthermore we can represent a theory of α-equivalence on these terms as a subset of "well-formed" trees in the inductively defined set

$$T \cong \mathbb{A} + T \times T + \mathbb{A} \times T \times T, \tag{2}$$

namely those inductively constructed using the rules

$$a =_\alpha a \quad (\mathbf{Var}) \qquad \frac{t_1 =_\alpha t_1' \quad t_2 =_\alpha t_2'}{t_1 t_2 =_\alpha t_1' t_2'} \quad (\mathbf{App}) \qquad \frac{t\{c/a\} =_\alpha t'\{c/a'\}}{\lambda a t =_\alpha \lambda a' t'} \quad (\mathbf{Lam}) \tag{3}$$

where in (**Lam**) there is a side-condition that $c \notin \{a, a'\} \cup n(t) \cup n(t')$ (thus 'fresh' for the conclusion). The usefulness of this way of looking at syntax and properties of syntax as Nominal Sets begins with the following trivial theorem:

Theorem 1.3. *If a property (on trees) is defined (inductively) using predicates whose validity is invariant under permuting atoms, then the property is invariant under permuting atoms.*

Here "invariant under permuting..." means "given some valid instance of the property, a permutation π uniformly applied to its arguments yields another valid instance".

We have an example in the property of well-formedness of proof-trees of $=_\alpha$ given in (3). $\overline{a =_\alpha a}$ is a valid instance of (**Var**) and if we apply $(b\ a)$ to this we obtain $\overline{b =_\alpha b}$, which is also a valid instance of

---

[1] See [3, eq. 3], [5, eq. 4], and 'finite support' in [2, Def. 3.3].

(**Var**). The case of (**App**) is simple. A permutation applied to a valid instance of (**Lam**) is also a valid instance of (**Lam**) because $c \notin \{a, a'\} \cup n(t) \cup n(t')$ if and only if $\pi \cdot c \notin \{\pi \cdot a, \pi \cdot a'\} \cup n(\pi \cdot t) \cup n(\pi \cdot t')$.[2]

We now have a very concrete demonstration that proofs of $=_\alpha$ are invariant under permutation; we permute the atoms in the proof as a tree. We can take this further. Consider proving transitivity of $=_\alpha$ by induction on proof-trees. The induction predicate is (in words) "given a valid proof-tree Π concluding in $t =_\alpha t'$, for all valid proof-trees Π' concluding in $t' =_\alpha t''$, there exists a valid proof-tree Π'' concluding in $t =_\alpha t''$". This property is constructed using predicates invariant under permutations (validity of proofs of $=_\alpha$) and so is itself invariant under permutations. Thus from Theorem 1.3 we know if we have the inductive hypothesis of Π, we have it of $\pi \cdot \Pi$ for any permutation Π.

We proceed by induction on Π. The case of (**Lam**) for $t = \lambda as$ and $t' = \lambda a's'$ causes problems: we may assume Π proves $s[c/a] =_\alpha s'[c/a']$ and Π' proves $s'[c'/a'] =_\alpha s''[c'/a'']$ and we assume the inductive predicate for Π, but we do not know $c = c'$ so we cannot proceed. However, we *can* apply a permutation $(d\ c)$ to Π, and $(d\ c')$ to Π', for d chosen completely fresh. Now we have valid proofs $(d\ c) \cdot \Pi$ concluding in $s[d/a] =_\alpha s'[d/a']$ and $(d\ c') \cdot \Pi'$ concluding in $s'[d/a'] =_\alpha s''[d/a'']$, and also the inductive predicate for $(d\ c) \cdot \Pi$. We can now complete the proof of transitivity.

Just these ideas of permutations have already been adopted and put to use by other authors also in published work (see for example [6]).

2 Taking it further

There is an equivalence class of proofs concluding in $\lambda at =_\alpha \lambda a't'$, one for each fresh c; we can take it as an object in its own right. This is an instance of FM abstraction $[\mathbb{A}]X$ which exists for any Nominal Set X by virtue of the permutation action, which lets us rename atoms and construct an equivalence class in the general case (see [2, Section 5]). We can apply this to syntax as well as proofs:

$$\Lambda_\alpha \cong \mathbb{A} + \Lambda_\alpha \times \Lambda_\alpha + [\mathbb{A}]\Lambda_\alpha \tag{4}$$

is a datatype of λ-terms up to α-equivalence. An element of $[\mathbb{A}]\Lambda_\alpha$ is (concretely) an equivalence class of pairs $\langle a, t\rangle$ for $a \in \mathbb{A}$ a 'bound atom' fresh for the other atoms in the 'body' $t \in \Lambda_\alpha$.

There are various ways of taking this further; Nominal Sets form a category, the Schanuel Topos. Because it is a topos we can construct a general theory of abstractions and equivariance reasoning within it (this is FMCat in [5, Section 2], see also [2, p.21]). Nominal Sets are also a special case of a general theory of FM sets, see [4] and [2]. Nominal Sets can be axiomatised in first-order logic, see [7]. A team in Cambridge has developed FreshML, a programming language based on these principles in which we can program using abstractions and permutations, see [1]. Finally, I am currently implementing FM sets in Isabelle, see [4]. Further reading can be found in any of the references below, and my homepage `www.cl.cam.ac.uk/~mjg1003`.

References

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

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

[3] Murdoch Gabbay, *FM for names and binding*, `cl.cam.ac.uk/~mjg1003/papers/talks/200212-jaist-talk.ps`.

[4] Murdoch J. Gabbay, *A theory of inductive definitions with alpha-equivalence*, Ph.D. thesis, Cambridge, UK, 2000.

[5] Murdoch J. Gabbay, *Theory and models of the π-calculus using fraenkel mostowski generalised*, Submitted to the commemorative volume for 35 years of Automath, F. Kamareddine (ed.)., September 2002.

[6] L. Cardelli, P. Gardner, G. Ghelli, *Manipulating trees with hiding*, FOSSACS 2003, to appear.

[7] A. M. Pitts, *Nominal logic, a first order theory of names and binding*, Information and Computation, 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.).

---

[2] This is not the case if we try to base the theorem on substitutions generated by $[b/a]$ instead of permutations generated by $(b\ a) = [b/a, a/b]$.