<table>
<thead>
<tr>
<th>Title</th>
<th>Nominal Sets, Equivariance Reasoning, and Variable Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td></td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2003), 1318: 46-47</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2003-05</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/43028">http://hdl.handle.net/2433/43028</a></td>
</tr>
<tr>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Kyoto University
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 A. Write typical elements a, b, c, ... ∈ A. For a, b ∈ A write (a b) for the function A → A such that a → b and b → a and n → n for all other atoms n ≠ a, b. This is a bijection with inverse itself, write PA for the group generated by (a b) for all a, b ∈ A under functional composition o. Write typical elements π, π' ∈ PA and Id ∈ PA for the identity.

Definition 1.2. A Nominal Set is a pair (X, ·) of an underlying set X and permutation action · (written infix) of type PA × X → X and satisfying the usual axioms, namely π · (π' · x) = π o π' · x and Id · 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 N satisfy a trivial permutation action given by π · n = n always. A datatype of trees for terms of the λ-calculus (using A as variable names)

Λ ∼ A + Λ × Λ + A × Λ

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 ∼ A + T × T + A × T × T,

namely those inductively constructed using the rules

a =₂ a (Var) \[ \frac{t_1 =_\alpha t_1' \quad t_2 =_\alpha t_2'}{t_1 t_2 =_\alpha t_1' t_2'} \] (App) \[ \frac{t\{c/a\} =_\alpha t'\{c/a'\}}{\lambda at =_\alpha \lambda at'} \] (Lam)

where in (Lam) there is a side-condition that c ∉ \{a, a'\} ∪ n(t) ∪ 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 =₂ given in (3). \[ a =_\alpha a \] 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

¹See [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 $\pi \cdot c \notin \{\pi \cdot a, \pi \cdot a'\} \cup n(\pi \cdot t) \cup n(\pi \cdot t')$.

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 $\Pi$ concluding in $t =_{\alpha} t'$, for all valid proof-trees $\Pi'$ concluding in $t' =_{\alpha} t''$, there exists a valid proof-tree $\Pi''$ 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 $\Pi$, we have it of $\pi \cdot \Pi$ for any permutation $\Pi$.

We proceed by induction on $\Pi$. The case of (Lam) for $t = \lambda a$ and $t' = \lambda a'$ causes problems: we may assume $\Pi$ proves $s[c/a] =_{\alpha} s'[c'/a']$ and $\Pi'$ proves $s'[c'/a'] =_{\alpha} s''[c'/a'']$ and we assume the inductive predicate for $\Pi$, but we do not know $c = c'$ so we cannot proceed. However, we can apply a permutation $(d \ c)$ to $\Pi$, and $(d \ c')$ to $\Pi'$, 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 $\lambda a =_{\alpha} \lambda a'$, one for each fresh $c$; we can take it as an object in its own right. This is an instance of FM abstraction $[A]X$ 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:

$$\Lambda_{\alpha} \equiv A + \Lambda_{\alpha} \times \Lambda_{\alpha} + [A]\Lambda_{\alpha}$$

is a datatype of $\lambda$-terms up to $\alpha$-equivalence. An element of $[A]\Lambda_{\alpha}$ is (concretely) an equivalence class of pairs $(a, t)$ for $a \in A$ 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 this is a topos we can construct a general theory of abstractions and equivalence 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


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