Nominal Logic and FM techniques: Reasoning about binding without pain

Murdoch J. Gabbay

Radboud University, Nijmegen, 29/11/2005

Thanks to Henk Barendregt and Bas Spitters

Excerpt from a recent correspondence...

Instead of doing induction to the generation of the formula, you could do it on the length of the formula, chosen in such a way that a formula and a changed version have the same length.

This is what I ended up doing in Coq. I defined the depth of a formula, and did induction over depth. Still, proving various properties of substitution is more difficult than it appears.

The problem

Reasoning on syntax-with-binding requires:

1. Choice of fresh variable symbol.

2. Abstraction of variable symbols.

3. Inductive reasoning.

4. Rename of variable symbols, even free ones, even in predicates.

Fix a countably infinite set of variable (symbols) $a, b, c, \ldots$. Let terms be defined by:

$$t ::= a \mid tt \mid \lambda a.t.$$

Henceforth let t, u be metavariables ranging over terms.

Define free variables inductively by:

$$fv(a) = \{a\} \qquad fv(tt') = fv(t) \cup fv(t')$$

$$fv(\lambda a.t) = (fv(t) \setminus \{a\})$$

Substitution in the λ-calculus

Fix a choice of fresh variable for each finite set of variables.

Define substitution by:

$$a[a \mapsto t] \equiv t$$

$$a[b \mapsto t] \equiv a$$

$$(uu')[a \mapsto t] \equiv (u[a \mapsto t])(u'[a \mapsto t])$$

$$(\lambda a.u)[a \mapsto t] \equiv \lambda a.u$$

$$(\lambda b.u)[a \mapsto t] \equiv \lambda b.(u[a \mapsto t]) \qquad b \notin fv(t)$$

$$(\lambda b.u)[a \mapsto t] \equiv \lambda b'.(u[b \mapsto b'][a \mapsto t]) \quad b \in fv(t),\ b' \text{ fresh}$$

Here 'b' fresh' means 'pick b' fresh for the variables in the terms to the left of $\equiv$'.

$$a \notin fv(t) \quad \implies \quad a \notin fv(u[a \mapsto t]).$$

Proof by induction on u: *(fails)*

- Suppose $u \equiv a$. Then $a[a \mapsto t] \equiv t$.

- Suppose $u \equiv b$. Then $b[a \mapsto t] \equiv b$.

- Suppose $u \equiv u'u''$. Then
  $(u'u'')[a \mapsto t] \equiv (u'[a \mapsto t])(u''[a \mapsto t])$. Now $a \notin fv(u')$ so
  $a \notin fv(u'[a \mapsto t])$ by inductive hypothesis. Similarly for u''. Result
  follows.

- Suppose $u \equiv \lambda a.u'$. Then $(\lambda a.u')[a \mapsto t] \equiv \lambda a.u'$. Result
  follows.

- Suppose $u \equiv \lambda b.u'$ and $b \notin fv(t)$. Then
  $(\lambda b.u')[a \mapsto t] \equiv \lambda b.(u'[a \mapsto t])$. By inductive hypothesis
  $a \notin fv(u'[a \mapsto t])$ and result follows.

A simple lemma

$$a \notin fv(t) \quad \Longrightarrow \quad a \notin fv(u[a{\mapsto}t]).$$

Proof by induction on u:

- Suppose $u \equiv \lambda b.u'$ and $b \in fv(t)$. Then
  $(\lambda b.u')[a{\mapsto}t] \equiv \lambda b'.(u'[b{\mapsto}b'][a{\mapsto}t]).$

Cannot apply the inductive hypothesis since $u'[b{\mapsto}b']$ is not a subterm of $\lambda b.u'$.

One solution: reason by induction on the size (length) of terms.

$Иa.\phi(a)$ means...

- There is some finite set S such that $\phi(a)$ need not hold for $a \in S$.

- However $\phi(a)$ holds for all $a \notin S$.

This enables us to pick a fresh element, without precise reference to what exactly it is fresh for.

For example

1. $Иa.(a \neq b)$,

2. $Иa.(a \notin \{b, c, d\})$, and

3. $\neg Иa.(a \notin \text{set of variable symbols})$

all hold.

Example definition of α-equivalence

$$a \approx_\alpha a \qquad \frac{s \approx_\alpha s' \quad t \approx_\alpha t'}{st \approx_\alpha s't'}$$

$$\frac{\mathsf{И}c.\ s[a{\mapsto}c] \approx_\alpha s[b{\mapsto}c]}{\lambda a.s \approx_\alpha \lambda b.t}$$

Ⅴ only good for predicates

This notion of 'is fresh' does not help us directly to choose b' as e.g. in the 'b' fresh' in the definition of explicit substitution.

This is because explicit substitution was a function — the result depends on which b' we choose. All is not lost.

A word on sets

Standard models of set theory are well-founded trees where the parent-daughter relation is exactly set-membership $\in$.

Set theory is more than that because we cleverly choose axioms to define such a model.

This is a separate issue from what to do if somebody you trust gives you a set.

It's a well-founded tree.

Fix some collection $\mathbb{A}$ of atoms a, b, c. Let an FM set universe be well-founded trees with leaves labelled by atoms or $\emptyset$.

- This is the tree which is a node labelled by a: 'a'.

- This is the tree with a node labelled by $\emptyset$: '$\emptyset$'.

- This is a tree with two daughters: '$\{\emptyset, a\}$'.

Let x, y vary over FM sets.

Define the FM set permutation action on sets by:

$$(b\,a) \cdot a = b \qquad (b\,a) \cdot b = a \qquad (b\,a) \cdot c = c$$

$$(b\,a) \cdot x = \{(b\,a) \cdot y \mid y \in x\}$$

I.e. 'swap b and a in x'.

Note that $(b\,a) \cdot (x \setminus y) = (b\,a) \cdot x \setminus (b\,a) \cdot y$.

Write

$$a \# x \qquad \text{when} \qquad \text{И}b.(b\ a) \cdot x = x.$$

1. $a\#b$ is true. $(c\ a) \cdot b = b$ for $c \notin S = \{b\}$.

2. $a\#a$ is false. For no finite S is it the case that for $b \notin S$, $(b\ a) \cdot a = a$.

3. $a\#\mathbb{A}$ is true. $(c\ a) \cdot \mathbb{A} = \mathbb{A}$ always.

4. $a\#\mathbb{A} \setminus \{a\}$ is false. $(b\ a) \cdot (\mathbb{A} \setminus \{a\}) = \mathbb{A} \setminus \{b\}$.

5. $a\#\mathbb{A} \setminus \{b, c\}$ is true. Take $S = \{b, c\}$.

Write

$$[a]x = \{(b, (b\,a) \cdot x) \mid b\#x \vee b = a\}.$$

This is the graph of a partial function which we can write as

$$[a]x = \lambda b. \begin{cases} (b\,a) \cdot x & b\#x \vee b = a \\ \bot & \text{otherwise.} \end{cases}$$

The 'traditional' model of abstraction is 'a function-set'. A function-set is a set satisfying a predicate which expresses 'this set is a set of pairs such that the first projections of two different pairs, are different'.

$[a]x$ is another model of abstraction which models α-equivalence.

The proof is in some facts:

$$\mathsf{И}c.(c\,a)\cdot x = (c\,b)\cdot y \quad \text{if and only if} \quad [a]x = [b]y.$$

E.g. this gives us a rule:

$$\frac{\mathsf{И}c.(c\,a)\cdot x = (c\,b)\cdot y}{[a]x = [b]y}$$

Also, $a\#[a]x$ and $a\#[b]x$ if and only if $a\#x$.

Substitution in the λ-calculus (FM)

Fix atoms. Let terms be defined by:

$$t ::= a \mid tt \mid \lambda[a]t.$$

Henceforth let t, u be a metavariable ranging over terms (terms are now abstract syntax trees in FM sets).

Define free variables 'magically' by:

$$fv(t) = \{a \mid \neg(a \# t)\}.$$

Don't fix a choice of fresh variable for each finite set of variables.

Let substitution take an abstraction of a term and a term $sub([a]u, t)$.
Sugar this to

$$u[a \mapsto t].$$

Define it by:

$$a[a \mapsto t] \equiv t$$

$$a[b \mapsto t] \equiv a$$

$$(uu')[a \mapsto t] \equiv (u[a \mapsto t])(u'[a \mapsto t])$$

$$(\lambda[b]u)[a \mapsto t] \equiv \lambda[b](u[a \mapsto t]) \qquad b \# t$$

Last clause can also be written as:

$$\forall a, t, \hat{u}. \; \text{И}b. \; (\lambda \hat{u})[a \mapsto t] \equiv \lambda[b]((\hat{u}b)[a \mapsto t]).$$

We still have an inductive principle

Suppose that:

- $\phi(a)$.

- If $\phi(t)$ and $\phi(t')$ then $\phi(tt')$.

- If $\phi(u)$ then $\phi(\lambda[a]u)$.

Then $\forall t.\ \phi(t)$.

Look at the inductive principle more closely

In practice ϕ the inductive hypothesis depends on some parameters; we should write it $\phi(t, x_1, \ldots, x_n)$. Write just $\phi(t, z)$ for short.

When we write

- If $\phi(u)$ then $\phi(\lambda[a]u)$.

do we know $a \# z$? If not, perhaps there is some special interaction between a and z which stops us from renaming a.

I claim that this is the essence of 'accidental name-clash'.

Better inductive principle

Suppose that:

- $\phi(a)$.

- If $\phi(t)$ and $\phi(t')$ then $\phi(tt')$.

- $\unicode{x019B}a.\forall u.\ \phi(u) \Rightarrow \phi(\lambda[a]u)$.

Then $\forall t.\ \phi(t)$.

Now we do not care about parameters — $\unicode{x019B}$ makes sure that $a\#z$.
Here $\phi(s)$ is really $\phi(s, z)$ and the last example is really $\forall z.\unicode{x019B}a.\forall u.\text{-}$.

What is the connection between the two principles? What if we choose some a and later want to guarantee that $a\#z$ for some other parameter? I.e. we have proved

$$\phi(t)$$

and now want to know

$$\phi(t) \wedge \psi(t, z)$$

but perhaps $\psi(t, z)$ is only true because $\neg(a\#z)$ for some a we used to construct t.

At this point we normally (informally) say 'we silently rename a to avoid clash with z'.

For any predicate ϕ

$$\phi(x_1, \ldots, x_n) \Leftrightarrow \phi((b\ a) \cdot x_1, \ldots, (b\ a) \cdot x_n).$$

Proof:

1. $x \in y$ if and only if $(b\ a) \cdot x \in (b\ a) \cdot y$.

2. $x = y$ if and only if $(b\ a) \cdot x = (b\ a) \cdot y$.

3. $(b\ a) \cdot \mathbb{A} = \mathbb{A}$.

Now given

$$\phi(t)$$

we conclude

$$\phi((b\ a) \cdot t).$$

For example in inductive reasoning we may write

> The property 't has the inductive hypothesis' has one free variable, t.

> So from our assumption that t has the inductive hypothesis, we also know that $(b\ a) \cdot t$ has the inductive hypothesis where b is chosen fresh.

We have formal license to rename variable names in inductive hypotheses.

We do not need to switch to induction on a coarser measure invariant under renaming, such as length.

A simple lemma

$$\forall a, t. \quad a \# t \quad \implies \quad a \# (u[a \mapsto t]).$$

Immediate from general FM nonsense since $a \# [a]u$ and $a \# t$ so $a \# sub([a]u, t)$.

A simple lemma

Proof by induction on u:

- Suppose $u \equiv a$. Then $a[a \mapsto t] \equiv t$.

- Suppose $u \equiv b$. Then $b[a \mapsto t] \equiv b$.

- Suppose $u \equiv u'u''$. Then
  $(u'u'')[a \mapsto t] \equiv (u'[a \mapsto t])(u''[a \mapsto t])$. Now $a \# u'$ so
  $a \#(u'[a \mapsto t])$ by inductive hypothesis. Similarly for u''. Result
  follows.

Proof by induction on u:

Suppose $u \equiv \lambda[a']u'$ and suppose the inductive hypothesis of u':

$$\forall a, t. \ a \# t \quad \Longrightarrow \quad a \#(u'[a \mapsto t]).$$

So suppose $a \# t$. We want to show

$$a \#(\lambda[a']u')[a \mapsto t].$$

We would like to say 'well,

$$(\lambda[a']u')[a \mapsto t] \equiv \lambda[a'](u'[a \mapsto t])$$

so we use the inductive hypothesis'.

But we can't, because we do not know $a' \# t$.

Choose a'' fresh (so $a'' \# a', a, u, u', t$). Then $u \equiv \lambda[a''](a''\ a') \cdot u'$ and we have the inductive hypothesis of $u'' \equiv (a''\ a') \cdot u'$:

$$\forall a, t. \ a \# t \quad \Longrightarrow \quad a \# u''[a \mapsto t].$$

Then $(\lambda[a'']u'')[a \mapsto t] \equiv \lambda[a''](u''[a \mapsto t])$.

By inductive hypothesis $a \# (u''[a \mapsto t])$ and result follows.

Conclusions

Abstraction does not necessarily equal functional abstraction.

Sometimes it may be convenient to use FM abstraction.

You get α-equivalence, inductive reasoning principles, even logic for picking fresh atoms (the N quantifier) and predicates (freshness $\#$) for saying when something is and is not fresh.

And you get FM equivariance which lets you have your inductive hypothesis in the cases when 'morally' you obviously should have.

(Even for ordinary syntax without FM abstraction.)

Conclusions

This material is covered better than I have presented it here, in papers and tools by Pitts, Urban, Cheney, and Shinwell, who are really trying to make this stuff accessible.

Of course I have lots of stuff on it too. 'Fresh Logic', 'A sequent calculus for nominal logic', 'A new approach to abstract syntax with binding', my thesis — as well as work with Fernández on Nominal Rewriting, with Aad Mathijssen on Nominal Universal Algebra.

Also a variety of crazy stuff available, such as a-logic and the NEW Calculus of Contexts.

Even more conclusions

This may be nonsense but I'll try saying it anyway:

It is standard to enrich syntax with constructs referring to models.

- Modal logic.

- Types.

- Calculi of explicit substitutions.

- μCRL.

- …

Less common is to enrich models with constructs from syntax.

FM arose out of that. Non-trivial and subtle mathematics.