Fraenkel-Mostowski atoms model variables as well as names

Murdoch J. Gabbay, Heriot-Watt University, Scotland

SPLS, Edinburgh University
Friday 9 March 2007

Thanks to Kenneth MacKenzie

Names

Fraenkel-Mostowski set theory models names.

A Fraenkel-Mostowski set z is either an atom (a name)

$$a \quad b \quad c \quad \ldots$$

or a proper set

$$\{\} \quad \{a\} \qquad \{a, b, c, d, \ldots\} = \mathbb{A} \qquad \{a, \{\}, \{a, \{b\}\}\}.$$

Examples of names

Variable names in abstract syntax, x, y, z, . . .

Names like Xavier, Yoda, Zorro, . . .

Pointers are also names; they name the memory location they point to.

Procedure names name procedures.

Ports (like http port 80) name services.

IP addresses name computers.

And so on.

Fraenkel-Mostowski sets model trees

Write (x, y) for $\{\{x\}, \{x, y\}\}$ and call this the pairset.

This is interesting because $(x, y) = (x', y')$ implies $x = x'$ and $y = y'$ — pairset is injective.

We can build numbers as usual: $0 = \{\}$, $1 = 0 \cup \{0\}$, $2 = 1 \cup \{1\}$, and so on.

That's enough to build a model of labelled trees.

So we can model quite complex data structures without any real difficulty in Fraenkel-Mostowski sets.

Fraenkel-Mostowski sets model abstraction

We can model abstraction in Fraenkel-Mostowski sets by taking an equivalence class.

To represent $[a](a, b)$ or 'abstract a in (a, b)' we use the set

$$\{(a,(a,b)),\ \cancel{(b,(b,b)),}\ (c,(c,b)),\ (d,(d,b)),\ (e,(e,b)),\ \ldots\}.$$

To represent $[b](a,(b,c))$ or 'abstract b in $(a,(b,c))$' we use the set

$$\{\cancel{(a,(a,(a,c))),}\ (b,(a,(b,c))),\ \cancel{(c,(a,(c,c))),}\ (d,(a,(d,c))),\ (e,(a,(e,c))),\ \ldots\}.$$

Fraenkel-Mostowski sets model abstraction

Think of it like this: a model of $\lambda a.t$ is the set of pairs (a', t') such that $\lambda a'.t'$.

We just throw in all 'renamed variants' of the set with the atom over which we abstract, renamed.

Obviously we avoid any other atoms in the set — b in the case (a, b), and a and c in the case $(a, (b, c))$.

This idea works for all sets, not just those representing trees. Some sets are large . . .

Abstract atoms in large sets

To abstract a in

$$\mathbb{A} \setminus \{a\} = \{b, c, d, \ldots\}$$

we use this equivalence class:

$$\{(a, \mathbb{A} \setminus \{a\}),\ (b, \mathbb{A} \setminus \{b\}),\ (c, \mathbb{A} \setminus \{c\}),\ \ldots\}.$$

a is not in $\mathbb{A} \setminus \{a\}$ — but it is not in it in a very conspicuous way.

We must abstract over the ‘hole’ left by a not being in $\mathbb{A} \setminus \{a\}$.

Abstract atoms in large complicated sets

To abstract a in

$$(\mathbb{A} \setminus \{a\}, (b, c))$$

we use this equivalence class:

$$\{(a, (\mathbb{A} \setminus \{a\}, (b, c))),\ (d, (\mathbb{A} \setminus \{d\}, (b, c))), \\ (e, (\mathbb{A} \setminus \{e\}, (b, c))),\ \ldots\}.$$

We must abstract so as to avoid clashing with the b and c, which are also distinguished.

Substituting atoms in atoms and finite sets

It's easy to substitute in atoms

$$a[a \mapsto x] = x \qquad b[a \mapsto x] = b$$

and also in finite sets; if Z is finite then

$$Z[a \mapsto x] = \{z[a \mapsto x] \mid z \in Z\}.$$

Substituting atoms in large sets

What about $(\mathbb{A}\setminus\{a\})[a\mapsto x]$. What should that be?

Recall $\mathbb{A}\setminus\{a\}=\{b,c,d,\ldots\}$.

We don't want $(\mathbb{A}\setminus\{a\})[a\mapsto x]$ to equal

$$\{b[a\mapsto x],c[a\mapsto x],d[a\mapsto x],\ldots\}=\mathbb{A}\setminus\{a\}$$

because a is still conspicuous by its absence in the RHS.

What kind of substitution for a is it that leaves a conspicuous in the result?

How do we substitute for the conspicuously absent a?

Renaming atoms in sets using swappings

Atoms have a swapping action $(a\ b)$ given by

$$(a\,b)a = b \quad (a\,b)b = a \quad (a\,b)c = c \quad (a\,b)X = \{(a\,b)x \mid x \in X\}.$$

For example

$$(a\ b)(x, y) = (a\ b)\{\{x\}, \{x, y\}\} = \{\{(a\ b)x\}, \{(a\ b)x, (a\ b)y\}\}.$$

Renaming atoms in sets using any permutation

This extends to any permutation (bijective function) on atoms: write πx.

For example,

$$\pi(x, y) = \{\{\pi x\}, \{\pi x, \pi y\}\}.$$

Note that

$$\pi\mathbb{A} = \mathbb{A}$$

and

$$\pi(\mathbb{A} \setminus \{a\}) = \{\pi b, \pi c, \pi d, \ldots\} = \mathbb{A} \setminus \{\pi a\}.$$

$\|$ (yes; twoverticallinesveryclose)

Suppose $A \subseteq \mathbb{A}$ is a finite set of atoms. Write

$$\mathsf{fix}(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}.$$

$\mathsf{fix}(A)$ is the set of permutations π that fix A pointwise.

Write

$$z\|_A = \{\pi z \mid \pi \in \mathsf{fix}(A)\}.$$

For example $\mathbb{A} \setminus \{a\} = b\|_{\{a\}}$.

Substituting atoms on large sets

What about $(\mathbb{A} \setminus \{a\})[a \mapsto x]$. What should that be?

How do we substitute for the conspicuously absent a?

Note that $\mathbb{A} \setminus \{a\} = b\|_{\{a\}}$.

We define

$$\begin{aligned}(\mathbb{A} \setminus \{a\})[a \mapsto x] &= b[a \mapsto x]\|_{\emptyset} \\ &= \mathbb{A}.\end{aligned}$$

Substituting atoms on large complicated sets

Take as our large complicated set

$$\mathbb{A}\backslash\{a,b\} \cup \{(b,c)\} = \{c,d,e,f,\ldots,\ (b,c)\}.$$

This contains one equivalence class $\mathbb{A} \setminus \{a,b\}$, and one small set (b,c).

So

$$\mathbb{A}\backslash\{a,b\} \cup \{(b,c)\} = f\|_{\{a,b\}}\ \cup\ (b,c)\|_{\{b,c\}}.$$

We define

$$\begin{aligned}\big(f\|_{\{a,b\}}\ \cup\ (b,c)\|_{\{b,c\}}\big)\ &[b{\mapsto}(d,e)] \\ &= f\|_{\{a,b\}}\,[b{\mapsto}(d,e)]\ \cup\ (b,c)\|_{\{b,c\}}\,[b{\mapsto}(d,e)]\end{aligned}$$

Substituting atoms on large complicated sets ... reduced to substituting on large or finite sets

$$(\mathbb{A}\setminus\{a,b\})[b\mapsto(d,e)] = f\|_{\{a,b\}}\,[b\mapsto(d,e)] = f[b\mapsto(d,e)]\|_{\{a\}}$$
$$= f\|_{\{a\}}$$
$$= \mathbb{A}\setminus\{a\}.$$

$$(b,c)\|_{\{b,c\}}\,[b\mapsto(d,e)] = (b,c)[b\mapsto(d,e)]\|_{\{d,e,c\}}.$$
$$= ((d,e),c)\|_{\{d,e,c\}}.$$

$$(\mathbb{A}\setminus\{a,b\}\cup\{(b,c)\})\,[b\mapsto(d,e)] = (\mathbb{A}\setminus\{a\})\,\cup\,\{((d,e),c)\}.$$

The overall idea:

To calculate $Z[a{\mapsto}x]$ do the following:

- Decompose Z as $\bigcup_{i\in I} z_i\|_{A_i}$.
- Calculate $z_i\|_{A_i}$ in some ‘capture-avoiding’ manner which I have not specified, to obtain $z_i[a{\mapsto}x]\|_{A'_i}$.
- Return $\bigcup_{i\in I} z_i[a{\mapsto}x]\|_{A'_i}$.

One more example

$$\{a,\ (a,a),\ (c,c),\ \ldots\}[a\mapsto(b,b)]$$
$$= (a\|_{\{a\}}\ \cup\ (c,c)\|_{\{b\}})\ [a\mapsto(b,b)]$$
$$= a\|_{\{a\}}\ [a\mapsto(b,b)]\ \cup\ (c,c)\|_{\{b\}}\ [a\mapsto(b,b)]$$

$$a\|_{\{a\}}\ [a\mapsto(b,b)] = a[a\mapsto(b,b)]\|_{\{\}} = (b,b)$$

$$(c,c)\|_{\{b\}}\ [a\mapsto(b,b)] = (c,c)[a\mapsto(b,b)]\|_{\{b\}} = (c,c)\|_{\{b\}}.$$

So the answer is $\{(a,a),\ (b,b),\ (c,c),\ \ldots\}$.

Why?

Why?

What is a variable?

A variable x represents an 'unknown element'.

So x has no denotational reality. It merely represents something else. For example $x = y$ judges whether the elements that x and y represent, are equal. $x = y$ may be true or false, depending on what they represent.

Yet some programming constructs suggest we should generalise this.

- A pointer looks like a name; $p = q$ is false. Yet pointers point to things, and $!p = !q$ may be true or false. So a pointer has some features of a name, and some features of a variable.

What is a variable?

- Variants of PROLOG have a non-logical predicate ‘var’ to identify whether a variable x has been instantiated to a value. Useful for directing proof-search.
- Functional programming languages with first-class patterns require the names of variables to be passed as arguments (inside patterns), and then instantiated (by pattern-matching).
- Calculi of explicit substitution may pass substitutions as arguments, suggesting that a variable ‘exists’ to be substituted for!
- Object-oriented programming uses named methods.
- Module systems, e.g. in ML and Haskell, create structures with named functions.

Variables in denotation

Variables are used in the theory of unification and rewriting.

Unification and rewriting are about syntax . . . or are they?

What if they are set theory, instead?

Variables in denotation

So I think it would be interesting to develop denotations in which variables populate the denotation itself.

Also an interesting philosophical issue. Set theory is a foundational structure. We can use it to model data structures (pairs; trees; numbers), functions (graphs), . . . and variables!

Slogan: A variable is a ‘name with a substitution action’.

Fraenkel-Mostowski sets made names denotational. It turns out they make variables denotational too.

Possible immediate applications

Rewriting (on sets).

Unification (of sets).

Solutions to simultaneous equations (on sets).

Functions (as sets; z represents the function 'x maps to $z[a \mapsto x]$').

Fraenkel-Mostowski set theory generalises syntax; syntax is a tree with a substitution action; so are these sets.

Being a foundational theory, a substitution action on names in Fraenkel-Mostowski sets gives a foundation to variables.