Denotations for functional programs with names

Murdoch J. Gabbay, Heriot-Watt University, Scotland

DSG Digest, Heriot-Watt University
Thursday 15 March 2007

Thanks to Gudmund Grov

λ-calculus

The λ-calculus is a canonical programming language. It underlies functional programming; ERLANG, ML, Haskell, and friends.

All of these languages are ‘just’ λ-calculus with bells and whistles.

So the character of the λ-calculus profoundly influences the character of computer science.

Programs

Consider a Gödel encoding function

$f \in \mathbb{N} \to \mathbb{N}$ maps to $i_f \in \mathbb{N}$ (provided f is recursive).

There is a function Z which

- Takes i_f and $x \in \mathbb{N}$, and
- encodes the triple $(f(x), i_f, x)$ in $\mathbb{N}$.

Programs

From $Z(i_f, x)$ we can recover f.

But $Z(i_f, x) * Z(i_f, x)$ is not the encoding of a triple (in general). The encoding is mangled by ordinary arithmetic operations.

We can encode f in $\mathbb{N}$, but we cannot recover f from $f(x)$ because we do not apply f to x, we apply f to the value associated to x.

Substitution sets

Substitution sets contain a set $\hat{\mathbb{N}}$.

$\hat{\mathbb{N}}$ contains ordinary numbers — but also elements which are names, like a, b, c. $\hat{\mathbb{N}}$ also contains arithmetic expressions like $a * b$.

Given a function $f \in \hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}}$ and $x : \hat{\mathbb{N}}$, we can calculate $f(x)$. This is just application.

There is a function Abs which if given a name $n \in \hat{\mathbb{N}}$ and $z \in \hat{\mathbb{N}}$, returns $\lambda x.z[n \mapsto x] \in \hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}}$.

$Abs(n, f(n)) = f$.

$Abs(n, f(n)) * Abs(n, f(n)) = \lambda x.f(x)^2$ — no mangling.

Substitution sets

There is a function $Unif$ which takes x and y in $\hat{\mathbb{N}}$ and returns a set of substitutions, write one σ, in $\hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}}$ such that $x\sigma = y\sigma$.

For example $\lambda x.(x[a \mapsto c][b \mapsto 2]) \in Unif(a * b, c * 2)$.

Why put variables in a denotation?

Many programming languages require like this. For example:

- In a language with linking you want to create a program with a ‘free name n’ and then explicitly connect n to a value, say 2 — later.

  This is similar to $Abs(n, f(n))2$.

- In a language with explicit pattern-matching, you want to pass a pattern as an argument to a function, and then you want to instantiate the names in it (when you pattern-match).

  This is similar to $Unif$.

Why put variables in a denotation?

- Logic programming has ‘non-logical’ predicates like ‘`isvar`’ which identify when a variable has been instantiated.

  This is just a function from $\hat{\mathbb{N}}$ to $\mathbb{B}$ which returns $\top$ if $x \in \hat{\mathbb{N}}$ is equal to a name n.

The model: datatypes

What is this crazy model?

Fix some set of names (or atoms) $\mathbb{A} = \{a, b, c, \ldots\}$.

Write $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

Let $\mathbb{A} \Rightarrow \mathbb{N}$ be the set of functions κ from $\mathbb{A}$ to $\mathbb{N}$ which are constant for 'most' atoms.

So there is some finite $S \subseteq \mathbb{A}$ such that

$$\text{if } a, b \notin S \text{ then } \kappa(a) = \kappa(b).$$

The model: datatypes

Write $\hat{\mathbb{N}}$ for the set of functions z from $(\mathbb{A} \Rightarrow \mathbb{N})$ to $\mathbb{N}$ which ignore 'most' atoms.

So there is some finite $S \subseteq \mathbb{A}$ such that if $\kappa(a) = \kappa'(a)$ for all $a \in S$, then $z(\kappa) = z(\kappa')$.

If $a \in \mathbb{A}$ then we interpret $\hat{a} \in \hat{\mathbb{N}}$ as $\lambda\kappa.\kappa(n)$. We may write $\hat{a}$ just as a.

This is just the usual environment model.

The model: datatypes

We can define a ‘substitution action’ easily by
$z[a \mapsto x] = \lambda\kappa.z(\kappa\{a \mapsto x\kappa\})$.

Here

$$\kappa\{a \mapsto x\kappa\}(a) = x\kappa$$
$$\kappa\{a \mapsto x\kappa\}(b) = \kappa(b).$$

Functions

There is a very easy function $\hat{2} \in \hat{\mathbb{N}}$ equal to $\lambda\kappa.2$. So 'normal numbers' live in $\hat{\mathbb{N}}$.

There is a very easy function $a * b$ equal to $\lambda\kappa.\kappa(a) * \kappa(b)$. So 'algebraic expressions' live in $\hat{\mathbb{N}}$.

There is an easy function $IsName \in \hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}}$ which

- Inputs $z \in \hat{\mathbb{N}}$.
- If $z = \hat{a}$ for some $a \in \mathbb{A}$ then return $\hat{1}$.
- Otherwise return $\hat{0}$.

Functions

Then $Abs \in \hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}}$ is as follows:

- Input $x \in \hat{\mathbb{N}}$. If $x \neq \hat{a}$ for all $\mathbb{A}$ then give up and return some junk.
- Suppose $x = \hat{a}$. Input $z \in \hat{\mathbb{N}}$.
- Return the function which inputs $x' \in \hat{\mathbb{N}}$ and returns $z[a \mapsto x']$.

Check that $Abs(n, f(n)) = \lambda x.f(x)$

$Abs(\hat{n}, f(\hat{n}))$ is the function which inputs $x' \in \hat{\mathbb{N}}$ and returns $f(\hat{n})[n \mapsto x']$.

So we need to check that $f(\hat{n})[n \mapsto x'] = f(x')$.

$(f(\hat{n})[n \mapsto x'])(\kappa) = f(\hat{n})(\kappa\{n \mapsto x'\kappa\})$.

$f(\hat{n})(\kappa\{n \mapsto x'\kappa\})$ is not in general equal to $f(x')$!

For example $IsName(\hat{n})(\kappa\{n \mapsto x'\kappa\}) = 1$, independently of x', because $IsName(\hat{n}) = \hat{1} = \lambda\kappa'.1$.

Check that $Abs(n, f(n)) = \lambda x.f(x)$

But if f is a function of the form $\lambda x.\lambda \kappa.f'(x\kappa)$ — that is if f comes from a 'normal function on numbers' then

$$f(\hat{n})(\kappa\{n \mapsto x'\kappa\}) = f'(\hat{n}(\kappa\{n \mapsto x'\kappa\})) = f'(x'\kappa)$$

as required.

So this is a denotation populated by 'ordinary functions on numbers' but enriched with extraordinary functions which manipulate names.

The beauty is, we can do anything with names we can imagine; it's just all there in $\hat{\mathbb{N}} \Rightarrow \hat{\mathbb{N}}$ and in the other function-spaces.

What needs to be done next

We need to look at some calculi out there and use this model to interpret the names.