On substitution

Murdoch J Gabbay

Tel Aviv University, Israel 11/5/2006

This talk. . .

. . . is the second in a series of about four talks in the framework of a mini-course describing (some? most?) of the mathematics I've done over the past six years (since I got my PhD).

Motivation

In this talk I'll discuss substitution. Let us specify this in 'nominal style':

Theory of substitution SUB

$$\mathsf{f}(X_1,\ldots,X_n)[a{\mapsto}T] = \mathsf{f}(X_1[a{\mapsto}T],\ldots,X_n[a{\mapsto}T])$$

$$b\#T \vdash ([b]X)[a{\mapsto}T] = [b](X[a{\mapsto}T])$$

$$a[a{\mapsto}T] = T$$

$$a\#X \vdash X[a{\mapsto}T] = X$$

$$b\#X \vdash X[a{\mapsto}b] = (b\,a)X$$

Nominal Terms

Nominal terms are a syntax inductively generated by

$$t ::= a \quad | \quad \pi X \quad | \quad [a]t \quad | \quad \mathsf{f}(t, \ldots, t).$$

Here:

- We fix $a, b, c, \ldots \in \mathbb{A}$ a countably infinite set of atoms.
- We fix $X, Y, Z, \ldots \in \mathbb{V}$ a countably infinite set of unknowns (disjoint from the atoms; everything's disjoint).
- We fix f, g, ...some term-formers.
- Call $[a]t$ an abstraction.

Nominal Terms (term-formers)

Explicit substitution is a term-former Σ. We write

$$\Sigma([a]u, t) \quad \text{as} \quad u[a \mapsto t].$$

Other possible term-formers (that's the f) are:

- $+$ as in $t + u$.
- λ as in $\lambda[a]t$.
- $\forall$ as in $\forall[a]t$.
- Q as in $\mathsf{Q}([a]t, [b]u)$ where $a\#u$ and $b\#t$ (modelling a 'simultaneous' quantifier).

Nominal Terms (permutations)

$$t ::= a \quad | \quad \pi X \quad | \quad [a]t \quad | \quad \mathsf{f}(t, \ldots, t).$$

π is a permutation. A permutation is a finitely supported bijection on $\mathbb{A}$.

Finitely supported means:

$\pi(a) = a$ for all $a \in \mathbb{A}$ except for a finite set of atoms.

Nominal Terms

For example permutations are:

$$(a\ b\ c) \quad \text{and} \quad \mathbf{Id}$$

(a to b to c to a, and the identity function). Permutations are not:

$$(a_1\ a_2)(a_3\ a_4)\ldots$$

for $\mathbb{A} = \{a_1, a_2, \ldots\}$.

$(a\ b) \cdot t$ means 'swap a and b in t'. For example, $(a\ b) \cdot [a]b \equiv [b]a$.

$(a\ b)X$ means 'we do not know what X is yet, but when we do, we will swap a and b in it'.

Freshness derivation rules (formally)

$$\frac{}{a\#b}\,(\#ab) \qquad \frac{a\#t_1 \;\cdots\; a\#t_n}{a\#\mathsf{f}(t_1,\ldots,t_n)}\,(\#f)$$

$$\frac{}{a\#[a]t}\,(\#[]a) \qquad \frac{a\#t}{a\#[b]t}\,(\#[]b) \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi X}\,(\#X)$$

Core equality derivation rules (formally)

$$\frac{}{t = t}\,(refl) \qquad \frac{t = u}{u = t}\,(symm) \qquad \frac{t = u \quad u = v}{t = v}\,(tran)$$

$$\frac{t = u}{C[t] = C[u]}\,(cong) \qquad \frac{a\#t \quad b\#t}{(a\ b) \cdot t = t}\,(perm)$$

For example

$$\dfrac{\dfrac{}{a\#b}\,(\#ab) \qquad \dfrac{}{a\#b}\,(\#ab)}{[a]a = [b]b}\,(perm) \qquad (b\ a)\cdot[b]b \equiv [a]a$$

$$\dfrac{\dfrac{\dfrac{b\#X}{b\#[a]X}\,(\#[]a) \qquad \dfrac{}{a\#[a]X}\,(\#[]a)}{[b](b\ a)X = [a]X}\,(perm) \qquad (b\ a)\cdot[a]X \equiv [b](b\ a)X$$

Here $\equiv$ is syntactic identity.

Syntactic model

The idea of SUB, the theory of substitution, can easily be given a concrete model based on syntax — the syntax of nominal terms themselves.

Call a term t a ground term when

- t does not mention explicit substitution (so $a[a \mapsto t']$ is not ground)
- t does not contain unknowns (so a is ground but $[a]X$ is not).

Syntactic model

The model $\mathcal{T}$ consists of ground terms, quotiented by provable equality in SUB. If t is any term write $[\![t]\!]$ for the intersection of the equivalence class of t, with the set of ground terms.

Interpret substitution by $[\![u]\!][a \mapsto [\![t]\!]] = [\![u[a \mapsto t]]\!]$.

Syntax as a model of substitution

This is capture-avoiding substitution. Let's do an example. Suppose one term-former λ. Then:

$$[\![\lambda[a]a]\!] = [\![\lambda[b]b]\!]$$

(recalling the derivation that $[a]a = [b]b$ and recalling $(cong)$).

So α-equivalence is 'for free' from the core theory of equality.

Syntax as a model of substitution

Also

$$[\![(\lambda[a]b)[b{\mapsto}a]]\!] = [\![\lambda(([a]b)[b{\mapsto}a])]\!].$$

However,

$$[\![\lambda(([a]b)[b{\mapsto}a])]\!] \neq [\![\lambda[a](b[b{\mapsto}a])]\!]$$

because the side-condition $a\#a$ is not derivable (recalling the rules; we can derive $a\#b$ but not $a\#a$).

However, we have α-renaming so

$$[\![\lambda(([a]b)[b{\mapsto}a])]\!] = [\![\lambda[a'](b[b{\mapsto}a])]\!].$$

Thus

$$[\![(\lambda[a]b)[b{\mapsto}a]]\!] = [\![\lambda[a']a]\!].$$

Syntax as a model of substitution

This (and some more simple calculations) show that ground terms up to provable equality, with substitution interpreted as . . . capture-avoiding substitution, is a sound model for our axioms.

Capture-avoiding distributivity

$$a\#Y \vdash_{\mathsf{SUB}} Z[a\mapsto X][b\mapsto Y] = Z[b\mapsto Y][a\mapsto X[b\mapsto Y]]$$

Write σ for $[b\mapsto Y]$; use unsugared syntax for other substitutions.

$$\dfrac{\dfrac{}{\Sigma([a]Z,X)\sigma = \Sigma(([a]Z)\sigma, X\sigma)} \quad \dfrac{\dfrac{a\#Y}{([a]Z)\sigma = [a](Z\sigma)} \quad \dfrac{}{X\sigma = X\sigma}}{\Sigma(([a]Z)\sigma, X\sigma) = \Sigma([a](Z\sigma), X\sigma)}}{\Sigma([a]Z,X)\sigma = \Sigma([a](Z\sigma), X\sigma)}$$

So: substitution is just another term-former.

Recall SUB

$$\mathsf{f}(X_1,\ldots,X_n)[a{\mapsto}T] = \mathsf{f}(X_1[a{\mapsto}T],\ldots,X_n[a{\mapsto}T])$$

$$b\#T \vdash ([b]X)[a{\mapsto}T] = [b](X[a{\mapsto}T])$$

$$a[a{\mapsto}T] = T$$

$$a\#X \vdash X[a{\mapsto}T] = X$$

$$b\#X \vdash X[a{\mapsto}b] = (b\,a)X$$

More example derivations (sketched)

$$b\#Z \vdash_{\mathsf{SUB}} Z[a \mapsto X] = ((b\ a)Z)[b \mapsto X]$$

This is $b\#Z \vdash_{\mathsf{SUB}} \Sigma([a]Z, X) = \Sigma([b]((b\ a)Z, X)$.

Using $(congf)$, $(refl)$, and $(symm)$, it suffices to derive $[b](a\ b)Z = [a]Z$. Using $(perm)$ it suffices to derive $a, b\#[a]Z$, which is easy.

More example derivations (sketched)

$\vdash_{\mathsf{SUB}} X[a{\mapsto}a] = X$

Use (fr) to give ourselves $b\#X$ for fresh $b \notin X, X[a{\mapsto}a]$.

By $(tran)$ it suffices to derive

$$X[a{\mapsto}a] = ((b\ a)X)[b{\mapsto}a]$$
$$((b\ a)X)[b{\mapsto}a] = X.$$

First is an instance of previous example. For the second, it suffices to derive $a\#(b\ a)X$; this follows from $b\#X$.

Conservativity

Theorem: SUB is conservative over CORE.

That is,

$$\Delta \vdash_{\mathsf{SUB}} t = u \quad \text{if and only if} \quad \Delta \vdash_{\mathsf{CORE}} t = u,$$

assuming that neither t nor u mention explicit substitution.

(CORE is the core derivation system, in essence α-equivalence up to abstracted atoms.)

Completeness with respect to the ground term model

Theorem: SUB is complete with respect to the ground term model.

That is,

- If $[\![t\sigma]\!] = [\![u\sigma]\!]$ for all closing σ such that $\vdash \Delta\sigma$
  (σ maps unknowns to ground terms and its action extends naturally, e.g. $\{a\#X\}\sigma = \{a\#\sigma(X)\}$),
- then $\Delta \vdash_{\mathsf{SUB}} t = u$.

So SUB is all of substitution.

Why are these results hard?

Substitution has the character of a computation, which suggests we recast it as a rewrite system and prove confluence; a standard prelude to results such as those mentioned.

But also it has a ‘simultaneous’ character, e.g.

$$\vdash_{\mathsf{SUB}} X[a{\mapsto}a'][b{\mapsto}b'][c{\mapsto}c'] = X[c{\mapsto}c'][b{\mapsto}b'][a{\mapsto}a']$$

is derivable but there is no obvious direction to the equality.

Solution to the problem

Dissect SUB as a rewrite system and an equational system, as follows:

SUBfr, a nominal rewrite system

$$
\begin{array}{lll}
(Rvar) & \vdash a[a{\mapsto}X] & \rightarrow X \\
(R\#) \quad a\#Z & \vdash Z[a{\mapsto}X] & \rightarrow Z \\
(Rf) & \vdash \mathsf{f}(Z_1,\ldots,Z_n)[a{\mapsto}X] & \rightarrow \mathsf{f}(Z_1[a{\mapsto}X],\ldots,Z_n[a{\mapsto}X]) \\
 & & (\mathsf{f}{\neq}\mathsf{sub}) \\
(Rsub) \quad a\#Y & \vdash Z[a{\mapsto}X][b{\mapsto}Y] & \rightarrow Z[b{\mapsto}Y][a{\mapsto}X[b{\mapsto}Y]] \\
(Rabs) \quad c\#X & \vdash ([c]Z)[a{\mapsto}X] & \rightarrow [c](Z[a{\mapsto}X]) \\
(Rren) \quad b\#Z & \vdash Z[a{\mapsto}b] & \rightarrow (b\ a)Z
\end{array}
$$

Recall SUB

$$\mathsf{f}(X_1,\ldots,X_n)[a{\mapsto}T] = \mathsf{f}(X_1[a{\mapsto}T],\ldots,X_n[a{\mapsto}T])$$

$$b\#T \vdash ([b]X)[a{\mapsto}T] = [b](X[a{\mapsto}T])$$

$$a[a{\mapsto}T] = T$$

$$a\#X \vdash X[a{\mapsto}T] = X$$

$$b\#X \vdash X[a{\mapsto}b] = (b\,a)X$$

The problem with SUBfr...

...is that we can use $(Rsub)$ forever; it is not directed and worse, it makes terms larger.

SUBe

So we work up to a provable equality a bit stronger than CORE (but much weaker than SUB):

$$(Eswap) \qquad a\#Y, b\#X \vdash Z[a{\mapsto}X][b{\mapsto}Y] = Z[b{\mapsto}Y][a{\mapsto}X]$$
$$(Egarbage) \qquad a\#Z \vdash Z[a{\mapsto}X] = Z$$

Assume $a\#Y$ and apply $(Rsub)$ twice to $Z[a{\mapsto}X][b{\mapsto}Y]$, we get

$$Z[a{\mapsto}X[b{\mapsto}Y]][b{\mapsto}Y[a{\mapsto}X[b{\mapsto}Y]]].$$

But $a\#Y$ so the RHS simplifies using $(Egarbage)$ to

$$Z[a{\mapsto}X[b{\mapsto}Y]][b{\mapsto}Y].$$

Further rewrites are equal using $(Eswap)$.

We are done.

It is relatively easy to prove that SUBe is conservative over CORE, and that if $\Delta \vdash_{\mathsf{SUBe}} t = u$ then t and u are interrewritable in SUBfr.

It is also easy to show SUBfr locally confluent (nontrivial critical pairs are joinable; uses results from Nominal Rewriting).

It is, finally, possible to show that rewriting in SUBfr is strongly normalising up to equality in SUBe, using a highly cunning measure.

Semantics

So we have a sound and complete axiomatisation of something which is recognisably capture-avoiding substitution on a concrete syntax-based model.

But the axioms are quite abstract — what other models are there?

Example models

On $\mathbb{A}$ interpret substitution by itself, namely:

$$x[a{\mapsto}y] = y \qquad (x = a \text{ or } x = y)$$
$$x[a{\mapsto}y] = x \qquad (x \neq a \text{ and } x \neq y).$$

Let $\mathbb{L}$ be the set of finite lists of atoms, e.g. $[]$ and $[a, b, b]$.

Interpret substitution by list insertion, e.g.

$$[a, b, c][b{\mapsto}[a, b, c]] = [a, a, b, c, c]$$
$$[a, a, b][b{\mapsto}[b, a]] = [a, a, b, a].$$

So far, so simple

Those were quite simple substitution actions which are still basically syntax (based on trees).

Say a set of atoms is cofinite when its complement in the set of all atoms is finite.

Write $\mathcal{P}_{fs}$ for the set of finite or cofinite sets.

If $U, V \in \mathcal{P}_{fs}$ write $U \mathsf{X} V$ for the elements in precisely one of U and V (exclusive or).

Non-syntactic models of substitution

Note that $a\#U$ (meaning Иb. $(b\ a)U = U$) when

- U is finite and a is in U, or
- U is cofinite and a is in $\mathbb{A} \setminus U$.

Interpret substitution by:

$$W[a{\mapsto}U] = W \quad \text{if } a\#W\text{, and otherwise...}$$

$$W[a{\mapsto}U] = (W\mathsf{X}\{a\})\mathsf{X}U.$$

Non-syntactic models of substitution

Note that this model is not syntax-based. Observe for example that

$$\{a, b\}[a \mapsto \{b\}] = \{\}.$$

Not what you'd expect of a syntactic model — atoms can 'explode' out of existence.

Non-syntactic models of substitution

λ-terms quotiented by $\alpha\beta$-equivalence model substitution.

This model is syntax-based but the equivalence classes are extremely complex. P under application has the same computational content as Pa under substituting for a, for some fresh $a \notin P$.

The environment model

Take an 'ordinary' set, such as $\mathbb{N}$. Then

$$(\mathbb{A} \Rightarrow \mathbb{N}) \Rightarrow \mathbb{N}$$

is a model of substitution, interpreting $\mu[a \mapsto \tau]$ by

$$\lambda\kappa \in (\mathbb{A} \Rightarrow \mathbb{N}).\ \mu(\kappa\{a \mapsto \tau\kappa\}).$$

Write this set $\hat{\mathbb{N}}$.

This is the correct notion of '$\mathbb{N}$ with atoms and a substitution added'.

An interesting element of $\hat{\mathbb{N}}$ is $\lambda\kappa.\kappa(a) * \kappa(b)$. This behaves like '$a * b$', our old friend from school/kindergarten.

Function models

Yes! Functions between models of substitution, are models of substitution. We can write $h[a \mapsto f]$ and it really does mean something.

Holy functions.

Function models

Fix $h, f \in \mathbb{X} \Rightarrow \mathbb{Y}$ and $a \in \mathbb{A}$.

Define:

$$h[a \mapsto b]x = ((b\ a)h)x \qquad \text{If } b\#h$$
$$(h[a \mapsto f])x = И a'.\ (((a'\ a)h)x)[a' \mapsto f\,x] \qquad \text{Otherwise.}$$

Here b in $h[a \mapsto b]$ is the interpretation of b in $\mathbb{X} \Rightarrow Y$, which is $\lambda x.b$.

Here if $F(a')$ is a function then $И a'.\ F(a')$ is the unique value of F for 'most' (all but finitely many) a', if this exists.

Examples

$t \mapsto [a]t$ is a function in $\mathbb{X} \Rightarrow [\mathbb{A}]\mathbb{X}$ mapping x to $[a]x$.

If $\mathbb{X}$ is the ground term model for the syntax of the λ-calculus (call it Λ) then we can compose with term-former λ in $[\mathbb{A}]\mathbb{X} \Rightarrow \mathbb{X}$ to obtain $t \mapsto \lambda a.t \in \Lambda \Rightarrow \Lambda$.

Examples

If $x \in \mathbb{X}$ then 'the substitution $[a \mapsto x]$' can be represented as $\lambda z.z[a \mapsto x] \in \mathbb{X} \Rightarrow \mathbb{X}$.

So not only do we have a theory of substitution; we can represent it explicitly as an element of a larger (functional) model of substitution.

Examples

Elements can be considered as functions over their atoms, yet also as data.

Thus we can write $x\big[a{\mapsto}y\big]$ and $y\big[a{\mapsto}x\big]$; there is no sense in which x is a priori the ‘master’ (like a function in the simply-typed λ-calculus) and y is a priori the ‘slave’ (its argument, of lower type).

Examples

$\lambda\kappa.\kappa(a) * \kappa(b) \in \hat{\mathbb{N}}$ represents $(\lambda n.\lambda m.n * m)$ if we use substitutions instead of function application to instantiate.

Substitutions $[a \mapsto 5]$ and $[b \mapsto 6]$ can arrive in any order.

Examples

Let $\mathbb{B}$ be a two-element set $\{\top, \bot\}$.

$\#$ in $(\mathbb{A} \times \mathbb{X}) \Rightarrow \hat{\mathbb{B}}$ is such that $\#(a, x) = \lambda\kappa.\top$ when $a\#x$ and otherwise $\#(a, x) = \lambda\kappa.\bot$.

Using $\#$, we can case-split on atoms which have not yet been substituted for, and perhaps (in some programming language based on this semantics) take appropriate action to fetch their values. Hugely flexible treatment of unknowns!

Final example

An ‘exception handler’

$$\lambda x.\texttt{if}\ a\#x\ \texttt{then}\ x\ \texttt{else}\ x' \in \mathbb{X} \Rightarrow \mathbb{X}$$

(using natural notation) treats a as an exception.

If it detects a it defaults to x'.

It is important to appreciate that the semantics propagates a for us. We do not need to program propagation of the exception upwards, i.e. make non-local changes to code to ‘chaperone’ a through intervening steps of the calculation.

Conclusions

Substitution is hard!

It is mathematically interesting!

It may have applications as an interesting new semantics for (meta-)programming!

Ask me about the associated λ-calculus, which can be interpreted in the cartesian closed category of models of substitution!