A New Approach to Abstract Syntax Involving Binders

Andrew M. Pitts

Joint work with Murdoch Gabbay

UNIVERSITY OF CAMBRIDGE

Paper: see www.cl.cam.ac.uk/users/ap/papers/

Slides: see www.cl.cam.ac.uk/users/ap/talks/

Logical frameworks for specifying/reasoning about formal languages involving binding operations

The problem:

Classical theory

> abstract syntax trees, algebraic data types, initial algebra semantics, structural recursion/induction, etc

applied to signatures involving binders yields overly concrete representations—lots of essentially routine constructions/proofs to do with renaming bound variables & capture avoiding substitution get done and re-done for each object-language on a case-by-case basis.

Logical frameworks for specifying/reasoning about formal languages involving binding operations

Desiderata:

1. Alpha-conversion part of the meta-logic, hence no need to develop it for each object logic separately.
2. Ditto for substitution.
3. Useful forms of structural recursion and induction.
4. Familiarity!—formalise existing practice, e.g.
   - support reasoning with names of bound variables
   - encompass usual, 'no-binder' theory of algebraic datatypes.

Logical frameworks for specifying/reasoning about formal languages involving binding operations

Conventional wisdom:

Use typed lambda calculi— the higher order abstract syntax (HOAS) approach. Satisfies desiderata 1 & 2. Makes 3 & 4 very difficult.

Proposal:

Use ideas from Fraenkel-Mostowski permutation model of set theory with atoms to fulfil desideratum 1, but not 2. Makes 3 & 4 possible in a really simple way.

Permutation actions

- $S_{\mathbb{A}}$ = group of permutations of c'tbly infinite set $\mathbb{A}$ (of 'atoms').
- An action of $S_{\mathbb{A}}$ on a class $\mathcal{X}$ is a function

$$\begin{array}{ccc} S_{\mathbb{A}} \times \mathcal{X} & \to & \mathcal{X} \\ (\pi, x) & \mapsto & \pi \cdot x \end{array}$$

  satisfying $id \cdot x = x$ and $\pi' \cdot (\pi \cdot x) = (\pi' \pi) \cdot x$.
- $S_{\mathbb{A}}$-class = class + action.

Examples of $S_{\mathbb{A}}$-classes

1. $\mathbb{A}$ itself, with action $\pi \cdot a = \pi(a)$.

2. Set of (abstract syntax trees for) lambda terms

$$\Lambda = \mu X \,.\, \mathsf{Var}(\mathbb{A}) \mid \mathsf{App}(X \times X) \mid \mathsf{Lam}(\mathbb{A} \times X)$$

with action
$$\begin{cases} \pi \cdot \mathsf{Var}(a) = \mathsf{Var}(\pi(a)) \\ \pi \cdot \mathsf{App}(M, M') = \mathsf{App}(\pi \cdot M, \pi \cdot M') \\ \pi \cdot \mathsf{Lam}(a, M) = \mathsf{Lam}(\pi(a), \pi \cdot M). \end{cases}$$

3. $pow(\mathcal{X})$ = subsets of $S_{\mathbb{A}}$-class $\mathcal{X}$, with action $\pi \cdot S = \{\pi \cdot x \mid x \in S\}$.

Finite support property

Let $\mathcal{X}$ be an $S_{\mathbb{A}}$-class.

$S \subseteq \mathbb{A}$ supports $x \in \mathcal{X}$ if for all $\pi \in S_{\mathbb{A}}$ that fix every element of S, we have $\pi \cdot x = x$.

x is finitely supported if exists finite $S \subseteq \mathbb{A}$ supporting x. In that case there is a least such S, the support of x, $\boxed{supp(x)}$.

Write $\boxed{a \# x}$ to mean $a \notin supp(x)$.

Examples:

- every $t \in \Lambda$ is finitely supported and $a \# t$ iff a does not occur in t.
- $S \in pow(\mathbb{A})$ is finitely supported iff it is either finite or cofinite.

Fraenkel-Mostowski universe, $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$

is the least $S_{\mathbb{A}}$-class $\mathcal{X}$ satisfying

$$\mathcal{X} \;=\; \mathbb{A} + pow_{\mathrm{fs}}(\mathcal{X})$$

where $pow_{\mathrm{fs}}(\mathcal{X})$ is the sub-$S_{\mathbb{A}}$-class of $\mathcal{X}$ consisting of subsets with finite support.

Axiomatics: $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ is a model of ZFA satisfying
$\mathbb{A} \notin pow_{\mathrm{fin}}(\mathbb{A})$ (‘cos $\mathbb{A}$ not finite),
$\forall x \,.\, \exists a \in \mathbb{A} \,.\, a \,\#\, x$ (‘cos every element has finite support),
and hence also
$\neg \mathrm{AC}$ (‘cos $\forall S \in pow_{\mathrm{fin}}(\mathbb{A}).\exists a \in \mathbb{A}.a \notin S$, but no choice function has finite support).

Three versions of variable-renaming for $\Lambda = \mu X\,.\,\mathsf{Var}(\mathbb{A}) \mid \mathsf{App}(X \times X) \mid \mathsf{Lam}(\mathbb{A} \times X)$

$[a'/a]M$ = capture-avoiding substitution of a' for all free occurrences of a in M ($a, a' \in \mathbb{A}$)

$\{a'/a\}M$ = textual substitution of a' for all free occurrences of a in M

$(a'\,a) \cdot M$ = interchange of *all* occurrences (be they free, bound, or binding) of a' and a in M. (Special case of $\pi \cdot M$ for $\pi \in S_{\mathbb{A}}$ the transposition $(a'\,a)$.)

Recall usual definition of α-conversion, $=_\alpha$, as smallest congruence relation on Λ containing $\mathsf{Lam}(a, M) =_\alpha \mathsf{Lam}(a', [a'/a]M)$.

Theorem. $=_\alpha$ coincides with the relation $\sim \subseteq \Lambda \times \Lambda$ inductively generated by the axioms and rules

$$\mathsf{Var}(a) \sim \mathsf{Var}(a)$$

$$\frac{M_1 \sim M_1' \qquad M_2 \sim M_2'}{\mathsf{App}(M_1, M_2) \sim \mathsf{App}(M_1', M_2')}$$

$$\frac{(a''\, a) \cdot M \sim (a''\, a') \cdot M'}{\mathsf{Lam}(a, M) \sim \mathsf{Lam}(a', M')} \quad \text{if } a'' \# M, M'$$

A quantifier for 'freshness'

Define $\mathsf{N}a \in \mathbb{A} \,.\, \phi$ to be '$\{a \in \mathbb{A} \mid \phi\}$ is a cofinite subset of $\mathbb{A}$'.

Fact: if $fv(\phi) \subseteq \{a, \vec{x}\}$, then $fv(\mathsf{N}a \in \mathbb{A} \,.\, \phi) \subseteq \{\vec{x}\}$ and

$$\exists a \in \mathbb{A} \,.\, a \# \vec{x} \;\&\; \phi$$

$$\Leftrightarrow \;\; \mathsf{N}a \in \mathbb{A} \,.\, \phi \;\; \Leftrightarrow$$

$$\forall a \in \mathbb{A} \,.\, a \# \vec{x} \Rightarrow \phi$$

Here ϕ is a formula of ZFA and we make use of the equivariance property of such formulas:

if $fv(\phi) \subseteq \{\vec{x}\}$, then $\forall \pi, \vec{x} \,.\, (\phi(\vec{x}) \;\Leftrightarrow\; \phi(\pi \cdot \vec{x}))$.

So can read $\mathsf{N}a \in \mathbb{A} \,.\, \phi$ as

'for some/any fresh atom a, it is the case that ϕ'.

Alpha-conversion of sets in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$

Write $\boxed{[a]x}$ for $\sim$-equivalence class of $(a, x) \in \mathbb{A} \times \mathcal{V}_{\mathrm{FM}}(\mathbb{A})$, where

$$(a, x) \sim (a', x') \stackrel{\text{def}}{\Leftrightarrow} \text{И} a'' \in \mathbb{A}\,.\,(a''\,a) \cdot x = (a''\,a') \cdot x'$$

Fact: $[a]x \in \mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ with $supp([a]x) = supp(x) \setminus \{a\}$.

Define the subclass $\mathcal{A}bs(\mathbb{A}) \subseteq \mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ of $\mathbb{A}$-abstractions to be

$$\mathcal{A}bs(\mathbb{A}) \stackrel{\text{def}}{=} \{[a]x \mid a \in \mathbb{A} \,\&\, x \in \mathcal{V}_{\mathrm{FM}}(\mathbb{A})\}.$$

$\mathbb{A}$-abstractions as functions

Fact: each $f \in \mathcal{A}bs(\mathbb{A})$ is a unary functional relation, i.e.

$$(a, x) \in f \ \& \ (a, x') \in f \ \Rightarrow \ x = x'$$

with $dom(f) = \mathbb{A} \setminus supp(f)$.

Hence can apply $f \in \mathcal{A}bs(\mathbb{A})$ to any a satisfying $a \# f$ to obtain $f(a)$—a concretion of the $\mathbb{A}$-abstraction f.

Fact: each $\mathbb{A}$-abstraction is uniquely determined by some/any of its concretions: for all $f, f' \in \mathcal{A}bs(\mathbb{A})$

$$(\text{И}a \in \mathbb{A} \,.\, f(a) = f'(a)) \ \Rightarrow \ f = f'.$$

The $\mathbb{A}$-abstraction set-former, $[\mathbb{A}](-)$

Given $X \in \mathcal{V}_{\mathrm{FM}}(\mathbb{A}) \setminus \mathbb{A}$, define

$$[\mathbb{A}]X \stackrel{\mathrm{def}}{=} \{f \in \mathcal{A}bs(\mathbb{A}) \mid \text{И} a \in \mathbb{A} \, . \, f(a) \in X\}.$$

Fact: $[\mathbb{A}](-)$ is monotone for $\subseteq$ and preserves unions of countable ascending chains in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$.

Hence can use $[\mathbb{A}](-)$ in combination with $\times$ and $+$ to form inductively defined sets in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ via usual Tarski construction:

$$\mu X \, . \, F(X) = \bigcup_{n \in \mathbb{N}} F^n(\emptyset).$$

Moreover such $F(-)$ are functors ($[\mathbb{A}](-)$ extends to a functor), and $\mu X \, . \, F(X)$ is an initial algebra for it.

$\Lambda/{=_\alpha}$ is an algebraic datatype in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$

Recall: $\Lambda = \mu X\,.\,\mathsf{Var}(\mathbb{A}) \mid \mathsf{App}(X \times X) \mid \mathsf{Lam}(\mathbb{A} \times X)$.

Theorem. In $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$, quotient set of lambda terms mod alpha-conversion, $\Lambda/{=_\alpha}$, is in bijection with the inductive set

$$\Lambda_\alpha \stackrel{\text{def}}{=} \mu X\,.\,\mathsf{Var}_\alpha(\mathbb{A}) \mid \mathsf{App}_\alpha(X \times X) \mid \mathsf{Lam}_\alpha([\mathbb{A}]X).$$

Fact: Λ_α is initial algebra for the functor

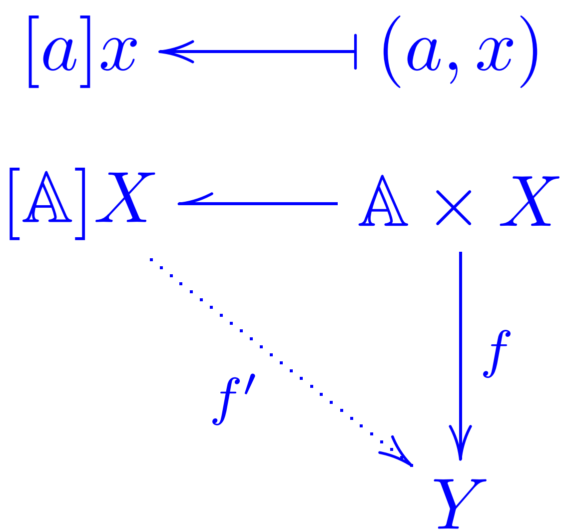
$$F(-) = \mathbb{A} + (- \times -) + [\mathbb{A}](-)$$

i.e. for every $f : F(X) \to X$ there is a unique $\overline{f} : \Lambda_\alpha \to X$ s.t. ...

To get useful structural recursion/induction principles from the initial algebra property, need to analyse nature of functions out of $[\mathbb{A}]X$ ...

Lemma

Given $f : \mathbb{A} \times X \to Y$ in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$,
$\exists! f'$ s.t. $\text{И} a \in \mathbb{A} \,.\, \forall x \in X \,.\, f'([a]x) = f(a, x)$



iff f satisfies $\text{И} a \in \mathbb{A} \,.\, \forall x \in X \,.\, a \,\#\, f(a, x)$.

Λ_α structural recursion

Given $f : \mathbb{A} \to X$, $\quad g : X \times X \times \Lambda_\alpha \times \Lambda_\alpha \to X$, and $h : \mathbb{A} \times X \times \Lambda_\alpha \to X$ in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ with h satisfying

$$(\dagger) \qquad \text{И} a \in \mathbb{A} \,.\, \forall x \in X \,.\, \forall t \in \Lambda_\alpha \,.\, a \# h(a, x, t)$$

then there is a unique $k : \Lambda_\alpha \to X$ such that

$$\forall a \in \mathbb{A} \,.\, k(\mathsf{Var}_\alpha(a)) = f(a)$$
$$\forall t, t' \in \Lambda_\alpha \,.\, k(\mathsf{App}_\alpha(t, t')) = g(k(t), k(t'), t, t')$$
$$\text{И} a \in \mathbb{A} \,.\, \forall t \in \Lambda_\alpha \,.\, k(\mathsf{Lam}_\alpha([a]t)) = h(a, k(t), t).$$

Also, $supp(k) = supp(X) \cup supp(f) \cup supp(g) \cup supp(h)$.

Example: capture-avoiding substitution

Given $a \in \mathbb{A}$ and $t \in \Lambda_\alpha$, can use structural recursion for Λ_α to define $[t/a](-)$ to be unique $k : \Lambda_\alpha \to \Lambda_\alpha$ in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ satisfying

$$\forall a' \in \mathbb{A} \,.\, k(\mathsf{Var}_\alpha(a')) = (\mathsf{if}\ a' = a\ \mathsf{then}\ t\ \mathsf{else}\ \mathsf{Var}_\alpha(a'))$$
$$\forall t', t'' \in \Lambda_\alpha \,.\, k(\mathsf{App}_\alpha(t', t'')) = \mathsf{App}_\alpha(k(t'), k(t''))$$
$$\mathsf{И} a' \in \mathbb{A} \,.\, \forall t' \in \Lambda_\alpha \,.\, k(\mathsf{Lam}_\alpha([a']t')) = \mathsf{Lam}_\alpha([a']k(t')).$$

N.B. condition (†) satisfied in this case because $a' \# [a']t'$.

Example: set of free variables

Can use structural recursion for Λ_α to deduce existence of function $fv : \Lambda_\alpha \longrightarrow pow_{\text{fin}}(\mathbb{A})$ in $\mathcal{V}_{\text{FM}}(\mathbb{A})$ satisfying

$$\forall a \in \mathbb{A} \,.\, fv(\mathsf{Var}_\alpha(a)) = \{a\}$$
$$\forall t, t' \in \Lambda_\alpha \,.\, fv(\mathsf{App}_\alpha(t, t')) = fv(t) \cup fv(t')$$
$$\forall a \in \mathbb{A} \,.\, \forall t \in \Lambda_\alpha \,.\, fv(\mathsf{Lam}_\alpha([a]t)) = fv(t) \setminus \{a\}$$

N.B. used fact that $supp(fv) = \emptyset$ to replace И by $\forall$ in last clause.

Condition (†) satisfied in this case because $a \# S \setminus \{a\}$ (any $S \in pow_{\text{fin}}(\mathbb{A})$).

Can prove $\boxed{\forall t \in \Lambda_\alpha \,.\, fv(t) = supp(t)}$

by using structural induction for $\Lambda_\alpha \ldots$

Λ_α structural induction

Given a subset $S \subseteq \Lambda_\alpha$ in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$, to prove that S is the whole of Λ_α it suffices to show

$$\forall a \in \mathbb{A} \,.\, \mathsf{Var}_\alpha(a) \in S$$

$$\forall t, t' \in S \,.\, \mathsf{App}_\alpha(t, t') \in S$$

$$\text{И} a \in \mathbb{A} \,.\, \forall t \in S \,.\, \mathsf{Lam}_\alpha([a]t) \in S.$$

Non-example: set of bound variables

There is no function $bv : \Lambda_\alpha \to pow_{\mathrm{fin}}(\mathbb{A})$ in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ satisfying

$$\forall a \in \mathbb{A} \,.\, bv(\mathsf{Var}_\alpha(a)) = \emptyset$$
$$\forall t, t' \in \Lambda_\alpha \,.\, bv(\mathsf{App}_\alpha(t, t')) = bv(t) \cup bv(t')$$
$$\forall a \in \mathbb{A} \,.\, \forall t \in \Lambda_\alpha \,.\, bv(\mathsf{Lam}_\alpha([a]t)) = \{a\} \cup bv(t).$$

(Can't use structural recursion to define bv, because condition (†) not satisfied—$a \,\#\, (\{a\} \cup S)$ fails.)

Proof. If such a bv existed, choose $a \neq a'$ not in its support. Then for $t = \mathsf{Lam}_\alpha([a]\mathsf{Var}_\alpha(a))$ have $a, a' \,\#\, bv(t)$, so

$$\{a'\} = (a'\,a) \cdot a = (a'\,a) \cdot bv(t) = bv(t) = \{a\}$$

contradicting $a \neq a'$. □

Summary

Can extend initial algebra semantics of algebraic (no-binders) signatures to Plotkin's 'binding signatures' using inductive sets in $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$.

$\exists$ other ways of achieving this (cf. recent use of presheaf categories by Fiore-Plotkin-Turi, and by Hofmann), but $\mathcal{V}_{\mathrm{FM}}(\mathbb{A})$ has some advantages:

- Our notion of abstraction co-exists with classical logic.
- Notion of finite support supports logical forms ($\#$-relation and И-quantifier) that seem to capture common informal reasoning about bound names.
- Straightforward principles of structural recursion/induction.

Further directions

'Equivariant' structural operational semantics: use of N-quantifier in inductively defined relations.

FM type theory: $\mathsf{N} a \in \mathbb{A} \,.\, (-)$ corresponds to dependently typed $\mathbb{A}$-abstraction under Curry-Howard.

$$[a \in \mathbb{A}]X(a) = \{f \in \mathcal{A}bs(\mathbb{A}) \mid \mathsf{N} a \in \mathbb{A} \,.\, f(a) \in X(a)\}$$

Metaprogramming: user-declared data types involving $[\mathbb{A}](-)$; inference about support via type system; pattern-matching with 'binding patterns' $[a]p$.