A New Approach to Abstract Syntax Involving Binders

Andrew M. Pitts

Joint work with Murdoch Gabbay

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

- $S_A = \text{group of permutations of c’tbly infinite set } A \text{ (of ‘atoms’).}$
- An action of $S_A$ on a class $\mathcal{X}$ is a function
  \[
  S_A \times \mathcal{X} \rightarrow \mathcal{X}
  \]
  \[
  (\pi, x) \mapsto \pi \cdot x
  \]
  satisfying $id \cdot x = x$ and $\pi' \cdot (\pi \cdot x) = (\pi'\pi) \cdot x$.
- $S_A$-class = class + action.
Examples of $\mathcal{S}_\mathcal{A}$-classes

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

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

$$\Lambda = \mu X . \text{Var}(\mathcal{A}) \mid \text{App}(X \times X) \mid \text{Lam}(\mathcal{A} \times X)$$

with action

$$\begin{align*}
\pi \cdot \text{Var}(a) &= \text{Var}(\pi(a)) \\
\pi \cdot \text{App}(M, M') &= \text{App}(\pi \cdot M, \pi \cdot M') \\
\pi \cdot \text{Lam}(a, M) &= \text{Lam}(\pi(a), \pi \cdot M).
\end{align*}$$

3. $\text{pow}(\mathcal{X}) = \text{subsets of } \mathcal{S}_\mathcal{A}$-class $\mathcal{X}$, with action

$$\pi \cdot S = \{ \pi \cdot x \mid x \in S \}.$$
Finite support property

Let $\mathcal{X}$ be an $S_A$-class.

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

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

Write $a \not\in x$ to mean $a \notin \text{supp}(x)$.

Examples:

- every $t \in \Lambda$ is finitely supported and $a \not\in t$ iff $a$ does not occur in $t$.
- $S \in \text{pow}(A)$ is finitely supported iff it is either finite or cofinite.
Fraenkel-Mostowski universe, $\mathcal{V}_{FM}(A)$

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

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

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

**Axiomatics:** $\mathcal{V}_{FM}(A)$ is a model of $ZFA$ satisfying

- $A \notin pow_{fin}(A)$ ('cos $A$ not finite),
- $\forall x. \exists a \in A. a \# x$ ('cos every element has finite support), and hence also
- $\neg AC$ ('cos $\forall S \in pow_{fin}(A). \exists a \in A. a \notin S$, but no choice function has finite support).
Three versions of variable-renaming for

\[ \Lambda = \mu X . \text{Var}(A) \mid \text{App}(X \times X) \mid \text{Lam}(A \times X) \]

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

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

\[ (a' a) \cdot M = \text{interchange of all occurrences (be they free, bound, or binding) of } a' \text{ and } a \text{ in } M. \quad (\text{Special case of } \pi \cdot M \text{ for } \pi \in S_A \text{ the transposition } (a' a).) \]
Recall usual definition of $\alpha$-conversion, $=\alpha$, as smallest congruence relation on $\Lambda$ containing $\text{Lam}(a, M) =_{\alpha} \text{Lam}(a', [a'/a]M)$.

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

\[
\text{Var}(a) \sim \text{Var}(a)
\]

\[
M_1 \sim M'_1 \quad M_2 \sim M'_2
\]

\[
\text{App}(M_1, M_2) \sim \text{App}(M'_1, M'_2)
\]

\[
(a'' a) \cdot M \sim (a'' a') \cdot M'
\]

\[
\text{Lam}(a, M) \sim \text{Lam}(a', M') \quad \text{if } a'' \not\# M, M'
\]
A quantifier for ‘freshness’

Define $\forall a \in \mathbb{A} . \phi$ to be ‘$\{a \in \mathbb{A} \mid \phi\}$ is a cofinite subset of $\mathbb{A}$’.

**Fact:** if $fv(\phi) \subseteq \{a, \bar{x}\}$, then $fv(\forall a \in \mathbb{A} . \phi) \subseteq \{\bar{x}\}$ and

\[
\exists a \in \mathbb{A} . a \not\equiv \bar{x} & \phi
\]

\[
\iff \forall a \in \mathbb{A} . \phi \iff \\
\forall a \in \mathbb{A} . a \not\equiv \bar{x} \Rightarrow \phi
\]

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

if $fv(\phi) \subseteq \{\bar{x}\}$, then $\forall \pi, \bar{x} . (\phi(\bar{x}) \iff \phi(\pi \cdot \bar{x}))$.

So can read $\forall a \in \mathbb{A} . \phi$ as

‘for some/any fresh atom $a$, it is the case that $\phi$’.
Alpha-conversion of sets in $\mathcal{V}_{FM}(A)$

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

$$(a, x) \sim (a', x') \overset{\text{def}}{\iff} \forall a'' \in A . (a'' a) \cdot x = (a'' a') \cdot x'$$

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

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

$$Abs(A) \overset{\text{def}}{=} \{[a]x \mid a \in A \& x \in \mathcal{V}_{FM}(A)\}.$$
\(A\)-abstractions as functions

**Fact:** each \( f \in Abs(A) \) is a unary functional relation, i.e.

\[(a, x) \in f \land (a, x') \in f \Rightarrow x = x'\]

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

Hence can apply \( f \in Abs(A) \) to any \( a \) satisfying \( a \not\in f \) to obtain \( f(a) \)—a concretion of the \( A \)-abstraction \( f \).

**Fact:** each \( A \)-abstraction is uniquely determined by some/any of its concretions: for all \( f, f' \in Abs(A) \)

\[(\forall a \in A. f(a) = f'(a)) \Rightarrow f = f'.\]
The $\mathbb{A}$-abstraction set-former, $[\mathbb{A}](\cdot)$

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

$$[\mathbb{A}]X \overset{\text{def}}{=} \{ f \in \mathcal{A}bs(\mathbb{A}) \mid \forall a \in \mathbb{A}. f(a) \in X \}.$$ 

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

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

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

Moreover such $F(\cdot)$ are functors ($[\mathbb{A}](\cdot)$ extends to a functor), and $\mu X . F(X)$ is an initial algebra for it.
\( \Lambda / \equiv_{\alpha} \) is an algebraic datatype in \( \nu_{\text{FM}}(\mathbb{A}) \)

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

**Theorem.** In \( \nu_{\text{FM}}(\mathbb{A}) \), quotient set of lambda terms mod alpha-conversion, \( \Lambda / \equiv_{\alpha} \), is in bijection with the inductive set

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

**Fact:** \( \Lambda_{\alpha} \) is initial algebra for the functor

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

i.e. for every \( f : F(X) \rightarrow X \) there is a unique \( \bar{f} : \Lambda_{\alpha} \rightarrow X \) 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 $\mathcal{V}_{FM}(\mathbb{A})$, 
\[ \exists! f' \text{ s.t. } \forall a \in \mathbb{A} . \forall x \in X . f'(a[x]) = f(a, x) \]

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

Heriot-Watt 13/04/99
\[ \Lambda_\alpha \text{ structural recursion} \]

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 \( \mathcal{V}_{FM}(\mathbb{A}) \) with \( h \) satisfying

\[(\dagger) \quad \forall a \in \mathbb{A} . \forall x \in X . \forall t \in \Lambda_\alpha . a \neq h(a, x, t) \]

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

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

Also, \( \text{supp}(k) = \text{supp}(X) \cup \text{supp}(f) \cup \text{supp}(g) \cup \text{supp}(h) . \)
Example: capture-avoiding substitution

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

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

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

Heriot-Watt 13/04/99
Example: set of free variables

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

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

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

Condition $(\uparrow)$ satisfied in this case because $a \not\in S \setminus \{a\}$ (any $S \in \text{pow}_{\text{fin}}(A)$).

Can prove $\forall t \in \Lambda_\alpha. \, fv(t) = \text{supp}(t)$ by using structural induction for $\Lambda_\alpha$. . .
Given a subset $S \subseteq \Lambda_\alpha$ in $\mathcal{V}_{FM}(\mathbb{A})$, to prove that $S$ is the whole of $\Lambda_\alpha$ it suffices to show

\[
\forall a \in \mathbb{A} \cdot \text{Var}_\alpha(a) \in S \\
\forall t, t' \in S \cdot \text{App}_\alpha(t, t') \in S \\
\forall a \in \mathbb{A} \cdot \forall t \in S \cdot \text{Lam}_\alpha([a]t) \in S.
\]
Non-example: set of bound variables

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

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

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

**Proof.** If such a \( bv \) existed, choose \( a \neq a' \) not in its support.

Then for \( t = \text{Lam}_{\alpha}([a]\text{Var}_{\alpha}(a)) \) have \( a, a' \not\in bv(t) \), so

\[
\{a'\} = (a' a) \cdot a = (a' a) \cdot bv(t) = bv(t) = \{a\}
\]

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

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

- Our notion of abstraction co-exists with classical logic.
- Notion of finite support supports logical forms (#-relation and $\forall$-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 $\forall$-quantifier in inductively defined relations.

FM type theory: $\forall a \in A . (\_)$ corresponds to dependently typed $A$-abstraction under Curry-Howard.

$$[a \in A]X(a) = \{ f \in \text{Abs}(A) \mid \forall a \in A . f(a) \in X(a) \}$$

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