Equational logic for higher-order abstract syntax

Alexander Kurz, Daniela Petrişan
University of Leicester

23rd September 2008
Domains IX
Overview

• Motivation

  ▶ Functors with finitary presentations

  ▶ Equational logic for higher-order abstract syntax

  ▶ Connection with nominal sets
Motivation

• Syntax with variable binders cannot be captured as an initial algebra for functors on Set.

• But we can do it if we move to functors on a presheaf category (Fiore, Plotkin and Turi).

• In particular, the lambda terms up to $\alpha$-equivalence form an initial algebra for a functor.
Approach

- Functors with finitary presentation.
  - Introduced by Bonsangue and Kurz in coalgebraic logic.
  - Give rise to adequate logics for coalgebras:

\[
L \xrightarrow{\mathcal{A}} \mathcal{X} \xleftarrow{T}
\]

- Moving to many-sorted varieties is necessary in certain situations
- Another application: modularity! How can we describe logics for \( T_1 \circ T_2 \)-coalgebras?
Approach

- Functors with finitary presentation.
  - Introduced by Kurz and Bonsangue in coalgebraic logic.
  - Give rise to adequate logics for coalgebras:
    
    $L \leftarrow A \xrightarrow{T} \mathcal{X} \rightarrow T$

- Moving to many-sorted varieties is necessary in certain situations
- Another application: modularity! How can we describe logics for $T_1 \circ T_2$-coalgebras?
Overview

- Motivation
- Functors with finitary presentations
- Equational logic for higher-order abstract syntax
- Connection with nominal sets
Finitary presentations for functors

\[
\begin{align*}
\mathcal{A} & \xrightarrow{L} \mathcal{A} \\
\downarrow F & \quad \downarrow U \\
\Sigma \setminus S & \xrightarrow{\Sigma} \text{Set}
\end{align*}
\]

where \( E \subseteq (M\Sigma MV)^2 \) and \( M \) is the monad \( UF \)
Two results

A characterization theorem (Kurz and Rosický)

$L$ has a finitary presentation by operations and equations if and only if $L$ preserves sifted colimits.

Alg$(L)$ as an equational class

Let $A = \text{Alg}(\Sigma_A, E_A)$ be an $S$-sorted variety and let $L : A \to A$ be a functor presented by operations $\Sigma_L$ and equations $E_L$. Then $\text{Alg}(L) \cong \text{Alg}(\Sigma_A + \Sigma_L, E_A + E_L)$. 
Two results

A characterization theorem (Kurz and Rosický)

$L$ has a finitary presentation by operations and equations if and only if $L$ preserves sifted colimits.

Alg$(L)$ as an equational class

Let $A = \text{Alg}(\Sigma_A, E_A)$ be an $S$-sorted variety and let $L : A \to A$ be a functor presented by operations $\Sigma_L$ and equations $E_L$. Then $\text{Alg}(L) \cong \text{Alg}(\Sigma_A + \Sigma_L, E_A + E_L)$. 
Overview

- Motivation

- Functors with finitary presentations

- Equational logic for higher-order abstract syntax

- Connection with nominal sets
The functor-category $\text{Set}^F$

- $F$ is the full subcategory of $\text{Set}$ with objects $\underline{n} = \{1, \ldots, n\}$ and $\underline{0} = \emptyset$.

- $A = \text{Set}^F$ is a many-sorted unary variety:
  - the sorts: objects of $F$
  - operation symbols: morphisms of $F$
  - equations: $h(x) = f(g(x))$ (when equality holds in $F$) or $id_{\underline{n}}(x) = x$
A suitable functor to describe the presheaf of $\lambda$-terms

- A coproduct structure on $\mathbb{F}$

\[
\begin{array}{c}
1 \\
\downarrow^{\text{new}} \\
 n + 1
\end{array}
\]

\[
\begin{array}{c}
\downarrow^{i} \\
 n 
\end{array}
\]

where $i$ is the inclusion and $\text{new}(1) = n + 1$.

- The type constructor $\delta : \mathcal{A} \to \mathcal{A}$ for context extension:

\[
\delta(A)(\rho) = A(\rho + id_1) \quad \forall A \in \mathcal{A} \quad \forall \rho \in \mathcal{F}^{Morph}
\]

- Let $L : \mathcal{A} \to \mathcal{A}$ be the functor given by

\[
LX = \delta X + X \times X
\]
The algebraic structure of $\Lambda V_\alpha$

- For an arbitrary presheaf of variables $V$, the $\alpha$-equivalence classes of $\lambda$-terms over $V$ form a presheaf in $\text{Set}^F$: $\Lambda V_\alpha$.

**Theorem.** (Fiore, Plotkin, Turi) $\Lambda V_\alpha$ is the free $L$-algebra on the presheaf of variables $V$

- But $\text{Alg}(L)$ is an equational class, and a presentation can be obtained from:

  - an equational presentation of $\mathcal{A}$ and
  - a finitary presentation of $L$. 
An equational presentation for $\mathcal{A}$: the signature

We consider the following operation symbols:

\[ \Sigma_{\mathcal{A}} = \{\sigma_n^{(i)} | 1 < n, 1 \leq i < n\} \cup \{w_n | n \geq 0\} \cup \{c_n | n > 0\} \cup \{\sigma_0\} \]

with the intended interpretation:

- $\sigma_n^{(i)}$ - the transposition $(i, i + 1)$ of the set $n$,
- $c_n$ - a contraction $c_n : n + 1 \rightarrow n$, given by
  \[ c_n(i) = i \quad \forall i \leq n, \quad c_n(n + 1) = n \]
- $w_n$ - the inclusion of $n$ into $n + 1$.
- $\sigma_0$ - the empty function.
An equational presentation for \( A \): the equations \( E_A \) (1)

-the equations coming from the presentation of the symmetric group:

\[
\begin{align*}
(\sigma_n^{(i)})^2(x) &= \text{id}_n(x) & 1 \leq i < n \\
\sigma_n^{(i)} \sigma_n^{(j)}(x) &= \sigma_n^{(j)} \sigma_n^{(i)}(x) & j \neq i \pm 1; 1 \leq i, j < n \quad (E_1) \\
(\sigma_n^{(i)} \sigma_n^{(i+1)})^3(x) &= \text{id}_n(x) & 1 \leq i < n - 1
\end{align*}
\]
An equational presentation for $\mathcal{A}$: the equations $E_\mathcal{A}$ (2)

-and some extra equations:

\[
\begin{align*}
c_n\sigma_{n+1}^{(n)}(y) &= c_n(y) & (E_2) \\
c_nw_n(x) &= id_n(x) & (E_3) \\
\sigma_{n+1}^{(i)}w_n(x) &= w_n\sigma_n^{(i)}(x) & 1 \leq i < n & (E_4) \\
\sigma_{n+2}^{(n+1)}w_{n+1}w_n(x) &= w_{n+1}w_n(x) & (E_5) \\
\sigma_n^{(i)}c_n(y) &= c_n\sigma_n^{(i)}(y) & i < n - 1 & (E_6) \\
c_n\sigma_{n+1}^{(n-1)}\sigma_{n+1}^{(n)}w_n(x) &= \sigma_n^{(n-1)}w_{n-1}c_{n-1}(x) & (E_7) \\
c_nc_{n+1}\sigma_{n+2}^{(n)} &= c_nc_{n+1} & (E_8) \\
((2, n - 1)(1, n)w_{n-1}c_{n-1})^2 &= (w_{n-1}c_{n-1}(2, n - 1)(1, n))^2 & (E_9)
\end{align*}
\]

(E_9) comes from the presentation of the monoid of functions on $n$, given by Aizenstat.
A finitary presentation for $L$

- **The operation symbols:** $\text{lam}_n, \text{app}_n$ for each $n \in \mathbb{N}$; (semantically they correspond to $\lambda$-abstraction and application).

- The respective signature functor $\Sigma_L : \text{Set}^{\mathbb{N}} \rightarrow \text{Set}^{\mathbb{N}}$ is given by:

\[
(\Sigma_L X)_m = \{\text{lam}_{m+1}\} \times X_{m+1} + \{\text{app}_m\} \times X_m \times X_m
\]

- For any presheaf $V \in \mathcal{A}$ let $\rho_V : \Sigma UV \rightarrow ULV$ be the map defined by

\[
\text{lam}_{n+1}(t) \mapsto t \quad \forall t \in V(n + 1) = (\delta V)(n)
\]

\[
\text{app}_n(t_1, t_2) \mapsto (t_1, t_2) \quad \forall t_1, t_2 \in V(n)
\]
A finitary presentation for $L$ - the equations

- The equations $E_L$ should correspond to the kernel pair of the adjoint transpose $\rho_V^\# : F\Sigma UFV \rightarrow LV$.

\[
\begin{align*}
\sigma^{(i)}_n \text{lam}_{n+1}(t) &= \text{lam}_{n+1}(\sigma^{(i)}_{n+1} t) \\
wn \text{lam}_{n+1}(t) &= \text{lam}_{n+2}(\sigma^{(n+1)}_{n+2} wn_{n+1} t) \\
cn \text{lam}_{n+2}(t') &= \text{lam}_{n+1}(\sigma^{(n)}_{n+1} cn+1 \sigma^{(n)}_{n+2} \sigma^{(n+1)}_{n+2} t') \\
\sigma^{(i)}_n \text{app}_n(t_1, t_2) &= \text{app}_n(\sigma^{(i)}_n t_1, \sigma^{(i)}_n t_2) \\
wn \text{app}_n(t_1, t_2) &= \text{app}_{n+1}(wn t_1, wn t_2) \\
cn \text{app}_{n+1}(t_1, t_2) &= \text{app}_n(cn t_1, cn t_2)
\end{align*}
\]
Representing different implementations of $\lambda$-terms

- If $V$ is the presheaf defined by $V(\rho) = \rho$ for all morphisms $\rho$ in $\mathcal{F}$, the free $L$-algebra over $V$ gives an implementation of $\lambda$-terms by the De Bruijn levels method.

- How can we obtain different implementations for $\lambda$-terms?

  - One possible approach: equip $\mathcal{F}$ with different coproduct structures!

  - But this implies working with a different functor than $L$.

  - Let's keep $L$ and use different presheaves of variables!
Representing different implementations of λ-terms

• If \( V \) is the presheaf defined by \( V(\rho) = \rho \) for all morphisms \( \rho \) in \( F \), the free \( L \)-algebra over \( V \) gives an implementation of \( \lambda \)-terms by the De Bruijn levels method.

• How can we obtain different implementations for \( \lambda \)-terms?

• One possible approach: equip \( F \) with different coproduct structures!

• But this implies working with a different functor than \( L \).

• Let’s keep \( L \) and use different presheaves of variables!
Overview

▷ Motivation

▷ Functors with finitary presentations

▷ Equational logic for higher-order abstract syntax

► Connection with nominal sets
Connection with nominal sets

- Replace $\mathbb{F}$ with $\mathbb{I}$
- Replace $\mathbb{F}$ with $\mathbb{S}$

  objects: finite subsets of a countable set of atoms $\mathbb{A}$

  morphisms: injective maps

- An equational presentation for $\text{Set}^S$:

  - operation symbols: $(a, b)_S : S \cup \{a\} \to S \cup \{b\}$ and $w_{S,c} : S \to S \cup \{c\}$, for $S \subseteq_{\text{fin}} \mathbb{A}$ and $a, b, c \notin S$

    \[
    \begin{align*}
    (b, a)(a, b) &= \text{id} : S \cup \{a\} \\
    (a, b)(c, d) &= (c, d)(a, b) : S \cup \{b, d\} \\
    (b, c)(a, b) &= (a, c) : S \cup \{c\} \\
    (a, b)w_c &= w_c(a, b) : S \cup \{c, b\} \\
    (a, b)w_a &= w_b : S \cup \{b\} \\
    w_aw_b &= w_bw_a : S \cup \{a, b\}
    \end{align*}
    \]
Syntactical differences

• A new signature: $\text{lam}_{S,a} : S \cup \{a\} \rightarrow S$, $\text{app}_S : S \times S \rightarrow S$ and $v_a : \rightarrow \{a\}$ for all $a \in A$, $S \subset \text{fin} A$.

• A slightly different type constructor $\delta : \text{Set}^S \rightarrow \text{Set}^S$

• Equations:

  $t : S \cup \{a, c\} \vdash (b, c)\text{lam}_a(t) = \text{lam}_a((b, c)t) : S \cup \{b\}$
  $t : S \cup \{a\} \vdash w_b\text{lam}_a(t) = \text{lam}_a(w_b t) : S \cup \{b\}$
  $t_1, t_2 : S \vdash w_b\text{app}(t_1, t_2) = \text{app}(w_b t_1, w_b t_2) : S \cup \{b\}$
  $t_1, t_2 : S \cup \{a\} \vdash (a, b)\text{app}(t_1, t_2) = \text{app}((a, b)t_1, (a, b)t_2) : S \cup \{b\}$
  $\vdash (a, b)v_a = v_b$
\(\alpha\beta\eta\)-equivalence

\[
\begin{align*}
\alpha &: S \vdash \\
\alpha &: S; \quad \beta &: S \vdash & \quad \text{app}(\text{lam}_a(w_{S\nu_a}), \alpha) = \alpha &: S \\
\alpha, \beta &: S \cup \{a\}; \quad \gamma &: S \vdash & \quad \text{app}(\text{lam}_a(\text{app}(\alpha, \beta)), \gamma) = \\
& & \quad \text{app}(\text{app}(\text{lam}_a(\alpha), \gamma), \text{app}(\text{lam}_a(\beta), \gamma)) : S \\
\alpha &: S \cup \{a, b\}; \quad \beta &: S \vdash & \quad \text{app}(\text{lam}_a(\text{lam}_b(\alpha)), \beta) = \\
& & \quad \text{lam}_b(\text{app}(\text{lam}_a(\alpha), w_a\beta)) : S \\
\alpha &: S \cup \{a\} \vdash & \quad \text{app}(w_b\text{lam}_a(\alpha), w_{S\nu_b}) = (a, b)\alpha &: S \cup \{b\} \\
\alpha &: S; \quad \beta &: a \vdash & \quad \text{lam}_a(\text{app}(w_a\alpha, w_{S\nu_a})) = \alpha &: S
\end{align*}
\]
Thank you!