Implementing a Normalizer Using Sized Heterogeneous Types

Normalization of λ-Terms by Structural Recursion

Andreas Abel

Mathematically Structured Functional Programming
Kuressaare, Estonia, July 2, 2006

Outline

Contents

1 Introduction

Interpreters

- Turing-complete Language L: can implement its own interpreter.
- What meta-language ML is required to interpret a total (=terminating) language L?
- ML should also be total.
- Here: L = simply-typed λ-calculus
- ML = $\mathsf{F}_\omega^{\widehat{}}$, a functional programming language with sized types
- Termination can be ensured by the type-checker!

2 The Meta-Language: $\mathsf{F}_{\widehat{\omega}}$

The meta-language: $\mathsf{F}_{\widehat{\omega}}$

- Pure functional language (no assignments, pointers, I/O)
- Higher-order functions
- Impredicative polymorphism
- Sized recursive types of higher-kind
- Recursion, restricted; termination guaranteed by type system
- Corecursion, restricted: productivity guaranteed by type system
- Mathematical structure: ordinals, transfinite induction

Sized types, semantically

- Recursive types defined from below by transfinite iteration.
- Example: $\mathsf{List}\,A = \mu^{\omega} F$
- $F\,X = \{\mathsf{nil}, \mathsf{cons}\,a\,as \mid a \in A, as \in X\}$
- Transfinite Iteration:

$$\begin{array}{lcl} \mu^0 F & = & \emptyset \\ \mu^{\alpha+1} F & = & F\,(\mu^{\alpha} F) \\ \mu^{\lambda} F & = & \bigcup_{\alpha<\lambda} \mu^{\alpha} F \end{array}$$

- F monotone: $\mu^{\alpha} F \subseteq \mu^{\beta} F$ for $\alpha \leq \beta$.
- Sized type: $\mathsf{List}^{\alpha} A = \mu^{\alpha} F$.

Sized types, syntactically

- Data types are equipped with a size index (upper bound)
- E.g., $\mathsf{List}^{\imath}\,A$ denotes lists of length $< \imath$ with elements in A
- Constructors (polymorphic):

$$\begin{array}{lcl} \mathsf{nil} & : & \forall\imath\forall A.\ \mathsf{List}^{\imath+1} A \\ \mathsf{cons} & : & \forall\imath\forall A.\ A \to \mathsf{List}^{\imath} \to \mathsf{List}^{\imath+1}\,A \end{array}$$

- Size expressions must have the form $\imath + n$ ($\imath$ size variable, n natural number) or ∞ (unbounded size).
- Subtyping: $\mathsf{List}^{\imath} A \leq \mathsf{List}^{\imath+1} A \leq \cdots \leq \mathsf{List}^{\infty} A$.

Recursion over sized types, semantically

- Prove that $\mathsf{fix}\, s = s\,(\mathsf{fix}\, s) \in A(\beta)$ by transfinite induction on β.

  1. Base: $\mathsf{fix}\, s \in A(0)$ (bottom-check)
  2. Step: $\mathsf{fix}\, s \in A(\alpha)$ implies $\mathsf{fix}\, s \in A(\alpha + 1)$
  3. Limit: $\mathsf{fix}\, s \in A(\lambda)$ if $\mathsf{fix}\, s \in A(\alpha)$ for all $\alpha < \lambda$.

- Proof skeleton:

  1. Base: holds e.g., for $A(\alpha) = \mathsf{List}^{\alpha} B \to C$.
  2. Step: holds if $s \in A(\alpha) \to A(\alpha + 1)$.
  3. Limit: holds for upper-semicontinuous A [CSL 06].

Recursion over sized types, syntactically

- Recursion restricted to this pattern:

$$\begin{array}{ll} f : \forall\imath. A(\imath) \to C(\imath) & \\ f(x : A(\imath + 1)) = & (\ldots f(t : A(\imath)) \ldots \\ & \ldots g(f : A(\imath) \to C(\imath)) \ldots) : C(\imath + 1) \end{array}$$

- Termination of recursion ensured by types.

- Example:

$$\begin{array}{l} \mathsf{filter} : \forall\imath\forall A.\, (A \to \mathsf{Bool}) \to \mathsf{List}^{\imath} A \to \mathsf{List}^{\imath} A \\ \mathsf{filter}\; p\; \mathsf{nil} = \mathsf{nil} \\ \mathsf{filter}\; p\; (\mathsf{cons}\, a\; as : \mathsf{List}^{\imath+1} A) = \\ \qquad \mathsf{if}\;\; p(a)\;\; \mathsf{then}\; \mathsf{cons}\, a\, (\mathsf{filter}\, p\, (as : \mathsf{List}^{\imath} A)) \\ \qquad \mathsf{else}\;\; \mathsf{filter}\, p\, (as : \mathsf{List}^{\imath} A) \end{array}$$

3 The Object Language: STL

The Simply-Typed λ-Calculus

- An even purer functional language (no data types, no recursion, no polymorphism)

- Programs consist of functions and application.

$$\begin{array}{rcll} r, s, t & ::= & x & \text{variable} \\ & | & \lambda x\!:\!a.t & \text{abstraction of } x \text{ in } t \\ & | & r\, s & \text{application} \end{array}$$

- Types:

$$\begin{array}{rcll} a, b, c & ::= & o & \text{base type} \\ & | & a \to b & \text{function type} \end{array}$$

Computation in STL

- Only reduction rule:

$$(\lambda x\!:\!a.t)\,s \longrightarrow [s/x]t$$

- Example:

$$\begin{array}{rl} & (\lambda x\!:\!((o \to o) \to (o \to o)).\,x\,(\lambda z\!:\!o.z))\;(\lambda y\!:\!(o \to o).\,y) \\ \longrightarrow & [(\lambda y\!:\!(o \to o).\,y)/x](x\,(\lambda z\!:\!o.z)) \\ = & (\lambda y\!:\!(o \to o).\,y)\;(\lambda z\!:\!o.z) \\ \longrightarrow & [(\lambda z\!:\!o.z)/y]y = \lambda z\!:\!o.z \end{array}$$

- Normal form $v ::= \lambda x\!:\!a.v \mid x\,v_1 \ldots v_n$.

A Big-Step Interpreter for STL

- For term t, $[\![t]\!]$ computes its normal form.

$$\begin{array}{lcll} [\![x]\!] & = & x & \\ [\![\lambda x\!:\!a.r]\!] & = & \lambda x\!:\!a.\,[\![r]\!] & \\[1ex] [\![r\,s]\!] & = & [[\![s]\!]^a/x]t & \text{if } [\![r]\!] = \lambda x\!:\!a.t \\ & & [\![r]\!]\,[\![s]\!] & \text{otherwise} \end{array}$$

- Substitution $[[\![s]\!]^a/x]t$ of one normal form s into another normal form t may trigger new reductions.

Hereditary Substitutions

- Normalizing substitution of normal forms: $[s^a/x]t$

$$\begin{array}{rcll} [s^a/x]x & = & s^a & \\ [s^a/x]y & = & y & \text{if } x \neq y \\ [s^a/x](\lambda y\!:\!b.r) & = & \lambda y\!:\!b.\,[s^a/x]r & \text{where } y \text{ fresh for } s, x \\[1ex] [s^a/x](t\,u) & = & ([\hat{u}^b/y]r')^c & \text{if } \hat{t} = (\lambda y\!:\!b'.r')^{b \to c} \\ & & \hat{t}\,\hat{u} & \text{otherwise} \\[1ex] \text{where } \hat{t} & = & [s^a/x]t & \\ \hat{u} & = & [s^a/x]u & \end{array}$$

- Invariant: $|b \to c| \leq |a|$ in line 4.

What is happening in hereditary substitutions?

- In $[s^a/x]t$, size of type $|a|$ is "fuel".
- As long as there is fuel, new her. substitutions can be performed.
- Each new substitution starts with less fuel.
- E.g. $[w^{a\to b\to c}/x](x\ v_1\ v_2)$
- Possibly new subst. of v_1 into w: fuel = $|a|$.
- Substitution of v_2 into the result: fuel = $|b|$.
- $[s^a/x]t$ terminates by lexicographic order $(|a|, |t|)$.

What happens for ill-typed terms?

- $[s^a/x]t$ also terminates for ill-typed or non-normal s, t.
- But fuel might run out before normal form is reached.
- Result might be non-normal form.
- Example:

$$\begin{array}{rl} & [\![(\lambda x\!:\!(o\to o).\,x\,x)\,(\lambda x\!:\!o.\,x\,x)]\!] \\ \longrightarrow & [(\lambda x\!:\!o.\,x\,x)^{o\to o}/x](x\,x) \\ \longrightarrow & (\lambda x\!:\!o.\,x\,x)^{o\to o}\,(\lambda x\!:\!o.\,x\,x)^{o\to o} \\ \longrightarrow & [(\lambda x\!:\!o.\,x\,x)^{o}/x](x\,x) \\ \longrightarrow & (\lambda x\!:\!o.\,x\,x)^{o}\,(\lambda x\!:\!o.\,x\,x)^{o} \end{array}$$

4 De Bruijn Implementation

Representation of STL in $\mathsf{F}_\omega^{\widehat{}}$

- STL-types represented as a sized type.

$$\begin{array}{lcl} \mathsf{o} & : & \mathsf{Ty}^{\imath+1} \\ \mathsf{arr} & : & \mathsf{Ty}^{\imath} \to \mathsf{Ty}^{\imath} \to \mathsf{Ty}^{\imath+1} \end{array}$$

- If $a : \mathsf{Ty}^{\imath}$ then $|a| < \imath$.
- STL-terms represented by nested data type $\mathsf{Tm}^{\imath}\,A$:

$$\begin{array}{lcl} \mathsf{var} & : & A \to \mathsf{Tm}^{\imath+1}\,A \\ \mathsf{app} & : & \mathsf{Tm}^{\imath}\,A \to \mathsf{Tm}^{\imath}\,A \to \mathsf{Tm}^{\imath+1}\,A \\ \mathsf{abs} & : & \mathsf{Ty}^{\infty} \to \mathsf{Tm}^{\imath}\,(1+A) \to \mathsf{Tm}^{\imath+1}\,A \end{array}$$

- $\mathsf{Tm}^{\imath}\,A$ contains terms of height $< \imath$ with free variables in A.

Results of hereditary substitution

- Result of a hereditary substitution can either be a term with remaining fuel or a term with no fuel.

$$\begin{aligned}&\mathsf{Res}^{\imath}\, A = \mathsf{Tm}^{\infty}\, A \times (1 + \mathsf{Ty}^{\imath})\\ &\mathsf{ne}_{\mathsf{Res}} : \mathsf{Tm}^{\infty}\, A \to \mathsf{Res}^{\imath}\, A\\ &\mathsf{nf}_{\mathsf{Res}} : \mathsf{Tm}^{\infty}\, A \to \mathsf{Ty}^{\imath} \to \mathsf{Res}^{\imath}\, A\end{aligned}$$

- Extracting the term from a result:

$$\mathsf{tm} : \mathsf{Res}^{\imath} A \to \mathsf{Tm}^{\infty}\, A$$

Simultaneous hereditary substitutions

- For $\mathsf{Tm}A$, only simultaneous substitution

$$\mathsf{Tm}A \to (A \to \mathsf{Tm}B) \to \mathsf{Tm}B$$

  is directly definable.

- Valuations for all variables:

$$\begin{aligned}&\mathsf{Val}^{\imath} A\, B = A \to \mathsf{Res}^{\imath} B\\ &\mathsf{sg}_{\mathsf{Val}} : \mathsf{Tm}^{\infty} A \to \mathsf{Ty}^{\imath} \to \mathsf{Val}^{\imath}\,(1 + A)\, A\\ &\mathsf{lift}_{\mathsf{Val}} : \mathsf{Val}^{\imath}\, A\, B \to \mathsf{Val}^{\imath}\,(1 + A)\,(1 + B)\end{aligned}$$

The $\mathsf{F}_{\omega}^{\frown}$-Code

$$\begin{aligned}
&\mathsf{subst} : \forall \imath.\, \mathsf{Ty}^{\imath} \to \forall A.\, \mathsf{Tm}^{\infty}\, A \to \mathsf{Tm}^{\infty}(1 + A) \to \mathsf{Tm}^{\infty}\, A\\
&\mathsf{subst}\ a\ s\ t = \mathsf{tm}\,(\mathsf{simsubst}\, t\, (\mathsf{sg}_{\mathsf{Val}}\, s\, a))\\
&\mathsf{where}\ \ \mathsf{simsubst} : \forall \jmath.\, \forall A \forall B.\, \mathsf{Tm}^{\jmath} A \to \mathsf{Val}^{\imath+1}\, A\, B \to \mathsf{Res}^{\imath+1} B\\
&\qquad \mathsf{simsubst}\ t\ \rho = \mathsf{match}\ t\ \mathsf{with}\\
&\qquad\quad \mathsf{var}\, x \quad \mapsto \quad \rho\, x\\
&\qquad\quad \mathsf{abs}\, b\, r \quad \mapsto \quad \mathsf{abs}_{\mathsf{Res}}\, b\, (\mathit{simsubst}\, r\, (\mathsf{lift}_{\mathsf{Val}}\, \rho))\\
&\qquad\quad \mathsf{app}\, t\, u \quad \mapsto \quad \mathsf{let}\ \ \hat{t} = \mathit{simsubst}\ t\ \rho\\
&\qquad\qquad\qquad\qquad\quad\ \ \hat{u} = \mathit{simsubst}\ u\ \rho\\
&\qquad\qquad\qquad\quad \mathsf{in\ match}\ \hat{t}\ \mathsf{with}\\
&\qquad\qquad\qquad\qquad \mathsf{nf}_{\mathsf{Res}}\,(\mathsf{abs}\, b'\, r')\,(\mathsf{arr}\, b\, c)\\
&\qquad\qquad\qquad\qquad\quad \mapsto \mathsf{nf}_{\mathsf{Res}}\,(\mathit{subst}\, b\, (\mathsf{tm}\, \hat{u})\, r')\, c\\
&\qquad\qquad\qquad\qquad _ \mapsto \mathsf{app}_{\mathsf{Res}}\, \hat{t}\, \hat{u}
\end{aligned}$$

5 Conclusion

Conclusion

- A natural implementation of a normalizer
- Structurally recursive
- Termination statically ensured by the type system
- Host language: $\mathsf{F}_\omega^{\widehat{\ }}$ (but ML-Polymorphism sufficient)
- Slogan:

  > In each recursive program there is a structurally recursive one struggling to get out.—Conor McBride