Type Structures and Normalization by Evaluation for System F^ω

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

Computer Science Logic (CSL'09)
Coimbra, Portugal
8 September 2009

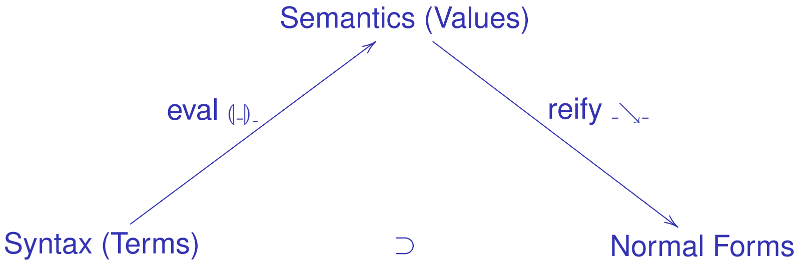
Introduction

- Normalizers appear in compilers (e.g., type-directed partial evaluation [Danvy,Filinski])
- and HOL theorem provers (Isabelle, Coq, Agda).

Normalization by evaluation is a framework to turn an evaluator for closed expressions (stop at lambda) into a normalizer for open expressions (go under lambda).

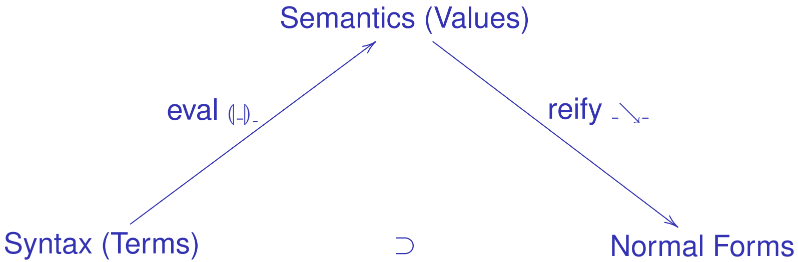
- Has clear semantic foundations.
- Is strong for extensional normalization (eta).
- My goal: NbE for Calculus of Constructions and Coq.

What is Normalization By Evaluation?



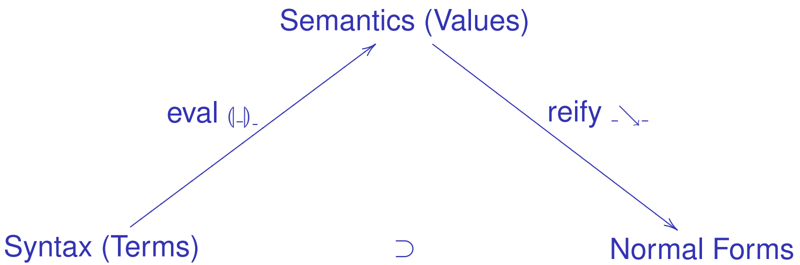
- You have: an interpreter ($(\!|\_|\!)_{\_}$).
- You buy: my reifyer ($\_ \searrow \_$).
- You get for free: a *full normalizer*!

What is Normalization By Evaluation?



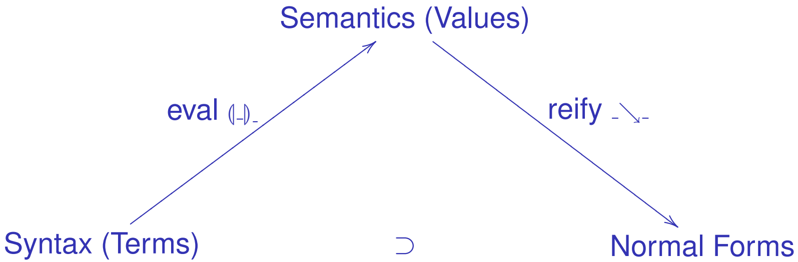
- You have: an interpreter (⦇_⦈_).
- You buy: my reifyer (_ ↘ _).
- You get for free: a *full normalizer*!

What is Normalization By Evaluation?



- You have: an interpreter $((\!|\_|\!)_{\_})$.
- You buy: my reifyer $(\_ \searrow \_)$.
- You get for free: a *full normalizer*!

What is Normalization By Evaluation?



- You have: an interpreter ($(\!|\_|\!)_{\_}$).
- You buy: my reifyer ($\_ \searrow \_$).
- You get for free: a *full normalizer*!

How to Reify a Function

- Functions are thought of as *black boxes*.
- How to print the code of a function?
- Apply it to a fresh variable!

$$\begin{array}{lcl} \downarrow(f) & = & \lambda x.\downarrow(f(x)) \\ \downarrow(x\,\vec{d}) & = & x\downarrow(\vec{d}) \end{array}$$

- Computation needs to be extended to handle variables (unknowns).

Choices of Semantics

1. β-normal forms (Agda 2, Ulf Norell)
2. Weak head normal forms (Constructive Engine, Randy Pollack)
3. Explicit substitutions (Twelf, Pfenning et.al.)
4. Closures (your favorite pure functional language, Epigram 2)
5. Virtual machine code (Coq: ZINC machine, Leroy/Gregoire)
6. Native machine code (Cayenne: i386, Dirk Kleeblatt)

These are all (partial) *applicative structures*.

Applicative Structures

An applicative structure consists of:

- A set D.
- Application operation $\_ \cdot \_ : \mathsf{D} \times \mathsf{D} \to \mathsf{D}$.
- Interpretation $(\!|t|\!)_\eta \in \mathsf{D}$ for term t and environment η, satisfying:

$$\begin{array}{rcl} (\!|x|\!)_\eta & = & \eta(x) \\ (\!|r\ s|\!)_\eta & = & (\!|r|\!)_\eta \cdot (\!|s|\!)_\eta \\ (\!|\lambda x t|\!)_\eta \cdot d & = & (\!|t|\!)_{\eta[x \mapsto d]} \end{array}$$

Simple examples:

1. $\mathsf{D} = (\mathsf{Tm}/{=_\beta})$ terms modulo β-equality.
2. $\mathsf{D} \cong [\mathsf{D} \to \mathsf{D}]$ reflexive (Scott) domain.

Applicative Structures with Variables

- For reification, enrich D with all neutral objects $x\, d_1 \ldots d_n$, where x a variable and $d_1, \ldots, d_n \in \mathsf{D}$.
- Application satisfies:

$$(x\, \vec{d}) \cdot d = x\, \vec{d}\, d$$

- Examples:
  1. $\mathsf{D} = (\mathsf{Tm}/{=_\beta})$ terms modulo β-equality.
  2. $\mathsf{D} \cong \mathsf{Var} \times \mathsf{D}^* + [\mathsf{D} \to \mathsf{D}]$ Scott domain with neutrals.

β-NbE for Untyped Lambda-Calculus

Let $I = \lambda y.\, y$ identity.

$$
\begin{aligned}
& \downarrow [\![\lambda x.\, I\, x\, I]\!] \\
= \;& \lambda x_1.\, \downarrow([\![\lambda x.\, I\, x\, I]\!] \cdot x_1) \\
= \;& \lambda x_1.\, \downarrow([\![I\, x\, I]\!]_{x \mapsto x_1}) \\
= \;& \lambda x_1.\, \downarrow([\![I]\!] \cdot [\![x]\!]_{x \mapsto x_1} \cdot [\![I]\!]) \\
= \;& \lambda x_1.\, \downarrow([\![y]\!]_{y \mapsto [\![x]\!]_{x \mapsto x_1}} \cdot [\![I]\!]) \\
= \;& \lambda x_1.\, \downarrow([\![x]\!]_{x \mapsto x_1} \cdot [\![I]\!])
\end{aligned}
$$

$$
\begin{aligned}
= \;& \lambda x_1.\, \downarrow(x_1 \cdot [\![I]\!]) \\
= \;& \lambda x_1.\, x_1\, (\downarrow [\![I]\!]) \\
= \;& \lambda x_1.\, x_1\, (\lambda x_2.\, \downarrow([\![I]\!] \cdot x_2)) \\
= \;& \lambda x_1.\, x_1\, (\lambda x_2.\, \downarrow [\![y]\!]_{y \mapsto x_2}) \\
= \;& \lambda x_1.\, x_1\, (\lambda x_2.\, \downarrow x_2) \\
= \;& \lambda x_1.\, x_1\, (\lambda x_2.\, x_2)
\end{aligned}
$$

System F^ω

- Girard's System F^ω is a term calculus for HOL.
  - Impredicative.
  - Computation on the type-level.
- *Kinds* (arities of type constructors).

$$\kappa ::= * \mid \kappa \to \kappa'$$

- *Types* and *type constructors* (simply-kinded lambda-calculus).

$$T, U, V ::= X \mid \lambda X{:}\kappa.\, T \mid T\, U \mid \to \mid \forall^\kappa$$

- *Objects* (polymorphic lambda-calculus).

$$t, u, v ::= x \mid \lambda x{:}T.\, t \mid t\, u \mid \Lambda X{:}\kappa.\, t \mid t\, U$$

Kinding, Typing, and Equality

- Type level.
  1. Kinding context $\Xi ::= X_1 : \kappa_1, \ldots, X_n : \kappa_n$.
  2. Kinding $\Xi \vdash T : \kappa$.
  3. Equality $\Xi \vdash T = T' : \kappa$.
  4. Let $\mathsf{Ty}^{\kappa}_{\Xi} = \{T \mid \Xi \vdash T : \kappa\}$.
- Object level.
  1. Typing context $\Gamma ::= x_1 : T_1, \ldots, x_n : T_n$.
  2. Typing $\Xi; \Gamma \vdash t : T$.
  3. Equality $\Xi; \Gamma \vdash t = t' : T$.

NbE for System F^ω

1. Type normalization.
   1. Organization of types into kinded *type structure*.
   2. Kind-directed reification.
   3. Soundness of NbE by glueing type structure.
2. Object normalization.
   1. Organization of objects into typed *object structure*.
   2. Type-directed reification.
   3. Soundness of NbE by glueing object structure.

Type Structures

- Kripke family $\mathcal{T}^{\kappa}_{\Xi}$ (monotonic in Ξ).
- Constants $\rightarrow \in \mathcal{T}^{* \to * \to *}_{\Xi}$, $\forall^{\kappa} \in \mathcal{T}^{(\kappa \to *) \to *}_{\Xi}$.
- Application $F \cdot G \in \mathcal{T}^{\kappa'}_{\Xi}$ for $F \in \mathcal{T}^{\kappa \to \kappa'}_{\Xi}$ and $G \in \mathcal{T}^{\kappa}_{\Xi}$.
- Evaluation $[\![T]\!]_{\rho}$ for $T \in \mathsf{Ty}^{\kappa}_{\Xi}$.
- Evaluation laws as for applicative structure.
- Examples for type structure:
  1. Syntax: $\mathcal{T}^{\kappa}_{\Xi} = (\mathsf{Ty}^{\kappa}_{\Xi}$ modulo equality).
  2. Values: $\mathcal{T}^{\kappa}_{\Xi} = \mathsf{D}$.
- Type structure *is term-like* if it has the variables $X \in \mathcal{T}^{\Xi(X)}_{\Xi}$ and neutrals.
- The category of type structures has products.

Fundamental Theorem in New Clothes

Theorem (Old)

$[\![\kappa \to \kappa']\!] = \{F \mid F \cdot G \in [\![\kappa']\!]$ for all $G \in [\![\kappa]\!]\}$.
Let $\Xi \vdash T : \kappa$. If $\rho(X) \in [\![\Xi(X)]\!]$ for all X, then $[\![T]\!]_\rho \in [\![\kappa]\!]$.

Theorem (New)

Let $F \in \mathcal{S}^{\kappa\to\kappa'}_{\Xi}$ iff $F \cdot G \in \mathcal{S}^{\kappa'}_{\Xi'}$ for all $G \in \mathcal{S}^{\kappa}_{\Xi'}$, Ξ' extends Ξ. (We write $\mathcal{S}^{\kappa\to\kappa'} = \mathcal{S}^{\kappa} \to \mathcal{S}^{\kappa'}$.)
Then $\mathcal{S}$ is a type substructure of $\mathcal{T}$.

Reification (Simply-Kinded)

- Consider term-like type structure $\mathcal{T}$ of values.
- Inductively defined relation $\Xi \vdash F \searrow V \Uparrow \kappa$.
- "value $F \in \mathcal{T}^{\kappa}_{\Xi}$ reifies to type constructor $V \in \mathsf{Ty}^{\kappa}_{\Xi}$ at kind κ."

$$\frac{\Xi, X\!:\!\kappa \vdash F \cdot X \searrow V \Uparrow \kappa'}{\Xi \vdash F \searrow \lambda X\!:\!\kappa.\, V \Uparrow \kappa \to \kappa'}$$

$$\frac{\Xi \vdash G_i \searrow V_i \Uparrow \kappa_i \text{ for all } i}{\Xi \vdash X\,\vec{G} \searrow X\,\vec{V} \Uparrow *}\,\Xi(X) = \vec{\kappa} \to *$$

- Inputs: Ξ, F, κ
- Output: V (β-normal η-long).

Reification (Step by Step)

- Reifying neutral values step by step:

$$\Xi \vdash H \searrow U \Downarrow \kappa \qquad H \text{ reifies to } U\text{, inferring kind } \kappa.$$

- Inputs: Ξ, H (neutral value).
- Outputs: U (neutral β-normal η-long), κ.
- Rules:

$$\frac{}{\Xi \vdash X \searrow X \Downarrow \Xi(X)} \qquad \frac{\Xi \vdash H \searrow U \Downarrow \kappa \to \kappa' \qquad \Xi \vdash G \searrow V \Uparrow \kappa}{\Xi \vdash H\,G \searrow U\,V \Downarrow \kappa'}$$

$$\frac{\Xi \vdash H \searrow U \Downarrow *}{\Xi \vdash H \searrow U \Uparrow *}$$

Normalization by Evaluation

- Compose evaluation with reification: Let $\Xi \vdash T : \kappa$.

$$\mathsf{Nbe}^{\kappa}(T) = \text{the } V \text{ with } \Xi \vdash [\![T]\!] \searrow V \Uparrow \kappa$$

- Soundness:

  If $\Xi \vdash T : \kappa$ *then* $\Xi \vdash T = \mathsf{Nbe}^{\kappa}(T) : \kappa$

- Completeness:

  If $\Xi \vdash T = T' : \kappa$ *then* $\mathsf{Nbe}^{\kappa}(T) \equiv \mathsf{Nbe}^{\kappa}(T')$.

Glueing Type Structure

- Glueing candidate $\underline{\mathsf{Gl}}, \overline{\mathsf{Gl}} \subseteq \mathcal{T} \times \mathsf{Ty}$

$$\overline{\mathsf{Gl}}^{\kappa}_{\Xi} = \{(F, T) \in \mathcal{T}^{\kappa}_{\Xi} \times \mathsf{Ty}^{\kappa}_{\Xi} \mid \Xi \vdash F \searrow V \Uparrow \kappa \text{ and } \Xi \vdash T = V : \kappa\}$$
$$\underline{\mathsf{Gl}}^{\kappa}_{\Xi} = \{(H, T) \in \mathcal{T}^{\kappa}_{\Xi} \times \mathsf{Ty}^{\kappa}_{\Xi} \mid \Xi \vdash H \searrow U \Downarrow \kappa \text{ and } \Xi \vdash T = U : \kappa\}$$

- Laws:

$$\begin{array}{lcl} \underline{\mathsf{Gl}}^{*} & \subseteq & \overline{\mathsf{Gl}}^{*} \\ \underline{\mathsf{Gl}}^{\kappa} \to \overline{\mathsf{Gl}}^{\kappa'} & \subseteq & \overline{\mathsf{Gl}}^{\kappa \to \kappa'} \\ \underline{\mathsf{Gl}}^{\kappa \to \kappa'} & \subseteq & \overline{\mathsf{Gl}}^{\kappa} \to \underline{\mathsf{Gl}}^{\kappa'} \end{array}$$

- Glueing type structure $\mathsf{G}^{*} = \underline{\mathsf{Gl}}^{*}$ and $\mathsf{G}^{\kappa \to \kappa'} = \mathsf{G}^{\kappa} \to \mathsf{G}^{\kappa'}$.
- Lemma: $\underline{\mathsf{Gl}}^{\kappa} \subseteq \mathsf{G}^{\kappa} \subseteq \overline{\mathsf{Gl}}^{\kappa}$.

Soundness of NbE for Types

Theorem (Soundness of NbE)

If $\Xi \vdash T : \kappa$ *then* $\Xi \vdash T = \mathsf{Nbe}^{\kappa}(T) : \kappa$.

Proof.

- Using the fundamental theorem for G.
- $([\![T]\!], T) \in \mathsf{G}^{\kappa}_{\Xi}$ for $\Xi \vdash T : \kappa$.
- $([\![T]\!], T) \in \overline{\mathsf{GI}}^{\kappa}_{\Xi}$.
- $\Xi \vdash [\![T]\!] \searrow V \Uparrow \kappa$ and $\Xi \vdash T = V : \kappa$ for some Inf V.
- $\Xi \vdash T = \mathsf{Nbe}^{\kappa}(T) : \kappa$.

□

Completeness of NbE for Types

Theorem (Completeness of NbE)

If $\Xi \vdash T = T' : \kappa$ *then* $\mathsf{Nbe}^{\kappa}(T) = \mathsf{Nbe}^{\kappa}(T')$.

- Consider type structure $\mathcal{T} \times \mathcal{T}$ of pairs (F, F').
- Groupoidal structure:
  1. *Transitivity* operation $(F_1, F_2) * (F_2, F_3) = (F_1, F_3)$.
  2. *Symmetry* operation $(F, F')^{-1} = (F', F)$.
- Model equality $\Xi \vdash T = T' : \kappa$ in type groupoid.

Object Level

- Fix some type structure $\mathcal{T}$.
- Define object structure $D_{\Delta}^{\Xi \vdash A}$ for $A \in \mathcal{T}_{\Xi}^{*}$.
- Fundamental theorem.
- Type-directed reification.
- Glueing object structure.
- Soundness of NbE ...

Conclusions

- Related work: Altenkirch, Hofmann, and Streicher (1997) describe NbE for System F using category theory.
- This work: Abstract NbE for System F^{ω} using type structures.
- “Algebraic” reorganization of a normalization proof.
- Future work: scale to the Calculus of Constructions.