Normalization by Evaluation (NbE) Techniques and Abstract Values

Andreas Abel[1], Thierry Coquand[2], Ulf Norell[2]

[1]Ludwig-Maximilians-University Munich
[2]Chalmers University of Technology

Curry-Howard Implementation Techniques 2006
December 19, 2006

Normalization by Evaluation

- Evaluate term t, obtain value $v = [\![t]\!]_\rho \in \mathsf{Val}$.
- Val contains the variables and neutral terms.
- Reify v to a normal form.
- Semantical type-checking is NbE, without reification.

The Language

- Set Tm of raw terms:

$$r, s, t, A, B, C ::= x \mid \lambda x t \mid r\, s \mid \Pi x : A.B \mid Type$$

- Judgements

$\Gamma \vdash$	Γ is a well-formed context
$\Gamma \vdash A$	A is a well-formed type
$\Gamma \vdash t : A$	t has type A

- Wellformed types

$$\frac{\Gamma \vdash}{\Gamma \vdash Type} \qquad \frac{\Gamma \vdash A : Type}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash B}{\Gamma \vdash \Pi x : A.\, B}$$

Typing Rules

- Lambda-calculus

$$\frac{\Gamma \vdash}{\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x t : \Pi x : A.\, B}$$

$$\frac{\Gamma \vdash r : \Pi x : A.\, B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B[s/x]}$$

- Π in Type, conversion:

$$\frac{\Gamma \vdash A : \mathit{Type} \qquad \Gamma, x : A \vdash B : \mathit{Type}}{\Gamma \vdash \Pi x : A.\, B : \mathit{Type}} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A'}{\Gamma \vdash t : A'}\ A =_{\beta\eta} A'$$

Evaluation, Abstractly

- λ-model: D with application $\cdot$ and evaluation $[\![t]\!]_\rho$
- Computation law for application

$$[\![\lambda x t]\!]_\rho \cdot v = [\![t]\!]_{\rho[x \mapsto v]}$$

- Computation laws for evaluation

$$\begin{aligned} [\![x]\!]_\rho &= \rho(x) \\ [\![r\ s]\!]_\rho &= [\![r]\!]_\rho \cdot [\![s]\!]_\rho \end{aligned}$$

- Substitution law: $[\![t[s/x]]\!]_\rho = [\![t]\!]_{\rho[x \mapsto [\![s]\!]_\rho]}$
- (Ir)relevance law: $[\![t]\!]_\rho = [\![t]\!]_{\rho'}$ if $\rho(x) = \rho'(x)$ for all $x \in \mathsf{FV}(t)$.

Values as Scott Domain

- Variables and neutral objects

$$\frac{v_1, \ldots, v_k \in \mathsf{Val}}{x\, v_1 \ldots v_k \in \mathsf{Val}}$$

- Canonical objects, using functions of the Metalanguage

$$\frac{f \in \mathsf{Val} \to \mathsf{Val}}{\mathsf{Lam}\, f \in \mathsf{Val}} \qquad \frac{v \in \mathsf{Val} \qquad f \in \mathsf{Val} \to \mathsf{Val}}{\mathsf{Pi}\, v\, f \in \mathsf{Val}} \qquad \frac{}{\mathsf{Type} \in \mathsf{Val}}$$

Evaluation, Concretely

- Application

$$\begin{aligned} (\mathsf{Lam}\, f) \cdot v &= f(v) \\ x\,\vec{v} \cdot v &= x\,(\vec{v}, v) \end{aligned}$$

- Evaluation

$$\begin{aligned} [\![x]\!]_\rho &= \rho(x) \\ [\![r\,s]\!]_\rho &= [\![r]\!]_\rho \cdot [\![s]\!]_\rho \\ [\![\lambda x t]\!]_\rho &= \mathsf{Lam}\, f && \text{where } f(d) = [\![t]\!]_{\rho[x \mapsto d]} \\ [\![\Pi x : A.\, B]\!]_\rho &= \mathsf{Pi}\, [\![A]\!]_\rho\, g && \text{where } g(d) = [\![B]\!]_{\rho[x \mapsto d]} \\ [\![\mathit{Type}]\!]_\rho &= \mathsf{Type} \end{aligned}$$

- $[\![t]\!] := [\![t]\!]_{\rho_0}$ with $\rho_0(x) = x$.

Semantical Type-Checking

- Bidirectional.
- Use evaluation instead of rewriting.
- Subject (context Δ, type V) is evaluated.
- Object (term) is raw.
- Each object is checked before evaluation.
- Only evaluation creates values.
- All other operations just take values apart.

Type Checking Rules

- Inference: $\Delta \vdash t \downarrow V$ (V is output).

$$\frac{}{\Delta \vdash x \downarrow \Delta(x)} \qquad \frac{\Delta \vdash r \downarrow \mathsf{Pi}\, V\, F \qquad \Delta \vdash s \uparrow V}{\Delta \vdash r\, s \downarrow F \cdot [\![s]\!]}$$

- Checking: $\Delta \vdash t \uparrow V$ (only inputs).

$$\frac{\Delta \vdash t \downarrow V \qquad \Delta \vdash V = V' \uparrow \mathsf{Type}}{\Delta \vdash t \uparrow V'} \qquad \frac{\Delta, x : V \vdash t \uparrow F \cdot x}{\Delta \vdash \lambda x t \uparrow \mathsf{Pi}\, V\, F}$$

$$\frac{\Delta \vdash A \uparrow \mathsf{Type} \qquad \Delta, x : [\![A]\!] \vdash B \uparrow \mathsf{Type}}{\Delta \vdash \Pi x : A.\, B \uparrow \mathsf{Type}} \qquad \frac{}{\Delta \vdash \mathit{Type} \uparrow \mathsf{Type}}$$

Bidirectional η-Equality on Semantics

- Structural: $\Delta \vdash v = v' \downarrow V$ (V is output).

$$\frac{}{\Delta \vdash x = x \downarrow \Delta(x)} \qquad \frac{\Delta \vdash v = v' \downarrow \mathsf{Pi}\, V\, F \qquad \Delta \vdash w = w' \uparrow V}{\Delta \vdash v\, w = v'\, w' \downarrow F \cdot v}$$

- Type-directed: $\Delta \vdash v = v' \uparrow V$ (only inputs).

$$\frac{\Delta \vdash v = v' \downarrow V \qquad \Delta \vdash V = V' \uparrow \mathsf{Type}}{\Delta \vdash v = v' \uparrow V'}$$

$$\frac{\Delta, x{:}V \vdash v \cdot x = v' \cdot x \uparrow F \cdot x}{\Delta \vdash v = v' \uparrow \mathsf{Pi}\, V\, F} \qquad \frac{}{\Delta \vdash \mathsf{Type} = \mathsf{Type} \uparrow \mathsf{Type}}$$

$$\frac{\Delta \vdash V = V' \uparrow \mathsf{Type} \qquad \Delta, x{:}V \vdash F \cdot x = F' \cdot x \uparrow \mathsf{Type}}{\Delta \vdash \mathsf{Pi}\, V\, F = \mathsf{Pi}\, V'\, F' \uparrow \mathsf{Type}}$$

On Higher-Order Values

- Values with embedded functions $\mathsf{Val} \to \mathsf{Val}$ give elegant type-checking.
- But scale badly to local definitions, metavariables...
- Survived in Agdalight for just one year.

Abstract Values

- Benjamin Gregoire: compiling terms in Coq.
- Keep values abstract!
- View: $u ::= x\,\vec{v} \mid \mathsf{Lam}\,f \mid \mathsf{Pi}\,V\,F \mid \mathsf{Type}$.
- Saves clear type-checking design.
- Values could be implemented as closures (Coquand 1996):

$$v ::= x\,\vec{v} \mid t\rho \qquad (t ::= \lambda xt \mid \Pi x\!:\!A.\,B \mid \mathit{Type})$$

$$\begin{array}{lcl} \mathsf{view}\ (\lambda xt)\rho & = & \mathsf{Lam}\,(v \mapsto [\![t]\!]_{\rho[x\mapsto v]}) \\ \mathsf{view}\ (\Pi x\!:\!A.\,B)\rho & = & \mathsf{Pi}\ [\![A]\!]_\rho\ (v \mapsto [\![B]\!]_{\rho[x\mapsto v]}) \\ \mathsf{view}\ (\mathit{Type})\rho & = & \mathsf{Type} \end{array}$$

Conclusions

- Values-as-functions allow clear type-checking design.
- But should only be a view on the values.
- Closures are more flexible (initial model?).