Normalization by Evaluation for Martin-Löf Type Theory with Typed Equality Judgements

Andreas Abel\textsuperscript{1}
Thierry Coquand\textsuperscript{2}  Peter Dybjer\textsuperscript{2}

\textsuperscript{1}Ludwig-Maximilians-University Munich
\textsuperscript{2}Chalmers University of Technology

Logic in Computer Science
Wrocław, Poland
10 July 2007
My Talk

- Dependent type theory basis for theorem provers (functional programming languages) Agda, Coq, Epigram, ...
- Intensional theory with predicative universes.
- Judgemental $\beta\eta$-equality.
- Deciding type equality with Normalization-By-Evaluation.
- Semantic proof of decidability of typing.
Dependent Types

- Dependent function space:
  \[ r : \prod_{x : A} B[x] \quad s : A \quad r s : B[s] \]

- Types contain terms, type equality non-trivial.
- Shape of types can depend on terms:
  \[ \text{Vec } A \ n = A \times \cdots \times A \]
  \[ n \text{ factors} \]

- Type conversion rule:
  \[ t : A \quad A \simeq B \quad \frac{}{t : B} \]

- Deciding type checking requires injectivity of \( \prod \)
  \[ \prod_{x : A} B \simeq \prod_{x : A'} B' \text{ implies } A \simeq A' \text{ and } B \simeq B' \]
Untyped $\beta$-Equality

- One solution: $A \simeq B$ iff $A$, $B$ have common $\beta$-reduct.
- Confluence of $\beta$ makes $\simeq$ transitive.
- Injectivity of $\Pi$ trivial.
- But we want also $\eta$! E.g.
  - Theorem prover should not distinguish between $P(\lambda x. f \, x)$ and $P \, f$,
  - or between two inhabitants of a one-element type.
- The stronger the type equality, the more (sound) programs are accepted by the type checker.
Untyped $\beta\eta$-Equality

- Try: $A \simeq B$ iff $A$, $B$ have common $\beta\eta$-reduct.

- $\beta\eta$-reduction (with surjective pairing) only confluent on strongly normalizing terms

- Proof of s.n. requires model construction

- ...which requires invariance of interpretation under reduction

- ...which requires subject reduction

- ...which requires strengthening

- ...hard to prove for pure type systems (van Benthem 1993)

- Even for untyped $\beta$, model construction difficult: Miquel Werner 2002: The not so simple proof-irrelevant model of CC
Typed $\beta\eta$-Equality

- Introduce equality judgement $\vdash A = B$.
- Relies on term equality $\vdash t = t' : C$.
- Simplifies model construction considerably.
- Now injectivity of $\Pi$ is hard.

Goguen 1994: Typed Operational Semantics for UTT.
- “Syntactical” model.
- Shows confluence, subject reduction, normalization in one go.
- Impressive, technically demanding work.

This work: simpler argument, in the same spirit.

Deciding judgemental equality

Normalization function \( \text{nf}^A(t) \).

- **Completeness:**
  \[ \vdash t = t' : A \text{ implies } \text{nf}^A(t) = \text{nf}^A(t') \] (syntactical equal).

- **Soundness:**
  \[ \vdash t : A \text{ implies } \vdash t = \text{nf}^A(t) : A. \]
Syntax of Terms and Types

- Lambda-calculus with constants

\[ r, s, t ::= c \mid x \mid \lambda x.t \mid r \cdot s \]

- Constants:
  - `c` ::= `N` (type of natural numbers)
  - `z` (zero)
  - `s` (successor)
  - `rec` (primitive recursion)
  - `Fun` (function space constructor)
  - `U` (universe of small types)

- \( \Pi x: A.B \) is written \( \text{Fun } A (\lambda x.B) \).
Judgements

- **Essential judgements**

\[ \Gamma \vdash A \]  
\( A \) is a well-formed type in \( \Gamma \)

\[ \Gamma \vdash t : A \]  
\( t \) has type \( A \) in \( \Gamma \)

\[ \Gamma \vdash A = A' \]  
\( A \) and \( A' \) are equal types in \( \Gamma \)

\[ \Gamma \vdash t = t' : A \]  
\( t \) and \( t' \) are equal terms of type \( A \) in \( \Gamma \)

- **Typing of functions:**

\[ \Gamma, x : A \vdash t : B \]  
\[ \Gamma \vdash \lambda x. t : \text{Fun} A (\lambda x. B) \]

\[ \Gamma \vdash r : \text{Fun} A (\lambda x. B) \quad \Gamma \vdash s : A \]

\[ \Gamma \vdash rs : B[s/x] \]
Rules for Judgmental Equality

- Equality axioms:

\[ \frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x. t) s = t[s/x] : B[s/x]} \]  
\[ \frac{\Gamma \vdash t : \text{Fun} A (\lambda x. B)}{\Gamma \vdash (\lambda x. t x) = t : \text{Fun} A (\lambda x. B)} \quad \text{x} \notin \text{FV}(t) \]

- Computation axioms for primitive recursion.
- Congruence rules.
Small and Large Types

- **Small types (sets):**

\[ \Gamma \vdash A : U \quad \Gamma, x : A \vdash B : U \]
\[ \Gamma \vdash \text{Fun} A (\lambda x. B) : U \]

- **U** includes types defined by recursion like \( \text{Vec} A n \).

- **(Large) types:**

\[ \Gamma \vdash A : U \]
\[ \Gamma \vdash A \]
\[ \Gamma \vdash \text{Fun} A (\lambda x. B) \]
Consider a (total) combinatorial algebra $D$ with constructors $N, z, s, \text{Fun}, U$.

Evaluation $\llbracket t \rrbracket_{\rho}$: Standard.

\[
\begin{align*}
\llbracket c \rrbracket_{\rho} & = c \quad (c \text{ constant}) \\
\llbracket x \rrbracket_{\rho} & = \rho(x) \\
\llbracket r \; s \rrbracket_{\rho} & = \llbracket r \rrbracket_{\rho} \llbracket s \rrbracket_{\rho} \\
\llbracket \lambda x. t \rrbracket_{\rho} \; d & = \llbracket t \rrbracket_{\rho} \{x \mapsto d\}
\end{align*}
\]

Example: $\llbracket \text{Fun} \; A \,(\lambda x. B) \rrbracket = \text{Fun} \; X \; F$ where $X = \llbracket A \rrbracket$ and $F \; d = \llbracket B \rrbracket_{\{x \mapsto d\}}$.

We enrich $D$ with term variables:

$\text{Up} \; u \in D$ for each neutral term $u ::= x \; \vec{v}$ (generalized variable).
Reification (Printing)

- Reification $\downarrow^X d$ produces a $\eta$-long $\beta$-normal term.
  
  \[
  \begin{align*}
  \downarrow^N z &= z \\
  \downarrow^N (s \ d) &= s (\downarrow^N d) \\
  \downarrow^N (Up \ u) &= u \\
  \downarrow Up\ u' (Up \ u) &= u \\
  \downarrow Fun\ X\ F \ f &= \lambda x. \downarrow F (\uparrow^X x) (f (\uparrow^X x)), \ x \text{ fresh}
  \end{align*}
  \]

- Reflection $\uparrow^X u$ embeds a neutral term $u$ into $D$, $\eta$-expanded.
  
  \[
  \begin{align*}
  (\uparrow^{Fun\ X\ F} u) \ d &= \uparrow F \ d (u \downarrow^X d) \\
  \uparrow^X u &= Up \ u
  \end{align*}
  \]

- Normalization of closed terms $\vdash t : A$
  
  \[
  \text{nf}^A(t) = \downarrow^{[A]}[t].
  \]
PER Model

- A PER is a symmetric and transitive relation on $D$.
- Small types: define a PER $U$ and a PER $[X]$ for $X \in U$.

\[
\begin{align*}
N = N & \in U & d = d' & \in [N] & u \text{ neutral} & \Up u = \Up u & \in [N] \\
z = z & \in [N] & s d = s d' & \in [N] & u, u' \text{ neutral} & \Up u = \Up u & \in [Up u] \\
\end{align*}
\]

\[
\begin{align*}
X = X' & \in U & F d = F' d' & \in U \text{ for all } d = d' \in [X] & \Fun X F = \Fun X' F' & \in U \\
\end{align*}
\]

\[
\begin{align*}
f d = f' d' & \in [F d] \text{ for all } d = d' \in [X] & f = f' & \in [\Fun X F] \\
\end{align*}
\]
Modelling Large Types

- Large types: Define PER $\mathcal{T}ype$ and extend $[\cdot]$ to $\mathcal{T}ype$.

$$ \mathcal{U} \subseteq \mathcal{T}ype $$

$$ X = X' \in \mathcal{T}ype \quad F \ d = F' \ d' \in \mathcal{T}ype \text{ for all } d = d' \in [X] $$

$$ \text{Fun} \ X \ F = \text{Fun} \ X' \ F' \in \mathcal{T}ype $$

$$ \mathcal{U} = \mathcal{U} \in \mathcal{T}ype \quad [\mathcal{U}] = \mathcal{U} $$

- PERs contain only total elements of $D$.
- These can be printed (converted to terms).
Checking Semantic Equality

Lemma

Let $X = X' \in \text{Type}$.

1. $\uparrow^X u = \uparrow^{X'} u \in [X]$.
2. If $d = d' \in [X]$ then $\downarrow^X d =_{\alpha} \downarrow^{X'} d'$.

Proof.

Simultaneously by induction on $X = X' \in \text{Type}$. □
Completeness of NbE

Theorem (Validity of judgements in PER model)

Let $\rho(x) = \rho'(x) \in \llbracket \Gamma(x) \rrbracket_\rho$ for all $x$.

- If $\Gamma \vdash t : A$ then $\llbracket t \rrbracket_\rho = \llbracket t \rrbracket_{\rho'} \in \llbracket [A]_\rho \rrbracket$.
- If $\Gamma \vdash t = t' : A$ then $\llbracket t \rrbracket_\rho = \llbracket t' \rrbracket_{\rho'} \in \llbracket [A]_\rho \rrbracket$.

Corollary (Completeness of nf)

If $\vdash t = t' : A$ then $\text{nf}^A(t) =_\alpha \text{nf}^A(t')$.

Soundness remains: If $\vdash t : A$ then $\vdash t = \text{nf}^A(t) : A$. 
Kripke Logical Relation

Relate well-typed terms modulo equality to inhabitants of PERs.

Lemma (Into and out of the logical relation)

1. If \( \Gamma \vdash r = u : C \) then \( \Gamma \vdash r : C \overset{R}{\uparrow} u \in [X] \).
2. If \( \Gamma \vdash r : C \overset{R}{d} \in [X] \) then \( \Gamma \vdash r = \downarrow^X d : C \).

Definition

\[ \Gamma \vdash r : C \overset{R}{d} \in [X] : \iff \Gamma \vdash r = \downarrow^X d : C \] for \( X \) base type,

\[ \Gamma \vdash r : C \overset{R}{f} \in [\text{Fun } X F] : \iff \Gamma \vdash C = \text{Fun } A (\lambda x.B) \text{ for some } A, B \text{ and for all } \Delta \geq \Gamma \text{ and } \Delta \vdash s : A \overset{R}{d} \in [X], \]
\[ \Delta \vdash rs : B[s/x] \overset{R}{f d} \in [F d]. \]
Soundness of NbE

- Prove the fundamental theorem.
- Corollary: $\vdash t : A$ implies $\vdash t : A \supseteq \llbracket t \rrbracket \in \llbracket [A] \rrbracket$.
- Escaping the log. rel.: $\vdash t = \down [A] \llbracket t \rrbracket : A$.
- Hence, $\text{nf}$ is also sound.
- Decidability of judgemental equality entails injectivity of $\Pi$. 
Conclusion

- Semantic metatheory of Martin-Löf Type Theory.
- Inference rules directly justified by PER model.
- No need to prove strengthening, subject reduction, confluence, normalization.
- Future work:
  - Extend to $\Sigma$-types, singleton-types, proof-irrelevance.
  - Adopt to syntax of categories-with-families (de Bruijn indices and explicit substitutions).
Related Work

- Martin-Löf 1975: NbE for Type Theory (weak conversion)
- Martin-Löf 2004: Talk on NbE (philosophical justification)
- Danvy et al: Type-directed partial evaluation
- Altenkirch Hofmann Streicher 1996: NbE for λ-free System F
- Berger Eberl Schwichtenberg 2003: Term rewriting for NbE
- Aehlig Joachimski 2004: Untyped NbE, operationally
- Filinski Rohde 2004: Untyped NbE, denotationally
- Danielsson 2006: strongly typed NbE for LF
- Altenkirch Chapman 2007: Tait in one big step

Special thanks to Klaus Aehlig.