Eta-equivalence in Core Dependent Haskell

You should machine-check your proofs

Stephanie Weirich
University of Pennsylvania
Core Dependent Haskell

A Specification for Dependent Types in Haskell

Stephanie Weirich, Antoine Voizard, Pedro Henrique Avezedo de Amorim, Richard A. Eisenberg

To appear in ICFP 2017

All Coq proofs available online
http://github.com/sweirich/corespec.git
and will be uploaded to ACM DL
Dependent types in core

- Idea: base Haskell Core language (FC) on a dependently-typed language with $\star : \star$

$$
terms, \text{types} \quad a, b, A, B \quad ::= \quad \star \mid x \mid \lambda x : A.b \mid a \ b \\
\quad \quad \quad \quad \quad \quad \quad \mid \Pi x : A.B
$$

- Full-spectrum dependently typed language with a single sort (Martin Löf, 1971 draft paper)
- "Radical impredicativity"
- Not good for proof checking $\ldots$but neither is Haskell
- Type checking is undecidable
- Type sound (Cardelli 1985)
Why Coq formalization?

• Part of DeepSpec project

• Be convincing about system soundness in presence of nontermination

• Errors in prior versions of the system pointed out in Eisenberg's dissertation
  – Missed a case in [Weirich et al. ICFP14], see Eisenberg 16 for fix.
  – Others have made mistakes too

• Fun!
Dependent types + FC

• Dependent types for functions
  \[ f :: \prod_{n:Nat}. \text{Vec} \text{Int} \ n \rightarrow \text{Int} \]

• Irrelevant dependent functions
  \[ f :: \prod_{a:Type}. \prod_{n:Nat}. a \rightarrow \text{Vec} \ a \ n \]

• Coercion abstraction
  \[ f :: \prod_{a:Type}. \forall c:(a \sim \text{Int}). \prod_{x:a}. \text{Int} \]
  \[ f = \lambda a:Type. \Lambda c:(a \sim \text{Int}). \lambda x:a. x \triangleright c \]

• Recursive definitions (toplevel only)
  \[ \text{Fix} :: \prod_{a:Type}. (a \rightarrow a) \rightarrow a \]
  \[ \text{Fix} \sim \lambda a:Type. \lambda f:a \rightarrow a. f \ (\text{Fix} \ a \ f) \]
### Two related languages

<table>
<thead>
<tr>
<th>D</th>
<th>DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Gamma \vdash a : A )</td>
<td>( \Gamma \vdash a : A )</td>
</tr>
<tr>
<td>( \Gamma \vdash \phi )</td>
<td>( \Gamma \vdash \phi )</td>
</tr>
<tr>
<td>( \Gamma; \Delta \vdash a \equiv b : A )</td>
<td>( \Gamma; \Delta \vdash \gamma : a \sim b )</td>
</tr>
<tr>
<td>( \Gamma; \Delta \vdash \phi_1 \equiv \phi_2 )</td>
<td>( \Gamma; \Delta \vdash \gamma : \phi_1 \sim \phi_2 )</td>
</tr>
<tr>
<td>( \vdash \Gamma )</td>
<td>( \vdash \Gamma )</td>
</tr>
</tbody>
</table>

- Curry style vs. Church style type systems (36/35 rules)
- Definitional equality in D is coercion checking in DC
- DC has decidable type checking, D does not
- Both languages have preservation/progress
- Language are equivalent via erasure/annotation
Comparison

**E-Conv**

\[
\begin{align*}
\Gamma &\vdash a : A \\
\Gamma;\overline{\Gamma} &\vdash A \equiv B : \star \\
\hline
\Gamma &\vdash a : B
\end{align*}
\]

- Implicit type conversion because the types are defined to be equal
- Operational semantics: 6 rules

**An-Conv**

\[
\begin{align*}
\Gamma &\vdash a : A \\
\Gamma;\overline{\Gamma} &\vdash \gamma : A \sim B \\
\hline
\Gamma &\vdash B : \star \\
\hline
\Gamma &\vdash a \triangleright \gamma : B
\end{align*}
\]

- Explicit type coercion because the types are not defined to be equal
- Operational semantics: 10 rules
Irrelevant arguments in \( D \)

- Implicit language contains only relevant subterms
- Use fv check for irrelevant abstractions [Miquel, ICC]

\[
\begin{align*}
terms, types & \quad a, b, A, B \quad ::= \quad \star | \ x | \ F | \lambda^\rho x.a | a \ b^\rho | \Box \\
& \quad \mid \ \Pi^\rho x:A \rightarrow B | \Lambda c.a | a[\bullet] | \forall c:\phi.A
\end{align*}
\]

**E-Pi**

\[
\Gamma, x : A \vdash B : \star \\
\frac{\Gamma \vdash \Pi^\rho x:A \rightarrow B : \star}{\Gamma \vdash \lambda^\rho x.a : \Pi^\rho x:A \rightarrow B}
\]

**E-Abs**

\[
(\rho = +) \lor (x \notin fv a) \\
\frac{\Gamma \vdash \lambda^\rho x.a : \Pi^\rho x:A \rightarrow B}{\Gamma, x : A \vdash a : B}
\]

**E-App**

\[
\Gamma \vdash b : \Pi^\ast x:A \rightarrow B \\
\frac{\Gamma \vdash a : A}{\Gamma \vdash b \ a^\ast : B\{a/x\}}
\]

**E-IApp**

\[
\Gamma \vdash b : \Pi^\ast x:A \rightarrow B \\
\frac{\Gamma \vdash a : A}{\Gamma \vdash b \Box^- : B\{a/x\}}
\]
Formalization in Coq

<table>
<thead>
<tr>
<th></th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ott spec</td>
<td>1423</td>
</tr>
<tr>
<td>LaTeX macros</td>
<td>1851</td>
</tr>
<tr>
<td>Paper sources</td>
<td>2317</td>
</tr>
<tr>
<td>Coq</td>
<td>32828</td>
</tr>
<tr>
<td>Language def</td>
<td>1432</td>
</tr>
<tr>
<td>Syntactic lemmas</td>
<td>11730</td>
</tr>
<tr>
<td>System D</td>
<td>5399</td>
</tr>
<tr>
<td>Consistency</td>
<td>2417</td>
</tr>
<tr>
<td>System DC</td>
<td>8142</td>
</tr>
<tr>
<td>Decidability</td>
<td>3529</td>
</tr>
<tr>
<td>Connection</td>
<td>2215</td>
</tr>
<tr>
<td>Utils</td>
<td>629</td>
</tr>
<tr>
<td>Other</td>
<td>2732</td>
</tr>
</tbody>
</table>

ott: Sewell, Zappa Nardelli, et al
Ingen: Aydemir
Locally Nameless Representation

\[
G, \ x:A \models B : \text{TYPE} \\
G \models A : \text{TYPE} \\
\text{----------------------------- :: \text{Pi}} \\
G \models \text{all } \rho \ x:A \rightarrow B : \text{TYPE}
\]

\[
\frac{\Gamma, x:A \models B:\ast \\
\Gamma \models A:\ast}{\Gamma \models \Pi^\rho x:A \rightarrow B:\ast} \quad \text{E}\_\text{Pi}
\]

\text{E}_{\Pi} : \\
\text{for all } (L: \text{vars}) (G: \text{context}) (\rho: \text{relflag}) (A \ B: \text{tm}), \\
\text{for all } x, x \notin \text{in } L \rightarrow \\
\text{Typing } ((x \sim \text{Tm } A) ++ G) \\
\text{(open_tm_wrt_tm } B (a\_Var\_f x)) a\_Star) \\
\rightarrow \text{Typing } G \ A \ a\_Star \\
\rightarrow \text{Typing } G (a\_Pi \rho \ A \ B) a\_Star
Proof Timeline

- Early July 2016, POPL deadline
- Mid July 2016, last 2.8 meeting
- October 2016, ESOP deadline
- February 2017, ICFP deadline
- April 2017, ICFP notification
- May 2017, public release

Paper writing / proofs
Eta-equivalence finished Mar 25
Decidability proof / paper writing
Clean-up & refactoring
Eta-equivalence

- Two new rules (definitional equality & coercion)

\[ \Gamma \vdash b : \Pi^+ x : A \rightarrow B \]
\[ a = b \ x^+ \]
\[ \Gamma; \Delta \vdash \lambda^+ x. a \equiv b : \Pi^+ x : A \rightarrow B \quad \text{E}_\text{ETA} \]

\[ \Gamma \vdash b : \Pi^+ x : A \rightarrow B \]
\[ a = b \ x^+ \]
\[ \Gamma; \Delta \vdash \text{eta } b : (\lambda^+ x : A. a) \sim b \quad \text{A}_N\text{ETA} \]
Consistency

• Progress lemma requires consistency of definitional equality $\left(\Gamma; \Delta \models a \equiv b : A\right)$
  – Definition: a and b are consistent when if they both have head forms then they have the same head forms.
  – Theorem: If a and b are definitionally equal then they are consistent.

• Consistency proof based on confluence of parallel reduction $\left(\vdash a \Rightarrow b\right)$
  – Definition: a and b are joinable when there is some common term that they both parallel reduce to (in any number of steps).
  – Lemma: If a and b are definitionally equal, then they are joinable.
  – Lemma: If a and b are joinable, then they are consistent.
**Eta-equivalence**

- **New parallel reduction rule**

  $$
  \begin{align*}
  \Gamma \vdash b : \Pi^+ x : A \rightarrow B \\
  a = b \ x^+ \\
  \Gamma; \Delta \vdash \lambda^+ x. a \equiv b : \Pi^+ x : A \rightarrow B \\
  \end{align*}
  $$
  \text{E}_{\text{ETA}}

  $$
  \begin{align*}
  \Gamma \vdash b : \Pi^+ x : A \rightarrow B \\
  a = b \ x^+ \\
  \Gamma; \Delta \vdash \text{eta} \ b : (\lambda^+ x : A. a) \sim b
  \end{align*}
  $$
  \text{A}_{\text{N-ETA}}

**Untyped reduction relation**

$$
\begin{align*}
\vdash b \Rightarrow b' \\
\vdash a = b \ x^+ \\
\vdash \lambda^+ x. a \Rightarrow b'
\end{align*}
\text{PAR-ETA}

Wait, what about x not in \text{fv} b?
Can x appear in b?

\[ \models b \Rightarrow b' \]
\[ a = b \; x^+ \]
\[ \models \lambda^+ x . a \Rightarrow b' \quad \text{PAR ETA} \]

Par_Eta : forall (L:vars) (a b' b:tm),
   Par b b'
   -> (forall x, x \not\in L
      -> open_tm_wrt_tm a (a_Var_f x) =
          a_App b Rel (a_Var_f x))
   -> Par (a_UAbs Rel a) b'
Update confluence proofs

- Confluence for \( \beta\eta \)-reduction proved by Tait-Martin Löf (for untyped lambda calculus, see Barendregt 1984)
- Good news: Coq points out three new required cases
- Not so good news: Need induction on height of term, not structure
  \[
  \frac{\Gamma \vdash b \Rightarrow b'}{
  \Gamma \vdash a = b \ x^+ \quad \frac{\vdash \lambda^+ x. a \Rightarrow b'}{\text{PAR}_\eta \text{ETA}}}
  \]
- Not so bad news:
  - Height function automatically defined by lngen
  - Existing tactics in proof for applying IH
  - Omega tactic easily handles all arithmetic
What could have gone wrong?

• Parallel reduction relation ignores types, reduces all terms
• But, a Church-style language lacks confluence for ill-typed terms!

\[ \lambda x: A. (\lambda y: B. a y) x \quad y \text{ appears free in } a, \text{ but not } x \]

\[ \Rightarrow \lambda x: A. a \{ y := x \} \quad \text{(via beta-reduction)} \]
\[ \Rightarrow \lambda y: B. a \quad \text{(via eta-reduction)} \]

If \( A \neq B \) then these terms are not equal
Par doesn't preserve types

• Have $\Gamma \models \lambda^+ x. b \ x^+ : \Pi^+ x : A. B$ and 
  $(\lambda^+ x. b \ x^+) \Rightarrow b$
  but not always
  $\Gamma \models b : \Pi^+ x : A. B$

• Counterexample:
  Say $y : \Pi^- b: \ast. \Pi^+ a : \ast. T \ a \ b$
  have $\Gamma \models \lambda^+ a : \ast. (y \square^-) a : \Pi^+ a : \ast. T \ a \ a$
  and $\lambda^+ a : \ast. (y \square^-) a \Rightarrow \eta \ (y \square^-)$
  but
  $\not\models (y \square^-) : \Pi^+ a : \ast. T \ a \ a$
Preservation not required

• Parallel reduction is a proof technique only, not part of the language definition

• Equals => joinable => consistent
  – More terms can be joinable than the language defines equal
  – Can join terms via ill-typed reductions, as long as the result is consistent

• Still somewhat worrisome, I'm glad I have a machine-checked proof
No more paper proofs!

• Good for typesetting
• Good for extension
  – eta / roles / levity polymorphism / higher-inductive types
• Good for refactoring
  – Removed unnecessary hypotheses from rules
  – Adopted alternative push rules in DC operational semantics
• Good for details
  – Decidability of type checking
  – Erasure / annotation lemma
  – Regularity lemmas
  – Low-level syntactic properties
    • Weakening, substitution
    • If $\Gamma \vdash a : A$ then $\text{fv} \ a \subseteq \text{dom} \ \Gamma$