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 $\star : \star$

$$\begin{array}{llcl} \textit{terms},\ \textit{types} & a, b, A, B & ::= & \star \mid x \mid \lambda x\!:\!A.b \mid a\ b \\ & & \mid & \Pi x\!:\!A.B \end{array}$$

- Full-spectrum dependently typed language with a single sort (Martin Löf, 1971 draft paper)
- **"Radical impredicativity"**
- Not good for proof checking …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 :: Πn:Nat. Vec Int n → Int`

- Irrelevant dependent functions
  `f :: Π⁻a:Type. Π⁺n:Nat. a → Vec a n`

- Coercion abstraction
  ```
  f :: Π⁻a:Type. ∀c:(a ~ Int). Π⁺x:a. Int
  f = λ⁻a:Type. Λc:(a ~ Int).  λ⁺x:a. x ▹ c
  ```

- Recursive definitions (toplevel only)
  ```
  Fix :: Π⁻a:Type. (a→a) → a
  Fix ~ λ⁻a:Type. λ⁺f:a→a. f (Fix a⁻ f)
  ```

Two related languages

For simplicity

For GHC

D	DC
$\Gamma \vDash a : A$	$\Gamma \vdash a : A$
$\Gamma \vDash \phi \text{ ok}$	$\Gamma \vdash \phi \text{ ok}$
$\Gamma ; \Delta \vDash a \equiv b : A$	$\Gamma ; \Delta \vdash \gamma : a \sim b$
$\Gamma ; \Delta \vDash \phi_1 \equiv \phi_2$	$\Gamma ; \Delta \vdash \gamma : \phi_1 \sim \phi_2$
$\vDash \Gamma$	$\vdash \Gamma$

- 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

D

E-Conv

$$\frac{\Gamma \vDash a : A \qquad \Gamma; \widetilde{\Gamma} \vDash A \equiv B : \star}{\Gamma \vDash a : B}$$

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

DC

An-Conv

$$\frac{\Gamma \vdash a : A \qquad \Gamma; \widetilde{\Gamma} \vdash \gamma : A \sim B \qquad \Gamma \vdash B : \star}{\Gamma \vdash a \triangleright \gamma : B}$$

- Explicit type coercion because the types are not defined to be equal
- Operational semantics: 10 rules

Irrelevant arguments in D

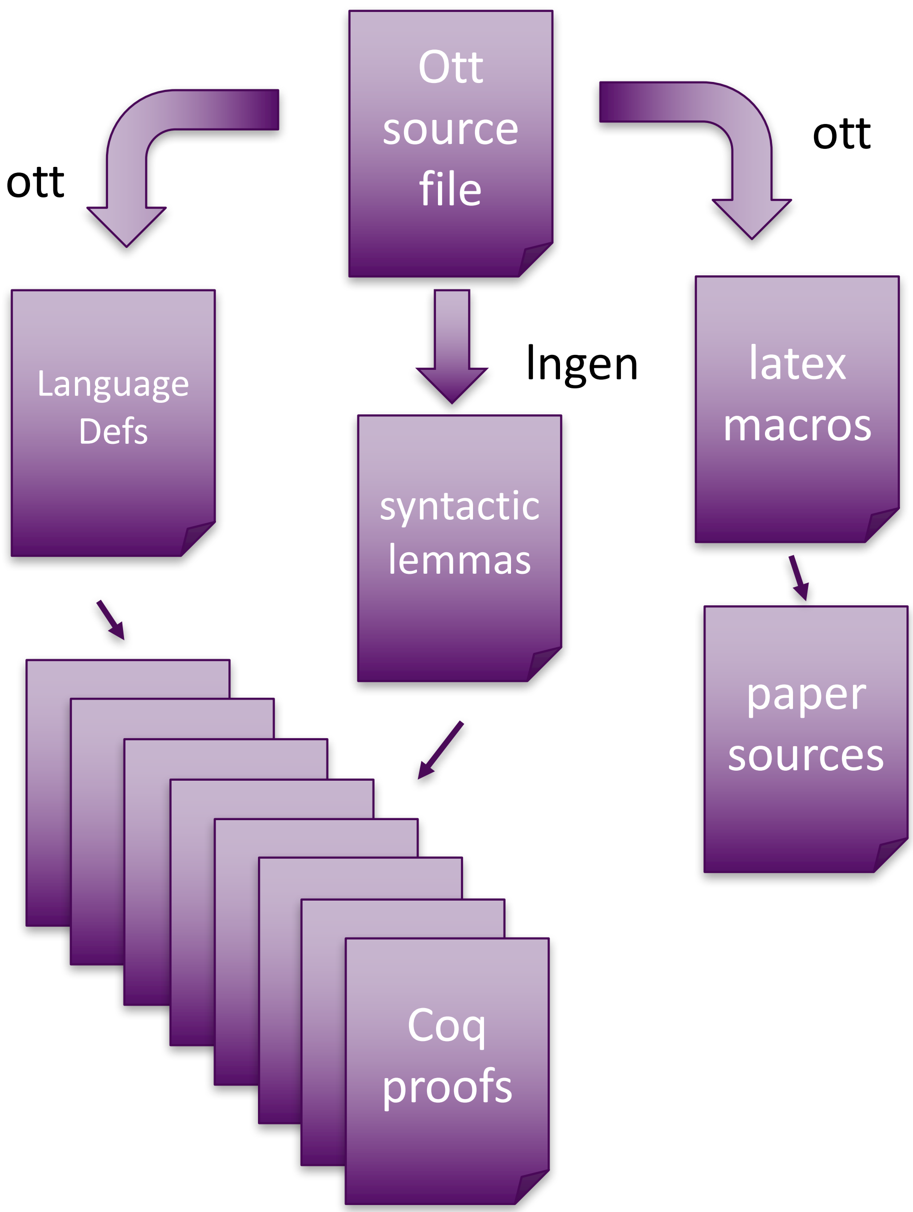
- Implicit language contains only relevant subterms
- Use fv check for irrelevant abstractions [Miquel, ICC]

$$
\begin{array}{llcl}
\textit{terms, types} & a, b, A, B & ::= & \star \mid x \mid F \mid \lambda^\rho x.a \mid a\; b^\rho \mid \Box \\
 & & \mid & \Pi^\rho x{:}A \to B \mid \Lambda c.a \mid a[\bullet] \mid \forall c{:}\phi.A
\end{array}
$$

$$
\textsc{E-Pi}\quad \frac{\Gamma, x : A \vDash B : \star}{\Gamma \vDash \Pi^\rho x{:}A \to B : \star}
\qquad
\textsc{E-Abs}\quad \frac{\Gamma, x : A \vDash a : B \qquad (\rho = +) \vee (x \notin \mathsf{fv}\ a)}{\Gamma \vDash \lambda^\rho x.a : \Pi^\rho x{:}A \to B}
$$

$$
\textsc{E-App}\quad \frac{\Gamma \vDash b : \Pi^+ x{:}A \to B \qquad \Gamma \vDash a : A}{\Gamma \vDash b\; a^+ : B\{a/x\}}
\qquad
\textsc{E-IApp}\quad \frac{\Gamma \vDash b : \Pi^- x{:}A \to B \qquad \Gamma \vDash a : A}{\Gamma \vDash b\; \Box^- : B\{a/x\}}
$$

Formalization in Coq



	LOC
Ott spec	1423
LaTeX macros	1851
Paper sources	2317
Coq	32828
Language def	1432
Syntactic lemmas	11730
System D	5399
Consistency	2417
System DC	8142
Decidability	3529
Connection	2215
Utils	629
Other	2732

ott: Sewell, Zappa Nardelli, et al
Ingen: Aydemir

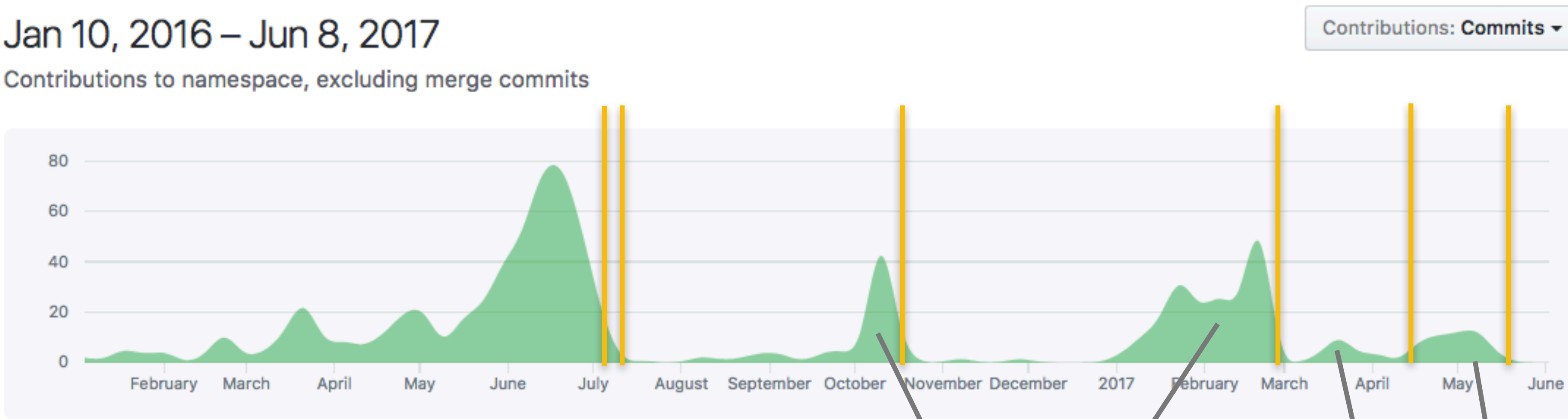
Locally Nameless Representation

```
   G, x:A |= B : TYPE
   G |= A : TYPE
------------------------------- :: Pi
  G |= all rho x:A -> B : TYPE
```

$$\frac{\Gamma, x : A \vDash B : \star \qquad \Gamma \vDash A : \star}{\Gamma \vDash \Pi^{\rho} x{:}A \to B : \star}\ \textsc{E\_Pi}$$

```
E_Pi :
forall (L:vars) (G:context) (rho:relflag)(A B:tm),
 (forall x , x \notin L ->
    Typing ((x ~ Tm A) ++ G)
              (open_tm_wrt_tm B (a_Var_f x)) a_Star)
-> (Typing G A a_Star)
-> 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)

$$\frac{\begin{array}{l}\Gamma \vDash b : \Pi^{+} x : A \to B \\ a = b\ x^{+}\end{array}}{\Gamma ; \Delta \vDash \lambda^{+} x . a \equiv b : \Pi^{+} x : A \to B} \quad \text{E\_ETA}$$

$$\frac{\begin{array}{l}\Gamma \vdash b : \Pi^{+} x : A \to B \\ a = b\ x^{+}\end{array}}{\Gamma ; \Delta \vdash \mathbf{eta}\, b : (\lambda^{+} x : A . a) \sim b} \quad \text{AN\_ETA}$$

Consistency

- Progress lemma requires consistency of definitional equality ($\Gamma; \Delta \vDash a \equiv b : A$)
  - *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 ($\vDash a \Rightarrow b$)
  - *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*

$$\frac{\begin{array}{c}\Gamma \vDash b : \Pi^+ x : A \to B \\ a = b\ x^+\end{array}}{\Gamma ; \Delta \vDash \lambda^+ x . a \equiv b : \Pi^+ x : A \to B} \quad \textsc{E\_Eta}$$

$$\frac{\begin{array}{c}\Gamma \vdash b : \Pi^+ x : A \to B \\ a = b\ x^+\end{array}}{\Gamma ; \Delta \vdash \mathbf{eta}\, b : (\lambda^+ x : A . a) \sim b} \quad \textsc{An\_Eta}$$

Untyped reduction relation

$$\frac{\begin{array}{c}\vDash b \Rightarrow b' \\ a = b\ x^+\end{array}}{\vDash \lambda^+ x . a \Rightarrow b'} \quad \textsc{Par\_Eta}$$

Wait, what about x not in fv b?

Can x appear in b?

$$\frac{\vDash b \Rightarrow b' \qquad a = b\ x^{+}}{\vDash \lambda^{+} x.a \Rightarrow b'} \quad \textsc{Par\_Eta}$$

```
Par_Eta : forall (L:vars) (a b' b:tm),
  Par b b'
  -> (forall x, x \notin  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

$$\frac{\models b \Rightarrow b' \qquad a = b\ x^{+}}{\models \lambda^{+} x.a \Rightarrow b'}\ \textsc{Par\_Eta}$$

- 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!

$\lambda x{:}A.\,(\lambda y{:}B.\,a\ y)\ x$ — y appears free in a, but not x

$\Rightarrow\ \lambda x{:}A.\,a\,\{\,y \coloneqq x\,\}$ (via beta-reduction)

$\Rightarrow\ \lambda y{:}B.\,a$ (via eta-reduction)

If $A \neq B$ then these terms are not equal

Par doesn't preserve types

- Have $\Gamma \vDash \lambda^{+}x.\, b\, x^{+} : \Pi^{+}x{:}A.\, B$ and

  $(\lambda^{+}x.\, b\, x^{+}) \Rightarrow b$

  but not always

  $\Gamma \vDash b : \Pi^{+}x{:}A.\, B$

- Counterexample:

  Say $y : \Pi^{-}b{:}\star.\, \Pi^{+}a{:}\star.\, T\ a\ b$

  have $\vDash \lambda^{+}a{:}\star.\, (y\ \square^{-})a : \Pi^{+}a : \star.\, T\ a\ a$

  and $\lambda^{+}a{:}\star.\, (y\ \square^{-})a \Rightarrow\eta\ (y\ \square^{-})$

  but

  $\nvDash (y\ \square^{-}) : \Pi^{+}a : \star.\, T\ a\ a$

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 $\mathrm{fv}\ a \in \mathrm{dom}\ \Gamma$