Irrelevance in Type Theory

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

Foundations of Software Science and Computation Structures
ETAPS 2011
Universität des Saarlandes
Saarbrücken, Germany
28 March 2011

Introduction

- Dependently typed = types + specifications + programs + proofs.

```
insert : (A : Set) -> A -> { l  : List A | sorted l }
                        -> { l' : List A | sorted l }
insert A a (nil | _) = (cons a nil | singleton_is_sorted)
...
```

- Program extraction should erase:
  - Type arguments.
  - Propositions.
  - Proofs.
- Letouzey's extraction does this for Coq.
- Depends on Prop/Set distinction.
- Alternative: proofs are dead code.

Type system for dead code

- A function argument is irrelevant if it is
  - not analyzed,
  - not returned.
- It may used in irrelevant argument of functions calls.
- Pfenning (LiCS 2001) and Reed (TPHOLs 2003): *Logical Framework with proof irrelevance.*
- This work: extension to *Martin-Löf Type Theory with universes*.
- Supports *large eliminations* (types defined by a program).
- Implemented in *Agda* (Norell, Danielsson, ...)

Proof Irrelevance in Agda

```
-- div' n m  computes  ceil(n / (m + 1))
div'  : {i : Size} → Nat i → Nat → Nat i
div'  zero    m = zero
div' (suc n) m = suc (div' (n - m) m)

-- div n m computes floor(n / m)
div : Nat → (m : Nat) → .(zero < m) → Nat
div n  zero    ()
div n (suc m) p  = div' (n - m) m
```

- The outcome of division `div` n m p does not depend on proof $p : 0 < m$.
- p only serves to eliminate impossible cases.
- At runtime p can be absent.

Proofs in Data Structures

```
Bounds = List A
_below_ : A → Bounds → Set
_above_ : A → Bounds → Set

data Tree (β γ : Bounds) : Set where
  leaf : Tree β γ
  node : (a : A) → .(a above β) → .(a below γ) →
         Tree β (a :: γ) → Tree (a :: β) γ → Tree β γ
```

- Proofs are irrelevant for equality: $\mathtt{node}\ a\ p\ q\ l\ r = \mathtt{node}\ a\ p'\ q'\ l\ r$.
- Agda (as opposed to Coq) has proof irrelevance at compile-time.

Irrelevant Function type $.A \to B$

Slogan: A term is a proof if computation does not depend on it.

- Conjunctions can be eliminated:

```
split : (.A → .B → C) → .(A × B) → C
split k (a , b) = k a b
```

- Matching can be translated away:

```
split : (.A → .B → C) → .(A × B) → C
split k p = k (fst p) (snd p)
```

- Disjunctions cannot be eliminated:

```
case : (.A → C) → (.B → C) → .(A ⊎ B) → C
case f g (inl a) = f a   -- type checking error
case f g (inr b) = g b
```

Ordinary Dependent Function Type

$$\frac{\Gamma \vdash A : \mathsf{Set} \qquad \Gamma, (x : A) \vdash B : \mathsf{Set}}{\Gamma \vdash (x : A) \to B : \mathsf{Set}}$$

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

$$\frac{\Gamma \vdash r : (x : A) \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B[s/x]}$$

$$\frac{\Gamma \vdash t : A \qquad (\Gamma \vdash A : \mathsf{Set}) =_{\beta\eta} (\Gamma \vdash B : \mathsf{Set})}{\Gamma \vdash t : B}$$

Irrelevant Dependent Function Type

$$\frac{\Gamma \vdash A : \mathsf{Set} \qquad \Gamma,\ .(x:A) \vdash B : \mathsf{Set}}{\Gamma \vdash .(x:A) \to B : \mathsf{Set}}$$

$$\text{no rule for } .(x:A) \in \Gamma \qquad \frac{\Gamma,\ .(x:A) \vdash t : B}{\Gamma \vdash \lambda x t : .(x:A) \to B}$$

$$\frac{\Gamma \vdash r : .(x:A) \to B \qquad \Gamma^{\oplus} \vdash s : A}{\Gamma \vdash r\,s : B[s/x]}$$

Resurrection $(-)^{\oplus}$ (Pfenning 2001) turns proof assumptions $.(x:A)$ into programs assumptions $(x:A)$.

Homogeneous Algorithmic $\beta\eta$-equality

- cf. Harper/Pfenning (TOCL 2005)

$$\Delta \vdash n \longleftrightarrow n' : T' \quad \text{infer } T' \text{ mode}$$
$$\Delta \vdash u \Longleftrightarrow u' : U' \quad \text{check against } U' \text{ mode}$$

- Dependent types lead to asymmetry.

$$\frac{\Delta \vdash n \longleftrightarrow n' : (x:U) \to T \qquad \Delta \vdash u \Longleftrightarrow u' : U}{\Delta \vdash n\,u \longleftrightarrow n'\,u' : T[u/x]}$$

- Vicious cycle:

transivity $\Leftarrow$ soundness $\Leftarrow$ subject reduction
$\Leftarrow$ function type injectivity $\Leftarrow$ completeness
$\Leftarrow$ semantic model $\Leftarrow$ transitivity

Heterogeneous Algorithmic $\beta\eta$-equality

- Mutual judgements:

$$\Delta \vdash n : T \longleftrightarrow \Delta' \vdash n' : T' \quad \text{infer mode}$$
$$\Delta \vdash u : U \Longleftrightarrow \Delta' \vdash u' : U' \quad \text{check mode}$$

- Ordinary application:

$$\frac{\Delta \vdash n : (x:U) \to T \longleftrightarrow \Delta' \vdash n' : (x:U') \to T' \qquad \Delta \vdash u : U \Longleftrightarrow \Delta' \vdash u' : U'}{\Delta \vdash n\,u : T[u/x] \longleftrightarrow \Delta' \vdash n'\,u' : T'[u'/x]}$$

- Irrelevant application:

$$\frac{\Delta \vdash n : .(x:U) \to T \longleftrightarrow \Delta' \vdash n' : .(x:U') \to T'}{\Delta \vdash n\,u : T[u/x] \longleftrightarrow \Delta' \vdash n'\,u' : T'[u'/x]}$$

On Paper (FoSSaCS'11)

- Kripke logical relation $\Delta \vdash t : T \circledS \Delta' \vdash t' : T'$.
- Entails declarative and algorithmic equality.
- Models declarative typing and equality rules.
- Proves consistency, normalization, decidability.

Implicit Calculus of Constructions

- ICC (Miquel, Barras, Bernardo) and EPTS (Mishra-Linger, Sheard)
- An irrelevant function argument can be relevant in the type.

$$\frac{\Gamma \vdash A : \mathrm{Set} \qquad \Gamma, x : A \vdash B : \mathrm{Set}}{\Gamma \vdash [x : A] \to B : \mathrm{Set}} \qquad \frac{\Gamma \vdash t : A}{\Gamma^{\oplus} \vdash A : \mathrm{Set}}$$

- Resembles Curry-Style terms.

$$\mathsf{cons} : [A : \mathit{Set}][n : \mathit{Nat}] \to A \to \mathsf{Vec}\, A\, n \to \mathsf{Vec}\, A\, (\mathsf{suc}\, n)$$
$$\mathsf{cons}\, A\, n\, a\, v = \mathsf{cons}\, A'\, n'\, a\, v$$

- Could not reconcile this with typed equality and large eliminations.

Conclusions

- Irrelevant arguments in Type Theory.
- Proofs can be discarded immediately after checking (open terms).
- Used in Agda to formalize categories (and algebraic structures).
- Solved performance issues.
- Program extraction discards more, but extracted programs (closed terms) cannot be used during equality checking.
- Distinct notions of irrelevance necessary.

Thanks to Hugo Herbelin & INRIA PI.R2, Bruno Barras, and Ulf Norell & Agda crowd.

Related Work

1. Proof Irrelevance in LF (Pfenning, Reed)
2. Bracket Types (Awodey, Bauer)
3. Uniform quantification (Berger, Schwichtenberg)
4. Program extraction in Coq (Paulin-Mohring, Letouzey)
5. Implicit Calculus of Constructions (Miquel, Barras, Bernardo)
6. Erasure Pure Type Systems (Mishra-Linger, Sheard)
7. Lightning (Brady, McBride)