

Irrelevance in Type Theory

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

TCS Oberseminar
LMU Munich
17 December 2010

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 \rightarrow Bounds \rightarrow Set

above : A \rightarrow Bounds \rightarrow Set

data Tree (β γ : Bounds) : Set where

leaf : Tree β γ

node : (a : A) \rightarrow .(a above β) \rightarrow .(a below γ) \rightarrow
 Tree β (a :: γ) \rightarrow Tree (a :: β) γ \rightarrow Tree β γ

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

Irrelevant Function type $.A \rightarrow B$

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

- Conjunctions can be eliminated:

$$\text{split} : (.A \rightarrow .B \rightarrow C) \rightarrow .(A \times B) \rightarrow C$$

$$\text{split } k \ (a , b) = k \ a \ b$$

- Matching can be translated away:

$$\text{split} : (.A \rightarrow .B \rightarrow C) \rightarrow .(A \times B) \rightarrow C$$

$$\text{split } k \ p = k \ (\text{fst } p) \ (\text{snd } p)$$

- Disjunctions cannot be eliminated:

$$\text{case} : (.A \rightarrow C) \rightarrow (.B \rightarrow C) \rightarrow .(A \uplus B) \rightarrow C$$

$$\text{case } f \ g \ (\text{inl } a) = f \ a$$

$$\text{case } f \ g \ (\text{inr } b) = g \ b$$

Ordinary Dependent Function Type

$$\frac{\Gamma \vdash A : \text{Set} \quad \Gamma, (x:A) \vdash B : \text{Set}}{\Gamma \vdash (x:A) \rightarrow B : \text{Set}}$$

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

$$\frac{\Gamma \vdash r : (x:A) \rightarrow B \quad \Gamma \vdash s : A}{\Gamma \vdash r s : B[s/x]}$$

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

Irrelevant Dependent Function Type

$$\frac{\Gamma \vdash A : \text{Set} \quad \Gamma, .(x:A) \vdash B : \text{Set}}{\Gamma \vdash .(x:A) \rightarrow B : \text{Set}}$$

no rule for $.(x:A) \in \Gamma$

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

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

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

Algorithmic $\beta\eta$ -equality

- Mutual judgements:

$$\begin{aligned} \Delta \vdash n : T &\longleftrightarrow \Delta' \vdash n' : T' && \text{infer mode} \\ \Delta \vdash u : U &\iff \Delta' \vdash u' : U' && \text{check mode} \end{aligned}$$

- Ordinary application:

$$\frac{\begin{array}{c} \Delta \vdash n : (x : U) \rightarrow T \longleftrightarrow \Delta' \vdash n' : (x : U') \rightarrow T' \\ \Delta \vdash u : U \iff \Delta' \vdash u' : U' \end{array}}{\Delta \vdash nu : T[u/x] \longleftrightarrow \Delta' \vdash n' u' : T'[u'/x]}$$

- Irrelevant application:

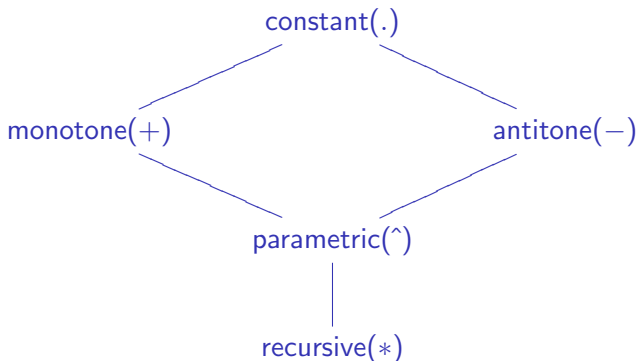
$$\frac{\Delta \vdash n : \lambda(x : U). T \longleftrightarrow \Delta' \vdash n' : \lambda(x : U'). T'}{\Delta \vdash nu : T[u/x] \longleftrightarrow \Delta' \vdash n' u' : T'[u'/x]}$$

On Paper (FoSSaCS'11)

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

Further Work: Combine with Polarities

- Function lattice:



- p -variant dependent function type $p(x:A) \rightarrow B$.
- Implemented in MiniAgda.

p -Variant Dependent Function Type

$$\frac{\Gamma \vdash A : \text{Set} \quad \Gamma, (x:A) \vdash B : \text{Set}}{\Gamma \vdash p(x:A) \rightarrow B : \text{Set}}$$

$$\frac{(x:A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\Gamma, p(x:A) \vdash t : B}{\Gamma \vdash \lambda x t : p(x:A) \rightarrow B}$$

$$\frac{\Gamma \vdash r : p(x:A) \rightarrow B \quad p^{-1}\Gamma \vdash s : A}{\Gamma \vdash r s : B[s/x]}$$

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

Related Work

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