Tarski and Coq

A. Assaf[1,2]

[1]INRIA Paris-Rocquencourt (Deducteam)
[2]Ecole Polytechnique

PPS Type Theory Work Group
7 January 2015

Tarski and Coq

Universes in Coq

- Infinite hierarchy

$$\text{Prop}, \text{Type}_0 : \text{Type}_1 : \text{Type}_2 : \ldots$$

- Cumulative

$$\text{Prop} \subseteq \text{Type}_0 \subseteq \text{Type}_1 \subseteq \text{Type}_2 : \ldots$$

$$\frac{\Gamma \vdash A : \text{Type}_i}{\Gamma \vdash A : \text{Type}_{i+1}}$$

Subtyping

- Relation $\leq$ between terms

$$\frac{}{\mathsf{Prop} \leq \mathsf{Type}_0} \qquad \frac{}{\mathsf{Type}_i \leq \mathsf{Type}_{i+1}}$$

$$\frac{A \equiv B}{A \leq B} \qquad \frac{B \leq C}{\Pi x : A.B \leq \Pi x : A.C}$$

- Subsumption rule

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

Problems with implicit subtyping

- Not syntax directed

$$\frac{\Gamma \vdash {\color{red}M} : A \qquad A \leq B}{\Gamma \vdash {\color{red}M} : B}$$

Problems with implicit subtyping

- Not syntax directed

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

- No type uniqueness

$$M : A \land M : B \;\not\Longrightarrow\; A \equiv B$$

Problems with implicit subtyping

- Not syntax directed

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

- No type uniqueness

$$M : A \wedge M : B \;\not\Rightarrow\; A \equiv B$$

- No subject reduction for minimal type

Example

$(\lambda x : \mathsf{Type}_2 . x)\ \mathsf{Type}_0 : \mathsf{Type}_2 \longrightarrow_\beta \mathsf{Type}_0 : \mathsf{Type}_1$

Explicit subtyping

- Explicit coercions

$$\uparrow_i : \mathsf{Type}_i \to \mathsf{Type}_{i+1}$$

- Only conversion rule

$$\frac{\Gamma \vdash M : A \qquad A \equiv B}{\Gamma \vdash M : B}$$

Explicit subtyping

- Explicit coercions

$$\uparrow_i : \mathsf{Type}_i \to \mathsf{Type}_{i+1}$$

- Only conversion rule

$$\frac{\Gamma \vdash M : A \qquad A \equiv B}{\Gamma \vdash M : B}$$

- Type uniqueness, subject reduction

Example

$(\lambda x : \mathsf{Type}_2 . x)\ (\uparrow_1 \mathsf{Type}_0) : \mathsf{Type}_2 \longrightarrow_\beta \uparrow_1 \mathsf{Type}_0 : \mathsf{Type}_2$

Martin-Lof's Intuitionistic Type Theory (ITT):

- Infinite hierarchy of *predicative* universes
- Cumulativity

Pure Type Systems (PTS):

- *Impredicativity*: System F, Calculus of constructions, ...
- No cumulativity

Universes in Coq

Calculus of Inductive Constructions (CIC):

- Infinite predicative universe hierarchy Type_i
- Impredicative universe Prop
- Cumulativity
- Inductive types
- (Universe polymorphism)

Intuitionistic Type Theory

Type formation rules

$$\frac{A \quad \text{type} \qquad x : A \vdash B \quad \text{type}}{\Pi x : A.B \quad \text{type}}$$

Introduction and elimination rules

$$\frac{x : A \vdash M : B}{\lambda x : A.M : \Pi x : A.B} \qquad \frac{M : \Pi x : A.B \qquad N : A}{M\,N : B\,[x \backslash N]}$$

(Typed) equalities

$$(\lambda x : A.M)\ N \equiv M\,[x \backslash N] \quad : \quad B\,[x \backslash N]$$

Russell vs. Tarski

- Russell style

$$\frac{}{\mathsf{U}_i \quad \text{type}} \qquad \frac{A : \mathsf{U}_i}{A \quad \text{type}}$$

$$\frac{}{\mathsf{U}_i : \mathsf{U}_{i+1}} \qquad \frac{A : \mathsf{U}_i}{A : \mathsf{U}_{i+1}}$$

- Tarski style

$$\frac{}{\mathsf{U}_i \quad \text{type}} \qquad \frac{A : \mathsf{U}_i}{\mathsf{T}_i\,(A) \quad \text{type}}$$

$$\frac{}{\mathsf{u}_i : \mathsf{U}_{i+1}} \qquad \frac{A : \mathsf{U}_i}{\uparrow_i (A) : \mathsf{U}_{i+1}}$$

Tarski style universes

- u_i is a *code* for U_i in U_{i+1}
- $\mathsf{T}_i\,()$ is a *decoding* function

$$\begin{aligned} \mathsf{T}_{i+1}\,(\mathsf{u}_i) &\equiv \mathsf{U}_i \\ \mathsf{T}_{i+1}\,(\uparrow_i(A)) &\equiv \mathsf{T}_i\,(A) \end{aligned}$$

Tarski style universes

- u_i is a *code* for U_i in U_{i+1}
- $\mathsf{T}_i\,()$ is a *decoding* function

$$\begin{aligned} \mathsf{T}_{i+1}\,(\mathsf{u}_i) &\equiv \mathsf{U}_i \\ \mathsf{T}_{i+1}\,(\uparrow_i (A)) &\equiv \mathsf{T}_i\,(A) \end{aligned}$$

- $\pi_i\, x : A.B$ is a code for product types in U_i

$$\frac{A : \mathsf{U}_i \qquad x : A \vdash B : \mathsf{U}_i}{\pi_i\, x : A.B : \mathsf{U}_i}$$

$$\mathsf{T}_i\,(\pi_i\, x : A.B) \equiv \Pi x : \mathsf{T}_i\,(A).\,\mathsf{T}_i\,(B)$$

Russell vs. Tarski

- Russell "informal version" of Tarski
- Erasure function $|M|$

Theorem

If $\Gamma \vdash_{Tarski} M : A$ *then* $|\Gamma| \vdash_{Russell} |M| : |A|$.

Russell vs. Tarski

- Russell "informal version" of Tarski
- Erasure function $|M|$

Theorem

If $\Gamma \vdash_{Tarski} M : A$ *then* $|\Gamma| \vdash_{Russell} |M| : |A|$.

Converse?

Notations

	ITT	Coq	LF	$\lambda\Pi$ modulo*	Assaf
Universes	U	Type	Type	$\mathsf{U}_{\mathsf{Type}}$	
Decoding	$\mathsf{T}()$		$\mathsf{El}\,()$	$\varepsilon_{\mathsf{Type}}()$	
Product codes	π			$\dot{\pi}_{\mathsf{Type}}$	
Universe codes	u			$\dot{\mathsf{Type}}$	
Code lifting	t				$\uparrow$

* Cousineau and Dowek, *Embedding pure type systems in the lambda-Pi calculus modulo*, TLCA 2007

Counter-example (Assaf 2014)

In the context

$$\begin{array}{rcl} a, b & : & \mathsf{Type}_0 \\ p, q & : & \mathsf{Type}_1 \to \mathsf{Type}_1 \\ f & : & \Pi a, b : \mathsf{Type}_1 .\, p\,(\Pi x : a.b) \\ g & : & \Pi c : \mathsf{Type}_0 .\, p\,(c) \to q\,(c) \end{array}$$

we have

$$g\,(\Pi x : a.b)\,(f\,a\,b) \quad : \quad q\,(\Pi x : a.b)$$

Counter-example (Assaf 2014)

In the context

$$\begin{array}{rcl} a, b & : & \mathsf{Type}_0 \\ p, q & : & \mathsf{Type}_1 \to \mathsf{Type}_1 \\ f & : & \Pi a, b : \mathsf{Type}_1 .\, \mathsf{T}_1\,(p\,(\pi_1\,x : \uparrow_0 a.\uparrow_0 b)) \\ g & : & \Pi c : \mathsf{Type}_0 .\, \mathsf{T}_1\,(p\,(\uparrow_0 c)) \to \mathsf{T}_1\,(q\,(\uparrow_0 c)) \end{array}$$

we have

$$g\,(\pi_0\,x : a.b)\,(f\,(\uparrow_0 a)\,(\uparrow_0 b)) \quad : \quad \mathsf{T}_1\,(q\,(\uparrow_0\,(\pi_0\,x : a.b)))$$

Counter-example (Assaf 2014)

In the context

$$\begin{array}{rcl} a, b & : & \mathsf{Type}_0 \\ p, q & : & \mathsf{Type}_1 \to \mathsf{Type}_1 \\ f & : & \Pi a, b : \mathsf{Type}_1 .\, \mathsf{T}_1\,(p\,(\pi_1\, x : \uparrow_0 a. \uparrow_0 b)) \\ g & : & \Pi c : \mathsf{Type}_0 .\, \mathsf{T}_1\,(p\,(\uparrow_0 c)) \to \mathsf{T}_1\,(q\,(\uparrow_0 c)) \end{array}$$

we have

$$\begin{array}{rcll} g\,(\pi_0\, x : a.b)\,(f\,(\uparrow_0 a)\,(\uparrow_0 b)) & \not: & \mathsf{T}_1\,(q\,(\uparrow_0\,(\pi_0\, x : a.b))) & \times \\ f\,(\uparrow_0 a)\,(\uparrow_0 b) & : & \mathsf{T}_1\,(p\,(\pi_1\, x : \uparrow_0 a. \uparrow_0 b)) & \end{array}$$

Culprit: Multiple representations

Different typing derivations yield different terms

$$\dfrac{\dfrac{A : \mathsf{Type}_0 \qquad x : A \vdash B : \mathsf{Type}_0}{\Pi x : A.B : \mathsf{Type}_0}}{\Pi x : A.B : \mathsf{Type}_1} \qquad \uparrow_i (\pi_i\, x : a.b)$$

$$\dfrac{\dfrac{A : \mathsf{Type}_0}{A : \mathsf{Type}_1} \qquad \dfrac{x : A \vdash B : \mathsf{Type}_0}{x : A \vdash B : \mathsf{Type}_1}}{\Pi x : A.B : \mathsf{Type}_1} \qquad \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b$$

Anti-solutions

- Consider that Russell style is unsound
- Put additional annotations on Π

Solution: Reflect equalities

Add equation

$$\uparrow_i (\pi_i\, x : a.b) \quad \equiv \quad \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b$$

Solution: Reflect equalities

Add equation

$$\uparrow_i (\pi_i\, x : a.b) \quad \equiv \quad \pi_{i+1}\, x : \uparrow_i a . \uparrow_i b$$

How does this help?

$$\begin{aligned}
a, b &: \mathsf{Type}_0 \\
p, q &: \mathsf{Type}_1 \to \mathsf{Type}_1 \\
f &: \Pi a, b : \mathsf{Type}_1 .\, \mathsf{T}_1\, (p\, (\pi_1\, x : \uparrow_0 a . \uparrow_0 b)) \\
g &: \Pi c : \mathsf{Type}_0 .\, \mathsf{T}_1\, (p\, (\uparrow_0 c)) \to \mathsf{T}_1\, (q\, (\uparrow_0 c))
\end{aligned}$$

$$\begin{aligned}
g\, (\pi_0\, x : a.b)\, (f\, (\uparrow_0 a)\, (\uparrow_0 b)) &: \mathsf{T}_1\, (q\, (\uparrow_0 (\pi_0\, x : a.b))) \\
f\, (\uparrow_0 a)\, (\uparrow_0 b) &: \mathsf{T}_1\, (p\, (\pi_1\, x : \uparrow_0 a . \uparrow_0 b))
\end{aligned}$$

Solution: Reflect equalities

Add equation

$$\uparrow_i (\pi_i\, x : a.b) \quad \equiv \quad \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b$$

How does this help?

$$\begin{array}{rcl} a, b & : & \mathsf{Type}_0 \\ p, q & : & \mathsf{Type}_1 \to \mathsf{Type}_1 \\ f & : & \Pi a, b : \mathsf{Type}_1.\, \mathsf{T}_1\, (p\, (\pi_1\, x : \uparrow_0 a. \uparrow_0 b)) \\ g & : & \Pi c : \mathsf{Type}_0.\, \mathsf{T}_1\, (p\, (\uparrow_0 c)) \to \mathsf{T}_1\, (q\, (\uparrow_0 c)) \end{array}$$

$$\begin{array}{rcl} g\, (\pi_0\, x : a.b)\, (f\, (\uparrow_0 a)\, (\uparrow_0 b)) & : & \mathsf{T}_1\, (q\, (\uparrow_0 (\pi_0\, x : a.b))) \quad \checkmark \\ f\, (\uparrow_0 a)\, (\uparrow_0 b) & : & \mathsf{T}_1\, (p\, (\uparrow_0 (\pi_0\, x : a.b))) \end{array}$$

A history of reflecting equalities

Reflection known but not used

- P. Martin-Löf, *Intuitionistic type theory*, 1984
- E. Palmgren, *On universes in type theory*, 1993

  "The usefulness of reflecting equalities of sets is not clear."

- Z. Luo, *Computation and reasoning*, 1994

  "We may also enforce the name uniqueness [...]. However, this is not essential."

Properties

- Terms must have a unique representation

Theorem (Canonicity)

If $|M| \equiv |M'|$ *then* $M \equiv M'$.

- Essential for completeness

Theorem

If $\Gamma \vdash_{Russell} M : A$ *then* $\Gamma' \vdash_{Tarski} M' : A'$ *such that* $|\Gamma'| = \Gamma,\ |M'| = M,\ |A'| = A$.

Deriving Tarski

To understand the Tarski style:

Deriving Tarski

To understand the Tarski style:

1. Start with the usual types

$$\frac{}{\mathsf{Nat} \quad \text{type}} \qquad \frac{A \quad \text{type} \qquad x : A \vdash B \quad \text{type}}{\Pi x : A.B \quad \text{type}}$$

To understand the Tarski style:

1. Start with the usual types

$$\frac{}{\mathsf{Nat}\quad \text{type}} \qquad \frac{A \quad \text{type} \qquad x : A \vdash B \quad \text{type}}{\Pi x : A.B \quad \text{type}}$$

2. Add a universe reflecting all the *currently existing types*

$$\frac{}{\mathsf{U}_0 \quad \text{type}} \qquad \frac{A : \mathsf{U}_0}{\mathsf{T}_0\,(A) \quad \text{type}}$$

$$\frac{}{\mathsf{nat}_0 : \mathsf{U}_0} \qquad \frac{A : \mathsf{U}_0 \qquad x : \mathsf{T}_0\,(A) \vdash B : \mathsf{U}_0}{\pi_0\, x : A.B : \mathsf{U}_0}$$

$$\begin{aligned} \mathsf{T}_0\,(\mathsf{nat}_0) \quad &\equiv \quad \mathsf{Nat} \\ \mathsf{T}_0\,(\pi_0\, x : A.B) \quad &\equiv \quad \Pi x : \mathsf{T}_0\,(A).\,\mathsf{T}_0\,(B\,x) \end{aligned}$$

3 Add another universe reflecting all the *currently existing types*...

$$\frac{}{\mathsf{U}_1 \quad \text{type}} \qquad \frac{A : U_1}{\mathsf{T}_1\,(A) \quad \text{type}}$$

$$\frac{}{\mathsf{nat}_1 : \mathsf{U}_1} \qquad \frac{A : \mathsf{U}_1 \qquad x : \mathsf{T}_1\,(A) \vdash B : \mathsf{U}_1}{\pi_1\,x : A.B : \mathsf{U}_1}$$

$$\begin{aligned} \mathsf{T}_1\,(\mathsf{nat}_1) &\equiv \mathsf{Nat} \\ \mathsf{T}_1\,(\pi_1\,x : A.B) &\equiv \Pi x : \mathsf{T}_1\,(A).\,\mathsf{T}_1\,(B\,x) \end{aligned}$$

3 Add another universe reflecting all the *currently existing types*...

$$\frac{}{\mathsf{U}_1 \quad \text{type}} \qquad \frac{A : U_1}{\mathsf{T}_1\,(A) \quad \text{type}}$$

$$\frac{}{\mathsf{nat}_1 : \mathsf{U}_1} \qquad \frac{A : \mathsf{U}_1 \qquad x : \mathsf{T}_1\,(A) \vdash B : \mathsf{U}_1}{\pi_1\, x : A.B : \mathsf{U}_1}$$

$$\begin{aligned} \mathsf{T}_1\,(\mathsf{nat}_1) &\equiv \mathsf{Nat} \\ \mathsf{T}_1\,(\pi_1\, x : A.B) &\equiv \Pi x : \mathsf{T}_1\,(A).\, \mathsf{T}_1\,(B\, x) \end{aligned}$$

$$\frac{}{\mathsf{u}_0 : \mathsf{U}_1} \qquad \frac{A : \mathsf{U}_0}{\mathsf{t}_0\,(A) : \mathsf{U}_1}$$

$$\begin{aligned} \mathsf{T}_1\,(\mathsf{u}_0) &\equiv \mathsf{U}_0 \\ \mathsf{T}_1\,(\mathsf{t}_0\,(A)) &\equiv \mathsf{T}_0\,(A) \end{aligned}$$

Deriving Tarski

… and all the *currently existing type equalities*!

$$\begin{aligned} \mathsf{t}_0\,(\mathsf{nat}_0) &\equiv \mathsf{nat}_1 \\ \mathsf{t}_0\,(\pi_0\,x : A.B) &\equiv \pi_1\,x : \mathsf{t}_0\,(A)\,.\mathsf{t}_0\,(B) \end{aligned}$$

Deriving Tarski

... and all the *currently existing type equalities*!

$$\begin{aligned} \mathsf{t}_0\,(\mathsf{nat}_0) &\equiv \mathsf{nat}_1 \\ \mathsf{t}_0\,(\pi_0\, x : A.B) &\equiv \pi_1\, x : \mathsf{t}_0\,(A)\,.\mathsf{t}_0\,(B) \end{aligned}$$

N.B.: A miracle just happened.

... and all the *currently existing type equalities*!

$$\mathsf{t}_0\,(\mathsf{nat}_0) \equiv \mathsf{nat}_1$$
$$\mathsf{t}_0\,(\pi_0\, x : A.B) \equiv \pi_1\, x : \mathsf{t}_0\,(A)\,.\mathsf{t}_0\,(B)$$

N.B.: A miracle just happened.

4 Iterate for fun and profit!

ITT vs. Coq

- Impredicativity
- Judgmental equality vs computational equality
- Operational semantics based on reductions
- (Universe polymorphism)

Impredicativity

- Russell style

$$\frac{}{\mathsf{Prop} : \mathsf{Type}_1} \qquad \frac{A : \mathsf{Prop}}{A : \mathsf{Type}_0}$$

$$\frac{A : \mathsf{Type}_i \qquad x : A \vdash B : \mathsf{Prop}}{\Pi x : A.B : \mathsf{Prop}}$$

Impredicativity

- Russell style

$$\frac{}{\mathsf{Prop} : \mathsf{Type}_1} \qquad \frac{A : \mathsf{Prop}}{A : \mathsf{Type}_0}$$

$$\frac{A : \mathsf{Type}_i \qquad x : A \vdash B : \mathsf{Prop}}{\Pi x : A.B : \mathsf{Prop}}$$

- Tarski style

$$\frac{}{\mathsf{prop} : \mathsf{Type}_1} \qquad \frac{A : \mathsf{Prop}}{\uparrow_{\mathsf{Prop}} A : \mathsf{Type}_0}$$

$$\frac{A : \mathsf{Type}_i \qquad x : A \vdash B : \mathsf{Prop}}{\forall_i\, x : A.B : \mathsf{Prop}}$$

Impredicativity

Circularity:

- Prop is included in Type_0,
- which is included in Type_1,
- which is included in Type_2,
- ...
- all of which can be injected in Prop with a product!

Impredicativity

Circularity:

- Prop is included in Type_0,
- which is included in Type_1,
- which is included in Type_2,
- ...
- all of which can be injected in Prop with a product!

Step-by-step construction does not work anymore!

Impredicativity

Circularity:

- Prop is included in Type_0,
- which is included in Type_1,
- which is included in Type_2,
- ...
- all of which can be injected in Prop with a product!

Step-by-step construction does not work anymore!

Solution: Look at multiplicity of typing derivations.

Prop ambiguity 1

Ambiguity in the level of the argument type

$$\frac{A : \mathsf{Type}_i \qquad x : A \vdash B : \mathsf{Prop}}{\Pi x : A.B : \mathsf{Prop}} \qquad \forall_i\, x : A.B$$

$$\frac{\dfrac{A : \mathsf{Type}_i}{A : \mathsf{Type}_{i+1}} \qquad x : A \vdash B : \mathsf{Prop}}{\Pi x : A.B : \mathsf{Prop}} \qquad \forall_{i+1}\, x : \uparrow_i A.B$$

Prop ambiguity 2

Ambiguity in the level of the product

$$\dfrac{\dfrac{A : \mathsf{Type}_i \qquad x : A \vdash B : \mathsf{Prop}}{\Pi x : A.B : \mathsf{Prop}}}{\Pi x : A.B : \mathsf{Type}_i} \qquad \uparrow^{(i)}_{\mathsf{Prop}} (\forall_i\, x : A.B)$$

$$\dfrac{A : \mathsf{Type}_i \qquad \dfrac{x : A \vdash B : \mathsf{Prop}}{x : A \vdash B : \mathsf{Type}_i}}{\Pi x : A.B : \mathsf{Type}_i} \qquad \pi_i\, x : A. \uparrow^{(i)}_{\mathsf{Prop}} B$$

Prop equalities

Add equations

$$\forall_{i+1}\, x : {\color{blue}\uparrow_i} A.B \quad \equiv \quad \forall_i\, x : A.B$$

$$ {\color{blue}\uparrow^{(i)}_{\mathsf{Prop}}} (\forall_i\, x : A.B) \quad \equiv \quad \pi_i\, x : A. {\color{blue}\uparrow^{(i)}_{\mathsf{Prop}}} B$$

Prop equalities

Add equations

$$\forall_{i+1}\, x : \uparrow_i A.B \quad \equiv \quad \forall_i\, x : A.B$$

$$\uparrow^{(i)}_{\mathsf{Prop}} (\forall_i\, x : A.B) \quad \equiv \quad \pi_i\, x : A. \uparrow^{(i)}_{\mathsf{Prop}} B$$

Theorem

If $\Gamma \vdash_{Russell} M : A$ *then* $\Gamma' \vdash_{Tarski} M' : A'$ *such that* $|\Gamma'| = \Gamma,\ |M'| = M,\ |A'| = A.$

Uniform equalities

- $s_1 \to s_2$ rules of the PTS

$$s_1 \to \mathsf{Prop} = \mathsf{Prop} \qquad \mathsf{Prop} \to s_2 = s_2 \qquad \mathsf{Type}_i \to \mathsf{Type}_j = \mathsf{Type}_{\max(i,j)}$$

- $s_1 \vee s_2$ join of the $\subseteq$ relation

$$s_1 \vee \mathsf{Prop} = s_1 \qquad \mathsf{Prop} \vee s_2 = s_2 \qquad \mathsf{Type}_i \vee \mathsf{Type}_j = \mathsf{Type}_{\max(i,j)}$$

- Single equality

$$\uparrow_{s_1 \to s_2}^{s_3 \to s_4} (\pi_{s_1, s_2}\, x : A.B) \quad \equiv \quad \pi_{s_1 \vee s_3, s_2 \vee s_4}\, x : \uparrow_{s_1}^{s_3} A. \uparrow_{s_2}^{s_4} B$$

Conversion

Judgmental equality

- Typed
- Could be undecidable

Computational equality

- Untyped
- Algorithmic aspect (e.g. based on reductions)
- Conditions for decidability (e.g. confluence + SN)

Conversion

Judgmental equality

- Typed
- Could be undecidable

Computational equality

- Untyped
- Algorithmic aspect (e.g. based on reductions)
- Conditions for decidability (e.g. confluence + SN)

Theorem (Herbelin and Siles 2012)

The two are equivalent for pure type systems.

Computational equality à la Tarski

To decide equivalence in the Tarski style, we can:

- erase and use the Russell style,

Computational equality à la Tarski

To decide equivalence in the Tarski style, we can:

- erase and use the Russell style,
- or devise an algorithm working directly in the Tarski style,

To decide equivalence in the Tarski style, we can:

- erase and use the Russell style,
- or devise an algorithm working directly in the Tarski style,
- or even try to specify everything with reduction rules only.

Reduction rules

Operational semantics based on reductions

$$M \longrightarrow_\beta N$$

Transform equations into rewrite rules

$$\begin{array}{rcl} \mathsf{T}_{i+1}\,(\mathsf{type}_i) & \equiv & \mathsf{Type}_i \\ \mathsf{T}_i\,(\pi_i\, x : A.B) & \equiv & \Pi x : \mathsf{T}_i\,(A).\, \mathsf{T}_i\,(B) \\ \mathsf{T}_{i+1}\,(\uparrow_i A) & \equiv & \mathsf{T}_i\,(A) \end{array}$$

Reduction rules

Operational semantics based on reductions

$$M \longrightarrow_\beta N$$

Transform equations into rewrite rules

$$\begin{array}{rcl} \mathsf{T}_{i+1}\,(\mathsf{type}_i) & \longrightarrow & \mathsf{Type}_i \\ \mathsf{T}_i\,(\pi_i\,x : A.B) & \longrightarrow & \Pi x : \mathsf{T}_i\,(A).\,\mathsf{T}_i\,(B) \\ \mathsf{T}_{i+1}\,(\uparrow_i A) & \longrightarrow & \mathsf{T}_i\,(A) \end{array}$$

With Type

Distributing $\uparrow_i$ is enough

$$\uparrow_i (\pi_i\, x : a.b) \quad \equiv \quad \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b$$

With Type

Distributing $\uparrow_i$ is enough

$$\uparrow_i (\pi_i\, x : a.b) \quad \longrightarrow \quad \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b$$

With Prop

- Distributing $\uparrow_i$ breaks confluence because of the rule

$$\forall_{i+1}\, x : \uparrow_i A.B \quad \longrightarrow \quad \forall_i\, x : A.B$$

With Prop

- Distributing $\uparrow_i$ breaks confluence because of the rule

$$\forall_{i+1}\, x : \uparrow_i A.B \quad \longrightarrow \quad \forall_i\, x : A.B$$

- Need to raise $\uparrow$ to the top

$$\begin{aligned} \uparrow_i (\pi_i\, x : a.b) \quad &\longleftarrow \quad \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b \\ \forall_i\, x : A.B \quad &\longleftarrow \quad \forall_{i+1}\, x : \uparrow_i A.B \\ \uparrow^{(i)}_{\mathsf{Prop}} (\forall_i\, x : A.B) \quad &\longleftarrow \quad \pi_i\, x : A. \uparrow^{(i)}_{\mathsf{Prop}} B \end{aligned}$$

With Prop

- Distributing $\uparrow_i$ breaks confluence because of the rule

$$\forall_{i+1}\, x : \uparrow_i A.B \quad \longrightarrow \quad \forall_i\, x : A.B$$

- Need to raise $\uparrow$ to the top

$$\begin{array}{rcl} \uparrow_i (\pi_i\, x : a.b) & \longleftarrow & \pi_{i+1}\, x : \uparrow_i a. \uparrow_i b \\ \forall_i\, x : A.B & \longleftarrow & \forall_{i+1}\, x : \uparrow_i A.B \\ \uparrow^{(i)}_{\mathsf{Prop}} (\forall_i\, x : A.B) & \longleftarrow & \pi_i\, x : A. \uparrow^{(i)}_{\mathsf{Prop}} B \end{array}$$

- Corresponds to minimal typing!

What else is there?

Inductive types: No problem (Luo 1994)

- Add equations to ensure canonicity between codes at different levels,
- or use *uniform constructions* (a single code that can be lifted).

What else is there?

Inductive types: No problem (Luo 1994)

- Add equations to ensure canonicity between codes at different levels,
- or use *uniform constructions* (a single code that can be lifted).

Universe polymorphism:

- Need to handle algebraic universe expressions.
- Conversion based on reductions seems impossible (AC, idempotence).
- Need additional equations for polymorphic constants to ensure canonicity (constant definitions, inductive types, ...).

Conclusion

- **Russell style** = implicit, **Tarski style** = explicit
- Tarski $\implies$ Russell **trivially**
- Tarski $\impliedby$ Russell **only under proper conditions**

Conclusion

Russell style:

- Implicit
- "Informal"
- "Bad" properties:
  - Not syntax directed
  - No type uniqueness
  - No minimal type preservation
- Simple conversion

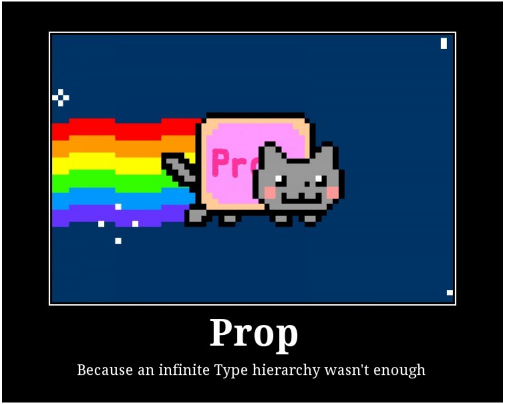
Conclusion

Tarski style:

- Explicit
- "Formal"
- All the usual "good" properties
- Simple conversion in ITT
- Equational theory more complex with Prop

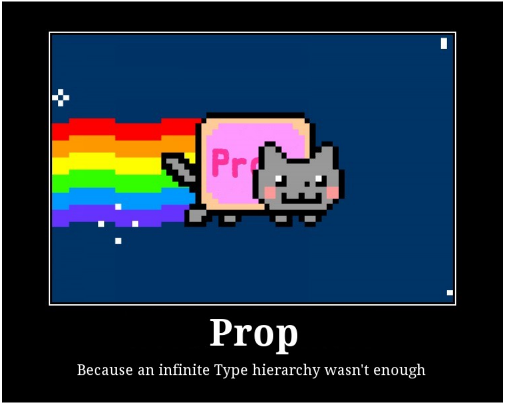
Conclusion

Prop is as annoying as ever.



Conclusion

Prop is as annoying as ever.



Thanks!

References

Martin-Löf
Intuitionistic type theory
Bibliopolis Naples, 1984

E. Palmgren
On universes in type theory
In Twenty-five years of constructive type theory
Oxford University Press, 1998

Z. Luo
Computation and Reasoning: A Type Theory for Computer Science,
Oxford University Press, 1994

H. Herbelin and V. Siles
Pure Type System conversion is always typable
In Journal of Functional Programming 22-02, 2012

A. Assaf
A Calculus of constructions with explicit subtyping
submitted to Post-proceedings of TYPES 2014