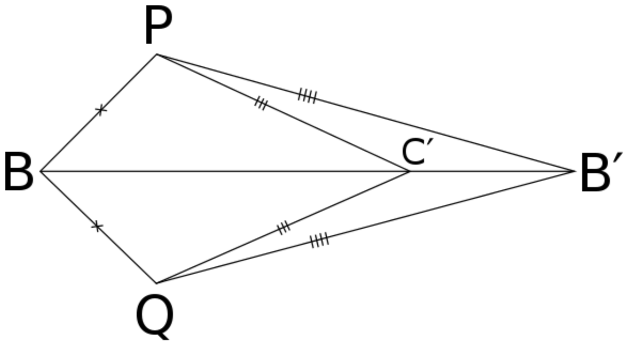
Coq à la Tarksi

A predicative calculus of constructions with explicit subtyping

A. Assaf[1]

[1] Deducteam, INRIA Paris-Rocquencourt

TYPES 2014


P
B
C′
B′
Q

Coq à la Tarski

Universes in Coq

- Infinite hierarchy

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

- Cumulative

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

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

Subtyping

- Relation $\leq$ between terms

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

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

- Subsumption rule

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

Problems with implicit subtyping

- Not syntax directed

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

Problems with implicit subtyping

- Not syntax directed

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

- No type uniqueness

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

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

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

Explicit subtyping

- Explicit coercions

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

- Only conversion rule

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

Explicit subtyping

- Explicit coercions

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

- Only conversion rule

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

- Type uniqueness, subject reduction

Example

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

Problem with higher-order types

In the context

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

we have

$$f\,(x\,(\quad a)) \quad : \quad b$$

Problem with higher-order types

In the context

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

we have

$$f\,(x\,(\uparrow_0 a)) \quad : \quad b$$

Problem with higher-order types

In the context

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

we have

$$\begin{array}{rcll} f\,(x\,(\uparrow_0 a)) & \not: & b & \times \\ x\,(\uparrow_0 a) & : & \uparrow_0 a & \end{array}$$

Attempt 1: adding equations

- Need to identify $M : \uparrow_i A$ with $M : A$.

Attempt 1: adding equations

- Need to identify $M : \uparrow_i A$ with $M : A$.
- Add equation
$$\uparrow_i A \equiv A$$
- Breaks subject reduction! $\times$

Attempt 2: adding coercions

- Need to identify $M : \uparrow_i A$ with $M : A$.

Attempt 2: adding coercions

- Need to identify $M : \uparrow_i A$ with $M : A$.
- Add another coercion
$$\downarrow_i^A : (\uparrow_i A) \to A$$
- What about $M : \downarrow_i A$? Add another coercion? ×

Solution: Oppan Tarski style!

- Russell style (Coq)

$$\frac{}{\mathtt{Type}_i \quad type} \qquad \frac{A : \mathtt{Type}_i}{A \quad type}$$

$$\frac{}{\mathtt{Type}_i : \mathtt{Type}_{i+1}} \qquad \frac{A : \mathtt{Type}_i}{A : \mathtt{Type}_{i+1}}$$

- Tarski style

$$\frac{}{\mathtt{Type}_i \quad type} \qquad \frac{A : \mathtt{Type}_i}{\varepsilon_i\,(A) \quad type}$$

$$\frac{}{\mathtt{type}_i : \mathtt{Type}_{i+1}} \qquad \frac{A : \mathtt{Type}_i}{\uparrow_i (A) : \mathtt{Type}_{i+1}}$$

Tarski style universes

- $\texttt{type}_i$ is a *code* for $\texttt{Type}_i$ in $\texttt{Type}_{i+1}$
- $\varepsilon_i\,()$ is a *decoding* function

$$\begin{aligned} \varepsilon_i\,(\texttt{type}_i) &\equiv \texttt{Type}_i \\ \varepsilon_{i+1}\,(\uparrow_i(A)) &\equiv \varepsilon_i\,(A) \end{aligned}$$

Tarski style universes

- $\mathtt{type}_i$ is a *code* for $\mathtt{Type}_i$ in $\mathtt{Type}_{i+1}$
- $\varepsilon_i\,()$ is a *decoding* function

$$\begin{aligned}\varepsilon_i\,(\mathtt{type}_i) &\equiv \mathtt{Type}_i\\ \varepsilon_{i+1}\,(\uparrow_i(A)) &\equiv \varepsilon_i\,(A)\end{aligned}$$

- How does this help?

$$\begin{aligned}a, b &: \mathtt{Type}_0\\ x &: \Pi\,(c : \mathtt{Type}_1)\,.\,\varepsilon_1\,(c)\\ f &: \varepsilon_0\,(a) \to \varepsilon_0\,(b)\end{aligned}$$

$$\begin{aligned}f\,(x\,(\uparrow_0 a)) &: \varepsilon_0\,(b)\\ x\,(\uparrow_0 a) &: \varepsilon_1\,(\uparrow_0 a)\end{aligned}$$

Tarski style universes

- $\texttt{type}_i$ is a *code* for $\texttt{Type}_i$ in $\texttt{Type}_{i+1}$
- $\varepsilon_i\left(\right)$ is a *decoding* function

$$\begin{aligned} \varepsilon_i\left(\texttt{type}_i\right) &\equiv \texttt{Type}_i \\ \varepsilon_{i+1}\left(\uparrow_i\left(A\right)\right) &\equiv \varepsilon_i\left(A\right) \end{aligned}$$

- How does this help?

$$\begin{aligned} a, b &: \texttt{Type}_0 \\ x &: \Pi\left(c : \texttt{Type}_1\right).\, \varepsilon_1\left(c\right) \\ f &: \varepsilon_0\left(a\right) \to \varepsilon_0\left(b\right) \end{aligned}$$

$$\begin{aligned} f\left(x\left(\uparrow_0 a\right)\right) &: \varepsilon_0\left(b\right) \quad \checkmark \\ x\left(\uparrow_0 a\right) &: \varepsilon_0\left(a\right) \end{aligned}$$

Tarski vs Russell

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

Theorem (Soundness)

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

Tarski vs Russell

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

Theorem (Soundness)

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

- Completeness?

Dependent products

- $\pi_i\,(x : A)\,.B$ is a *code* for $\Pi\,(x : A)\,.B$ in $\mathtt{Type}_i$

$$\frac{A : \mathtt{Type}_i \qquad B : \mathtt{Type}_i}{\pi_i\,(x : A)\,.B : \mathtt{Type}_i}$$

- $\varepsilon_i\,()$ is a *decoding* function

$$\varepsilon_i\,(\pi_i\,(x : A)\,.B) \quad \equiv \quad \Pi\,(x : A)\,.B$$

Multiple representations

- Different typing derivations yield different terms

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

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

Problem with dependent types

In the context

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

we have

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

Problem with dependent types

In the context

$$\begin{array}{rcl} a, b & : & \mathtt{Type}_0 \\ p, q & : & \mathtt{Type}_1 \to \mathtt{Type}_1 \\ f & : & \Pi\,(a, b : \mathtt{Type}_1)\,.\,\varepsilon_1\,(p\,(\pi_1\,(x : \uparrow_0 a)\,.\uparrow_0 b)) \\ g & : & \Pi\,(c : \mathtt{Type}_0)\,.\,\varepsilon_1\,(p\,(\uparrow_0 c)) \to \varepsilon_1\,(q\,(\uparrow_0 c)) \end{array}$$

we have

$$g\,(\pi_0\,(x : a)\,.b)\,(f\,(\uparrow_0 a)\,(\uparrow_0 b)) \quad : \quad \varepsilon_1\,(q\,(\uparrow_0\,(\pi_0\,(x : a)\,.b)))$$

Problem with dependent types

In the context

$$\begin{array}{rcl} a, b & : & \mathtt{Type}_0 \\ p, q & : & \mathtt{Type}_1 \to \mathtt{Type}_1 \\ f & : & \Pi\,(a, b : \mathtt{Type}_1)\,.\,\varepsilon_1\,(p\,(\pi_1\,(x : \uparrow_0 a)\,.\uparrow_0 b)) \\ g & : & \Pi\,(c : \mathtt{Type}_0)\,.\,\varepsilon_1\,(p\,(\uparrow_0 c)) \to \varepsilon_1\,(q\,(\uparrow_0 c)) \end{array}$$

we have

$$\begin{array}{rcl} g\,(\pi_0\,(x : a)\,.b)\,(f\,(\uparrow_0 a)\,(\uparrow_0 b)) & \not: & \varepsilon_1\,(q\,(\uparrow_0\,(\pi_0\,(x : a)\,.b))) \quad \times \\ f\,(\uparrow_0 a)\,(\uparrow_0 b) & : & \varepsilon_1\,(p\,(\pi_1\,(x : \uparrow_0 a)\,.\uparrow_0 b)) \end{array}$$

Solution: Reflecting equalities

- Add equation

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

Solution: Reflecting 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?

$$\begin{array}{rcl} a, b & : & \texttt{Type}_0 \\ p, q & : & \texttt{Type}_1 \to \texttt{Type}_1 \\ f & : & \Pi\,(a, b : \texttt{Type}_1)\,.\,\varepsilon_1\,(p\,(\pi_1\,(x : \uparrow_0 a)\,.\uparrow_0 b)) \\ g & : & \Pi\,(c : \texttt{Type}_0)\,.\,\varepsilon_1\,(p\,(\uparrow_0 c)) \to \varepsilon_1\,(q\,(\uparrow_0 c)) \end{array}$$

$$\begin{array}{rcl} g\,(\pi_0\,(x : a)\,.b)\,(f\,(\uparrow_0 a)\,(\uparrow_0 b)) & : & \varepsilon_1\,(q\,(\uparrow_0\,(\pi_0\,(x : a)\,.b))) \\ f\,(\uparrow_0 a)\,(\uparrow_0 b) & : & \varepsilon_1\,(p\,(\pi_1\,(x : \uparrow_0 a)\,.\uparrow_0 b)) \end{array}$$

Solution: Reflecting 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?

$$\begin{array}{rcl} a, b & : & \texttt{Type}_0 \\ p, q & : & \texttt{Type}_1 \to \texttt{Type}_1 \\ f & : & \Pi\,(a, b : \texttt{Type}_1)\,.\,\varepsilon_1\,(p\,(\pi_1\,(x : \uparrow_0 a)\,.\uparrow_0 b)) \\ g & : & \Pi\,(c : \texttt{Type}_0)\,.\,\varepsilon_1\,(p\,(\uparrow_0 c)) \to \varepsilon_1\,(q\,(\uparrow_0 c)) \end{array}$$

$$\begin{array}{rcl} g\,(\pi_0\,(x : a)\,.b)\,(f\,(\uparrow_0 a)\,(\uparrow_0 b)) & : & \varepsilon_1\,(q\,(\uparrow_0\,(\pi_0\,(x : a)\,.b))) \quad \checkmark \\ f\,(\uparrow_0 a)\,(\uparrow_0 b) & : & \varepsilon_1\,(p\,(\uparrow_0\,(\pi_0\,(x : a)\,.b))) \end{array}$$

Properties

- Terms *must* have a unique representation

Theorem (Canonicity)

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

- Essential for completeness

Theorem (Completeness)

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

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."

One more thing...

- Impredicative `Prop`

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

$$\frac{A : \texttt{Type}_i \qquad x : A \vdash B : \texttt{Prop}}{\Pi\,(x : A)\,.B : \texttt{Prop}}$$

One more thing...

- Impredicative $\texttt{Prop}$

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

$$\frac{A : \texttt{Type}_i \qquad x : A \vdash B : \texttt{Prop}}{\Pi\,(x : A)\,.B : \texttt{Prop}}$$

- Tarski style

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

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

Prop ambiguity 1

- Ambiguity in the level of the argument type

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

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

Prop ambiguity 2

- Ambiguity in the level of the product

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

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

Prop equalities

- Add equations

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

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

Uniform equality

- $s_1 \to s_2$ rule of the PTS

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

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

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

- Single equality

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

Uniform equality

- $s_1 \to s_2$ rule of the PTS

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

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

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

- Single equality

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

Theorem (Completeness)

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

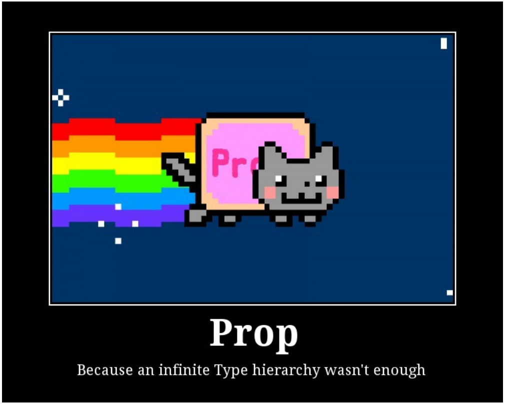
Conclusion

- Explicit subtyping using Tarski style
- Reflecting equality $\Longleftarrow$ completeness

Conclusion

- Explicit subtyping using Tarski style
- Reflecting equality $\Longleftarrow$ completeness

- `Prop` is as annoying as ever



Reduction rules

- Operational semantics based on reductions

$$M \longrightarrow_\beta N$$

- Transform equations into rewrite rules

$$\begin{aligned}
\varepsilon_{i+1}\left(\mathtt{type}_i\right) &\equiv \mathtt{Type}_i \\
\varepsilon_i\left(\pi_i\left(x : A\right).B\right) &\equiv \Pi\left(x : A\right).B \\
\varepsilon_{i+1}\left(\uparrow_i A\right) &\equiv \varepsilon_i\left(A\right)
\end{aligned}$$

Reduction rules

- Operational semantics based on reductions

$$M \longrightarrow_{\beta} N$$

- Transform equations into rewrite rules

$$\begin{array}{rcl} \varepsilon_{i+1}\left(\mathtt{type}_i\right) & \longrightarrow & \mathtt{Type}_i \\ \varepsilon_i\left(\pi_i\left(x : A\right).B\right) & \longrightarrow & \Pi\left(x : A\right).B \\ \varepsilon_{i+1}\left(\uparrow_i A\right) & \longrightarrow & \varepsilon_i\left(A\right) \end{array}$$

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

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

With $\texttt{Prop}$

- Distributing $\uparrow_i$ breaks confluence

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

- Need to raise $\uparrow$ to the top

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

- Corresponds to minimal typing!