Dependent Types (cont.)

A rule for PTSs: type conversion

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

β-equivalent types are "equal"

Ex:

$(\lambda \alpha : *. \alpha)\, \mathbb{N} =_\beta \mathbb{N}$

$P((\lambda x : \mathbb{N}. x)\, 5) =_\beta P\, 5$

(reminder: for all the PTS rules, see Barendregt)

For CoC:

- type uniqueness (up to β)
- strong normalization
- Church Rosser
- Subject Reduction

More advanced PTSs exist, allowing more / different sorts than $*, \Box$

Ex: "infinite universes"

$5 : \mathbb{N} : *_0 : *_1 : *_2 : \cdots$

Some choices for sorts, however, lead to an inconsistent logic !!

E.g. $5 : \mathbb{N} : * : *$

leads to the Girard's Paradox

(possible project)

Encoding more types in CoC

We can use the same System F encodings for $\mathbb{B}$, $\mathbb{N}$, $+$, $\times$, 0, 1, - - - -

Ex: for $\mathbb{B}$ (booleans)

$$\mathbb{B} = \prod_{\alpha : *} \alpha \to \alpha \to \alpha$$

intros: $tt = \lambda \alpha : *.\ \lambda x, y : \alpha.\ x$

$ff = \lambda \alpha : *.\ \lambda x, y : \alpha.\ y$

elim: if b then t else e

$= b\ \tau\ t\ e$

where τ is the common type of t and e

However, we now want more!

We would like to "prove":

$$\prod_{b : \mathbb{B}} \prod_{P : \mathbb{B} \to *} P\ tt \to P\ ff \to P\ b$$

"dependent elimination"

≈ the returned type is no longer a fixed type "$\alpha : *$", but is a family of types "$P : \mathbb{B} \to *$" which "depends" on the value we are eliminating.

If $P\ b$ is a constant family, i.e.

$P\ b = \alpha$ for all $b : \mathbb{B}$

then we get the regular, non-dependent elimination:

$$\prod_{b : \mathbb{B}} \prod_{P : \mathbb{B} \to *} P\ tt \to P\ ff \to P\ b$$

$$\mathbb{B} \to \left(\prod_{\alpha : *} \alpha \to \alpha \to \alpha \right)$$

In CC we can NOT prove the dependent elimination principle!

More expressive type theories are needed. E.g. the Calculus of Inductive Constructions (Coq)

We now see some examples in Coq.