

overview

logical verification lecture 12
2011 05 17
prop2 and lambda2

logic		lambda	Coq
prop1 intuitionistic classical	~	λ^{\rightarrow}	ind pred
pred1 prop2	~	λ^P λ^2	ind data prog ext

polymorphic identity in λ^2

in the polymorphic λ -calculus, or λ^2 , or system F :
 $\lambda A : * . \lambda x : A . x : \Pi A : * . A \rightarrow A$

instantiation by application:

$$\frac{\lambda a : * . \lambda x : a . x : \Pi a : * . a \rightarrow a \quad \text{nat} : *}{(\lambda a : * . \lambda x : a . x) \text{ nat} : \text{nat} \rightarrow \text{nat}}$$

then beta-reduction:

$$(\lambda a : * . \lambda x : a . x) \text{ nat} \rightarrow_{\beta} \lambda x : \text{nat} . x$$

polymorphic lambda-calculus: intuition

- abstraction over variables in $*$ (Coq: in Set or Prop)
- beta-reduction as before:
 $(\lambda x : A . M) N \rightarrow_{\beta} M[x := N]$

typing system for $\lambda 2$

approach: as for λP

pseudo-terms

all expressions we can form according to the grammar

examples:

\square

$*$

$\square \square$

$\lambda n : \text{nat}. \lambda x : n. x$

$\lambda n : \text{nat}. n n$

typing system

selects the terms from the pseudo-terms

used to derive judgements of the form

$\Gamma \vdash M : N$

M is a term if we can derive

$\Gamma \vdash M : A$ or $\Gamma \vdash N : M$

terms

all pseudo-terms that can be typed

M is a term if we can derive

$\Gamma \vdash M : A$ or $\Gamma \vdash N : M$

typing system

for every kind of term there is a rule:

- axiom rule (for * and \square)
- variable rule
- product rule
- abstraction rule
- application rule

and two more rules:

- weakening rule
- conversion rule

three product rules

$$\frac{\lambda \rightarrow, \lambda P, \lambda 2: \Gamma \vdash A : * \quad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A . B : *}$$

$$\frac{\text{only } \lambda P: \Gamma \vdash A : * \quad \Gamma, x : A \vdash B : \square}{\Gamma \vdash \Pi x : A . B : \square}$$

$$\frac{\text{only } \lambda 2: \Gamma \vdash A : \square \quad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A . B : *}$$

typing system

only difference with λP : the product rule

we consider a few rules

three product rules: examples

in $\lambda \rightarrow, \lambda P, \lambda 2$:

$$\text{nat} : *, \text{bool} : * \vdash \Pi x : \text{nat} . \text{bool} : *$$

in λP (only):

$$\text{nat} : * \vdash \Pi x : \text{nat} . * : \square$$

in $\lambda 2$ (only):

$$\vdash \Pi a : * . \Pi x : a . a : *$$

Curry-Howard-De Bruijn isomorphism

$\text{prop1} \sim \lambda \rightarrow$
 $\text{pred1} \sim \lambda P$
 $\text{prop2} \sim \lambda 2$

proof checking is type checking

proof checking

is the proof P of the formula A using assumptions Γ correct?

corresponds to

type checking:

is the typing derivation $\Gamma \vdash P : A$ correct?

decidability issues

- type inhabitation problem (TIP)
 $\Gamma \vdash ? : A$
 decidable in $\lambda \rightarrow$, undecidable in λP and in $\lambda 2$
- type checking problem (TCP)
 $\Gamma \vdash P : A?$
 decidable in $\lambda \rightarrow$, in λP , in $\lambda 2$
- type synthesis problem (TSP) or typability problem
 $\Gamma \vdash P : ?$
 decidable in $\lambda \rightarrow$, in λP , in $\lambda 2$

minimal prop2

grammar of formulas:

a
 $A \rightarrow B$
 $\forall a. B$

example of a formula in minimal prop2:

$a \rightarrow \forall b. ((a \rightarrow b) \rightarrow b)$

introduction rules:

$$\frac{B}{A \rightarrow B} \quad \frac{B}{\forall a. B}$$

elimination rules:

$$\frac{A \rightarrow B \quad A}{B} \quad \frac{\forall a. B}{B[a := A]}$$

minimal prop2: detour

introduction rule for a connective
immediately followed by an
elimination rule for the same connective

elimination of an implication detour

(as in prop1)

$$\frac{\frac{\frac{\vdots}{B}}{A \rightarrow B} I[\rightarrow] \rightarrow \frac{\vdots}{A} E \rightarrow}{B}$$

is replaced by

$$\frac{\vdots}{B}$$

where every occurrence of the assumption A^x is replaced by the proof

$$\frac{\vdots}{A}$$

elimination of an universal quantification detour

(similar to pred1)

$$\frac{\frac{B}{\forall a. B} I\forall}{B[a := A]} E\forall \rightarrow B[a := A]$$

everywhere a is replaced by A