

Interactive Theorem Proving

Week 5

Cezary Kaliszyk

April 14, 2015



Summary

So far

Proof Assistants, HOL Light, λ_{\rightarrow} , Gentzen-style, Tactics

- Properties of λ_{\rightarrow}
- BHK interpretation and λ -cube again
- Dependent types
- λ_P

Today

- Properties of λ_P
- Curry Howard for λ_P
- HOL Light more advanced tactics

Intuition behind λ_P

functions from A to B

$$A \rightarrow B$$

dependent functions from A to B

$$\Pi x : A. B$$

- Also called: dependent product
- Type of B can now depend on the argument x
- arrow type becomes a special case of dependent product

Three kinds of judgements

Kind formation judgements

$$\Gamma \vdash k : \square$$

Kinding judgements

$$\Gamma \vdash \varphi : k$$

Typing judgements

$$\Gamma \vdash M : \tau$$

The meaning of $k : \square$ is that k is a well-formed kind.

Syntax of λ_P

- variables

x, y, z, \dots

- abstraction

$\lambda x : M.N$

- function application

MN

- dependent product

$\Pi x : M.N$ (sometimes $\forall x : M.N$)

- two sorts

\star, \square

Abbreviations

If x is not free in k

We write $\tau \Rightarrow k$ instead of $(\Pi x : \tau)k$.

If x is not free in σ

We write $\tau \rightarrow \sigma$ instead of $(\forall x : \tau)\sigma$.

β -reduction in λ_P

- Like in λ_{\rightarrow}

$$(\lambda x : \tau. M)N \rightarrow_{\beta} M[x := N]$$

- Under lambda and application
- In the type of a λ -expression or Π -expression
- In a type application or under a Π

Rules of λ_P (1/3)

Axiom rule

$$\overline{\vdash \star : \square}$$

Variable rule (condition: $x \notin \Gamma$)

$$\frac{\Gamma \vdash A : \{\star, \square\}}{\Gamma, x : A \vdash x : A}$$

Weakening rule (condition: $x \notin \Gamma$)

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash C : \{\star, \square\}}{\Gamma, x : C \vdash A : B}$$

Rules of λ_P (2/3)

Dependent product rule (same \star or \square)

$$\frac{\Gamma \vdash A : \star \quad \Gamma, x : A \vdash B : \{\star, \square\}}{\Gamma \vdash \Pi x : A. B : \{\star, \square\}}$$

Abstraction rule

$$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x : A. B : \{\star, \square\}}{\Gamma \vdash \lambda x : A. M : \Pi x : A. B}$$

Application Rule

$$\frac{\Gamma \vdash M : \Pi x : A. B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

Rules of λ_P (3/3)

Conversion Rule

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : \{\star, \square\}}{\Gamma \vdash A : B'}$$

where $B =_{\beta} B'$

Examples

-

$$X : *, x : X \vdash x : X$$

-

$$X : * \vdash (X \rightarrow X) : *$$

-

$$A : *, P : A \rightarrow *, a : A \vdash (P a) \rightarrow * : \square$$

-

$$\begin{array}{l} \alpha : 0 \rightarrow *, \beta : 0 \rightarrow * \quad \vdash \\ \vdash \lambda y : (\forall x : 0. \alpha x \rightarrow \beta x). \lambda z : (\forall x : 0. \alpha x). \lambda x : 0. yx(zx) \quad : \\ : (\forall x : 0. \alpha x \rightarrow \beta x) \rightarrow (\forall x : 0. \alpha x) \rightarrow \forall x : 0. \beta x \end{array}$$

-

$$\alpha : 0 \Rightarrow * \vdash (\prod y : 0)(\alpha y \Rightarrow *) : \square$$

Properties of λ_P

- Possible to extend by an existential quantifier
 - Disjoint union (coproduct) of types
- Strong Normalization (using forgetting map)
- Church-Rosser (corollary)
- Subject Reduction
- Type reconstruction is decidable in PTIME.
 - Type checking is undecidable!
- Type inhabitation in λ_P is undecidable
 - ?

Properties of λ_P

- Possible to extend by an existential quantifier
 - Disjoint union (coproduct) of types
- Strong Normalization (using forgetting map)
- Church-Rosser (corollary)
- Subject Reduction
- Type reconstruction is decidable in PTIME.
 - Type checking is undecidable!
- Type inhabitation in λ_P is undecidable
 - First order intuitionistic logic is undecidable

Correspondence with first order logic

FOL corresponds to a fairly weak fragment of λ_P

- Only one type variable 0 (constant, type of individuals)
- All kinds are of the form $0 \Rightarrow \dots \Rightarrow 0 \Rightarrow \star$
- Function symbols are distinguished object variables $0 \rightarrow \dots \rightarrow 0$
- Constants are distinguished variables of type 0
- Other declarations may only be of the form $x : 0$

Proof correspondence

- Proof by generalization
 - abstraction $\lambda x : 0. M^\varphi$
- Proof by MP
 - application $M^{\forall x:0\varphi} N^0$

More: Proseminar.

Rules, Conversions, Tactics

- Rule: `thm -> thm`
 - Examples?
- Conversion: `term -> thm`
 - Examples?
- Tactic: goalstate refinement
 - Type?
- Thm_tactical: `thm_tactic -> thm_tactic`
- How to combine them?
 - THEN...
 - DEPTH_CONV
- How to transform one to other?
 - CONV_TAC, CONV_RULE

Rules, Conversions, Tactics

- Rule: `thm -> thm`
 - Examples?
- Conversion: `term -> thm`
 - Examples? `BETA_CONV`
- Tactic: goalstate refinement
 - Type?
- Thm_tactical: `thm_tactic -> thm_tactic`
- How to combine them?
 - `THEN...`
 - `DEPTH_CONV`
- How to transform one to other?
 - `CONV_TAC, CONV_RULE`

Rules, Conversions, Tactics

- Rule: `thm -> thm`
 - Examples?
- Conversion: `term -> thm`
 - Examples? `BETA_CONV`
- Tactic: goalstate refinement
 - Type? `goal -> goalstate`
- Thm_tactical: `thm_tactic -> thm_tactic`
- How to combine them?
 - `THEN...`
 - `DEPTH_CONV`
- How to transform one to other?
 - `CONV_TAC, CONV_RULE`

More advanced tactics

Tautologies

```
ITAUT '(A ==> B) ==> A ==> B';;
```

```
TAUT '(p <=> (q <=> r)) <=> ((p <=> q) <=> r)';;
```

Rewriting

- Matching
 - Slightly more general than first-order
- REWR_CONV does rewriting on top level

```
ADD_ASSOC;;
```

```
⊢  $\forall mnp. m+n+p = (m+n)+p$ 
```

```
REWR_CONV ADD_ASSOC 'x+y+z';;
```

```
⊢  $x+y+z = (x+y)+z$ 
```

- REWRITE_RULE, REWRITE_CONV, REWRITE_TAC
- Conditional rewriting: SIMP...

Summary

Today

- Properties of λ_P
- Curry Howard for λ_P
- More on HOL Light

Next time

- Polymorphism
- Set Theory Introduction