universität
innsbruck

Interactive Theorem Proving

Lecture & Exercises Week 7

Cezary Kaliszyk

Summary

Previous Lecture

- Type Checking and Type Inference
- Exercises only

Today

- Simple Type Theory
- Beta reduction and cut elimination
- Lambda-P

$\lambda_{\rightarrow}$ is weak... What else do we want?

Printf

What type does it have?

$\lambda_{\rightarrow}$ is weak... What else do we want?

Printf

What type does it have?

Bit-strings of length n

- Type of bit-strings: $bs : \mathbb{N} \rightarrow \star$
- Bit-string made of zeros: $0_{bs} : (\forall n : \mathbb{N}) bs(n)$

$\lambda_{\rightarrow}$ is weak... What else do we want?

Printf

What type does it have?

Bit-strings of length n

- Type of bit-strings: $bs : \mathbb{N} \rightarrow \star$
- Bit-string made of zeros: $0_{bs} : (\forall n : \mathbb{N}) bs(n)$

Vectors (later polymorphic)

- Type of hd?

$\lambda_{\rightarrow}$ is weak... What else do we want?

Printf

What type does it have?

Bit-strings of length n

- Type of bit-strings: $bs : \mathbb{N} \rightarrow \star$
- Bit-string made of zeros: $0_{bs} : (\forall n : \mathbb{N}) bs(n)$

Vectors (later polymorphic)

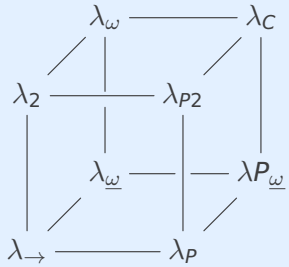
- Type of hd?

Constructive Division (proofs in the same language)

$$a/b/\!\!/P \qquad \approx \qquad \frac{a}{b \neq 0} \quad .$$

Lambda cube (Barendregt, 1991)

Four kinds of dependencies



- terms on terms (already in $\lambda_\to$)
- dependent types (λ_P)
- polymorphism (λ_2)
- terms depend on types

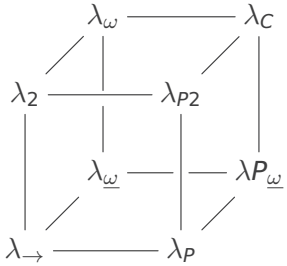
Lambda cube (again)

propositional logic $\longleftrightarrow$ $\lambda_{\rightarrow}$

predicate logic $\longleftrightarrow$ λ_P (dependent types)

2nd order propositional logic $\longleftrightarrow$ λ_2, System F (2nd order typed λ-calc)

$\lambda_{\underline{\omega}}$ (type operators)



Intuition behind λ_P

functions from A to B

$$A \to B$$

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

$$x, y, z, \ldots$$

- abstraction

$$\lambda x : M.N$$

- function application

$$MN$$

- dependent product

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

- two sorts

$$\star, \Box$$

Abbreviations

If x is not free in k

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

Note: Some researchers distinguish products on different levels [*Sorensen, Urzyczyn*]

If x is not free is σ

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

β-reduction in λ_P

- Like in $\lambda_{\to}$

$$(\lambda x : \tau . M)N \to_\beta M[x := N]$$

- 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

$$\frac{}{\vdash \star : \Box}$$

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

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

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

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : \{\star, \Box\}}{\Gamma, x : C \vdash A : B}$$

Rules of λ_P (2/3)

Dependent product rule (same $\star$ or $\Box$)

$$\frac{\Gamma \vdash A : \star \qquad \Gamma, x : A \vdash B : \{\star, \Box\}}{\Gamma \vdash \Pi x : A.B : \{\star, \Box\}}$$

Abstraction rule

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

Application Rule

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

Rules of λ_P (3/3)

Conversion Rule

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : \{\star, \Box\}}{\Gamma \vdash A : B'} \quad \text{where } B =_\beta B'$$

Product rule

With $s = \Box$ we can build $A \to * \text{ or } A \to A \to *$.

With $s = *$ we can build $A \to A \text{ or } \Pi x : A.Px \to Px$.

Examples

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

Examples

- $$X : *, x : X \vdash x : X$$
- $$X : * \vdash (X \to X) : *$$

Examples

- $$X : *,\ x : X \vdash x : X$$
- $$X : * \vdash (X \to X) : *$$
- $$A : *,\ P : A \to *,\ a : A \vdash (P\,a) \to * : \Box$$

Examples

- $$X : *,\ x : X \vdash x : X$$
- $$X : * \vdash (X \to X) : *$$
- $$A : *,\ P : A \to *,\ a : A \vdash (P\,a) \to * : \Box$$
- $$\begin{array}{cc} 0 : \star, \alpha : 0 \to \star, \beta : 0 \to \star & \vdash \\ \vdash \quad \lambda y : (\forall x : 0.\alpha x \to \beta x).\lambda z : (\forall x : 0.\alpha x).\lambda x : 0.yx(zx) & : \\ : \qquad (\forall x : 0.\alpha x \to \beta x) \to (\forall x : 0.\alpha x) \to \forall x : 0.\beta x & \end{array}$$

Examples

- $$X : *,\ x : X \vdash x : X$$
- $$X : * \vdash (X \to X) : *$$
- $$A : *,\ P : A \to *,\ a : A \vdash (P\,a) \to * : \Box$$
- $$\begin{array}{cc} 0 : \star, \alpha : 0 \to \star, \beta : 0 \to \star & \vdash \\ \vdash \quad \lambda y : (\forall x : 0.\alpha x \to \beta x).\lambda z : (\forall x : 0.\alpha x).\lambda x : 0.yx(zx) & : \\ : \qquad (\forall x : 0.\alpha x \to \beta x) \to (\forall x : 0.\alpha x) \to \forall x : 0.\beta x & \end{array}$$

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
- Church style type reconstruction is decidable in PTIME.
  - Type checking / reconstruction is undecidable in Curry style [*Dowek'93*]
- 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
- Church style type reconstruction is decidable in PTIME.
  - Type checking / reconstruction is undecidable in Curry style [*Dowek'93*]
- 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 $0 \Rightarrow \ldots \Rightarrow 0 \Rightarrow \star$
- Function symbols are distinguished object variables $0 \to \ldots \to 0$
- 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 $M^{\forall x:0\varphi}N^{0}$

Exercises (1/2)

- Give λ_P derivations whose types are the conclusions of (1 per student):
  1. $(\forall y.Qy \to Py) \to Qa \to Pa$
  2. $(\forall x.\forall y.Px \to Py) \to Pa \to \forall x.Px$
  3. $(\forall x.Px \to A) \to (A \to \forall x.\forall y.Qyx) \to Pa \to Qbb$