

Interactive Theorem Proving

Week 4

Cezary Kaliszyk (VO)
Vincent van Oostrom (PS)

Oct 28, 2016



Summary

So far

Proof Assistants, HOL Light, λ_{\rightarrow}

- Gentzen Style Natural Deduction
- Intuitionistic Logic
- Properties of λ_{\rightarrow}
- BHK interpretation

Today

- Principal types
- λ -cube
- Dependent types

Definitions

Definition: type substitution

A map from type variables to types

Definition: unifier

For two given types σ and τ their unifier, is such a type substitution s that $s(\sigma) = s(\tau)$

Definition: mgu (most general unifier)

Given types σ and τ their mgu is a type substitution s , such that:

- $s(\sigma) = s(\tau)$
- $\forall t. t(\sigma) = t(\tau) \rightarrow \exists r. t = r \circ s$

Definitions

Definition: type substitution

A map from type variables to types

Definition: unifier

For two given types σ and τ their unifier, is such a type substitution s that $s(\sigma) = s(\tau)$

Definition: mgu (most general unifier)

Given types σ and τ their mgu is a type substitution s , such that:

- $s(\sigma) = s(\tau)$
- $\forall t. t(\sigma) = t(\tau) \rightarrow \exists r. t = r \circ s$

The above notions generalize to lists of types

Algorithm computing mgu

In λ_{\rightarrow} only one function

Input: $\sigma_1, \dots, \sigma_n$, output: mgu or “not unifiable”.

$$\frac{E_1; g(\tau_1, \dots, \tau_n) \approx g(\tau'_1, \dots, \tau'_n); E_2}{E_1; \tau_1 \approx \tau'_1; \dots; \tau_n \approx \tau'_n; E_2} d_1$$

$$\frac{E_1; \tau_1 \rightarrow \tau_2 \approx \tau'_1 \rightarrow \tau'_2; E_2}{E_1; \tau_1 \approx \tau'_1; \tau_2 \approx \tau'_2; E_2} d_2$$

$$\frac{E_1; \alpha \approx \tau; E_2 \quad \alpha \notin V(\tau)}{(E_1; E_2)\{\alpha/\tau\}} v_1$$

$$\frac{E_1; \tau \approx \alpha; E_2 \quad \alpha \notin V(\tau)}{(E_1; E_2)\{\alpha/\tau\}} v_2$$

$$\frac{E_1; \tau \approx \tau; E_2}{E_1; E_2} t$$

Definition: Principal type

σ is a principal type for an untyped λ -term M if:

- $M : \sigma$ in STT à la Curry
- $\forall \tau, M : \tau \rightarrow \exists s. \tau = s(\sigma)$

Principal Types: example

$$\lambda x^\alpha . \lambda y^\beta . y^\beta (\lambda z^\gamma . y^\beta x^\alpha)$$

1. Assign type variables to all variables: $x : \alpha, y : \beta, z : \gamma$.
2. Assign type variables to all applicative subterms: $y x : \delta$, $y(\lambda z . y x) : \epsilon$.
3. Generate equations between types, necessary for the term to be typable:

$$\beta = \alpha \rightarrow \delta \quad \beta = (\gamma \rightarrow \delta) \rightarrow \epsilon$$

4. Find a most general unifier that solves the above equations:

$$\alpha := \gamma \rightarrow \delta, \beta := (\gamma \rightarrow \delta) \rightarrow \epsilon, \delta := \epsilon$$

5. The principal type of $\lambda x . \lambda y . y(\lambda z . xy)$ is now:

$$(\gamma \rightarrow \epsilon) \rightarrow ((\gamma \rightarrow \epsilon) \rightarrow \epsilon) \rightarrow \epsilon$$

Typical questions in Type Theory

TCP (type checking problem)

$M : \sigma?$

TSP (type synthesis problem) (variant of well-typedness)

$M : ?$

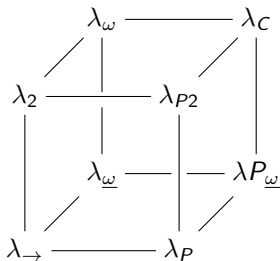
TIP (type inhabitation problem) (variants: closed term, \emptyset)

$? : \sigma$

- For λ_{\rightarrow} all are decidable
 - both in Curry and in Church style
- TCP and TSP are usually equivalent
 - application typing rule is to blame
- TCP and TSP quickly become undecidable in Curry style
 - TIP corresponds to provability in some logic

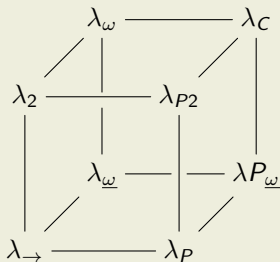
Lambda cube (Barendregt, 1991)

propositional logic	\longleftrightarrow	λ_{\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)



Lambda cube (again)

Four kinds of dependencies



- terms on terms (already in λ_{\rightarrow})
- dependent types (λ_P)
- polymorphism (λ_2)
- terms depend on types
- Combining the three is hard! (Girard's paradox)

Dependent types vs Polymorphism vs CoC

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: $0_{bs} : (\forall n : \mathbb{N})bs(n)$
- $\mathbb{R}^{\mathbb{N}}$

Vectors (later polymorphic)

- Type of hd ?

Constructive Division

$$a/b // P$$

$$\approx$$

$$\frac{a}{b \neq 0}$$

$$.$$

Today

- Principal types
- λ -cube
- Dependent types

Next time

- λ_P
- λ_{ω} , polymorphism
- More advanced features in HOL-Light