Computational Logic

Interactive Theorem Proving

Week 3

Vincent van Oostrom (VO)

March 21, 2019

Summary

So far

Proof Assistants, $\lambda_{\rightarrow}$

- Natural Deduction
- Intuitionistic Logic
- Properties of $\lambda_{\rightarrow}$
- BHK interpretation

Today

- Church-typing vs. Curry Typing
- Principal types
- Tautology Checking
- Kripke Semantics

Inductive definition of terms

Rule form

$$\frac{\cdot}{x^\sigma : \sigma} \qquad \frac{M : \sigma \to \tau \quad N : \sigma}{M\,N : \tau} \qquad \frac{P : \tau}{\lambda x^\sigma . P : \sigma \to \tau}$$

With a context

- Declare the free variables

$$x_1 : \sigma_1 \ \ldots, x_n : \sigma_n \vdash t : \tau$$

- Usually denoted Γ
- Derivation tree

Terms in STT ($\lambda_{\to}$)

- Can we find a term for every type?

Terms in STT ($\lambda_\to$)

- Can we find a term for every type?

$$x^\alpha : \alpha$$

- Can we find a closed term for every type?

Terms in STT ($\lambda_\to$)

- Can we find a term for every type?

$$x^\alpha : \alpha$$

- Can we find a closed term for every type?

$$(\alpha \to \alpha) \to \alpha$$

- No! Not every type is inhabited.

Type assignment

Typing à la Church

- All terms have the type information in the λ-abstractions
- Unique term types can be computed from the variable types

Typing à la Curry

- Given an untyped λ-term assign types
- Types are no longer unique
- Unification gives principal types

Example: Type $\lambda x.\lambda y.x(\lambda z.y)$

Type assignment

Typing à la Church

- All terms have the type information in the λ-abstractions
- Unique term types can be computed from the variable types

Typing à la Curry

- Given an untyped λ-term assign types
- Types are no longer unique
- Unification gives principal types

Example: Type $\lambda x.\lambda y.x(\lambda z.y)$

- $((\beta \to \alpha) \to \alpha) \to \alpha \to \alpha$
- $((\beta \to \alpha) \to \gamma) \to \alpha \to \gamma$
- $((\beta \to \alpha \to \alpha) \to \alpha) \to (\alpha \to \alpha) \to \alpha$

Type assignment

Typing à la Church

- All terms have the type information in the λ-abstractions
- Unique term types can be computed from the variable types
- Useful in proving

Typing à la Curry

- Given an untyped λ-term assign types
- Types are no longer unique
- Unification gives principal types
- Useful in programming

Example: Type $\lambda x.\lambda y.x(\lambda z.y)$

- $((\beta \to \alpha) \to \alpha) \to \alpha \to \alpha$
- $((\beta \to \alpha) \to \gamma) \to \alpha \to \gamma$
- $((\beta \to \alpha \to \alpha) \to \alpha) \to (\alpha \to \alpha) \to \alpha$

Connection between STT à la Church and à la Curry

Erasure map: $|\cdot|$

$$|x^{\alpha}| = x$$
$$|MN| = |M||N|$$
$$|\lambda x^{\alpha}.M| = \lambda x.|M|$$

Theorem

If $M : \sigma$ in STT à la Church, then $|M| : \sigma$ in STT à la Curry

Theorem

If $N : \sigma$ in STT à la Curry, then $\exists M.|M| = N \wedge M : \sigma$ in STT à la Church

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 $\lambda_{\to}$ only one function

Input: $\sigma_1, \ldots, \sigma_n$, output: mgu or "not unifiable".

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

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

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

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

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

Principal Types

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:
$$\beta = \alpha \to \delta \quad \beta = (\gamma \to \delta) \to \epsilon$$
4. Find a most general unifier that solves the above equations:
$$\alpha := \gamma \to \delta, \beta := (\gamma \to \delta) \to \epsilon, \delta := \epsilon$$
5. The principal type of $\lambda x.\lambda y.y(\lambda z.xy)$ is now:
$$(\gamma \to \epsilon) \to ((\gamma \to \epsilon) \to \epsilon) \to \epsilon$$

Understanding β-reduction

Theorem

Isomorphism Detour-elimination in natural deduction for MPL is β-reduction for Church-typed simply typed λ-calculus.

Theorem

A simple type ϕ is inhabited iff ϕ is provable in MPL.

Theorem

Inhabitation is PSPACE-complete.

Inhabitation via normal forms

BLACKBOARD

Understanding IPL

Theorem (Glivenko)

ϕ tautology in classical logic iff $\vdash \neg\neg\phi$ in IPL.

Theorem (Disjunction Property of IPL)

$\vdash \phi \vee \psi$ iff $\vdash \phi$ or $\vdash \psi$ in IPL.

Example

$p \vee \neg p$ tautology in classical logic even if neither p nor $\neg p$ is.

Lattice

Implicational formulae ϕ having only propositional variable p, ordered by implication.

- In classical logic: 4-point diamond.
- In intuitionitistic logic: Rieger-Nishimura lattice.