Interactive Theorem Proving

Week 2

Cezary Kaliszyk

March 10, 2015

Summary

So far

Proof Assistants

HOL Light

- HOL Systems
- OCaml and LCF style
- Types and Terms
- Rules
- Introducing Connectives

Today

- Typed λ-calculus, STT
- Type Assignment
- Curry-Howard Isomorphism and example derivations

Different Foundations

Set Theory

- sets and membership
- semantic information
- "collections of things"
- membership is undecidable
- extensional; talk about things that exist

Type Theory

- typing judgement
- syntactic information
- what objects can be constructed
- intentional
- type checking (and sometimes inference) is decidable

Typed λ-calculus

Basis for a Proof Assistant

- Terms: Programs and Proofs
- Types: Specifications and Formulas

Brings together

- Programming
- Proving

Simple Type Theory (STT) or $\lambda_\to$

Types

- Atomic types $\alpha \quad \beta \quad \gamma \quad \ldots$,
- Function types $\alpha \to \beta$.

For example: $(\alpha \to \beta) \to \alpha \to \beta$

Terms

- Variables with explicit types: $x_1^\sigma, x_2^\sigma, \ldots$
  - Countably many for each σ
- Applications: if $M : \sigma \to \tau$ and $N : \sigma$ then $(MN) : \tau$
- Abstractions: if $P : \tau$ then $(\lambda x^\sigma . P) : \sigma \to \tau$

Examples

$$\lambda x^\sigma . \lambda y^\tau . x : \sigma \to \tau \to \sigma$$

$$\lambda x^{\alpha \to \beta \to \gamma} . \lambda y^{\alpha \to \beta} . \lambda z^\alpha . xz : \beta \to \gamma$$

Conventions

Parentheses

- Types associate to the right
- Applications associate to the left

α-convertibility

$$\lambda x^\sigma . \ldots x \ldots x \ldots \approx_\alpha \lambda y^\sigma . \ldots y \ldots y \ldots$$

Capture avoiding substitution

$$M[x := N]$$

β-reduction

$$(\lambda x^\sigma . M)N \longrightarrow_\beta M[x := N]$$

Terms in STT ($\lambda_\rightarrow$)

- 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 $\lambda x.\lambda y.x(\lambda z.y)$

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

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 $\lambda x.\lambda y.x(\lambda z.y)$

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

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

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

The three typing rules with a context

Γ treated as a set: not possible for a variable to appear twice

variable rule

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

abstraction rule

$$\frac{\Gamma, x : \sigma \vdash P : \tau}{\Gamma \vdash (\lambda x : \sigma . P) : (\sigma \to \tau)}$$

application rule

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\, N : \tau}$$

Provability

Provability in $\lambda_{\rightarrow}$

$$\Gamma \vdash_{\lambda_{\rightarrow}} M : \sigma$$

iff there exists a derivation using the rules with the conclusion $\Gamma \vdash M : \sigma$

Formulas as Types (Curry-Howard isomorphism)

A typing judgement $M : \sigma$ can be read in two ways:

M is a function with the type σ

- term is an algorithm (program)
- type is its specification

M is a proof of the proposition σ

- type is a proposition
- term is its proof

One to one correspondence between

- Terms in $\lambda_{\rightarrow}$ (typable)
- Derivations in minimal propositional logic

Example derivations in $\lambda_\rightarrow$

Blackboard

Minimal Proposition Logic

Subset of Intuitionistic Propositional Logic

Only one connective: $\rightarrow$

Definition *cut*

$$\dfrac{\dfrac{\begin{matrix}[\sigma^1] \\ \mathbb{D}_1 \\ \tau\end{matrix}}{\sigma \rightarrow \tau}1 \qquad \begin{matrix}\mathbb{D}_2 \\ \sigma\end{matrix}}{\tau}$$

Minimal Proposition Logic

Subset of Intuitionistic Propositional Logic

Only one connective: $\rightarrow$

Definition *cut*-elimination

$$
\dfrac{\dfrac{\begin{matrix}[\sigma^1] \\ \mathbb{D}_1 \\ \tau\end{matrix}}{\sigma \rightarrow \tau}1 \qquad \begin{matrix}\mathbb{D}_2 \\ \sigma\end{matrix}}{\tau}
\qquad\qquad
\begin{matrix}\mathbb{D}_2 \\ \sigma \\ \mathbb{D}_1 \\ \tau\end{matrix}
$$

Cut Elimination vs $\lambda_{\rightarrow}$

Lemma

Cut-elimination in minimal proposition logic corresponds to β-reduction in $\lambda_{\rightarrow}$.

$$\text{if} \;\; \mathbb{D}_1 \longrightarrow_{\mathit{cut}} \mathbb{D}_2 \;\; \text{then} \;\; \mathbb{D}_1 \longrightarrow_{\beta} \mathbb{D}_2$$

Summary

Today

- $\lambda_{\rightarrow}$, Curry, Church styles
- Type Assignment, Erasure map
- Inductive definition of Terms
- Curry-Howard, Example derivations
- Cut-elimination

Next time

- Untyped lambda calculus, Principal Types
- Gentzen-style natural deduction
- Type Checking Problem, Synthesis, Inhabitation
- HOL Light tactics