universität innsbruck

Interactive Theorem Proving

Lecture & Exercises Week 6

Cezary Kaliszyk

Summary

Previous Lecture

- Lambda calculus
- Type checking and inference

Today

- Beta reduction and cut elimination

Presentation Topic Assignment

- (!) Dedukti and Lambda-calculus modulo
- (!) Programming with dependent types: (Epigram / Agda2 / Idris)
- ACL2 (and Nqthm)
- Lean and its type theory
- HOL Zero and checking formal proofs
- Metamath
- SMTCoq and decision procedures
- Interaction between proof assistants (Flyspeck...,)
- Project Cristal and verified compiler
- seL4 operating system
- your idea: Formalization / System / Extension ...

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

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

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

Terms in STT ($\lambda_\to$)

- Can we find a term for every type?

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

- Can we find a term for every type?

$$x^{\alpha} : \alpha$$

- Can we find a closed term for every type?

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

- Can we find a term for every type?

$$x^{\alpha} : \alpha$$

- Can we find a closed term for every type?

$$(\alpha \rightarrow \alpha) \rightarrow \alpha$$

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

- $((\beta \to \alpha) \to \alpha) \to \alpha \to \alpha$
- $((\beta \to \alpha) \to \gamma) \to \alpha \to \gamma$
- [illegible]

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

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

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 $\lambda_{\to}$

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

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

- K and S combinators

Minimal Proposition Logic

Subset of Intuitionistic Propositional Logic

Only one connective: $\rightarrow$

Definition *cut*

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

Minimal Proposition Logic

Subset of Intuitionistic Propositional Logic

Only one connective: $\rightarrow$

Definition *cut*-elimination

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

Cut Elimination vs $\lambda_{\rightarrow}$

Lemma

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

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

Gentzen style natural deduction

assumption

$$\dfrac{\vdots}{A} \quad \rightarrow \quad [A]^{\mathtt{H}}$$

conjunction introduction

$$\dfrac{\vdots}{A \wedge B} \quad \rightarrow \quad \dfrac{\dfrac{\vdots}{A} \qquad \dfrac{\vdots}{B}}{A \wedge B} \wedge \mathrm{i}$$

Gentzen style natural deduction

conjunction elimination left

$$\begin{array}{c}\vdots\\ \hline A\end{array} \quad \rightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \hline A \wedge B\end{array}}{A}\ \wedge e_1$$

conjunction elimination right

$$\begin{array}{c}\vdots\\ \hline B\end{array} \quad \rightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \hline A \wedge B\end{array}}{B}\ \wedge e_2$$

Gentzen style natural deduction

disjunction introduction left

$$\dfrac{\vdots}{A \vee B} \quad \rightarrow \quad \dfrac{\dfrac{\vdots}{A}}{A \vee B}\,\vee\mathrm{i}_1$$

disjunction introduction right

$$\dfrac{\vdots}{A \vee B} \quad \rightarrow \quad \dfrac{\dfrac{\vdots}{B}}{A \vee B}\,\vee\mathrm{i}_2$$

Gentzen style natural deduction

disjunction elimination

$$\dfrac{\vdots}{C} \quad \color{red}{\rightarrow} \quad \dfrac{\dfrac{\vdots}{A \lor B} \quad \dfrac{\begin{matrix}[A]^{\color{green}{H1}} \\ \vdots\end{matrix}}{C} \quad \dfrac{\begin{matrix}[B]^{\color{green}{H2}} \\ \vdots\end{matrix}}{C}}{C} \lor\text{e } \color{green}{[H1,H2]}$$

implication introduction

$$\dfrac{\vdots}{A \rightarrow B} \quad \color{red}{\rightarrow} \quad \dfrac{\dfrac{\begin{matrix}[A]^{\color{green}{H}} \\ \vdots\end{matrix}}{B}}{A \rightarrow B} \rightarrow\text{i } \color{green}{[H]}$$

Gentzen style natural deduction

implication elimination

$$\dfrac{\vdots}{B} \quad \color{red}{\rightarrow} \quad \dfrac{\dfrac{\vdots}{A \rightarrow B} \qquad \dfrac{\vdots}{A}}{B} \rightarrow\text{e}$$

negation introduction

$$\dfrac{\vdots}{\neg A} \quad \color{red}{\rightarrow} \quad \dfrac{\begin{matrix}[A]^{\color{green}{\text{H}}} \\ \vdots \\ \hline \bot\end{matrix}}{\neg A} \neg\text{i}\ \color{green}{[\text{H}]}$$

Gentzen style natural deduction

negation elimination

$$\begin{array}{c}\vdots\\ \bot\end{array} \quad \rightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \neg A\end{array} \qquad \begin{array}{c}\vdots\\ A\end{array}}{\bot}\,\neg\mathrm{e}$$

bottom elimination

$$\begin{array}{c}\vdots\\ A\end{array} \quad \rightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \bot\end{array}}{A}\,\bot\mathrm{e}$$

Gentzen style natural deduction

universal introduction

$$\begin{array}{c} \vdots \\ \hline \forall x\, A \end{array} \quad \rightarrow \quad \begin{array}{c} \vdots \\ \hline A[y/x] \\ \hline \forall x\, A \end{array} \;\forall\text{i}$$

universal elimination

$$\begin{array}{c} \vdots \\ \hline A[t/x] \end{array} \quad \rightarrow \quad \begin{array}{c} \vdots \\ \hline \forall x\, A \\ \hline A[t/x] \end{array} \;\forall\text{e}$$

Gentzen style natural deduction

existential introduction

$$\dfrac{\vdots}{\exists x\, A} \quad \rightarrow \quad \dfrac{\dfrac{\vdots}{A[t/x]}}{\exists x\, A}\ \exists\text{i}$$

existential elimination

$$\dfrac{\vdots}{B} \quad \rightarrow \quad \dfrac{\dfrac{\vdots}{\exists x\, A} \qquad \dfrac{\begin{matrix}[A[y/x]]^{\text{H}} \\ \vdots\end{matrix}}{B}}{B}\ \exists\text{e}\ [\text{H}]$$

Example Derivation

$$
\dfrac{\dfrac{[\exists x(P(x) \lor \neg Q(a))]^{H1} \quad \dfrac{[P(b) \lor \neg Q(a)]^{H3} \quad \dfrac{[P(b)]^{H4}}{\exists x\, P(x)}\,\exists i \quad \dfrac{\dfrac{[\neg Q(a)]^{H5} \quad [Q(a)]^{H2}}{\bot}\,\neg e}{\exists x\, P(x)}\,\bot e}{\exists x\, P(x)}\,\lor e\ [H4,H5]}{\dfrac{\exists x\, P(x)}{Q(a) \to \exists x\, P(x)}\,\to i\ [H2]}\,\exists e\ [H3]}{\exists x(P(x) \lor \neg Q(a)) \to Q(a) \to \exists x\, P(x)}\,\to i\ [H1]
$$

Corresponding Box-style Proof

1		$\exists x(P(x) \lor \neg Q(a))$	assumption
2		$Q(a)$	assumption
3	b	$P(b) \lor \neg Q(a)$	assumption
4		$P(b)$	assumption
5		$\exists x\, P(x)$	$\exists$i 4
6		$\neg Q(a)$	assumption
7		$\bot$	$\neg$e 6,2
8		$\exists x\, P(x)$	$\bot$e 7
9		$\exists x\, P(x)$	$\lor$e 3,4—5,6—8
10		$\exists x\, P(x)$	$\exists$e 1,3—9
11		$Q(a) \to \exists x\, P(x)$	$\to$i 2—10
12		$\exists x(P(x) \lor \neg Q(a)) \to Q(a) \to \exists x\, P(x)$	$\to$i 1—11

Properties of $\lambda_\to$

- Uniqueness of Types

$$\text{If } \Gamma \vdash M : \sigma \text{ and } \Gamma \vdash M : \tau \text{ , then } \sigma = \tau.$$

- Subject Reduction

$$\text{If } \Gamma \vdash M : \sigma \text{ and } M \to_{\beta\eta} N \text{ , then } \Gamma \vdash N : \sigma.$$

- Substitution Property

$$\text{If } \Gamma, x : \tau, \Delta \vdash M : \sigma, \Gamma \vdash P : \tau, \text{ then } \Gamma, \Delta \vdash M[x := P] : \sigma.$$

- Thinning

$$\text{If } \Gamma \vdash M : \sigma \text{ and } \Gamma \subset \Delta, \text{ then } \Delta \vdash M : \sigma.$$

- Strengthening

$$\text{If } \Gamma, x : \tau \vdash M : \sigma \text{ and } x \notin FV(M), \text{ then } \Gamma \vdash M : \sigma.$$

- Strong Normalization

$$\text{If } \Gamma \vdash M : \sigma, \text{ then all } \beta\eta\text{-reductions from } M \text{ terminate.}$$

Consequences

- Subterm property
- Condensing: $\Gamma|_{FV(M)}$
- Permutation
- No self application
- β-normal forms
- Some terms do not have fixed points

Intuitionistic Logic

Drawbacks of classical logic

- There are $x \notin \mathbb{Q}$ and $y \notin \mathbb{Q}$ st. $x^y \in \mathbb{Q}$.
  - Proof: by cases $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$
- There are seven 7s in a row in the decimal representation of π.

Brouwer, beginning of 20th century

Intuitionistic logic developed later around 1930

- $A \to \neg\neg A$ has an intuitionistic interpretation
- but $\neg\neg A \to A$ does not

Easier correspondence to $\lambda_?$-calculi

Constructive proofs have computational content

Brouwer-Heyting-Kolmogorov interpretation

Proof of $A \to B$

Function that maps proofs of A to proofs B

Proof of $A \wedge B$

Pair of proofs of A and B

Proof of $A \vee B$

Either a proof of A or a proof of B

Proof of $\forall x.P(x)$

Function that maps an object x to a proof of $P(x)$

Proof of $\bot$

Does not exist.Negation of A turns a proof of A into a nonexistant object

Exercises

- A more natural way of working with proofs, is to start with a goal and simplify it in order to prove it.
- Propose a mechanism that works with the HOL inference rules, that allows working backwards
- How would you add a type inhabitation procedure to your minimal type checker / reducer? (No implementation)