Proofs Are Programs

COS 441
Princeton University
Fall 2004

Logic is Computation

Want to end the course with an interesting historical perspective about the essence of programming and proving

This course has been a hopefully interesting combination of proving and programming

Today we're going to learn how they really are the same thing!

Outline

- Gentzen's Natural Deduction
- Church's lambda calculus
- Connection between the two
- Extending the connection for mobile code on the internet

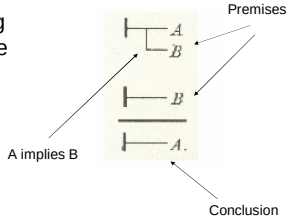
Brief History of Logic

- Aristotle (384 BCE – 322 BCE)
  - Organon (10 works on logic)
- William of Ockham (1285-1349)
  - Summa Logicae (1327) Published 1487
- Gottolob Frege's 1848-1925
  - Begriffsschrift (1879) "Concept Script"

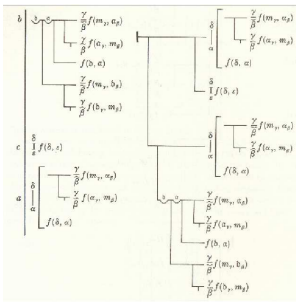
Frege's *modus ponens*

Frege introduce a pictorial formalizing for logical inference modus ponens

From B implies A and B
conclude A



Extract from Frege's *Begriffsschrift*



Systems Derived From Ferge's Work

$$\frac{\vdash B \to A \qquad \vdash B}{\vdash A}$$

$$\frac{}{\vdash A \to (B \to A)} \qquad \frac{}{\vdash A \to A}$$

$$\frac{}{\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))}$$

Gentzen's Natural Deduction

Gerhard Gentzen (1909 – 1945)

> Ich wollte zunächst einmal einen Formalismus aufstellen, der dem wirklichen Schließen möglichst nahe kommt. So ergab sich ein „Kalkül des natürlichen Schließens". (*First I wished to construct a formalism that comes as close as possible to actual reasoning. Thus arose a "calculus of natural deduction".*)

Gentzen, *Untersuchungen über das logische Schließen* (Mathematische Zeitschrift 39, pp.176-210, 1935)

Natural Deduction

Reaction to "sentential" axiomizations used by Hilbert, Frege, and Rusell

Also proposed by Stanislaw Jaskowski

Many different contributors but of course one person tends to get all the credit!

Fragment of Natural Deduction Rules

$$\frac{}{A \vdash A}\,\mathrm{Id}$$

$$\frac{\Gamma,\ B \vdash A}{\Gamma \vdash B \to A}\,{\to}\text{-I} \qquad \frac{\Gamma \vdash B \to A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A}\,{\to}\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A \wedge B}\,{\wedge}\text{-I} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\,{\wedge}\text{-E}_1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\,{\wedge}\text{-E}_2$$

Generalize to Include Contexts

- Assuming B_1,..., B_n conclude A

$$B_1,\ \ldots,\ B_n \vdash A$$

- Γ and Δ stand for lists of propositions A and B single propositions
- Γ, Δ is the union of propositions removing any duplicates

Types of Natural Deduction Rules

structural rules

$$\frac{}{A \vdash A}\,\mathrm{Id}$$

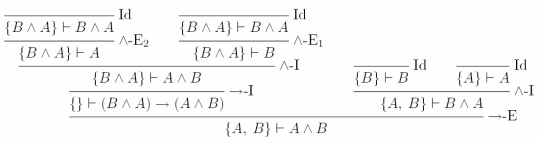
$$\frac{\Gamma,\ B \vdash A}{\Gamma \vdash B \to A}\,{\to}\text{-I} \qquad \frac{\Gamma \vdash B \to A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A}\,{\to}\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma,\ \Delta \vdash A \wedge B}\,{\wedge}\text{-I} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\,{\wedge}\text{-E}_1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\,{\wedge}\text{-E}_2$$

introduction rules | elimination rules

logical rules

A Roundabout Proof



Leads to the notion of proof simplification

Subforumula property means any proof of $\Gamma \vdash A$ can be reduced to a proof that only formulas in Γ and A or subformulas of Γ and A

Sequent Calculus

- Gentezen introduced to logics natural deduction and sequent calculus
- Sequent calculus is simpler form where proving subformula property is easier
- Gentezen later showed natural deduction and sequent calculus are equivalent
- Sequent calculus is a form of “logical assembly code” when compare to natural deduction

Direct Proof of Subformula Property

A direct proof of the subformula property can be derived form ideas presented by Church and his formulation of the lambda calculus

Church and the Lambda-Calculus

Alonzo Church (1903-1995)

B.S. (1924) and PhD (1927) From Princeton University

Lambda calculus introduce in 1932 as a reformulation of logic

Original formulation was buggy! Allowed for paradoxes ((λx. x) (λx.x))

Seen as a foundation for computation in 1936

Refresher Course in λ

- Everything reduced to substitution
- Mathematical function $f(x) = x \times x$

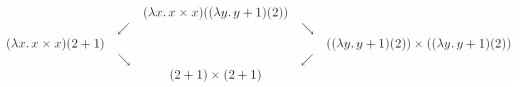
  $f(3) = 3 \times 3 = 9$
- Represented with lambda term

  $\lambda x.\ x \times x$
- Plus basic reduction rule

  $(\lambda x.t)(u) \Rightarrow [u/x]t$

Church-Rosser Theorem

Order of reduction of lambda term does not matter



Untyped Lambda Calculus

Can directly encode multi-argument functions via “currying”

Can directly encode the natural numbers as lambda terms

Can encode pairs and many structure in pure lambda calculus

Can encode any computable function in the ***untyped*** lambda calculus

Typed Lambda Calculus

- Introduce (circa 1940) by Church to avoid paradoxes in original lambda logic as well as Ferge’s and Rusell’s system
- The following slide should look vaguely familiar!

Rules for the TLC

$$\frac{}{x : A \vdash x : A}\ \text{Id}$$

$$\frac{\Gamma,\ x : B \vdash t : A}{\Gamma \vdash \lambda x.\, t : B \to A}\ \to\text{-I} \qquad \frac{\Gamma \vdash t : B \to A \qquad \Delta \vdash u : B}{\Gamma,\ \Delta \vdash t(u) : A}\ \to\text{-E}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma,\ \Delta \vdash \langle t,\, u\rangle : A \wedge B}\ \wedge\text{-I} \qquad \frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash t.fst : A}\ \wedge\text{-E}_1 \qquad \frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash t.snd : B}\ \wedge\text{-E}_2$$

Reductions Preserve Type

$$\frac{\dfrac{\Gamma \cup \{x : B\} \vdash t : A}{\Gamma \vdash \lambda x.\, t : B \to A}\ \to\text{-I} \qquad \Delta \vdash u : B}{\Gamma \cup \Delta \vdash (\lambda x.\, t)(u) : A}\ \to\text{-E} \quad \Rightarrow \quad \Gamma \cup \Delta \vdash t[u/x] : A$$

$$\frac{\dfrac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma \cup \Delta \vdash \langle t,\, u\rangle : A \wedge B}\ \wedge\text{-I}}{\Gamma \cup \Delta \vdash \langle t,\, u\rangle.fst : A}\ \wedge\text{-E}_1 \quad \Rightarrow \quad \Gamma \vdash t : A$$

$$\frac{\dfrac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma \cup \Delta \vdash \langle t,\, u\rangle : A \wedge B}\ \wedge\text{-I}}{\Gamma \cup \Delta \vdash \langle t,\, u\rangle.snd : B}\ \wedge\text{-E}_2 \quad \Rightarrow \quad \Delta \vdash u : B$$

Strong Normalization

- Unlike the untyped lambda calculus the type lambda calculus does not allow you to express a term with an infinite sequence of reductions
- Types get simpler after each reductions, types are finite therefore you have to stop
- TLC is not Turing complete (this is a feature)

The Curry-Howard Isomorphism

Take the TLC erase the “red” terms and you get Gentzen’s natural deduction!

Lambda terms are one-to-one with proof rules

Types are one-to-one with logical formula

Term reduction is the same as proof simplification

Type-checking is proof checking!

The Long Road to Discovery

1934 – Gentzen's simplification via sequents

1940 – Church's TLC

1956 – Prawitz direct simplification of ND

?? – Curry and Feys work on combinators draw connection with Hilbert's axioms

1969 – W.A. Howard connects the dots of Curry and Prawitz

1980 – Officially published!

Logics and Computer Science

Hindley-Milner (type inference)

- Hindley - logician discovered 1969
- Milner – computer scientist re-discovered 1978

Girard-Reynolds (2nd order polymorphic lambda calculus)

- Girard – logician 1972
- Reynolds – computer scientist 1974

Intuitionist Logic

CHI based on intuitionist fragments of logic

Intuitionist logic does not include the law of the excluded middle

$$(\neg A) \vee A$$

Timothy Griffin (1990) extends CHI to classical logic

Roughly requires CPS conversion

Programming Languages and Logic

- Great deal of effort to establish formally verified properties of software
- Theorem Proving – HOL, LCF, Isabelle, Twelf, Coq,…
- Proof Carrying Code
- Typed Assembly Language

Challenges for the Future

- Digital Rights Management
  - Logics will be use to enforce contracts and protect rights of content providers (XRML)
- Data Privacy (information flow)
  - Design new languages that don't leak information
- Verification of software systems
  - Systems that don't crash

Summary

- There are deep connections between logical reasoning and programming
  - Programs are proofs
  - Types are formulas
- Understanding the foundations of both are the key to moving forward in the next century