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
• Gottlob 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*
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 Russell
Also proposed by Stanislaw Jaskowski
Many different contributors but of course
one person tends to get all the credit!

Types of Natural Deduction Rules

- **Structural Rules**
  - **Id**
    - $A \vdash A$

- **Logical Rules**
  - **Elimination Rules**
    - $\Gamma \vdash A \land B \vdash A \land -E_1$
    - $\Gamma \vdash B \vdash A \land -E_2$
  - **Introduction Rules**
    - $\Gamma \vdash A \Delta \vdash B \vdash A \land B \land -I$

- **Generalize to Include Contexts**
  - Assuming $B_1,..., B_n$ conclude $A$
    - $B_1, ..., B_n \vdash A$
  - $\Gamma$ and $\Delta$ stand for lists of propositions $A$ and $B$ single propositions
  - $\Gamma, \Delta$ is the union of propositions removing any duplicates

- **Fragment of Natural Deduction Rules**
  - $\Gamma, B \vdash A$
    - $\Gamma \vdash B \rightarrow A \rightarrow I$
  - $\Gamma \vdash B \rightarrow A \vdash B \rightarrow A \rightarrow E$
  - $\Gamma \vdash A \Delta \vdash B \vdash A \land B \land -I$
  - $\Gamma \vdash A \Delta \vdash B \vdash A \land B \land -E_1$
  - $\Gamma \vdash A \Delta \vdash B \vdash A \land B \land -E_2$
A Roundabout Proof

Leads to the notion of proof simplification
Subformula property means any proof of \( \Gamma \vdash A \)
can be reduced to a proof that only formulas in \( \Gamma \) and \( A \) or subformulas of \( \Gamma \) 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 ((\(\lambda x. x\) (\(\lambda x. x\))))
Seen as a foundation for computation in 1936

Refresher Course in \( \lambda \)

- 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 \textit{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

Reductions Preserve Type

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
   (¬A) ∨ 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 used 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