Comp 311
Functional Programming

Eric Allen, PhD
Vice President, Engineering
Two Sigma Investments, LLC
Homework 4 More Time Needed

Frequency

Hours

0 2 4 6 8 10

0.0

0.5

1.0

1.5

2.0

2.5

3.0
Homework 4 Helpful

![Bar chart showing frequency of ratings](image-url)
Class Pace

Frequency

Rating

0 1 2 3 4 5

1 2 3 4

0 1 2 3 4 5

Class Pace

Frequency

Rating
Comments and Actions

• Lectures are easy to follow but then it is difficult to know how to apply the material to new situations
  • Worksheets? Smaller homeworks?
  • Not enough practice with types
    • Some dealt with in Homework 6
    • Add an additional homework with a type-heavy component
Mechanical Proof Checking
Syntax of Propositional Logic

\[ S ::= x \]
\[ \quad | \quad S \land S \]
\[ \quad | \quad S \lor S \]
\[ \quad | \quad S \rightarrow S \]
\[ \quad | \quad \neg S \]
Factory Methods for Construction

case object Formulas {
    def evar(name: String): Formula
    def and(left: Formula, right: Formula): Formula
    def or(left: Formula, right: Formula): Formula
    def implies(left: Formula, right: Formula): Formula
    def not(body: Formula): Formula
}

Sequents

\[ S^* \vdash S \]
Sequents

- Sequents consist of two parts:
  - The *antecedents* to the left of the turnstile
  - The *consequent* to the right of the turnstile
- Example:
  \[
  \{p, \; q, \; \neg r, \; p \rightarrow r\} \vdash \neg p
  \]
Sequents

• When the set of antecedents consists of a single formula, we often elide the enclosing braces:

\[ \{p\} \vdash p \]

• is equivalent to:

\[ p \vdash p \]
Inference Rules

\[
\frac{\Gamma \vdash p \quad \Delta \vdash q}{\Gamma \cup \Delta \vdash p \land q} \text{AND-INTRO}
\]
Inference Rules:
General Form

\[ \frac{Q^*}{Q} \]
Inference Rules

$$\frac{\Gamma \vdash p \land q}{\Gamma \vdash p} \text{ AND-ELIM-LEFT}$$
Inference Rules

\[
\frac{\Gamma \vdash p \land q}{\Gamma \vdash q} \quad \text{And-Elim-Right}
\]
Inference Rules

\[
\begin{align*}
\Gamma \vdash p \\
\Gamma \vdash p \lor q & \quad \text{Or-Intro-Left}
\end{align*}
\]
Inference Rules

\[
\frac{\Gamma \vdash p}{\Gamma \vdash q \lor p} \quad \text{OR-INTRO-RIGHT}
\]
Inference Rules

\[
\frac{\Gamma \vdash p \lor q \quad \Gamma' \cup \{p\} \vdash r \quad \Gamma'' \cup \{q\} \vdash r}{\Gamma \cup \Gamma' \cup \Gamma'' \vdash r} \quad \text{Or-Elim}
\]
Inference Rules

\[
\begin{align*}
\Gamma \cup \{p\} & \vdash q & \Gamma' \cup \{p\} & \vdash \neg q \\
\Gamma \cup \Gamma' & \vdash \neg p & \text{NEG-INTRO}
\end{align*}
\]
Inference Rules

\[\Gamma \vdash \neg\neg p\]

\[\Gamma \vdash p\quad \text{NEG-ELIM}\]
Inference Rules

\[
\frac{
\Gamma \cup \{p\} \vdash q
}{
\Gamma \vdash p \rightarrow q}
\]

\text{IMPLIES-INTRO}
Inference Rules

\[
\frac{\Gamma \vdash p \rightarrow q \quad \Gamma' \vdash p}{\Gamma \cup \Gamma' \vdash q} \quad \text{IMPLIES-ELIM}
\]
Inference Rules

\[ p \vdash p \]
Inference Rules

\[
\Gamma \cup \{p\} \vdash p \quad \text{Assumption}
\]
Inference Rules

\[
\frac{\Gamma \vdash p}{\Gamma \cup \{q\} \vdash p} \text{ Generalization}
\]
Example Proof 1

\[
\frac{p \vdash p}{\emptyset \vdash p \rightarrow p} \text{ Identity, Implies-Intro}
\]
Example Proof 2

\[
p \rightarrow q \vdash p \rightarrow q \\
\{p, p \rightarrow q\} \vdash q
\]
Example Proof 3

\[
\begin{array}{c}
p \land \neg p \vdash p \land \neg p \\
\hline
p \land \neg p \vdash p \\
\hline
\emptyset \vdash \neg (p \land \neg p)
\end{array}
\]

Identity

And-Elim-Left

\[
\begin{array}{c}
p \land \neg p \vdash p \land \neg p \\
\hline
p \land \neg p \vdash \neg p
\end{array}
\]

Identity

And-Elim-Right

Neg-Intro
case object Rules {
  def identity(p: Formula): Sequent
  def assumption(s: Sequent): Sequent
  def generalization(p: Formula)(s: Sequent): Sequent
  def andIntro(left: Sequent, right: Sequent): Sequent
  def andElimLeft(s: Sequent): Sequent
  def andElimRight(s: Sequent): Sequent
  def orIntroLeft(p: Formula)(s: Sequent): Sequent
  def orIntroRight(p: Formula)(s: Sequent): Sequent
  def orElim(s0: Sequent, s1: Sequent, s2: Sequent): Sequent
  def negIntro(p: Formula)(s0: Sequent, s1: Sequent): Sequent
  def negElim(s: Sequent): Sequent
  def impliesIntro(s: Sequent): Sequent
  def impliesElim(p: Formula)(s: Sequent): Sequent
}
The Curry-Howard Isomorphism
Simply Typed Expressions

\[
E ::= x \\
\mid 0 \mid 1 \mid 2 \ldots \\
\mid \text{true} \mid \text{false} \\
\mid (x:T) \Rightarrow E \\
\mid E(E)
\]
Simple Types

T ::= Int
    | Boolean
    | T => T
Simple Type Assertions

E : T
Simple Type Assertions

$0: \text{Int}$
Simple Type Assertions

true:Boolean
Simple Type Assertions

(x:Int) => x : Int => Int
Simple Type Assertions

x: Boolean
Assertions Within a Type Environment

\{x: \text{Boolean}\} \vdash x: \text{Boolean}
Rules for Checking the Type of an Expression

\[ n \in \text{IntLiteral} \quad \frac{}{\Gamma \vdash n: \text{Int}} \]
Rules for Checking the Type of an Expression

\[
\Gamma \vdash \text{true:} \text{Boolean} \quad \text{T-TRUE}
\]

\[
\Gamma \vdash \text{false:} \text{Boolean} \quad \text{T-FALSE}
\]
Rules for Checking the Type of an Expression

\[
\frac{\Gamma \cup \{x:S\} \vdash E:T}{\Gamma \vdash (x:S) \Rightarrow E : S \Rightarrow T} \quad T\text{-ABS}
\]
Rules for Checking the Type of an Expression

\[
\Gamma \vdash E : S \rightarrow T \quad \Gamma \vdash E' : S \\
\Gamma \vdash E(E') : T \quad \text{T-APP}
\]
Contrast with Implies-Intro
For Propositional Logic

\[ \frac{\Gamma \cup \{p\} \vdash q}{\Gamma \vdash p \rightarrow q} \text{ IMPLIES-INTRO} \]

\[ \frac{\Gamma \cup \{x:S\} \vdash E:T}{\Gamma \vdash (x:S)=\rightarrow E : S=\rightarrow T} \text{ T-ABS} \]
Contrast with Implies-Intro
For Propositional Logic

\[
\frac{\Gamma \cup \{p\} \vdash q}{\Gamma \vdash p \rightarrow q} \text{ IMPLIES-INTRO}
\]

\[
\frac{\Gamma \cup \{S\} \vdash T}{\Gamma \vdash S \rightarrow T} \text{ T-ABS}
\]
Contrast with Implies-Elim
From Propositional Logic

\[
\frac{\Gamma \vdash p \rightarrow q \quad \Gamma' \vdash p}{\Gamma \cup \Gamma' \vdash q} \quad \text{IMPLIES-ELIM}
\]

\[
\frac{\Gamma \vdash E : S \Rightarrow T \quad \Gamma \vdash E' : S}{\Gamma \vdash E(E') : T} \quad \text{T-APP}
\]
Contrast with Implies-Elim
From Propositional Logic

\[
\begin{align*}
\Gamma \vdash p \rightarrow q & \quad \Gamma' \vdash p \\
\hline
\Gamma \cup \Gamma' \vdash q & \quad \text{IMPLIES-ELIM}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \text{S} \Rightarrow \text{T} & \quad \Gamma \vdash \text{S} \\
\hline
\Gamma \vdash \text{T} & \quad \text{T-APP}
\end{align*}
\]
Types and Propositions

• We can think of the types in our simple type system as corresponding to propositions:

  • Primitive types (Boolean, Int) correspond to simple propositions \((p, q)\)

  • Arrow types correspond to logic implication:

    \[ p \rightarrow q, \ (p \rightarrow (q \rightarrow r)), \text{etc.} \]
Types and Propositions

• For each syntactic form of expression, there is exactly one form rule that contains that syntactic form as its result

• Example:

$$\Gamma \cup \{x:S\} \vdash E:T$$

$$\Gamma \vdash (x:S) => E : S=>T$$  \text{T-ABS}
Types and Propositions

• If we wish to use type rules to prove that an expression has a specific type

• We can start with the expression, and apply the rules backwards:

\[
\begin{align*}
\emptyset \vdash (x : T) \Rightarrow x : T & \Rightarrow T \\
\frac{x : T \vdash x : T}{T-\text{Identity}} \quad \frac{T-\text{Abs}}{}
\end{align*}
\]
Types and Propositions

- While working backwards with expressions, there is only one choice at each step.
- Thus a well-typed expression E entirely determines the form of the proof that E:T.
- But the proof of E:T in our type system is equivalent to a proof of T in propositional logic.
Types and Propositions

• So, E effectively encodes a proof of type T, thought of as a proposition

• Checking the type T of an expression E is equivalent to proving the validity of T
The Curry-Howard Isomorphism

- This deep correspondence between types and logical assertions is known as the *Curry-Howard Isomorphism*.

- This correspondence goes far beyond just propositional logic, extending to predicate calculus, modal logic, etc.

- This leads to the surprising result that the arrow in arrow types is really just the implication symbol from propositional logic!