Comp 311
Functional Programming

Eric Allen, Two Sigma Investments
Robert “Corky” Cartwright, Rice University
Sagnak Tasirlar, Two Sigma Investments
Mechanical Proof Checking
Syntax of Propositional Logic

\[ S ::= x \]
\[ \quad \mid S \land S \]
\[ \quad \mid S \lor S \]
\[ \quad \mid S \rightarrow S \]
\[ \quad \mid \neg S \]
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

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

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

\[ \Gamma \vdash p \land q \quad \text{AND-ELIM-LEFT} \]

\[ \Gamma \vdash p \]
Inference Rules

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

\[ \frac{\Gamma \vdash p}{\Gamma \vdash p \lor q} \text{ Or-Intro-Left} \]
Inference Rules

\[ \Gamma \vdash p \quad \frac{}{\Gamma \vdash q \lor p} \quad \text{Or-Intro-Right} \]
Inference Rules

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

\[
\frac{\Gamma \cup \{p\} \vdash q \quad \Gamma' \cup \{p\} \vdash \lnot q}{\Gamma \cup \Gamma' \vdash \lnot p} \text{ NEG-INTRO}
\]
Inference Rules

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

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

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

\[ \Gamma \vdash p \rightarrow q \quad \text{IMPLIES-INTRO} \]
Inference Rules

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

\[ p \vdash p \quad \text{Identity} \]
Inference Rules

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

\[ \Gamma \vdash p \]

\[ \Gamma \cup \{q\} \vdash p \quad \text{GENERALIZATION} \]
Example Proof 1

\[
\begin{align*}
\text{Identity} & \quad \frac{p \vdash p}{\emptyset \vdash p \rightarrow p} \quad \text{Implies-Intro}
\end{align*}
\]
Example Proof 2

\[
\begin{align*}
\frac{p \rightarrow q \vdash p \rightarrow q}{\text{Identity}} & \quad \frac{p \vdash p}{\text{Identity}} \quad \frac{}{\{p, p \rightarrow q\} \vdash q} \quad \text{Implies-Elim}
\end{align*}
\]
Example Proof 3

\[ \vdash p \land \lnot p \vdash p \land \lnot p \]

\[ \vdash p \land \lnot p \vdash p \]

\[ \vdash \emptyset \vdash \lnot (p \land \lnot p) \]

\[ \vdash p \land \lnot p \vdash p \land \lnot p \]

\[ \vdash p \land \lnot p \vdash \lnot p \]

\[ \vdash \emptyset \vdash \lnot (p \land \lnot p) \]
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

\[ T ::= \text{Int} \]
\[ \mid \text{Boolean} \]
\[ \mid T \Rightarrow T \]
Simple Type Assertions

E : T
Simple Type Assertions

0: 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

\[ n \in \text{IntLiteral} \quad \frac{}{\Gamma \vdash n:\text{Int}} \quad T-\text{INT} \]
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

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

\[
\frac{\Gamma \vdash E : S \Rightarrow T \quad \Gamma \vdash E' : S}{\Gamma \vdash E(E') : T} \quad \text{T-App}
\]
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

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

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

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

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

From Propositional Logic

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

\[
\frac{\Gamma \vdash \text{S} \Rightarrow \text{T} \quad \Gamma \vdash \text{S}}{\Gamma \vdash \text{T}} \quad \text{T-APP}
\]
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 of rule that contains that syntactic form as its result

• Example:

\[
\Gamma \cup \{x:S\} \vdash E:T \\
\Gamma \vdash (x:S) => E : S => T \quad T-\text{ABS}
\]
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:

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