# CS 4110

# Programming Languages & Logics

## Lecture 29
## Propositions as Types

# Propositions as Types

Logics = Type Systems

# Inference Rules for Logic

We have used inference rules to build up inductively defined sets of PL concepts: operational steps, valid Hoare triples, associations between terms and types, etc.

Logicians use the same kind of notation to build up the set of true logical formulas.

# Inference Rules for Logic

We have used inference rules to build up inductively defined sets of PL concepts: operational steps, valid Hoare triples, associations between terms and types, etc.

Logicians use the same kind of notation to build up the set of true logical formulas.

Here's a rule from natural deduction, a *constructive* logic invented by logician Gerhard Gentzen in 1935:

$$\frac{\phi \qquad \psi}{\phi \land \psi} \; \land\text{-INTRO}$$

Given a proof of $\phi$ and a proof of $\psi$, the rule lets you *construct* a proof of $\phi \land \psi$.

# Natural Deduction

Let’s use our usual 4110 tools to define the set of true formulas (“theorems”).

# Natural Deduction

Let's use our usual 4110 tools to define the set of true formulas ("theorems").

We'll start with a grammar for formulas:

$$\begin{array}{rcl} \phi & ::= & \top \\ & | & \bot \\ & | & X \\ & | & \phi \wedge \psi \\ & | & \phi \vee \psi \\ & | & \phi \rightarrow \psi \\ & | & \neg\phi \\ & | & \forall X.\ \phi \end{array}$$

where $X$ ranges over Boolean variables
and $\neg\phi$ is an abbreviation for $\phi \rightarrow \bot$.

# Natural Deduction

Let's define a judgment that that a formula is true given a set of assumptions Γ:

$$\Gamma \vdash \phi$$

where Γ is just a list of formulas.

# Natural Deduction

Let's define a judgment that that a formula is true given a set of assumptions $\Gamma$:

$$\Gamma \vdash \phi$$

where $\Gamma$ is just a list of formulas.

If $\vdash \phi$ (with no assumptions), we say $\phi$ is a *theorem*.

Examples:

- $\vdash A \wedge B \to A$

# Natural Deduction

Let's define a judgment that that a formula is true given a set of assumptions Γ:

$$\Gamma \vdash \phi$$

where Γ is just a list of formulas.

If $\vdash \phi$ (with no assumptions), we say $\phi$ is a *theorem*.

Examples:

- $\vdash A \land B \to A$
- $\vdash \neg(A \land B) \to \neg A \lor \neg B$

# Natural Deduction

Let's define a judgment that that a formula is true given a set of assumptions $\Gamma$:

$$\Gamma \vdash \phi$$

where $\Gamma$ is just a list of formulas.

If $\vdash \phi$ (with no assumptions), we say $\phi$ is a *theorem*.

Examples:

- $\vdash A \wedge B \rightarrow A$
- $\vdash \neg(A \wedge B) \rightarrow \neg A \vee \neg B$
- $A, B, C \vdash B$

# Natural Deduction

Let's write the rules for our judgment:

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \land \psi}\ \land\text{-INTRO}$$

# Natural Deduction

Let's write the rules for our judgment:

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi} \wedge\text{-INTRO}$$

$$\frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi} \wedge\text{-ELIM1} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi} \wedge\text{-ELIM2}$$

# Natural Deduction

Let's write the rules for our judgment:

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \land \psi} \land\text{-INTRO}$$

$$\frac{\Gamma \vdash \phi \land \psi}{\Gamma \vdash \phi} \land\text{-ELIM1} \qquad \frac{\Gamma \vdash \phi \land \psi}{\Gamma \vdash \psi} \land\text{-ELIM2}$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi} \to\text{-INTRO}$$

# Natural Deduction

Let's write the rules for our judgment:

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \land \psi} \land\text{-INTRO}$$

$$\frac{\Gamma \vdash \phi \land \psi}{\Gamma \vdash \phi} \land\text{-ELIM1} \qquad \frac{\Gamma \vdash \phi \land \psi}{\Gamma \vdash \psi} \land\text{-ELIM2}$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi} \to\text{-INTRO}$$

...and so on.

# Natural Deduction

$$\frac{}{\Gamma, \phi \vdash \phi}\ \textsc{Axiom}$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\ \to\text{-INTRO} \qquad \frac{\Gamma \vdash \phi \to \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi}\ \to\text{-ELIM}$$

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\ \wedge\text{-INTRO} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\ \wedge\text{-ELIM1} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\ \wedge\text{-ELIM2}$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi}\ \vee\text{-INTRO1} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi}\ \vee\text{-INTRO2}$$

$$\frac{\Gamma \vdash \phi \vee \psi \qquad \Gamma \vdash \phi \to \chi \qquad \Gamma \vdash \psi \to \chi}{\Gamma \vdash \chi}\ \vee\text{-ELIM}$$

$$\frac{\Gamma, P \vdash \phi}{\Gamma \vdash \forall P.\, \phi}\ \forall\text{-INTRO} \qquad \frac{\Gamma \vdash \forall P.\, \phi}{\Gamma \vdash \phi\{\psi/P\}}\ \forall\text{-ELIM}$$

# Natural Deduction

Let's try a proof! We can write a proof that $A \wedge B \to B \wedge A$ is a theorem.

# Natural Deduction

Let's try a proof! We can write a proof that $A \wedge B \rightarrow B \wedge A$ is a theorem.

$$
\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\ \textsf{AXIOM}}{A \wedge B \vdash B}\ \wedge\textsf{-ELIM2} \qquad \dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\ \textsf{AXIOM}}{A \wedge B \vdash A}\ \wedge\textsf{-ELIM1}}{\dfrac{A \wedge B \vdash B \wedge A}{\vdash A \wedge B \rightarrow B \wedge A}\ \rightarrow\textsf{-INTRO}}\ \wedge\textsf{-INTRO}
$$

# Natural Deduction

Let's try a proof! We can write a proof that $A \wedge B \rightarrow B \wedge A$ is a theorem.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{A \wedge B \vdash A \wedge B}\ \textsf{AXIOM}
    }{A \wedge B \vdash B}\ \wedge\textsf{-ELIM2}
    \qquad
    \dfrac{
      \dfrac{}{A \wedge B \vdash A \wedge B}\ \textsf{AXIOM}
    }{A \wedge B \vdash A}\ \wedge\textsf{-ELIM1}
  }{A \wedge B \vdash B \wedge A}\ \wedge\textsf{-INTRO}
}{\vdash A \wedge B \rightarrow B \wedge A}\ \rightarrow\textsf{-INTRO}
$$

Does this look familiar?

# Natural Deduction

Let's try a proof! We can write a proof that $A \land B \to B \land A$ is a theorem.

$$
\dfrac{
  \dfrac{\dfrac{}{A \land B \vdash A \land B}\ \textsf{AXIOM}}{A \land B \vdash B}\ \land\textsf{-ELIM2}
  \qquad
  \dfrac{\dfrac{}{A \land B \vdash A \land B}\ \textsf{AXIOM}}{A \land B \vdash A}\ \land\textsf{-ELIM1}
}{
  \dfrac{A \land B \vdash B \land A}{\vdash A \land B \to B \land A}\ \to\textsf{-INTRO}
}\ \land\textsf{-INTRO}
$$

Does this look familiar?

$$
\dfrac{
  \dfrac{\dfrac{}{x\!:\!A \times B \vdash x\!:\!A \times B}\ \textsf{T-VAR}}{x\!:\!A \times B \vdash \#2\ x\!:\!B}\ \textsf{T-\#1}
  \qquad
  \dfrac{\dfrac{}{x\!:\!A \times B \vdash x\!:\!A \times B}\ \textsf{T-VAR}}{x\!:\!A \times B \vdash \#1\ x\!:\!A}\ \textsf{T-\#2}
}{
  \dfrac{x\!:\!A \times B \vdash (\#2\ x, \#1\ x)\!:\!B \times A}{\vdash \lambda x.\,(\#2\ x, \#1\ x)\!:\!A \times B \to B \times A}\ \textsf{T-ABS}
}\ \textsf{T-PAIR}
$$

# Propositions as Types

Every natural deduction proof tree has a corresponding type tree in System F with product and sum types! And vice-versa!

| Type Systems | | Formal Logic | |
|---|---|---|---|
| $\tau$ | Type | $\phi$ | Formula |
| $\tau$ | is inhabited | $\phi$ | is a theorem |
| $e$ | Well-typed expression | $\pi$ | Proof |

A program with a given type acts as a *witness* that the type's corresponding formula is true.

# Propositions as Types

Every type rule in System F with product and sum types corresponds 1-1 with a proof rule in natural deduction:

| Type Systems | | Formal Logic | |
|---|---|---|---|
| $\to$ | Function | $\to$ | Implication |
| $\times$ | Product | $\wedge$ | Conjunction |
| $+$ | Sum | $\vee$ | Disjunction |
| $\forall$ | Universal | $\forall$ | Quantifier |

You can even add existential types to correspond to existential quantification. It still works!

# Propositions as Types

Every type rule in System F with product and sum types corresponds 1-1 with a proof rule in natural deduction:

| **Type Systems** | | **Formal Logic** | |
|---|---|---|---|
| $\rightarrow$ | Function | $\rightarrow$ | Implication |
| $\times$ | Product | $\wedge$ | Conjunction |
| $+$ | Sum | $\vee$ | Disjunction |
| $\forall$ | Universal | $\forall$ | Quantifier |

You can even add existential types to correspond to existential quantification. It still works!

Is this a coincidence? Natural deduction was invented by a German logician in 1935. Types for the $\lambda$-calculus were invented by Church at Princeton in 1940.

# Propositions as Types Through the Ages

| | | |
|---|---|---|
| **Natural Deduction**<br>Gentzen (1935) | $\Leftrightarrow$ | **Typed $\lambda$-Calculus**<br>Church (1940) |
| **Type Schemes**<br>Hindley (1969) | $\Leftrightarrow$ | **ML's Type System**<br>Milner (1975) |
| **System F**<br>Girard (1972) | $\Leftrightarrow$ | **Polymorphic $\lambda$-Calculus**<br>Reynolds (1974) |
| **Modal Logic**<br>Lewis (1910) | $\Leftrightarrow$ | **Monads**<br>Kleisli (1965), Moggi (1987) |
| **Classical–Intuitionistic Embedding**<br>Gödel (1933) | $\Leftrightarrow$ | **Continuation Passing Style**<br>Reynolds (1972) |

# Term Assignment

This all means that we have a new way of proving theorems: writing programs!

# Term Assignment

This all means that we have a new way of proving theorems: writing programs!

To prove a formula $\phi$:

1. Convert the $\phi$ into its corresponding type $\tau$.
2. Find some program $v$ that has the type $\tau$.
3. Realize that the existence of $v$ implies a type tree for $\vdash v : \tau$, which implies a proof tree for $\vdash \phi$.

# Negation and Continuations

Let's explore one extension. We'd like to use this rule from classical logic:

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \neg\neg\phi}$$

but there's no obvious correspondence in System F.

# Negation and Continuations

Let's explore one extension. We'd like to use this rule from classical logic:

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \neg\neg\phi}$$

but there's no obvious correspondence in System F.

Recall that $\neg\phi$ is shorthand for $\phi \to \bot$. So $\neg\neg\phi$ corresponds to the System F function type $(\tau \to \bot) \to \bot$.

So what we need is a way to take any program of any type $\tau$ and turn it into a program of type $(\tau \to \bot) \to \bot$.

# Negation and Continuations

Let's explore one extension. We'd like to use this rule from classical logic:

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \neg\neg\phi}$$

but there's no obvious correspondence in System F.

Recall that $\neg\phi$ is shorthand for $\phi \to \bot$. So $\neg\neg\phi$ corresponds to the System F function type $(\tau \to \bot) \to \bot$.

So what we need is a way to take any program of any type $\tau$ and turn it into a program of type $(\tau \to \bot) \to \bot$.

Shockingly, that's exactly what the CPS transform does! A $\tau$ becomes a function that takes a continuation $\tau \to \bot$ and invokes it, producing $\bot$.