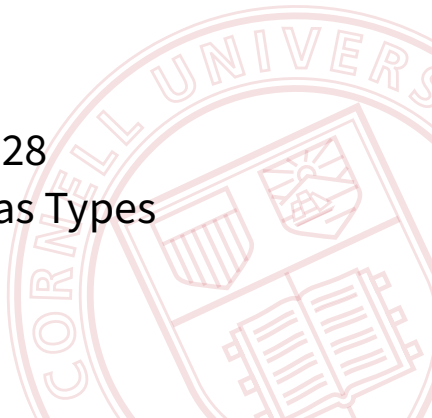


CS 4110

# Programming Languages & Logics

---

## Lecture 28 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:

$$\frac{\phi \quad \psi}{\phi \wedge \psi} \wedge\text{-INTRO}$$

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

# 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:

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

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

# 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.

# 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:

- $\vdash A \wedge B \rightarrow A$



# 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:

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

# 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:

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

# Natural Deduction

Let's write the rules for our judgment:

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

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

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

# Natural Deduction

Let's write the rules for our judgment:

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

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

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

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

# Natural Deduction

Let's write the rules for our judgment:

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

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

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

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

...and so on.

# Natural Deduction

$$\frac{}{\Gamma, \phi \vdash \phi} \text{AXIOM}$$

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

$$\frac{\Gamma \vdash \phi \rightarrow \psi \quad \Gamma \vdash \phi}{\Gamma \vdash \psi} \rightarrow\text{-ELIM}$$

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

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

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

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

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

$$\frac{\Gamma \vdash \phi \vee \psi \quad \Gamma \vdash \phi \rightarrow \chi \quad \Gamma \vdash \psi \rightarrow \chi}{\Gamma \vdash \chi} \vee\text{-ELIM}$$

$$\frac{\Gamma, P \vdash \phi}{\Gamma \vdash \forall P. \phi} \forall\text{-INTRO}$$

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

# Natural Deduction

---

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

$$\frac{\frac{\frac{}{A \wedge B \vdash A \wedge B} \text{AXIOM}}{A \wedge B \vdash B} \wedge\text{-ELIM2} \quad \frac{\frac{\frac{}{A \wedge B \vdash A \wedge B} \text{AXIOM}}{A \wedge B \vdash A} \wedge\text{-ELIM1}}{A \wedge B \vdash B \wedge A} \wedge\text{-INTRO}}{\vdash A \wedge B \rightarrow B \wedge A} \rightarrow\text{-INTRO}$$

# Natural Deduction

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

$$\frac{\frac{\frac{}{A \wedge B \vdash A \wedge B} \text{AXIOM}}{A \wedge B \vdash B} \wedge\text{-ELIM2} \quad \frac{\frac{\frac{}{A \wedge B \vdash A \wedge B} \text{AXIOM}}{A \wedge B \vdash A} \wedge\text{-ELIM1}}{A \wedge B \vdash B \wedge A} \wedge\text{-INTRO}}{\vdash A \wedge B \rightarrow B \wedge A} \rightarrow\text{-INTRO}$$

Does this look familiar?

# Natural Deduction

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

$$\begin{array}{c}
 \frac{}{A \wedge B \vdash A \wedge B} \text{AXIOM} \qquad \frac{}{A \wedge B \vdash A \wedge B} \text{AXIOM} \\
 \frac{}{A \wedge B \vdash B} \wedge\text{-ELIM2} \qquad \frac{}{A \wedge B \vdash A} \wedge\text{-ELIM1} \\
 \frac{}{A \wedge B \vdash B \wedge A} \wedge\text{-INTRO} \\
 \frac{}{\vdash A \wedge B \rightarrow B \wedge A} \rightarrow\text{-INTRO}
 \end{array}$$

Does this look familiar?

$$\begin{array}{c}
 \frac{}{x:A \times B \vdash x:A \times B} \text{T-VAR} \qquad \frac{}{x:A \times B \vdash x:A \times B} \text{T-VAR} \\
 \frac{}{x:A \times B \vdash \#2 x:B} \text{T-\#1} \qquad \frac{}{x:A \times B \vdash \#1 x:A} \text{T-\#2} \\
 \frac{}{x:A \times B \vdash (\#2 x, \#1 x):B \times A} \text{T-PAIR} \\
 \frac{}{\vdash \lambda x. (\#2 x, \#1 x):A \times B \rightarrow B \times A} \text{T-ABS}
 \end{array}$$

# 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	
$\rightarrow$	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!

# 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**

Gentzen (1935)



## **Typed $\lambda$ -Calculus**

Church (1940)

## **Type Schemes**

Hindley (1969)



## **ML's Type System**

Milner (1975)

## **System F**

Girard (1972)



## **Polymorphic $\lambda$ -Calculus**

Reynolds (1974)

## **Modal Logic**

Lewis (1910)



## **Monads**

Kleisli (1965), Moggi (1987)

## **Classical-Intuitionistic Embedding**

Gödel (1933)



## **Continuation Passing Style**

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  $\phi \rightarrow \perp$ . So  $\neg\neg\phi$  corresponds to the System F function type  $(\tau \rightarrow \perp) \rightarrow \perp$ .

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

# 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  $\phi \rightarrow \perp$ . So  $\neg\neg\phi$  corresponds to the System F function type  $(\tau \rightarrow \perp) \rightarrow \perp$ .

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

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