CS 4110

Programming Languages & Logics

Lecture 28
Propositions as Types

7 November 2016

Propositions as Types

Logics = Type Systems

Constructive Logic

Let's start with *constructive logic*, where the rules work like functions that take smaller proofs and generate larger proofs.

Constructive Logic

Let's start with *constructive logic*, where the rules work like functions that take smaller proofs and generate larger proofs.

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

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

Given a proof of ϕ and a proof of ψ, it lets you *construct* a proof of $\phi \wedge \psi$.

Natural Deduction

In natural deduction, we define the set of true formulas ("theorems") inductively.

Natural Deduction

In natural deduction, we define the set of true formulas ("theorems") inductively.

We'll start with a grammar for formulas:

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

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

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 Γ:

$$\Gamma \vdash \phi$$

where Γ is just a list of formulas.

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

Examples:

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

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 ϕ is a *theorem*.

Examples:

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

Natural Deduction

Let's write the rules for our judgment:

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

Natural Deduction

Let's write the rules for our judgment:

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

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

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:

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

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

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

...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! Here's a proof that $A \wedge B \rightarrow B \wedge A$ is a theorem.

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

Natural Deduction

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

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

Doesn't this look a little... familiar?

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

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**	
τ	Type	ϕ	Formula
τ	is inhabited	ϕ	is a theorem
e	Well-typed expression	π	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 λ-calculus were invented by Church at Princeton in 1940.

Propositions as Types Through the Ages

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

1. Convert the ϕ into its corresponding type τ.
2. Find some program v that has the type τ.
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 τ 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 τ and turn it into a program of type $(\tau \to \bot) \to \bot$.

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