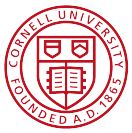
CS 4110 – Programming Languages and Logics
Lecture #29: Propositions as Types



1 Propositions as Types

There is a deep connection between type systems and formal logic. This connection, known as propositions-as-types, was recognized by early 20th century mathematicians and later developed substantially by Haskell Curry and William Howard. Although it is usually formulated in terms of simple type systems (or System F) and proof systems like natural deduction, the connection is actually quite robust and has been extended to many other systems including classical logic. It continues to bear fruit today: recent work by Abramsky, Pfenning, Wadler, and others has developed a connection between the session types used in concurrent process calculi and linear logic.

The main intuitions for propositions-as-types comes from thinking of proofs constructively. For example, the proof rule for introducting a conjunction $\phi \wedge \psi$,

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

can be thought of as a function that takes a proof of ϕ and a proof of ψ and builds a proof of $\phi \wedge \psi$. This is a significant departure from classical logic, which has rules such as excluded middle or double-negation elimination,

$$\frac{}{\Gamma \vdash \psi \vee \neg\psi} \text{ EXCLUDED MIDDLE}$$

that do not have an obvious constructive interpretation.

Propositions-as-types recongizes that each constructive proof can be turned into a program that witnesses the proof, as summarized by the following table.

Type Systems		Formal Logic	
τ	Type	ϕ	Formula
τ	Inhabited type	ϕ	Theorem
e	Well-typed expression	π	Proof

Hence, for every proof in first-order logic, we can obtain a well-typed expression in λ-calculus, and vice versa.

2 Natural Deduction

To illustrate propositions-as-types formally, we begin by reviewing natural deduction—a proof system for first-order logic. The syntax of first-order logic formulas is as follows,

$$\phi ::= \top \mid \bot \mid P \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \to \psi \mid \neg\phi \mid \forall x.\, \phi$$

where P ranges over propositional variables. We will let negation $\neg P$ be an abbreviation for $P \to \bot$.

The proof rules for natural deduction are as follows:

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

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\ \to\textsc{-intro} \qquad \frac{\Gamma \vdash \phi \to \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi}\ \to\textsc{-elim}$$

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\ \wedge\textsc{-intro} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\ \wedge\textsc{-elim1} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\ \wedge\textsc{-elim2}$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi}\ \vee\textsc{-intro1} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi}\ \vee\textsc{-intro2} \qquad \frac{\Gamma \vdash \phi \vee \psi \qquad \Gamma \vdash \phi \to \chi \qquad \Gamma \vdash \psi \to \chi}{\Gamma \vdash \chi}\ \vee\textsc{-elim}$$

$$\frac{\Gamma, P \vdash \phi}{\Gamma \vdash \forall P.\ \phi}\ \forall\textsc{-intro2} \qquad \frac{\Gamma \vdash \forall P.\ \phi}{\Gamma \vdash \phi\{\psi/P\}}\ \forall\textsc{-elim}$$

Note that some rules from classical logic, such as excluded middle,

$$\frac{}{\Gamma \vdash P \vee \neg P}$$

are not included, nor are they derivable from these rules.

3 System F Type System

It should be obvious that these proof rules bear a close correspondence to the type systems we have been developing over the class few weeks. Here are the typing rules for System F, annotated with the same labels:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ \textsc{Axiom}$$

$$\frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x : \sigma . e : \sigma \to \tau}\ \to\text{-INTRO} \qquad \frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\ e_2 : \tau}\ \to\text{-ELIM}$$

$$\frac{\Gamma \vdash e_1 : \sigma \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash (e_1, e_2) : \sigma \times \tau}\ \wedge\text{-INTRO} \qquad \frac{\Gamma \vdash e : \sigma \times \tau}{\Gamma \vdash \#1\ e : \sigma}\ \wedge\text{-ELIM1} \qquad \frac{\Gamma \vdash e : \sigma \times \tau}{\Gamma \vdash \#2\ e : \tau}\ \wedge\text{-ELIM2}$$

$$\frac{\Gamma \vdash e : \sigma}{\Gamma \vdash inl_{\sigma+\tau}\ e : \sigma + \tau}\ \vee\text{-INTRO1} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash inr_{\sigma+\tau}\ e : \sigma + \tau}\ \vee\text{-INTRO2}$$

$$\frac{\Gamma \vdash e\sigma + \tau \qquad \Gamma \vdash e_1 : \sigma \to \rho \qquad \Gamma \vdash e_2 : \tau \to \rho}{\Gamma \vdash case\ e\ of\ e_1\ e_2 : \rho}\ \vee\text{-ELIM}$$

$$\frac{\Delta, \alpha; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \Lambda\alpha.\ e : \forall\alpha.\tau}\ \forall\text{-INTRO2} \qquad \frac{\Gamma \vdash e : \forall\alpha.\ \tau}{\Gamma \vdash e\ [\sigma] : \tau\{\sigma/\alpha\}}\ \forall\text{-ELIM}$$

We can summarize the relationship between formulas and types using the following table:

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

This relationship has also be extended to many other types.

4 Term Assignment

Given a natural deduction proof, there is a corresponding well-typed System F expression. The transformation from a proof to an expression is often called *term assignment*. For example, given the following proof,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \phi \to \psi, \phi \vdash \phi \to \psi}\ \textsc{Axiom} \qquad \dfrac{}{\Gamma, \phi \to \psi, \phi \vdash \phi}\ \textsc{Axiom}}{\Gamma, \phi \to \psi, \phi \vdash \psi}\ \to\text{-ELIM}}{\Gamma, \phi \to \psi, \vdash \phi \to \psi}\ \to\text{-INTRO}}{\Gamma \vdash (\phi \to \psi) \to \phi \to \psi}\ \to\text{-INTRO}$$

we can build the following typing derivation:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : \sigma \to \tau, y : \sigma \vdash x : \sigma \to \tau}\ \textsc{Axiom} \qquad \dfrac{}{\Gamma, x : \sigma \to \tau, y : \sigma \vdash \sigma}\ \textsc{Axiom}}{\Gamma, x : \sigma \to \tau, y : \sigma \vdash x\ y : \tau}\ \to\text{-ELIM}}{\Gamma, x : \sigma \to \tau \vdash \lambda y{:}\sigma.\, x\ y : \sigma \to \tau}\ \to\text{-INTRO}}{\Gamma \vdash \lambda x{:}\sigma \to \tau.\, \lambda y{:}\sigma.\, x\ y : (\sigma \to \tau) \to \sigma \to \tau}\ \to\text{-INTRO}$$

Hence, the term

$$\lambda x{:}\sigma \to \tau.\, \lambda y{:}\sigma.\, x\ y$$

witnesses the proof of

$$(\phi \to \psi) \to \phi \to \psi$$

More generally, to prove a proposition ϕ, it suffices to find a well-typed expression e of type τ where ϕ and τ are related under propositions as types.

5 Negation and Continuations

The problem of extending propositions-as-types to classical logic was an open question for many years. It was not at all obvious how to do this, as the usual constructive interpretation of proofs does not readily extend to rules such as excluded middle. In the late 1980s, Griffin showed that continuation-passing style corresponds to a way of embedding classical logic into constructive logic.

Given an expression e of type τ, we can think of the continuation-passing style transformation as converting the expression into one of type $(\tau \to \bot) \to \bot$. Intuitively, the $\tau \to \bot$ is the type of the continuation, which "never returns." Since $\neg\phi$ is just an abbreviation for $\phi \to \bot$, this corresponds to the following classical rule:

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

This yields a way to prove any formula that is classically valid in constructive logic using the double-negation embedding.