

Type Systems

Lecture 2: The Curry-Howard Correspondence

Neel Krishnaswami
University of Cambridge

Type Systems for Programming Languages

- Type systems lead a double life
- They are a fundamental concept from logic and proof theory
- They are an essential part of modern programming languages

Natural Deduction

- In the early part of the 20th century, mathematics grew very abstract
- As a result, simple numerical and geometric intuitions no longer seemed to be sufficient to justify mathematical proofs (eg, Cantor's proofs about infinite sets)
- Big idea of Frege, Russell, Hilbert: what if we treated theorems and proofs as ordinary mathematical objects?
- Dramatic successes and failures, but the formal systems they introduced were unnatural – proofs didn't look like human proofs
- In 1933 (at age 23!) Gerhard Gentzen invented natural deduction
- “Natural” because the proof style is natural (with a little squinting)

Natural Deduction: Propositional Logic

What are propositions?

- $\top$ is a proposition
- $P \wedge Q$ is a proposition, if P and Q are propositions
- $\perp$ is a proposition
- $P \vee Q$ is a proposition, if P and Q are propositions
- $P \supset Q$ is a proposition, if P and Q are propositions

These are the formulas of propositional logic (i.e., no quantifiers of the form “for all x , $P(x)$ ” or “there exists x , $P(x)$ ”).

Judgements

- Some claims follow (e.g. $P \wedge Q \supset Q \wedge P$).
- Some claims don't. (e.g., $\top \supset \perp$)
- We judge which propositions hold, and which don't with judgements
- In particular, “ P true” means we judge P to be true.
- How do we justify judgements? With inference rules!

Truth and Conjunction

$$\frac{}{\top \text{ true}} \top I$$

$$\frac{P \text{ true} \quad Q \text{ true}}{P \wedge Q \text{ true}} \wedge I$$

$$\frac{P \wedge Q \text{ true}}{P \text{ true}} \wedge E_1$$

$$\frac{P \wedge Q \text{ true}}{Q \text{ true}} \wedge E_1$$

Implication

- To prove $P \supset Q$ in math, we assume P and prove Q
- Therefore, our notion of judgement needs to keep track of assumptions as well!
- So we introduce $\Psi \vdash P \text{ true}$, where Ψ is a list of assumptions
- Read: “Under assumptions Ψ , we judge P true”

$$\frac{P \in \Psi}{\Psi \vdash P \text{ true}} \text{ HYP} \qquad \frac{\Psi, P \vdash Q \text{ true}}{\Psi \vdash P \supset Q \text{ true}} \supset I$$

$$\frac{\Psi \vdash P \supset Q \text{ true} \quad \Psi \vdash P \text{ true}}{\Psi \vdash Q \text{ true}} \supset E$$

Disjunction and Falsehood

$$\frac{\Psi \vdash P \text{ true}}{\Psi \vdash P \vee Q \text{ true}} \vee I_1$$

$$\frac{\Psi \vdash Q \text{ true}}{\Psi \vdash P \vee Q \text{ true}} \vee I_2$$

$$\frac{\Psi \vdash P \vee Q \text{ true} \quad \Psi, P \vdash R \text{ true} \quad \Psi, Q \vdash R \text{ true}}{\Psi \vdash R \text{ true}} \vee E$$

(no intro for $\perp$)

$$\frac{\Psi \vdash \perp \text{ true}}{\Psi \vdash R \text{ true}} \perp E$$

Example

$$\frac{\frac{\frac{}{(P \vee Q) \supset R, P \vdash (P \vee Q) \supset R \text{ true}}}{(P \vee Q) \supset R, P \vdash P \supset R \text{ true}} \quad \frac{\frac{}{(P \vee Q) \supset R, P \vdash P \text{ true}}}{(P \vee Q) \supset R, P \vdash P \vee Q \text{ true}}}{(P \vee Q) \supset R, P \vdash R \text{ true}} \quad \dots$$
$$\frac{(P \vee Q) \supset R \vdash P \supset R \text{ true}}{(P \vee Q) \supset R \vdash (P \supset R) \wedge (Q \supset R) \text{ true}}$$
$$\vdash ((P \vee Q) \supset R) \supset ((P \supset R) \wedge (Q \supset R)) \text{ true}$$

The Typed Lambda Calculus

Types	$X ::= 1 \mid X \times Y \mid 0 \mid X + Y \mid X \rightarrow Y$
Terms	$e ::= x \mid \langle \rangle \mid \langle e, e \rangle \mid \text{fst } e \mid \text{snd } e$ $\mid \text{abort} \mid L e \mid R e \mid \text{case}(e, L x \rightarrow e', R y \rightarrow e'')$ $\mid \lambda x : X. e \mid e e'$
Contexts	$\Gamma ::= \cdot \mid \Gamma, x : X$

A typing judgement is of the form $\Gamma \vdash e : X$.

Units and Pairs

$$\frac{}{\Gamma \vdash \langle \rangle : 1} \text{1I}$$

$$\frac{\Gamma \vdash e : X \quad \Gamma \vdash e' : Y}{\Gamma \vdash \langle e, e' \rangle : X \times Y} \times\text{I}$$

$$\frac{\Gamma \vdash e : X \times Y}{\Gamma \vdash \text{fst } e : X} \times\text{E}_1$$

$$\frac{\Gamma \vdash e : X \times Y}{\Gamma \vdash \text{snd } e : Y} \times\text{E}_1$$

$$\frac{X : X \in \Gamma}{\Gamma \vdash x : X} \text{HYP} \qquad \frac{\Gamma, X \vdash e : Y}{\Gamma \vdash \lambda x : X. e : X \rightarrow Y} \rightarrow\text{I}$$
$$\frac{\Gamma \vdash e : X \rightarrow Y \quad \Gamma \vdash e' : X}{\Gamma \vdash e e' : Y} \rightarrow\text{E}$$

Sums and the Empty Type

$$\frac{\Gamma \vdash e : X}{\Gamma \vdash L e : X + Y} +I_1$$

$$\frac{\Gamma \vdash e : Y}{\Gamma \vdash R e : X + Y} +I_2$$

$$\frac{\Gamma \vdash e : X + Y \quad \Gamma, x : X \vdash e' : Z \quad \Gamma, y : Y \vdash e'' : Z}{\Gamma \vdash \text{case}(e, L x \rightarrow e', R y \rightarrow e'') : Z} +E$$

(no intro for 0)

$$\frac{\Gamma \vdash e : 0}{\Gamma \vdash \text{abort } e : Z} 0E$$

Example

$$\begin{aligned} \lambda f : (X + Y) \rightarrow Z. \langle \lambda x : X. f (L\ x), \lambda y : Y. f (R\ y) \rangle \\ : \\ ((X + Y) \rightarrow Z) \rightarrow (X \rightarrow Z) \times (Y \rightarrow Z) \end{aligned}$$

You may notice a similarity here...!

The Curry-Howard Correspondence, Part 1

Logic	Programming
Formulas	Types
Proofs	Programs
Truth	Unit
Falsehood	Empty type
Conjunction	Pairing/Records
Disjunction	Tagged Union
Implication	Functions

Something missing: language semantics?

Operational Semantics of the Typed Lambda Calculus

Values $v ::= \langle \rangle \mid \langle v, v' \rangle \mid \lambda x : A. e \mid L v \mid R v$

The transition relation is $e \rightsquigarrow e'$, pronounced “e steps to e’”.

Operational Semantics: Units and Pairs

(no rules for unit)

$$\frac{e_1 \rightsquigarrow e'_1}{\langle e_1, e_2 \rangle \rightsquigarrow \langle e'_1, e_2 \rangle}$$

$$\frac{e_2 \rightsquigarrow e'_2}{\langle v_1, e_2 \rangle \rightsquigarrow \langle v_1, e'_2 \rangle}$$

$$\frac{}{\text{fst } \langle v_1, v_2 \rangle \rightsquigarrow v_1}$$

$$\frac{}{\text{snd } \langle v_1, v_2 \rangle \rightsquigarrow v_2}$$

Operational Semantics: Void and Sums

$$\frac{e \rightsquigarrow e'}{\text{abort } e \rightsquigarrow \text{abort } e'}$$

$$\frac{e \rightsquigarrow e'}{L e \rightsquigarrow L e'}$$

$$\frac{e \rightsquigarrow e'}{R e \rightsquigarrow R e'}$$

$$\frac{e \rightsquigarrow e'}{\text{case}(e, L x \rightarrow e_1, R y \rightarrow e_2) \rightsquigarrow \text{case}(e', L x \rightarrow e_1, R y \rightarrow e_2)}$$

$$\frac{}{\text{case}(L v, L x \rightarrow e_1, R y \rightarrow e_2) \rightsquigarrow [v/x]e_1}$$

$$\frac{}{\text{case}(R v, L x \rightarrow e_1, R y \rightarrow e_2) \rightsquigarrow [v/y]e_2}$$

Operational Semantics: Functions

$$\frac{e_1 \rightsquigarrow e'_1}{e_1 e_2 \rightsquigarrow e'_1 e_2}$$

$$\frac{e_2 \rightsquigarrow e'_2}{v_1 e_2 \rightsquigarrow v_1 e'_2}$$

$$\frac{}{(\lambda x : X. e) v \rightsquigarrow [v/x]e}$$

Five Easy Lemmas

1. (Weakening) If $\Gamma, \Gamma' \vdash e : X$ then $\Gamma, z : Z, \Gamma' \vdash e : X$.
2. (Exchange) If $\Gamma, y : Y, z : Z, \Gamma' \vdash e : X$ then $\Gamma, z : Z, y : Y, \Gamma' \vdash e : X$.
3. (Substitution) If $\Gamma \vdash e : X$ and $\Gamma, x : X \vdash e' : Y$ then $\Gamma \vdash [e/x]e' : Y$.
4. (Progress) If $\cdot \vdash e : X$ then e is a value, or $e \rightsquigarrow e'$.
5. (Preservation) If $\cdot \vdash e : X$ and $e \rightsquigarrow e'$, then $\cdot \vdash e' : X$.

Proof technique similar to previous lecture. But what does it mean, logically?

Two Kinds of Reduction Step

Congruence Rules	Reduction Rules
$\frac{e_1 \rightsquigarrow e'_1}{\langle e_1, e_2 \rangle \rightsquigarrow \langle e'_1, e_2 \rangle}$	$\frac{}{\text{fst } \langle v_1, v_2 \rangle \rightsquigarrow v_1}$
$\frac{e_2 \rightsquigarrow e'_2}{v_1 e_2 \rightsquigarrow v_1 e'_2}$	$\frac{}{(\lambda x : X. e) v \rightsquigarrow [v/x]e}$

- Congruence rules recursively act on a subterm
 - Controls evaluation order
- Reduction rules actually transform a term
 - Actually evaluates!

A Closer Look at Reduction

Let's look at the function reduction case:

$$(\lambda x : X. e) v \rightsquigarrow [v/x]e$$

$$\frac{\frac{\boxed{x : X \vdash e : Y}}{\cdot \vdash \lambda x : X. e : X \rightarrow Y} \rightarrow I \quad \boxed{\cdot \vdash v : X}}{\cdot \vdash (\lambda x : X. e) v : Y} \rightarrow E$$

- Reducible term = intro immediately followed by an elim
- Evaluation = removal of this detour

All Reductions Remove Detours

$$\frac{}{\text{fst } \langle v_1, v_2 \rangle \rightsquigarrow v_1}$$

$$\frac{}{\text{snd } \langle v_1, v_2 \rangle \rightsquigarrow v_2}$$

$$\frac{}{\text{case}(\text{L } v, \text{L } x \rightarrow e_1, \text{R } y \rightarrow e_2) \rightsquigarrow [v/x]e_1}$$

$$\frac{}{\text{case}(\text{R } v, \text{L } x \rightarrow e_1, \text{R } y \rightarrow e_2) \rightsquigarrow [v/y]e_2}$$

$$\frac{}{(\lambda x : X. e) v \rightsquigarrow [v/x]e}$$

Every reduction is of an introduction followed by an eliminator!

Values as Normal Forms

Values $v ::= \langle \rangle \mid \langle v, v' \rangle \mid \lambda x : A. e \mid L v \mid R v$

- Note that values are introduction forms
- Note that values are not reducible expressions
- So programs evaluate towards a normal form
- Choice of which normal form to look at it determined by evaluation order

The Curry-Howard Correspondence, Continued

Logic	Programming
Formulas	Types
Proofs	Programs
Truth	Unit
Falsehood	Empty type
Conjunction	Pairing/Records
Disjunction	Tagged Union
Implication	Functions
Normal form	Value
Proof normalization	Evaluation
Normalization strategy	Evaluation order

The Curry-Howard Correspondence is Not an Isomorphism

The logical derivation:

$$\frac{\frac{}{P, P \vdash P \text{ true}} \quad \frac{}{P, P \vdash P \text{ true}}}{P, P \vdash P \wedge P \text{ true}}$$

has 4 type-theoretic versions:

$$\frac{\vdots}{x : X, y : X \vdash \langle x, x \rangle : X \wedge X}$$

$$\frac{\vdots}{x : X, y : X \vdash \langle y, y \rangle : X \wedge X}$$

$$\frac{\vdots}{x : X, y : X \vdash \langle x, y \rangle : X \wedge X}$$

$$\frac{\vdots}{x : X, y : X \vdash \langle y, x \rangle : X \wedge X}$$

For the $1, \rightarrow$ fragment of the typed lambda calculus, prove type safety.

1. Prove weakening.
2. Prove exchange.
3. Prove substitution.
4. Prove progress.
5. Prove type preservation.