Proofs as programs (and programs as proofs)

Computational semantics seminar [with Dylan Bumford]

April 26, 2017

And now for something completely different?

Today, we're going to look at one of the deepest, prettiest results anywhere: a fundamental equivalence of logic and computation.

It might at first seem far removed from what we've been doing so far in the class, but (of course) it turns out not to be!

Another view on computational semantics, an **operational** one, with connections to monadic and scopal composition.

Natural deduction

Axioms for implicational (classical) logic

Here's a complete axiomatization of classical logic:

K	$(A \supset (B \supset A))$
S	$(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$
DNE	$(\underbrace{\neg\neg A}_{(A \supset \bot) \supset \bot} \supset A)$

Great, let's prove that $p \supset p$:

$$
\begin{array}{rl}
(p \supset ((p \supset p) \supset p)) & \textbf{K} \\
((p \supset ((p \supset p) \supset p)) \supset ((p \supset (p \supset p)) \supset (p \supset p))) & \textbf{S} \\
((p \supset (p \supset p)) \supset (p \supset p)) & [\textbf{MP}] \\
(p \supset (p \supset p)) & \textbf{K} \\
(p \supset p) & [\textbf{MP}]
\end{array}
$$

Axioms for implicational (classical) logic

Here's a complete axiomatization of classical logic:

K	$(A \supset (B \supset A))$
S	$(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$
DNE	$(\underbrace{\neg\neg A}_{(A \supset \bot) \supset \bot} \supset A)$

Great, let's prove that $p \supset p$:

$$
\begin{array}{rl}
(p \supset ((p \supset p) \supset p)) & \textbf{K} \\
((p \supset ((p \supset p) \supset p)) \supset ((p \supset (p \supset p)) \supset (p \supset p))) & \textbf{S} \\
((p \supset (p \supset p)) \supset (p \supset p)) & [\textbf{MP}] \\
(p \supset (p \supset p)) & \textbf{K} \\
(p \supset p) & [\textbf{MP}]
\end{array}
$$

This *sucks*.

Natural deduction (Gentzen 1935)

$$\frac{A \qquad B}{A \wedge B}\,\wedge\mathsf{I} \qquad \frac{A \wedge B}{A}\,\wedge\mathsf{E}_1 \qquad \frac{A \wedge B}{B}\,\wedge\mathsf{E}_2$$

$$\frac{\begin{array}{c}[A]^x \\ \vdots \\ B\end{array}}{A \supset B}\,\supset\mathsf{I}^x \qquad \frac{A \supset B \qquad A}{B}\,\supset\mathsf{E}$$

$$\frac{A}{A \vee B}\,\vee\mathsf{I}_1 \qquad \frac{B}{A \vee B}\,\vee\mathsf{I}_2 \qquad \frac{A \vee B \qquad A \supset C \qquad B \supset C}{C}\,\vee\mathsf{E}$$

Introduction and elimination rules

Introduction (I) rules tell you what a connective **means**

Elimination (E) rules tell you how to **use (up)** a connective, and the idea is that a connective's E rules follow from the its I rule(s).

Deriving **I** and **K**

$$\frac{[A]^x}{A \supset A} \supset \mathsf{I}^x \qquad\qquad \frac{[B]^y \quad \dfrac{[A]^x}{B \supset A} \supset \mathsf{I}^y}{A \supset (B \supset A)} \supset \mathsf{I}^x$$

[**S** is of course derivable as well. Left as an exercise.]

Warm-up examples: Two-way 'modus ponens'

$$\dfrac{\dfrac{[A \supset B]^y \qquad [A]^x}{B}\supset\mathsf{E}}{\dfrac{B}{A \supset B}\supset\mathsf{I}^x}\supset\mathsf{I}^y$$
$$(A \supset B) \supset (A \supset B)$$

$$\dfrac{\dfrac{[A \supset B]^y \qquad [A]^x}{B}\supset\mathsf{E}}{\dfrac{B}{(A \supset B) \supset B}\supset\mathsf{I}^y}\supset\mathsf{I}^x$$
$$A \supset ((A \supset B) \supset B)$$

A **proof** is a tree of deductions, with each deduction licensed by an I or E rule, and with every assumption x eliminated by a corresponding $\supset\mathsf{I}^x$.

Example: Transitivity of implication

$$\dfrac{\dfrac{\dfrac{\dfrac{[A \supset B]^x \qquad [A]^y}{B}\supset E \qquad [B \supset C]^z}{C}\supset E}{A \supset C}\supset I^y}{(B \supset C) \supset (A \supset C)}\supset I^z}{(A \supset B) \supset ((B \supset C) \supset (A \supset C))}\supset I^x$$

Example: Import/Export

$$\dfrac{\dfrac{[A \supset (B \supset C)]^x \qquad \dfrac{[A \wedge B]^y}{A}\wedge E_1}{B \supset C}\supset E \qquad \dfrac{[A \wedge B]^y}{B}\wedge E_2}{\dfrac{C}{(A \wedge B) \supset C}\supset I^y}\supset E$$
$$\dfrac{}{(A \supset (B \supset C)) \supset ((A \wedge B) \supset C)}\supset I^x$$

$$\dfrac{\dfrac{[(A \wedge B) \supset C]^x \qquad \dfrac{[A]^y \qquad [B]^z}{A \wedge B}\wedge I}{C}\supset E}{\dfrac{\dfrac{C}{B \supset C}\supset I^z}{A \supset (B \supset C)}\supset I^y}\supset E$$
$$\dfrac{}{((A \wedge B) \supset C) \supset (A \supset (B \supset C))}\supset I^x$$

Detours in proofs

Proofs can be more indirect than they need to be:

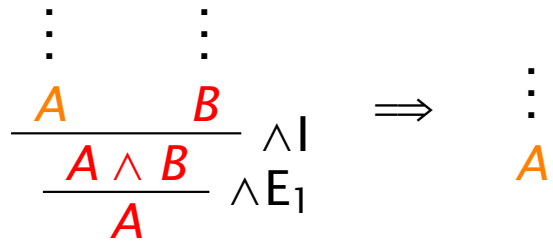
$$\dfrac{\dfrac{\begin{array}{c}[A]^x\\ \vdots\\ B\end{array}}{A \supset B}\supset \mathrm{I}^x \qquad \begin{array}{c}\vdots\\ A\end{array}}{B}\supset \mathrm{E}$$

Since we already have a proof of A, we can just use it **directly** in our proof of B, instead of needing to assume it:

$$\begin{array}{c}\vdots\\ A\\ \vdots\\ B\end{array}$$

Thus, the first proof is said to involve a **detour**.

Another kind of detour



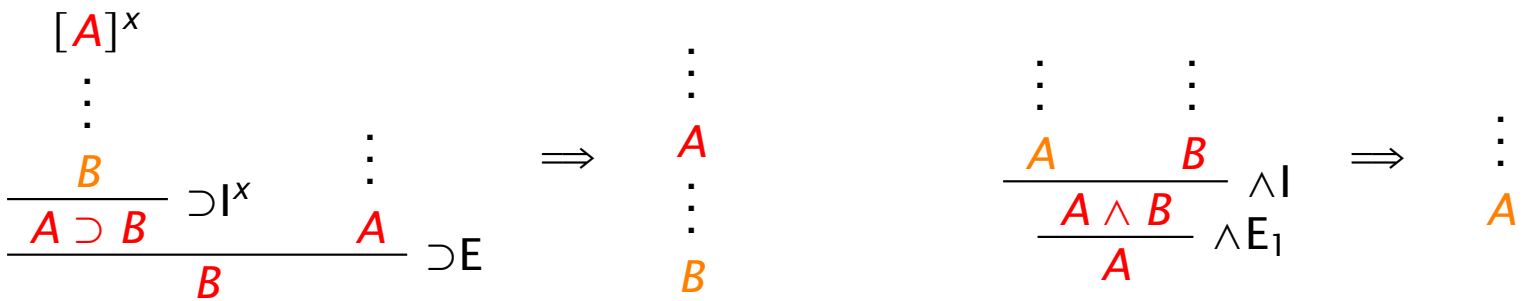
Here, we introduce a $\wedge$, only to eliminate it in short order. Much more direct would be to simply use our proof of A!

[There are also rules for eliminating disjunctive detours, but we'll ignore them.]

Normal forms

A proof is in **normal form** if it makes use of no detours.

Somewhat more formally, normal form proofs don't use the outputs of introduction rules as a (major) premise of elimination rules:



Example: chaining reductions to produce a normal form

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]^x}{A}\wedge\mathsf{E}_2 \quad \dfrac{[B \wedge A]^x}{B}\wedge\mathsf{E}_1}{A \wedge B}\wedge\mathsf{I}}{(B \wedge A) \supset (A \wedge B)}\supset\mathsf{I}^x \quad \dfrac{B \quad A}{B \wedge A}\wedge\mathsf{I}}{A \wedge B}\supset\mathsf{E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{B \quad A}{B \wedge A}\wedge\mathsf{I}}{A}\wedge\mathsf{E}_2 \quad \dfrac{\dfrac{B \quad A}{B \wedge A}\wedge\mathsf{I}}{B}\wedge\mathsf{E}_1}{A \wedge B}\wedge\mathsf{I}$$

$$\Downarrow$$

$$\dfrac{A \quad B}{A \wedge B}\wedge\mathsf{I}$$

[Example borrowed from Wadler (2015)]

The subformula property

Proofs in normal form display something called the **subformula** property: nothing is ever mentioned in the proof besides parts of its conclusion!

We won't focus on this today, but it's important! It makes constructing proofs much easier, and guarantees that proof search **terminates**.

Type theory

A natural deduction system for the simply typed λ-calculus

$$\frac{M : A \qquad N : B}{(M, N) : A \times B}\ \times\mathsf{I} \qquad \frac{L : A \times B}{\mathbf{fst}\, L : A}\ \times\mathsf{E}_1 \qquad \frac{L : A \times B}{\mathbf{snd}\, L : B}\ \times\mathsf{E}_2$$

$$\frac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x.N : A \to B}\ \to\mathsf{I}^x \qquad \frac{M : A \to B \qquad N : A}{M\,N : B}\ \to\mathsf{E}$$

$$\frac{M : A}{\mathbf{inl}\, M : A \mid B}\ \mid\mathsf{I}_1 \qquad \frac{M : B}{\mathbf{inr}\, M : A \mid B}\ \mid\mathsf{I}_2 \qquad \frac{M : A \mid B \qquad F : A \to C \qquad G : B \to C}{\mathbf{cases}\, M\, F\, G : C}\ \mid\mathsf{E}$$

Focusing on the red parts, do you notice anything?

Identical systems!

$$\frac{M : A \qquad N : B}{(M, N) : A \times B}\ \times\mathsf{I} \qquad \frac{L : A \times B}{\mathbf{fst}\, L : A}\ \times\mathsf{E}_1 \qquad \frac{L : A \times B}{\mathbf{snd}\, L : B}\ \times\mathsf{E}_2$$

$$\frac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x.\, N : A \to B}\ \to\mathsf{I}^x \qquad \frac{M : A \to B \qquad N : A}{M\, N : B}\ \to\mathsf{E}$$

$$\frac{M : A}{\mathbf{inl}\, M : A \mid B}\ \mid\mathsf{I}_1 \qquad \frac{M : B}{\mathbf{inr}\, M : A \mid B}\ \mid\mathsf{I}_2 \qquad \frac{M : A \mid B \qquad F : A \to C \qquad G : B \to C}{\mathbf{cases}\, M\, F\, G : C}\ \mid\mathsf{E}$$

Identical systems!

$$\frac{A \qquad B}{A \wedge B}\ \wedge\mathrm{I} \qquad \frac{A \wedge B}{A}\ \wedge\mathrm{E}_1 \qquad \frac{A \wedge B}{B}\ \wedge\mathrm{E}_2$$

$$\frac{\begin{array}{c}[A]^x \\ \vdots \\ B\end{array}}{A \supset B}\ \supset\mathrm{I}^x \qquad \frac{A \supset B \qquad A}{B}\ \supset\mathrm{E}$$

$$\frac{A}{A \vee B}\ \vee\mathrm{I}_1 \qquad \frac{B}{A \vee B}\ \vee\mathrm{I}_2 \qquad \frac{A \vee B \qquad A \supset C \qquad B \supset C}{C}\ \vee\mathrm{E}$$

Curry-Howard

The logic of (simple) types is in *exact* correspondence with the (minimal) propositional system sketched the previous section.

In an important sense, then, the two systems are **equivalent**!!!

This connection, between propositional logic and computation, is known as the **Curry-Howard isomorphism**. “Curry” is for Haskell Curry (1934). “Howard” is for William Howard (1980, in circulation since 1969).

A note on coproducts

The **coproduct** of A and B, written '$A \mid B$', is their **disjoint union**:

$$A \mid B := \{\mathbf{inl}\, a \mid a \in A\} \cup \{\mathbf{inr}\, b \mid b \in B\}$$

The **cases** function applies F or G, depending on how M is tagged:

$$\mathbf{cases} \quad : \quad (A \mid B) \to (A \to C) \to (B \to C) \to C$$

$$\mathbf{cases}\, M\, F\, G := \begin{cases} F\, X & \text{if } M = \mathbf{inl}\, X \\ G\, Y & \text{if } M = \mathbf{inr}\, Y \end{cases}$$

An example of how this works:

$$\mathbf{cases}\, (\mathbf{inr}\, 2)\, (\lambda x.\, \mathbb{T})\, (\lambda x.\, \mathbb{F}) =$$

A note on coproducts

The **coproduct** of A and B, written '$A \mid B$', is their **disjoint union**:

$$A \mid B := \{\mathbf{inl}\, a \mid a \in A\} \cup \{\mathbf{inr}\, b \mid b \in B\}$$

The **cases** function applies F or G, depending on how M is tagged:

$$\mathbf{cases} \quad : \quad (A \mid B) \to (A \to C) \to (B \to C) \to C$$

$$\mathbf{cases}\, M\, F\, G := \begin{cases} F\, X & \text{if } M = \mathbf{inl}\, X \\ G\, Y & \text{if } M = \mathbf{inr}\, Y \end{cases}$$

An example of how this works:

$$\mathbf{cases}\, (\mathbf{inr}\, 2)\, (\lambda x.\, \mathbb{T})\, (\lambda x.\, \mathbb{F}) = \mathbb{F}$$

Warm-up examples: Two-way ‘modus ponens’

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda x.f\,x : A \to B}\,{\to}\mathsf{I}^x}{\lambda f.\lambda x.f\,x : (A \to B) \to A \to B}\,{\to}\mathsf{I}^f$$

What have we derived?

Warm-up examples: Two-way 'modus ponens'

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda x.f\,x : A \to B}\,{\to}\mathsf{I}^x}{\lambda f.\lambda x.f\,x : (A \to B) \to A \to B}\,{\to}\mathsf{I}^f$$

What have we derived? A term that takes a function and an argument, and applies the function to the argument. Ho-hum?

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda f.f\,x : (A \to B) \to B}\,{\to}\mathsf{I}^f}{\lambda x.\lambda f.f\,x : A \to (A \to B) \to B}\,{\to}\mathsf{I}^x$$

What have we derived?

Warm-up examples: Two-way 'modus ponens'

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda x.f\,x : A \to B}\,{\to}\mathsf{I}^x}{\lambda f.\lambda x.f\,x : (A \to B) \to A \to B}\,{\to}\mathsf{I}^f$$

What have we derived? A term that takes a function and an argument, and applies the function to the argument. Ho-hum?

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda f.f\,x : (A \to B) \to B}\,{\to}\mathsf{I}^f}{\lambda x.\lambda f.f\,x : A \to (A \to B) \to B}\,{\to}\mathsf{I}^x$$

What have we derived? A term that takes an argument and a function, and applies the function to the argument. Aka. . .

Warm-up examples: Two-way 'modus ponens'

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda x.f\,x : A \to B}\,{\to}\mathsf{I}^x}{\lambda f.\lambda x.f\,x : (A \to B) \to A \to B}\,{\to}\mathsf{I}^f$$

What have we derived? A term that takes a function and an argument, and applies the function to the argument. Ho-hum?

$$\dfrac{\dfrac{\dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{\lambda f.f\,x : (A \to B) \to B}\,{\to}\mathsf{I}^f}{\lambda x.\lambda f.f\,x : A \to (A \to B) \to B}\,{\to}\mathsf{I}^x$$

What have we derived? A term that takes an argument and a function, and applies the function to the argument. Aka. . . **LIFT**!

Example: Transitivity of implication

$$\dfrac{\dfrac{\dfrac{\dfrac{[g : B \to C]^g \qquad \dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{g\,(f\,x) : C}\,{\to}\mathsf{E}}{\lambda x.\,g\,(f\,x) : A \to C}\,{\to}\mathsf{I}^x}{\lambda g.\lambda x.\,g\,(f\,x) : (B \to C) \to A \to C}\,{\to}\mathsf{I}^g}{\lambda f.\lambda g.\lambda x.\,g\,(f\,x) : (A \to B) \to (B \to C) \to A \to C}\,{\to}\mathsf{I}^f$$

This corresponds to...

Example: Transitivity of implication

$$\dfrac{\dfrac{\dfrac{\dfrac{[g : B \to C]^g \qquad \dfrac{[f : A \to B]^f \qquad [x : A]^x}{f\,x : B}\,{\to}\mathsf{E}}{g\,(f\,x) : C}\,{\to}\mathsf{E}}{\lambda x.\,g\,(f\,x) : A \to C}\,{\to}\mathsf{I}^x}{\lambda g.\lambda x.\,g\,(f\,x) : (B \to C) \to A \to C}\,{\to}\mathsf{I}^g}{\lambda f.\lambda g.\lambda x.\,g\,(f\,x) : (A \to B) \to (B \to C) \to A \to C}\,{\to}\mathsf{I}^f$$

This corresponds to. . . **function composition**!

Example: Import/Export

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[f : A \to B \to C]^f \qquad \dfrac{[p : A \times B]^p}{\mathbf{fst}\, p : A}\times \mathsf{E}_1}{f\,(\mathbf{fst}\, p) : B \to C}\to \mathsf{E} \qquad \dfrac{[p : A \times B]^p}{\mathbf{snd}\, p : B}\times \mathsf{E}_2}{f\,(\mathbf{fst}\, p)\,(\mathbf{snd}\, p) : C}\to \mathsf{E}}{\lambda p.\, f\,(\mathbf{fst}\, p)\,(\mathbf{snd}\, p) : (A \times B) \to C}\to \mathsf{I}^p}{\lambda f.\, \lambda p.\, f\,(\mathbf{fst}\, p)\,(\mathbf{snd}\, p) : (A \to B \to C) \to (A \times B) \to C}\to \mathsf{I}^f$$

$$\dfrac{\dfrac{\dfrac{\dfrac{[f : (A \times B) \to C]^f \qquad \dfrac{[x : A]^x \qquad [y : B]^y}{(x, y) : A \times B}\times \mathsf{I}}{f\,(x, y) : C}\to \mathsf{E}}{\lambda y.\, f\,(x, y) : B \to C}\to \mathsf{I}^y}{\lambda x.\, \lambda y.\, f\,(x, y) : A \to B \to C}\to \mathsf{I}^x}{\lambda f.\, \lambda x.\, \lambda y.\, f\,(x, y) : ((A \times B) \to C) \to A \to B \to C}\to \mathsf{I}^f$$

So the Import-Export equivalence is...

Example: Import/Export

$$\dfrac{\dfrac{\dfrac{\dfrac{[f : A \to B \to C]^f \qquad \dfrac{[p : A \times B]^p}{\mathbf{fst}\, p : A}\times \mathsf{E}_1}{f\,(\mathbf{fst}\, p) : B \to C}\to \mathsf{E} \qquad \dfrac{[p : A \times B]^p}{\mathbf{snd}\, p : B}\times \mathsf{E}_2}{f\,(\mathbf{fst}\, p)\,(\mathbf{snd}\, p) : C}\to \mathsf{E}}{\lambda p.f\,(\mathbf{fst}\, p)\,(\mathbf{snd}\, p) : (A \times B) \to C}\to \mathsf{I}^p}{\lambda f.\lambda p.f\,(\mathbf{fst}\, p)\,(\mathbf{snd}\, p) : (A \to B \to C) \to (A \times B) \to C}\to \mathsf{I}^f$$

$$\dfrac{\dfrac{\dfrac{\dfrac{[f : (A \times B) \to C]^f \qquad \dfrac{[x : A]^x \qquad [y : B]^y}{(x, y) : A \times B}\times \mathsf{I}}{f\,(x, y) : C}\to \mathsf{E}}{\lambda y.f\,(x, y) : B \to C}\to \mathsf{I}^y}{\lambda x.\lambda y.f\,(x, y) : A \to B \to C}\to \mathsf{I}^x}{\lambda f.\lambda x.\lambda y.f\,(x, y) : ((A \times B) \to C) \to A \to B \to C}\to \mathsf{I}^f$$

So the Import-Export equivalence is... **the curry-uncurry isomorphism**!

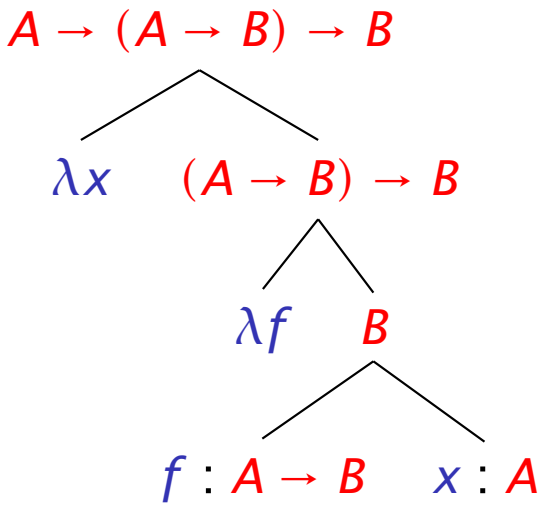
Terms *as* proofs

Flip the proof tree upside down, then convert it into a λ-term (given as a tree here to make the connection pop). The result's well-typedness is *itself* a proof of the corresponding theorem in minimal logic!

$$\dfrac{\dfrac{\dfrac{}{\lambda x.\lambda f.f\,x : A \to (A \to B) \to B}\,{\to}\mathsf{I}^{x}}{\lambda f.f\,x : (A \to B) \to B}\,{\to}\mathsf{I}^{f}}{f\,x : B}\,{\to}\mathsf{E}$$

$$[f : A \to B]^{f} \qquad [x : A]^{x}$$

$\Longrightarrow$



Stepping back

The Curry-Howard isomorphism can be summed up thusly:

Proofs are programs, and programs are proofs.

Is there a correspondent of proof normalization? One of the beautiful things about the Curry-Howard isomorphism is that *we can just check*.

Our earlier example, then with labeling

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \times A]^p}{A}\times\mathsf{E}_2 \quad \dfrac{[B \times A]^p}{B}\times\mathsf{E}_1}{A \times B}\times\mathsf{I}}{(B \times A) \to (A \times B)}\to\mathsf{I}^p \quad \dfrac{B \quad A}{B \times A}\times\mathsf{I}}{A \times B}\to\mathsf{E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{B \quad A}{B \times A}\times\mathsf{I}}{A}\times\mathsf{E}_2 \quad \dfrac{\dfrac{B \quad A}{B \times A}\times\mathsf{I}}{B}\times\mathsf{E}_1}{A \times B}\times\mathsf{I}$$

$$\Downarrow$$

$$\dfrac{A \quad B}{A \times B}\times\mathsf{I}$$

[Example still borrowed from Wadler (2015)]

Our earlier example, then with labeling

$$\dfrac{\dfrac{\dfrac{\dfrac{[p : B \times A]^p}{\mathbf{fst}\, p : A}\times \mathsf{E}_2 \qquad \dfrac{[p : B \times A]^p}{\mathbf{snd}\, p : A}\times \mathsf{E}_1}{(\mathbf{snd}\, p, \mathbf{fst}\, p) : A \times B}\times \mathsf{I}}{\lambda p.\, (\mathbf{snd}\, p, \mathbf{fst}\, p) : (B \times A) \to (A \times B)}\to \mathsf{I}^x \qquad \dfrac{M : B \qquad N : A}{(M, N) : B \times A}\times \mathsf{I}}{(\lambda p.\, (\mathbf{snd}\, p, \mathbf{fst}\, p))\, (M, N) : A \times B}\to \mathsf{E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{M : B \qquad N : A}{(M, N) : B \times A}\times \mathsf{I}}{\mathbf{snd}\, (M, N) : A}\times \mathsf{E}_2 \qquad \dfrac{\dfrac{M : B \qquad N : A}{(M, N) : B \times A}\times \mathsf{I}}{\mathbf{fst}\, (M, N) : B}\times \mathsf{E}_1}{(\mathbf{snd}\, (M, N), \mathbf{fst}\, (M, N)) : A \times B}\times \mathsf{I}$$

$$\Downarrow$$

$$\dfrac{N : A \qquad M : B}{(N, M) : A \times B}\times \mathsf{I}$$

[Example still borrowed from Wadler (2015)]

In a slogan

The Curry-Howard isomorphism tells us that proofs are programs, and vice versa. What does it tell us about proof normalization?

In a slogan

The Curry-Howard isomorphism tells us that proofs are programs, and vice versa. What does it tell us about proof normalization?

Normalization corresponds to **term simplification**, or, **evaluation**.

Test

$$\dfrac{\dfrac{\dfrac{\dfrac{[p : B \times A]^p}{\mathbf{fst}\, p : A}\times\mathrm{E}_2 \qquad \dfrac{[p : B \times A]^p}{\mathbf{snd}\, p : A}\times\mathrm{E}_1}{(\mathbf{snd}\, p, \mathbf{fst}\, p) : A \times B}\times\mathrm{I}}{\lambda p.\,(\mathbf{snd}\, p, \mathbf{fst}\, p) : (B \times A) \to (A \times B)}\to\mathrm{I}^x \qquad \dfrac{M : B \qquad N : A}{(M, N) : B \times A}\times\mathrm{I}}{(\lambda p.\,(\mathbf{snd}\, p, \mathbf{fst}\, p))\,(M, N) : A \times B}\to\mathrm{E}$$

$\Downarrow$ β-reduction

$$\dfrac{\dfrac{\dfrac{M : B \qquad N : A}{(M, N) : B \times A}\times\mathrm{I}}{\mathbf{snd}\,(M, N) : A}\times\mathrm{E}_2 \qquad \dfrac{\dfrac{M : B \qquad N : A}{(M, N) : B \times A}\times\mathrm{I}}{\mathbf{fst}\,(M, N) : B}\times\mathrm{E}_1}{(\mathbf{snd}\,(M, N), \mathbf{fst}\,(M, N)) : A \times B}\times\mathrm{I}$$

$\Downarrow$ definitions of **fst**, **snd**

$$\dfrac{N : A \qquad M : B}{(N, M) : A \times B}\times\mathrm{I}$$

Normalization as evaluation

Eliminating an implicational detour is the same thing as... β-reduction:

$$\dfrac{\dfrac{\begin{array}{c}[x : A]^x \\ \vdots \\ M[x] : B\end{array}}{\lambda x.M[x] : A \to B}\,{\to}\mathsf{I}^x \qquad \begin{array}{c}\vdots \\ N : A\end{array}}{(\lambda x.M[x])\,N : B}\,{\to}\mathsf{E} \quad \Longrightarrow \quad \begin{array}{c}\vdots \\ N : A \\ \vdots \\ M[N] : B\end{array}$$

And eliminating a conjunction detour defines the **fst** (or **snd**) projection:

$$\dfrac{\dfrac{\begin{array}{cc}\vdots & \vdots \\ M : A & N : B\end{array}}{(M, N) : A \times B}\,{\times}\mathsf{I}}{\mathbf{fst}\,(M, N) : A}\,{\times}\mathsf{E}_1 \quad \Longrightarrow \quad \begin{array}{c}\vdots \\ M : A\end{array}$$

Monads

Other logics, other notions of computation

The Curry-Howard isomorphism tells us that logics determine programming languages, and vice versa.

Thus, computational systems other than the simply typed λ-calculus may correspond to logics different from the minimal propositional calculus.

And vice versa! If you have a logic in hand, you may be able to find a notion of computation that it characterizes.

A fruitful history

- Axiomatic systems (**SK** calculus) = combinatory logic
- Second-order prop. logic ($\forall p \ldots$) = polymorphic λ-calculus
- First-order logic ($\forall x \ldots$) = dependent types
- The list goes on. We'll look at a couple more connections today.

A logic for monads

Benton, Bierman & de Paiva (1998):

$$\frac{M : A}{\eta\, M : \Diamond A}\ \Diamond\mathrm{I} \qquad \frac{M : \Diamond A \qquad F : A \to \Diamond B}{M \gg\!= F : \Diamond B}\ \Diamond\mathrm{E}$$

A proof that $(\Diamond A \times \Diamond B) \to \Diamond(A \times B)$

$$\dfrac{\dfrac{[\Diamond A \times \Diamond B]^r}{\Diamond A}\times\mathsf{E}_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[A]^p \quad [B]^q}{A \times B}\times\mathsf{I}}{\Diamond(A \times B)}\Diamond\mathsf{I}}{B \to \Diamond(A \times B)}\to\mathsf{I}^q \qquad \dfrac{[\Diamond A \times \Diamond B]^r}{\Diamond B}\times\mathsf{E}_2}{\Diamond(A \times B)}\Diamond\mathsf{E}}{A \to \Diamond(A \times B)}\to\mathsf{I}^p}{\Diamond(A \times B)}\Diamond\mathsf{E}}{(\Diamond A \times \Diamond B) \to \Diamond(A \times B)}\to\mathsf{I}^r$$

This term ends up labeled as follows (cf. donkey anaphora!):

$$\lambda(m, n).\, m \gg\!= \lambda p.\, n \gg\!= \lambda q.\, \eta\,(p, q)$$

Modal logic?

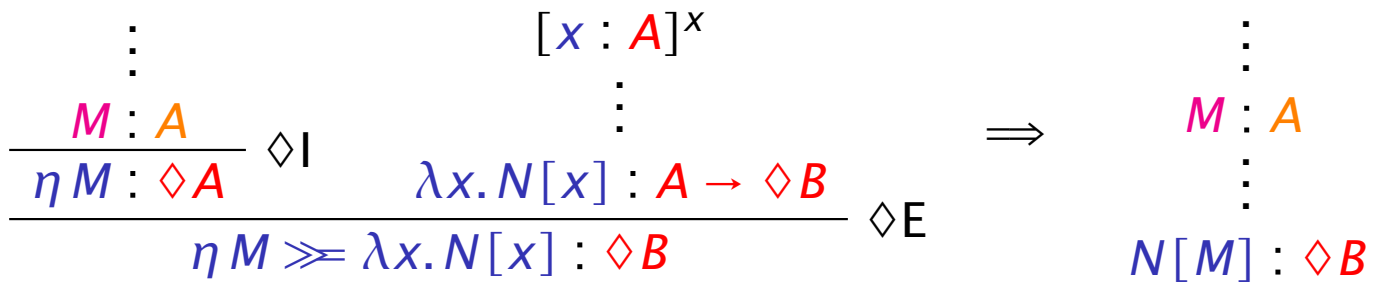
On the other hand, $(\Diamond A \times \Diamond B) \to \Diamond(A \times B)$ is a rather strange theorem for a "modal" logic. Looks rather more like a necessity modality.

Though the modality does correspond in certain ways to possibility:

- $\nvdash \Diamond p \to p$
- $\vdash p \to \Diamond p$
- $\vdash \Diamond\Diamond p \to \Diamond p$
- $\vdash \Diamond p \to \Diamond\Diamond p$

Normalization

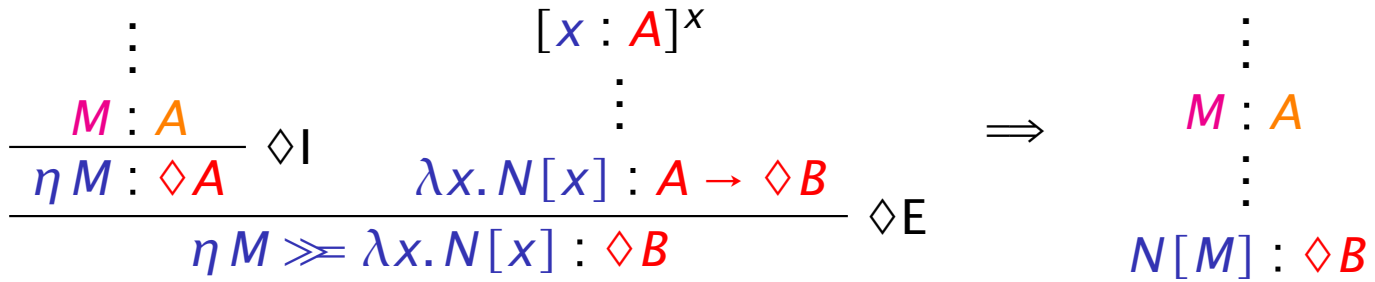
Here is a detour, an I-rule immediately followed by an E:



Does this remind you of anything? (LOL.)

Normalization

Here is a detour, an I-rule immediately followed by an E:



Does this remind you of anything? (LOL.) Yes! It corresponds exactly to Left Identity, one of the monad laws:

$$\eta\, x \gg\!= f = f\, x$$

Intuitionistic logic

Intuitionistic logic

A word on negation. Given a special formula $\bot$ representing falsity, the following two inference rules seem to completely characterize negation:

$$\dfrac{\begin{array}{c}[A]^x \\ \vdots \\ \bot\end{array}}{\neg A}\ \neg\mathrm{I}^x \qquad \dfrac{\neg A}{A \supset \bot}\ \neg\mathrm{E}$$

As a result, $\neg A$ is usually taken to simply abbreviate $A \supset \bot$.

EFQ

On top of this, **intuitionistic** logic adds to minimal logic the following inference rule, known as EFQ: anything at all follows from a contradiction.

$$\frac{\bot}{A}\ \bot\mathrm{E}$$

Intuitively, a proof of $\bot$ can always be transformed into a proof of any A, because there are no proofs of $\bot$ in the first place!

The computational content of EFQ?

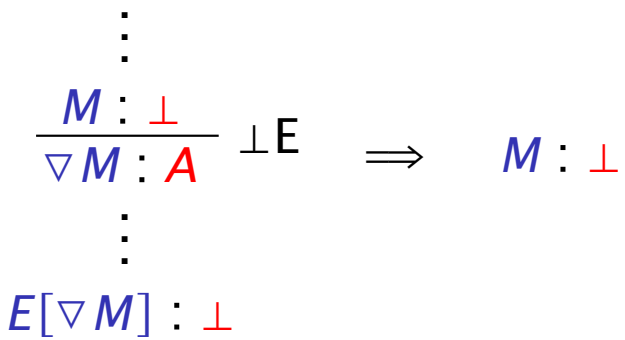
$\neg A \equiv A \supset \bot$ and EFQ allow for a limited amount of *reductio* reasoning. . .

But what sorts of **computations** do these inferences correspond to?

$$\frac{Z : \bot}{\triangledown Z : A}\ \bot\mathrm{E}$$

Normalizing EFQ

The simplest possible EFQ-related detour:



Normalizing EFQ

The simplest possible EFQ-related detour:

$$\dfrac{\dfrac{\vdots}{M : \bot}}{\dfrac{\nabla M : A}{\vdots}\ \bot\mathrm{E}}\ \bot\mathrm{E} \quad \Longrightarrow \quad M : \bot$$
$$E[\nabla M] : \bot$$

The general case, for abitrary "result types" O:

$$\dfrac{\dfrac{\vdots}{M : \bot}}{\dfrac{\nabla M : A}{\vdots}\ \bot\mathrm{E}}\ \bot\mathrm{E} \quad \Longrightarrow \quad \dfrac{\dfrac{\vdots}{M : \bot}}{\nabla M : O}\ \bot\mathrm{E}$$
$$E[\nabla M] : O$$

An example, *sans* proof

$$\textbf{cases}\,(\textbf{fst}\,(\nabla\, b\, c\, d))\,(\lambda u.\lambda v.\, u)\,(\lambda u.\lambda v.\, v)$$

$$\Downarrow$$

$$\nabla\, b$$

The result: as if we'd **broken out** of a computation.

Going classical

Classical logic: double-negation elimination

Adds to minimal logic the following inference rule

$$\frac{M : \neg\neg A}{\Delta M : A}\ \mathrm{DNE}$$

Note that EFQ follows from DNE:

$$\dfrac{\dfrac{\dfrac{M : \bot \qquad [f : \neg A]^f}{M : \bot}}{\lambda f.M : \neg\neg A}\ \to\mathrm{I}^f}{\Delta\,(\lambda f.M) : A}\ \mathrm{DNE}$$

- Intuitionistically: transforming A to $\bot$ suffices to establish $\neg A$
- Classically: transforming $\neg A$ to $\bot$ suffices to establish A

Normalizing DNE: The Easy Case

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta\, M : A\\ \vdots\\ E[\Delta\, M] : \bot\end{array}}\ \mathsf{DNE} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : \bot\end{array}}{\lambda x.\, E[x] : \neg A}\ {\to}\mathsf{I}^x}{M\,(\lambda x.\, E[x]) : \bot}\ {\to}\mathsf{E}$$

Following the shapes

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta M : A\\ \vdots\\ E[\Delta M] : \bot\end{array}}\ \mathrm{DNE} \quad \Longrightarrow \quad \dfrac{\begin{array}{cc}\vdots & [x : A]^x\\ & \vdots\\ M : \neg\neg A & E[x] : \bot\end{array}}{\begin{array}{cc}\multicolumn{2}{c}{M\,(\lambda x.\,E[x]) : \bot}\\ & \lambda x.\,E[x] : \neg A\end{array}}\ \begin{array}{l}\to\mathrm{I}^x\\ \to\mathrm{E}\end{array}$$

Following the shapes

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta\, M : A\\ \vdots\\ E[\Delta\, M] : \bot\end{array}}\ \mathsf{DNE} \quad\Longrightarrow\quad \dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\begin{array}{c}[x : A]^{x}\\ \vdots\\ E[x] : \bot\end{array}}{\lambda x.\, E[x] : \neg A}\ {\to}\mathsf{I}^{x}}{M\,(\lambda x.\, E[x]) : \bot}\ {\to}\mathsf{E}$$

everyone $::\ e$

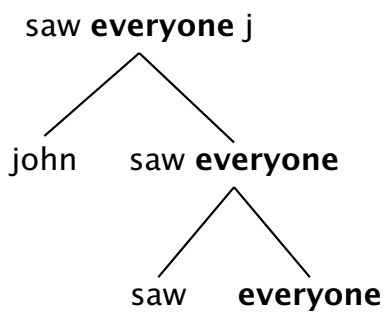
$$= \Delta\,(\lambda k.\, \forall x.\, k\, x)$$

Following the shapes

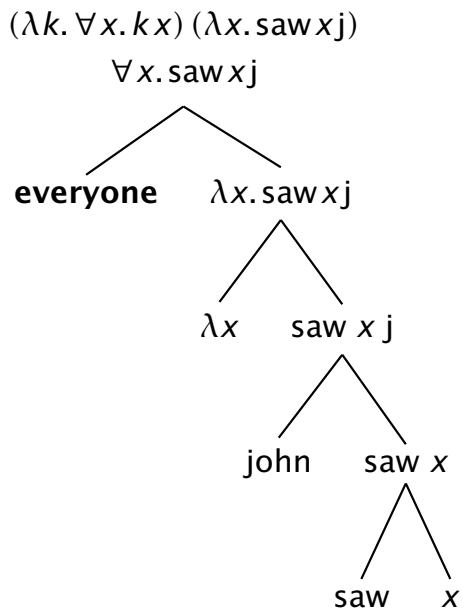
$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta\, M : A\\ \vdots\\ E[\Delta\, M] : \bot\end{array}}\ \text{DNE} \quad\Longrightarrow\quad \dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : \bot\end{array}}{\lambda x.\, E[x] : \neg A}\ \to\text{I}^x}{M\,(\lambda x.\, E[x]) : \bot}\ \to\text{E}$$

everyone $::\ e$

$$= \Delta\,(\lambda k.\, \forall x.\, k\, x)$$



$\Longrightarrow$



Normalizing DNE: The General Case

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta M : A\\ \vdots\\ E[\Delta M] : O\end{array}}\ \text{DNE} \quad \Longrightarrow \quad \begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : O\\ \\ ??? : \neg A\end{array}$$

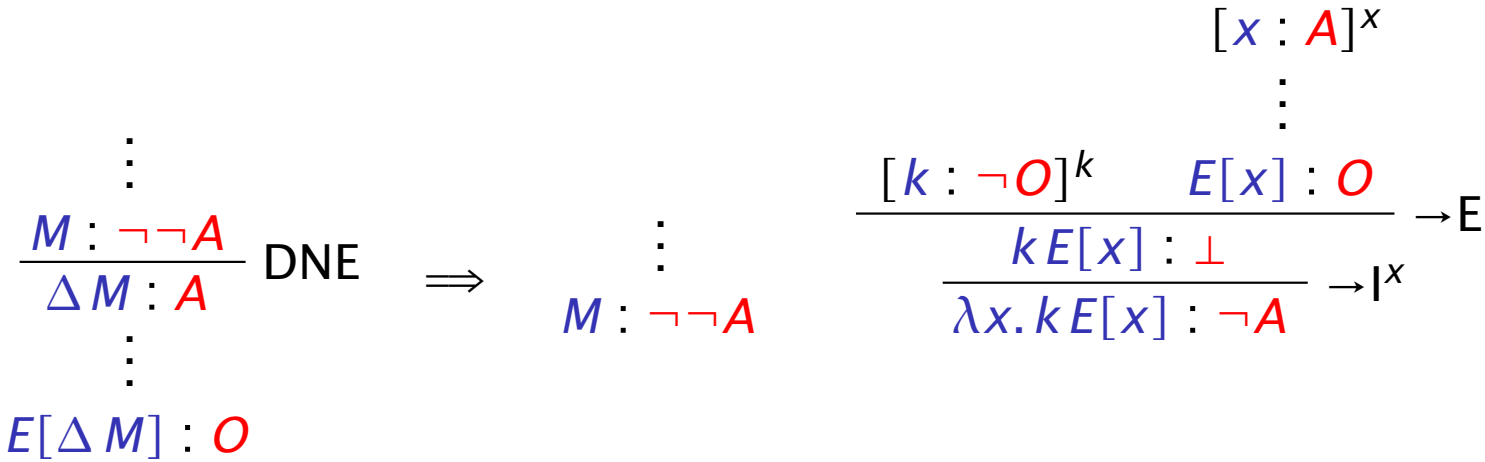
- Need something of type $\neg A$ (i.e., $A \to \bot$)

Normalizing DNE: The General Case

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta M : A\\ \vdots\\ E[\Delta M] : O\end{array}}\ \text{DNE} \quad \Longrightarrow \quad \begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : O\end{array}}{??? : \bot}}{\lambda x.??? : \neg A}\ \to\!\mathsf{I}^x$$

- Need something of type $\neg A$ (i.e., $A \to \bot$)
- We clearly have $A \to O$, so if we could turn the O into $\bot$, we could achieve $\neg A$ by abstracting over $x : A$

Normalizing DNE: The General Case



- Need something of type $\neg A$ (i.e., $A \to \bot$)
- We clearly have $A \to O$, so if we could turn the O into $\bot$, we could achieve $\neg A$ by abstracting over $x : A$
- Ok: just (temporarily) assume O is false in order to contradict O

Normalizing DNE: The General Case

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta M : A\\ \vdots\\ E[\Delta M] : O\end{array}}\ \mathsf{DNE} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\dfrac{[k : \neg O]^k \qquad \begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : O\end{array}}{k\,E[x] : \bot}\ \to\mathsf{E}}{\lambda x.\, k\,E[x] : \neg A}\ \to\mathsf{I}^x}{M\,(\lambda x.\, k\,E[x]) : \bot}\ \to\mathsf{E}$$

- Need something of type $\neg A$ (i.e., $A \to \bot$)
- We clearly have $A \to O$, so if we could turn the O into $\bot$, we could achieve $\neg A$ by abstracting over $x : A$
- Ok: just (temporarily) assume O is false in order to contradict O
- But we're not done! We need to get rid of the additional assumption $[k : \neg O]$, and turn this contradiction back into a proof of O

Normalizing DNE: The General Case

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta M : A\\ \vdots\\ E[\Delta M] : O\end{array}}\ \mathrm{DNE} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\dfrac{[k : \neg O]^k \qquad \begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : O\end{array}}{k\,E[x] : \bot}\ \to\mathrm{E}}{\lambda x.\, k\,E[x] : \neg A}\ \to\mathrm{I}^x}{M\,(\lambda x.\, k\,E[x]) : \bot}\ \to\mathrm{E}}{\lambda k.\, M\,(\lambda x.\, k\,E[x]) : \neg\neg O}\ \to\mathrm{I}^k$$

- Need something of type $\neg A$ (i.e., $A \to \bot$)
- We clearly have $A \to O$, so if we could turn the O into $\bot$, we could achieve $\neg A$ by abstracting over $x : A$
- Ok: just (temporarily) assume O is false in order to contradict O
- But we're not done! We need to get rid of the additional assumption $[k : \neg O]$, and turn this contradiction back into a proof of O

Normalizing DNE: The General Case

$$\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array}}{\begin{array}{c}\Delta\, M : A\\ \vdots\\ E[\Delta\, M] : O\end{array}}\ \text{DNE} \quad\Longrightarrow\quad \dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ M : \neg\neg A\end{array} \qquad \dfrac{\dfrac{[k : \neg O]^k \qquad \begin{array}{c}[x : A]^x\\ \vdots\\ E[x] : O\end{array}}{k\,E[x] : \bot}\ \to\text{E}}{\lambda x.\, k\,E[x] : \neg A}\ \to\text{I}^x}{M\,(\lambda x.\, k\,E[x]) : \bot}\ \to\text{E}}{\lambda k.\, M\,(\lambda x.\, k\,E[x]) : \neg\neg O}\ \to\text{I}^k}{\Delta\,(\lambda k.\, M\,(\lambda x.\, k\,E[x])) : O}\ \text{DNE}$$

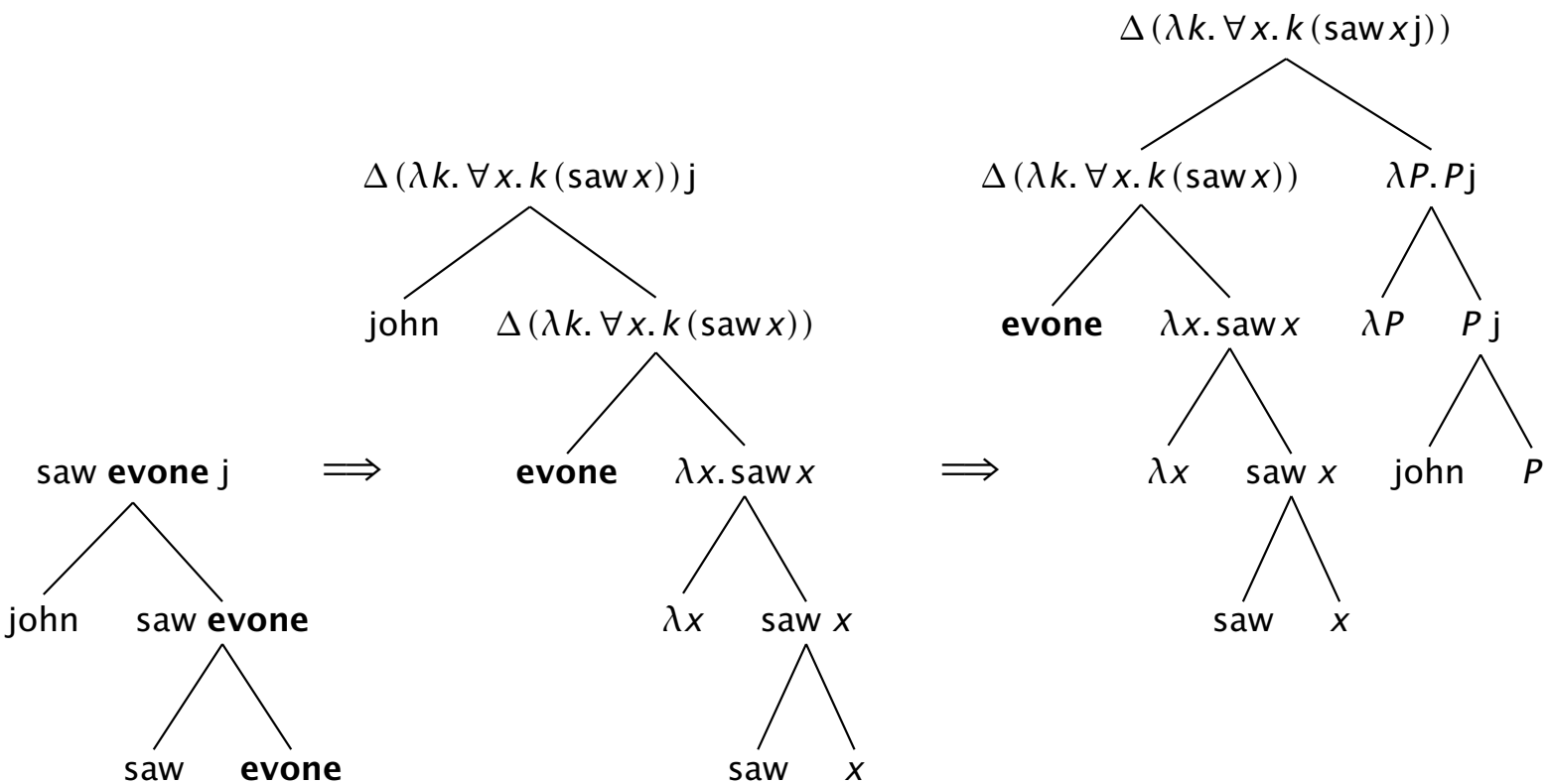
- Need something of type $\neg A$ (i.e., $A \to \bot$)
- We clearly have $A \to O$, so if we could turn the O into $\bot$, we could achieve $\neg A$ by abstracting over $x : A$
- Ok: just (temporarily) assume O is false in order to contradict O
- But we're not done! We need to get rid of the additional assumption $[k : \neg O]$, and turn this contradiction back into a proof of O

Following the shapes (cont'd)

$$E[\Delta\, M] : O \quad \Longrightarrow \quad \Delta\,(\lambda k.\, M\,(\lambda x.\, k\, E[x])) : O$$

Following the shapes (cont'd)

$$E[\Delta\, M] : O \quad \Longrightarrow \quad \Delta\,(\lambda k.\, M\,(\lambda x.\, k\, E[x])) : O$$



Denotations and CPS

DNE and QR

These operational equivalences suggest that classical inference rules have something to do with scope-taking? The heart of the connection:

$$\begin{aligned}\Delta &:: \neg\neg A \to A \\ &\equiv ((A \to \bot) \to \bot) \to A\end{aligned}$$

If we identify $\bot$ as the type of a sentence (a complete program), then DNE allows us to treat an A-type GQ as if it were a mere A!

So it must do *something* like scoping the term out over the sentence.

Semantics?

Is there a denotational semantics for the classical lambda calculus? In particular is there a $[\![\Delta]\!]$?

If there is, it can't simply be a function of type $((A \to \bot) \to \bot) \to A$, because there is no such function!

Turns out, there *is* a way to interpret $[\![\Delta]\!]$, but it means generalizing function application, in a way we're now bizarrely familiar with. . .

Gödel/Gentzen/Kolmogorov

It turns out, for every classical tautology ϕ, there is a classical tautology ψ that can be proved *without* DNE. The crux of the matter:

- $A \vee \neg A$ needs DNE: there's no way to conjure A or $\neg A$ from scratch.
- $\neg\neg(A \vee \neg A)$ is a different story. In fact, here's a DNE-less proof:

$$\lambda k.\, k\,(\mathbf{inr}\,(\lambda x.\, k\,(\mathbf{inl}\,x)))$$

And indeed, we can easily verify that this type-checks in Haskell:

```
data Void
type Not a = a -> Void

test :: Not (Not (Either a (Not a)))
test k = k (Right (\x -> k (Left x)))
```

Continuation Passing Style

Along these lines, we can in fact intuitionistically *construct* all classical propositional validities by inserting strategic double negations:

$$P^* = P$$
$$(A \to B)^* = A^* \to \neg\neg B^*$$

Double-negating our entire classical lambda calculus therefore provides a constructive interpretation. This is often called a translation into **continuation-passing style**. From Griffin (1990):

$$\begin{aligned} x^* &= \lambda k.\, k\, x \\ (\lambda x.\, M)^* &= \lambda k.\, k\, (\lambda x.\, M^*) \\ (M\, N)^* &= \lambda k.\, M^*\, (\lambda m.\, N^*\, (\lambda n.\, m\, n\, k)) \\ (\Delta M)^* &= \lambda k.\, M^*\, (\lambda m.\, m\, (\lambda z.\, \lambda d.\, k\, z)\, (\lambda x.\, x)) \end{aligned}$$

CPS Cont'd

But setting aside the computational interpretation of classical logic, the double-negation/CPS technique is flexible, with many variations.

You might recognize this one from Kuroda (1951): translate every classical propositional validity ϕ into $\psi \equiv \neg\neg\phi$. Then ψ is intuitionistically valid.

Kuroda (1951): $\phi^* = \neg\neg\phi$

Why might it look familiar? It's the *same transformation* proposed in Barker (2002) for analyzing quantifiers in terms of continuations!

> The goal of continuizing the grammar is to provide each expression with its own continuation as an argument. Therefore if an expression of category A with semantic type α has direct denotation $[\![A]\!]$, then the continuized denotation $\{\!\{A\}\!\}$ will be a function from A-continuations (type $\langle\alpha, t\rangle$) to the type of the larger expression as a whole, i.e., to a truth value: type $(\{\!\{A\}\!\}) = \langle\langle\alpha, t\rangle, t\rangle$.

(8)

Continuized Denotation	Type
$\{\!\{S\}\!\}$	$\langle\langle t, t\rangle, t\rangle$
$\{\!\{VP\}\!\}$	$\langle\langle\langle e, t\rangle, t\rangle, t\rangle$
$\{\!\{NP\}\!\}$	$\langle\langle e, t\rangle, t\rangle$
$\{\!\{Vt\}\!\}$	$\langle\langle\langle e, \langle e, t\rangle\rangle, t\rangle, t\rangle$

Barker, Chris. 2002. Continuations and the nature of quantification. *Natural Language Semantics* 10(3). 211–242. http://dx.doi.org/10.1023/A:1022183511876.

Benton, P. N., G. M. Bierman & V. C. V. de Paiva. 1998. Computational types from a logical perspective. *Journal of Functional Programming* 8(2). 177–193. http://dx.doi.org/10.1017/s0956796898002998.

Curry, H. B. 1934. Functionality in combinatory logic. *Proceedings of the National Academy of Sciences* 20(11). 585–590.

Gentzen, Gerhard. 1935. Untersuchungen über das logische schließen. *Mathematische Zeitschrift* 39(2-3). 176–210, 405–431.

Griffin, Timothy G. 1990. A formulae-as-types notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages* (POPL '90), 47–58. New York: ACM Press. http://dx.doi.org/10.1145/96709.96714.

Howard, W. A. 1980. The formulae-as-types notion of construction. In J. P. Seldin & J. R. Hindley (eds.), *To H. B. Curry: Essays on combinatory logic, lambda calculus, and formalism*, 479–491. New York: Academic Press.

Kuroda, Sigekatu. 1951. Intuitionistische untersuchungen der formalistischen logik. *Nagoya Mathematical Journal* 3. 35–47. http://dx.doi.org/10.1017/s0027763000010023.

Wadler, Philip. 2015. Propositions as types. *Communications of the ACM* 58(12). 75–84. http://dx.doi.org/10.1145/2699407.