PROOF101, Spring 2026, Lecture Slides 04

American University of Beirut

The Curry-Howard Isomorphism

Assaf Kfoury

March 2026

Outline of Lecture Slides 04

- Main reference for these lecture slides is again the book:

  Lectures on the Curry-Howard Isomorphim

  by Sorensen and Urzyczyn – henceforth denoted [LCHI] .

- Many details, but not all, are from [LCHI, Chapter 4] .

- Headings in this set of slides (different from [LCHI, Chapt 4]) :
  - *Extended λ-Calculus* , slides 3 – 8
  - *Transition*: from Strongly-Typed PL to Annotated Formal Proofs , slide 9
  - *Adding Proof Terms* to Natural Deduction , slides 10 – 14
  - *Erasing Propositions* from Natural Deduction , slides 15 – 17
  - *Proof Normalization* vs. *Term Reduction* , slides 18 – 22
  - *Coda*: pulling all the strands together , slide 23 – 24

- Contents in these slides are not organized as in [LCHI, Chapter 4] , here the initial focus is on PL principles (slides 3 – 8), with logic and proof theory gradually added after .

Extended λ-Calculus: *Syntax*

- We extend the set Λ of untyped λ-expressions in Lecture Slides 01 to include *pair* terms and *case* terms. $\Lambda^{\times,+}$ denotes the new extended set: *

$$
\begin{array}{rcl}
M, N \in \Lambda^{\times,+} & ::= & x \mid (M\,N) \mid (\lambda x.\, M) \mid \\
& & \langle M_1, M_2 \rangle \mid (\pi_1\, M) \mid (\pi_2\, M) \mid \\
& & (\mathbf{in}_1\, M) \mid (\mathbf{in}_2\, M) \mid (\mathbf{case}\ M\ \mathbf{of}\ [x_1]N_1\ \mathbf{or}\ [x_2]N_2) \mid \\
& & (\varepsilon\, M)
\end{array}
$$

  line 2 above includes a *constructor* and two *destructors* for pair terms,
  line 3 includes two *constructors* and a *destructor* for **case** terms,
  line 4 includes the *destructor* ε for the empty type .

- Later we set up a simply-typed version of $\Lambda^{\times,+}$ where:

  new types for line 2 above are binary *product* types,
  new types for line 3 above are binary *sum* (or *variant*) types,
  with the destructor ε, a term M without a type is assigned any type.

---

*This extended λ-calculus is closer to a real-world programming language, but it is still missing an important feature: A construct to carry out *recursion* or *iteration*. More on this later in the course.

Extended λ-Calculus: *Reduction Rules* (Operational Semantics)

- β-reduction is extended to $\Lambda^{\times,+}$ as the smallest compatible relation $\rightarrow_\beta$ extending ordinary β-reduction such that:

$$\pi_1 \langle M_1, M_2 \rangle \rightarrow_\beta M_1$$

$$\pi_2 \langle M_1, M_2 \rangle \rightarrow_\beta M_2$$

$$\textbf{case } (\textbf{in}_1\ M) \textbf{ of } [x_1]N_1 \textbf{ or } [x_2]N_2 \rightarrow_\beta N_1[x_1 := M]$$

$$\textbf{case } (\textbf{in}_2\ M) \textbf{ of } [x_1]N_1 \textbf{ or } [x_2]N_2 \rightarrow_\beta N_2[x_2 := M]$$

- Other notions of reduction $\{\rightarrow, \twoheadrightarrow, \Rightarrow, \ldots\}$ already presented in Lecture Slides 01, appropriately subscripted with 'β' or 'η' or both, are generalized to the extended language $\Lambda^{\times,+}$ in the obvious way .

Extended λ-Calculus: *Typing Rules* (Static Semantics)

- From Lecture Slides 03 , the set $\Phi_{\to}$ of simple types is:

$$\varphi, \psi \in \Phi_{\to} \ ::= \ p \mid \varphi \to \psi$$

where p ranges over a set PV of type variables (*i.e.*, propositional variables).

- $\Phi_{\to}$ is extended to $\Phi_{\to}^{\bot,\times,+}$ to include *product* types, *sum* types, and a special constant type $\bot$ (the type of no well-typed program) : †

$$\varphi, \psi \in \Phi_{\to}^{\bot,\times,+} \ ::= \ \bot \mid p \mid \varphi \to \psi \mid \varphi \wedge \psi \mid \varphi \vee \psi$$

---

† In books and articles dealing with *programming-language theory*, it is common to see a different notation, which also reflects the program constructs they qualify more appropriately:

$$\mathbf{0} \mid p \mid \varphi \to \psi \mid \varphi \times \psi \mid \varphi + \psi$$

where **0** is said to be the *empty* type, *i.e.*, a type (not the only one) with no inhabitant. Sometimes `void` is used instead of **0**, though `void` as used in many programming languages should not be viewed as $\bot$ here (see also footnote ¶). We stay with the symbols $\{\bot, \wedge, \vee\}$ because these are used to denote the corresponding logic notions of *absurdity*, *conjunction*, and *disjunction*.

Extended λ-Calculus: *Typing Rules* (Static Semantics)

The simply-typed *Extended λ-Calculus* *à la* Church (and similarly *à la* Curry): [‡]

(Var) $\quad \Gamma,\ x : \varphi \ \vdash\ x : \varphi$

(Abs) $\quad \dfrac{\Gamma,\ x : \varphi \ \vdash\ x : \psi}{\Gamma \ \vdash\ (\lambda x : \varphi.\ M) : \varphi \to \psi}$

(App) $\quad \dfrac{\Gamma \ \vdash\ M : \varphi \to \psi \qquad \Gamma \ \vdash\ N : \varphi}{\Gamma \ \vdash\ (M\ N) : \psi}$

(ProjL) $\quad \dfrac{\Gamma \ \vdash\ M : \varphi \wedge \psi}{\Gamma \ \vdash\ (\pi_1\ M) : \varphi}$

(ProjR) $\quad \dfrac{\Gamma \ \vdash\ M : \varphi \wedge \psi}{\Gamma \ \vdash\ (\pi_2\ M) : \psi}$

(Pair) $\quad \dfrac{\Gamma \ \vdash\ M : \varphi \qquad \Gamma \ \vdash\ N : \psi}{\Gamma \ \vdash\ \langle M, N\rangle : \varphi \wedge \psi}$

(InL) $\quad \dfrac{\Gamma \ \vdash\ M : \varphi}{\Gamma \ \vdash\ (\mathbf{in}_1^{\varphi \vee \psi}\ M) : \varphi \vee \psi}$

(InR) $\quad \dfrac{\Gamma \ \vdash\ M : \psi}{\Gamma \ \vdash\ (\mathbf{in}_2^{\varphi \vee \psi}\ M) : \varphi \vee \psi}$

(Case) $\quad \dfrac{\Gamma \ \vdash\ L : \varphi \vee \psi \qquad \Gamma,\ x : \varphi \ \vdash\ M : \theta \qquad \Gamma,\ y : \psi \ \vdash\ N : \theta}{\Gamma \ \vdash\ \mathbf{case}\ L\ \mathbf{of}\ [x]\,M\ \mathbf{or}\ [y]\,N\ :\ \theta}$

[‡] I here use rule names, (Abs), (App), (ProjL), etc., that are typically found in books on *programming-language theory*, rather than ($\to$I), ($\to$E), ($\wedge$E$_1$), etc. The latter are more common in books on *mathematical logic* and *proof theory*.

Extended λ-Calculus: *Typing Rules* (Static Semantics)

Two more typing rules for the simply-typed *Extended λ-Calculus* :

$$\text{(Top)} \quad \frac{}{\Gamma \vdash \langle\ \rangle : \top} \qquad\qquad \text{(Bot)} \quad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash (\varepsilon_{\varphi}\, M) : \varphi}$$

where:

$\langle\ \rangle$ is the empty tuple ,

$\top$ abbreviates $\bot \to \bot$ (same as type `unit` in programming-language theory) .

In theories of programming languages, both (Top) and (Bot) are problematic rules, and the second more so than the first. [§]

I include (Top) and (Bot) here to maintain a complete parallel with *intuitionistic proposional logic / calculus* .

[§] In some accounts, a rule such as (Bot) is delayed or omitted. See the discussion on `unit` (where it is written `Unit`) in Section 11.2, and again on the `Top` and `Bottom` types in Section 15.4, of the book Types and Programming Languages by B.C. Pierce . Complementary discussions on (Top) and (Bot) are found elsewhere, *e.g.*, in Chapter 5 of the book Theories of Programming Languages by J.C. Reynolds .

Extended λ-Calculus: *Normalization and Confluence*

- Basic properties of the simply-typed *λ-Calculus* (review their formulations in Lecture Slides 03) are enjoyed by the simply-typed *Extended λ-Calculus*, in particular :
  - ***Subjet Reduction / Type Preservation*** ,
  - ***Progress*** ,
  - ***Confluence / Church-Rosser Property*** ,
  - ***Strong Normalization*** .
- *Church-Rosser* and *Strong Normalization* are discussed at the end of [LCHI, Sect 4.5, p 88] , with proofs delayed to later sections.
- *Subject Reduction* and *Progress* are not discussed in [LCHI, Chapt 4] , though they are implicitly part of the discussion on *normalization* in [LCHI, Sects 4.4, 4.5] .

Transition: from Strongly-Typed PL *to* Annotated Formal Proofs

- Preceding slides 3 – 8 presented a strongly-typed programming language:

  Every intermediate program phrase is assigned a type (i.e., proposition).

- Succeeding slides 10 – 14 shift to formal proofs and how to annotate them:

  Every intermediate proposition (i.e., type) is assigned a proof term.

- STOP HERE:

  Keep handy the *unannotated proof rules* in Lecture Slides 02 , slides 7–12, you will need them as a reminder and for purposes of comparison.

- NOW READ ON:

  All added annotations in succeeding slides are highlighted like this .

Adding Proof Terms to Natural Deduction

for $\wedge$

$$\frac{M : \varphi \qquad N : \psi}{\langle M, N \rangle : \varphi \wedge \psi}\,(\wedge\mathbf{I}) \qquad \frac{M : \varphi \wedge \psi}{(\pi_1\, M) : \varphi}\,(\wedge\mathbf{E}_1) \qquad \frac{M : \varphi \wedge \psi}{(\pi_2\, M) : \psi}\,(\wedge\mathbf{E}_2)$$

for $\vee$

$$\frac{M : \varphi}{(\mathbf{in}_1^{\varphi \vee \psi}\, M) : \varphi \vee \psi}\,(\vee\mathbf{I}_1) \qquad \frac{M : \psi}{(\mathbf{in}_2^{\varphi \vee \psi}\, M) : \varphi \vee \psi}\,(\vee\mathbf{I}_2)$$

$$\frac{L : \varphi \vee \psi \qquad \boxed{\begin{array}{c} x : \varphi \\ \vdots \\ M : \theta \end{array}} \qquad \boxed{\begin{array}{c} y : \psi \\ \vdots \\ N : \theta \end{array}}}{(\mathbf{case}\ L\ \mathbf{of}\ [x]\, M\ \mathbf{or}\ [y]\, N) : \theta}\,(\vee\mathbf{E})$$

for $\rightarrow$

$$\frac{\boxed{\begin{array}{c} x : \varphi \\ \vdots \\ M : \psi \end{array}}}{(\lambda x : \varphi.\, M) : \varphi \rightarrow \psi}\,(\rightarrow\mathbf{I}) \qquad \frac{M : \varphi \rightarrow \psi \qquad N : \varphi}{(M\, N) : \psi}\,(\rightarrow\mathbf{E})$$

Adding Proof Terms to Natural Deduction

- ***for*** $\bot$, only one annotated proof rule $\dfrac{M : \bot}{(\varepsilon_\varphi\, M) : \varphi}$ $(\bot\mathbf{E})$
  - '$(\varepsilon_\varphi\, M)$' is the notation used in [LCHI, p 87], called *a miracle of type* φ.
  - In some accounts they write $\dfrac{M : \bot}{(\mathbf{abort}_\varphi\, M) : \varphi}$ or even $\dfrac{M : \bot}{M : \varphi}$
- ***for*** $\top$, with $\top \triangleq \bot \to \bot$, only one annotated proof rule $\dfrac{}{\langle\ \rangle : \top}$ $(\top\mathbf{I})$
  - '$\langle\ \rangle$' is the empty tuple, the only constant of type $\top$.
  - '$\top$' here is the same as `unit` in strongly-typed programming languages, in Haskell it is written `()`, in Scala it is `Unit`, in Python it is `NoneType`, etc. [¶]

[¶] See the Wikipedia page Unit Type for a comprehensive list. Another special type is `void`, which is however understood differently in different languages : Is `void` the same as $\bot$ here? Not always; see the Wikipedia page Void Type . Staying within a pure functional setting, it is reasonable to take `void` to be $\bot$, a type (not the only one) which has no inhabitants.

Adding Proof Terms to Natural Deduction

- ***for*** $\neg$, two (dual) derived proof rules, $(\neg\mathbf{I})$ + $(\neg\mathbf{E})$:

$$\dfrac{\boxed{\begin{array}{c} x : \varphi \\ \vdots \\ M : \bot \end{array}}}{(\lambda x : \varphi.\, M) : \neg\varphi}\ (\neg\mathbf{I}) \qquad \dfrac{M : \neg\varphi \qquad N : \varphi}{(M\,N) : \bot}\ (\neg\mathbf{E})$$

or in the style of [LCHI], where '$\vdash$' is now used as a formal symbol (separating the two parts of a judgement):

$$\dfrac{\Gamma,\ x : \varphi \vdash M : \bot}{\Gamma \vdash (\lambda x : \varphi.\, M) : \neg\varphi}\ (\neg\mathbf{I}) \qquad \dfrac{\Gamma \vdash M : \neg\varphi \qquad \Gamma \vdash N : \varphi}{\Gamma \vdash (M\,N) : \bot}\ (\neg\mathbf{E})$$

Note that $(\neg\mathbf{I})$ and $(\neg\mathbf{E})$ are special cases of $(\to\mathbf{I})$ and $(\to\mathbf{E})$ when ψ is $\bot$ in the latter two rules .

Adding Proof Terms to Natural Deduction

- for the alternative rule $(\vee\mathbf{E}^*)$ for *disjunction elimination* found in some accounts of IPC: [‖]

$$\dfrac{\begin{array}{cc}\boxed{\begin{array}{l}x:\varphi\\ \vdots\\ M:\theta\end{array}} & \boxed{\begin{array}{l}y:\psi\\ \vdots\\ N:\theta\end{array}}\end{array}}{[\lambda x:\varphi.\,M,\ \lambda y:\psi.\,N]:(\varphi\vee\psi)\to\theta}\ (\vee\mathbf{E}^*)$$

[‖] We name it $(\vee\mathrm{E}^*)$ with a superscript '$*$' to distinguish it from the more standard rule $(\vee\mathrm{E})$.

Adding Proof Terms to Natural Deduction

- As pointed out in Lecture Slides 02 , slide 12, ***classical propositional logic / calculus*** (CPC) is obtained by adding any of the following four rules:

$$\frac{}{\varphi \vee \neg\varphi}\ (\textbf{LEM}) \qquad \frac{\begin{array}{|c|}\hline \neg\varphi \\ \vdots \\ \bot \\ \hline\end{array}}{\varphi}\ (\textbf{PBC})$$

$$\frac{\neg\neg\varphi}{\varphi}\ (\neg\neg\textbf{E}) \qquad \frac{}{((\varphi \to \psi) \to \varphi) \to \varphi}\ (\textbf{Peirce's})$$

- Is it possible to annotate natural-deduction proofs that use these four rules with proof terms from the *Extended λ-Calculus* in order to establish a *Curry-Howard Isomorphism* with CPC? Answer: No, it is not possible!
- But it is possible if we extend the *Extended λ-Calculus* further (to something called the *$\lambda\mu$-Calculus*) in order to have appropriate proof terms at our disposal, which can then be used to annotate natural-deduction proofs in CPC and thus satisfy a *Curry-Howard Isomorphism* for CPC .

  This further extension to the *$\lambda\mu$-Calculus* is taken up in [LCHI, Chapter 6] .

Erasing Propositions from Annotated Natural Deduction

- In slides 10 – 14 we added *proof terms* (*i.e.*, *λ-expressions* or *untyped programs*) to natural-deduction proofs of *propositions* (*i.e.*, *types*).
- It is instructive to complement this process by erasing *propositions/types* from annotated natural deductions in order to obtain the formation rules of *proof terms/untyped programs*, shown on the next slide 16 .
- If Γ is an environment of type assumptions, the *domain* of Γ is: $\mathrm{dom}(\Gamma) \triangleq \{ x \mid (x : \tau) \in \Gamma \}$.

  In the formation rules on slide 16 , $\Delta = \mathrm{dom}(\Gamma)$ is a set of variables without their type attributes . **

** Δ is like the "symbol table" in a compiler, where all user-defined identifiers occurring in the program being compiled are stored, here without any attributes .

Erasing Propositions from Annotated Natural Deduction

- Below are the rules in [LCHI, Fig 4.1, p 88] after erasing all propositions / types (we can do the same with our style of rules in slides 10 – 11):

$$\frac{\Delta \vdash M \qquad \Delta \vdash N}{\Delta \vdash \langle M, N \rangle}$$

$$\frac{\Delta \vdash M}{\Delta \vdash (\pi_1\, M)}$$

$$\frac{\Delta \vdash M}{\Delta \vdash (\pi_2\, M)}$$

$$\frac{\Delta \vdash M}{\Delta \vdash (\mathbf{in}_1\, M)}$$

$$\frac{\Delta \vdash M}{\Delta \vdash (\mathbf{in}_2\, M)}$$

$$\frac{\Delta \vdash L \qquad \Delta, x \vdash M \qquad \Delta, y \vdash N}{\Delta \vdash (\mathbf{case}\ L\ \mathbf{of}\ [x]\, M\ \mathbf{or}\ [y]\, N)}$$

$$\frac{\Delta, x \vdash M}{\Delta \vdash (\lambda x.\, M)}$$

$$\frac{\Delta \vdash M \qquad \Delta \vdash N}{\Delta \vdash (M\, N)}$$

Erasing Propositions from Annotated Natural Deduction

- By erasing propositions / types from rules $(\bot\mathbf{E})$ and $(\top\mathbf{I})$ on slide 11 , we obtain two formation rules for *untyped λ-expressions* :

$$\frac{\Delta \vdash M}{\Delta \vdash (\varepsilon\ M)} \qquad \left(\text{or also } \frac{\Delta \vdash M}{\Delta \vdash (\mathbf{abort}\ M)}\right)$$

$$\frac{}{\Delta \vdash \langle\ \rangle}$$

- Erasing propositions/types from $(\neg\mathbf{I})$ and $(\neg\mathbf{E})$ on slide 12 produces the same formation rules as erasing them from $(\rightarrow\mathbf{I})$ and $(\rightarrow\mathbf{E})$ on slide 10.
- Erasing propositions / types from the alternative $(\vee\mathbf{E}^*)$ on slide 13 :

$$\frac{\Delta, x \vdash M \qquad \Delta, y \vdash N}{\Delta \vdash [\lambda x.M, \lambda y.N]}$$

Proof Normalization vs. Term Reduction

- Counterpart of *reduction of λ-terms / evaluation of programs* is:

  normalization of formal proofs ,

  involving operations on natural-deduction proofs under such rubrics as *detour elimination*, *cut elimination*, *proof inlining*, *proof substitution*, etc.,

  and the goal of which is to produce natural deductions in *normal form*.

- Existence of *λ-terms in normal form / outputs of terminating programs / irreducible λ-terms* $=$ existence of natural deductions in *normal form*.

- **EXERCISE.** We know that *λ-terms in normal form* are the values returned by programs. What is the point of the existence of *formal proofs in normal form*?
  Hint: Consult the webpage The Development of Proof Theory .

- [LCHI, Chapt 4] avoids a detailed discussion of operations on formal proofs, relying instead on *Curry-Howard Isomorphism* for their existence, as we already know that every *typed λ-term* is normalizing.

- For completeness here, we show schematically rules for normalizing natural deductions on slides 19 – 22 . We leave it to you to look up discussions on *cut elimination* and *detour elimination* on the Web.

Proof Normalization vs. Term Reduction

- Conventions and nomenclature for the rules on slides 20 – 22 :
  - Each rule is presented twice:
    first unannotated and *second* annotated with proof terms .
  - The script letters $\mathscr{D}$ and $\mathscr{E}$ denote natural deductions in the form of *proof trees* (not in the form of *linear numbered sequences*) .

    For examples showing differences between the two forms, with advantages and disadvantages of both, see the handout Curry-Howard Examples .
  - If $\mathscr{D}$ and $\mathscr{E}$ are unannotated proof trees, then $\widetilde{\mathscr{D}}$ and $\widetilde{\mathscr{E}}$ are their annotated versions .
  - Each rule is intended to eliminate a pair $(\#\mathsf{I}) + (\#\mathsf{E})$ from a *proof tree* (not a *linear numbered sequence*) where $\# \in \{\rightarrow, \wedge, \vee\}$.

Proof Normalization vs. Term Reduction

Rule $(\to\mathbf{Cut})$ for eliminating one occurrence of $(\to\mathbf{I}) + (\to\mathbf{E})$ in a proof tree:

$$\dfrac{\dfrac{\mathscr{D}}{\Gamma \vdash \varphi} \qquad \dfrac{\dfrac{\mathscr{E}}{\Gamma, \varphi \vdash \psi}}{\Gamma \vdash \varphi \to \psi}\,(\to\mathsf{I})}{\Gamma \vdash \psi}\,(\to\mathsf{E}) \quad \Longrightarrow \quad \begin{array}{c}\mathscr{D}\\ \Gamma \vdash \varphi\\ \mathscr{E}\\ \Gamma \vdash \psi\end{array}$$

After annotation with proof terms:

$$\dfrac{\dfrac{\widetilde{\mathscr{D}}}{\widetilde{\Gamma} \vdash M : \varphi} \qquad \dfrac{\dfrac{\widetilde{\mathscr{E}}}{\widetilde{\Gamma}, x : \varphi \vdash N : \psi}}{\widetilde{\Gamma} \vdash (\lambda x : \varphi.\, N) : \varphi \to \psi}\,(\to\mathsf{I})}{\widetilde{\Gamma} \vdash (\lambda x : \varphi.\, N)M : \psi}\,(\to\mathsf{E}) \quad \Longrightarrow \quad \begin{array}{c}\widetilde{\mathscr{D}}\\ \widetilde{\Gamma} \vdash M : \varphi\\ \widetilde{\mathscr{E}}\\ \widetilde{\Gamma} \vdash N[x := M] : \psi\end{array}$$

- In the annotated rule, the proof term $N[x := M]$ has to be justified (left to you!) .
- This transformation on *proof trees* clearly corresponds to the reduction of a β-redex, namely: $(\lambda x : \varphi.\, N)\, M \to_\beta N[x := M]$.

Proof Normalization vs. Term Reduction

Rule $(\wedge\mathbf{Cut}_1)$ for eliminating one occurrence of $(\wedge\mathbf{I}) + (\wedge\mathbf{E}_1)$ in a proof tree:

$$\dfrac{\dfrac{\begin{array}{cc}\mathscr{D} & \mathscr{E} \\ \Gamma \vdash \varphi & \Gamma \vdash \psi\end{array}}{\Gamma \vdash \varphi \wedge \psi}\,(\wedge\mathrm{I})}{\Gamma \vdash \varphi}\,(\wedge\mathrm{E}_1) \quad \Longrightarrow \quad \begin{array}{c}\mathscr{D} \\ \Gamma \vdash \varphi\end{array}$$

And similarly $(\wedge\mathbf{Cut}_2)$ for eliminating one occurrence of the pair $(\wedge\mathbf{I}) + (\wedge\mathbf{E}_2)$. After annotation with proof terms, rule $(\wedge\mathbf{Cut}_1)$ becomes:

$$\dfrac{\dfrac{\begin{array}{cc}\widetilde{\mathscr{D}} & \widetilde{\mathscr{E}} \\ \widetilde{\Gamma} \vdash M : \varphi & \widetilde{\Gamma} \vdash N : \psi\end{array}}{\widetilde{\Gamma} \vdash \langle M, N\rangle : \varphi \wedge \psi}\,(\wedge\mathrm{I})}{\widetilde{\Gamma} \vdash \pi_1\langle M, N\rangle : \varphi}\,(\wedge\mathrm{E}_1) \quad \Longrightarrow \quad \begin{array}{c}\widetilde{\mathscr{D}} \\ \widetilde{\Gamma} \vdash M : \varphi\end{array}$$

- This transformation on *proof trees* corresponds to the reduction of a β-redex, namely: $\pi_1 \langle M, N\rangle \rightarrow_\beta M$.

Proof Normalization vs. Term Reduction

EXERCISE.

Write the rule $(\vee\mathbf{Cut}_1)$ for eliminating one occurrence of the pair $(\vee\mathbf{I}_1)+(\vee\mathbf{E})$, and similarly $(\vee\mathbf{Cut}_2)$ for one occurrence of the pair $(\vee\mathbf{I}_2)+(\vee\mathbf{E})$, in proof trees, *without* and *with* annotation with proof terms.

Draw a correspondence with the β-reduction of λ-terms containing **case**-subterms .

Hint: Make sure you understand the rule for eliminating one occurrence of the $(\rightarrow\mathbf{I})+(\rightarrow\mathbf{E})$ on slide 20 before you try this exercise .

EXERCISE.

Write an inductive definition (*e.g.*, in a BNF grammar similar to that on slide 3) of *terms in normal form* in the *Extended λ-Calculus*. If you get stuck, search the Web.

Write an inductive definition of *natural deductions in normal form* (as unannotated proof trees, to simplify a little). If you get stuck, again search the Web.

Define a correspondence between the two normal forms: *extended λ-terms* and *unannotated proof trees*, as a bijection between the two.

Hint: Limit attention to *closed* extended λ-terms (*i.e.*, terms without free variables), and to *tautologies* of IPC (*i.e.*, proof trees where all axioms/premises are discharged).

Coda: Pulling all the strands together

Several theorems connect the different pieces presented in this set of slides. We mention the most significant below. Recall the acronyms:

$\mathrm{IPC}(\to)$ = *intuitionistic implicational PC*,

$\mathrm{IPC}(\to, \bot, \wedge, \vee) = \mathrm{IPC}$ = full *intuitionistic PC*,

$\mathrm{NJ}(\to)$ = natural-deduction *proof system* for $\mathrm{IPC}(\to)$,

$\mathrm{NJ}(\to, \bot, \wedge, \vee)$ = natural-deduction *proof system* for $\mathrm{IPC} = \mathrm{IPC}(\to, \bot, \wedge, \vee)$,

$\lambda_{\to}$ = the *simply-typed λ-Calculus*,

$\lambda_{\to}^{\bot,\times,+}$ = the *simply-typed Extended λ-Calculus* .

- **THEOREM** (*Curry-Howard Isomorphism for* $\mathrm{IPC}(\to)$) :
  1. *If* $\Gamma \vdash M : \varphi$ *in* $\lambda_{\to}$, *then* $\mathrm{range}(\Gamma) \vdash \varphi$ *in* $\mathrm{IPC}(\to)$.
  2. *If* $\Delta \vdash \varphi$ *in* $\mathrm{IPC}(\to)$, *then* $\Gamma \vdash M : \varphi$ *in* $\lambda_{\to}$ *for some* M *and some* Δ *with* $\mathrm{range}(\Gamma) = \Delta$.

  This is [LCHI, Prop 4.1.1, p 77] . A nice consequence is the proof of the next.

- **COROLLARY** ($\mathrm{NJ}(\to)$ *Is Consistent*) : *Not all judgements of* $\mathrm{IPC}(\to)$ *are derivable by* $\mathrm{NJ}(\to)$. A short elegant proof is in [LCHI, Prop 4.1.2, p 78] .

Coda: Pulling all the strands together

The next result is more general than the theorem on the preceding slide :

- **THEOREM** (*Curry-Howard Isomorphism for* IPC) :
  1. *If* $\Gamma \vdash M : \varphi$ *in* $\lambda_{\rightarrow}^{\perp,\times,+}$, *then* $\mathrm{range}(\Gamma) \vdash \varphi$ *in* IPC .
  2. *If* $\Delta \vdash \varphi$ *in* IPC, *then* $\Gamma \vdash M : \varphi$ *in* $\lambda_{\rightarrow}^{\perp,\times,+}$ *for some* M *and some* Δ *with* $\mathrm{range}(\Gamma) = \Delta$.

  This is [LCHI, Prop 4.5.3, p 87] .
- **EXERCISE** : Show that IPC is consistent, as a corollary of the preceding theorem.

  Hint: First understand how the corollary on slide 23 is a consequence of the CH Isomorphism for IPC($\rightarrow$).
- **THEOREM** :
  1. *Every derivable judgement of* IPC($\rightarrow$) *has a formal proof in normal form.*
  2. *Every derivable judgement of* IPC *has a formal proof in normal form.*

  The first part is stated and proved in [LCHI, Prop 4.4.1, p 85] , the second part is proved in the same way, though not done in the book .

(THIS SLIDE INTENTIONALLY LEFT BLANK)