Lecture Notes 8
Curry–Howard

Carlo Angiuli

B522: PL Foundations
March 23, 2026

In this lecture, we briefly pause our discussion of typed λ-calculi to consider the seemingly unrelated topic of propositional logic and its *verificationistic theory of meaning*, which to varying degrees of precision is also known as the *Brouwer–Heyting–Kolomogorov (BHK) interpretation*, the *Curry–Howard correspondence/isomorphism*, the *propositions-as-types correspondence*, the *proofs-as-programs correspondence*, and the *meaning explanations of type theory*. Along the way, we will also discover the origin of the notion of *judgment* that we have used all semester.

The latter parts of these notes are covered in Chapter 12 of Harper [Har16]; Section 1 is based on the philosophical writings of Martin-Löf [Mar87]. We will resume studying programming languages in the next lecture.

1 The verificationistic theory of meaning

How do we define *logic*? Standard introductions to mathematical logic typically define classical propositional and predicate logic with machinery such as well-formed formulas, truth tables (in the case of propositional logic), and first-order structures (in the case of predicate logic). The result is a mathematically precise definition of these logics as formal systems, in much the same way that abstract binding trees and type systems allow us to formally define programming languages.

However, one might reasonably object that it is wrongheaded to treat logic in this way; after all, any mathematical explanation of logic must be conducted with respect to some ambient logic, and thus can only reduce one logic to another. Such objectors claim that what is instead needed is a *premathematical* explanation of logic in terms of even more primitive notions.

Remark 8.1. This objection may remind you of my objection in the first lecture to defining a programming language in terms of an interpreter, which is itself

inside a programming language (perhaps even the same language!). In both cases there is nothing *actually* invalid or "circular" about the practice, but the result is nevertheless somewhat unsatisfactory in that it simply shifts the burden of explanation "one level up."

Regardless of whether one sympathizes with this complaint, the starting point of today's lecture will be one such premathematical explanation of logic, which the Swedish logician Per Martin-Löf calls the *verificationistic theory of meaning* for intuitionistic predicate logic, in his lecture *Truth of a Proposition, Evidence of a Judgment, Validity of a Proof* [Mar87]. According to this perspective, there are three fundamental concepts in logic that we must explain: the notion of *proposition*, the notion of *truth* of a proposition, and the notion of validity of a *proof*.

1.1 Propositions

A proposition is a particular kind of assertion: one for which we must know what it would mean for the assertion to be true. We do not have to know whether it *is* true, but we do have to know what would count as evidence for its truth. That may sound silly or trivial, but in fact there are many hotly-debated assertions whose truth conditions are not at all clear, e.g., "A hot dog is a sandwich."

Suppose A and B are propositions, which is to say that we know what it would mean for each of these assertions to be true. Then we also know what it would mean for the assertion $A \wedge B$ ("A and B") to be true, namely that both A is true and B is true. Thus $A \wedge B$ is a proposition whenever A and B are propositions.

Remark 8.2. "But that is circular too—you used the word 'and' to define $\wedge$!" Yes, but the point is that we have reduced our understanding of logical conjunction to our innate understanding of the concept of what it means for two things to hold simultaneously, as expressed in natural language. We could certainly pick a phrasing that avoids the word 'and,' but at the end of the day it seems fair enough that logical conjunction boils down to the abstract concept of andness.

Similarly, if A and B are propositions, then

- $A \vee B$ ("A or B") is the proposition that is true when A is true or B is true;
- $\top$ ("true") is the proposition that is always true;
- $\bot$ ("false") is the proposition that is never true; and
- $A \Rightarrow B$ ("A implies B") is the proposition that is true when, supposing that A is true, B is true.

These truth conditions are known as the Brouwer–Heyting–Kolomogorov (BHK) interpretation of the logical connectives, and they provide a "verificationistic theory of meaning" in the sense that they explain the meaning of propositions by explaining how to verify them (rather than by, e.g., truth tables).

What kind of assertion is the statement that "A is a proposition" or that "A is true"? We do not want these to be *propositions*, because they are part of our explanation of what a proposition is. They are instead something more primitive: something that is possible to know, or to judge to be the case. Martin-Löf calls these *judgments*, and we can write them using the same notation we have used all semester: we write A prop for the judgment that A is a proposition, and we write inference rules to explain how various judgments relate to one another.[13]

$$\frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop}}{A \wedge B\ \mathsf{prop}} \qquad\qquad \frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop}}{A \vee B\ \mathsf{prop}}$$

$$\frac{}{\top\ \mathsf{prop}} \qquad\qquad \frac{}{\bot\ \mathsf{prop}} \qquad\qquad \frac{A\ \mathsf{prop} \qquad B\ \mathsf{prop}}{A \Rightarrow B\ \mathsf{prop}}$$

1.2 Truth of a proposition

Whenever A prop holds, it is sensible to ask whether A is true. We write this judgment A true, and its definition can be read directly off of the truth conditions described above.

$$\frac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}} \qquad\qquad \frac{A\ \mathsf{true}}{A \vee B\ \mathsf{true}} \qquad\qquad \frac{B\ \mathsf{true}}{A \vee B\ \mathsf{true}}$$

$$\frac{}{\top\ \mathsf{true}} \qquad\qquad \textit{(no rules for } \bot\textit{)} \qquad\qquad \frac{\begin{array}{c}(A\ \mathsf{true}) \\ \vdots \\ B\ \mathsf{true}\end{array}}{A \Rightarrow B\ \mathsf{true}}$$

We will discuss the last rule and its odd notation shortly, but for now, it has a single premise which is B true under the hypothesis A true. (The notation is meant to evoke a derivation of B true that is allowed to use the fact that A true.)

[13] In fact, this notation was all invented in the context of logic and not type systems.

2 Validity of a proof

This is all well and good, but we have still not explained the notion of *proof*, or how to actually verify that a given proposition is true (as opposed to what it *means* for a given proposition to be true). For this we introduce a proof calculus known as *natural deduction*, which consists of a collection of valid reasoning steps (written as inference rules) that can be chained together to form valid proofs (similarly to typing derivations).

The above rules for the A true judgment are certainly valid reasoning steps, being exactly the defining truth conditions of the connectives; however, they turn out not to be sufficient to derive interesting theorems. In particular, although they explain how to *introduce* connectives (e.g., from A true we can deduce $A \vee B$ true), they do not explain how to *eliminate* (use) connectives (e.g., from $A \wedge B$ true we should be able to deduce A true). The elimination rules look like this:

$$\frac{A \wedge B \text{ true}}{A \text{ true}} \qquad \frac{A \wedge B \text{ true}}{B \text{ true}} \qquad \dots$$

These rules may seem pointless at first: how could we have derived $A \wedge B$ true in the first place, if not in a "direct" fashion by combining derivations of A true and B true via the introduction rule? Hypotheses are the main "indirect" source of facts, so we should start by clearing up how they work in natural deduction. There are two standard notations, one being the odd schematic notation above, and the other being the trusty turnstile:

$$A_1 \text{ true}, A_2 \text{ true}, \dots A_n \text{ true} \vdash B \text{ true}$$

As usual, this means that B is true assuming that $A_1, A_2, \dots, A_n$ are true.

If we use the turnstile notation for hypothetical judgments and write down the elimination rule for each connective, we arrive at the standard collection of natural deduction rules in Figure 8.1, which define a system known as *intuitionistic propositional logic* (IPL). A proof of $\Gamma \vdash A$ true in IPL is a closed derivation tree with conclusion $\Gamma \vdash A$ true; to verify such a proof it suffices to check that each step of the derivation is a valid application of one of the rules in Figure 8.1.

Remark 8.3. The rules in Figure 8.1 may look somewhat familiar.

We can now see why elimination rules and "indirect" proofs are so useful.

Example 8.4. We can prove $\cdot \vdash (A \vee B) \Rightarrow (B \vee A)$ true as follows:

$$\dfrac{\dfrac{}{A \vee B \text{ true} \vdash A \vee B \text{ true}} \qquad \dfrac{\dfrac{}{A \vee B \text{ true}, A \text{ true} \vdash A \text{ true}}}{A \vee B \text{ true}, A \text{ true} \vdash B \vee A \text{ true}} \qquad \dfrac{\dfrac{}{A \vee B \text{ true}, B \text{ true} \vdash B \text{ true}}}{A \vee B \text{ true}, B \text{ true} \vdash B \vee A \text{ true}}}{\dfrac{A \vee B \text{ true} \vdash B \vee A \text{ true}}{\cdot \vdash (A \vee B) \Rightarrow (B \vee A) \text{ true}}}$$

$$\frac{\Gamma \vdash A \text{ true} \qquad \Gamma \vdash B \text{ true}}{\Gamma \vdash A \wedge B \text{ true}}\ \wedge\text{-INTRO} \qquad \frac{\Gamma \vdash A \wedge B \text{ true}}{\Gamma \vdash A \text{ true}}\ \wedge\text{-ELIM}_1$$

$$\frac{\Gamma \vdash A \wedge B \text{ true}}{\Gamma \vdash B \text{ true}}\ \wedge\text{-ELIM}_2 \qquad \frac{\Gamma, A \text{ true} \vdash B \text{ true}}{\Gamma \vdash A \Rightarrow B \text{ true}}\ \Rightarrow\text{-INTRO}$$

$$\frac{}{\Gamma, A \text{ true} \vdash A \text{ true}}\ \text{HYP} \qquad \frac{\Gamma \vdash A \Rightarrow B \text{ true} \qquad \Gamma \vdash A \text{ true}}{\Gamma \vdash B \text{ true}}\ \Rightarrow\text{-ELIM}$$

$$\frac{}{\Gamma \vdash \top \text{ true}}\ \top\text{-INTRO} \qquad \frac{\Gamma \vdash A \text{ true}}{\Gamma \vdash A \vee B \text{ true}}\ \vee\text{-INTRO}_1 \qquad \frac{\Gamma \vdash B \text{ true}}{\Gamma \vdash A \vee B \text{ true}}\ \vee\text{-INTRO}_2$$

$$\frac{\Gamma \vdash A \vee B \text{ true} \qquad \Gamma, A \text{ true} \vdash C \text{ true} \qquad \Gamma, B \text{ true} \vdash C \text{ true}}{\Gamma \vdash C \text{ true}}\ \vee\text{-ELIM}$$

$$\frac{\Gamma \vdash \bot \text{ true}}{\Gamma \vdash C \text{ true}}\ \bot\text{-ELIM}$$

Figure 8.1: Natural deduction for intuitionistic propositional logic.

Note that neither $A \vee B$ true $\vdash A \vee B$ true nor $A \vee B$ true $\vdash B \vee A$ true are proven by introduction rules; instead, they are proven by HYP and $\vee$-ELIM respectively.

3 Constructive logic

The basic version of our story is now complete: we have explained propositions, the truth of propositions, and the validity of proofs. But there is still much more to do and to say. First and foremost, we have talked at length about *truth*, but what about *falsity*? How do we represent the negation $\neg A$ of a proposition A?

Unlike in the traditional truth table account of logic, in which truth and falsehood play symmetric roles, the verificationistic theory of meaning privileges truth. The negation of a proposition is then not primitive, but rather definable as that proposition implying the false proposition.

$$\neg A := (A \Rightarrow \bot)$$

Intuitively, if $\neg A$ holds, then A is so untrue that if A were to be true, then even $\bot$ would be true!

Exercise 8.5. Prove the *principle of explosion*, $(A \wedge \neg A) \Rightarrow B$.

A downside of IPL's asymmetry between truth and falsehood is that neither the *law of excluded middle* $(A \vee \neg A)$ nor *double-negation elimination* $(\neg\neg A \Rightarrow A)$ hold. This is starkly different from classical propositional logic, in which both principles hold by case analysis: every classical proposition is either true or false.

The *upside* of IPL's asymmetry is that disjunction is much stronger in intuitionistic logic: in order to prove a disjunction $A \vee B$, one must *positively* establish one of the two disjuncts, rather than just showing that it is impossible that neither disjunct holds. Indeed, the "verificationistic" perspective is that propositions are true when we verify them; it is not sufficient to show that they cannot fail to hold.

The strength of intuitionistic disjunction is highlighted by the famous *disjunction property* that characterizes intuitionistic/constructive logics:

Theorem 8.6 (Disjunction property)**.** *If there is a closed derivation of* $\cdot \vdash A \vee B$ true *in IPL, then there must be a closed derivation of* $\cdot \vdash A$ true *or of* $\cdot \vdash B$ true.

Remark 8.7. The other hallmark property of intuitionistic logics is the *existence property*: if there is a closed derivation of $\cdot \vdash \exists x.A(x)$ true, then there must be some closed term t such that there is a closed derivation of $\cdot \vdash A(t)$ true.

A common first reaction to the disjunction property is that it must obviously hold for *every* logic, but note that this is not the case: in classical logic it is trivially

the case that (P = NP) ∨ (P ≠ NP) but this fact does not bring us any closer to proving either one, nor does it even imply that either is provable at all.

Even in intuitionistic logic it is not obvious that the disjunction property holds, because there are infinitely many "indirect" ways of proving disjunctions that do not obviously contain a proof of either disjunct, such as:

$$\dfrac{\dfrac{\bigtriangledown \qquad\qquad \bigtriangledown}{\cdot \vdash D \Rightarrow ((A \vee B) \wedge C)\ \mathsf{true} \qquad \cdot \vdash D\ \mathsf{true}}}{\dfrac{\cdot \vdash (A \vee B) \wedge C\ \mathsf{true}}{\cdot \vdash A \vee B\ \mathsf{true}}}$$

Remark 8.8. Note also that the disjunction and existence properties only hold in an empty context: there is a closed derivation of $A \vee B\ \mathsf{true} \vdash A \vee B\ \mathsf{true}$ but it is quite possible that neither $A \vee B\ \mathsf{true} \vdash A\ \mathsf{true}$ nor $A \vee B\ \mathsf{true} \vdash B\ \mathsf{true}$ are derivable.

The solution to this problem is to introduce a notion of *proof reduction* which cancels out all unnecessary detours in proofs, in order to eventually arrive at fully "direct" proofs of each connective, in particular of disjunctions. These reductions explain how to simplify instances of elimination rules applied to introduction rules:

$$\dfrac{\dfrac{\dfrac{\mathcal{D} \quad\quad \mathcal{D}'}{\Gamma \vdash A\ \mathsf{true} \qquad \Gamma \vdash B\ \mathsf{true}}\ \wedge\text{-INTRO}}{\Gamma \vdash A \wedge B\ \mathsf{true}}}{\Gamma \vdash A\ \mathsf{true}}\ \wedge\text{-ELIM}_1 \qquad \rightsquigarrow \qquad \begin{array}{c}\mathcal{D}\\ \bigtriangledown\\ \Gamma \vdash A\ \mathsf{true}\end{array}$$

Remark 8.9. To establish the disjunction and existence properties it is sufficient to consider a notion of proof reduction over derivations whose conclusions are in the empty context, but the reduction rules also apply to arbitrary Γ.

4 Propositions as types

It is notationally challenging to define proof reduction as a relation on schematic derivation trees, so it is natural to consider a linearized representation of proofs as *terms*. We introduce a new judgment $\Gamma \vdash a : A$ meaning that $\Gamma \vdash A\ \mathsf{true}$ where a is a *proof object*, or a representation of a particular valid proof of A.

For example, if we apply ∧-INTRO to $\Gamma \vdash a : A$ and $\Gamma \vdash b : B$, then we must obtain a new proof object for $\Gamma \vdash A \wedge B\ \mathsf{true}$ which records both of the subproofs

a and b as well as which proof rule was applied. If on the other hand we apply $\wedge$-ELIM$_1$ to $\Gamma \vdash p : A \wedge B$, then we obtain a new proof object for $\Gamma \vdash A$ true which records the subproof p as well as which proof rule was applied.

We might notate those proof objects as follows:

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \wedge B}\ \wedge\text{-INTRO} \qquad\qquad \frac{\Gamma \vdash p : A \wedge B}{\Gamma \vdash \mathtt{fst}(p) : A}\ \wedge\text{-ELIM}_1$$

in which case we can notate the proof reduction of $\wedge$-ELIM$_1$ applied to $\wedge$-INTRO as:

$$\mathtt{fst}((a, b)) \rightsquigarrow a$$

Similarly, the proof objects for disjunction might be notated as:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathtt{inl}(a) : A \vee B}\ \vee\text{-INTRO}_1 \qquad\qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathtt{inr}(b) : A \vee B}\ \vee\text{-INTRO}_2$$

$$\frac{\Gamma \vdash p : A \vee B \qquad \Gamma, x_1 : A \vdash c_1 : C \qquad \Gamma, x_2 : B \vdash c_2 : C}{\Gamma \vdash \mathtt{case}\ p\ [\mathtt{inl}(x_1) \to c_1]\ [\mathtt{inr}(x_2) \to c_2] : C}\ \vee\text{-ELIM}$$

in which case their proof reductions can be written:

$$\mathtt{case}\ \mathtt{inl}(a)\ [\mathtt{inl}(x_1) \to c_1]\ [\mathtt{inr}(x_2) \to c_2] \rightsquigarrow c_1[a/x_1]$$
$$\mathtt{case}\ \mathtt{inr}(b)\ [\mathtt{inl}(x_1) \to c_1]\ [\mathtt{inr}(x_2) \to c_2] \rightsquigarrow c_2[b/x_2]$$

By now you get the joke: if we add proof terms to the rules in Figure 8.1, then we notice an exact correspondence between propositions in IPL and types in STLC.

IPL propositions	**STLC types**
$A \wedge B$	$\tau_1 \times \tau_2$
$A \vee B$	$\tau_1 + \tau_2$
$\top$	`unit`
$\bot$	`void`
$A \Rightarrow B$	$\tau_1 \to \tau_2$

This correspondence cuts across every layer of the two systems—

propositions $\leftrightsquigarrow$ types
proof rules $\leftrightsquigarrow$ typing rules
proof objects $\leftrightsquigarrow$ programs
closed proof reductions $\leftrightsquigarrow$ operational semantics

—and is known as the *propositions-as-types* correspondence, the *proofs-as-programs* correspondence, and the *Curry–Howard isomorphism* or *correspondence*.

Type soundness and termination for STLC therefore imply that repeatedly applying proof reduction to any closed proof object $\cdot \vdash p : P$ will terminate with a "canonical" or "direct" proof object for the same proposition. Several important properties of IPL follow directly as corollaries. First is the disjunction property: given $\cdot \vdash p : A \vee B$ we either have $p \rightsquigarrow^* \texttt{inl}(a)$ where $\cdot \vdash a : A$ is canonical, or $p \rightsquigarrow^* \texttt{inr}(b)$ where $\cdot \vdash b : B$ is canonical. Second is *logical consistency*: there are no closed proofs of false $\cdot \vdash p : \bot$, for any such p must eventually reduce to a canonical proof of $\bot$, of which there are none.

The propositions-as-types correspondence also plays an important role in type-theoretic proof assistants such as Agda, Lean, and Rocq.

Discuss connection between classical logic and continuations: `(call/cc (lambda (k) (inr (lambda (p) (k (inl p))))))` : $P \vee \neg P$; CPS embeds classical logic into intuitionistic logic, providing the double-negation translation of P to $(P \Rightarrow \bot) \Rightarrow \bot$.

References

[Har16] Robert Harper. *Practical Foundations for Programming Languages*. Second Edition. Cambridge University Press, 2016. ISBN: 9781107150300. DOI: 10.1017/CBO9781316576892.

[Mar87] Per Martin-Löf. "Truth of a Proposition, Evidence of a Judgement, Validity of a Proof". In: *Synthese* 73.3 (1987), pp. 407–420. DOI: 10.1007/bf00484985.