The Curry-Howard Isomorphism for Propositional Logic

Edgar Lin

2022 February 28

1 Introduction

Intuitionistic logic attempts to formalize the the *BHK-interpretation* (named for Brouwer, Heyting, and Kolmogorov) of logical connectives. In particular, the BHK-interpretation of the logical connective $\to$ holds that a proof of $\varphi_1 \to \varphi_2$ is a program transforming any demonstration of φ_1 into one of φ_2. In order to demonstrate that a proof-system in intuitionistic logic properly formalizes the BHK-interpretation, then, we would like to show that any proof of $\varphi_1 \to \varphi_2$ does in fact correspond to a program in some formalism of computation that takes some representation of a proof of φ_1 to a proof of φ_2.

This correspondence is called the Curry-Howard correspondence. Here, we will explicate the Curry-Howard correspondence between the natural deduction proof-system for the implicational fragment of intuitionistic propositional logic, that is, the subset of intuitionistic propositional logic that only uses the logical connectives $\to$ and $\perp$, and the simply-typed lambda calculus, a formalism for representing computable functions.

The Curry-Howard isomorphism goes as follows: first, we interpret sentences in propositional logic as types of expressions in the simply-typed lambda calculus, in particular, we interpret sentences of the form $\varphi \to \psi$ as types of functions from expressions of the type corresponding to φ to expressions of the type corresponding to ψ. Then, a proof of a sentence is simply an expression in the simply-typed lambda calculus whose type is that sentence. The Curry-Howard isomorphism, then, statesthat provability in intuitionistic propositional logic is equivalent to type-inhabitation in the simply typed lambda-calculus by providing a one-to-one correspondence between every natural deduction proof of a sentence in intuitionistic propositional logic and every term in the simply-typed lambda calculus such that the type of the lambda-term is the sentence to be proved. Further, this correspondence is computable by a structure-preserving transformation between the abstract syntax-trees of the proof and the program.

In the following sections, we will outline the implicational fragment of intuitionistic propositional logic (IPC($\to$)) along with the natural deduction proof system, before demonstrating the Curry-Howard isomorphism between the two systems and some of its uses and abuses. When it comes to demonstrating the computational properties of the correspondence, we will write our computations as Haskell programs instead of using formal systems of defining computations for the sake of readability.

2 The Natural Deduction Proof System for the IPC($\to$)

The syntactic rules for the implicational fragment of intuitionistic propositional formulae are much the same as that for classical logic. We start with a countably infinite set of propositional variables/atomic propositions PV. Then, our set of formulae/sentences Φ is the smallest set of finite strings of characters in containing all of the below:

- Every $A \in PV$,
- $(A \to B)$ for every $A, B \in \Phi$.

We will assume the unique reading lemma. Then, we can equivalently define logical formulae as strings corresponding to the following abstract syntax tree (in Haskell):

```
module CurryHoward where
import qualified Data.Set as Set
import qualified Data.List as List
type PV = String
data Formula = Atomic PV
    | Implies Formula Formula
  deriving (Show, Eq, Ord)
```

A *theory* is a subset $\Gamma \subset \Phi$. We call a finite theory Γ a *context*. For a formula φ we wrote Γ, ϕ for $\Gamma \cup \{\varphi\}$.

```
type Theory = Set.Set Formula
```

For a context Γ and a formula φ we write $\Gamma \vdash \varphi$ to denote the fact that Γ proves φ. We call any φ such that $\vdash \varphi$ a theorem of IPC($\rightarrow$). A proof of $\Gamma \vdash \varphi$ is a (finite) tree whose nodes are labeled with context-formula pairs, which we will denote $\Delta \vdash \psi$, and that obey the following rules:

- The root node is labeled by $\Gamma \vdash \varphi$, that is, the conclusion of the proof.
- The leaf nodes are labeled $\Delta, \psi \vdash \psi$. These correspond to using an axiom in a proof.
- Non-leaf nodes have one or more children. The labels of parent nodes are inferred from those of their children by one of the following rules:
  - Introduction of $\rightarrow$ ($\rightarrow$ I): From $\Delta, \psi \vdash \pi$, infer $\Delta \vdash \psi \rightarrow \pi$.
  - Elimination of $\rightarrow$ ($\rightarrow$ E): From $\Delta \vdash \psi \rightarrow \pi$ and $\Delta \vdash \psi$, infer $\Delta \vdash \pi$.

Proof checking is computable, moreover, it is computable in linear time. We can write a natural deduction tree and proof-checking rules in Haskell as follows:

```
data Proof = Axiom (Theory, Formula)
  | Introduction (Theory, Formula) Proof
  | Elimination (Theory, Formula) Proof Proof
  deriving (Show, Eq)
label :: Proof → (Theory, Formula)
label (Axiom a) = a
label (Introduction a _) = a
label (Elimination a _ _) = a
valid :: Proof → Bool
valid (Axiom (g, p)) = Set.member p g
valid (Introduction (g, Implies p q) c) =
  (fst $ label c) ≡ Set.insert p g ∧ (snd $ label c) ≡ q
valid (Introduction _ _) = False
valid (Elimination (g, p) major minor) =
```

$$\begin{aligned}&(\mathit{fst}\ \$\ \mathit{label\ major}) \equiv g\\ &\wedge (\mathit{fst}\ \$\ \mathit{label\ minor}) \equiv g\\ &\wedge (\mathit{snd}\ \$\ \mathit{label\ major}) \equiv \mathit{Implies}\ (\mathit{snd}\ \$\ \mathit{label\ minor})\ p\end{aligned}$$

We note that we usually use the following notation to write nodes of natural deduction trees:

$$\frac{\mathit{Children}}{\mathit{Label}},$$

2.1 Natural Deduction & BHK

We can interpret the rules of natural deduction in BHK as follows. First, we take $\{\varphi_1, \varphi_2, \ldots, \varphi_n\} \vdash \psi$ as "Given demonstrations for each φ_i, there is some construction built out of the φ_i that is a demonstration of ψ." Then, we can interpret the →I rule, $\frac{\Gamma,\phi\vdash\psi}{\Gamma\vdash\phi\to\psi}$ as follows: "We have that given demonstrations for the contents of Γ and for ϕ we can build out of those demonstrations a demonstration of ψ. Then, using the demonstrations for the contents of Γ, we can write that demonstration of ψ with the demonstrations for ϕ erased and then write a program that takes as input a new demonstration of ϕ and inserts it into the blanks where the old demonstrations for ϕ were. This program takes as input a demonstration of ϕ and produces a demonstration of ψ, so it is a proof of $\phi \to \psi$." Finally, the →E rule can be interpreted as applying the program that is the proof of the major premise to the proof of the minor premise to get a proof of the conclusion.

2.2 Natural Deduction & Hilbert-Style Proofs

Recall the Hilbert-style proof system for classical logic where we had only one inference rule, modus ponens, and instead had three axiom schemes (we assume that → associates to the right for brevity):

P_1: $\phi \to \psi \to \phi$.

P_2: $(\phi \to \psi \to \chi) \to (\phi \to \psi) \to \phi \to \chi$.

P_3: $((\phi \to \bot) \to \bot) \to \phi$.

The →E rule clearly corresponds to modus ponens.

What does →I correspond to? We note that P_1 and P_2 in the Hilbert proof system are sufficient to prove the deduction theorem:

Theorem 1 (Deduction Theorem)**.** *In the Hilbert proof system with only axioms P_1 and P_2, if $\Gamma, \phi \vdash \psi$, then, $\Gamma \vdash \phi \to \psi$.*

We will delay the proof until we have introduced a slight variation on proof trees in order to prove a slightly stronger version of the theorem.

Further, we can show that every member of the axiom schemes P_1 and P_2 are theorems of IPC($\to$) in natural deduction:

Theorem 2. *For any triplet of propositions (ϕ, ψ, χ), denote their corresponding axioms in the axiom schemes P_1 and P_2 as $P_{1,\phi,\psi}$ and $P_{2,\phi,\psi,\chi}$ respectively. Then, for all $\phi, \psi, \chi \in \Phi$, we have that in IPC($\to$), $\vdash P_{1,(\phi,\psi)}$ and $\vdash P_{2,(\phi,\psi,\chi)}$.*

Proof. For any ϕ, ψ, the following is a natural deduction proof of $P_{1,\phi,\psi}$:

$$\cfrac{\cfrac{\phi, \psi \vdash \phi}{\phi \vdash (\psi \to \phi)}}{\vdash (\phi \to (\psi \to \phi))}.$$

We delay the proof of P_2 until after we have the Curry-Howard Isomorphism. □

2.3 Intuitionistic Algebraic Semantics

We would like to dwell a bit on the semantics of intuitionistic logic before moving on to the simply typed lambda-calculus. In specific, we would like to show that the set of statements provable by natural deduction in the implicational fragment is equal to the set of statements that are *semantically* necessarily true according to the algebraic semantics of the full system of intuitionistic logic.

Further, we want to use the semantics of logical deduction to illustrate some important differences between intuitionistic logic and classical logic. We noted in the previous section that natural deduction for the implicational fragment of intuitionistic logic corresponds to the Hilbert proof system in classical logic without the axiom $((\phi \to \bot) \to \bot) \to \phi$. This axiom can be interpreted as formalizing the idea of a proof (of a positive statement) by contradiction: that is, if $\neg\phi$ proves a contradiction, then ϕ must be true. The fact that this axiom is not given in intuitionistic logic can be seen as a consequence of the *constructive* nature of intuitionistic logic: it's not clear how a construction of a contradiction out of $\phi \to \bot$ can directly be used to make a construction of ϕ itself. We would like to be able to actually prove that intuitionistic logic can't actually prove that in general $((\phi \to \bot) \to \bot) \to \phi$. We can do this semantically by appeal to a topological interpretation of the algebraic semantics of intuitionistic logic.

2.3.1 Heyting Algebras

We recall that we defined the semantics of classical logic by reference to models that induced mappings of each sentence

2.3.2 Soundness and Completeness

2.3.3 Topological Intepretation of Heyting Algebras

3 Simply-Typed Lambda Calculus

3.1 Church-Rosser Theorem

3.2 Weak-Normalization

3.3 Computational Power of Typed and Untyped Lambda Calculus

4 The Curry-Howard Isomorphism

4.1 Natural Deduction Proofs with Labeled Assumptions

4.2 Proof of the Isomorphism

4.3 Application: Consistency

4.4 Application: Untypability of the Y combinator

4.5 Application: Combinatory Logic & Hilbert System