CHAPTER 3

Propositional Proofs

3.1 Introduction

Checking logical entailment with truth tables has the merit of being conceptually simple. However, it is not always the most practical method. The number of truth assignments of a language grows exponentially with the number of logical constants. When the number of logical constants in a propositional language is large, it may be impossible to process its truth table.

Proof methods provide an alternative way of checking logical entailment that addresses this problem. In many cases, it is possible to create a proof of a conclusion from a set of premises that is much smaller than the truth table for the language; moreover, it is often possible to find such proofs with less work than is necessary to check the entire truth table.

We begin this lesson with a discussion of simple, linear proofs. We then move on to hierarchically structured proofs, where proofs can be nested inside of other proofs. Once we have seen both linear and structured proofs, we show how they are combined in the popular Fitch proof system. We finish with definitions for soundness and completeness - the standards by which proof systems are judged.

3.2 Linear Proofs

As we saw in the introductory lesson, the essence of logical reasoning is symbolic manipulation. We start with premises, apply rules of inference to derive conclusions, stringing together such derivations to form logical proofs. The idea is simple. Getting the details right requires a little care. Let's start by defining schemas and rules of inference.

A *schema* is an expression satisfying the grammatical rules of our language except for the occurrence of *metavariables* (written here as Greek letters) in place of various subparts of the expression. For example, the following expression is a pattern with metavariables ϕ and ψ.

$$\phi \Rightarrow \psi$$

A *rule of inference* is pattern of reasoning consisting of some schemas, called *premises*, and one or more additional schemas, called *conclusions*. Rules of inference are often written as shown below. The schemas above the line are the premises, and the schemas below the line are the conclusions. The rule in this case is called *Implication Elimination* (or IE), because it eliminates the implication from the first premise.

$$\begin{array}{l} \phi \Rightarrow \psi \\ \phi \\ \hline \psi \end{array}$$

An instance of a rule of inference is the rule obtained by consistently substituting sentences for the metavariables in the rule. For example, the following is an instance of Implication Elimination.

$$\frac{\begin{array}{l} p \Rightarrow q \\ p \end{array}}{q}$$

If a metavariable occurs more than once, the same expression must be used for every occurrence. For example, in the case of Implication Elimination, it would not be acceptable to replace one occurrence of ϕ with one expression and the other occurrence of ϕ with a different expression.

Note that the replacement can be an arbitrary expression so long as the result is a legal expression. For example, in the following instance, we have replaced the variables by compound sentences.

$$\frac{\begin{array}{l} (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \\ (p \Rightarrow q) \end{array}}{(q \Rightarrow r)}$$

Remember that there are infinitely many sentences in our language. Even though we start with finitely many propositional constants and finitely many operators, we can combine them in arbitrarily many ways. The upshot is that there are infinitely many instances of any rule of inference involving metavariables.

A rule *applies* to a set of sentences if and only if there is an instance of the rule in which all of the premises are in the set. In this case, the conclusions of the instance are the results of the rule application.

For example, if we had a set of sentences containing the sentence p and the sentence $(p \Rightarrow q)$, then we could apply Implication Elimination to derive q as a result. If we had a set of sentences containing the sentence $(p \Rightarrow q)$ and the sentence $(p \Rightarrow q) \Rightarrow (q \Rightarrow r)$, then we could apply Implication Elimination to derive $(q \Rightarrow r)$ as a result.

Note that it is okay to have rules of inference with no premises. Rules with no premises are sometimes called *axiom schemata*. Axiom schemata are usually written without the horizontal line used in displaying ordinary rules of inference. Here are some examples.

The *Implication Creation schema* (IC), when used in combination with Implication Elimination, allows us to infer implications.

$$\phi \Rightarrow (\psi \Rightarrow \phi)$$

The *Implication Distribution schema* (ID) allows us to distribute one implication over another. If a sentence ϕ implies that ψ implies χ, then, if ϕ implies ψ, ϕ implies χ.

$$(\phi \Rightarrow (\psi \Rightarrow \chi)) \Rightarrow ((\phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow \chi))$$

The *Contradiction Realization schema* (CR) permits us to infer a sentence if the negation of that sentence implies some sentence and its negation.

$$(\neg\phi \Rightarrow \psi) \Rightarrow ((\neg\phi \Rightarrow \neg\psi) \Rightarrow \phi)$$

These three axiom schemata together form the *Mendelson axiom schemata* for Propositional Logic. The interesting thing about the Mendelson axiom schemata is that, together with implication Elimination, they alone are sufficient to prove all logical consequences from any set of premises.

By writing down premises, writing instances of axiom schemata, and applying rules of inference, it is possible to derive conclusions that cannot be derived in a single step. This idea of stringing things together in this way leads to the notion of a linear proof.

A *linear proof* of a conclusion from a set of premises is a sequence of sentences terminating in the conclusion in which each item is either (1) a premise, (2) an instance of an axiom schema, or (3) the result of applying a rule of inference to earlier items in sequence.

Here is an example. Suppose we have the set of sentence we saw earlier. We start our proof by writing out our premises. We believe p; we believe $(p \Rightarrow q)$; and we believe that $(p \Rightarrow q) \Rightarrow (q \Rightarrow r)$. Using Implication Elimination on the first premise and the second premise, we derive q. Applying Implication Elimination to the second premise and the third premise, we derive $(q \Rightarrow r)$. Finally, we use the derived premises on lines 4 and 5 to arrive at our desired conclusion.

1. p	Premise
2. $p \Rightarrow q$	Premise
3. $(p \Rightarrow q) \Rightarrow (q \Rightarrow r)$	Premise
4. q	IE: 2, 1
5. $q \Rightarrow r$	IE: 3, 2
6. r	IE: 5, 4

Here is a somewhat more complicated example, which illustrates the use of axiom schemata. Whenever p is true, q is true. Whenever q is true, r is true. With these as premises, we can prove that, whenever p is true, r is true. On line 3, we write down an instance of Implication Creation. From this conclusion and premise 2, we use Implication Elimination to derive the sentence $(p \Rightarrow (q \Rightarrow r))$. On line 5, we write down an instance of Implication Distribution. We combine the conclusions from lines 4 and 5 to derive the sentence on line 6. Finally, we use Implication Elimination once again to derive the desired conclusion. Not an easy proof, but it works. Later in this chapter, we see how to produce a simpler proof of this result.

1. $p \Rightarrow q$	Premise
2. $q \Rightarrow r$	Premise
3. $(q \Rightarrow r) \Rightarrow (p \Rightarrow (q \Rightarrow r))$	IC
4. $p \Rightarrow (q \Rightarrow r)$	IE: 3, 2
5. $p \Rightarrow (q \Rightarrow r) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$	ID
6. $(p \Rightarrow q) \Rightarrow (p \Rightarrow r)$	IE: 5, 4
7. $p \Rightarrow r$	IE: 6, 1

Let R be a set of rules of inference. If there exists a proof of a sentence ϕ from a set Δ of premises using the rules of inference in R, we say that ϕ is *provable* from Δ using R. We usually write this as $\Delta \vdash_R \phi$, using the provability operator $\vdash$ (which is sometimes called *single turnstyle*). If the set of rules is clear from context, we sometimes drop the subscript, writing just Δ |- φ.

The Mendelson System, mentioned earlier, is a well-known proof system for Propositional Logic. It consists of the Implication Elimination rule of inference and the three axiom schemata we saw earlier. The Mendelson System is interesting in that is sufficient to prove all logical consequences from any set of premises expressed using only ¬ and ⇒. And, as we shall see later, this subset of sentences is interesting because, for every sentence in Propositional Logic, there is a logically equivalent sentence written in terms of these two operators. Unfortunately, proofs in the Mendelson system can be complicated. The good news is that we can eliminate much of this complexity by moving from linear proofs to structured proofs.

3.3 Structured Proofs

Structured proofs are similar to linear proofs in that they are sequences of reasoning steps. However, they differ from linear proofs in that they have more structure. In particular, sentences can be grouped into subproofs nested within outer superproofs.

It resembles a linear proof except that we have grouped the sentences on lines 3 through 5 into a subproof within our overall proof.

$$\begin{array}{r|l|l}
1. & p \Rightarrow q & \text{Premise} \\
2. & q \Rightarrow r & \text{Premise} \\
3. & \quad\vert\; p & \text{Assumption} \\
4. & \quad\vert\; q & \text{Implication Elimination: 3, 1} \\
5. & \quad\vert\; r & \text{Implication Elimination: 4, 2} \\
6. & p \Rightarrow r & \text{Implication Introduction: 3, 5}
\end{array}$$

The main benefit of structured proofs is that they allow us to prove things that cannot be proved using only ordinary rules of inference. In structured proofs, we can make assumptions within subproofs; we can prove conclusions from those assumptions; and, from those derivations, we can derive implications outside of those subproofs, with our assumptions as antecedents and our conclusions as consequents.

The structured proof above illustrates this. On line 3, we begin a subproof with the assumption that *p* is true. Note that *p* is not a premise in the overall problem. In a subproof, we can make whatever assumptions that we like. From *p*, we derive *q* using the premise on line 1; and, from that *q*, we prove *r* using the premise on line 2. That terminates the subproof. Finally, from this subproof, we derive (*p*⇒*r*) in the outer proof. Given *p*, we can prove *r*; and so we know (*p*⇒*r*). The rule used in this case is called Implication Introduction, or II for short.

As this example illustrates, there are three basic operations involved in creating structured proofs - (1) making assumptions, (2) using ordinary rules of inference to derive conclusions, and (3) using structured rules of inference to derive conclusions outside of subproofs. Let's look at each of these operations in turn.

In a structured proof, it is permissible to make an arbitrary assumption in any nested proof. The assumptions need not be members of the initial premise set. Note that such assumptions cannot be used directly outside of the subproof, only as conditions in derived implications, so they do not contaminate the superproof or any unrelated subproofs.

For example, in the proof we just saw, we used this assumption operation in the nested subproof even though p was not among the given premises.

An ordinary rule of inference applies to a particular subproof of a structured proof if and only if there is an instance of the rule in which all of the premises occur earlier in the subproof or in some superproof of the subproof. Importantly, it is not permissible to use sentence in subproofs of that subproof or in other subproofs of its superproofs.

For example, in the structured proof we have been looking at, it is okay to apply Implication Elimination to 1 and 3. And it is okay to use Implication Elimination on lines 2 and 4.

However, it is *not* acceptable to use a sentence from a nested proof in applying an ordinary rule of inference to an outer proof, as in the malformed proof shown below.

Utilizing results proved in subproofs is the responsibility of a new type of rule of inference. Like an ordinary rule of inference, a structured rule of inference is pattern of reasoning consisting of one or more premises and one or more conclusions. As before, the premises and conclusions can be schemas. However, the premises can also include conditions of the form $\phi \vdash \psi$, as in the following example. The rule in this case is called Implication Introduction, because it allows us to introduce new implications.

$$\frac{\phi \vdash \psi}{\phi \Rightarrow \psi}$$

Once again, looking at our example, we see that there is an instance of Implication Introduction (shown here on the left) in deriving line 6 from the subproof on lines 3 - 5.

Finally, we define a *structured proof* of a conclusion from a set of premises to be a sequence of (possibly nested) sentences terminating in an occurrence of the conclusion at the *top level* of the proof. Each step in the proof must be either (1) a premise (at the top level) or an assumption (other than at the top level) or (2) the result of applying an ordinary or structured rule of inference to earlier items in the sequence (subject to the constraints given above).

3.4 Fitch

Fitch is a proof system that is particularly popular in the Logic community. It is as powerful as many other proof systems and is far simpler to use. Fitch achieves this simplicity through its support for structured proofs and its use of structured rules of inference in addition to ordinary rules of inference.

Fitch has ten rules of inference in all. Nine of these are ordinary rules of inference. The other rule (Implication Introduction) is a structured rule of inference.

And Introduction (shown below on the left) allows us to derive a conjunction from its conjuncts. If a proof contains sentences ϕ_1 through ϕ_n, then we can their conjunction. *And Elimination* (shown below on the right) allows us to derive conjuncts from a conjunction. If we have the conjunction of ϕ_1 through ϕ_n, then we can infer any of the conjuncts.

And Introduction

$$\frac{\begin{array}{l}\phi_1 \\ \ldots \\ \phi_n\end{array}}{\phi_1 \wedge \ldots \wedge \phi_n}$$

And Elimination

$$\frac{\phi_1 \wedge \ldots \wedge \phi_n}{\phi_i}$$

Or Introduction allows us to infer an arbitrary disjunction so long as at least one of the disjuncts is already in the proof. *Or Elimination* is a little more complicated than And Elimination. Since we do not know which of the disjuncts is true, we cannot just drop the $\vee$. However, if we know that every disjunct entails some sentence, than we can infer that sentence even if we do not know which disjunct is true.

Or Introduction

$$\frac{\phi_i}{\phi_1 \vee \ldots \vee \phi_n}$$

Or Elimination

$$\frac{\begin{array}{l}\phi_1 \vee \ldots \vee \phi_n \\ \phi_1 \Rightarrow \psi \\ \ldots \\ \phi_n \Rightarrow \psi\end{array}}{\psi}$$

Negation Introduction allows us to derive the negation of a sentence if it leads to a contradiction. If we believe ($\phi\Rightarrow\psi$) and ($\phi\Rightarrow\neg\psi$), then we can derive that phi is false. *Negation Elimination* allows us to delete double negatives.

Negation Introduction

$$\frac{\begin{array}{l}\phi \Rightarrow \psi \\ \phi \Rightarrow \neg\psi\end{array}}{\neg\phi}$$

Negation Elimination

$$\frac{\neg\neg\phi}{\phi}$$

Implication Introduction is the structured rule we saw in section 3.3. If, by assuming ϕ, we can derive ψ, then we can derive ($\phi \Rightarrow \psi$). *Implication Elimination* is the first rule we saw Section 3.2.

Implication Introduction

$$\frac{\phi \vdash \psi}{\phi \Rightarrow \psi}$$

Implication Elimination

$$\frac{\begin{array}{l}\phi \Rightarrow \psi \\ \phi\end{array}}{\psi}$$

Equivalence Introduction allows us to infer an equivalence from an implication and its inverse. *Equivalence Elimination* goes the other way, allowing us to infer two implications from a single equivalence.

Equivalence Introduction

$\phi \Rightarrow \psi$

Equivalence Elimination

$\phi \Leftrightarrow \psi$

$$\begin{array}{l} \psi \Rightarrow \phi \\ \hline \phi \Leftrightarrow \psi \end{array} \qquad\qquad \begin{array}{l} \hline \phi \Rightarrow \psi \\ \psi \Rightarrow \phi \end{array}$$

In addition to these rules of inference, it is common to include in Fitch proof editors several additional operations that are of use in constructing Fitch proofs. For example, the Premise operation allows one to add a new premise to a proof. The Reiteration operation allows one to reproduce an earlier conclusion for the purposes of clarity. The blank operation allows one to add a blank line in a proof for formatting and subsequent editing. the Open Proof operation allows one to start a subproof. The Close Subproof operation allows one to close a subproof. Finally, the Delete operation allows one to delete unnecessary lines.

3.5 Soundness and Completeness

In talking about logic, we now have two notions - logical entailment and provability. A set of premises logically entails a conclusion if and only if every truth assignment that satisfies the premises also satisfies the conclusion. A sentence is provable from a set of premises if and only if there is a finite proof of the conclusion from the premises.

The concepts are quite different. One is based on truth assignments; the other is based on symbolic manipulation of expressions. Yet, for the proof systems we have been examining, they are closely related.

We say that a proof system is *sound* if and only if every provable conclusion is logically entailed. In other words, if $\Delta |\text{-} \phi$, then $\Delta |= \phi$. We say that a proof system is *complete* if and only if every logical conclusion is provable. In other words, if $\Delta |= \phi$, then $\Delta |\text{-} \phi$.

The Mendelson System is both sound and complete for all problems in which the premises and conclusions are expressed using only ¬ and ⇒. In other words, for this system, logical entailment and provability are identical.

The Fitch system is sound and complete for the full language. In other words, for this system, logical entailment and provability are also identical. An arbitrary set of sentences Δ logically entails an arbitrary sentence ϕ if and only if ϕ is provable from Δ using Fitch.

The upshot of this result is significant. On large problems, the proof method often takes fewer steps than the truth table method. (Disclaimer: In the worst case, the proof method may take just as many or more steps to find an answer as the truth table method.) Moreover, proofs are usually much smaller than the corresponding truth tables. So writing an argument to convince others does not take as much space.

Recap

A *pattern* is an expression satisfying the grammatical rules of our language except for the occurrence of *metavariables* in place of various subparts of the expression. An *instance* of a pattern is the expression obtained by substituting expressions of the appropriate sort for the metavariables in the pattern so that the result is a legal expression. A *rule of inference* is a pattern of reasoning consisting of one set of patterns, called *premises*, and a second set of patterns, called *conclusions*. An *axiom schema* is effectively a rule of inference with no premises. A *linear proof* of a conclusion from a set of premises is a sequence of sentences terminating in the conclusion in which each item is either (1) a premise or (2) the result of applying a rule of inference to earlier items in sequence. If there exists a proof of a sentence ϕ from a set Δ of premises and the axiom schemata and rules of inference of a proof system, then ϕ is said to be *provable* from Δ (written as $\Delta \vdash \phi$) and is called a *theorem* of Δ. *Fitch* is a powerful yet simple proof system that supports structured proofs. A proof system is *sound* if and only if every provable conclusion is logically entailed. A proof system is *complete* if and only if every logical conclusion is provable. Mendelson is sound and complete for all sentences that can be written in terms of $\neg$ and $\Rightarrow$. Fitch is sound and complete for the full language.