CHAPTER 4

Propositional Proofs

4.1 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 linear reasoning and linear proofs. We then move on to hypothetical reasoning and structured proofs. Once we have seen both linear and structured proofs, we show how they are combined in the popular Fitch proof system, and we provide some tips for finding proofs using the Fitch system. We finish with definitions for soundness and completeness - the standards by which proof systems are judged.

4.2 Linear Reasoning

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 schema with metavariables ϕ and ψ.

$$\phi \Rightarrow \psi$$

A *rule of inference* is a 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.

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

The rule in this case is called *Implication Elimination* (or IE), because it eliminates the implication from the first premise.

Implication Creation (IC), shown below, is another example. This rule tells us that, if a sentence ψ is true, we can infer $(\phi \Rightarrow \psi)$ for any ϕ whatsoever.

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

Implication Distribution (ID) tells us that implication can be distributed over other implications. If $(\phi \Rightarrow (\psi \Rightarrow \chi))$ is true, then we can infer $((\phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow \chi))$.

$$\begin{array}{l} \phi \Rightarrow (\psi \Rightarrow \chi) \\ (\phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow \chi) \end{array}$$

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.

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

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 of Implication Elimination, we have replaced the variables by compound sentences.

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

Remember that there are infinitely many sentences in our language. Even though we start with finitely many propositional constants (in a propositional vocabulary) 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.

In using rules of inference, it is important to remember that they apply only to top-level sentences, not to components of sentences. While applying to components sometimes works, it can also lead to incorrect results.

As an example of such a problem, consider the incorrect application of Implication Elimination shown below. Suppose we believe $(p \Rightarrow q)$ and $(p \Rightarrow r)$. We might try to apply Implication Elimination here, taking the first premise as the implication and taking the occurrence of p in the second premise as the matching condition, leading us to conclude $(q \Rightarrow r)$.

$$p \Rightarrow q$$
$$p \Rightarrow r$$
$$\overline{q \Rightarrow r}$$

Unfortunately, this is not a proper logical conclusion from the premises, as we all know from experience and as we can quickly determine by looking at the associated truth table. It is important to remember that rules of inference apply only to top-level sentences.

By writing down premises, writing instances of axiom schemas, 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 sentences 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	Implication Elimination: 2, 1
5. $q \Rightarrow r$	Implication Elimination: 3, 2
6. r	Implication Elimination: 5, 4

Here is another example. 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 use Implication Creation to derive $(p \Rightarrow (q \Rightarrow r))$. On line 4, we use Implication Distribution to distribute the implication in line 3. Finally, on line 5, we use Implication Elimination to produce the desired result.

1. $p \Rightarrow q$	Premise
2. $q \Rightarrow r$	Premise
3. $p \Rightarrow (q \Rightarrow r)$	Implication Creation: 2
4. $(p \Rightarrow q) \Rightarrow (p \Rightarrow r)$	Implication Distribution: 3
5. $p \Rightarrow r$	Implication Elimination: 4, 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 turnstile*). If the set of rules is clear from context, we usually drop the subscript, writing just $\Delta \vdash \phi$.

Note that the set of rules presented here is not powerful enough to prove everything that is entailed by a set of premises in Propositional Logic. There is no support for using or deducing negations or conjunctions or disjunctions or biconditionals. Even if we restrict ourselves to implications, we need more rules. While such rules of inference exist, they are a little complicated. For many people, it is easier to reason about implications using hypothetical reasoning.

4.3 Hypothetical Reasoning

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.

As an example, consider the structured proof shown below. It resembles a linear proof except that we have grouped the sentences on lines 3 through 5 into a subproof within our overall proof.

1.	$p \Rightarrow q$	Premise
2.	$q \Rightarrow r$	Premise
3.	$\quad p$	Assumption
4.	$\quad q$	Implication Elimination: 3, 1
5.	$\quad r$	Implication Elimination: 4, 2
6.	$p \Rightarrow r$	Implication Introduction: 3, 5

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 $(p \Rightarrow r)$ in the outer proof. Given p, we can prove r; and so we know $(p \Rightarrow r)$. 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 useful subproofs - (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 subproof. 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 sentences 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 subproof in applying an ordinary rule of inference in a superproof.

The last line of the malformed proof shown below gives an example of this. It is *not* permissible to use Implication Elimination as shown here because it uses a conclusion from a subproof as a premise in an application of an ordinary rule of inference in its superproof.

	1.	$p \Rightarrow q$	Premise	
	2.	$q \Rightarrow r$	Premise	
	3.	$\quad p$	Assumption	
	4.	$\quad q$	Implication Elimination: 1, 3	
	5.	$\quad r$	Implication Elmination: 2, 4	
	6.	$p \Rightarrow r$	Implication Introduction: 3, 5	
Wrong!	7.	r	Implication Elimination: 2, 4	**Wrong!**

The malformed proof shown below is another example. Here, line 8 is illegal because line 4 is not in the current subproof or a superproof of this subproof.

	1.	$p \Rightarrow q$	Premise	
	2.	$q \Rightarrow r$	Premise	
	3.	$\quad p$	Assumption	
	4.	$\quad q$	Implication Elmination: 1, 3	
	5.	$\quad r$	Implication Elmination: 2, 4	
	6.	$p \Rightarrow r$	Implication Introduction: 3, 5	
	7.	$\quad \neg r$	Assumption	
Wrong!	8.	$\quad r$	Implication Elmination: 2, 4	**Wrong!**
	9.	$\neg r \Rightarrow r$	Implication Introduction: 7, 8	

Correctly utilizing results derived 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 a 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 $\phi \vdash \psi$. The rule in this case is called Implication Introduction, because it allows us to introduce new implications.

$$\phi \vdash \psi$$
$$\phi \Rightarrow \psi$$

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).

4.4 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 infer 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	**And Elimination**
ϕ_1	$\phi_1 \land \ldots \land \phi_n$
...	ϕ_i
ϕ_n	
$\phi_1 \land \ldots \land \phi_n$	

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 $\lor$. However, if we know that every disjunct entails some sentence, then we can infer that sentence even if we do not know which disjunct is true.

Or Introduction	**Or Elimination**
ϕ_i	$\phi_1 \lor \ldots \lor \phi_n$
$\phi_1 \lor \ldots \lor \phi_n$	$\phi_1 \Rightarrow \psi$
	...
	$\phi_n \Rightarrow \psi$
	ψ

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 ϕ is false. *Negation Elimination* allows us to delete double negatives.

Negation Introduction	**Negation Elimination**
$\phi \Rightarrow \psi$	$\neg\neg\phi$
$\phi \Rightarrow \neg\psi$	ϕ
$\neg\phi$	

Implication Introduction is the structured rule we saw in section 4.3. If, by assuming ϕ, we can derive ψ, then we can derive $(\phi \Rightarrow \psi)$. *Implication Elimination* is the first rule we saw Section 4.2.

Implication Introduction	**Implication Elimination**
$\phi \vdash \psi$	$\phi \Rightarrow \psi$
$\phi \Rightarrow \psi$	ϕ
	ψ

Biconditional Introduction allows us to deduce a biconditional from an implication and its inverse. *Biconditional Elimination* goes the other way, allowing us to deduce two implications from a single biconditional.

Biconditional Introduction	**Biconditional Elimination**
$\phi \Rightarrow \psi$	$\phi \Leftrightarrow \psi$
$\psi \Rightarrow \phi$	$\phi \Rightarrow \psi$
$\phi \Leftrightarrow \psi$	$\psi \Rightarrow \phi$

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. Finally, the Delete operation allows one to delete unnecessary lines.

4.5 Reasoning Tips

The Fitch rules are all fairly simple to use; and, as we discuss in the next section, they are all that we need to prove any result that follows logically from any set of premises. Unfortunately, figuring out which rules to use in any given situation is not always that simple. Fortunately, there are a few tricks that help in many cases.

If the goal has the form ($\phi \Rightarrow \psi$), it is often good to assume ϕ and prove ψ and then use Implication Introduction to derive the goal. For example, if we have a premise q and we want to prove ($p \Rightarrow q$), we assume p, reiterate q, and then use Implication Introduction to derive the goal.

1.	$\vert\ q$	Premise
2.	$\vert\ \ \vert\ p$	Assumption
3.	$\vert\ \ \vert\ q$	Reiteration: 1
4.	$\vert\ p \Rightarrow q$	Implication Introduction: 2, 3

If the goal has the form ($\phi \wedge \psi$), we first prove ϕ and then prove ψ and then use And Introduction to derive ($\phi \wedge \psi$).

If the goal has the form ($\phi \vee \psi$), all we need to do is to prove ϕ or prove ψ, but we do not need to prove both. Once we have proved either one, we can disjoin that with anything else whatsoever.

If the goal has the form ($\neg\phi$), it is often useful to assume ϕ and prove a contradiction, meaning that ϕ must be false. To do this, we assume ϕ and derive some sentence ψ leading to ($\phi \Rightarrow \psi$). We assume ϕ again and derive some sentence $\neg\psi$ leading to ($\phi \Rightarrow \neg\psi$). Finally, we use Negation Introduction to derive $\neg\phi$ as desired.

More generally, whenever we want to prove a sentence ϕ of any sort, we can sometimes succeed by assuming $\neg\phi$, proving a contradiction as just discussed and thereby deriving $\neg\neg\phi$. We can then apply Negation Elimination to get ϕ.

The following two tips suggest useful things we can try based on the form of the premises and the goal or subgoal we are trying to prove.

If there is a premise of the form ($\phi \Rightarrow \psi$) and our goal is to prove ψ, then it is often useful to try proving ϕ. If we succeed, we can then use Implication Elimination to derive ψ.

If we have a premise $(\phi \vee \psi)$ and our goal is to prove χ, then we should try proving $(\phi \Rightarrow \chi)$ and $(\psi \Rightarrow \chi)$. If we succeed, we can then use Or Elimination to derive χ.

As an example of using these tips in constructing the proof, consider the following problem. We are given $p \vee q$ and $\neg p$, and we are asked to prove q. Since the goal is not an implication or a conjunction or a disjunction or a negation, only the last of the goal-based tips applies. Unfortunately, this does not help us in this case. Luckily, the second of the premise-based tips is relevant because we have a disjunction as a premise. To use this all we need is to prove $p \Rightarrow q$ and $q \Rightarrow q$. To prove $p \Rightarrow q$, we use the first goal-based tip. We assume p and try to prove q. To do this we use that last goal-based tip. We assume $\sim q$ and prove p. Then we assume $\sim q$ and prove $\neg p$. Since we have proved p and $\neg p$ from $\neg q$, we can infer q. Using Implication Introduction, we then have $p \Rightarrow q$. Proving $q \Rightarrow q$ is easy. Finally, we can apply or elimination to get the desired result.

```
 1. │p | q        Premise
 2. │¬p           Premise
 3. │ │p          Assumption
 4. │ │ │¬q       Assumption
 5. │ │ │p        Reiteration: 3
 6. │ │¬q ⇒ p     Implication Introduction: 4, 5
 7. │ │ │¬q       Assumption
 8. │ │ │¬p       Reiteration: 2
 9. │ │¬q ⇒ ¬p    Implication Introduction: 7, 8
10. │ │¬¬q        Negation Introduction: 6, 9
11. │ │q          Negation Elimination: 10
12. │p ⇒ q        Implication Introduction: 3, 11
13. │ │q          Assumption
14. │q ⇒ q        Implication Introduction: 13
15. │q            Or Elimination: 1, 12, 14
```

In general, when trying to generate a proof, it is useful to apply the premise tips to derive conclusions. However, this often works only for very short proofs. For more complex proofs, it is often useful to think backwards from the desired conclusion before starting to prove things from the premises in order to devise a strategy for approaching the proof. This often suggests subproblems to be solved. We can then work on these simpler subproblems and put the solutions together to produce a proofs for our overall conclusion.

4.6 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 $\Delta \vdash \phi$, then $\Delta \vDash \phi$. We say that a proof system is *complete* if and only if every logical conclusion is provable. In other words, if $\Delta \vDash \phi$, then $\Delta \vdash \phi$.

The Fitch system is sound and complete for the full language. In other words, for this system, logical entailment and provability are 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 *schema* 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 schema is the expression obtained by substituting expressions of the appropriate sort for the metavariables in the schema so that the result is a legal expression. A *rule of inference* is a pattern of reasoning consisting of one set of schemas, called *premises*, and a second set of schemas, called *conclusions*. 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 schemas 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. Fitch is sound and complete for Propositional Logic.

Exercises

Exercise 4.1: Given p and q and $(p \wedge q \Rightarrow r)$, use the Fitch system to prove r.

Exercise 4.2: Given $(p \wedge q)$, use the Fitch system to prove $(q \vee r)$.

Exercise 4.3: Given $p \Rightarrow q$ and $q \Leftrightarrow r$, use the Fitch system to prove $p \Rightarrow r$.

Exercise 4.4: Given $p \Rightarrow q$ and $m \Rightarrow p \vee q$, use the Fitch System to prove $m \Rightarrow q$.

Exercise 4.5: Given $p \Rightarrow (q \Rightarrow r)$, use the Fitch System to prove $(p \Rightarrow q) \Rightarrow (p \Rightarrow r)$.

Exercise 4.6: Use the Fitch System to prove $p \Rightarrow (q \Rightarrow p)$.

Exercise 4.7: Use the Fitch System to prove $(p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$.

Exercise 4.8: Use the Fitch System to prove $(\neg p \Rightarrow q) \Rightarrow ((\neg p \Rightarrow \neg q) \Rightarrow p)$.

Exercise 4.9: Given p, use the Fitch System to prove $\neg\neg p$.

Exercise 4.10: Given $p \Rightarrow q$, use the Fitch System to prove $\neg q \Rightarrow \neg p$.

Exercise 4.11: Given $p \Rightarrow q$, use the Fitch System to prove $\neg p \vee q$.

Exercise 4.12: Use the Fitch System to prove $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$.

Exercise 4.13: Given $\neg(p \vee q)$, use the Fitch system to prove $(\neg p \wedge \neg q)$.

Exercise 4.14: Use the Fitch system to prove the tautology $(p \vee \neg p)$.