LogicSpaces

An Introduction to Propositional and Quantificational Logics

Michael Nelson
Department of Philosophy
University of California, Riverside
Riverside, CA 92521

Last revised Summer 2018

Contents

Introduction

This text is an introduction to logic. The reader is introduced to basic logical notions like argument, validity, and logical consequence, as well as basic techniques in logic, from using truth tables, presenting semantic proofs, to constructing derivations. The first four chapters focus on truth-functions, which are the most basic and fundamental of all logical operations and serve as the foundation for almost every more powerful logic, are widely used in computer science and programming languages, and are widely used in every day reasoning and argumentation. In English, truth-functions are expressed by words like 'not', 'and', 'or', and 'if, then'. Grammatically, these words are used to create more complex sentences from simpler sentences, where the truth value of the complex sentence is a function of the truth values of its simpler parts. For example, the truth value of the complex sentence 'John is home and Jane is at work' is a function of the truth values of the simpler sentences 'John is home' and 'Jane is at work', where the complex sentence is true exactly when both of the simpler sentences are true. The sentence 'It is not the case that John is home' (which is admittedly much less natural than 'John is not at home', but is still obviously grammatical and sensible) is formed from the simpler sentence 'John is home' and the truth value of the complex sentence is a function of the truth value of the simpler sentence; the former is true exactly when the latter is false. In the last three chapter of this text, we broaden our horizons and study the logic of terms, predicates, and quantifiers.

We will first learn to symbolize the truth-functional expressions of English into a formal language: The language of Propositional Logic. The language has a vocabulary, which is a set of symbols from which sentences are constructed, and a syntax, which is a set of rules establishing which combinations of symbols from this vocabulary are sentences of the language. We will then study the semantics of this formal language, which tells us what the items in the vocabulary and the sentences composed of that vocabulary mean and, in particular, when sentences are true and when they are false. Part of our study of the semantics of Propositional Logic will include learning about truth tables, which we will use to determine various features of sentences and sets of sentences of the language of Propositional Logic (and, by extension, the English sentences those sentences symbolize). Finally, we will learn a proof theory for Propositional Logic and learn how to construct derivations or proofs within that system. These three components, the *syntax* or grammar of a formal language, the *semantics* of a formal language, and the *proof theory*, are the three components of any and every logic.

It is standard for the second stop in one's study of logic to be the logic of quantifiers and this text is no exception. Expressions like 'some', 'every', and 'at most 10' are all quantifiers and they have a very rich and fascinating logic covered in the last three chapters. The text will put the reader in a position to go on to learn more complicated logics that promise to model the logical relations that go beyond the truth-functional operators of Propositional Logic and the relations between first-order terms and quantifiers.

Chapter 1

Arguments and Validity

An "argument" in ordinary parlance is a dispute or disagreement between parties. You might argue with your housemate about whose turn it is to clean the bathroom, have an argument with your mother about your holiday plans, etc. This is *not* what an argument is in logic, although we will see some connections below.

An **argument** is a sequence of sentences, the last member of which is the **conclusion** and the earlier members of which are **premises**. Here are four examples of arguments in this sense. (The words 'so' and 'therefore' mark the drawing of an inference and so the conclusion of the argument. We will employ the convention of numbering sentences for ease of later reference. So, for example, '(1) of argument 3' refers to the sentence 'George is sitting'.)

Argument 1:

(1) The study of logic traces back to ancient Greece.

So, (C) Logic is fun.

Argument 2:

(1) George is sitting and thinking.

So, (C) George is thinking.

Argument 3:

(1) George is sitting.
(2) If George is eating, then George is sitting.

Therefore, (C) George is eating.

Argument 4:

(1) If George is sitting, then he is eating or he is working on his logic problems.
(2) George is not eating.

Therefore, (C) If George is sitting, then he is working on his logic problems.

The last sentence of each argument, the sentences labeled '(C)', is the argument's conclusion and the earlier sentences are the premises. So, the premise of argument 1 is the sentence 'The study of logic traces back to ancient Greece' and the conclusion of argument 3 is 'George is eating' (note that the word 'therefore' is not part of the conclusion itself). In a nonformal presentation of an argument, the conclusion needn't appear last and the premises first; sometimes the order is mixed up, premises are implicit and unstated, and there are various other ways in which the presentation of an argument does not follow this canonical mode of presentation. It often takes substantial work just to uncover and adequately represent the argument. In this course, however, we will present arguments in the above form, with every premise explicitly stated and every conclusion the last sentence in the series. Our interest is not to learn to uncover and represent arguments contained in natural language texts but instead to learn to evaluate those arguments once uncovered. (More exactly, our interest is to evaluate arguments along a single dimension: Namely, that of validity, which we will begin to discuss in the following paragraph.) So, we will always have our arguments "prepackaged" in the above canonical form.

The premises of a good argument provide support for the argument's conclusion. Arguments persuade by moving one from previously accepted truths (the premises) to a conclusion. The persuasive force of a good argument is grounded in the rational transition of this movement from premises to conclusion. While there are many facets to a persuasive argument, one key component is validity, which is the feature we will be primarily concerned with in this course. We first characterize the notion of validity and then discuss what other ingredients go into a "good" argument at the end of this chapter.

An argument is **valid** when it is absolutely impossible for the premises to all be true together and the conclusion false. Validity is like a guarantee that you will not move from true premises to a false conclusion, as that arrangement of truth values (all premises true while the conclusion is false) is impossible. Two of the arguments presented above, arguments 2 and 4, are valid while the other two arguments, arguments 1 and 3, are invalid.

We begin with the invalid arguments, as it is much more straightforward to demonstrate invalidity than validity. The primary way to show that an argument is invalid is to construct a **countermodel**, where a countermodel to an argument is a possibility in which all of the argument's premises are true while its conclusion is false. The first argument is obviously invalid. The conclusion is unrelated to the premise and so it is quite easy to imagine a situation in which

the premise is true (i.e., that logic traces back to ancient Greece) and yet the conclusion is false (that logic is not fun). So, argument 1 is invalid.

Notice that the premise and the conclusion are both in fact true (believe us that logic is fun, whether or not you ever realize it). The argument is nonetheless invalid because it is *possible* for the premise to be true and the conclusion false. There are plenty of invalid arguments with true conclusions and with true premises. Mere truth does not suffice for validity, as the joint truth of all of the premises of a valid argument *necessitates* the truth of its conclusion. A sequence of sentences might well all be true without the truth of the first members of the sequence in any way guaranteeing the truth of the last. (Argument 1 is an example.) The truth of the premises of a valid argument *guarantee* the truth of the conclusion. So, the validity of an argument goes beyond the mere truth of its conclusion and premises.

Argument 3 is more interesting. It too is invalid, being an instance of a classic fallacy: Affirming the consequent. Sentence (2) of argument 3 is a conditional sentence of the form '*if —, then...*', where the 'if'-clause is the conditional's **antecedent** and the 'then'-clause its **consequent**. The form that argument 3 takes is one where a conditional sentence and the consequent of that conditional sentence serve as premises and the antecedent of that conditional serves as conclusion. We will discuss this, including the notion of an argument's *form*, in greater detail in future chapters. For now, we can see that the argument is invalid by thinking through a possible way for all of the premises to be true while the conclusion is false, and thus a countermodel, as follows. Let George be sitting (to make the first premise true) and not eating (to make the conclusion false). The second premise is also true in those circumstances, as its truth only requires that it is not the case that George is both eating and not sitting. (This may not be obvious to you. We will discuss the truth conditions of conditional sentences in detail in chapter 3.) So, there is a possible situation in which all of the premises are true and the conclusion is false: Namely, the possibility in which George is sitting and not eating. So, the argument is invalid.

We said earlier that a situation that makes all of the premises of an argument true and its conclusion false is a countermodel to the argument. If an argument has a countermodel, then the argument is invalid. We demonstrate the invalidity of an argument by constructing or describing a countermodel. Note: It only takes one possibility, however improbable or remote, in which all of the premises are true and the conclusion is false to make an argument invalid. Note as well: A countermodel must be a single possibility in which all of the premises are true and the conclusion false; one can't switch to a distinct possibility in the middle of the description of the countermodel. There are plenty of valid arguments where it is possible that all of the premises are true and possible that the conclusion is false, although there is no single possibility in which both all of the premises are true and the conclusion is false. The thing to remember, then, is that an argument is invalid exactly when there is at least one possibility in which all of the premises are true together while the conclusion is false. Such a possibility is a countermodel and an argument with a countermodel is invalid.

Arguments 2 and 4 are valid. We demonstrate the validity of an argument

by showing that it is absolutely impossible for all of its premises to be true and its conclusion false. Demonstrating validity is significantly more difficult than demonstrating invalidity. The latter only requires, recall from above, describing a single possibility in which all of the premises are true and the conclusion false; a countermodel. Demonstrating validity, by contrast, requires establishing the *nonexistence* of a countermodel. That cannot be done with a single possibility, as one must show that, of *all* the possibilities there are, none of them make all of the premises true and the conclusion false. Merely showing that a single possibility is not a countermodel leaves open that some other possibility around the corner is a countermodel.

There are several general methods for demonstrating validity. We will here distinguish three. The first is the direct method. Under certain conditions, we can list all of the relevant possibilities and show, for each possibility individually, that either one of the premises is false in that possibility or the conclusion is true in that possibility, and thus that none are countermodels. Insofar as we have really considered *every* possibility, we will have demonstrated the validity of the argument by this method. We will develop a version of the direct method in future chapters, once we introduce truth tables and learn how to use them to determine logical consequence. (See chapter 3, sections 6 and 7.) However, the direct method has a limited application. It becomes unmanageable when the number of possibilities to consider are great in number and downright unworkable when there are infinitely (or at least indefinitely) many different possibilities to consider, as is the case with logics more powerful than Propositional Logic and with informal arguments like arguments 2 and 4.

The shortcomings of the direct method spur us to develop other methods for demonstrating the validity of an argument. We will introduce two additional methods, neither of which suffer from the shortcomings of the direct method, although a flip side of this increase in power is that these methods are significantly more difficult to use. The second of our three methods is similar to the first in that it proves validity by unpacking the conditions under which the premises are all true and the conclusion false in order to show that no possibility meets them all, hence establishing the nonexistence of a countermodel. It is unlike the first method, however, in that it does not involve directly inspecting each possibility in turn. Instead this method establishes that there are no countermodels indirectly. The most general version of this method relies on a form of argumentation widely used in mathematics called *reductio ad absurdum*, where a claim is established by showing that the supposition that it is false leads to a contradiction (or an "absurdity," from whence it earned its name), where a contradiction, for our purposes, is a sentence and its negation. This is the form of argumentation Euclid used, for example, to prove that there are infinitely many prime numbers.

> (A convention: We will set in smaller font, with wider margins, and in blue text digressions and expansions of points that go deeper than necessary for the purposes of an introductory course, but, we hope, are still of interest to the engaged student. These sections are optional.) We will present a modern,

algebraic version of Euclid's proof, not faithful to his geometric version, in this note, as it is a model of elegance and cleverness and well illustrates *reductio* arguments. A prime number is a natural number greater than 1 that is only evenly divisible by 1 and itself. Suppose, for *reductio*, that there were only finitely many primes. Then some finite list $p_1, \ldots, p_n$ would include every prime number. Consider now the product of this list of numbers plus 1, which we shall call q; i.e., $q = p_1 \times \ldots \times p_n + 1$. Either q is a prime or it isn't. If q is a prime, then there is a prime number not on our list, as q is larger than any number on the list and so not identical to any of those numbers, which contradicts our initial assumption that $p_1, \ldots, p_n$ are *all* of the prime numbers. If, on the other hand, q is not a prime, then, by the unique factorization theorem, it is the unique product of a set of prime numbers. Because $q - 1$ is the product of all of the primes on our original list, all of those primes are factors of $q - 1$. For any number n, however, no prime is a factor of both n and $n + 1$. So, none of the primes on our original list is a factor of q (as all are factors of $q - 1$). So, there is at least one other prime number not on the original list that is among the prime factorization of q, which contradicts our initial assumption that $p_1, \ldots, p_n$ are *all* of the prime numbers. So, whether or not q is a prime, there is a prime number not on our list, which contradicts our earlier assumption that every prime occurs on our list. So, there are infinitely many primes.

The idea behind this method of argumentation is that, because contradictions are not true, any claim that entails a contradiction is also not true and so its contrary is true. In our case, we show that an argument is valid by first supposing, for *reductio*, that there is a countermodel and showing that that supposition leads to a contradiction, concluding that there are no countermodels and so that the argument is valid. The third method for establishing validity is the method of derivation. In this method, we derive the conclusion from the premises using only sound inference rules. We will discuss this method, both in general and in detail, in chapter 4 below, where we will describe what a derivation is, what inference rules are, and what it is for an inference rule to be sound, as well as learn how to construct derivations in PL. For now, we will just note this third method's existence and note that it is very different from the first two, as it does not explicitly consider the truth conditions of the premises and conclusion, focusing instead on the syntactic forms of sentences; it is a *proof-theoretic* method while the first two methods are *semantic* methods. This difference is of great significance, but it is not possible to discuss it until we have more material on the table. So, we will set it aside until chapter 4. Our focus for the remainder of this chapter will be the second method. We will first develop an informal version of the method and then return to a more formal version of the method in chapter 3, section 9.

We will develop an informal version of our second, indirect method of establishing validity by using the *reductio* form of argumentation described above to show that argument 2 is valid. We begin by assuming (for the purposes of *reductio*) that it is possible for premise (1) to be true while conclusion (C) is false.

(1) George is sitting and thinking.

(C) George is thinking.

We now unpack the truth conditions that these suppositions impose. (1) is true at a possibility **w** just in case both George is sitting at **w** and George is thinking at **w**. (C) is false at **w** just case it is not the case that George is thinking at **w**. But, by (1), George is thinking at our possibility **w**. So, putting things back together, George is both sitting and not sitting at **w**. That is a contradiction. So, there is no such possibility where (1) is true and (C) is false. So, argument 2 is valid.

It is important to note that the actual truth values of the premise(s) and conclusion of an argument are irrelevant to its validity. A valid argument may have a false premise and even a false conclusion (although, if the argument is valid and its conclusion is false, then at least one premise is also false, as the argument's validity requires that it is absolutely impossible for all of the premises to be true and the conclusion false). So, when we are interested in the validity of argument 2, we don't ask whether or not George is sitting and thinking; indeed, we don't even ask who George is and how we might find him to assess his state. We don't need to consider the actual truth value of the premises nor of the conclusion to determine the validity of an argument. Instead, we *suppose* the truth of the premises and falsity of the conclusion, unpacking that supposition in search of a contradiction.

A valid argument whose premises, and hence conclusion, are true is said to be **sound**. When we use arguments to either determine what to believe or convince someone else of what to believe, it is soundness, not mere validity, that should be our primary concern. But validity is an indispensable component of soundness and so it is worthy of its own study, which is what we will do in this course. (Indeed, soundness breaks into two largely independent components: Validity and truth.)

Our discussion of argument 2 illustrates the basic idea behind the second indirect method. You most likely had an intuitive sense that argument 2 is valid prior to this demonstration just by understanding how 'and' works. Also, the use of the above informal version of our second method is easy to follow. However, matters are different with argument 4. It is by no means easy "just to see" that argument 4 is valid. Worse, the informal version of our second method to demonstrate the validity of this argument is itself difficult to follow and, even when followed, leaves room to doubt the correctness of the application, given its complexity. For completeness sake, we will present an informal demonstration that argument 4 is valid. However, we encourage you to not get too bogged down with the details and, should you find it difficult, set it aside until we have presented the machinery needed to develop formal methods to prove its validity in chapters to come, at which point we will apply them to rigorously demonstrate the validity of argument 4. We will return repeatedly to argument 4 in future chapters, applying the machinery and methods we will be developing to this argument. We encourage you to return to the discussion in the paragraphs that follow again after you have mastered those later methods.

Suppose, for *reductio*, that it is possible for both premises (1) and (2) of argument 4, reproduced below, are true while (C) is false.

> (1) If George is sitting, then he is eating or he is working on his logic problems.
>
> (2) George is not eating.
>
> (C) If George is sitting, then he is working on his logic problems.

Then there is a single possibility **w** such that both (1) and (2) are true at **w** and (C) is false at **w**. Because (C) is false at **w**, George is sitting at **w** and George is not working on his logic problems at **w**. (We ask you to simply accept on authority our claims about the truth and falsity of conditional sentences for the present.) Because (2) is true at **w**, George is not eating at **w**.

We also supposed that (1) is true at **w**. (1)'s truth conditions are significantly more complex than (2) and (C)'s. Because (1) is true at **w**, either i) George is not sitting at **w** or ii) either a) George is eating at **w** or b) George is working on his logic problems at **w**. So, because (1) is true, at least one of the following conditions obtains.

> (i) George is not sitting at **w**.
>
> (iia) George is eating at **w**.
>
> (iib) George is working on his logic problems at **w**.

Thus, (1)'s truth condition can be represented as a disjunction: (1) is true exactly when at least one of the above three conditions is met. We show that the truth of (1) contradicts our earlier assumptions that (2) is true and (C) is false by showing that, whichever of these three conditions obtains, a contradiction results (although not, immediately at least, the same contradiction in each case). We do this by supposing, for each condition individually, that it obtains and arguing that a contradiction follows. (This is an extremely useful and powerful form of argumentation, sometimes called *Disjunctive Syllogism*.) Because our assumption that (1) is true at **w** requires that at least one of those conditions obtains and, for each individually, a contradiction follows, we can conclude that a contradiction follows from the assumption that (1) is true.

Suppose, then, that i) obtains. We saw earlier that, because (C) is false, George is sitting at **w**. So, George is both sitting and not sitting at **w**. Contradiction. Suppose second that iia) obtains. We saw earlier that, because (2) is true, George is not eating at **w**. Then George is both eating and not eating at **w**. Contradiction. Suppose finally that iib) obtains. We saw earlier that, because (C) is false, George is not working on his logic problems at **w**. Then George is both working and not working on his logic problems at **w**. Contradiction. So, whichever of the three alternatives obtains, a contradiction results. So, because at least one of these alternatives obtains (given our initial *reductio* assumption that (1) and (2) are true while (C) is false), a contradiction results.

We have shown that our original supposition that, for some possibility **w**, both (1) and (2) are true at **w** and (C) is false at **w** leads to a contradiction. So, there is no such possibility and hence no countermodel to argument 4. So, argument 4 is valid.

1.1 Validity, Good Argumentation, and Reasonable Inferences

We have focused on one specific property of arguments: Validity. But we are ultimate interested in an argument's rationally persuasive force. A good argument provides reasons for the truth its conclusion. When one is given a good argument for some claim, one is given convincing reason to think that that claim is true and so, all other things being equal and absent countervailing reasons to the contrary, one should believe that claim. The relationship between a valid argument and a rationally persuasive, good argument is complex. In this section, we shall explore some of those connections, although our discussion is far from complete.

First, there are plenty of valid arguments that are *not* good arguments and do not correspond to good inferences. Here are two examples.

Argument 5:

(1) George is innocent.

So, (C) George is innocent.

Argument 6:

(1) George is sitting and not sitting.

So, (C) George is innocent.

Both arguments 5 and 6 are valid. It is absolutely impossible for (1) of argument 5 to be true and (C) false, as they are one and the same sentence and a single sentence cannot be both true and false at the same time. (1) of argument 6 is a contradiction. As there are not any possibilities in which a contradiction is true, it follows that there are no countermodels to argument 6, as there are not any possibilities in which the contradictory (1) is true. Despite their validity, neither argument is rationally persuasive. Anyone in doubt of George's guilt should not be persuaded of his innocence by either argument. Even if George is in fact innocent and so argument 5 is sound, as it is valid and its premise is true, it does not rationally persuade. Neither premise provides evidential support for the truth of (C), even though, in both cases, (C) follows from (1), in the sense that it is absolutely impossible for (1) to be true and (C) false.

The above considerations show that an argument's being valid is not sufficient for its being rationally persuasive. x is **sufficient** for y just in case x's

being the case guarantees y's being the case; intuitively, x's being the case is "enough" or "all it takes" for y to be the case. For example, being really fast is sufficient for being fast. To say that x is not sufficient for y, then, is to say that it is possible for x to be the case and y to not be the case. For example, being fast is not sufficient for being really fast, as Usain Bolt's opponents quickly learn. Because arguments 5 and 6 are valid but not rationally persuasive, an argument's being valid is not *sufficient* for its being rationally persuasive. A deep, delicate, and controversial set of issues are opened when we try to find what valid but unpersuasive arguments like 5 and 6 lack that persuasive arguments possess. These issues are beyond the scope of the present text.

The flip side of sufficiency is necessity. x's being the case is **necessary** for y's being the case when it is absolutely impossible for y to be the case and x to not be the case. For example, having a sibling is a necessary condition for having a brother, as it is absolutely impossible to have a brother and not have a sibling. (Having a sibling is not sufficient for having a brother, of course, as one might have only sisters. Returning to our case illustrating the notion of sufficiency, we can also see that being really fast is not necessary for being fast, as some people are just fast but not really fast.) An argument's being valid is not a necessary condition for its being rationally persuasive. We can see this by showing that there are arguments that are rationally persuasive but not valid. As an example, consider the following.

Argument 7:

(1) Everyone (in a large random sampling) who drove into the station filled his or her car with gas.

(C) The next person that drives into the station will fill up his or her car with gas.

It is perfectly possible for (1) to be true and (C) false, which makes argument 7 invalid; no matter how many previous visitors to the station filled his or her car with gas, there is no logical guarantee that the next visitor will also fill up. Despite the argument's invalidity, argument 7 is nonetheless a virtuous argument, unlike arguments 1, 3, 5, and 6. While the truth of the conclusion (C) is not necessitated by the truth of the premise (1) of argument 7, nonetheless the truth of (1) makes (C) more probable and provides compelling rational grounds for believing it, even though those grounds are defeasible. Argument 7 is an example of a good *inductive argument*, where the premises of an inductive argument fall short of logically entailing its conclusion (as is the case in a deductively valid argument) but nonetheless make the truth of the conclusion more probable and hence raises its credence. Some inductively valid arguments are rationally persuasive, but none are valid, where validity is defined as we have in this chapter: An argument is valid when it is absolutely impossible for all of its premises to be true together while its conclusion is false. So, an argument's being (deductively) valid is not a necessary condition for its being rationally persuasive.

Inductive arguments are commonplace, both in philosophy, other academic fields, and in everyday life. They are the basis of inferences to the best explanation, where a claim is supported on the grounds that its truth best explains some given phenomena. A familiar example of an inductive argument is illustrated by the following case. The court knows that George was in the bar where John was shot at the time that John was shot, that George carries a gun just like the murder weapon, that George had been fighting with John before John was shot, and that George had gun powder on his hand when the police questioned him the next morning. The court quite reasonably infers that George is guilty of murder. The evidence supports the conclusion that George murdered John, but the evidence hardly deductively entails that conclusion. The reasoning that leads one to conclude George's guilt from this body of evidence, then, when formulated as an argument, is an example of good inductive but not deductive argumentation. The study of inductive arguments, important as it is to both every day life and an understanding of the scientific method, is beyond the scope of this course. We will focus solely on deductive validity.

We have seen that an argument's being deductively valid is neither necessary nor sufficient for its being rationally persuasive. Why, then, do we focus on validity? There are several reasons. The first is that validity is the property that is easiest to formalize and study rigorously. It it thus the natural starting point for introductory and intermediate courses. Trying to study inductive arguments or directly attacking the notion of rational persuasion before first learning the logic of deductive arguments is a little bit like trying to learn about exponents without first learning addition. Second, other, more complicated connections one might identify between the premises and conclusion of a rationally persuasive argument will bear a resemblance to that of deductive validity. For example, the notion of probability raising, central to an account of good inductive arguments, illustrated in the previous paragraph, is deeply connected to the property of deductive validity. Deductive validity, then, is a central and important notion in understanding good argumentation and reasonable inference, even if it does not tell the whole story.

In the following chapters, we will develop formal methods for determining and establishing both the validity and invalidity of arguments.

CHAPTER REVIEW

- An argument is a sequence of sentences, the last member of which is the conclusion and the earlier members (if the sequence has more than one element; there are premiseless arguments) are the premises.
- An argument is valid just in case it is absolutely impossible for all of the premises to be true and the conclusion false.
- A countermodel to an argument is a possibility in which all of the premises of the argument are true and its conclusion is false.
- An invalid argument has a countermodel and a valid argument does not.
- We demonstrate the invalidity of an argument by showing that there is a countermodel. We do this by describing a possibility in which all of its premises are true and its conclusion is false.
- We demonstrate the validity of an argument by showing the nonexistence of a countermodel. There are three methods. On the first, brute method, we consider every possibility for the sentences comprising the argument and show that none of them are countermodels. On the second, indirect method, we establish the nonexistence of a countermodel by showing that the assumption that there is a countermodel to lead to a contradiction. The third method is the method of derivation, in which we derive the conclusion from the premises using sound inference rules.
- x's being the case is a necessary condition for y's being the case when it is absolutely impossible for y to be the case and x to not be the case. x's being the case is a sufficient condition for y's being the case when it is absolutely impossible for x to be the case and y to not be the case.

Chapter 2

The Language of Propositional Logic

In the previous chapter, we distinguished valid from invalid arguments and presented an informal version of the method of countermodels for establishing invalidity and introduced an informal method for establishing validity, briefly mentioning two other methods. Our discussion of arguments 3 and 4 illustrate the fact that it can be difficult to tell whether or not an argument is valid and that our informal methods are often difficult to use. This motivates developing rigorous and formal methods to replace the informal methods of the earlier chapter. Our formal methods will, however, take their cues from their informal cousins. In this chapter, then, we take the first steps towards developing formal tests for invalidity and validity.

2.1 Symbolizing Arguments

We begin by *symbolizing* arguments, translating natural language sentences, what we will call our **target** sentences, into a formal language, the **language of Propositional Logic** ('PL', for short). Learning to symbolize arguments of a natural language like English into a formal language like PL carries several benefits. First, it encourages abstract thought, which is useful in computer science, all varieties of aptitude tests, critical reasoning, etc. Second, it allows us to make explicit and precise the meanings and forms of the arguments we are studying, features often absent in informal versions of those arguments. This will in turn allow us to develop step by step procedures for determining and establishing the validity and invalidity of natural language arguments that can be adequately translated into our formal language of PL.

We first introduce our formal language in this chapter by showing how to translate sentences of English into sentences of this new language. Because you are antecedently familiar with the meanings of the original English sentences, this reveals something about the meanings of the sentences of PL. This is the

method of direct translation for learning a new language. In the next chapter we will present a more rigorous account of the meanings of PL sentences by developing what is called a *semantics* for that language, where we will describe rules for determining what arbitrary sentences of PL mean. The semantics will offer us our first rigorous method for testing and establishing the validity and invalidity of arguments. Once our discussion of the semantics of PL and associated semantic methods for determining validity and invalidity are complete, we will move on to a presentation of the proof theory of PL, where we will present a distinct method for establishing the validity of arguments.

Recall argument 4 from the previous chapter, reproduced, with some underlining whose import will soon become apparent, below.

Argument 4:

(1) If George is sitting, then either he is eating or he is working on his logic problems.
(2) George is not eating.

Therefore, (C) If George is sitting, then he is working on his logic problems.

Compare that argument with the following.

Argument 4′:

(1) If Katherine is home, then either she left work early or it is Sunday.
(2) Katherine did not leave work early.

So, (C) If Katherine is home, then it is Sunday.

While the subject matters of the two arguments differ – the first concerns George and his activities while the second concerns Katherine and her comings and goings – they share a common form, as is highlighted by the underlined words they share, which pattern in exactly the same way in the two arguments.

We can make more explicit the common form arguments 4 and 4′ share by abstracting away their differences, as follows.

(1) If . . ., then either — or ***.
(2) Not —.

So, (C) If . . ., then ***.

Here '. . .', '—', and '***' stand for sub-sentences in the above arguments. For example, '. . .' stands for 'George is sitting' in argument 4 and for 'Katherine is home' in argument 4′, and so on. The remaining words, the words underlined in the earlier presentations of the arguments, are the what we will call the *logic words*; more exactly, they express truth-functions, which we will define below.

We can improve on this more abstract representation of the common logical form of arguments 4 and 4′ by using atomic sentence-letters in place of '. . .',

'—', and '***'. Atomic sentence-letters are the basic building blocks of the language of PL. The language includes an infinite stock of atomic sentence-letters, $\{p, q, r, s, p_1, q_1, r_1, s_1, p_2, q_2, \ldots\}$. (Luckily, we'll usually only need the first few for our purposes.) We can then replace '...', '—', and '***' in the above representation of the common form of arguments 4 and 4′ with atomic sentence-letters, in effect associating each sub-sentence in the target sentences to be translated with a unique atomic sentence-letter. We do this with a translation manual, which we will describe below.

There are three basic sentential structures that occur in sentences (1), (2), and (C) of argument 4, our target sentences, each of which we associate with its own unique atomic sentence-letter, as follows.

Translation manual for argument 4:

$p \Longrightarrow$ George is sitting

$q \Longrightarrow$ George is eating

$r \Longrightarrow$ George is working on George's logic problems

(Notice that we have more fully "filled out" and resolved certain ambiguities and underdeterminations of the linguistic material in the target sentences. In particular, we replaced all pronouns with 'George' and we moved the negation 'not' from a predicate modifier to a sentential modifier.) We can now represent the structure of argument 4 as follows.

Logical Form 4:

(1) If p, then either q or r.
(2) It is not the case that q.

(C) If p, then r.

Now return to argument 4′. There are again three basic sentential structures present in sentences (1), (2), and (C) of argument 4′, which we will assign to atomic sentence-letters as follows.

Translation manual for argument 4′:

$p \Longrightarrow$ Katherine is home

$q \Longrightarrow$ Katherine left work early

$r \Longrightarrow$ It is Sunday

With this translation manual in place, we can now see that Logical Form 4 translates both argument 4 and argument 4′. So, arguments 4 and 4′ share the same form. Because argument 4 is valid and argument 4′ shares the same logical form, it follows that argument 4′ is valid. Indeed, regardless of what the atomic sentence-letters 'p', 'q', and 'r' represent (i.e., what English, or any other language, sentences they translate) and regardless of their actual truth values, any argument with the above represented form is valid. This is because the validity of an argument in English is due to the logical forms of the sentences

involved in that argument.

2.2 Translating Truth-Functions Into the Language of Propositional Logic

The sentences in Logical Form 4 contain English words (like 'if, then' and 'not') as well as atomic sentence-letters (like 'p' and 'q'). Only the latter are vocabulary in the formal language of PL. So these sentences are sentences of neither PL nor English, but are instead *hybrid* sentences, containing vocabulary from two different languages. Ultimately we want to learn to translate sentences in English into the language of PL. But it is helpful to break the translation process up into two stages, first translating our target sentences into the above hybrid language, leaving untranslated the *logic words* of the target sentences, and then translating the hybrid sentences into the language of PL.

We now discuss this second step, beginning in general terms how to translate truth-functional logic words, and then return, at the end of this chapter, to our translations of arguments 4 and 4′. There is no way around it; the material in the remainder of this chapter is hard going, especially as its connection to the earlier discussions of ordinary arguments and tests of validity and invalidity may seem distant indeed. We ask for your forbearance in working through this material and your trust that the material is both relevant and necessary for our ultimate goal of developing rigorous methods of establishing validity and invalidity. By the middle of chapter 3, the pieces will be tied together.

Propositional Logic is the logic of what are called *truth-functional operators*, which are expressed by words like 'not', 'and', 'or', 'if, then', and 'if and only if'. Grammatically, these expressions operate on sentences to create new sentences. So, for example, 'and' operates on a pair of simpler sentences ('George is sitting' and 'Bill is eating', for example) to create more complex sentences (like 'George is sitting and Bill is eating'). They are *truth-functional* operators as their meanings are functions on the truth values of the simpler sentences that compose the complex sentences they help form. We will focus on five truth-functional operators, which correspond to the five logic words identified above. The following hybrid sentences are examples of each of these forms of sentences.

English Truth-Functions

Negation: It is not the case that A.

Conjunction: A and B.

Disjunction: Either A or B.

Conditional: If A, then B.

Bi-Conditional: A if and only if B.

We can now introduce symbols for these truth-functional operators, as follows, which are included in the vocabulary of the language of PL.

Symbols for Truth-Functional Operators

Negation: $\neg A$

Conjunction: $A \wedge B$

Disjunction: $A \vee B$.

Conditional: $A \rightarrow B$

Bi-Conditional: $A \leftrightarrow B$

Here is some useful terminology.

– A and B in the above table are *metalanguage variables*. These symbols are not sentences of the language of PL but instead stand proxy for such sentences; they are place holders *for any* sentence of PL, however simple or complex. A sentence is an **instance** of a schema stated using those variables. So, for example, both '$\neg p$' and '$\neg\neg p$' are instances of $\lceil \neg A \rceil$, A standing proxy for 'p' in the first and '$\neg p$' in the second.

– The component parts of a conjunction are called *conjuncts*, the first the *left conjunct* and the second the *right conjunct*.

–The component parts of a disjunction are called *disjuncts*, the first the *left disjunct* and the second the *right disjunct*.

–The first component of a conditional sentence is called the *antecedent* and the second component the *consequent*.

–The first component of a bi-conditional sentence is called the *lefthand side [of the bi-conditional]* and the second component the *righthand side [of the bi-conditional]*.

– The operator '$\neg$' is a *one-place operator*, in the sense that it operates on a single sentence A to create a more complex sentence $\ulcorner \neg A \urcorner$. The other operators are *two-place operators*, in the sense that they operate on ordered pairs of sentences A and B (although they may operate on a pair of the same sentence, as '$p \wedge p$', for example, is an instances of the conjunction schema above).

Earlier in this chapter we described a two-step process of translating sentences of English into the language of PL. At the first step, we identify all of the atomic sentences in the target sentences, create a translation manual, and translate those target sentences into a hybrid language with atomic sentence-letters and English logic words. At the second step, we translate those remaining logic words into the language of PL, giving us the ultimate translation of the target sentences into the language of PL. We illustrate this with the following four examples. We will use the same translation manual for all four sentences, which is provided in the first table below, after the list of the target sentences to be

translated. Then, in the second table, we translate those sentences first into hybrid language in the middle and then the language of PL on the right.

(A) If George is sitting, then Sally is home.

(B) Sally is home or it is raining.

(C) George is sitting if and only if Bill is standing.

(D) George is sitting and it is not raining.

Translation Manual

$p \Longrightarrow$ George is sitting
$q \Longrightarrow$ Sally is home
$r \Longrightarrow$ It is raining
$s \Longrightarrow$ Bill is standing

Sample Translations

(A) If p, then $q \Longrightarrow p \rightarrow q$

(B) q or $r \Longrightarrow q \vee r$

(C) p if and only if $s \Longrightarrow p \leftrightarrow s$

(D) p and not $r \Longrightarrow p \wedge \neg r$

These four examples illustrate the basics of the translation of English sentences into the language of PL and, piggybacking on your antecedent knowledge of the meanings of the English logic words, provide insights into the meanings of the five logical operators of that language, although you will likely see that your prior understanding of these meanings are likely partial and perhaps even mistaken.

The table above labeled 'Symbols for Truth-Functional Operators' contains what we can call the canonical English expressions of these operators. So, for example, the canonical expression of negation is 'it is not the case that' and the canonical expression of the material conditional is 'if ..., then ...'. But there are other ways of expressing these functions available in English, as well as a number of complications worth mentioning.

The phrase 'it is not the case that', while certainly grammatical and meaningful, is very wooden and rarely, if ever, used in ordinary speech. When was the last time you uttered or heard the sentence 'It is not the case that George is sitting'? But 'Not George is sitting' is simply ungrammatical. The word 'not' in English typically modifies predicates, not complete sentences. So, the more acceptable way to express the thought behind the earlier strings of words is with the sentence 'George is not sitting'. (And, of course, the negation is typically contracted, so we would normally say, "George isn't sitting.") Because all of

the operators in PL are sentential operators, operating not on predicates, which don't have truth values, but only on full sentences, we must rewrite all negations in the stilted form. This is a fairly straightforward transition to make and 'It is not the case that George is sitting' and 'George isn't sitting' pretty clearly have the same literal meanings.

A further complexity with respect to negations arises from the fact that one can negate in English by using prefixes like 'non', 'de', 'un', etc. So, the sentence 'George is a nonhuman' should be treated as the negation of the sentence 'George is a human', lest the logical connection between these two sentences be lost in our translation. It would be incorrect to translate either 'George is not a human', 'George isn't a human', or 'George is a nonhuman' as an atomic sentence-letter in the translation manual. Instead, assuming 'r' has not already been assigned, we should assign the atomic, unnegated sentence 'George is a human' to the atomic sentence-letter 'r' and translate all three sentences as '$\neg r$'. Finally, and most difficult to judge, there are contrary pairs of sentences neither of which appear as negations, even in the covert way in which 'George is nonhuman' appears as a negation, which are best translated using '$\neg$'. Consider, for example, the sentences 'George is sitting' and 'George is standing'. The sentences are contraries, as standing logically excludes sitting. A proper translation of these sentences should preserve that logical connection. (Some may dispute that the sentences are really *logically* contraries, as the excluding relation resides in the nonlogical meaning of the predicates 'is standing' and 'is sitting'. While we think that this concern has merit, we will set it aside and continue on the assumption that our two sentences are logically contraries.) Doing so requires converting one of them into the negation of the other. So, the pair should first be transformed into the sentences 'George is sitting' and 'George is not sitting' before being translated into the language of PL as 'r' and '$\neg r$'. There aren't hard and fast rules about when an overtly positive sentence should be translated as a negation; one must rely on one's intuitive sense of when the original contrary sentences being translated are contradictory.

The expression 'and' is more complicated. In surface form, it does not always conjoin two complete sentences, even when it is clear that that is its true function. The sentence 'Both George and Bill are teachers' is an example. On the surface, 'and' conjoins two noun phrases, 'George' and 'Bill'. However, our sentence is clearly expressively equivalent to 'George is a teacher and Bill is a teacher', in which 'and' conjoins two sentences. It would be inappropriate, then, to translate the original sentence with a single atomic sentence-letter, as that translation masks some of the truth-functional structure present in the original target sentence. Instead, we should assign each component atomic sentence 'George is a teacher' and 'Bill is a teacher' atomic sentence-letters in the translation manual, assigning, say, the first to 'p' and the second to 'q', and translate the sentence as a conjunction, '$p \wedge q$'.

Some occurrences of 'and' *do not* conjoin two separate sentences, but instead form a complex plural term. A clear example of this is the sentence 'George and Bill lifted the piano together'. This sentence should *not* be translated as the conjunction 'George lifted the piano and Bill lifted the piano', as that last

sentence is true in a circumstance in which George goes to the piano and lifts it, then, as he is returning to his seat, Bill comes to the piano and lifts it, neither lifting the piano together. Instead, the whole sentence should be treated as a truth-functional atom, void of any (truth-functional) logical complexity. The addition of indicating an word like 'together' in the joint-action sentence serve as a helpful guide. But sometimes you must make a decision. For example, is the sentence 'George and Bill have children' a conjunction (being equivalent to 'George has children and Bill has children') or a plurality (being equivalent to 'George and Bill have children together', which is false in cases in which all of George's children are distinct from Bill's children, even though both have their own children)? The sentence can be interpreted in either way and you must make a decision in translating it. If taken as a genuine conjunction, then, where 'p' translates 'George has children' and 'q' translates 'Bill has children', the sentence is translated as '$p \wedge q$'. If, on the other hand, the occurrence of 'and' is taken to conjoin 'George' and 'Bill' to make the plural noun phrase 'George and Bill', then, where 'r' translates 'George and Bill have children [together]', the sentence is translated simply as 'r'. (The language of PL is not rich enough to deal with complex plurals like 'George and Bill', even though the logic of PL's conjunction '$\wedge$' should serve as a model for understanding the construction of these terms. The issues surrounding the logic and associated metaphysics of these plural expressions are complex, controversial, and deeply interesting, but are beyond the scope of an introductory logic course and so we will set them aside.)

The canonical way to form a conjunction in English is with the word 'and', but there are other ways as well, like with the word 'but', as demonstrated in this very sentence. It is standard to regiment 'but' as sentential conjunctions formalized with the symbol '$\wedge$'.

> This is controversial. Consider the intuitive difference between the sentence 'George is sitting and Bill is standing' and 'George is sitting but Bill is standing', where the latter suggests a contrast and perhaps inappropriateness to Bill's position lacking in the first sentence. Those who defend formalizing both sentences as conjunctions with '$\wedge$' claim that these coloring differences are not part of the strict and literal meaning of the sentences and so not part of the semantics of the sentences. Instead, these differences are to be accounted for as part of a pragmatic theory of the uses of sentences in real life, complex conversations. Capturing the logical relations between sentences of a natural language like English is done at the level of the strict and literal meaning sentences, a level at which the above two sentences are equivalent. While these issues are still lively debated, for the purposes of an introduction to logic class, we suggest you translate 'but' as '$\wedge$'. Any account that would capture the extra suggested meaning of 'but' would, by necessity, not treat 'but' as a pure and simple truth-functional operator.

Other ways of expressing a conjunction in English include 'although' (as in 'George ate his potatoes, although not his peas'), 'nevertheless' (as in 'I stumbled on my lines; nevertheless, I got the part'), 'yet' (as in 'George doesn't like

Susan, yet he gave her the job'), 'furthermore' (as in 'George ate his potatoes; furthermore, he ate his peas, too'), 'whereas' (as in 'George doesn't work with Susan whereas he does work with Jane'), and 'however', all of which should be translated using '$\wedge$'.

There is a complexity concerning uses of 'and' in ordinary conversation that we should note. Compare the sentence 'They fell in love and got married' with the sentence 'They married and fell in love'. Intuitively, an utterance of the second sentence carries the suggestion that the couple fell in love only after their wedding, while an utterance of the first carries the suggestion that the couple were in love on their wedding day. Often, although not in all cases, uses of 'and' carry a suggestion of temporal order, meaning something along the lines of 'and then'. As we will see in the following chapter when we discuss the semantics of the five truth-functional operators of PL, '$\wedge$' is insensitive to the temporal order of the reported events and, indeed, any mechanism that is so sensitive cannot be purely truth-functional and so exceeds the resources available in the language of PL. While it is still a live issue whether or not these facts prove that 'and' is not a purely truth-functional connective, as opponents of a classical account of 'and' insist, for the purposes of an introductory logic course, we suggest that you simply translate such sentences using '$\wedge$' while being sensitive to and acknowledging the shortcomings of that translation. (Proponents of the traditional view that the meaning of 'and' is adequately captured with the truth-functional connective '$\wedge$' claim that information concerning the temporal order of the reported events is not part of the strict and literal semantics of the language but instead part of the information that is conveyed by utterances of those sentence by pragmatic mechanisms.)

The material conditional is by far the most difficult of the truth-functions to translate from English into the language of PL. To indicate some of the difficulties, notice that the following sentences are all interpreted into the language of PL in the same way, as '$p \rightarrow q$', where 'p' translates 'George is sitting' and 'q' translates 'George is eating'.

(1) If George is sitting, then George is eating

(2) George is eating if George is sitting.

(3) George is sitting only if George is eating.

(4) George is eating when George is sitting.

(5) George is eating provided George is sitting.

The 'if'-clause, the sentence 'George is sitting' in sentences (1)-(5) above, is called the **antecedent** and the 'then'-clause, the sentence 'George is eating' in sentences (1)-(5), the **consequent**. We can see, by comparing sentences (1) and (2), for example, that in English, sometimes the antecedent comes first and sometimes second. However, in the language of PL, the antecedent *always* comes first, being on the left of '$\rightarrow$'. It would, then, be incorrect to translate either of the above sentences as '$q \rightarrow p$', holding fixed the earlier translation manual.

(Unlike '$\wedge$', the order of the component sentences of a material conditional matter. In general, $\ulcorner A \rightarrow B \urcorner$ has a different meaning from $\ulcorner B \rightarrow A \urcorner$.) The function of the antecedent is to *conditionalize* the consequent. We can think, then, of sentences (1)-(5) as saying that George is eating, conditional on his sitting; that is, *given* that George is sitting, then he is eating. We'll discuss the semantics of '$\rightarrow$' in the next chapter, but it will help you translate natural language conditionals like (1)-(5) into the language of PL, and in particular correctly order the sentences, if you ask yourself the following question: Which component sentence is being asserted to be true conditional on the truth of the other? The sentence being asserted to be conditionally true is the *consequent* of the conditional and so should be placed to the right of '$\rightarrow$'. The sentence conditioning the other's truth is the antecedent. In (1)-(5), for example, George's eating is being asserted to be true conditional on his sitting. So, 'George is eating' is the consequent and 'George is sitting' is the antecedent in all five sentences.

Earlier we said that 'although' expresses a conjunction while 'provided' expresses a material conditional. We can use the above idea of one sentence's truth being conditional on the truth of another sentence to see why this is correct. Contrast 'Although I hate peas, my mom will make me eat them' with 'Provided I hate peas, my mom will make me eat them'. The first is true only if I both hate peas and my mom makes me eat them. It thus functions primarily as a conjunction. (The words 'although', unlike 'and', however, also suggests a contrast of some form between the two conjuncts. We have suggested that this difference is not part of the strict and literal meaning of the word and so both 'although' and 'and' should be translated into the language of PL as '$\wedge$'.) The word 'provided', however, functions as a conditionalizer. My mom's forcing me to eat peas is claimed to be conditional on my hating them. So, the sentence 'Provided I hate peas, my mom will make me eat them', unlike its 'although' counterpart, is true in the circumstances in which I like peas and my mom does not make me eat them; that is to say, my mom's forcing me to eat peas is not required for the truth of the second sentence while it is required for the truth of the first. This difference between the two sentences shows that the 'although' sentence should be translated with '$\wedge$' and the 'provided' sentence should be translated with '$\rightarrow$'.

The word 'unless' also expresses a material conditional. For example, the sentence 'George is eating unless he is working on his logic problem' (as well as the sentence 'Unless George is working on his logic problem, he is eating') can be paraphrased as 'If George is not working on his logic problem, then George is eating'. Letting 'q' still translate 'George is eating' and 'r' translate 'George is working on his logic problem'. We can then translate the sentence into our hybrid language as 'If it is not the case that r, then q', which is then translated into the language of PL as '$\neg r \rightarrow q$'. The 'unless'-clause is the antecedent of the sentence and the 'unless'-clause conditionalizes the truth of the consequent, which is the sentence's main clause.

The following table provides a helpful guide to translating the less straightforward English logic words that express material conditionals.

Translating Material Conditionals

A only if $B \Longrightarrow A \rightarrow B$

A if $B \Longrightarrow B \rightarrow A$

A when $B \Longrightarrow B \rightarrow A$

A provided $B \Longrightarrow B \rightarrow A$

A unless B/ Unless B, $A \Longrightarrow \neg B \rightarrow A$

We suggest that you consult this table when translating English sentences into the language of PL as you work on your translation problems as you work on your translation problems.

PL has limited expressive power. We have encountered some of these limitations above in our discussion of conjunctions, but the expressive limitations are more evident with conditionals. Consider the sentence 'Had George W. Bush not been elected president in 2000, then the US would not have invaded Iraq'. One could make a convincing case that that sentence is true. After all, had Bush not been elected, then Gore would have been and a Gore administration would not have been as hawkish as the Bush administration. By contrast, 'Had George W. Bush not been elected president in 2000, then Bush would have been elected president in 2000' is intuitively false. But, because 'George W. Bush was not elected president in 2000' is false, if we regiment our sentences as material conditionals using '$\rightarrow$', then both sentences are vacuously true. (This may not be immediately obvious. We will discuss the truth conditions of material conditionals in section 1 of chapter 3 below.) These sentences are *counterfactual* conditionals, marked by the subjunctive mode, and counterfactual conditionals cannot be adequately treated with the resources of PL (or, indeed, with the resources of any extensional logic). Furthermore, it is very natural to interpret the sentence 'If I open the door, then the handle falls off' as expressing something stronger than a mere truth-functional connection between a particular event of my opening the door and the handle falling off. It is natural to read it as a universalization, made more explicit as 'Every time I open the door, the handle falls off'. This, too, cannot be adequately treated with the resources of PL. (An adequate treatment requires quantification over either times or events. Both require a nonextensional logic provided the events are typed, as they must be.) These expressive limitations are only genuine problems when one expects too much of PL. PL is the logic of truth-functions and so it should not be surprising that it is ill-equipped to handle phenomena that are not purely truth-functional.

The phrases 'even if' and 'whether or not' merit special attention. They typically accompany a subjunctive mood, or at least a future tense, where there is some kind of uncertainty or unsettledness in play. For example, consider the sentences 'I would have come to your party even if that jerk had come' and 'I will come to your party whether or not that jerk comes'. The subjunctive mood of the verb ('would have' and 'had') mark that the first sentence is not purely

truth-functional. The most appropriate uses of the first sentence are ones in which the antecedent of the conditional is actually false; so, the most natural circumstances in which one would utter that sentence are ones in which the jerk in question is not at the party. As we mentioned above, counterfactuals, and subjunctives more generally, cannot be treated with the purely truth-functional resources of PL. So, although there is clearly a conditional operator embedded in the sentence, and standard accounts of counterfactual conditional employ the resources of PL and the logic of '$\rightarrow$', additional machinery is needed to adequately account for the logical affects of the subjunctive and so, when only the resources of PL are available, it is necessary to treat the entire sentence as a logical atom and so assign it an atomic sentence-letter in the translation manual. By contrast, when the subjunctive mood is removed, giving us the sentence 'I came to your party even if that jerk came', then the sentence should be translated as a conditional, '$q \rightarrow p$', where 'p' translates 'I came to your party' and 'q' translates 'That jerk came to your party'. This entails, then, that this most recent sentence has exactly the same translation into the language of PL as the sentence 'If that jerk came to your party, then I came to your party', as, given the same translation manual, this last sentence is clearly to be translated as '$q \rightarrow p$'. This may seem counterintuitive, but insofar as 'even if' expresses a truth-functional connective, this is as it should be. The intuitive difference between the two sentences should be explained pragmatically and not in terms of difference semantic meanings. Also the natural home for 'even if' is with counterfactuals, as we have noted above.

Let's turn now to 'whether or not' and the second of the sentences mentioned at the beginning of the previous paragraph. A fair paraphrase of this sentence seems to be 'I will come to your party if that jerk comes and I will come to your party if that jerk doesn't come'. Reflection reveals that, insofar as that jerk is either coming or not coming, this is truth-conditionally equivalent to 'I will come to your party'. (Even though the original sentence may seem to contain a disjunction, as it contains the word 'or', again, reflection on the meaning of the sentence suggests that it is actually a conjunction of two conditionals.) The sentence is complex because of the future tense, but if we ignore the tense, as we must for the purposes of translating into the language of PL, we can see, then, that sentence is a conjunction of two conditionals, each with a common consequent, in our case, the sentence 'I will come to your party', and one with a given sentence as its antecedent, in our case, the sentence 'That jerk will come to your party', and the other the negation of that same sentence as its antecedent. We must wait until the following section to develop the resources to write a sentence with this complexity in the language of PL and so we will return to this sentence below.

Our last truth-functional operator, '$\leftrightarrow$', translates 'if and only if'. For example, the sentence 'Bill is at home if and only if he is not working' is translated as '$p \leftrightarrow \neg q$', letting 'p' translate 'Bill is at home' and 'q' translate 'Bill is working'. Other ways of expressing a bi-conditional include 'exactly when', as in 'Bill is at home exactly when he is not working', and 'just in case', as in 'Bill is at home just in case he is not working', both of which have the same translation into the

language of PL. Oftentimes the difference between a simple material conditional and a material bi-conditional is not given sufficient care, the speaker using a material conditional when she in fact really intends to express a bi-conditional or *vice versa*. The material conditional ⌜A only if B⌝, which is translated, remember, as ⌜$A \rightarrow B$⌝, says, in effect, that the truth of B is *necessary for* the truth of A while the truth of A is *sufficient for* the truth of ⌜B⌝. (More generally, a material conditional says, in effect, that the truth of the consequent is necessary for the truth of the antecedent and the truth of the antecedent is sufficient for the truth of the consequent. We will see this more clearly in chapter 3, when we discuss the truth definition of '$\rightarrow$'.) By contrast, the material bi-conditional ⌜A if and only if B⌝ says, in effect, that the truth of A is both *necesary and sufficient for* the truth of B and *vice versa*. In that way, the bi-conditional is stronger than the conditional. Appreciating the difference and so being in a position to be precise about what one's intended message is and so which words to use to best express it requires understanding the difference between the material conditional and bi-conditional (as well as the differences between these and universalized conditionals and subjunctive conditionals, briefly mentioned above), which will have to wait until chapter 3, when we present the semantics for the language of PL. For the purposes of translating sentences of English into the language of PL, however, one can simply translate the words used: 'If, then', 'only if', 'given that', etc., are all material conditionals that should be translated using '$\rightarrow$', while 'if and only if', 'exactly when', etc., are material bi-conditionals that should be translated using '$\leftrightarrow$'.

2.3 The Syntax of the Language of PL

We began this chapter wanting to translate the sentences in arguments 4 and 4′ into the language of PL. We now have most of the necessary material to do just that, but we are missing one important ingredient whose absence can be brought out by considering the translation of (1) into the hybrid language as 'If p, then q or r'. One may be initially tempted to simply translate this as '$p \rightarrow q \vee r$'. After all, we already said that 'if, then' is a material conditional symbolized as '$\rightarrow$' and 'or' is a disjunction symbolized as '$\vee$'. However, this string is *not* a sentence of the language of PL. A sentence with multiple occurrences of operators like '$\rightarrow$' and '$\vee$' must contain parentheses in order to avoid ambiguity and imprecision and, given the syntax for the language of PL to be described below, to even count as a sentence of the language at all. So, we must discuss the use of parentheses in the formation of sentences of PL. Once we have discussed these matters, we will be in a position to finally translate the sentences in arguments 4 and 4′ (as well as a great many other sentences!) into the language of PL. We will also be in a better position to discuss the formal semantics of PL and truth tables, the subject of the following chapter.

You are already familiar with the idea of parentheses and "order of operation" from basic arithmetic. Consider the arithmetical expression '2+3*5', ignoring conventions governing the order of operations. What is its value? 25

or 17? It is (absent the disambiguating conventions we asked you to set aside) undefined, as there are in fact two distinct functions that the term can be taken to express, as follows: '(2+3)*5' and '2+(3*5)'. The first is primarily a multiplication function (with an addition function used to determine its first argument) and the second primarily an addition function (with a multiplication function used to determine its second argument). A true sentence results from adding '=25' to the first and a false sentence results from adding the same to the second. But adding '=25' to the original, ambiguous expression results in neither a true nor false sentence.

The same ambiguities and imprecision arise in sentences containing multiple truth-functional operators. Return to (1) of argument 4, 'If George is sitting, then George is eating or George is working on his logic problem'. There are two ways to interpret this sentence and it is perfectly possible for one to be true and the other false. On one understanding, the sentence is primarily a conditional sentence with a disjunction as its consequent. We can encourage this reading by inserting 'either' after 'then', as follows: 'If George is sitting, then *either* George is eating or George is working on his logic problem'. On the second understanding, the sentence is primarily a disjunction with a conditional as the left disjunction. We can encourage this understanding with the following addition: 'If George is sitting, then George is eating, or George is working on his logic problem'. If we let 'p' translate 'George is sitting', 'q' translate 'George is eating', and 'r' translate 'George is working on his logic problem', we can translate these two disambiguations of the original sentence with the following two sentences of our hybrid language (with parentheses), followed by their respective translations into the language of PL.

(a) If George is sitting, then either George is eating or George is working on his logic problem $\Longrightarrow$If p, then (q or r) $\Longrightarrow p \rightarrow (q \vee r)$

(b) If George is sitting, then George is eating, or George is working on his logic problem $\Longrightarrow$(If p, then q) or r $\Longrightarrow (p \rightarrow q) \vee r$

These two sentences of PL do not suffer from the ambiguities of our original English sentence.

The preferred interpretation of (1) is pretty clearly (a). But consider, by contrast, the following structurally similar sentence: 'If George is sitting, then George is eating and Obama is still president'. It seems extremely natural to interpret this sentence along the lines of (b), adjusting for the exchange of 'or' with 'and'; that is, as a conjunction with a conditional as a left conjunct. It is a very good question why the preferred interpretation for the former is (a) when the preferred interpretation for latter is along the lines of (b), given that both original sentences have such similar surface structures. One might argue that the processing of syntactic forms, meanings, and speaker intentions do not run independently, the output of the first feeding into the second and the output of the second feeding into the third without any further feedback or interference.

That is, in interpreting others, we don't first grasp the syntactic form of the sentences she utters, then assign meanings to those words, and then wonder what she could have intended by saying that in the context, holding fixed at each later step the determinations of the earlier. Instead, because it is so implausible to attribute to a speaker the thought that Obama's being president is conditional in any way on George's sitting, we charitably interpret the person as both informing us about George, as one unit of information, and additionally about Obama, as another unit of information, and so interpret the syntactic form appropriately, as a conjunction whose left conjunct is a material conditional, in light of the meanings of the words used and the topic of discourse, which are of course different between the two sentences. These issues are very complex and beyond the scope of introductory and intermediate logic classes. A similar phenomena can be seen in so-called garden path sentences, like 'We convinced her children are messy'. Most readers begin the sentence reading 'her children' as a unit but then go back and revisit that reading when it leads to (grammatical) nonsense, inserting, on a second reading, a hidden 'that' after 'her'.

We will now present a syntax for the language of PL. The syntax of a language tells us the conditions under which a string of symbols from the vocabulary of that language constitute a sentence of the language. The vocabulary of PL consists of the truth-functional connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$, an infinite stock of atomic sentence-letters $\{p, q, r, s, p_1, q_1, r_1, s_1, \ldots\}$, and parentheses '(' and ')'. Earlier discussions have likely already put you in a position to recognize, at least intuitively, that the string '$pq\vee$' is not a sentence of PL and so is ungrammatical, even though all of its elements are from the vocabulary of PL, while '$(r \rightarrow (p \vee q))$' is a sentence of PL. The following will make explicit why that is the case.

We define the notion of a *sentence of the language of PL* by recursion. Recursive definitions are widely used in philosophy, mathematics, and computer science. A **recursive definition** characterizes a set of items in terms of that very set. We start with items stipulated to belong to the set our definition characterizes and then define the other items in the set in terms of items already determined to be in that set. One can think of a recursive definition as first providing some basic examples, the base cases, and then providing a recipe for constructing further items that the notion applies to from the base cases, where the recipe can be applied to the results of earlier applications, and then applied again to those results, and so on indefinitely. (The set of items so constructed needn't be infinite, although an obvious advantage of recursive definitions is that they can nicely and succinctly characterize sets of infinitely many objects.) In our case, the recursive definition will characterize the set of all and only sentences of PL. Our definition will tell us why the string '$(r \rightarrow (p \vee q))$' is a sentence of PL while the string '$pq\vee$' is not.

The simplest example of a recursive definition is the following definition of a positive integer. Base case: 1 is a positive integer; Recursive step: For all n, if n is a positive integer, then $n + 1$ is also a positive integer; Extremal clause: Nothing else is a positive integer. Here, we are told that 1 is a positive integer as

our base case. We then apply our recursive step to our base case by adding 1 to 1, and we get that the result, i.e., 2, is also a positive integer. We can now repeat the recursive step to 2, again adding 1, and the result, i.e., 3, is also a positive integer. We are now off to the races, continually applying the recursive step again to the output of our previous application. (By way of contrast, consider the following characterization of the set of all and only perfect squares as the set of all numbers x such that $x = n * n$, where n is any integer. This, while a perfectly good definition, is *not* a recursive definition.) Our recursive definition of the notion of *a sentence of PL* will take a form similar to the above definition of positive integer.

The base case of our definition of the notion of *a sentence of PL* are the atomic sentence-letters.

> **Base case**: Every atomic sentence-letter is a sentence of PL.

The fact that atomic sentence-letters are the base case of sentences of PL proves how fitting the name "atomic sentence-letter" is; the atomic sentence-letters are the atoms from which every sentence of PL is ultimately built. The next part of our definition, the recursive step, provides procedures for constructing more complex sentences of PL from simpler sentences of PL using the logical operators '$\neg$', '$\wedge$', '$\vee$', '$\rightarrow$', and '$\leftrightarrow$'.

> **Inductive clause**: For every string of symbols A and B, if A and B are sentences of PL, then $\ulcorner\neg A\urcorner$, $\ulcorner(A \wedge B)\urcorner$, $\ulcorner(A \vee B)\urcorner$, $\ulcorner(A \rightarrow B)\urcorner$, and $\ulcorner(A \leftrightarrow B)\urcorner$ are all sentences of PL as well.
>
> **Extremal clause**: Nothing else is a sentence of PL.

We will explain in plain(er) English what this means and work through some examples in which we apply the definition to strings of symbols to determine whether or not those strings are sentences of PL.

2.4 Construction Procedure for the Sentences of the Language of PL: The Simple Version

Let's conceive of this definition as encoding a procedure for constructing the sentences of PL. At the start of this procedure, our bin of sentences of PL is empty. We follow the recipe to fill that bin with all and only sentences of PL. The base case tells us that, at the first stage of our construction procedure, every atomic sentence-letter is a sentence of PL. So, 'p', 'q', 'r', and all their friends are sentences of PL and so placed in our bin at this stage. To simplify, however, we will pretend that we *only* have these three atomic sentence-letters. In fact, there are infinitely many atomic sentence-letters and so infinitely many sentences of PL in the bin of sentences of PL at the first stage of construction. (Adequately describing the construction procedure in full generality is not straightforward, as we immediately hit difficulties with infinite sets of items. That is why we here make the simplifying assumption that the only atomic sentence-letters are 'p', 'q', and 'r', in an effort to avoid those issues while intro-

ducing the grammaticality rules for the language of PL. In the appendix to this chapter, we describe the construction procedure in full detail and discuss some of the puzzles concerning infinity that description gives rise to. The discussion is of interest to computer scientists and touch on issues concerning the halting problem and ill-formed procedure instructions that endlessly loop.) At the second step of the construction, we apply the recursive clause to the items in the bin at the end of the first step. Given our simplifying assumption, then, we apply the recursive clause to 'p', 'q', and 'r' to create more complex sentences of PL.

The construction instructions encoded by the inductive step really have two parts, the first concerning '$\neg$' and the second the other four truth-functional operators. The first part tells us to perform the following operation at this stage: Take an atomic sentence-letter and prefix it with '$\neg$' and the resulting string is a sentence of PL. Given our simplifying assumption that 'p', 'q', and 'r' are the only atomic sentence-letters, this means that we will have three negations at the end of the first part of the first application of the inductive step, '$\neg p$', '$\neg q$', and '$\neg r$'. For the second part of the first application of the inductive step, we construct all of the conjunctions, disjunctions, material conditionals, and material bi-conditionals from our three atomic sentence-letters. Our instructions tell us to take any pair of strings that have been determined at the end of the first step to be sentences of PL and concatenate one of those strings, exactly one occurrence of one of our four two-place operators '$\wedge$', '$\vee$', '$\rightarrow$', or '$\leftrightarrow$', followed by the other string, enclosing the result with a '(' on the far left and ')' on the far right, and the resulting string is a sentence of PL. (To "concatenate" strings is to combine them together into a single string.) Consider first the conjunctions. There are nine resulting conjunctions: '$(p \wedge p)$', '$(p \wedge q)$', '$(p \wedge r)$', '$(q \wedge p)$', '$(q \wedge q)$', '$(q \wedge r)$', '$(r \wedge p)$', '$(r \wedge q)$', and '$(r \wedge r)$'. Notice that parentheses enter the language for the first time at this point. More generally, parentheses *always* are added whenever a two-place truth-functional operator is added and that is the *only* way for them to enter into a sentence. We want to be sure to get *all* of the different permutations of atomic sentence-letters and '$\wedge$', without prejudicing anything about the meaning or use of those sentences. So, for example, even though there may well be no "use" in uttering the sentence '$(p \wedge p)$', for example, we still want it to count as a sentence and so want our definition of the notion of a sentence of PL to apply to it. We do the same for the other three truth-functional operators, giving us nine new sentences for each, constructing strings like '$(p \rightarrow q)$', '$(r \leftrightarrow q)$', and '$(q \vee r)$', all of which are sentences of PL constructed at the first application of the recursive definition. By the end of this second stage, we will have constructed 36 sentences with a two-place truth-functional operator and three negations. These 39 sentences are all of the sentences (constructed from the three atomic sentence-letters 'p', 'q', and 'r') with a logical complexity of 1, where the **logical complexity** of a sentence is determined by the number of occurrences of truth-functional operators the sentence contains. (So, to figure out the logical complexity of a sentence, one simply counts the number of (occurrences of) truth-functional operators there are in the sentence.) Combining them with the original three

atomic sentence-letters, there are then 42 sentences of PL after the completion of the first application of the recursive step of the definition of the notion of sentence of PL. At the completion of this step, the only strings determined to be sentences of PL are the atomic sentence-letters and strings with no more than one occurrence of one of the five of our truth-functional operators.

It is important to see that strings with a logical complexity greater than 1, i.e., sentences with more than one occurrence of a truth-functional operator, like the sentence '$\neg(p \wedge \neg q)$') are *not* constructed during the first two steps of this construction procedure. Such sentences are only constructed at later steps in the procedure. We can see why this is so by noting that the constituent parts of the sentence displayed above, in particular, the sentences '$(p \wedge \neg q)$' and '$\neg q$', have not all been constructed at the earlier step and so do not satisfy the condition, *at that step*, of being sentences of PL (i.e., they are not in the bin of sentences of PL constructed at the completion of the previous step). Remember, the recursive definition tells us to take something that has *already* been constructed as a sentence of PL and perform an operation on it (by either adding an occurrence of '$\neg$' to its front or concatenating it with another sentence, an occurrence of a two-place truth-functional operator, and enclosing the result in a pair of parentheses). We can't perform operations on items that we have yet to create!

We have described the first two steps of the "construction" of sentences of PL. The hallmark of a recursive definition is that we can continue to reapply the inductive clause again to the results of earlier applications. So, at the third step of the construction, we have both all atomic sentence-letters and all sentences of PL with logical complexity of no more than 1 as sentences of PL in our bin of constructed sentences. This means that we now have available sentences like '$(p \rightarrow q)$' and '$(q \vee p)$' as material for constructing new sentences. Instead of having available only three sentences of PL as building blocks, as we did at the first application of the recursive definition above, we now have available 42 sentences of PL as building blocks. So, when we reapply the instructions encoded by the inductive clause to this new set of sentences, we can construct sentences that were not generated at the earlier, second step. Here are some examples. Because '$(p \rightarrow q)$' is a sentence of PL constructed at the second step, by applying the inductive clause at the third step of the construction, we will be able to count '$\neg(p \rightarrow q)$' and '$((p \rightarrow q) \wedge (p \rightarrow q))$' as sentences of PL constructed at the third step. Neither of these sentences were constructed at the earlier second stage of the construction, as their logical complexity is too great. The first sentence has a logical complexity of 2 and the second a logical complexity of 3. So, after the third step of the construction, the strings of symbols that are determined to be sentences of PL are the atomic sentence-letters and all sentences (constructed in the above manner) with a logical complexity no greater than 3. By the end of this stage, there will be a total of 7059 new sentences, for a total of 7101 sentences of PL.

> You may be relieved to hear that we won't list all 7059 of these new sentences but will instead describe the new types of sentences that are constructed and give a few sample examples.

– First, we can construct 39 new negations that were not available at the earlier step 2. In particular, we construct double negations, like '$\neg\neg p$' and '$\neg\neg q$', as well as negations of the 36 two-place sentences, like '$\neg(p \wedge q)$' and '$\neg(r \rightarrow p)$'.

– Next, we can construct 936 new sentences that are concatenations of one of the original three atomic sentence-letters 'p', 'q', or 'r', with one of the 39 new sentences of logical complexity of 1 constructed at the earlier step 2 with one of the four two-place truth-functional operators, '$\wedge$', '$\vee$', '$\rightarrow$', or '$\leftrightarrow$'. Among these sentences are '$(p \rightarrow (q \vee r))$', '$((r \leftrightarrow p) \rightarrow p)$', and '$(r \vee \neg r)$'. These are the non-negation sentences with a logical complexity of 2. Here's how we get 936 new sentences. We start with the 39 new sentences of logical complexity of 1 constructed during the previous stage. We then get 39 new sentences by, for each of these sentences, concatenating 'p' with '$\wedge$' and the sentence and enclosing the result in a pair of parentheses. We then get 39 other sentences by doing the same thing in the reverse order. We then do the same thing for the other three two-place truth-functional operators. We then do the same thing for the other two atomic sentence-letters.

– Finally, we construct 6084 sentences with a logical complexity of three (sentences like '$((p \wedge p) \wedge (p \wedge p))$' and '$((p \rightarrow q) \leftrightarrow (q \vee r))$') as follows. First, list all of the 39 sentences with a logical complexity of 1 constructed at the previous second stage. Take the first sentence and, for each of those 39 sentences (including itself!), concatenate the two sentences with '$\wedge$' and enclose the result in a pair of parentheses. Then do the same thing in the reverse order. Then do the same thing for each of the three other two-place truth-functional operators. Then repeat the whole procedure for the second sentence and so on for all 39 sentences.

We will only very briefly describe the fourth step of the construction of sentences of PL. At this step, we have available as material for constructing new sentences all of the sentences of PL constructed in the above manner with a logical complexity no greater than 3. So, because both '$((p \rightarrow q) \wedge (q \vee p))$' and '$((q \vee p) \leftrightarrow (p \rightarrow q))$' have been constructed as sentences of PL at the early third step, by applying the inductive clause at the fourth step, we construct '$(((p \rightarrow q) \wedge (q \vee p)) \vee ((q \vee p) \leftrightarrow (p \rightarrow q)))$' as a sentence of PL constructed at the fourth step. This sentence has a logical complexity of 7. (And, of course, we can also construct sentences with a logical complexity of 4, 5, and 6 at this step of the construction of sentences of PL. For example, because '$((p \rightarrow q) \wedge (q \vee p))$' has been determined to be a sentence of PL at the third step, by applying the inductive clause, we construct at the fourth step '$\neg((p \rightarrow q) \wedge (q \vee p))$' (a sentence with a logical complexity of 4) as a sentence of PL. And, because both '$((p \rightarrow q) \wedge (q \vee p))$' and '$\neg(p \wedge q)$' have been determined to be sentences of PL at the third step, by applying the inductive clause, we construct at the fourth step '$(((p \rightarrow q) \wedge (q \vee p)) \rightarrow \neg(p \wedge q))$' (a sentence with a logical complexity of 6) as a sentence of PL.)

This procedure is continued indefinitely, constructing sentences with indefinitely more logical complexity, although you will no doubt be happy to hear that we are stopping the description here, leaving you free to carry out the rest on your own. For each step n in the construction, at the completion of n, sentences of PL with a logical complexity up to $2n - 1$ will be constructed. To get to the next greatest logical complexity, we must apply the construction

procedure encoded by the inductive clause to the set of sentences completed at step n, which will take us to step $n + 1$ in the construction.

2.5 Testing for Grammaticality

In the previous section we illustrated the recursive definition of the notion of a sentence of PL by construing that definition as a construction procedure for creating sentences of PL. We can use the construction procedure outlined above to test whether or not a given string is a sentence of PL by "reverse engineering" its construction, hence providing us with a **test for grammaticality**. We noted earlier that the string '$p \rightarrow q \vee r$' is *not* a sentence of PL. We can now see why that is so by seeing that it is at no point the product of the above construction procedure and, by the Extremal Clause of the definition of the notion of a sentence of PL, nothing that is not produced by that procedure is a sentence of PL. Because there are two occurrences of truth-functional operators in the string, both '$\rightarrow$' and '$\vee$', the string is not a sentence of PL in the base case and is not a sentence of PL after the first application of the recursive step, as it has a logical complexity of 2. It must instead be created, if at all, at the second application, by combining two sentences created at the first stage with an occurrence of a truth-functional operator. Assume, for *reductio*, that the string is a sentence created at the second stage of construction. There are then two options of which logical operator is added at this stage: The leftmost '$\rightarrow$' or the rightmost '$\vee$'. Suppose first that we used the leftmost operator '$\rightarrow$', concatenating 'p' and '$q \vee r$'. Then both of those strings must be constructed by the first stage of the construction process in order to satisfy the condition of the inductive clause. But '$q \vee r$' is not a sentence of PL at the first stage, as it contains an occurrence of '$\vee$' but no parentheses. So we could not have added the leftmost operator '$\rightarrow$' at the second and final stage of the construction. Suppose, then, that we used the rightmost operator '$\vee$', concatenating '$p \rightarrow q$' and 'r', in order to construct the string. Then both of those strings must be constructed by the first stage of the construction process. But '$p \rightarrow q$' is not a sentence of PL at the first stage, for much the same reason. So, we could not have added the rightmost operator '$\vee$' at the second stage of the construction. As those exhaust the options, there are no ways to employ the construction procedure encoded by the recursive definition of sentence of PL to generate '$p \rightarrow q \vee r$'. So, that string is not a sentence of PL.

By contrast, the string '$(p \rightarrow (q \vee r))$' *is* a sentence of PL and we can see why by again reverse engineering its construction. We "deconstruct" the string into parts of either pairs of sentences of PL, an outermost pair of parentheses, and a two-place truth-functional operator '$\wedge$', '$\vee$', '$\rightarrow$', '$\leftrightarrow$' or a one-place truth-functional operator '$\neg$' and a sentence, where, in either case, each of those constituent sentences similarly "deconstruct" ultimately into nothing but atomic-sentence letters. At the first stage, we drop the outermost parentheses and pull apart 'p', '$\rightarrow$', and '$(q \vee r)$'. The first and last are a pair of sentences of PL and the second a two-place truth-functional operator. The first is a sentence

of PL because it is an atomic sentence-letter. The last is a sentence of PL because it "deconstructs" in a similar way: Dropping the outermost parentheses, we can pull apart 'q', '$\vee$', and 'r', where the first and last are sentences of PL (because atomic sentence-letters) and the second a two-place truth-functional operator. So, the original string '$(p \rightarrow (q \vee r))$' is a sentence of PL.

To recognize grammaticality, we must determine, for any arbitrary string of symbols, whether or not that string is a sentence of the language of PL. Confronted, then, with any string of symbols, we want to be able to determine whether or not the string is a sentence of PL, according to the definition of the notion of a sentence of PL given above in section 2.2. There are several shortcuts we can take to this end that involve noticing patterns sufficient for making a string *not* a sentence of PL, as follows.

- If a string contains any items not in the vocabulary of PL, then it is not a sentence of PL. So, for example, the string '$((p \vee r) + 6)$' is not a sentence of PL, as the symbols '$+$' and '6' are not in the vocabulary of PL.
- If a string of symbols from the vocabulary of PL has atomic sentence-letters immediately adjacent to one another, as in the string '$(pq \vee r)$', then the string is not a sentence of PL.
- If a string has an occurrence of a two-place truth-functional operator [note the restriction to *two*-place operators; what follows is not true of occurrences of the negation operator '$\neg$'] that is either (a) immediately followed by an occurrence of a two-place truth-functional operator or an occurrence of a right parenthesis or (b) immediately preceded by an occurrence of a two-place truth-functional operator or an occurrence of a left parenthesis, as in the strings '$(p\vee \rightarrow q)$' and '$(\wedge(p \leftrightarrow))$', then the string is not a sentence of PL.
- If a string has an occurrence of '$\neg$' that is immediately followed by anything other than another occurrence of '$\neg$' (as in the sentence of PL displayed above), an atomic sentence-letter (as in the sentence '$\neg p$'), or a left parenthesis (as in the sentence '$\neg(p \wedge q)$', then the string is not a sentence of PL. This last claim entails that the following strings are *not* sentences of PL, as they contain occurrences of '$\neg$' immediately followed by a symbol other than these listed: '$\neg \wedge p$' and '$(p \rightarrow q\neg)$'.
- If a string has any parentheses at all and does not end with a right parenthesis '$)$', then the string is not a sentence of PL.
- If a string contains an uneven number of left and right parentheses, as in the string '$((p \wedge q)$', for example, then the string is not a sentence of PL.
- If the number of occurrences of two-place truth-functional operators in a string does not equal the number of pairs of left and right parentheses in the string, then the string is not a sentence of PL. So, for example, all of the following strings are not sentences of PL, the first string having too many parentheses and the second too few: '$((p\vee q))$' and '$(p\vee(q \rightarrow r\wedge p))$'.

These seven negative heuristics provide quick ways to determine that a string is not a sentence of PL. They are all justified by the fact that there is no way to follow the procedure for constructing sentences of PL described above in section 2.3 to produce sequences of symbols that meet the conditions described above (having an uneven number of left and right parentheses, having an occurrence of '$\neg$' followed by an occurrence of '$\vee$', etc.). However, when it comes to showing that a string *is* a sentence of PL, one must show that it is the product of the construction procedure contained in the recursive definition of the notion of a sentence of PL described above. And the surest way to do that is to go through, step by step, the procedure that resulted in its construction.

Prior to our introduction of the recursive definition of the notion of a sentence of PL, we said that strings like '$p \rightarrow q$' and '$p \rightarrow (q \wedge r)$' are sentences of PL. You can now see that those strings cannot be constructed following the above procedure, as, by our above definition, any sentence of PL containing an occurrence of '$\wedge$', '$\vee$', '$\rightarrow$', or '$\leftrightarrow$' must be surrounded by left and right parentheses. We distinguish between a permissive and strict conception of a sentence of PL. When discussing the syntax of PL and distinguishing strings that are sentences from those that are not, we operate with the strict conception, which count those strings as nonsentences. However, in other contexts, we operate with a permissive conception that allows the outermost parentheses to be dropped, as their presence or absence does not lead to the ambiguities that led us to add them to our language. On the permissive conception, the outer parentheses are present but invisible, much like the phenomena of verb phrase deletion in English. (We say, for example, "George went to class and Sally did, too," omitting the second occurrence of 'went to class'.) So, when a string, like the two displayed above, are strictly speaking not sentences of PL but would be if they included a pair of outermost parentheses, then we will treat those parentheses as present but hidden and so count the string as a sentence of PL under the permissive sense; namely, the sentence it would be on the strict conception if those hidden outermost parentheses were present. Three notes: First, for sentence identification problems ("Grammar and Scope" problems in the problem sets), operate *only* with the strict conception. Second, in contexts where we operate with the permissive conception, we do not include hidden outermost parentheses for atomic sentence-letters like 'p', as they are sentences of PL without parentheses, and for negations like '$\neg(p \rightarrow (q \vee r))$', as they too are sentences of PL without parentheses, as can be verified by looking carefully at the use of parentheses in the inductive step of the above definition. (Indeed, adding outer parentheses to an atomic sentence-letter results in ungrammaticality, as the string '(p)', for example, is not a sentence of PL.) Finally, we *only* apply the hidden parentheses principle to outermost parentheses. We cannot insist that the ungrammatical string '$p \rightarrow q \vee r$' is in fact grammatical because of the presence of two pairs of hidden parentheses!

2.6 Main Operator and Scope

Consider the difference between the following two sentences of PL displayed earlier, '$(p \rightarrow (q \vee r))$' and '$((p \rightarrow q) \vee r)$'. In both cases, the sentences are constructed from the same vocabulary of PL: The atomic sentence-letters 'p', 'q', and 'r', the truth-functional operators '$\rightarrow$' and '$\vee$', and parentheses. For the first sentence, we first construct the sentence '$(q \vee r)$' by concatenating 'q' and 'r' with '$\vee$' at the first stage and then concatenating 'p' and '$(q \vee r)$' with '$\rightarrow$' at the second stage. So, regarding the order of construction, we first make a disjunction and then a conditional sentence. The sentence is *primarily* a conditional and the first occurrence of '$\rightarrow$' is the sentence's *main operator*. The **main operator** of a sentence is the occurrence of a truth-functional operator in the sentence that is the last operator added in the construction of the sentence. For the second sentence, '$((p \rightarrow q) \vee r)$', the main operator is the last occurrence of '$\vee$' and so the sentence is primarily a disjunction with a conditional as its left disjunct. We can see why by considering how the sentence is constructed. At the first stage, we concatenate the atomic sentence-letters 'p' and 'q' with the operator '$\rightarrow$' to get the sentence '$(p \rightarrow q)$', which, at the second stage, is in turn concatenated with the atomic sentence-letter 'r' and the operator '$\vee$'. Thus, the last operator added in constructing this sentence is the last occurrence of '$\vee$'.

Note that we say *occurrence* of a given truth-functional operator. In the above sentences, the two occurrences of truth-functional operators are occurrences of the different operators and so there is no real harm in saying that '$\rightarrow$' is the main operator of the first sentence and '$\vee$' is the main operator of the second. Consider, however, the following pair of sentences: '$((p \rightarrow q) \rightarrow p)$' and '$(p \rightarrow (q \rightarrow p))$'. For these sentences, we can't simply say that the main operator is '$\rightarrow$', as that doesn't mark their difference. Instead, we must say that the main operator of the first sentence is the *last* (or rightmost) occurrence of '$\rightarrow$' and the main operator of the second of the second sentence is the *first* occurrence of '$\rightarrow$'.

Here are some principles that help to determine whether or not a given occurrence of a truth-functional operator in a sentence is the sentence's main operator. If the occurrence is a negation, '$\neg$', then, unless it is the leftmost symbol in the sentence, it is *not* the main operator of the sentence. If the occurrence is any of the remaining two-place truth-functional operators, delete the outermost parentheses and that symbol and see if both of the remaining pair of strings are sentences of PL. If they are, then the occurrence is the sentence's main operator; if they are not, then the occurrence is not the sentence's main operator. So, for example, consider the sentence '$((p \rightarrow q) \vee r)$'. When we drop the outermost parentheses and delete the occurrence of '$\rightarrow$', we are left with the strings '$(p$' and '$q) \vee r$'. As neither is a sentence of PL, the occurrence of '$\rightarrow$' is *not* the sentence's main operator. (Notice that we must be exact in using this technique. We *only* delete the outermost parentheses and occurrence of the operator in question, leaving everything else in the string exactly in place.) Now drop the outermost parentheses and delete the occurrence of '$\vee$', leaving us

with the strings '$(p \rightarrow q)$' and 'r'. As both of the resulting strings are sentences of PL, the occurrence of '$\vee$' *is* the sentence's main operator. These are useful heuristics in determining whether or not a given occurrence of a truth-functional operator in a sentence is that sentence's main operator.

A sentence's main operator is the occurrence of a truth-functional operator with the widest scope. In fact, the main operator of a sentence is always the occurrence of a truth-functional operator whose scope is the entire sentence. What does that mean, you reasonably ask? Explaining that requires that we first explain the notion of the scope of an occurrence of an operator.

The **scope** of an occurrence of an operator is the smallest string of immediately adjacent symbols in the sentence that includes the occurrence of the operator and is a sentence of PL. Thinking again in terms of the construction of the sentence, this means that the scope of an occurrence of an operator is everything present at the completion of the stage in the sentence's generation using the construction procedure encoded by the recursive definition of the notion of a sentence of PL where that occurrence is added. Let's return to our earlier pair of sentences, '$(p \rightarrow (q \vee r))$' and '$((p \rightarrow q) \vee r)$'. We saw in our earlier discussion that the first sentence is constructed by first concatenating 'q' and 'r' with '$\vee$', producing '$(q \vee r)$'. Thus, the scope of the occurrence of '$\vee$' in the first sentence is '$(q \vee r)$'. That is also the smallest unit that includes the occurrence of the operator that is itself a sentence. At the second stage of the construction of the first sentence, 'p' and '$(q \vee r)$' are concatenated with '$\rightarrow$', producing '$(p \rightarrow (q \vee r))$', which is also the scope of the occurrence of '$\rightarrow$'. That is the entire sentence, and so that occurrence of '$\rightarrow$' is the sentence's main operator. The scope of the occurrence of '$\rightarrow$' is wider than than the scope of the occurrence of '$\vee$', as '$\vee$' occurs inside the scope of '$\rightarrow$' but not vice versa. The scope of the occurrence of '$\vee$' is narrower and the occurrence of '$\rightarrow$' has the widest scope.

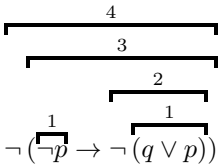
The opposite is true of the second sentence, '$((p \rightarrow q) \vee r)$'. That sentence is constructed by first concatenating 'p' and 'q' with '$\rightarrow$', producing '$(p \rightarrow q)$'. Thus the occurrence of '$\rightarrow$' in the second sentence is '$(p \rightarrow q)$'. At the completion of second stage of the construction of the second sentence, '$(p \rightarrow q)$' and 'r' are concatenated with '$\vee$', producing '$((p \rightarrow q) \vee r)$', which is the scope of the occurrence of '$\vee$'. So, the scope of the occurrence of '$\vee$' in the second sentence is wider than the scope of the occurrence of '$\rightarrow$'.

It will be very useful to have an ordering of occurrences of truth-functional operators in a sentence from narrowest to widest. We will call this the **scope degree** of an occurrence of an operator in a sentence. Let's say that atomic sentences have a scope degree of zero, occurrences of truth-functional operators with the narrowest scope, operating only on atomic sentence-letters, have a scope degree of 1, occurrences of truth-functional operator with the next narrowest scope have a scope degree of 2, and so on. So, a sentence's main operator has the greatest scope degree of any occurrence in the sentence. (Because truth-functional operators are always added one at a time and every sentence of PL is finite in length, even though there is no longest sentence of PL, as, for any sentence, whatever its length, there is at least one sentence that is longer —

for example, that sentence's negation — for every sentence, there is a single occurrence of a truth-functional operator that is that sentence's main operator.) Applying this notion, we would say of the first sentence above, '$(p \to (q \vee r))$', that the scope degree of all of the atomic sentence-letters 'p', 'q', and 'r' is 0, the scope degree of the occurrence of '$\vee$' is 1, and the scope degree of the occurrence of '$\to$', the main operator of the sentence, is 2. (Remember: The wider the scope of an occurrence of an operator, the greater its scope degree.) For the second sentence, '$((p \to q) \vee r)$', by contrast, the scope degree of the occurrence of '$\to$' is 1 and the scope degree of the occurrence of '$\vee$', the main operator of the sentence, is 2.

It is important to see that two occurrences of operators in a sentence might have the same scope degree, when neither occurs in the scope of the other, although this is only the case when there is an occurrence of an operator in the sentence with a wider scope that encompasses them both. As an example, consider the sentence '$\neg(\neg p \to \neg(q \vee p))$'. Let's start from the narrowest scope and work outward. The atomic sentence-letters all have a scope degree of 0. The second occurrence of '$\neg$', the one attaching immediately to the first occurrence of 'p', has a scope degree of 1, as '$\neg p$' is the smallest unit containing that symbol that is a sentence. Similarly, the occurrence of '$\vee$' between the occurrence of 'q' and last occurrence of 'p' also has a scope degree of 1, as '$(q \vee p)$' is the smallest unit containing that symbol that is a sentence. Both of those occurrences have the narrowest scope, not containing any other truth-functional operators within their scopes and added to the sentence on the first application of the recursive clause of the definition of the notion of a sentence of PL. The last occurrence of '$\neg$' has the next narrowest scope, its scope being the subsentence '$\neg(q \vee p)$', and so a scope degree of 2. The occurrence of '$\to$' has the next narrowest scope, its scope being the subsentence '$(\neg p \to \neg(q \vee p))$', and so a scope degree of 3. While the scope degrees of the other occurrences of truth-functional operators in the sentence that we have considered above correspond the the number of occurrences of truth-functional operators in their scopes — for example, the second occurrence of '$\neg$' has a scope degree of 1 and has one occurrence of a truth-functional operator in its scope, the last occurrence of '$\neg$' has a scope degree of 2 and has two occurrences of truth-functional operators in its scope, itself and the occurrence of '$\vee$', etc. — the scope degree of the occurrence of '$\to$' proves that this is not always the case. There are four occurrences of truth-functional operators within its scope, as can be seen above, and yet its scope degree is 3 because it is added to the sentence in its order of construction at the third application of the recursive clause of the definition of the notion of a sentence of PL. Finally, the first occurrence of '$\neg$' has a scope degree of 4 and so has the widest scope. We can see, then, the two occurrences of truth-functional operators in this sentence have the same scope degree, both the second occurrence of '$\neg$' and the occurrence of '$\vee$, as neither are in the scope of the other and both are added to the sentence at the same stage in the construction of the sentence.

We can represent this with the following diagram.



We don't bother marking the scope degree of atomic sentence-letters, as we know that they are always 0.

In the last several paragraphs we suggested determining the scope and degree of scope of an operator by working through the construction of the sentence using the recursive definition. This method always works and is the best method for coming to understand the notion of scope. But there are simpler methods for determining the scope of an occurrence of an operator that are useful to use when working on problem sets. If the operator is a two-place operator (i.e., '$\wedge$', '$\vee$', '$\rightarrow$', or '$\leftrightarrow$'), then, beginning with that occurrence of the operator in the sentence, move one spot to the left, one spot to the right, another spot to the left, another spot to the right, including ever more elements of the sentence in that manner, until the resulting string is a sentence of PL. (Hint: One can stop when one has included the same number of left parentheses as right parentheses via that method.) That is the smallest unit of the sentence that includes the occurrence of the operator that is itself a sentence. And that is the scope of the occurrence of the operator. If the operator is the one-place operator '$\neg$', then one performs a similar procedure, but only moving to the right, including ever more elements of the sentence, until the resulting string is a sentence of PL. (Hint: One can stop when either the symbol immediately adjacent to the occurrence of '$\neg$' is an atomic sentence-letter, one encounters an uninterrupted stack of occurrences of '$\neg$' followed immediately by an atomic sentence-letter, or one has included the same number of left parentheses as right parentheses in the string.)

Let's illustrate this second method by considering the scopes and degree of scope of the occurrences of operators in the sentence '$\neg\neg((p \rightarrow q) \rightarrow q)$'. We start from the rightmost occurrences of operators, moving to the left. Applying the procedure to the right most occurrence of '$\rightarrow$', we first get the string '$q) \rightarrow q)$'. This string is not a sentence of PL, so we continue, including more material from each side, but, as we have reached the end of the sentence on the right, we only include more material from the left. And we can here take a shortcut. We know that we have not reached the end of the scope of the occurrence of the operator until we have the same number of left parentheses as right parentheses. We have two open right parentheses, so we can simply continue moving further to the left until we have two left parentheses (assuming we do not encounter more right parentheses along the way, in which case that number rises accordingly). So, using this method, we see that the scope of the rightmost occurrence of '$\rightarrow$' is '$((p \rightarrow q) \rightarrow q)$'. Now that we have that occurrence's scope worked out, we can deconstruct the sentence to figure its scope degree. Its component parts are '$(p \rightarrow q)$' and 'q'. The sentence cannot be constructed until after its component parts are constructed and its left component part is not constructed

until the first application of the recursive clause, as it has logical complexity. The leftmost occurrence of '$\rightarrow$' has a scope degree of 1, as its only components are atomic sentence-letters, and the rightmost occurrence of '$\rightarrow$' has a scope degree of 2.

Consider now the second, rightmost occurrence of '$\neg$'. We apply the method describe above, moving only to the right, as the operator is '$\neg$', including more and more material from the sentence until the resulting string is a sentence, at which point we stop, having found the scope of the occurrence of that operator. At our first step, we hit a left parentheses, resulting in the string '$\neg($', which is not a sentence of PL. We know that we will only have a sentence when we reach the matching right parenthesis. So, we continue rightward adding material to our string until we reach the matching right parenthesis. The first right parenthesis we reach results in the string '$\neg((p \rightarrow q)$'. However, because there is another left parenthesis between our initial left parenthesis and this right parenthesis and parentheses always attach to the nearest open opposing parenthesis, we know that our initial left parenthesis is still open in this string. So, we must continue further to the right, as that string is not a sentence of PL, having an uneven number of left and right parentheses. It isn't until we reach the end of the sentence that we find the matching right parenthesis and so have a sentence of PL. So, the scope of the rightmost occurrence of '$\neg$' is '$\neg((p \rightarrow q) \rightarrow q)$'. Notice that the two occurrences of '$\rightarrow$' each occur within the scope of the rightmost occurrence of '$\neg$'. So, it has a wider scope than either occurrence of '$\rightarrow$' and has a scope degree one greater than the occurrence with the highest scope order, having a scope degree of 3. The initial, leftmost occurrence of '$\neg$' is the main operator of the sentence, which we know by employing our heuristic above of noting that it is an initial, leftmost '$\neg$' immediately followed by another occurrence of '$\neg$'. This occurrence of '$\neg$', then, has the entire sentence as its scope and a scope degree of 4.

2.7 Translations with Multiple Occurrences of Operators

In this section we briefly return to more complicated translation problems, armed with our sense of the grammar of PL, the function of parentheses, and the notion of scope.

One very common form of sentence are negations of conjunctions and disjunctions. An example is the sentence 'Bill neither called nor came over'. This is an example of a negation of a disjunction and should be translated, where 'p' translates 'Bill called' and 'q' translates 'Bill came over', as '$\neg(p \vee q)$'. Another example is the sentence 'Bill didn't take both call and come', which can be paraphrased as 'It's not the case that Bill both called and came' and is best translated as '$\neg(p \wedge q)$'.

It is important to be aware of the difference between negations of conjunctions and disjunctions and conjunctions and disjunctions of negations. Contrast

the first sentence with the sentence 'Bill either didn't call or he didn't come'. It should be hard to confuse the two, as the second has two negations, strongly suggesting that the negations attach to each disjunct instead of the disjunction as a whole. This sentence should be translated as '$(\neg p \vee \neg q)$'. Now contrast the second of the sentences in the previous paragraph with the sentence 'Bill both didn't come and didn't call'. Again, this latter sentence is pretty clearly the conjunction of two negations, as again is suggested by the fact that it has two occurrences of the contracted negation. It should be translated as '$(\neg p \wedge \neg q)$'. In two sentences from the previous paragraph, the negation has the widest scope, while in the sentences in this paragraph, the disjunction and conjunction have the widest scope.

It may be tempting to hear the first sentence 'Bill neither called nor came over' as a conjunction of two negations. "Look, I'm saying that he didn't come over and what's more he didn't even call." What is correct in this thought is that '$\neg(p \vee q)$' and '$(\neg p \wedge \neg q)$' *say* the same thing in the sense that the two sentences are equivalent. (We will prove this in section 3.6 below.) However, the two sentences are clearly different and the more faithful translation of the sentence is as the negation of a disjunction. In translation problems, you should strive to be as true to the surface form of the target sentence as possible consistent with the demands of the syntax of the language of PL.

CHAPTER REVIEW

- We translate English target sentences into the formal language of Propositional Logic by first identifying the truth-functional logic words in the target sentences from the atomic sentences (i.e., the subsentences that are full sentences, at least in their underlying form, but that do not contain any truth-functional complexity induced by truth-functional logic words). We then create a Translation Manual by associating each atomic sentence in the target sentences with its own atomic sentence-letter. We next provide intermediate translations of the target sentences by replacing the atomic sentences in the target sentence by their associated atomic sentence-letter, leaving the original truth-functional logic words in place. We then translate the remaining English logic words by replacing each with the appropriate truth-functional operator, adding parentheses as needed.

- The vocabulary of the language of Propositional Logic consists of an infinite stock of atomic sentence-letters $\{p, q, r, s, p_1, q_1, r_1, s_1, \ldots\}$, the truth-functional operators $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$, and parentheses '(' and ')'.

- PL sentences of the form $\ulcorner \neg A \urcorner$ are *negations* and translate sentences of the form $\ulcorner$It is not the case that $A\urcorner$; PL sentences of the form $\ulcorner A \wedge B \urcorner$ are *conjunctions*, A and B are said to be *conjuncts*, and translate sentences of the form $\ulcorner A$ and $B\urcorner$; PL sentences of the form $\ulcorner A \vee B \urcorner$ are *disjunctions*, A and B are said to be *disjuncts*, and translate sentences of the form $\ulcorner A$ or $B\urcorner$; PL sentences of the form $\ulcorner A \rightarrow B \urcorner$ are *conditionals*, A, the subsentence to the left of '$\rightarrow$', is said to be the *antecedent* and B, the sentence to the right of '$\rightarrow$', the *consequent*, and translate sentences of the form $\ulcorner$If A, then $B\urcorner$; finally, sentences of the form $\ulcorner A \leftrightarrow B \urcorner$ are *bi-conditionals* and translate sentences of the form $\ulcorner A$ if and only if $B\urcorner$.

- The following are useful translation schemas for material conditionals.

  A only if $B \Longrightarrow A \rightarrow B$

  A if $B \Longrightarrow B \rightarrow A$

  A when $B \Longrightarrow B \rightarrow A$

  A provided $B \Longrightarrow B \rightarrow A$

  A unless $B \Longrightarrow \neg B \rightarrow A$

- The notion of a sentence of PL is defined recursively as follows.

  Base Case: Every atomic sentence-letter is a sentence of PL.

  Inductive Clause: For every string of symbols A and B, if A and B are sentences of PL, then $\ulcorner\neg A\urcorner$, $\ulcorner(A \wedge B)\urcorner$, $\ulcorner(A \vee B)\urcorner$, $\ulcorner(A \rightarrow B)\urcorner$, and $\ulcorner(A \leftrightarrow B)\urcorner$ are all sentences of PL as well.

  Extremal Clause: Nothing else is a sentence of PL.

- We test a string for grammaticality for seeing whether or not the string can be "reverse engineered" in accordance with the above definition.

- The **scope** of an occurrence of an operator in a sentence is the smallest string of immediately adjacent symbols in the string that includes the occurrence of the operator and is a sentence of PL. As an example, consider the sentence '$\neg(\neg p \leftrightarrow (q \vee r))$'. The scope of the occurrence of '$\leftrightarrow$' is '$(\neg p \leftrightarrow (q \vee r))$'; any fewer adjacent symbols that contain '$\leftrightarrow$' is not a sentence, as can be verified by covering some of the symbols in the above string and checking for grammaticality.

- One occurrence of a truth-functional operator is **wider** than another occurrence when the second is inside the scope of the first. So, for example, because '$\vee$' occurs within the scope of '$\leftrightarrow$' in the sentence '$\neg(\neg p \leftrightarrow (q \vee r))$' discussed above, the scope of the occurrence of '$\vee$' is narrower than the scope of the occurrence of '$\leftrightarrow$'.

- The main operator of a sentence is the occurrence of a truth-functional operator that is added last in the construction of the sentence by the above described construction procedure and has the widest scope.

- The **degree of scope** of an occurrence of a truth-functional operator in a sentence is its place in the order of the occurrences of operators from narrowest to widest. As an example, here are the degrees of scope of the above sentence.

3		1		2		1	
$\neg$	$($	$\neg$	p	$\leftrightarrow$	$(q$	$\vee$	$r))$

2.8 Appendix. Construction Procedure for the Sentences of the Language of PL: The Complete Version

In this section we offer a more precise account of the construction of sentences of PL. The issues are complex and necessary for a complete understanding of the logic of Propositional Logic, but unnecessary for a basic understanding of the material in this text. The section is thus optional for students interested in deepening their understanding of the material.

Earlier we described how the recursive definition of the notion of a sentence of PL can be conceived as a construction procedure or instructions for building all and only sentences of PL. Our description was, however, woefully inadequate, as we assumed that there were only three atomic sentence-letters, 'p', 'q', and 'r', at the first step of the construction, noting that there are in fact infinitely many atomic sentence-letters. In this section, we lift the pretense that there are only three atomic sentence-letters and describe the construction procedure using every atomic sentence-letter.

We face several issues once we lift that limit. First, if we have infinitely many sentences of PL at the first stage of the construction, do we ever produce any more, new sentences at later stages of the construction? The answers are, "yes," we produce new sentences at the later stages of the construction, in the sense that, at the second stage, we produce sentences (sentences like '$\neg p$' and '$(p \rightarrow q)$') that are not among the sentences at the first stage of the construction, and, more generally, at each stage of the construction, new sentences are constructed, sentences with a greater logical complexity, than the sentences constructed at the earlier stages, but, "no," we do not construct any *more* sentences at any later stage than there are at the first stage. This is because there are, at the first step of the construction procedure, before applying the inductive step, infinitely many sentences of PL, as there are infinitely many atomic sentence-letters, and there are exactly that many sentences of PL in sum total! (More precisely, there are aleph-naught atomic sentence-letters and aleph-naught sentences of PL.) So, although there are sentences of PL with logic complexity while atomic sentence-letters do not have any logical complexity, the logically complex sentences being constructed only at the later steps of the construction, there are not *more* sentences of PL than there are atomic sentence-letters. This is one of the many wonders of infinity. (These issues are related to the fact that there are just as many even numbers as there are positive integers, just as many prime numbers as there are positive integers, and, most disturbingly, just as many rational numbers as positive integers, where a rational number is a number expressible as a fraction. This last is disturbing because the rational numbers are densely packed, which means that, for any two rational numbers, there is another rational number, and so infinitely many rational numbers, between them! Cantor famously proved the equinumerosity of the positive integers and the rationals, on his way to proving that there are strictly *more* real numbers than positive integers and so different sizes of infinity.)

At the first stage of the construction, all atomic sentence-letters are knighted as sentences of PL. At the second stage, the first application of the inductive step, where sentences with a logical complexity of 1 are constructed, there are an infinite number of tasks to perform. At this stage, new sentences are constructed by taking, for each pair of atomic sentence-letters, $\{A, B\}$, including all identity pairs, and constructing the following sentence forms: $\ulcorner\neg A\urcorner$, $\ulcorner(A \wedge B)\urcorner$, $\ulcorner(A \vee B)\urcorner$, $\ulcorner(A \rightarrow B)\urcorner$, and $\ulcorner(A \leftrightarrow B)\urcorner$. There are, however, two problems with this. The first, and easiest to solve, concerns the notion of taking *all* the pairs of atomic sentence-letters and the second concerns the notion of completing the task of constructing all of these sentences.

How do we list all of the pairs of atomic sentence-letters? The following doesn't work: "Start with 'p' and couple it with all of the atomic sentence-letters $\{p, q, r, s, p_1, q_1, r_1, s_1, \ldots\}$, then move to '$q$' and couple it with all of the atomic sentence-letters $\{p, q, r, s, p_1, q_1, r_1, s_1, \ldots\}$, and continue with that pattern." These instructions suffer

from a problem that we will see again. It seems that one will never complete the first step of coupling all of the atomic sentence-letters with 'p', as that requires completing an infinite number of tasks, in which case one will only ever operate with pairs of sentences that contain 'p', never operating, for example, on the pair $\{q, r\}$. Here is a method for ensuring that we operate on each pair of atomic sentence-letters. We do this by *enumerating* all of the pairs of atomic sentence-letters; that is, we need to put all of the pairs of atomic sentence-letters in a single, linear list. We begin by creating a table with all of the atomic sentence-letters as both columns and rows, as in the following representation. (The relevance of the arrows in each cell will be made apparent later.)

	p	q	r	s	p_1	q_1	r_1	...
p	$\{p, p\}$ $\Rightarrow$	$\{q, p\}$ $\Downarrow$	$\{r, p\}$ $\Rightarrow$	$\{s, p\}$ $\Downarrow$	$\{p_1, p\}$ $\Rightarrow$	$\{q_1, p\}$ $\Downarrow$	$\{r_1, p\}$ $\Rightarrow$	...
q	$\{p, q\}$ $\Downarrow$	$\{q, q\}$ $\Leftarrow$	$\{r, q\}$ $\Uparrow$	$\{s, q\}$ $\Downarrow$	$\{p_1, q\}$ $\Uparrow$	$\{q_1, q\}$ $\Downarrow$	$\{r_1, q\}$ $\Uparrow$	...
r	$\{p, r\}$ $\Rightarrow$	$\{q, r\}$ $\Rightarrow$	$\{r, r\}$ $\Uparrow$	$\{s, r\}$ $\Downarrow$	$\{p_1, r\}$ $\Uparrow$	$\{q_1, r\}$ $\Downarrow$	$\{r_1, r\}$ $\Uparrow$	...
s	$\{p, s\}$ $\Downarrow$	$\{q, s\}$ $\Leftarrow$	$\{r, s\}$ $\Leftarrow$	$\{s, s\}$ $\Leftarrow$	$\{p_1, s\}$ $\Uparrow$	$\{q_1, s\}$ $\Downarrow$	$\{r_1, s\}$ $\Uparrow$	...
p_1	$\{p, p_1\}$ $\Rightarrow$	$\{q, p_1\}$ $\Rightarrow$	$\{r, p_1\}$ $\Rightarrow$	$\{s, p_1\}$ $\Rightarrow$	$\{p_1, p_1\}$ $\Uparrow$	$\{q_1, p_1\}$ $\Downarrow$	$\{r_1, p_1\}$ $\Uparrow$	...
q_1	$\{p, q_1\}$ $\Downarrow$	$\{q, q_1\}$ $\Leftarrow$	$\{r, q_1\}$ $\Leftarrow$	$\{s, q_1\}$ $\Leftarrow$	$\{p_1, q_1\}$ $\Leftarrow$	$\{q_1, q_1\}$ $\Leftarrow$	$\{r_1, q_1\}$ $\Uparrow$	...
r_1	$\{p, r_1\}$ $\Rightarrow$	$\{q, r_1\}$ $\Rightarrow$	$\{r, r_1\}$ $\Rightarrow$	$\{s, r_1\}$ $\Rightarrow$	$\{p_1, r_1\}$ $\Rightarrow$	$\{q_1, r_1\}$ $\Rightarrow$	$\{r_1, r_1\}$ $\Uparrow$	...
s_1	$\{p, s_1\}$ $\Downarrow$	$\{q, s_1\}$ $\Leftarrow$	$\{r, s_1\}$ $\Leftarrow$	$\{s, s_1\}$ $\Leftarrow$	$\{p_1, s_1\}$ $\Leftarrow$	$\{q_1, s_1\}$ $\Leftarrow$	$\{r_1, s_1\}$ $\Leftarrow$	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

This table obviously won't serve as a list, as it extends infinitely far to the right and infinitely far down, and a list must be linear. However, following a trick devised by Cantor in proving the enumerability of the rational numbers mentioned above, we can transform this table into a list by starting with the first cell, move one cell to the right, one cell down, one cell to the left, then moving one cell down, two cells to the right, two cells up, then moving one cell to the right, three cells down, three cells to the left, etc., as mapped by following the arrows through the array, with each step writing the contents of the cell on one's list, as follows: $< \{p, p\}$, $\{q, p\}$, $\{q, q\}$, $\{p, q\}$, $\{p, r\}$, $\{q, r\}$, $\{r, r\}$, $\{r, q\}$, $\{r, p\}$, $\{s, p\}$, ... $>$. Every pair of atomic sentence-letters appears at least once (and many more than once, as the pair $\{p, q\} = \{q, p\}$, for example). We can now say that the second stage of the construction procedure involves constructing the 5 sentence forms described above for each pair of sentences on this list.

We now face our second problem. At the second stage, we construct sentences with a logical complexity of 1. Once this stage is complete, we can move to the second stage, where we construct sentences with a logical complexity of 2 and 3. We must, however, perform an infinite number of tasks before the second stage of the construction is complete. It is quite reasonable to point out that, work as you may, you won't ever finish step 2, as doing so would require completing an infinite number of actions, and so will never produce any sentences of a logical complexity greater than 1. At best, the procedure is a way of constructing ever more sentences of PL with a logical complexity of 1, but not all sentences of PL whatsoever, as we never get past the second stage of the construction and so never construct sentences with a logical complexity of 2 or greater.

This is a serious concern that touches on a number of complex issues connected to a favorite topic: Infinitity. We will briefly describe two solutions. The first solution describes a way to finish each step in a finite period of time and the second solution describes an infinite series of steps each with finitely many actions for constructing all of the sentences of PL.

A **supertask** is a task whose completion requires the completion of an infinite number of actions. The pre-Socratic ancient Greek philosopher Zeno thought that completing a supertask is absolutely impossible and used that as the basis of a number of fascinating arguments for wild theses, like the impossibility of motion. To move from here to there, whether the distance to be covered is three billion light years or just two inches, requires an infinite number of trips: First, one must travel half the distance, then, three-quarters of the distance, then seven-eighths

of the distance, and so on indefinitely. (Zeno assumed that space was infinitely divisible.) Of course, you never even get halfway, because you would first have to get one-quarter of the way, but that requires you first get one-eighth of the way, and so on indefinitely. You might as well just stay where you are! Zeno concluded that motion is impossible. Most people find that conclusion a bit crazy. It would appear that you only need walk to the kitchen for water, stand up to stretch, or just scratch your head in puzzlement to disprove it, as those all involve actual movement and if something actually happens, then it is not impossible. Some may claim that space is in fact not infinitely divisible and that that was Zeno's mistake. But most think that supertasks are instead perfectly possible and that certain supertasks, the supertasks involved in moving from one place to another, say, are actually complete. (The literature on this topic is rich and we encourage you to check it out for yourself. The best place to start is with a collection of papers edited by Wesley Salmon 1970 titled *Zeno's Paradoxes* [Indianapolis: Bobbs-Merril], which includes Paul Benacerraf's important paper of 1962, 'Tasks, Super-Tasks, and Modern Eleatics'. Also important is Charles Chihara, 1965, 'On the Possibility of Completing an Infinite Task' [in *Philosophical Review*, 74: 74–87], Aldolf Grünbaum, 1967, *Modern Science and Zeno's Paradoxes* [Middletown, CT: Wesleyan University Press] — selections from this book are included in Salmon (ed.) — and John Earman and John Norton's important paper of 1996 titled 'Infinite Pains: The Trouble with Supertasks' [in Morton and Stich (eds.), *Benacerraf and His Critics*, Oxford: Blackwell, 231–261].

Let's suppose that supertasking is possible. Then, if we can list all of the pairs of atomic sentence-letters, the following procedure is a way to complete our first task of constructing all of the sentences of PL with a logical complexity of 1 in just five minutes! Take the first pair in the list and construct the relevant instances of the five sentence forms from them in the first 2.5 minutes. Then do the same procedure to the second pair in the list in half the time (i.e., in the first 3.75 minutes). Continue moving through the list, performing the series of operations on each pair of atomic sentence-letters on the list in half the time it took to perform the operations on the pair before. Insofar as time is infinitely divisible (and you can continue to move faster and faster, something logically possible even if not, as it most certainly is not, physically possible), after 5 minutes of faithfully carrying out this procedure, you will have constructed every sentence of PL with a logical complexity of 1. We can then move to the third stage of the construction procedure. We enumerate all of the sentences of a logical complexity of 1 or less, enumerate all of the pairs of sentences of a logical complexity of 1 or less using the same Cantorian technique described above, and supertask our way through the construction of all sentences of a logical complexity of 3 or less. We will then complete stage 3 after an additional 5 minutes and then move to stage 4, and so on.

Supertasks are fun. But there is an elegant solution to our problem that does not rely on them. This is because we can describe a single infinite series of *finite tasks* for constructing all and only the sentences of PL. We begin with a list of the atomic sentence-letters of PL. We take the first n items of that list, for some finite number n, and construct all of the sentences of PL with a logical complexity of 1 *from only those sentence-letters* by taking all of the pairs of sentences on that set and, for each pair, constructing the relevant instance of all five of the sentence forms in the manner described above, which is a finite task as we have taken only n-many atomic sentence-letters. At the third stage, we take this finite set of sentences and extend it with the next atomic sentence-letter, the $n + 1^{th}$ in the list, and repeat the same procedure, constructing the instances of the five sentence forms from each pair of sentences on the set of all pairs from that set. We will now have sentences with a logical complexity of no more than 3, as well as sentences containing the new atomic sentence-letters we have added. As there are only finitely many pairs of sentences in this set, there are only finitely many tasks required to complete this stage of the construction. At the fourth stage, we add the next atomic sentence-letter in the original list and repeat the same procedure, and so on infinitely. Taking this procedure to its completion, we have constructed every sentence of PL and nothing that isn't.

Here's a more concrete description of this procedure. List the atomic sentence-letters $< p, q, r, s, p_1, q_1, \ldots >$. Take the first item from that list. By an application of the Base Case Clause, it is a sentence of PL. We apply the Recursive Clause to this finite set of sentences

of PL, first constructing all of the negations from this set — namely, '$\neg p$' — and all of the two-place truth-functional sentences on the set of all of the pairs of sentences on this set — namely, the pair $\{p, p\}$, yielding the sentences '$(p \wedge p)$', '$(p \vee p)$', '$(p \rightarrow p)$', and '$(p \leftrightarrow p)$'. That is the completion of the first stage and it did not require supertasking. At the end of this stage, we have six sentences of PL. We now move to the second stage. We add the next item in our list of atomic sentence-letters, 'q', to the six sentences of PL constructed in the previous stage. Applying the Recursive Clause to the set of sentences $\{p, \neg p, (p \wedge p), (p \vee p), (p \rightarrow p), (p \leftrightarrow p), q\}$, resulting in 202 new sentences of PL, for a total, adding the original seven, in 209 sentences of PL constructed at the end of stage 2. (You needn't worry, we won't try to list them all! The most logically complex among them, however, are sentences like '$((p \wedge p) \rightarrow (p \vee p))$', with a logical complexity of 3.) We have created sentences of PL with both more logical complexity than those completed at the earlier stage and that include more atomic sentence-letters as constituents. However, the task, while exponentially larger than the first task, requires only a finite number of actions. Once complete, we move to the third stage of the construction, adding the next item in the original list of atomic sentence-letters, 'r', to the set of 209 sentences just constructed and again apply the Recursive Clause to that set 210 sentences, constructing sentences like '$(((p \wedge p) \rightarrow (p \vee p)) \leftrightarrow ((p \vee p) \vee (p \vee p)))$', with a logical complexity of 7. We continue, in a like manner, each time adding another atomic sentence-letter to the finite set of sentences constructed at the stage immediately before and reapplying the Recursive Clause to the resulting set, constructing more sentences of PL with a greater logical complexity and containing occurrences of the newly added atomic sentence-letter. At no finite stage do we have all of the sentences of PL constructed, or even all of the sentences of PL with a given logical complexity constructed. (For example, at any stage n of the construction, where A is the $n+1$-th atomic sentence-letter on the original list of atomic sentence-letters, the sentence $\ulcorner \neg A \urcorner$ is not constructed and so there is at least one sentence of PL with a logical complexity of 1 that is not yet constructed.) Indeed, it is this very fact that ensures that each stage of the procedure can be completed with only finitely many actions, as we already know that we cannot construct even all of the negations of atomic sentence-letters in a finite number of steps. At the completion of all of the stages of this task, however, absolutely every sentence of PL is constructed. (Note that this last notion requires supertasking, as it involves the idea of a completion of an infinite number of actions. The sense, however, in which this procedure does not involve supertasking is that every step in that series requires only a finite number of actions to complete.)

Chapter 3

Truth Tables and the Semantics of Propositional Logic

In the previous chapter, we introduced the truth-functional operators of Propositional Logic, the symbols '$\neg$', '$\wedge$', '$\vee$', '$\rightarrow$', and '$\leftrightarrow$', by showing how English sentences can be translated into sentences of PL containing those symbols. Those translation mappings provide some indication of the meanings of the operators. In this chapter, we give a more precise account of their meanings, and more generally of the sentences of PL that contain them, by developing a formal semantics for the language of PL. This will then provide us with our first rigorous method for determining validity and invalidity.

3.1 Basic Truth Definitions and Introduction to Constructing Truth Tables

Our five truth-functional operators are functions on truth values. A function takes inputs and delivers an output. The inputs of a truth-function is a sequence of truth values and its output a truth value. So, the meanings of our truth-functional operators are functions from a sequence of truth values to a truth value. Let's explain this more concretely with the simplest truth-functional operator, '$\neg$'. '$\neg$' "flips" the truth value of a sentence within its scope. The logically complex sentence $\ulcorner\neg A\urcorner$ is true exactly when its component part A is false. This shouldn't be too surprising. You already know that 'George is not sitting' is true exactly when 'George is sitting' is false; the first is the "opposite" of the second. Similarly, then, '$\neg p$' is true exactly when 'p' is false.

Let's make explicit two assumptions that form the basis of the classic logic studied here. There are exactly two truth values, **True** and **False**, and, for every sentence A of PL, A has exactly one of these values. (Later we will see

that a sentence of PL only has a truth value *under an interpretation*, but we'll omit that for now.) Then, we can say that the logically complex sentence $\ulcorner\neg A\urcorner$ is true exactly when its constituent part A is false. (These assumptions — that there are only two truth values and that every sentence has exactly one of them — are controversial. Some have argued that vagueness proves them false, as someone who is a "borderline case" of being bald, for example, is neither bald nor not bald and so the sentence 'That person is bald' is neither true nor false. We won't discuss the matter here. The debate over whether a classical logic, with its assumptions about truth values and bivalence, are adequate for natural languages is still a live debate. But the only way to be in a position to appreciate those debates is to first learn the classic logic in terms of which they are shaped, which is the subject of this present text.)

We can represent the truth definition giving the meaning of '$\neg$' with a truth table, as follows.

$\neg$	A
F	T
T	F

Each row of the truth table represents a different possible arrangement of truth values for the constituent parts of the complex sentence: On the first top row, the constituent sentence A is true, and on the second row, the constituent sentence A is false. The table then tells us what truth value the complex sentence has under each of the only two possible truth values its constituent part A might have. The truth value of the complex sentence under an interpretation is the truth value on the row associated with that interpretation in the column under the sentence's main operator, in this case '$\neg$', which we have here colored cyan. So, on the first row, when A is true, the complex sentence $\ulcorner\neg A\urcorner$ is false; on the second row, when A is false, the complex sentence $\ulcorner\neg A\urcorner$ is true. Because A can have exactly one truth value, true or false, and the truth value of $\ulcorner\neg A\urcorner$ is a function only of the truth value of A, these two rows exhaust the possible truth arrangements of the sentence.

The remaining four truth-functional operators are two-place operators and so their truth definitions are functions from pairs of truth values to a truth value. We will present the abstract truth definitions for '$\wedge$' and '$\vee$', work through concrete examples with the three operators '$\neg$', '$\wedge$', and '$\vee$', and then add the truth definitions for the remaining truth-functional operators '$\rightarrow$' and '$\leftrightarrow$'.

A conjunction of the form $\ulcorner(A\wedge B)\urcorner$ is true exactly when both the component sentences A and B are true. We can represent this with a truth table, as follows.

A	$\wedge$	B
T	T	T
T	F	F
F	F	T
F	F	F

Conjunctions are functions on the truth values of *two* sentences and so there are four permutations of truth values we must consider. This table represents all four of those permutations, one on each row of the truth table. On the first row, both A and B are true (and so the conjunction is true); on the second row, A is true and B is false (and so the conjunction is false); on the third row, A is false and B is true (and so the conjunction is false); and on the fourth row, both A and B are false (and so the conjunction is false). The table represents the truth definition that a conjunction is true exactly when both of its conjuncts are true and false otherwise.

A disjunction of the form $\ulcorner(A \vee B)\urcorner$ is false exactly when both A and B are false. We can represent this with a truth table, as follows.

A	$\vee$	B
T	T	T
T	T	F
F	T	T
F	F	F

Again, the four rows of this table represent all of the permutations of truth values relevant to the truth value of the sentence $\ulcorner(A \vee B)\urcorner$, as there are two constituent sentences A and B whose truth values are relevant to the truth value of the complex sentence. The table represents the truth definition that the disjunction is false exactly when both disjuncts are false and true otherwise.

> Some claim that English 'and' and 'or' have different truth definitions from those captured by the above truth definitions. Let's start with the case of 'or'. The above truth definition for '$\vee$' defines what is sometimes called an *inclusive disjunction*, so-called as it is true even when both disjuncts are true. But there is evidence that that is not how 'or' works in English. When George asks about the plans for the rest of the day and Peter responds, "We'll either go on a hike or to the beach," George misunderstands if he gets both his hiking shoes and his beach towel, saying, "Great, which one first?" Ordinary uses of 'or' are meant to be taken, then, in the *exclusive* sense, so-called as the speaker intends to impart the information that exactly one of the disjuncts are true but not both together. Some have taken this to show that we should translate English 'or' with the something like the following defined symbol.

A	$\triangle$	B
T	F	T
T	T	F
F	T	T
F	F	F

Where 'p' translates 'George and Peter will go on a hike together' and 'q' translates 'George and Peter will go to the beach together', '$p \triangle q$' seems to capture the intent of Peter's utterance.

There is also a case to be made against the view that the best translation of 'and' in English is as '$\wedge$', defined as it is in the text above. Compare the sentence 'Bill and Mary had two children and got married' with the sentence 'Bill and Mary got married and had two children'. An utterance of the first would naturally be taken to mean that Bill and Mary *first* had two children and *then* got married, while an utterance of the second does not. This suggests that the order in which the events reported occur in the sentence affect the truth value of the sentence. But the order of the conjuncts of a sentence of the form $\ulcorner(A \wedge B)\urcorner$ are irrelevant to the truth of the conjunction, as can be seen by noting that both $\ulcorner(A \wedge B)\urcorner$ and $\ulcorner(B \wedge A)\urcorner$ are true exactly when both conjuncts are true. So, translating 'and' in the above sentence as '$\wedge$' is inadequate.

The philosopher Paul Grice developed a theory of what he called *conversational implicatures* in part as a defense of the adequacy of the traditional account of English logic words like 'and' and 'or' defended in the main text above. The guiding thought behind Grice's theory is that conversations are cooperative activities that are governed by rational principles. Insofar as the participants of the conversation all assume that everyone in the conversation is co-operatively pursuing the common end of successfully communicating information, the parties will interpret one another in light of that assumption. Speakers can exploit that fact to mean by their utterance more, or even altogether other, than what the sentence uttered strictly and literally means. Here is an example to illustrate the basic idea. Frank asks Sally whether or not they should start dinner without Sam. Sally responds, "Sam called to say that his car broke down." Frank naturally takes Sally to mean that they should start without Sam. The sentence 'Sam called to say that his car broke down' doesn't mean *Let's start dinner without Sam*. Sally instead conversationally implicates that they should start dinner without Sam. The full details of the mechanisms by which this information is conveyed are beyond the scope of an introduction to logic course, but the interested party should read Paul Grice, 1975, 'Logic and conversation', in P. Cole and J. Morgan (eds.), *Syntax and Semantics*, 3, pp. 41?58, New York: Academic Press. Reprinted in H. P. Grice (ed.), 1989, *Studies in the Way of Words*, pp. 22?40, Cambridge, MA: Harvard University Press.

Let's apply the theory of conversational implicatures to the cases described two paragraphs above. Although a typical utterance of a conjunction describing events like Bill and Mary's getting married and having children, for example, may *convey* the thought that the event described in the first conjunct occurred before the event described in the second, that extra bit of information is an implicature and is not part of the strict and literal meaning of the sentence uttered and so the temporal order of the events is not part of the meaning of the world 'and'. Similarly, while Peter's utterance conveys the information that he and George will do at most one of the activities of going on a hike and going to the beach that afternoon, that is extra information conversationally implicated and is not part of the strict and literal meaning of the sentence he uttered and of the word 'or' in particular. On this view, then, the linguistic phenomenon used to challenge the standard regimentation of logic words like 'or' and 'and' into the language of PL are to be explained as conversational implicatures that are not part of the strict and literal meaning of the words being translated and so should not be included in a logic of those words.

We presented the above truth definitions schematically, with metalanguage

variables A and B, because, *for any sentence or subsentence* whose main operator is, say, '$\neg$', the truth conditions of that sentence is given by the above stated truth definition of '$\neg$'. This is easy enough to apply for sentence with a logical complexity of 1, like '$\neg p$' and '$p \vee q$'. Their truth tables pattern exactly the schematic truth tables above. So, for example, the first looks like this.

¬	p
F	T
T	F

However, complexities begin to arise when we calculate the truth table for sentences with greater logical complexity.

Consider first the sentence '$\neg(\neg p \wedge \neg q)$', which has a logical complexity of 4. We can't calculate its truth value by simply calculating from left to right. Instead, we must first determine the dependencies between those operators and first calculate the occurrences whose values only depend on the truth values of the component atomic sentence-letters, then calculate the next least complex subsentences, continuing outward following the patterns of dependency. Those dependency relations are exactly represented by the notion of a scope degree from the previous chapter: The occurrences of logical operators in a sentence with a scope degree of 1 are calculated first, those with a scope degree of 2 are calculated second, and so on. Applying those notions here, we can see that our sentence is primarily a negation; the first occurrence of '$\neg$' is the sentence's main operator. This occurrence has the highest degree of scope and so its value depends on the truth values of the complex sentence it operates on; namely, the conjunction '$(\neg p \wedge \neg q)$'. The entire sentence's truth value on a row is the truth value in the column under the first, leftmost occurrence of '$\neg$'. However, that will be the *last* truth value we calculate, as that truth value depends on the truth values of subsentences in the sentence, most immediately, in this case, the conjunction '$(\neg p \wedge \neg q)$'. And *that* sentence also has logical complexity. Before we can calculate *its* truth value, we must first know the truth values of *its* dependent parts; namely, the two negations of atomic sentence-letters.

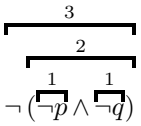
We first calculate the truth values of the least logically complex subsentences of a sentence and then successively work to the more logically complex. The least logically complex subsentences are always the atomic sentence-letters. The truth values for the atomic sentence-letters are not calculated but assigned. That assignment of truth values to atomic sentence-letters is called an **interpretation** of the sentence. A complete truth table for a sentence with two different atomic sentence-letters, such as our sentence here, will have four rows, as there are four possible interpretations for the sentence. (In general, the number of rows for a complete truth table for a sentence with n distinct atomic sentence-letters is 2^n. So, a sentence with exactly one distinct atomic sentence-letter has a truth table with 2 rows and a sentence with exactly 3 distinct atomic sentence-letters has a truth table with 8 rows.) So, the truth table frame for our sentence looks like this.

p	q	$\neg$	($\neg$	p	$\wedge$	$\neg$	q	)
T	T			T			T	
T	F			T			F	
F	T			F			T	
F	F			F			F	

A few things to notice. First, we pulled the atomic sentence-letters out to the left; call this the **reference column**. Creating a reference column makes it easier to consider all of the different permutations of truth values on those atomic sentence-letters, rather than trying to track the atomic sentence-letters through the sentence itself. While it ultimately does not matter what order of truth value permutations one uses, it is a useful to implement a convention based on the pattern above. We can describe that pattern as follows. First, create the reference column by ordering the atomic sentence-letters in the sentence alphabetically. Then, start with the rightmost atomic sentence-letter in the reference column and alternate T and F to fill all the rows. Next, move to the left one atomic sentence-letter (if there are more than one) and alternate *pairs* of Ts and Fs to fill all the rows. Then, move to the next left atomic sentence-letter (if there are more than two) and alternate *quadruples* of Ts and Fs to fill all the rows. Continue this pattern, each time alternating twice as many Ts and Fs as was done with the sentence-letter before, until the reference column is filled. The Truth Table Application on the LogicSpaces website requires that you use this convention and your truth table will be marked incorrect if you don't follow it, even if your answer is otherwise an accurate representation of the sentence's truth table. Second, we can read the rows of the truth table as follows: On the first row, where 'p' and 'q' are both true, the sentence has the truth value calculated in the column under the leftmost occurrence of '$\neg$'; on the second row, where 'p' is true and 'q' is false, the sentence has the truth value calculated in the column under the leftmost occurrence of '$\neg$', etc. Finally, we can now make more precise the notion of an interpretation of sentences in the language of PL mentioned above: An interpretation of a given sentence of the language of PL is just an assignment of truth values to atomic sentence-letters in that sentence. Truth tables are a visually convenient and accessible way of displaying all of the different possible interpretations of the sentences being interpreted.

We now begin to calculate the truth value of the sentence for each row. But that requires knowing which column to calculate first. We said above that we calculate from the least logical complexity to the most logically complex. We already set the truth values of the atomic sentence-letters, which are always the subsentences in a sentence with the least logical complexity. We now calculate the truth values of the occurrences of operates with the narrowest scope, working successively outward. We do this because the truth value we write in a column under an occurrence of a truth-functional operator is dependent on the truth value of the subsentence(s) that sentence operates on. So, in determining the order of calculation (i.e., which columns on a row to calculate in which order), it helps to first know the scope degrees of the occurrences of truth-functional

operators in the sentence, using the the techniques developed in the previous chapter, which we determine for our sample sentence below.



We now follow the scope degrees in calculating truth values, first calculating the columns for the occurrences of operators, appealing to their truth definitions captured by the schematic truth tables above, with the lowest scope degree and working our way up, as follows.

First calculation

		↱	↓		↱	↓		
		1			1			
p	q	¬	(¬	p	∧	¬	q)	
T	T		F	Ⓣ		F	T	
T	F		F	Ⓣ		T	F	
F	T		T	Ⓕ		F	T	
F	F		T	Ⓕ		T	F	

The first newly filled column in this table, the column under the second occurrence of '¬', operates on the circled truth value in the column immediately to its right, under the occurrence of 'p', "flipping" it. The second newly filled column in this table, the column under the last occurrence of '¬', operates on the boxed truth value in the column immediately to its right, under the occurrence of 'q', "flipping" it. We now have calculated the truth values of all of the logical operators with a scope degree of 1, in which case we have the necessary material to calculate the logical operators with a scope degree of 2.

Second calculation

		↓	⟵	↰⇈	↓			
		1			2	1		
p	q	¬	(¬	p	∧	¬	q)	
T	T		Ⓕ	T	F	Ⓕ	T	
T	F		Ⓕ	T	F	Ⓣ	F	
F	T		Ⓣ	F	F	Ⓕ	T	
F	F		Ⓣ	F	T	Ⓣ	F	

The newly filled column in this table, the column under the occurrence of '∧', operates on the pair of circled truth values in the columns calculated in the previous table. Notice that the function is exactly like the schematic truth table for conjunction given earlier: The truth value in the column under '∧' is true exactly when both of the circle truth values are true and is false when either of those cells is false. (So, A in the schematic truth table stands for '¬p' and B for '¬q' and so the truth value in the column under the second occurrence

of '¬', i.e., the second column in the truth table, corresponds to A's truth value in the schematic truth table and the truth value in the column under the third occurrence of '¬', i.e., the fifth column in the truth table, corresponds to B's truth value.) We are now in a position to continue to calculate the truth values of the operator with the next highest degree of scope, which in our case is the main operator of the sentence, the first occurrence of '¬'.

Final calculation

		↱	⟶	⟶	↓		
		3	1		2	1	
p	q	¬	(¬	p	∧	¬	q)
T	T	T	F	T	Ⓕ	F	T
T	F	T	F	T	Ⓕ	T	F
F	T	T	T	F	Ⓕ	F	T
F	F	F	T	F	Ⓣ	T	F

The newly filled column in this table, the column under the first occurrence of '¬', operates on the circled truth value in the column calculated in the previous table. (So, A in the schematic truth table for ¬ stands for the conjunction '$(\neg p \wedge \neg q)$' and so the truth value in the column under the occurrence of '∧' corresponds to A's truth value in the schematic truth table.) Because the leftmost occurrence of '¬' is the sentence's main operator, the truth values in the column under that occurrence represents the truth value of the sentence as a whole.

We have now completed our initial discussion of constructing complete truth tables for complex sentences. However, we have yet to discuss the truth definitions for our last two truth-functional operators, '→' and '↔'. A conditional sentence of the form $\ulcorner(A \rightarrow B)\urcorner$ is true (in an interpretation) when either the antecedent A is false (in that interpretation) or the consequent B is true (in that interpretation). It is false, then, exactly when A is true (in that interpretation) while B is false (in that interpretation). We can represent this truth definition with the following truth table.

A	→	B
T	T	T
T	F	F
F	T	T
F	T	F

Many claim that English conditional constructions using 'if, then' do not function like this. The sentence 'If UC-San Diego was founded before UC-Riverside, then UC-Riverside is older than UC-San Diego' seems intuitively false, but, as the antecedent is false (and the consequent true), the sentence is true when translated using '→'. Call this the problem of false antecedents.

Second, often an utterance of an 'if, then' sentence conveys causal or explanatory relations between the antecedent and the consequent. So, for example, we would naturally take an utterance of the sentence 'If you don't maintain proper air pressure in your tires, then you will get worse gas mileage' to mean that not maintaining proper air pressure *causes* lower gas mileage. However, no such

causal or explanatory relation is captured by the above truth definition of '$\rightarrow$'. Some take this to show that 'if, then' in English should not be translated as '$\rightarrow$'.

We can again appeal to Grice's theory of conversational implicature to help defend the traditional position. Whereas 'if, then' is a simple truth-function defined by the table above, ordinary English speakers and hearers conflate the material conditional 'If UC-San Diego was founded before UC-Riverside, then UC-Riverside is older than UC-San Diego' with the *subjunctive* conditional '*Had* UC-San Diego *been* founded before UC-Riverside, then UC-Riverside *would have been* older than UC-San Diego', which is false. Ordinary speakers use the material conditional sentence to mean the subjunctive conditional, even though the sentence itself does not have that meaning. Furthermore, the causal and explanatory connection often conveyed by an utterance of an 'if, then' sentence is part of the implicated meaning of the utterance but not the literal meaning of the sentence uttered.

Finally, a bi-conditional sentence of the form $\ulcorner(A \leftrightarrow B)\urcorner$ is true (in an interpretation) when A and B have the same truth value (in that interpretation). We can represent this truth definition with the following truth table.

A	$\leftrightarrow$	B
T	T	T
T	F	F
F	F	T
F	T	F

This completes our account of the semantics of the five truth-functional operators of the language of PL and our initial introduction to constructing truth tables. In the next section, we will continue our discussion of how to construct truth tables and then in the remaining sections of this chapter we will put our truth tables to work.

3.2 Calculating Truth Tables

The methods described in the previous section are sufficient for calculating the truth value of any sentence of PL, no matter how long and how complex, under every interpretation. We first identify the atomic sentence-letters in the sentence and create a reference column where all of the permutations of truth values on those atomic sentence-letters are considered. Each of these rows of truth tables represents an interpretation of the sentence and together they constitute every interpretation. We next determine the scope degrees for all of the occurrences of truth-functional operators in the sentence in order to determine the dependency relations between the truth values of the sentence's component parts and so the order in which the columns in the truth table are calculated. We then follow that order, applying the appropriate truth definitions presented in the sections above to determine the truth values of each component subsentence composing the sentence. We can then read the truth value of the sentence as a whole

under a given interpretation (i.e., when the atomic sentence-letters have a given assignment of truth values) by consulting the truth value in the cell under the sentence's main operator on the row corresponding to that assignment of truth values.

Let's illustrate the method more concretely by calculating the truth table for the sentence '$((p \to q) \to (q \leftrightarrow p))$'. We begin by constructing the reference columns and calculating the scope degrees for the truth-functional operators in the sentence.

		2		1		3	1		2		
p	q	$(\neg$	$(p$	$\to$	$q)$	$\to$	$(\neg$	q	$\leftrightarrow$	p	$))$
T	T		T		T			T		T	
T	F		T		F			F		T	
F	T		F		T			T		F	
F	F		F		F			F		F	

We next calculate the columns with the occurrences of operators with a scope degree of 1.

		2		1		3	1		2		
p	q	$(\neg$	$(p$	$\to$	$q)$	$\to$	$(\neg$	q	$\leftrightarrow$	p	$))$
T	T		T	T	T		F	T		T	
T	F		T	F	F		T	F		T	
F	T		F	T	T		F	T		F	
F	F		F	T	F		T	F		F	

We can now calculate the columns with the occurrences of operators with a scope degree of 2. (We will add arrows to illustrate which column each occurrence operates on.)

		↱	⟶	↓			↓	⟵	↰↱	↓	
		2		1		3	1		2		
p	q	$(\neg$	$(p$	$\to$	$q)$	$\to$	$(\neg$	q	$\leftrightarrow$	p	$))$
T	T	F	T	T	T		F	T	F	T	
T	F	T	T	F	F		T	F	T	T	
F	T	F	F	T	T		F	T	T	F	
F	F	F	F	T	F		T	F	F	F	

We now calculate the truth values under the main operator of the sentence, which is the rightmost occurrence of '$\to$'.

		↓	⟵	⟵	⟵	↰↱	⟶	⟶	↓		
		2		1		3	1		2		
p	q	$(\neg$	$(p$	$\rightarrow$	$q)$	$\rightarrow$	$(\neg$	q	$\leftrightarrow$	p	$))$
T	T	F	T	T	T	T	F	T	F	T	
T	F	T	T	F	F	T	T	F	T	T	
F	T	F	F	T	T	T	F	T	T	F	
F	F	F	F	T	F	T	T	F	F	F	

This example illustrates the procedure to use to construct any truth table.

Let's calculate the truth table for the sentence '$(((q \rightarrow (q \vee (p \wedge q))) \leftrightarrow \neg p) \wedge \neg\neg(p \wedge q))$'. We frist identify the atomic sentence-letters ('p' and 'q', in this case), create the reference column, and calculate the scope degree of each occurrence of a truth-functional operator in the sentence, as follows.

					3		2		1				4	1		5	3	2		1		
p	q	$(($	$($	q	$\rightarrow$	$(q$	$\vee$	$(p$	$\wedge$	q	$)))$	$\leftrightarrow$	$\neg$	$p)$	$\wedge$	$\neg$	$\neg$	$(p$	$\wedge$	q	$))$
T	T			T		T		T		T				T				T		T	
T	F			F		F		T		F				T				T		F	
F	T			T		T		F		T				F				F		T	
F	F			F		F		F		F				F				F		F	

We now apply the truth definitions for the truth-functional operators to calculate the column of truth values under each occurrence of a truth-functional operator in the sentence, following the order determined by the scope degrees. (We will calculate occurrences with a scope degree of 1-3 in the first table, combining three distinct steps, and then the remaining in the second table below.)

					3		2		1				4	1		5	3	2		1		
p	q	$(($	$($	q	$\rightarrow$	$(q$	$\vee$	$(p$	$\wedge$	q	$)))$	$\leftrightarrow$	$\neg$	$p)$	$\wedge$	$\neg$	$\neg$	$(p$	$\wedge$	q	$))$
T	T			T	T	T	T	T	T	T			F	T		T	F	T	T	T	
T	F			F	T	F	F	T	F	F			F	T		F	T	T	F	F	
F	T			T	T	T	T	F	F	T			T	F		F	T	F	F	T	
F	F			F	T	F	F	F	F	F			T	F		F	T	F	F	F	

We still must calculate the remaining two truth value columns. The first is the column under the only occurrence of '$\leftrightarrow$'. The two component subsentences that that symbol conjoins are '$(q \rightarrow (q \vee (p \wedge q)))$' and '$\neg p$'. So, the truth value in a cell in a row in the column under '$\leftrightarrow$' is a function of the truth values in the cells in the same row under the occurrence of '$\rightarrow$' and under the relevant occurrence of '$\neg$'. (Because the truth function is that defining the meaning of '$\leftrightarrow$', recall, the truth value is **T** just in case those two cells have the same truth values. The second remaining column to calculate is the column under the middle, or second, occurrence of '$\wedge$', which is the sentence's main operator. The two component subsentences that that symbol conjoins are '$((q \rightarrow (q \vee (p \wedge q))) \leftrightarrow \neg p)$' and '$\neg\neg(p \wedge q)$'. So, the truth value in a cell in a row under the column under the second occurrence of '$\wedge$' is a function of the truth values in the cells in the

same row under the occurrence of '$\leftrightarrow$' and the leftmost occurrence of '$\neg$' in the second subsentence. Because the truth function is that defining the meaning of '$\wedge$', recall, the truth value is **T** just in case both of those cells have **T**'s.

Complete Truth Table

			3		2		1				4	1		5	3	2		1			
p	q	(((	q	$\rightarrow$	(q	$\vee$	(p	$\wedge$	q	)))	$\leftrightarrow$	$\neg$	p)	$\wedge$	$\neg$	$\neg$	(p	$\wedge$	q	))
T	T		T	T	T	T	T	T	T		F	F	T	F	T	F	T	T	T		
T	F		F	T	F	F	T	F	F		F	F	T	F	F	T	T	F	F		
F	T		T	T	T	T	F	F	T		T	T	F	F	F	T	F	F	T		
F	F		F	T	F	F	F	F	F		T	T	F	F	F	T	F	F	F		

The main operator of the sentence is the middle or second occurrence of '$\wedge$' and so the truth values in the column represent the truth values of the entire sentence under each of its different interpretations.

There are two sources of complexity for a truth table. The first, illustrated by the sentence discussed immediately above, follows from the logical complexity of the sentence. The sentence whose table we just calculated, '$(((q \rightarrow (q \vee (p \wedge q))) \leftrightarrow \neg p) \wedge \neg\neg(p \wedge q))$', has a logical complexity of 9 and a highest scope degree of 5. The logical complexity of a sentence affects the number of columns a truth table will have. The above table, for example, has 16 columns, 7 of which are associated with atomic sentence-letters. The second source of complexity for a truth table, however, is set by the number distinct atomic sentence-letters that occur in the sentence. The number of distinct atomic sentence-letters in a sentence completely determines the number of rows in that sentence's complete truth table. For any sentence with one distinct atomic sentence-letter, whatever its logical complexity and so however many columns its truth table has and however many times that atomic sentence-letter occurs in that sentence, a complete truth table for that sentence has exactly two rows. For any sentence with two distinct atomic sentence-letters, a complete truth table for that sentence has exactly four rows. (Compare, for example, the truth table immediately above for our logically complex sentence with the truth table constructed earlier in this chapter for the simpler sentence '$\neg(\neg p \wedge \neg q)$'. Both tables have exactly four rows, as both sentences contain occurrences of exactly two distinct atomic sentence-letters.) For any sentence with three distinct atomic sentence-letters, a complete truth table for that sentence has exactly eight rows. In general, where n is the number of distinct atomic sentence-letters that occurrence in a sentence (note: we are *not* counting the number of occurrences of atomic sentence-letters!), a complete table for that sentence will have 2^n rows. So, the number of rows grows exponentially. While a sentence with four atomic sentence-letters has a complete truth table with 16 rows, a sentence with five atomic sentence-letters has a truth table with 32 rows and a sentence with 10 atomic sentence-letters has 1024 rows!

As an example, consider the sentence '$((p \vee q) \rightarrow (r \vee s))$', which has four distinct atomic sentence-letters and three occurrences of truth-functional operators (the minimum for a sentence containing four distinct atomic sentence-letters).

Its truth table frame looks like this.

						1		2		1		
p	q	r	s	((	p	$\vee$	q)	$\rightarrow$	(r	$\vee$	s	))
T	T	T	T		T		T		T		T	
T	T	T	F		T		T		T		F	
T	T	F	T		T		T		F		T	
T	T	F	F		T		T		F		F	
T	F	T	T		T		F		T		T	
T	F	T	F		T		F		T		F	
T	F	F	T		T		F		F		T	
T	F	F	F		T		F		F		F	
F	T	T	T		F		T		T		T	
F	T	T	F		F		T		T		F	
F	T	F	T		F		T		F		T	
F	T	F	F		F		T		F		F	
F	F	T	T		F		F		T		T	
F	F	T	F		F		F		T		F	
F	F	F	T		F		F		F		T	
F	F	F	F		F		F		F		F	

Any truth table for a sentence with four distinct atomic sentence-letters will have sixteen rows, as there are sixteen distinct interpretations (i.e., assignments of truth values to atomic sentence-letters) to consider. Notice that we constructed our reference column in accordance with the convention introduced above in section 3.0. We alternated T's and F's sixteen times under the atomic sentence-letter 's', pairs of T's and F's under 'r', quadruples of T's and F's under 'q', and octuples of T's and F's under 'p'. Once the truth table frame is in place, it is straightforward, if tedious, to calculate the columns under the truth-functional operators, as follows.

						1		2		1			
p	q	r	s	((	p	$\vee$	q)	$\rightarrow$	(r	$\vee$	s	))
T	T	T	T		T	T	T	T	T	T	T	
T	T	T	F		T	T	T	T	T	T	F	
T	T	F	T		T	T	T	T	F	T	T	
T	T	F	F		T	T	T	F	F	F	F	
T	F	T	T		T	T	F	T	T	T	T	
T	F	T	F		T	T	F	T	T	T	F	
T	F	F	T		T	T	F	T	F	T	T	
T	F	F	F		T	T	F	F	F	F	F	
F	T	T	T		F	T	T	T	T	T	T	
F	T	T	F		F	T	T	T	T	T	F	
F	T	F	T		F	T	T	T	F	T	T	
F	T	F	F		F	T	T	F	F	F	F	
F	F	T	T		F	F	F	T	T	T	T	
F	F	T	F		F	F	F	T	T	T	F	
F	F	F	T		F	F	F	T	F	T	T	
F	F	F	F		F	F	F	T	F	F	F	

This completes our discussion of the "basics" of truth table construction.

3.3 Reading Truth Tables

In previous sections we discussed the meanings of the five truth-functional operators of the language of PL and presented basic techniques for constructing complete truth tables for arbitrary sentences of the language of PL. In this section we discuss how to read a truth table. Suppose we wanted to know what truth value our complex conjunction from the previous section, '$((\neg p \leftrightarrow (q \rightarrow (q \vee (p \wedge q)))) \wedge (p \wedge q))$', has when '$p$' is false and '$q$' is true. Look back to that table. We identify the row in the truth table with that assignment of truth values, namely, the third row, and consult the truth value in the cell on that row in the column under the main operator of the sentence, which is the cyan colored truth value on the third row. We see that that truth value is false and so we say that the complex sentence is false when 'p' is false and 'q' is true. If we were asked to identify an interpretation in which the complex conjunction is false, we could then select that same row and say that the complex sentence is false in the interpretation in which 'p' is false and 'q' is true. (Indeed, our complex conjunction is false on all of the rows of its truth table, as the cells in the column under its main operator are all false, and so the sentence is false under every interpretation. We will discuss this property in the next section.)

While it is never incorrect to calculate the truth value for every cell of a truth table for a sentence and you *must* complete every cell of the truth table in the truth table application on the LogicSpaces website when you are doing your problem sets to get credit for your answers and to engage the feedback mechanisms on the site, which are activated only when the table is completely

filled out, it is sometimes unnecessary to calculate the truth value for every cell in order to determine and fully justify the truth value of the sentence as a whole on a given row. To see why, let's begin anew with our sentence, '$((\neg p \leftrightarrow (q \rightarrow (q \vee (p \wedge q)))) \wedge (p \wedge q))$'.

Recall that the sentence's main operator is the second occurrence of '$\wedge$', with '$(\neg p \leftrightarrow (q \rightarrow (q \vee (p \wedge q))))$' as the sentence's left conjunct and '$(p \wedge q)$' as its right conjunct. Just in virtue of knowing the truth table for '$\wedge$', we can infer that the whole sentence is false whenever at least one of these conjuncts is false. So, if we can show that one or the other of these conjuncts is false on a row, then we are justified in claiming that the whole sentence is false on that row, regardless of the truth value of the sentence's other conjunct. Because the right conjunct is significantly less logically complex, determining its truth value involves less calculation. It makes sense, then, to *first* calculate the truth value of the right conjunct to check if it is false. If it is false, as it is on all the rows on the truth table below with a box around the truth value in the column under the last occurrence of '$\wedge$' in the sentence, then we have shown that the complex conjunction itself is false, regardless of the truth value of the left conjunct on that row.

p	q	$((\neg p \leftrightarrow (q \rightarrow (q \vee (p \wedge q)))) \wedge (p \wedge q))$
T	T	? T T T
T	F	F F̲ F
F	T	F F F̲
F	F	F F F̲

Notice that we cannot make a determination as to whether or not the sentence is true or false on the first row from just determining that the right conjunct is true. For the first row, then, we must calculate the truth value of the left conjunct. We have still saved significant time, however, as we do not need to calculate the truth value of that subsentence for the other three rows of the truth table.

We might simply calculate a truth value for every cell of the first row. (Again, it is never a mistake to calculate the truth value for every cell.) But we can again employ shortcuts to save steps. The left conjunct is a bi-conditional. There aren't any shortcuts for calculating bi-conditionals, as determining the truth value of a bi-conditional requires determining whether or not the two sentences it operates on have the same truth value and we can find that out only by calculating the truth values of both sentences. So, we can justify the attribution of a truth value to the bi-conditional only after we have justified the truth value of both of its component sub-sentences, which are '$\neg p$' and the material conditional '$(q \rightarrow (q \vee (p \wedge q)))$'. The left sub-sentence has a logical complexity of 1 and so is easy to calculate. The right sub-sentence has a logical complexity of 3 and so we should look again for shortcuts. The main operator of the right sub-sentence is '$\rightarrow$' and so there are two things to look for: A false antecedent or a true consequent, either of which is sufficient to demonstrate

the truth of the conditional. The antecedent is the least logically complex, so it is good policy to start there. 'q' is true, which does not help us with our shortcut. So, we move to the consequent, which is a disjunction. We know that a disjunction is true if either of its disjuncts are true. So, we look for the least logically complex disjunct, which, in this case is 'q', and calculate that subsentence's truth value first. Because 'q' is true on the first row, the disjunction is true, which was the consequent of the conditional. So, because the consequent is true, the conditional itself is true, regardless of the truth value of the right disjunct and regardless of the truth value of the antecedent. So, focusing just on the truth values of the left conjunct on the first row, we have the following truth values.

p	q	(	$\neg$	p	$\leftrightarrow$	(q	$\rightarrow$	(q	$\vee$	(p	$\wedge$	q	))))
T	T		[F]	T			[T]	T	T				

We can now continue to calculate the truth value of the left conjunct. It's main operator is '$\leftrightarrow$', whose value is a function of the two boxed truth values in the table above, the first for '$\neg p$' and the second for the material conditional '$(q \rightarrow (q \vee (p \wedge q)))$'. Because those truth values are different from one another, the bi-conditional is false. And because one of the conjunct for the sentence's main operator '$\wedge$' is false, the sentence itself is false. We can thus finish the truth table as follows.

p	q	((	$\neg$	p	$\leftrightarrow$	(q	$\rightarrow$	(q	$\vee$	(p	$\wedge$	q	))))	$\wedge$	(p	$\wedge$	q))
T	T				[F]		T	T	T					F			
T	F													F		[F]	F
F	T													F	F	[F]	
F	F													F	F	[F]	

The truth table above is the minimal number of cells needed to justify the truth values given under the main operator, in the sense that calculating the truth values for any other cells is unnecessary and calculating fewer cells would have left the ultimate truth assignments unjustified. We have saved ourselves quite a bit of time in calculating truth values for cells compared to our earlier determination in which every cell was filled.

Here is a list of shortcut heuristics that can be used in calculated truth tables. (Again, remember that you should not use these shortcuts on your problem sets on LogicSpaces, where you must completely fill out your truth tables to receive credit for your work.) For conjunctions, look for a false conjunct, as that is sufficient to demonstrate the falsity of the conjunction. (One must calculate the truth values of *both* conjuncts to establish a conjunction's truth.) For disjunctions, look for a true disjunct, as that is sufficient to demonstrate the truth of the disjunction. (One must calculate the truth values of *both* disjuncts to establish a disjunction's falsity.) For material conditionals, look for either a false antecedent or a true consequent, as either individually is sufficient

to demonstrate the truth of the material conditional. (One must calculate the truth values of *both* antecedent and consequent to establish a material conditional's falsity.) There are no shortcuts for bi-conditionals, as demonstrating both the truth and the falsity of a bi-conditional requires calculating both sub-sentences. An overarching shortcut principle is to look to calculate the least logically complex sub-sentences first, assuming their values are relevant given the above heuristics.

3.4 Logical Status

We assume that you are now in a position to construct truth tables for any sentence of PL. We will now put that skill to work in establishing a number of important logical properties of sentences and sets of sentences of PL, ultimately a property directly related to the validity or invalidity of an argument. We will then be in a position to complete our journey: We will be able to start with a sequence of English sentences, an argument of English sentences, translate those sentences into the language of PL, and establish rigorously whether or not the argument is valid, at least relative to the translation offered. But we first will learn about establishing other properties.

We begin with what we will call a sentence's **logical status**. Every sentence of PL has exactly one of the following three statuses: Contingent, logically true, and logically false. A sentence is **contingent** when it is true in some interpretations and false in other interpretations. As an interpretation of a sentence of PL is just an assignment of truth values to atomic sentence-letters, this means that a contingent sentence is a sentence whose truth table contains a row where the sentence is true and another where it is false. A sentence is **logically true** when it is true in every interpretation. (This property, when applied to sentences of PL, is sometimes called 'a tautology'.) As an interpretation of a sentence of PL is just an assignment of truth values to atomic sentence-letters, this means that a logically true sentence is a sentence that is true on every row of its truth table. Finally, a sentence is **logically false** when it is false in every interpretation. As an interpretation of a sentence of PL is just an assignment of truth values to atomic sentence-letters, this means that a logically false sentence is a sentence that is false on every row of its truth table.

We said that every sentence of PL has exactly one of these statuses. For example, no sentence is both contingent and logically true. For suppose that it were contingent. Then, by the definition of contingency above, it is true on some row of its truth table and false on another. So, if it is contingent, it is not logically true, as there is a row of its truth table where it is false, and not logically false, as there is a row of its truth table where it is true. Furthermore, every sentence has at least one of these three statuses. So, the three status, contingent, logically true, and logically false, are *exclusive* and *exhaustive*.

Let's consider some very simple examples. Consider first the sentence '$(p \wedge \neg p)$', whose truth table is constructed below.

p	$(p$	$\wedge$	$\neg$	$p)$
T	T	F	F	T
F	F	F	T	F

Notice that '$(p \wedge \neg p)$' is false on every row of its truth table. (Remember: The sentence's truth value on the a row of its truth table is the truth value in the cell of the column under the sentence's main operator.) So, '$(p \wedge \neg p)$' is *logically false*; it is false *whatever* the truth value of its atomic sentence-letter 'p'.

Consider now the truth table for the negation of the above sentence, constructed below.

p	$\neg$	$(p$	$\wedge$	$\neg$	$p)$
T	T	T	F	F	T
F	T	F	F	T	F

Notice that '$\neg(p \wedge \neg p)$' is *true* on every row of its truth table and so is *logically true*. Indeed, this sentence is an instance of what is typically called *the law of noncontradiction* and is thought by many to be a basic "law of logic" and thought. (There are logics, paraconsistent logics, that allow for true contradictions and some very famous philosophers, both historically and in contemporary times, have argued that there are true contradictions.)

In previous sections above, we constructed truth tables for the following four sentences: '$\neg p$', '$\neg(\neg p \wedge \neg q)$', '$\neg p \leftrightarrow (q \rightarrow (q \vee (p \wedge q)))$', and '$((p \vee q) \rightarrow (r \vee s))$'. Looking back over those tables, we can see that the first two and fourth are contingent; the first sentence is false on the first row of its truth table and true on the second, the second sentence is true on the first three rows of its truth table and false on the fourth, and the fourth is true on the first row and false on the fourth (we won't bother to specify the truth values on all sixteen rows!). The third sentence, however, is logically false, as it is false on every row of its truth table.

Compare now the truth table for this third sentence with the truth table for the sentence '$(p \leftrightarrow (q \rightarrow (q \vee (p \wedge q))))$'.

Logical Status: Logically false

p	q	$(($	$\neg$	p	$\leftrightarrow$	$(q$	$\rightarrow$	$(q$	$\vee$	$(p$	$\wedge$	q	$))))$	$\wedge$	$(p$	$\wedge$	q	$))$
T	T		F	T	F	T	T	T	T	T	T	T		F	T	T	T	
T	F		F	T	F	F	T	F	F	T	F	F		F	T	F	F	
F	T		T	F	T	T	T	T	T	F	F	T		F	F	F	T	
F	F		T	F	T	F	T	F	F	F	F	F		F	F	F	F	

Logical Status: Contingent

	p	q	((	p	$\leftrightarrow$	(q	$\rightarrow$	(q	$\vee$	(p	$\wedge$	q	))))	$\wedge$	(p	$\wedge$	q	))
$\Rightarrow$	T	T		T	T	T	T	T	T	T	T	T		T	T	T	T	
$\Rightarrow$	T	F		T	T	F	T	F	F	T	F	F		F	T	F	F	
	F	T		F	F	T	T	T	T	F	F	T		F	F	F	T	
	F	F		F	F	F	T	F	F	F	F	F		F	F	F	F	

Because the second sentence is true on the first row and false on the second row, the sentence is contingent. These two rows by themselves suffice to demonstrate the contingency of the sentence, as they show that it is possible to make the sentence true and yet it is also possible to make it false. By contrast, nothing short of every row of the truth table demonstrates that the first sentence is logically false. And the same is true for a logically true sentence. This is because a logically true sentence is a sentence that is true on *every* row of its truth table and a logically false sentence is a sentence that is false on *every* row of its truth table.

3.5 Consistency

The next three properties we will be interested in are properties not just of individual sentences but of sets of sentences. A set of sentences is a collection of individual sentences. We name sets by putting brackets ('{' and '}') around their elements or *members*. So, '$\{p \rightarrow q, \neg(p \rightarrow q)\}$' picks out the set whose members are the sentences '$(p \rightarrow q)$' and '$\neg(p \rightarrow q)$', which is the same set picked out by '$\{\neg(p \rightarrow q), p \rightarrow q\}$', as they each have the same members. Finally, we shall represent an arbitrary set of sentences with the Greek letter 'Γ'.

A set of sentences Γ is **consistent** if it is possible for them to all be true together; that is, if there is an interpretation (for PL, an assignment of truth values to atomic sentence-letters) that makes all of the members of Γ true. A set of sentences Γ is **inconsistent**, on the other hand, when it is not possible for them to all be true together; that is, if, for every interpretation $\mathcal{I}$, at least one of the sentences in Γ is false in $\mathcal{I}$ (although it is not necessarily the same sentence from Γ that is false on each interpretation; it may be different sentences that are false in different interpretations).

We can use truth tables to determine consistency for sets of sentences of PL (assuming the set is finite in size). But we must first be able to construct what are called **joint truth tables**, which is a single truth table for multiple sentences. We must do this so that we can check for a single interpretation (i.e., a single row of the joint truth table) that makes all of the sentences in that set true. Let's consider first the set $\{p \rightarrow q, \neg(p \rightarrow q)\}$. We construct their joint truth table by writing the sentences side by side, creating a reference column by pulling out all of the atomic sentence-letters that occur in any of the sentences in the set, and then creating 2^n rows of the truth table, where n is the number of atomic sentence-letters.

p	q	$(\ p$	$\rightarrow$	$q)$	$\neg$	$(p$	$\rightarrow$	$q)$
T	T							
T	F							
F	T							
F	F							

We now calculate the truth value of each sentence separately on each row, as follows.

p	q	$(\ p$	$\rightarrow$	$q)$	$\neg$	$(p$	$\rightarrow$	$q)$
T	T	T	T	T				
T	F	T	F	F				
F	T	F	T	T				
F	F	F	T	F				

Now that we have completed the truth table for the first sentence, we can proceed to the second sentence.

p	q	$(\ p$	$\rightarrow$	$q)$	$\neg$	$(p$	$\rightarrow$	$q)$
T	T	T	T	T	F	T	T	T
T	F	T	F	F	T	T	F	F
F	T	F	T	T	F	F	T	T
F	F	F	T	F	F	F	T	F

With the joint truth table for these two sentences complete, we can now check to see if there is a row of that table where both sentences are true together. We see that at least one of the sentences is false on each row – the left sentence is false on the second row and the right sentence is false on the first, third, and fourth rows. So, we can conclude that the set of sentences $\{p \rightarrow q, \neg(p \rightarrow q)\}$ is *inconsistent*, as there is no row where they are all true together.

Let's now consider the set of sentences $\{(p \leftrightarrow q), (p \rightarrow \neg q), \neg(r \vee \neg r)\}$. This is a consistent set of sentences just in case there is a row of the joint truth table for these three sentences where they are all true together. So, we start by constructing their joint truth table. We see that there are three atomic sentence-letters composing these three sentences and so their joint truth table will have three reference columns and eight rows, as follows.

p	q	r	$(\ p \leftrightarrow q)$	$(p \rightarrow \neg\ q)$	$\neg\ (r \vee \neg\ r)$
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

We can now calculate the truth values of each sentence individually on each row. When we start to calculate the leftmost sentence on the first row, we see that the reference column contains information about the truth value of the atomic sentence-letter 'r', which does not occur in the sentence whose truth value we are calculating. The truth value of the sentence '$(p \leftrightarrow q)$' is *only* dependent to the truth values of 'p' and 'q'. So, whatever truth value is assigned to 'r' (or 's' or any other atomic sentence-letters not occuring in the sentence), the truth value of the leftmost sentence is only dependent on the truth values of 'p' and 'q'. Similarly, the truth value of the rightmost sentence is only dependent on the truth value of 'r'. When calculating the truth value of a sentence in an interpretation that specifies the truth values of atomic sentence-letters that do not occur in the sentence we are concerned with, then, we simply ignore the other truth values in the reference columns, as their values to not affect the truth value of the sentence. So, we fill out the truth table as follows.

p	q	r	$(\ p \leftrightarrow q)$	$(p \rightarrow \neg\ q)$	$\neg\ (r \vee \neg\ r)$
T	T	T	T T T	T F F T	F T T F T
T	T	F	T T T	T F F T	F F T T F
T	F	T	T F F	T T T F	F T T F T
T	F	F	T F F	T T T F	F F T T F
F	T	T	F F T	F T F T	F T T F T
F	T	F	F F T	F T F T	F F T T F
F	F	T	F T F	F T T F	F T T F T
F	F	F	F T F	F T T F	F F T T F

There is no row of the joint truth table where all three sentences are true, so the above table proves that the set of sentences $\{(p \leftrightarrow q), (p \rightarrow \neg q), \neg(r \vee \neg r)\}$ is inconsistent. In this case, there is a very good reason for this inconsistency: The sentence '$\neg(r \vee \neg r)$' is logically false (the above truth table establishes this fact as well, as the sentence is false on every row of its truth table). *Any* set of sentences that contains a logically false sentence as a member is inconsistent, as a logically false sentence is false on every row of its truth table and so there will not be a row where it is true and so, for any row of a joint truth table containing that sentence, there will be at least one sentence false on that row – namely, the logically false sentence!

Consider, however, the set of sentences $\{(p \leftrightarrow q), (p \rightarrow \neg q)\}$, which is the

above set of sentences less the logically false sentence '$\neg(r \vee \neg r)$'. (This set is said to be a **proper subset** of the earlier set, as every member of the second set is also a member of the first set, but there is a member of the first set that is not a member of the second.) This set *is* consistent. While the above joint truth table illustrates this fact, we can make it easier to see by eliminating the unnecessary information from that table, as follows.

	p	q	$(p$	$\leftrightarrow$	$q)$	$(p$	$\rightarrow$	$\neg$	$q)$
	T	T	T	T	T	T	F	F	T
	T	F	T	F	F	T	T	T	F
	F	T	F	F	T	F	T	F	T
$\Rightarrow$	F	F	F	T	F	F	T	T	F

Both sentences are true on the last row of the joint truth table. So, under the interpretation in which 'p' and 'q' are both false, the sentences '$(p \leftrightarrow q)$' and '$(p \rightarrow \neg q)$' are both true. So, there is an interpretation in which they are both true together. So, the set $\{(p \leftrightarrow q), (p \rightarrow \neg q)\}$ is consistent.

Here are two further examples to illustrate how we can use truth tables to determine consistency. Consider the set of sentences $\{(\neg p \rightarrow q), (p \rightarrow \neg q)\}$. We establish this set's consistency by constructing the following joint truth table.

	p	q	$(\neg$	p	$\rightarrow$	$q)$	$(p$	$\rightarrow$	$\neg$	$q)$
	T	T	F	T	T	T	T	F	F	T
$\Rightarrow$	T	F	F	T	T	F	T	T	T	F
$\Rightarrow$	F	T	T	F	T	T	F	T	F	T
	F	F	T	F	F	F	F	T	T	F

Because both sentences are true on the second and third rows of their joint truth table, the set of sentences $\{(\neg p \rightarrow q), (p \rightarrow \neg q)\}$ is consistent.

As a final example, consider the set of sentences $\{((p \vee q) \rightarrow r), (\neg r \wedge p)\}$. The joint truth table for the members of this set has eight rows, as there are three atomic sentence-letters whose truth values are all relevant to the truth values of these sentences, as follows.

p	q	r	$((p$	$\vee$	$q)$	$\rightarrow$	$r)$	$(\neg$	r	$\wedge$	$p)$
T	T	T	T	T	T	T	T	F	T	F	T
T	T	F	T	T	T	F	F	T	F	T	T
T	F	T	T	T	F	T	T	F	T	F	T
T	F	F	T	T	F	F	F	T	F	T	T
F	T	T	F	T	T	T	T	F	T	F	F
F	T	F	F	T	T	F	F	T	F	F	F
F	F	T	F	F	F	T	T	F	T	F	F
F	F	F	F	F	F	T	F	T	F	F	F

Because there are no rows of this table where both sentences are true ('$((p \vee q) \rightarrow r)$' is false on the second, fourth, and sixth rows and '$(\neg r \wedge p)$' is false on the

first, third, and fifth through eighth rows), the set $\{((p \vee q) \rightarrow r), (\neg r \wedge p)\}$ is inconsistent.

3.6 Logical Consequence and Logical Equivalence

One sentence A is a **logical consequence** of a set of sentences Γ, which we write as '$\Gamma \models A$', just in case it is absolutely impossible for all of the sentences in Γ to be true together while A is false; that is, just in case there are no interpretation that make all of the sentences of Γ true while making A false. Call an interpretation that makes all of the sentences in Γ true while making A false a **countermodel** to the claim that $\Gamma \models A$. (The notion of a countermodel is familiar from our discussion of validity in chapter 1. This is no accident; the notion of logical consequence is directly related to the notion of validity, as we will see below.) We can then say that $\Gamma \models A$ just in case there are no countermodels. On the flip side, A is *not* a logical consequence of Γ (written '$\Gamma \not\models A$') just in case it is possible for all of the sentences in Γ to be true while A is false; that is, when there is an interpretation under which all of the sentences in Γ are true and A is false. Such an interpretation is a *countermodel* and so we can say that $\Gamma \not\models A$ just in case there is a countermodel.

We characterized the notion of logical consequence generally, to apply to sentences in any formal language. Let's now restrict our attention to sentences of PL and assume that all of the sentences in Γ and A are sentences of PL. Then we can say that $\Gamma \models A$ when, for every row of their joint truth table, either at least one of the sentences in Γ is false or A is true, and we can say that $\Gamma \not\models A$ when, for some row of their joint truth table, every sentence in Γ is true while A is false. Thus, truth tables provide a very useful method for determining claims of logical consequence for sentences of PL.

To use truth tables to establish whether or not a sentence A is a logical cons$A$equence of a set of sentences Γ, we first set up a joint truth table for all of the sentences in Γ followed by A. Once the table is completed, we check for a row on which all of the sentences in Γ together true while A is false, which is a countermodel. If there is even a single countermodel, then the claim of logical consequence is false; if, on the other hand, there are no countermodels, then the claim of logical consequence is true. And, as long as we are dealing with a finite number of sentences, we can simply inspect each and every row to see whether or not there is a countermodel. So, truth tables are an effective method for determining and establishing and disproving claims of logical consequence. (It is a version of what we called in chapter 1 a *direct method* of establishing and disproving claims of validity.)

Consider the following example to illustrate this more concretely. Suppose you are asked to determine whether or not $\{\neg(p \wedge (q \vee r)), \neg r\} \models ? \neg q$. (We will use the convention of putting a question-mark after claims that are to be determined. Once the question is settled, we will either remove the question-mark, in the cases in which the claim is true, or change '$\models$' to '$\not\models$', in the cases in which the claim is false.) We first set up the joint truth table for the sentences.

Remember that we want to see if we can make all of the sentences to the left of '$\models$' true while making the sentence to the right false, as such an interpretation is a countermodel.

			T							**T**		**F**	?
p	q	r	$\neg$	$(p$	$\wedge$	$(q$	$\vee$	$r))$		$\neg$	r	$\neg$	q
T	T	T											
T	T	F											
T	F	T											
T	F	F											
F	T	T											
F	T	F											
F	F	T											
F	F	F											

We can now complete the joint truth table, as follows.

				T							**T**		**F**	?
	p	q	r	$\neg$	$(p$	$\wedge$	$(q$	$\vee$	$r))$		$\neg$	r	$\neg$	q
	T	T	T	F	T	T	T	T	T		F	T	F	T
	T	T	F	F	T	T	T	T	F		T	F	F	T
	T	F	T	F	T	T	F	T	T		F	T	T	F
	T	F	F	T	T	F	F	F	F		T	F	T	F
	F	T	T	T	F	F	T	T	T		F	T	F	T
$\Rightarrow$	F	T	F	T	F	F	T	T	F		T	F	F	T
	F	F	T	T	F	F	F	T	T		F	T	T	F
	F	F	F	T	F	F	F	F	F		T	F	T	F

We see that both '$\neg(p \wedge (q \vee r))$' and '$\neg r$' are true while '$\neg p$' is false on the sixth row and thus that there is a countermodel; namely, when 'p' and 'r' are both false and 'q' is true. So, we can conclude that $\{\neg(p \wedge (q \vee r)), \neg r\} \not\models \neg q$.

Consider now the following claim: $\{\neg(p \rightarrow (q \wedge r)), r\} \models ?\neg q$. To determine whether or not this claim is true, we set up a joint truth table for the three sentences much as before (as there are the same three atomic sentence-letters in these sentences) and calculate each sentence's truth value individually on each of those eight rows, as follows.

			T						**T**	**F**	?
p	q	r	$\neg$	$(p$	$\rightarrow$	$(q$	$\wedge$	$r))$	r	$\neg$	q
T	T	T	F	T	T	T	T	T	T	F	T
T	T	F	T	T	F	T	F	F	F	F	T
T	F	T	T	T	F	F	F	T	T	T	F
T	F	F	T	T	F	F	F	F	F	T	F
F	T	T	F	F	T	T	T	T	T	F	T
F	T	F	F	F	T	T	F	F	F	F	T
F	F	T	F	F	T	F	F	T	T	T	F
F	F	F	F	F	T	F	F	F	F	T	F

We see by inspecting each row that there is not interpretation that makes both '$\neg(p \rightarrow (q \wedge r))$' and '$r$' true while making '$\neg q$' false (we drew a box around a cell on each row with the wrong truth value) and hence there are not any countermodels. So, we can conclude that $\{\neg(p \rightarrow (q \wedge r)), r\} \models \neg q$.

We end our introduction to logical consequence by presenting three final, simpler examples without discussion.

Case 1: $\{p \rightarrow q, \neg q\} \models ? \neg p$.

		T			**T**		**F**	?
p	q	p	$\rightarrow$	q	$\neg$	q	$\neg$	p
T	T	T	T	T	F	T	F	T
T	F	T	F	F	T	F	F	T
F	T	F	T	T	F	T	T	F
F	F	F	T	F	T	F	T	F

Because there are no countermodels, $\{p \rightarrow q, \neg q\} \models \neg p$.

Case 2: $\{p \rightarrow q, \neg p\} \models ? \neg q$.

			T		**T**		**F**	?
	p	q	p	$\rightarrow$	q	$\neg$	p	$\neg$	q
	T	T	T	T	T	F	T	F	T
	T	F	T	F	F	F	T	T	F
$\Rightarrow$	F	T	F	T	T	T	F	F	T
	F	F	F	T	F	T	F	T	F

Because '$p \rightarrow q$' and '$\neg q$' are both true on the third row, when 'p' is false and 'q' is true, but '$\neg q$' is false, there is a countermodel. So, $\{p \rightarrow q, \neg q\} \not\models \neg p$.

Case 3: $\{p \vee \neg q, \neg q\} \models ? \neg p$.

					T			**T**		**F**	?
	p	q	p	$\vee$	$\neg$	q	$\neg$	q	$\neg$	p
	T	T	T	T	F	T	F	T	F	T
⇒	T	F	T	T	T	F	T	F	F	T
	F	T	F	F	F	T	F	T	T	F
	F	F	F	T	T	F	T	F	T	F

Because '$p \vee \neg q$' and '$\neg q$' are both true on the second row, when 'p' is true and 'q' is false, but '$\neg p$' is false, there is a countermodel. So, $\{p \vee \neg q, \neg q\} \not\models p$.

Our final notion, the notion of logical equivalence, can be defined directly in terms of our previous notion of logical consequence. A pair of sentences A and B are logically equivalent just in case $A \models B$ and $B \models A$. So, A and B are logically equivalent just in case, for every interpretation $\mathcal{I}$, the truth value of A in $\mathcal{I}$ is the same as the truth value of B in $\mathcal{I}$. The notion has a broader application, however, not applying to just pairs of sentences but to sets of sentences of any cardinality. (An example: The infinite set of sentences that consists of 'p', '$\neg\neg p$', '$\neg\neg\neg\neg p$', ..., is an infinite set of logically equivalent sentences.) So, our official definition of logical equivalence is this: A set of sentences Γ are **logically equivalent** just in case, for every interpretation $\mathcal{I}$, the truth value of any sentence in Γ in $\mathcal{I}$ is the same as the truth value of every sentence in Γ in $\mathcal{I}$. If all of the sentences concerned are sentences of PL, we can test for logical equivalence by constructing a joint truth table and checking whether or not there is a row where one of the sentences has a different truth value than the others. If there are, then the sentences are not logically equivalent; if there aren't, then the sentences are logically equivalent.

We saw above that $\{\neg(p \to (q \wedge r)), r\} \models \neg q$, in which case $(\neg(p \to (q \wedge r)) \wedge r) \models \neg q$, too. (If a sentence is a logical consequence of two sentences, then it is also the logical consequence of the conjunction of those sentences. This follows from the truth definition of '$\wedge$'. We will rely on this fact again below.) But are the sentences logically equivalent? That depends on whether the claim of logical consequence holds in the other direction as well; i.e., whether or not $\neg q \models ?(\neg(p \to (q \wedge r)) \wedge r)$. We make a joint truth table with the two sentences and check to see if there are any rows where the one is true and the other false.

	p	q	r	$(\neg$	$(p$	$\to$	$(q$	$\wedge$	$r))$	$\wedge$	$r)$	$\neg$	q
	T	T	T	F	T	T	T	T	T	F	T	F	T
	T	T	F	T	T	F	T	F	F	F	F	F	T
	T	F	T	T	T	F	F	F	T	T	T	T	F
⇒	T	F	F	T	T	F	F	F	F	F	F	T	F
	F	T	T	F	F	T	T	T	T	F	T	F	T
	F	T	F	F	F	T	T	F	F	F	F	F	T
⇒	F	F	T	F	F	T	F	F	T	F	T	T	F
⇒	F	F	F	F	F	T	F	F	F	F	F	T	F

Because the sentences have different truth values on rows 4, 7, and 8, the two sentences are *not logically equivalent*. Although $(\neg(p \to (q \wedge r)) \wedge r) \models \neg q$,

$\neg q \not\models (\neg(p \rightarrow (q \wedge r)) \wedge r)$.

Consider now the set of sentences $\{(p \vee q), \neg(\neg p \wedge \neg q), (\neg p \rightarrow q)\}$. We can determine whether or not this set is logically equivalent by constructing a joint truth table with all three sentences and checking for a row of that table where one of the sentences has a different truth value from the other two.

p	q	$(p$	$\vee$	$q)$	$\neg$	$(\neg$	p	$\wedge$	$\neg$	$q)$	$(\neg$	p	$\rightarrow$	$q)$
T	T	T	T	T	T	F	T	F		F	F	T	T	T
T	F	T	T	F	T	F	T	F		T	F	T	T	F
F	T	F	T	T	T	T	F	F		F	T	F	T	T
F	F	F	F	F	F	T	F	T		T	T	F	F	F

Because, for each row of the their joint truth table, the truth value of one of our three sentences on that row is the same as the others' (all three are true on the first three rows and all three are false on the fourth row), this truth table demonstrates that there are no interpretations where one of those sentences is true while the others false (or vice versa). So, the sentences in the set $\{(p \vee q)$, $\neg(\neg p \wedge \neg q)$, $(\neg p \rightarrow q)\}$ are logically equivalent.

There are important equivalences, sometimes call the DeMorgan Equivalences, after the 19^{th}-century mathematician and logician Augustus De Morgan, who originally studied them, or at least the equivalences involving only '$\neg$', '$\wedge$', and '$\vee$', that reveal interesting relationships between our five truth-functional operators. We'll first present and prove, using truth tables, the equivalences and then discuss their significance.

Conjunction

p	q	$(p$	$\wedge$	$q)$	$\neg$	$(\neg$	p	$\vee$	$\neg$	$q)$	$\neg$	$(p$	$\rightarrow$	$\neg$	$q)$
T	T	T	T	T	T	F	T	F	F	T	T	T	F	F	T
T	F	T	F	F	F	F	T	T	T	F	F	T	T	T	F
F	T	F	F	T	F	T	F	T	F	T	F	F	T	F	T
F	F	F	F	F	F	T	F	T	T	F	F	F	T	T	F

Disjunction

p	q	$(p$	$\vee$	$q)$	$\neg$	$(\neg$	p	$\wedge$	$\neg$	$q)$	$(\neg$	p	$\rightarrow$	$q)$
T	T	T	T	T	T	F	T	F	F	T	F	T	T	T
T	F	T	T	F	T	F	T	F	T	F	F	T	T	F
F	T	F	T	T	T	T	F	F	F	T	T	F	T	T
F	F	F	F	F	F	T	F	T	T	F	T	F	F	F

Material Bi-Conditional

p	q	$(p$	$\leftrightarrow$	$q)$	$\neg$	$(($	p	$\rightarrow$	$q)$	$\rightarrow$	$\neg$	$(q$	$\rightarrow$	p	$))$
T	T	T	T	T	T		T	T	T	F	F	T	T	T	
T	F	T	F	F	F		T	F	F	T	F	F	T	T	
F	T	F	F	T	F		F	T	T	T	T	T	F	F	
F	F	F	T	F	T		F	T	F	F	F	F	T	F	

There are also equivalences for '$\rightarrow$' and between '$\leftrightarrow$' and the other truth-functional operators that we will not display here.

We said above that these equivalences reveal several interesting relations between the different truth-functional operators. The first thing these equivalences show is that there is no difference in expressive power of the language of PL and a language that is just like PL but lacks the truth-functional operators '$\wedge$', '$\vee$', and '$\leftrightarrow$'. Here's why, focusing just on the loss of '$\wedge$', but where similar facts hold for the loss of the other two operators. For any conjunction $\ulcorner A \wedge B \urcorner$ in the language of PL, there is a sentence $\ulcorner \neg(A \rightarrow \neg B) \urcorner$ of both the languages of PL and the more minimal language that is logically equivalent. Because these two sentences are logically equivalent and because both languages only have truth-functions, the two sentences can be taken to be "expressively equivalent"; the differences between them are not semantically meaningful. More precisely, for any interpretation, the truth value of the first sentence in that interpretation is also the truth value of the second sentence in that interpretation. Similar points hold true for a language with only '$\neg$' and '$\wedge$' and for a language with only '$\neg$' and '$\vee$'; both of these languages are expressively equivalent to the language of PL. (However, any language lacking '$\neg$', assuming the only other logical vocabulary are any combination of the other four truth-functional operators of PL, is *expressively incomplete*, in the sense that that language lacks the resources to express, for example, '$\neg p$', there being no sentence in the deficient language that is logically equivalent to '$\neg p$'. Similarly, a language that has only '$\neg$' and '$\leftrightarrow$' is expressively incomplete, lacking the resources to express any of sentence of PL of the forms $\ulcorner (A \wedge B) \urcorner$, $\ulcorner (A \vee B) \urcorner$, and $\ulcorner (A \rightarrow B) \urcorner$.

In the previous paragraph we noted that there are three minimal propositional languages expressively equivalent with the language of PL. Each are otherwise just like the language of PL but for the following differences: The only two truth-functional operators of the first are '$\neg$' and '$\rightarrow$', of the second are '$\neg$' and '$\wedge$', and of the second are '$\neg$' and '$\vee$'. While these three minimal languages are expressively equivalent to the language of PL, the increase in vocabulary does bring with it an increase in *ease* of expression and, when we get to the proof-theory in the following chapter, an associated ease in manipulation involved in deriving theorems. For our purposes, then, the extra baggage is worth the cost. By contrast, some systems of propositional logic include other vocabulary, such as the symbol '$\perp$', meaning "contradiction" or "the impossible," which is not a truth-functional operator but instead a logical atomic-sentence that is assigned the truth value false in every interpretation. These languages are also expressively equivalent to the language of PL (and hence the three minimal languages as well) but fail to bring a significant increase in ease of

expression. (For example, the symbol '$\bot$' can be expressed in the language of PL as '$(p \wedge \neg p)$' in PL and as '$\neg(p \rightarrow \neg p)$' in the language considered above with '$\neg$' and '$\rightarrow$' as the only truth-functional operators.) Because the language of PL is *truth-functionally complete*, in the sense that any truth-function can be expressed by a combination of the logical symbols, it follows the three minimal languages are also truth-functionally complete. The considerations driving the choice between them, as well as languages with yet more bloated vocabularies, are competing ones of economy and ease of expression.

A second thing the above equivalences, and especially the first two, reveal is the way in which negations "distribute" across the other truth-functional operators. Consider, for example, the equivalent sentences '$\neg(\neg p \vee \neg q)$' and '$(p \wedge q)$' from above. Considering this equivalence reveals to us the nature of the distribution principle for '$\neg$' and '$\vee$': '$\neg$' can be distributed across a disjunction by negated each of the original disjuncts and changing the relevant occurrence of '$\vee$' to '$\wedge$'. This is revealed more clearly if we turn to a schematic version and begin with unnegated disjuncts in the original negated disjunction, as follows: $\lceil \neg(A \vee B) \rceil$. When we distribute the negation across the disjunction, this gives us the following equivalent sentence: $\lceil (\neg A \wedge \neg B) \rceil$. We can see that these sentence forms are equivalent with the following schematic joint truth table.

$\neg$	$(A$	$\vee$	$B)$	$(\neg$	A	$\wedge$	$\neg$	$B)$
F	T	T	T	F	T	F	F	T
F	T	T	F	F	T	F	T	F
F	F	T	T	T	F	F	F	T
T	F	F	F	T	F	T	T	F

(Note that we do not create reference columns because we do not know what the atomic parts of the sentences are. A and B are *schematic sentence-letters* ranging over every arbitrary sentence of PL, however simple or complex and with however many different atomic sentence-letters.) Similar principles hold for the distribution of negation across '$\wedge$', '$\rightarrow$', and even '$\leftrightarrow$', the first two of which can be read directly off the initial set of equivalences above.

3.7 Logical Consequence and Determining Validity

Recall from chapter 1 that we said that an argument is a sequence of sentences the last member of which is the conclusion and any earlier members of which are premises and that an argument is valid when it is absolutely impossible for all of its premises to be true while its conclusion false. So, when an argument is valid, its conclusion is a logical consequence of its premises. Here's a simple proof that highlights the close connection between the definitions of validity and logical consequence. Let Γ be a set of sentences that constitute the premises of an argument and let A be that argument's conclusion. If the argument is valid, then, by the definition of validity, it is absolutely impossible for all of the

sentences in Γ to be true while A is false. But then there are not countermodels. So, by the definition of logical consequence, $\Gamma \models A$. As we developed in the previous section a rigorous method for determining claims of logical consequence for sentences of PL, provided we can adequately translate an argument into the language of PL, then we will have a rigorous method for determining the validity of that argument as well.

We can now, finally, swing full circle. Suppose we have an argument of English sentences, like argument 3 from chapter 1, reproduced below.

> (1) George is sitting.
> (2) If George is eating, then George is sitting.
>
> Therefore, (C) George is eating.

Insofar as we can translate this argument into the language of PL and do adequate justice to its logical form (note: some arguments have logical complexity that are beyond the resources of PL to capture and so we shouldn't rely upon our translations of those arguments into the language of PL to fully reveal their validity), we can then use our above method to determine its validity.

We did not offer a translation of argument 3 in earlier chapters. But recalling our translation technique, we can see that the following does adequate justice to the logical form of argument 3.

> **Translation manual for argument 3**:
>
> $p \Longrightarrow$ George is sitting
> $q \Longrightarrow$ George is eating

We can now translate argument 3 as follows.

> (1) p
> (2) $(q \rightarrow p)$
>
> $\therefore$ (C) q

(Notation: '$\therefore$' stands for 'therefore', and thus marks the conclusion.) We can now rigorously establish the invalidity of argument 3 by proving $\{p, (q \rightarrow p)\} \not\models q$ using the truth table method described above, first constructing the joint truth table for those three sentences, checking to see whether or not there is a row of the interpretation that makes both 'p' and '$(q \rightarrow p)$' true while making 'q' false.

	p	q	p	$(q$	$\rightarrow$	$p)$	q
	T	T	T	T	T	T	T
$\Rightarrow$	T	F	T	F	T	T	F
	F	T	F	T	F	F	T
	F	F	F	F	T	F	F

When we complete the table, we see that there is a row of the joint truth table where both 'p' and '$(q \rightarrow p)$' are true and 'q' is false; namely, the second row, where 'p' is true and 'q' is false. We have now rigorously proven that argument

3 (and, indeed, *any* argument with the same logical form, brought out by our translation) is invalid. And we have shown this by proving that $\{p, q \to p\} \not\models q$.

Let's now consider argument 4, discussed in both chapters 1 and 2 above, also reproduced here.

Argument 4:

(1) If George is sitting, then he is eating or he is working on his logic problems.
(2) George is not eating.

Therefore, (C) If George is sitting, then he is working on his logic problems.

In chapter 2 we discussed how to translate this argument, the results of that discussion being reproduced below.

Translation manual for argument 4:

$p \Longrightarrow$ George is sitting

$q \Longrightarrow$ George is eating

$r \Longrightarrow$ George is working on George's logic problems

Translation for argument 4:

(1) $(p \to (q \vee r))$
(2) $\neg q$

$\therefore$ (C) $(p \to r)$

We can now use our truth table method to rigorously establish that argument 4 is valid by showing that $\{(p \to (q \vee r)), \neg q\} \models (p \to r)$, which we do with the following joint truth table.

p	q	r	$(p$	$\to$	$(q$	$\vee$	$r))$	$\neg$	q	$(p$	$\to$	$r)$
T	T	T	T	T	T	T	T	F	T	T	T	T
T	T	F	T	T	T	T	F	F	T	T	F	F
T	F	T	T	T	F	T	T	T	F	T	T	T
T	F	F	T	F	F	F	F	T	F	T	F	F
F	T	T	F	T	T	T	T	F	T	F	T	T
F	T	F	F	T	T	T	F	F	T	F	T	F
F	F	T	F	T	F	T	T	T	F	F	T	T
F	F	F	F	T	F	F	F	T	F	F	T	F

Because there are no rows where both '$(p \to (q \vee r))$' and '$\neg q$' are true while '$(p \to r)$' is false, we have shown that it is impossible for the first two to be true and the second false. So, $\{(p \to (q \vee r)), \neg q\} \models (p \to r)$. So, argument 4 is

valid.

It is important to stress a point mentioned above. Transferring the results of the truth table test for logical consequence to the validity or invalidity of the original natural language argument depends on the adequacy of the translation of those sentences into PL. There are two ways that a translation might be inadequate. The first is simple user error; that is, you mistranslate the sentences. For example, suppose that you are asked to determine the validity of an argument with the sentence 'Unless you study for the exam, you will not pass' and you translate that sentence as '$(p \wedge \neg q)$', where 'p'='You study for the exam' and 'q'='You will pass the exam'. Any results one concerning this sentence are irrelevant to the logical properties of the original sentence, as it is a poor translation. So, the first way in which the truth table method for determining the validity or invalidity of arguments of English sentences depends on the adequacy of the translation is that you actually translate the English sentence in a correct way. There is hardly a purely mechanical and effective procedure for translating sentences of English into the language of PL, the way that there is, say, for constructing truth tables for sentences of PL and using those truth tables to determine claims of logical consequence. Translation is as much art as it is science. This is a way in which, then, our methods for establishing validity and invalidity of arguments of English sentences are not fully rigorous.

The second way in which the adequacy of a translation might skew the ultimate assessment of the validity of the original argument can be illustrated with the following argument.

Argument 8:

(1) Everyone must study for the exam to pass.
(2) George didn't study for the exam.

Therefore, (C) George will not pass.

This argument is clearly valid. If it really is true that absolutely everyone needs to study for the exam to pass, then the same holds true for George in particular. So, if George didn't study, then he isn't going to pass. But now consider how we would translate that argument into the language of PL.

Translation manual for argument 8:

$p \Longrightarrow$ Everyone must study for the exam to pass

$q \Longrightarrow$ George studied for the exam

$r \Longrightarrow$ George will pass the exam

Translation for argument 8:

(1) p.
(2) $\neg q$.

$\therefore$ (C) $\neg r$

We can easily see that $\{p, \neg q\} \not\models \neg r$. In particular, when '$p$' is true, '$q$' is false, and '$r$' is true, the premises are both true but the conclusion is false, as can be seen by the following row of the joint truth table of those sentences.

	p	q	r	p	$\neg$	q	$\neg$	r
$\Rightarrow$	T	F	T	T	T	F	F	T

We might be tempted to conclude from this that argument 8 is invalid. After all, the best translation of argument 8 into the language of PL is the translation above and there is a clear countermodel to the validity of that translated argument. Yielding to this temptation is to be led astray. Argument 8 is valid, but its validity relies upon logical relations, in particular the logical relations between quantifiers like 'everyone' and singular terms like 'Bill' (not to mention the modal 'must'), that cannot be adequately represented with the resources of PL. PL *only* gives us the resources to represent truth-functional logical relations. Any more advanced, complex logical relations cannot be adequately represented in the language of PL. So, when we translate into the language of PL, we will obliterate the logical relations supporting the validity of the original target argument and turn that valid argument into an invalid argument through inadequate translation. Unlike the first inadequacy mentioned earlier, however, this inadequacy is due to the expressive limitations of the language of PL.

We cannot infer, then, that argument 8 is invalid from the fact that we have found that the best translation of the argument into the language of PL has a countermodel. Instead, we can infer that the original argument is either invalid or, if valid, its validity is due to nontruth-functional logical structures. So, if we are convinced that the argument is valid, are satisfied with our translation, and find that the translated argument is invalid, as its conclusion is not a logical consequence of its premises, then the onus is on us to uncover nontruth-functional structure present in the original argument that explains the validity of the original argument. Following this wonderful path will lead one to richer and more powerful logics, as we will inevitably be compelled to want to study, formalize, and model those logical structures. As we said before: The logic of truth-functions is but the trailhead to the interesting logical relations to study and learn and there remain, to this day, many such relations poorly understood and remaining to be adequately formalized.

3.8 How the Four Notions Interrelate

We have already seen how the notions of logical consequence and logical equivalence relate. But there are important relations between the notions of logical consequence and our other notions, as well. We will explore some of those relationships here.

Suppose that $\Gamma \models A$. Then, by the definition of logical consequence, there are no interpretations in which all of the sentences in Γ are true together while A is false. In that case, then, the set of sentences that is the union of Γ and $\ulcorner\neg A\urcorner$, written "$\Gamma \cup \neg A$", (i.e., the set of sentences that results from adding $\ulcorner\neg A\urcorner$ to the set of sentences Γ) is inconsistent. For suppose that set were consistent. Then, by the definition of consistency, there is an interpretation where all of the sentences are true. If $\ulcorner\neg A\urcorner$ is true in that interpretation, then, by the truth definition for '$\neg$', A is false. So, such an interpretation makes all of the sentences in Γ true while making A false. But, because $\Gamma \models A$, there are no such interpretations. This is a contradiction. So, $\Gamma \cup \neg A$ is inconsistent. And the converse is also true: If $\Gamma \cup \neg A$ is inconsistent, then $\Gamma \models A$, and for much the same reason. (We leave it as an exercise for the reader to work out the argument in detail.)

Here's the take away. If a sentence (A) is a logical consequence of some sentences (Γ), then the union of the *negation* of the original sentence (i.e., $\ulcorner\neg A\urcorner$) with the other sentences is inconsistent. If those sentences are inconsistent, then there is no interpretation in which they are all true together.

We now need to briefly digress. There is no restriction on the size of Γ (although we have restricted it to only sentences of PL); it may be finite or it may be infinite. We would like to transform Γ into a single sentence (a conjunction with each and only sentences in Γ as conjuncts), but we can't do that if Γ has an infinite cardinality, as every sentence of PL has a finite length and so no sentence of PL has an infinite number of conjuncts. Luckily for us, however, the famous mathematician and logician Kurt Gödel proved, in the 1930s, an important result, called the Compactness Theorem, that, for any sentence A and any set of sentences Γ, whatever the cardinality of Γ, such that $\Gamma \models A$, there is a finite subset of Γ, call it Θ, written '$\Theta \subset \Gamma$,' such that $\Theta \models A$. (Now, if Γ has an infinite cardinality, then Θ is a *proper* subset of Γ, written "$\Theta \subseteq \Gamma$." In the cases in which Γ has a finite cardinality, the compactness result it trivial, as the subset may well just be the set itself.) Thus, if $\Gamma \models A$, then, for some finite set of sentences $\{B_1, \ldots, B_n\}$ such that $\{B_1, \ldots, B_n\} \subset \Gamma$, $\{B_1, \ldots, B_n\} \models A$. This is an extremely fascinating result and we won't try to discuss its proof or more general significance here. We will instead help ourselves to it and note that this means that, because we can always turn a finite set of sentences $\{B_1, \ldots, B_n\}$ into a single sentence (a conjunction of all of the individual sentences in that set) that is satisfiable exactly when the original set is satisfiable, by compactness, it follows that, whatever the cardinality of Γ, if $\Gamma \models A$, then, for some finite set $\{B_1, \ldots, B_n\}$ such that $\{B_1, \ldots, B_n\} \subset \Gamma$, $(B_1 \wedge \ldots \wedge B_n) \models A$ (with the appropriate parentheses added, in any order, as all ways of resolving scope ambiguities between series of conjunctions are equivalent).

We can now state our next result. Suppose that $\Gamma \models A$ and that $\{B_1, \ldots, B_n\}$ is the finite set such that $\{B_1, \ldots, B_n\} \subset \Gamma$ that the Compactness Theorem guarantees exists. Then, by the truth definition for '$\wedge$', $(B_1 \wedge \ldots \wedge B_n) \models A$. But then the sentence $\ulcorner B_1 \wedge \ldots \wedge B_n) \wedge \neg A \urcorner$ is logically false. For suppose, for *reductio*, that $\ulcorner B_1 \wedge \ldots \wedge B_n) \wedge \neg A \urcorner$ were not logically false. Then there would be an interpretation in which all of the sentences in $\{B_1, \ldots, B_n\}$ are true and $\ulcorner \neg A \urcorner$ is true, in which case A is false. But such an interpretation is a countermodel, disproving our earlier supposition that $\{B_1, \ldots, B_n\} \models A$. Contradiction. So, $\ulcorner B_1 \wedge \ldots \wedge B_n \wedge \neg A \urcorner$ is logically false.

We have seen, then, that there are the following relations between our notions.

– $\Gamma \models A$, just in case

– $\Gamma \cup \neg A$ is inconsistent, just in case

– where $\{B_1, \ldots, B_n\}$ is a finite subset of Γ such that $\{B_1, \ldots, B_n\} \models A$, $\ulcorner (B_1 \wedge \ldots \wedge B_n \wedge \neg A) \urcorner$ is logically false.

While the notion of logical consequence bears the most direct relation to the notion of validity, which is of primary interest to the study of logic, we can see that that notion is directly definable in terms both of consistency and logical falsity individually. So, all three of our notions, logical falsity, consistency, and logical consequence, are intimately related to the notion of validity, and indeed to one another.

Suppose that a sentence A is logically true. We can then infer that, for any set of sentences Γ whatsoever, $\Gamma \models A$. For suppose that $\Gamma \not\models A$. Then there is a countermodel and so an interpretation such that all of the sentences in Γ are true in that interpretation while A is false in that interpretation. But if A is logically true, there is no interpretation in which it is false. Contradiction. So, if A is logically true, then $\Gamma \models A$.

Suppose now that a sentence A is logically false. Then, for any sentence B and any set of sentences Γ such that $A \in \Gamma$ (i.e., A is a member of the set of sentences Γ), $\Gamma \models B$. For suppose that $\Gamma \not\models B$. Then there is a countermodel and so an interpretation such that all of the sentences in Γ (including A!) are true in that interpretation and B is false in that interpretation. But if A is logically false, there is no such interpretation in which it is true. Contradiction. So, if A is logically false, then, for any sentence A and set of sentences Γ such that $\ulcorner A \urcorner \in \Gamma$, $\Gamma \models B$.

Finally, there is an important relationship between logical consequence and the material conditional (and logical equivalence and the material bi-conditional). If $\Gamma \models A$, then, for some finite subset of Γ, $\{B_1, \ldots, B_n\}$, such that $\{B_1, \ldots, B_n\} \models A$, for every interpretation $\mathcal{I}$, the material conditional $\ulcorner ((B_1 \wedge \ldots \wedge B_n) \rightarrow A) \urcorner$ (where parentheses inside the antecedent are added in any order to make the string grammatical, but the rightmost occurrence of '$\rightarrow$' is given widest scope) is true in $\mathcal{I}$; that is, $\ulcorner ((B_1 \wedge \ldots \wedge B_n) \rightarrow A) \urcorner$ is logically true. And, again, the converse is also true. So, if $\ulcorner ((B_1 \wedge \ldots \wedge B_n) \rightarrow A) \urcorner$ is a logical truth, then, for every

set of sentences Γ such that $\{B_1, \ldots, B_n\} \subset \Gamma$, $\Gamma \models A$. We can express these pair of facts by saying that $\{B_1, \ldots, B_n\} \models A$ (and so, for any set of sentences Γ such that $\{B_1, \ldots, B_n\} \subset \Gamma$, $\Gamma \models A$) just in case $\ulcorner((B_1 \wedge \ldots \wedge B_n) \rightarrow A)\urcorner$ is logically true (which we can write as $\models ((B_1 \wedge \ldots \wedge B_n) \rightarrow A)$). However, we should notice that the material conditional itself is still much weaker than the notion of logical consequence. Just because a material conditional is true in an interpretation, we cannot infer anything about whether or not the consequent of that conditional is a logical consequence of the antecedent of that conditional, as the conditional may well be merely contingently true, in which case the conditional is not logically true. This difference between mere material truth and what is sometimes called *strict* implication has been (and, in some circles at least, continues to be) a source of great philosophical confusion, confusion which did, however, lead to the discovery and development of modal logic, in the hands of C.I. Lewis, which has proven to be extremely fruitful and important. Despite those fruits, however, we should be careful to distinguish the material conditional from strict implication, even as we appreciate their connections.

3.9 Semantic Proofs

In chapter 1 we presented an informal, indirect method for establishing the validity of an argument by way of *reductio* on the assumption that a countermodel exists. We are now in position to make this method more rigorous, as we now have explicit truth definitions for the logical operators of PL that we can appeal to in the *reductio*. While the direct truth table method developed earlier in this chapter is always available to establish the validity of any argument of PL, we have already seen that that method can quickly become unruly, as the joint table needed for a claim of logical consequence can quickly become unwieldy, and for more advanced logics, including quantificational logic discussed in chapters 5-7 below, a direct semantic method for establishing validity is nonexistent. So, it is a good idea to introduce a rigorous indirect semantic method here, beginning with the simpler logic of PL.

Think back to your high school geometry class and your study of geometric proofs. You might have been asked to prove that a triangle has at most one obtuse angle. The simplest way to proceed is by *reductio*. So, we suppose that a triangle ABC has more than one obtuse angle. Name the three angles of our triangle be 'a', 'b', and 'c'. By the sum property of triangles, a+b+c=180. If two of those angles are obtuse, then they each have an angle greater than 90 degrees. But then their sum is greater than 180 and so, as all three angles of a triangle have a positive, nonzero measure, the sum of the angles a, b, and c is greater than 180, which contradicts our earlier claim that a+b+c=180. So, our assumption that there is a triangle with more than one obtuse angle must be rejected. So, every triangle has at most one obtuse angle.

Indirect semantic proofs of validity and hence of logical consequence take a similar form. They are arguments expressed in natural language that work by appealing to definitions and employ basic argumentative inferences. The defini-

tions we appeal to are the truth definitions that define the five logical operators of PL. We will present those truth definitions here in the form most readily applicable to semantic proofs, labeling each definition for ease of reference in our actual semantic proofs.

Truth Definitions:

$\neg$**T**: A negation of the form $\ulcorner\neg A\urcorner$ is true in an interpretation $\mathcal{I}$ just in case A is false in $\mathcal{I}$.

$\wedge$**T**: A conjunction of the form $\ulcorner(A \wedge B)\urcorner$ is true in an interpretation $\mathcal{I}$ just in case A is true in $\mathcal{I}$ and B is true in $\mathcal{I}$.

$\vee$**T**: A disjunction of the form $\ulcorner(A \vee B)\urcorner$ is true in an interpretation $\mathcal{I}$ just in case either A is true in $\mathcal{I}$ or B is true in $\mathcal{I}$.

$\rightarrow$**T**: A material conditional of the form $\ulcorner(A \rightarrow B)\urcorner$ is true in an interpretation $\mathcal{I}$ just in case either A is false in $\mathcal{I}$ or B is true in $\mathcal{I}$.

$\leftrightarrow$**T**: A material bi-conditional of the form $\ulcorner(A \leftrightarrow B)\urcorner$ is true in an interpretation $\mathcal{I}$ just in case either (A is true in $\mathcal{I}$ and B is true in $\mathcal{I}$) or (A is false in $\mathcal{I}$ and B is false in $\mathcal{I}$).

Remember that an interpretation for sentences of PL is an assignment of truth values to all of the atomic sentence-letters occurring in those sentences.

We will work through a simple example to illustrate how these definitions are applied. We will establish by semantic proof the following claim of logical consequence: $(p \wedge q) \models q$. (We include quotation marks, as we are talking about the sentences of PL and not using them. When you do semantic proofs yourself, you are more than welcome to omit the quotation marks, as there is little real room for use-mention confusions.)

Semantic Proof that $(p \wedge q) \models q$

Suppose, for *reductio*, that there is an interpretation $\mathcal{I}$ such that 1. '$(p \wedge q)$' is true in $\mathcal{I}$ while 2. 'q' is false in $\mathcal{I}$. By $\wedge$T and 1, 'p' is true in $\mathcal{I}$ and 'q' is true in $\mathcal{I}$. So, 'q' is both true (by 1) and false (by 2) in $\mathcal{I}$. Contradiction. So, there is no such interpretation as $\mathcal{I}$. So, $(p \wedge q) \models q$.

This illustrates the basic form that semantic proofs take.

We can identify three phases in a semantic proof. The first is the *reductio* assumption, which is the assumption of the existence of a countermodel to the claim of logical consequence in question. (This corresponds to the first sentence in the example above.) The second phase is the unpacking of the truth conditions imposed by that initial assumption. At this stage, we break apart the logical complexity involved in the initial sentences in the original claim by

systematically pealing off outermost operators by appeal to the appropriate truth conditions listed above. The third phase of a semantic proof involves explicitly drawing out the contradiction implicit in the initial assumption. (This corresponds to the third sentence in the example above.)

Let's now look at a slightly more complicated case that will motivate adding to our initial set of definitions an associated set of derivative falsity definitions. Suppose that we are asked to establish by semantic proof that $p \models (q \rightarrow p)$. We begin by making our *reductio* assumption that there is an interpretation $\mathcal{I}$ such that 1. 'p' is true in $\mathcal{I}$ while 2. '$(q \rightarrow p)$' is false in $\mathcal{I}$. Next, at the second stage, we unpack the truth conditions involved in that assumption. 'p' is logically simple and so we do not reduce its complexity. By 2 and $\rightarrow$T, it is not the case that either 'q' is true in $\mathcal{I}$ or 'p' is false in $\mathcal{I}$. In the third phase of our proof, we explicitly state the contradiction implicit in those conditions. But we now face a difficulty. While our conditions are contradictory, the form that they are in keeps that contradiction blurred. The problem is that we have stated the conditions, in particular, the conditions for the conditional, negatively. It is simpler when we state all of our conditions positively, using the truth value false for negative conditions on atomic sentences. This requires offering separate, derivative falsity conditions for each of our five complicated sentence forms, as follows.

Derivative Falsity Definitions:

$\neg$**F**: A negation of the form $\ulcorner\neg A\urcorner$ is false in an interpretation $\mathcal{I}$ just in case A is true in $\mathcal{I}$.

$\wedge$**F**: A conjunction of the form $\ulcorner(A \wedge B)\urcorner$ is false in an interpretation $\mathcal{I}$ just in case either A is false in $\mathcal{I}$ or B is false in $\mathcal{I}$.

$\vee$**F**: A disjunction of the form $\ulcorner(A \vee B)\urcorner$ is false in an interpretation $\mathcal{I}$ just in case both A is false in $\mathcal{I}$ and B is false in $\mathcal{I}$.

$\rightarrow$**F**: A material conditional of the form $\ulcorner(A \rightarrow B)\urcorner$ is false in an interpretation $\mathcal{I}$ just in case A is true in $\mathcal{I}$ and B is false in $\mathcal{I}$.

$\leftrightarrow$**F**: A material bi-conditional of the form $\ulcorner(A \leftrightarrow B)\urcorner$ is false in an interpretation $\mathcal{I}$ just in case either (A is true in $\mathcal{I}$ and B is false in $\mathcal{I}$) or (A is false in $\mathcal{I}$ and B is true in $\mathcal{I}$).

With $\rightarrow$F at our disposal, we can turn our above attempt at a semantic proof into the following.

Semantic Proof that $p \models (q \rightarrow p)$

Suppose, for *reductio*, that there is an interpretation $\mathcal{I}$ such that 1. 'p' is true in $\mathcal{I}$ while 2. '$(q \rightarrow p)$' is false in $\mathcal{I}$. By $\rightarrow$F and 2, 'q' is true in $\mathcal{I}$ and 'p' is false in $\mathcal{I}$. But then 'p' is both true (by 1) and false (by 2) in $\mathbf{I}$, which is a contradiction. So, there is no such interpretation as $\mathcal{I}$. So, $p \models (q \rightarrow p)$.

While the falsity definitions are extra, derivative machinery that does not increase the power of our system, including them makes it possible to give much clearer, shorter, and easier to follow proofs. So, the extra baggage is worthwhile.

We end this section by giving two more examples of more complicated semantic proofs, without any commentary, as models for more complex problems. The first example also gives an example of a useful form of argumentation that we can make use of, mentioned above: Namely, Argument by Cases, in which we infer that something follows from a disjunction by showing that it follows from each disjunct individually.

Example 1: Semantic proof that $((p \wedge q) \vee (r \wedge q)) \models (p \rightarrow q)$.

Phase 1: Suppose, for *reductio*, that there is an interpretation $\mathcal{I}$ such that 1. '$((p \wedge q) \vee (r \wedge q))$' is true in $\mathcal{I}$ while 2. '$(p \rightarrow q)$' is false in $\mathcal{I}$. Phase 2: By 1 and $\vee$T, either

–1a. '$(p \wedge q)$' is true in $\mathcal{I}$, and so, by $\wedge$T, 'p' is true in $\mathcal{I}$ and 'q' is true in $\mathcal{I}$, or

–1b. '$(r \wedge q)$' is true in $\mathcal{I}$, and so, by $\wedge$T, 'r' is true in $\mathcal{I}$ and 'q' is true in $\mathcal{I}$. By 2 and $\rightarrow$F, 'p' is true in $\mathcal{I}$ and 'q' is false in $\mathcal{I}$. Phase 3: Because 1, either 1a or 1b. Suppose that 1a is the case. Then 'q' is true in $\mathcal{I}$. But by 2, 'q' is false in $\mathcal{I}$. Contradiction. Suppose then that 1b is is the case. Then 'q' is true in $\mathcal{I}$. But by 2, 'q' is false in $\mathcal{I}$. Contradiction. So, either way, a contradiction follows. So, there is no such interpretation as $\mathcal{I}$. So, $((p \wedge q) \vee (r \wedge q)) \models (p \rightarrow q)$.

Example 2: Semantic proof that $\models (((p \wedge \neg q) \vee q) \leftrightarrow (p \vee q))$.

Phase 1: Suppose, for *reductio*, that there is an interpretation $\mathcal{I}$ such that '$(((p \wedge \neg q) \vee q) \leftrightarrow (p \vee q))$' is false in $\mathcal{I}$. Phase 2: By $\leftrightarrow$F, either

1. both '$((p \wedge \neg q) \vee q)$' is true in $\mathcal{I}$, and so, by $\vee$T, either:

   –1a. '$(p \wedge \neg q)$' is true in $\mathcal{I}$, in which case, by $\wedge$T, 'p' is true in $\mathcal{I}$ and '$\neg q$' is true in $\mathcal{I}$, and so, by $\neg$T, 'q' is false in $\mathcal{I}$, or

   –1b. 'q' is true in $\mathcal{I}$

   and '$(p \vee q)$' is false in $\mathcal{I}$, in which case, by $\vee$F, 'p' is false in $\mathcal{I}$ and 'q' is false in $\mathcal{I}$

   or

2. both '$((p \wedge \neg q) \vee q)$' is false in $\mathcal{I}$, in which case, by $\vee$F, both '$(p \wedge \neg q)$' is false in $\mathcal{I}$, in which case, by $\wedge$F, either:

   –2a. 'p' is false in $\mathcal{I}$ or

   –2b. '$\neg q$' is false in $\mathcal{I}$, and so, by $\neg$F, 'q' is true in $\mathcal{I}$

and 'q' is false in $\mathcal{I}$,

and '$(p \vee q)$' is true in $\mathcal{I}$, and so, by $\vee$T, either:

–2c. 'p' is true in $\mathcal{I}$ or

–2d. 'q' is true in $\mathcal{I}$.

We can summarize this as follows. Either [1. 'p' is false in $\mathcal{I}$ and 'q' is false in $\mathcal{I}$ and either (1a. 'p' is true in $\mathcal{I}$ or 1b. 'q' is false in $\mathcal{I}$)] or [2. 'q' is false in $\mathcal{I}$ and either (2a. 'p' is false in $\mathcal{I}$ or 2b. 'q' is true in $\mathcal{I}$) and either (2c. 'p' is true in $\mathcal{I}$ or 2d. 'q' is true in $\mathcal{I}$)].

Suppose first that 1 is the case. Then either 1a is the case or 1b is the case. Suppose that 1a is the case. Then 'p' is both true and false in $\mathcal{I}$. Contradiction. Suppose then that 1b is the case. Then 'q' is both true and false in $\mathcal{I}$. Contradiction. Either way, contradiction. So, if 1 is the case, then there is a contradiction. Suppose that 2 is the case. Then either 2a or 2b is the case. Suppose first that 2b is the case. Then 'q' is both true and false in $\mathcal{I}$. Contradiction. Suppose then the 2a is the case. Either 2c or 2d is the case. Suppose that 2c is the case. Then 'p' is both true and false in $\mathcal{I}$. Contradiction. Suppose that 2d is the case. Then 'q' is both true and false in $\mathcal{I}$. Contradiction. Either way, contradiction. So, if 2a is the case, contradiction. So, if 2 is the case, contradiction. So, whether 1 or 2 is the case, there is a contradiction. So, our initial assumption leads to a contradiction. So, there is no such interpretation as $\mathcal{I}$. So, $\models (((p \wedge \neg q) \vee q) \leftrightarrow (p \vee q))$.

Our last example illustrates several lessons. First, it illustrates the importance of using formatting and parentheses to keep one's semantic proof organized. Second, it illustrates how to establish that a sentence is a logical truth: One argues that the assumption that the sentence is false is contradictory.

This ends our discussion of semantic proofs and, more generally, the semantics of PL.

CHAPTER REVIEW

The basic truth definitions for our five truth-functional operators.

A sentence of the form $\ulcorner \neg A \urcorner$ is true in an interpretation $\mathcal{I}$ just in case A is false in $\mathcal{I}$.

A sentence of the form $\ulcorner (A \wedge B) \urcorner$ is true in an interpretation $\mathcal{I}$ just in case both A is true in $\mathcal{I}$ and B is true in $\mathcal{I}$.

A sentence of the form $\ulcorner (A \vee B) \urcorner$ is true in an interpretation $\mathcal{I}$ just in case either A is true in $\mathcal{I}$ or B is true in $\mathcal{I}$.

A sentence of the form $\ulcorner (A \rightarrow B) \urcorner$ is true in an interpretation $\mathcal{I}$ just in case either A is false in $\mathcal{I}$ or B is true in $\mathcal{I}$.

A sentence of the form $\ulcorner (A \leftrightarrow B) \urcorner$ is true in an interpretation $\mathcal{I}$ just in case either (both A is true in $\mathcal{I}$ and B is true in $\mathcal{I}$) or (both A is false in $\mathcal{I}$ and B is false in $\mathcal{I}$).

Truth table representations of the truth definitions of the truth-functional operators.

$\neg$	A
F	T
T	F

A	$\wedge$	B
T	T	T
T	F	F
F	F	T
F	F	F

A	$\vee$	B
T	T	T
T	T	F
F	T	T
F	F	F

A	$\rightarrow$	B
T	T	T
T	F	F
F	T	T
F	T	F

A	$\leftrightarrow$	B
T	T	T
T	F	F
F	F	T
F	T	F

Truth Table Cheat Sheet

- **Logical Status** A sentence is *logically true* when it is true in every interpretation and so every cell in the column under its main operator of its truth table has a T. (Slogan: Always true.) A sentence is *contingent* when it is true in some interpretations and false in other interpretations and so at least one cell in the column under its main operator of its truth table has a T and some cell has an F. (Slogan: Sometimes true and sometimes false.) A sentence is *logically false* when it is false in every interpretation and so every cell in the column under its main operator of its truth table has an F. (Slogan: Always false.)

- **Consistency** A set of sentences is *consistent* when there is an interpretation in which they are all true together and so there is at least one row of their joint truth table where the cell under each sentence's main operator is a T. (Slogan: Sometimes all true together.) A set of sentences is *inconsistent* when there are no interpretations in which they are all true together and so for each row of their joint truth table, at least one of the cells under the main operator of at least one of the sentences is an F. (Slogan: Never all true together.)

- **Logical Consequence** $\Gamma \models B$ (read 'B is a logical consequence of Γ) when there are no countermodels, where a *countermodel* is an interpretation in which all of the sentences in Γ are true while B is false, and so for every row of their joint truth table either: i) for at least one of the sentences in Γ is false or ii) B is true. $\Gamma \not\models B$, on the other hand, when there is a countermodel and so for some row of their joint truth table, for each sentence in Γ, the cell in the column under its main operator is a T, while the cell in the column under the main operator of B is an F.

- **Logical Equivalence** A set of sentences are *logically equivalent* just in case, for every interpretation, the truth value of any sentence in that set in that interpretation is the same as the truth value of every other sentence in that set in that interpretation. So, for every row of their joint truth table, the truth value in the cell under the column of the main operator of any sentence in the set is the same as the truth value of any other cell in the column under the main operator of every sentence in the set. (Note: This coordination only needs to be *within* a row.) A set of sentences are *logically inequivalent* (or not logically equivalent) just in case, for at least one interpretation, the truth value of one of the sentences in that interpretation is different from the truth value of another sentence in that interpretation. So, for some row of their joint truth table, the truth value in the cell under the column of the main operator of any sentence in the set is different from the truth value of the cell in the column under the main operator of some other sentence in the set. (Note: This difference *must* be within a row.)

Chapter 4

Proof Theory for Propositional Logic

In chapter 1, we said that the later chapters of this text contain a development of two formal methods for establishing the validity of an argument. We presented one of those methods in chapter 3: The truth table method. In this chapter, we will discuss the second method: The method of derivation. The two methods are, by design, always in agreement, in the sense that it is never the case that one delivers the verdict that an argument is valid while the other delivers the verdict that that argument is invalid. There are, however, two important differences between the methods. The first and most fundamental difference is that the truth table method is a *semantic* method, operating on the *truth conditions* of the sentences involved. The second method, by contrast, is based entirely on the *syntactic* forms of the sentences involved and so is nonsemantic. This difference has had profound differences in the history of philosophy and has led some to think that we should only operate with a proof-theoretic notion of provability, eschewing semantic notions of truth and meaning that are independent of provability. A second important difference is that, whereas the first method is powerful enough to deliver both positive and negative answers concerning an argument's validity, the method of derivation only delivers positive answers. That is, we can use truth tables both to discover countermodels and so prove that an argument is invalid and prove the nonexistence of a countermodel and so prove that an argument is valid; by contrast, the method of derivation can *only* prove that an argument is valid. These two differences are very significant and central to what is called *metalogic*, which is the study of logical systems and methods for proving that a logical system has certain interesting features. We will return to the first difference briefly below.

A **derivation** is a series of lines, each containing a sentence, beginning with a set of premises and ending with a conclusion, where each step after the premises is a line justified by an inference rule and so earlier lines in the derivation or is an instance of an axiom. In this chapter we develop what is

called a *Natural Deduction* system for the logic of truth-functions. A natural deduction system consists of a series of inference rules, which are principles that justify the transitions between sentences.

Suppose, then, that we have an argument that we would like to prove is valid. We use our proof theory to do this by showing how the conclusion can be *derived from* the premises using the inference rules of our system. However, this assumes that the inference rules of our system preserve truth, in the sense that any sentence A that we derive from some set of sentences Γ using these inference rules is such that there is absolutely no way for all of the sentences in Γ to be true and A false. This should sound familiar, as it is exactly our familiar notion of logical consequence. If our inference rules preserve truth in this sense, then we can rest assured that an argument is valid provided we can derive its conclusion from its premises using legal applications of our inference rules.

Let's illustrate a derivation somewhat more concretely but still in completely general terms. Suppose that our premises are the sentences $\{B_1, \ldots, B_n\}$ and that our conclusion is A. A derivation of A from $\{B_1, \ldots, B_n\}$ will then take the following form.

1. $\mathsf{B_1}$	**Hyp**
⋮	
n. $\mathsf{B_n}$	**Hyp**
u. C	**justification**
⋮	
v. A	**justification**

This is called a *Fitch format*, after the great logician Frederic Fitch, who, among his many other contributions, was one of the early developers of the form of proof theory we shall be employing. The horizontal "Fitch" bar marks a break between the assumptions, the lines justified by the Hypothesis Inference Rule, to be discussed below, and the derived sentences. Each line below the Fitch bar must then be derived, using an inference rule, again to be discussed below, as justification. Insofar as those inference rules preserve truth, in the sense that employing those inference rules will never involve transition for truth to falsity, then every sentence below the Fitch bar, including the conclusion A, is a logical consequence of the sentences above the Fitch bar. So, by showing how one can legally derive A from $\{B_1, \ldots, B_n\}$, we have shown that the original argument is indeed valid. We can use the following notation as shorthand for the claim that A is derivable or provable from $\{B_1, \ldots, B_n\}$ in the proof theory S: $\{B_1, \ldots, B_n\} \vdash_S A$.

Let's make this more concrete. We will describe a proof theory for PL that we shall call $\mathcal{F}$, for Fitch. $\mathcal{F}$ is a natural deduction proof theory. This means that it consists of a set of inference rules, as opposed to an axiomatic systems which consist of a set of axioms and fewer inference rules (and for standard axiomatic systems for PL, a single inference rule). Natural deduction

systems are almost universally taught in introductory logic classes, as they are far easier to use in constructing derivations than their axiomatic counterparts. By contrast, axiomatic systems are standardly discussed in metalogic classes and preferred by most logicians, because it is typically much easier to prove that the systems have interesting features. Regardless of the merits of alternatives, we will unapologetically focus exclusively on a natural deduction system in this text. (And we are of the minority who think that natural deduction systems are preferable to axiomatic systems for both beginning and advanced work in logic.)

In total, $\mathcal{F}$ has eleven inference rules, a pair of inference rules for each of our truth-functional operators $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ and a special inference rule Hyp. We will start by introducing and describing seven basic rules, working through several sample derivations that use only those rules in the next section, and then introduce the remaining four more complex rules, completing our presentation of $\mathcal{F}$, in the following sections.

4.1 Seven Basic Inference Rules

We will first present schematically the individual inference rules, discuss them in general terms, and then provide example derivations that employ these rules. You should refer to this table, a complete version of which is included at the end of this chapter, as you are constructing your derivations, as they represent the form that every use of the inference rules in our system takes. Let A and B be any sentences of PL, whether a simple atomic sentence-letter or a logically complex sentence of PL. Furthermore, suppose that each line without a specified justification in the schemas below has a legal justification, where what counts as a legal justification is described by the schematic rules themselves. Our aim here is to specify the functioning of the particular inference rule whose form is being illustrated.

Basic Inference Rules for $\mathcal{F}$

Hyp

$i.$ A

$\wedge$Elim

$i.$ A $\wedge$ B
$\triangleright$ $j.$ A/B **$\wedge$Elim:** i

$\vee$Intro

$i.$ A/B
$\triangleright$ $j.$ A $\vee$ B **$\vee$Intro:** i

$\neg$Elim

$i.$ $\neg\neg$A
$\triangleright$ $j.$ A **$\neg$Elim:** i

$\rightarrow$Elim

$i.$ A $\rightarrow$ B
$j.$ A
$\triangleright$ $k.$ B **$\rightarrow$Elim:** i, j

$\wedge$Intro

$i.$ A
$j.$ B
$\triangleright$ $k.$ A $\wedge$ B **$\wedge$Intro:** i, j

$\leftrightarrow$Elim

$i.$ A $\leftrightarrow$ B
$j.$ A/B
$\triangleright$ $k.$ B/A **$\leftrightarrow$Elim:** i, j

Hyp is a unique rule. It tells us that, at any point in our derivation, we are entitled to write any sentence whatsoever, provided we put it above a Fitch-bar. Sentences justified by **Hyp** are then "assumed" to be true. In the first instance, these sentences are the premises of the argument whose validity we are concerned to establish. In the next section, however, we will describe, in connection with the more complex rules, other occasions where a sentence is introduced into our derivation as an assumption (as well will see, they are introduced as *temporary* assumptions that are then discharged upon the use of one of the more complex inference rules to be described below). Because this rule is so easy to abuse and its abuse misleads beginning students, you will see that the proof application on LogicSpaces that you will use to do your derivation problems controls the way that you can use Hyp. In fact, you will never directly use the rule. All of your premises will be given in the initial state of your derivation and the temporary assumptions mentioned above are added only when you select one of the complex inference rules to justify a later line. When you turn to doing your derivations on paper, in an exam setting, for example, you will be set free. We encourage you, then, to be mindful and attentive to the ways the website application controls the introduction of assumptions and so the use of **Hyp**.

We mentioned earlier that, for each of our five truth-functional operators, there is a pair of inference rules: An introduction and an elimination rule. We can think of the introduction rule for an operator as providing the paradigmatic form of justification for adding as a later line of our derivation a sentence whose main operator is that operator. So, consider $\wedge$**Intro**. Provided one has as two earlier lines of one's derivation the sentence A and the sentence B, one is then entitled to the conjunction of those two sentences, the sentence $\ulcorner(A \wedge B)\urcorner$, as a later line, justified by citing the inference rule $\wedge$**Intro** and the two earlier lines containing the sentences A and B. (Notice that a legal, albeit degenerate, use of this rule involves citing the same line containing the sentence A twice in order to derive the conjunction $\ulcorner(A \wedge A)\urcorner$ by $\wedge$**Intro**.) It does not matter the order of the earlier lines on which A and B occur. So, even if B occurs on an earlier line than the line where A occurs, we can still use $\wedge$**Intro** to justify $\ulcorner(A \wedge B)\urcorner$ as well as $\ulcorner(B \wedge A)\urcorner$.

The inference rules of $\mathcal{F}$ are keyed exclusively to the syntactic forms of the sentences occurring in the derivations. So, $\wedge$**Intro**, for example, can *only* be cited to justify a conjunction, i.e., a sentence whose main operator is '$\wedge$', and *only* when the sentence on one of the cited lines *exactly* matches, symbol for symbol, one of the conjuncts of that conjunction and the sentence on the other cited line *exactly* matches, symbol for symbol, the other conjunct. The sentences cannot be mere close cousin or even logically equivalent; they must exactly match. Notice as well that each of those conjuncts must be the *entire* sentence on those earlier lines; that is, it is not enough that, say, A occurs as a component of a larger sentence on an earlier line. A, and A itself, must be the whole sentence that occurs on the earlier line.

An introduction rule, we said, provides the paradigmatic way in which a sentence whose main operator is the truth-function associated with that rule gets added as later lines of a derivation; that is, it is the paradigmatic way

to write, as some later line of our derivation, a line with a sentence whose main operator is that operator. An elimination rule, by contrast, provides the paradigmatic inference that one can make with a sentence whose main operator is that operator as an earlier line of the derivation. (Intuitively, an elimination rule for an operator tell us what we can *do with* a sentence with an operator as its main connective when it is already an earlier line of the derivation.) Consider ∧**Elim**. The rule says that, provided one has as an earlier line of one's derivation the sentence ⌈$(A \wedge B)$⌉, then one is entitled to write either of the conjuncts A or B of that conjunction individually as a new, later line of one's derivation, justified by citing the inference rule ∧**Intro** and the earlier line containing the conjunction.

To illustrate the use of these rules more concretely, let's consider how we would derive '$(p \wedge r)$' from '$((p \wedge q) \wedge r)$'. We will first present the derivation in its completed form and then talk through each step.

Derivation establishing that $((p \wedge q) \wedge r) \vdash (p \wedge r)$

1. $((\mathsf{p} \wedge \mathsf{q}) \wedge \mathsf{r})$	**Hyp**
2. $(\mathsf{p} \wedge \mathsf{q})$	∧**Elim**:1
3. p	∧**Elim**:2
4. r	∧**Elim**:1
5. $(\mathsf{p} \wedge \mathsf{r})$	∧**Intro**:3,4

Notice that the main connective of the beginning premise, '$((p \wedge q) \wedge r)$', is the rightmost occurrence of '∧' and so that sentence is an instance of the form ⌈$(A \wedge B)$⌉, where A stands proxy for '$(p \wedge q)$' and B stands proxy for 'r'. So, on line 2, we derive the left conjunct from that conjunction, A in the schema of the rule from earlier in this section. Because the sentence on line 2 is another conjunction, we can again use ∧**Elim** to justify as a later line one of its conjuncts, the sentence 'p', on line 3. After another use of ∧**Elim** again on line 1, this time deriving the right conjunct of the conjunction on line 1, we use ∧**Intro** to conjoin the sentences on two earlier lines into the conjunction on line 5, justifying our ultimate conclusion '$(p \wedge r)$'.

In our discussion of the derivation, we started from the top of the page and worked our way down, which is likely the way you yourself worked through the derivation even before reading our discussion. When we have a completed derivation before us, that is quite naturally the way we would read it, trying to understand each transition from premises to the ultimate conclusion. In simpler cases many can simply "see" from the premises to the conclusion, quickly filling in each step. And some people are very good at seeing the transition path from top to bottom even in difficult derivations. However, for the vast majority of us and for the vast majority of derivations, it will be far easier to construct the derivation in the opposite order, from the bottom to the top. Here is what we mean by that, returning again to our above derivation. (Pretend that you have not just seen the completed derivation and that you are facing the problem anew, having to establish that '$(p \wedge r)$' is provable from '$((p \wedge q) \wedge r)$' without already

knowing how to do it.) You should start at the bottom of the derivation, with the conclusion, asking what introduction rule would justify the ultimate conclusion of the derivation, '$(p \wedge r)$'. Because the main operator of that sentence is '$\wedge$', we should expect that we will use '$\wedge$**Intro**, giving us the following structure.

1. $((\mathsf{p} \wedge \mathsf{q}) \wedge \mathsf{r})$		**Hyp**
⋮		
z. $(\mathsf{p} \wedge \mathsf{r})$		$\wedge$**Intro**:-,-

Notice that we have left a "gap" between the premise and the conclusion, as we expect to have to add extra intermediate steps in order to transition to the conclusion, as none of our inference rules will take us directly from the sentence on line 1 to our ultimate conclusion. Rather than starting with our premise, however, we have started with our conclusion, devising a plan to justify that sentence. Our first attempt should always be to use the main operator of the conclusion, in this case, $\wedge$**Intro**. We now return to the rule schematic from the beginning of this section and see that, in order to use $\wedge$**Intro** to justify a sentence of the form $\ulcorner(A \wedge B)\urcorner$, where, in this particular case, A stands proxy for 'p' and B stands proxy for 'r', we must have two earlier lines, one with the sentence A and the other the sentence B. This, then, gives us new subgoals as earlier lines, as follows.

1. $((\mathsf{p} \wedge \mathsf{q}) \wedge \mathsf{r})$		**Hyp**
⋮		
x. p	newsubgoal	
y. r	newsubgoal	
z. $(\mathsf{p} \wedge \mathsf{r})$		$\wedge$**Intro**:x,y

Provided we can justify lines x and y, then, we have a plan for justifying line z: Use $\wedge$**Intro** on lines x and y. We now turn our attention to justifying those new subgoals.

We once again apply our strategy of working from the bottom up to those new subgoals, identifying each sentence's main operator in turn and planning to use the associated introduction rule to justify it. In this case, however, each of our new subgoals, both 'p' and 'r', are atomic sentence-letters without any logical structure. So, we cannot use an introduction rule to justify either of these sentences; we must instead use an elimination rule. (Remember, an introduction rule can only be used to justify a sentence whose main operator is that operator associated with that rule. So, atomic sentence-letters can *never* be justified using an introduction rule.) Using an elimination rule requires that we cite an earlier line of our derivation. So, we look at the earlier lines — line 1 is the only earlier line — and look to see which combination of elimination rules might allow us to justify our new subgoals.

Let's start with 'r'. Because 'r' is the right conjunct of the conjunction on line 1, we know that we can justify that new subgoal directly, by $\wedge$**Elim** on line 1. We can see that by again consulting the schematic form of $\wedge$**Elim** and

noticing that it entitles us to detach either conjunct from a conjunction. Let's move, then, to our other subgoal, 'p'. 'p' is a component of the sentence on line 1, but it is embedded within multiple layers of logical complexity. So, we cannot directly derive this subgoal from line 1. We must instead use at least one intermediate step, pulling out the left conjunct in which 'p' is embedded, as follows.

1. $((\mathsf{p} \wedge \mathsf{q}) \wedge \mathsf{r})$		**Hyp**
2. $(\mathsf{p} \wedge \mathsf{q})$		$\wedge$**Elim**:1
x. p	newsubgoal	
y. r		$\wedge$**Elim**:1
z. $(\mathsf{p} \wedge \mathsf{r})$		$\wedge$**Intro**:x,y

Now that we have line 2 added to the party, we have more sentences we can use our elimination rules on in an effort to justify 'p' on line x. 'p' is the left conjunct of the conjunction on line 2. So, we can use $\wedge$**Elim** on line 2 to justify line x, giving us the completed derivation presented earlier.

The above discussion gives concrete applications of our first three basic inference rules, Hyp, $\wedge$**Intro**, and $\wedge$**Elim**. It also illustrates a basic strategy in derivation construction: Working from the bottom up. Notice, however, that while we work from the bottom up in filling the steps of the transition from initial premises to final conclusion, for any line i of our derivation, we must only cite earlier lines, lines above line i, to justify line i. We cannot, then, justify a line in terms of anything that comes after it, even though, in using the bottom up strategy, we may have added that line as a new subgoal for the purposes of justifying some later line. Mastering the bottom up strategy early, and learning it with the easier derivations in the beginning, will serve you well when you turn to more complicated derivations. While you may be tempted to simply sail from the premises through to the conclusion in the easier derivations at the beginning, already "seeing" the path of transitions, we encourage you to practice the technique of working from the bottom up here. A little discipline in the earlier, easier derivations will pay dividends when you reach the harder derivations to come.

Let's now consider another derivation that employs more of our basic inference rules, deriving the sentence '$((p \vee r) \wedge (q \vee r))$' from '$(\neg\neg p \wedge q)$'. We will again present the derivation in its entirety and then talk through the derivation in its order of construction, using the method of working from the bottom up.

Derivation establishing that $(\neg\neg p \wedge q) \vdash ((p \vee r) \wedge (q \vee r))$

1. $(\neg\neg \mathsf{p} \wedge \mathsf{q})$		**Hyp**
2. $\neg\neg \mathsf{p}$		$\wedge$**Elim**:1
3. p		$\neg$**Elim**:2
4. $(\mathsf{p} \vee \mathsf{r})$		$\vee$**Intro**:3
5. q		$\wedge$**Elim**:1
6. $(\mathsf{q} \vee \mathsf{r})$		$\vee$**Intro**:5
7. $((\mathsf{p} \vee \mathsf{r}) \wedge (\mathsf{q} \vee \mathsf{r}))$		$\wedge$**Intro**:4,6

This derivation employs Hyp, $\wedge$**Intro**, $\wedge$**Elim**, $\neg$**Elim**, and $\vee$**Intro**. We will illustrate applications of the remaining two basic inference rules later in this section.

In constructing the derivation, we begin by putting the premise(s) and ultimate conclusion into our Fitch format, leaving space between those two points in order to work the path from conclusion back up to premise(s) using our inference rules.

1. $(\neg\neg \mathsf{p} \wedge \mathsf{q})$		**Hyp**
z. $((\mathsf{p} \vee \mathsf{r}) \wedge (\mathsf{q} \vee \mathsf{r}))$		

We now look at the conclusion, identify its main operator, and form a default plan to use the introduction rule associated with that operator in order to justify that line. As the main operator of our conclusion is '$\wedge$', this means that we should plan to use $\wedge$**Intro** in order to justify our ultimate conclusion. We consult the schematic form of that rule and we see that using $\wedge$**Intro** to justify a conjunction requires that we have earlier in our derivation a line where one of the conjuncts of that conjunction occur and a line where the other of the conjuncts occur. This then provides us with both a (default) plan for justifying our conclusion and new subgoals that serve as means of implementing that plan: In our case, deriving both '$(p \vee r)$' and '$(q \vee r)$' as earlier lines of our derivation.

1. $(\neg\neg \mathsf{p} \wedge \mathsf{q})$		**Hyp**
x. $(\mathsf{p} \vee \mathsf{r})$	newsubgoal	
y. $(\mathsf{q} \vee \mathsf{r})$	newsubgoal	
z. $((\mathsf{p} \vee \mathsf{r}) \wedge (\mathsf{q} \vee \mathsf{r}))$		$\wedge$**Intro**:x,y

We now have two new subgoals: Derive on line x '$(p \vee r)$' and derive on line y '$(q \vee r)$'. We turn our attention to each of those lines in turn, applying the same basic strategy again to achieve those new goals.

Let's begin with line y. (It typically doesn't matter which of the two subgoals we start with.) We identify the main operator of the sentence we wish to derive and form the default plan of using the introduction rule associated with that operator. In the case of the sentence on line y, the main operator is '$\vee$', so we

form the default plan of using ∨**Intro** to derive that line. We again consult the schematic form of that rule and we see that using ∨**Intro** to justify a disjunction requires that we have earlier in our derivation a line that contains one or the other of the disjuncts of that disjunction. In our case, then, we must have as a line above line y either the sentence 'q' or the sentence 'r'. This provides us, then with the following extra structure for our derivations.

1. $(\neg\neg p \wedge q)$		**Hyp**
x. $(p \vee r)$	newsubgoal	
yy. q OR r	newsubgoal	
y. $(q \vee r)$		∨**Intro**:yy
z. $((p \vee r) \wedge (q \vee r))$		∧**Intro**:x,y

We have two possibilities for the sentence that will occur on line yy: Either 'q' or 'r'. This is because the use of ∨**Intro** has this kind of disjunctive structure: You can justify a disjunction by ∨**Intro** provided you have at least one of the disjuncts of that disjunction as an earlier line. We consider each path individually, settling on whichever one pans out and discarding the other.

We begin, then, with the subgoal of deriving 'r' on line yy. We again apply our strategy of working from the bottom up to this subgoal. However, because the sentence is an atomic sentence-letter without logical complexity, we cannot justify it using any of our introduction rules. We instead change directions, now looking at the earlier lines of our derivation, the sentences on lines 1, in particular, but also x, and ask ourselves whether or not there are any *elimination* rules that can be used to derive our subgoal 'r'. Because 'r' does not occur as a subsentence of the sentence on line 1 (i.e., 'r' does not occur in the sentence '$(\neg\neg p \wedge q)$'), we know that we cannot directly derive 'r' from 1. Now, it is true that 'r' occurs in the sentence on line x (i.e., the sentence '$(p \vee r)$'). However, it occurs as the disjunct of a disjunction and we do not yet have an elimination rule for '$\vee$'. (And even when we do have ∨**Elim**, we will see that it will *not* allow us to simply infer one of the disjuncts from a disjunction, which is a very good thing indeed, as such a rule would not preserve truth, as, in general $A \vee B \not\models A$.) We should suspect, then, that 'r' is a dead possibility and pursue the other option of deriving 'q'.

Our subgoal now is to derive 'q'. Once again, that sentence lacks logical complexity, so we look at the earlier sentences in the derivation with an eye to use an elimination rule to infer 'q'. 'q' occurs as the conjunct of the conjunction '$(\neg\neg p \wedge q)$' on line 1. When we consult the schematic form of ∧**Elim**, we see that it allows us to infer either conjunct from a conjunction. So, we know that we can derive 'q' on line yy by ∧**Elim** citing line 1.

1. $(\neg\neg \mathsf{p} \wedge \mathsf{q})$	**Hyp**
x. $(\mathsf{p} \vee \mathsf{r})$ newsubgoal	
yy. q	$\wedge$**Elim**:1
y. $(\mathsf{q} \vee \mathsf{r})$	$\vee$**Intro**:yy
z. $((\mathsf{p} \vee \mathsf{r}) \wedge (\mathsf{q} \vee \mathsf{r}))$	$\wedge$**Intro**:x,y

We now have a plan for justifying the subgoal on line y and can turn our attention to our first subgoal, deriving '$(p \vee r)$' on line x.

The main operator of '$(p \vee r)$' is '$\vee$' and so our default plan for deriving the line should be to use $\vee$**Intro**, in which case we would need to derive at some earlier line at least one of the disjuncts, either 'p' or 'r', as follows.

1. $(\neg\neg \mathsf{p} \wedge \mathsf{q})$	**Hyp**
xx. p OR r newsubgoal	
x. $(\mathsf{p} \vee \mathsf{r})$	$\vee$**Intro**:xx
yy. q	$\wedge$**Elim**:1
y. $(\mathsf{q} \vee \mathsf{r})$	$\vee$**Intro**:yy
z. $((\mathsf{p} \vee \mathsf{r}) \wedge (\mathsf{q} \vee \mathsf{r}))$	$\wedge$**Intro**:x,y

As before, we consider each of these options in turn. Each involve justifying atomic sentence-letters, which we know requires using elimination rules on earlier lines as opposed to any introduction rule. Our attempt to derive 'r' from line 1, the only line above line xx, suffers the same problem discussed above: Namely, 'r' does not occur any where in that sentence. So, we eliminate that option and turn to the other, deriving 'p'. We see that 'p' occurs as a constituent of the sentence on line 1, although it is deeply embedded inside more complex sentences, first as the conjunct of the conjunction and then inside a pair of negations. So, we cannot directly derive 'p' from 1. We must instead decompose that logical structure by a sequence of applications of elimination rules, first applying the elimination rule for the outermost logical operator and then working our way inward until 'p' has been completely detached and isolated by itself. That is, to infer 'p' from '$(\neg\neg p \wedge q)$', we cannot simply derive it directly in a single step, as our rules can only be applied to the main operator of a sentence. So, our first step is to isolate '$\neg\neg p$' using $\wedge$**Elim** and then derive 'p' from that derived sentence using $\neg$**Elim**. We will break this sequence down more explicitly.

1. $(\neg\neg \mathsf{p} \wedge \mathsf{q})$	**Hyp**
xxx. $\neg\neg \mathsf{p}$	$\wedge$**Elim**:1
xx. p newsubgoal	
x. $(\mathsf{p} \vee \mathsf{r})$	$\vee$**Intro**:xx
yy. q	$\wedge$**Elim**:1
y. $(\mathsf{q} \vee \mathsf{r})$	$\vee$**Intro**:yy
z. $((\mathsf{p} \vee \mathsf{r}) \wedge (\mathsf{q} \vee \mathsf{r}))$	$\wedge$**Intro**:x,y

Now that we have '$\neg\neg p$' on line xxx, above line xx, we have more material to

use to justify 'p' on line xx. Because 'p' occurs immediately inside a double negation '$\neg\neg$', we can use $\neg$**Elim** to justify the sentence on line xx from line xxx, completing our derivation and giving us the derivation initially presented. Notice that the sentence on the line cited by a use of $\neg$**Elim** must, in every case, be a double negated sentence; i.e., a sentence whose leftmost two symbols are a stack of two negation operators '$\neg\neg$'.

We have gone through the procedure in such excruciating detail to thoroughly illustrate the method of working from the bottom up. It is, of course, not always necessary to employ the strategy in such an exacting fashion. For example, you might have "just seen" the attempt to derive 'r' on lines xx and yy would not work and wouldn't even bother to explicitly consider it as an option. Or, you might have just "seen" the path from line 1 to line xx. But, to repeat our earlier advice, whether or not that is the case, we suggest that you develop good habits in constructing your derivations while the problems are easier, as those habits will pay dividends when the problems become more difficult. We encourage you, that is, to get in the habit of working from the bottom up even if you already see the path from the premises to the conclusion.

We end this section with a final sample derivation, this one employing our remaining two basic inference rules $\rightarrow$**Elim** and $\leftrightarrow$**Elim**.

Derivation establishing that $\{p, ((p \rightarrow (q \wedge r)) \leftrightarrow p)\} \vdash (r \wedge q)$

1. p	**Hyp**
2. $((\mathsf{p} \rightarrow (\mathsf{q} \wedge \mathsf{r})) \leftrightarrow \mathsf{p})$	**Hyp**
3. $(\mathsf{p} \rightarrow (\mathsf{q} \wedge \mathsf{r}))$	$\leftrightarrow$**Elim**:1,2
4. $(\mathsf{q} \wedge \mathsf{r})$	$\rightarrow$**Elim**:3,1
5. q	$\wedge$**Elim**:4
6. r	$\wedge$**Elim**:4
7. $(\mathsf{r} \wedge \mathsf{q})$	$\wedge$**Intro**:5,6

This derivation involves two premises and so two lines justified by Hyp. In employing the strategy of working from the bottom up to construct this derivation, one would first add lines 5 and 6 as subgoals as required by the plan to use $\wedge$**Intro** to justify the ultimate conclusion '$(r \wedge q)$', as the main operator of that sentence is '$\wedge$'. Because those new subgoals are both atomic sentence-letters, one would switch from searching for an introduction rule to use to justify those lines and search for elimination rules to use on earlier lines, and in particular, lines 1 and 2. Because the logical structure of the sentences on those earlier lines is '$\leftrightarrow$', '$\rightarrow$', '$\wedge$', in that order, we are led to add line 3, then line 4, and finally justify lines 5 and 6 by $\wedge$**Intro**.

Basic Inference Rules Derivation Problems

For each of the following, construct a derivation that establishes the relevant claim of provability. The problems in this section should employ only some of the seven Basic Inference rules. The problems are grouped by difficulty

crudely based on the shortest length derivation that establishes the claim. The number to the right marks the length of the optimal derivation. Note that your derivation may well be longer. Remember: For any true claim of provability, there are infinitely many distinct derivations that establish the claim. So, if your derivation is longer than the marked optimal length, your answer is not necessarily wrong. However, when it comes to derivation construction, shorter is better and so you should always strive to reach your conclusion in the fewest possible steps.

Easy

1. $p \wedge q, r \vdash p \wedge r$ (4)
2. $p \wedge q, r \vdash p \wedge r$ (4)
3. $(p \vee q) \rightarrow r, p \vdash r$ (4)

Medium

4. $((\neg\neg\neg q \rightarrow p) \leftrightarrow \neg q)^{(}\neg\neg\neg q \wedge r) \vdash p$ (7)
5. $p \wedge (q \wedge r) \vdash (p \wedge r) \wedge q$ (7)
6. $p \rightarrow q, q \rightarrow (r \rightarrow s), r \wedge p \vdash s$ (8)
7. $p \wedge \neg q, \neg q \wedge r, (p \wedge r) \rightarrow s \vdash p \wedge s$ (8)
8. $q, \neg\neg\neg p \leftrightarrow (q \wedge \neg r), \neg\neg\neg r \vdash \neg p \vee (s \rightarrow t)$ (8)
9. $p, q, ((r \wedge p) \wedge q \leftrightarrow \neg\neg s, r \vdash s$ (8)
10. $(p \wedge r) \rightarrow s, p \wedge q, p \leftrightarrow r \vdash s \vee t$ (8)

Hard

11. $(p \wedge r) \rightarrow s, p \wedge (q \vee t), r \leftrightarrow p \vdash (s \wedge r) \vee t$ (9)
12. $p \rightarrow (q \rightarrow (r \rightarrow s)), r \wedge (p \wedge q) \vdash s$ (9)
13. $((p \vee q) \leftrightarrow (r \rightarrow s)) \wedge \neg\neg(r \wedge q) \vdash s$ (9)
14. $(p \wedge \neg q) \wedge r, p \rightarrow s, r \rightarrow t \vdash s \wedge t$ (9)
15. $(p \vee q) \rightarrow r, (p \vee s) \rightarrow t, p \wedge u \vdash r \wedge t$ (9)

4.2 Subproofs and the More Complex Inference Rules: $\rightarrow$Intro and $\leftrightarrow$Intro

In the previous section, we presented what we called the "basic" inference rules of $\mathcal{F}$. The remaining four inference rules that constitute $\mathcal{F}$ are more complex. Their complexity derives from the fact that they involve temporarily introducing

an assumption or hypothesis into one's derivation, typically using that assumption to derive further sentences, and ultimately discharging that assumption once the complex rule in question is employed. Because the assumption is a *temporary* assumption, only in force until it is discharged upon the application of the complex rule associated with its introduction to the derivation, we use a subproof, which is a proof or derivation inside our main proof. The key idea behind the subproof structure is to contain the influence of the temporary assumption by marking the point at which that assumption is introduced and then drawing a line at which the assumption and all sentences potentially tainted by that assumption can no longer be used. Once a subproof is closed, lines that occur within that subproof can no longer be cited to justify later lines. This containment is necessary as there is otherwise a threat that the temporary assumption will be used to justify the ultimate conclusion, which would mean that there would be no way to tell whether or not the conclusion was really being derived from just the initial premises, a threat which would in turn call into question the very idea that deriving a sentence A from a set of sentences $\{B_1, \ldots, B_n\}$ proves that the argument with $\{B_1, \ldots, B_n\}$ as premises and A as conclusion is valid. Suppose, then, that we had an alleged derivation of A from $\{B_1, \ldots, B_n\}$, but that the derivation involved making an additional assumption, which was then used to justify the ultimate conclusion A. Then the derivation could hardly be taken to demonstrate or establish logical connections between the initial premises $\{B_1, \ldots, B_n\}$ and the conclusion A. The subproof structure ensures, then, that the derivation does establish a logical connection between the initial premises and the ultimate conclusion by ensuring that the additional, temporary assumptions introduced in the derivation are just that: Temporary.

To illustrate subproofs, consider the following proof structure.

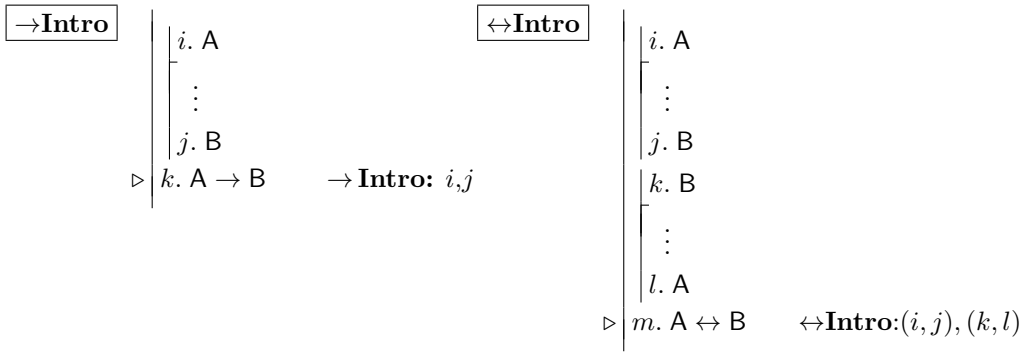
1. B_1	**Hyp**
2. B_2	**Hyp**
3. C	**justification**
4. D	**Hyp** (temporary)
5. E	**justification**
6. F	**complex rule**
7. A	**justification**

In this derivation, B_1 and B_2 are the premises and A is the conclusion. At line 4, a subproof is introduced, where D is assumed true. That assumption is then used to derive E at line 5 (assuming the justification for line 5 involve reference to line 4). The assumption that D is true is then discharged at line 6, once the complex rule associated with the introduction of the subproof at line 4 is used. Later lines of the derivation, like line 7, then, cannot make reference to lines 4–5, as the entire subproof environment is tainted by an assumption that D is true, which does not necessarily follow from the initial premises B_1 and B_2. We should think, then, of the subproof lines 4–5 as being boxed in and lines later in the derivation, after that subproof has been completed and the

complex inference rule in conjunction with which it was introduced has been used, cannot cite lines inside that subproof box.

We will make these ideas more concrete by discussing two complex rules in detail: Namely, →**Intro** and ↔**Intro**, which share a common structure.

Conditional Inference Rules for $\mathcal{F}$



Here's the intuitive idea behind →**Intro**. One is entitled to write a conditional sentence of the form $\ulcorner(A \to B)\urcorner$ as a line of one's derivation at line k provided one has shown earlier in one's derivation, above line k, that B can be derived from A. (In that case, one will have shown, in effect, that, if A is true, then B is true as well, which is what is required to ensure the truth of $\ulcorner(A \to B)\urcorner$, as one has shown, in that case, that, given the truth of A, the antecedent of the conditional, B is true, too.) One shows that B can be derived from A by introducing a subproof in which one assumes, for the purposes of establishing the conditional $\ulcorner(A \to B)\urcorner$, A as a line of one's proof and then deriving B.

Let's look at a concrete example of →**Intro** in action. Consider the following derivation, establishing $\{r, (p \to q)\} \vdash (r \wedge (p \to (q \vee s)))$.

Derivation establishing that $\{r, (p \to q)\} \vdash (r \wedge (p \to (q \vee s)))$

1. r	**Hyp**
2. (p → q)	**Hyp**
3. p	**Hyp** (for →**Intro**)
4. q	→**Elim**:2,3
5. (q ∨ s)	∨**Intro**:4
6. (p → (q ∨ s))	→**Intro**:3,5
7. (r ∧ (p → (q ∨ s)))	∧**Intro**:1,6

→**Intro** is used on line 6, justifying the conditional sentence '$(p \to (q \vee s))$'. Use of →**Intro** always requires a subproof immediately above the line where the rule is used, where the antecedent of the conditional being derived (in the case of this derivation, 'p') is assumed at the start of the subproof (line 3 in the derivation above) and the consequent (in our case, '$(q \vee s)$') is derived at the end of the subproof (line 5). The subproof is then "sealed" off at its close. This

means that after line 6, where →**Intro** is used and the assumption introduced at line 3 is "discharged," none of the lines in the subproof, i.e., none of the lines 3-5, can be cited again. So, in particular, it would be illegal to cite lines 3–5 at line 7 (or any later line that might be added after line 7). This sealing of the lines inside the subproof ensure, then, that the assumption made at the start of the subproof is a temporary assumption and so the ultimate conclusion does not depend upon that assumption.

This "sealing off" of closed subproofs is crucial to ensuring that our inference rules preserve truth. To see why, let's suppose that we were to allow the lines in a subproof to be cited after the subproof is sealed and after the complex inference rule that introduced the subproof is employed. It would then be possible to carry out derivations where the last line of the derivation is not a logical consequence of the original premises of the derivation and so our proof theory would not preserve truth. As an example, we know that $(p \rightarrow q) \not\models (p \vee q)$, as the following row of the joint truth table of those sentences, where '$(p \rightarrow q)$' is true while '$(p \vee q)$' is false, proves.

	p	q	r	$(p$	$\rightarrow$	$q)$	$(q$	$\vee$	$r)$
$\Rightarrow$	F	F	F	F	T	F	F	F	F

Now consider the following (FAULTY!) derivation. (Note: The presence of the asterisks in the justification of line 6 indicate that the use of the rule is illegal.)

1. $(\mathsf{p} \rightarrow \mathsf{q})$	**Hyp**
2. p	**Hyp** (for →**Intro**)
3. q	→**Elim**:1,2
4. $(\mathsf{q} \vee \mathsf{r})$	∨**Intro**:3
5. $(\mathsf{p} \rightarrow (\mathsf{q} \vee \mathsf{r}))$	→**Intro**:2,4
6. $(\mathsf{q} \vee \mathsf{r})$	***→**Elim**:5,2***

The last line of the derivation is illegal, as line 2 occur inside a sealed subproof and so is not available for citation in justifying line 6. If we allowed the lines in a subproof to always be available for citation, however, then the above derivation would be legal and our proof theory would not be a reliable guide to the validity of an argument as our inference rules would not preserve truth, as it would allow us to derive '$(p \vee q)$' from '$(p \rightarrow q)$' even though $(p \rightarrow q) \not\models (p \vee q)$. That would be disastrous. Avoiding this result motivates the complexities associated with the subproof structures. (Note: It is a much more difficult thing to prove that these complexities satisfy those motives and prevent all undesirable effects of lingering temporary assumptions. We won't try to prove here that they do but will instead simply assert that they are sufficient. The proof is the content of more advanced courses on metalogic, where the soundness and completeness of a system like $\mathcal{F}$ is established.)

→**Intro** tells us that we are entitled to add a conditional of the form $\ulcorner(A \rightarrow B)\urcorner$ as a later line of our derivation provided we can derive the consequent B

from the temporary assumption that the antecedent A is true. Recall from our discussion of the truth definition of '$\leftrightarrow$' in the previous chapter that a bi-conditional of the form $\ulcorner(A \leftrightarrow B)\urcorner$ is equivalent to the conjunction of both directions of the corresponding conditional sentences (i.e., $\ulcorner((A \rightarrow B) \wedge (B \rightarrow A))\urcorner$). So, it makes sense that $\leftrightarrow$**Intro** is set up the way it is, requiring, in effect, that we derive both directions of the corresponding conditionals, i.e., derive both $\ulcorner(A \rightarrow B)\urcorner$ and $\ulcorner(B \rightarrow A)\urcorner$, in order to establish the bi-conditional $\ulcorner(A \leftrightarrow B)\urcorner$. Let's give a very simple case to illustrate the functioning of this rule. Consider, then, the following derivation, establishing $\{q, ((p \wedge q) \leftrightarrow r)\} \vdash (r \leftrightarrow p)$.

Derivation establishing that $\{q, ((p \wedge q) \leftrightarrow r)\} \vdash (r \leftrightarrow p)$

1. q	**Hyp**
2. ((p ∧ q) ↔ r)	**Hyp**
3. r	**Hyp** (for ↔**Intro**)
4. (p ∧ q)	↔**Elim**:2,3
5. p	∧**Elim**:4
6. p	**Hyp** (for ↔**Intro**)
7. (p ∧ q)	∧**Intro**:1,6
8. r	↔**Elim**:2,7
9. (r ↔ p)	↔**Intro**:(3,5),(6,8)

Notice that the two subproofs in the above derivation, the subproof from line 3 to 5 and the subproof from 6 to 8, are "sealed off" from one another. So, for example, it would be illegal to cite line 4 after line 5, when the first subproof is sealed. In the first subproof, we assume the left side of the bi-conditional '$(r \leftrightarrow p)$' and derive its right side. (By assuming 'r', we can use $\leftrightarrow$**Elim** on line 2 to derive '$(p \wedge q)$', from which we can then derive 'p', our goal of the first subproof, by $\wedge$**Elim**.) In the second subproof, we assume the right side of the bi-conditional '$(r \leftrightarrow p)$' and derive its left side. (By assuming 'p', we can use $\wedge$**Intro** on line 1 to derive '$(p \wedge q)$', from which we can then derive, in concert with line 2, 'r', our goal of the second subproof, by $\leftrightarrow$**Elim**.) Notice that the first subproof entitles us to derive '$(r \rightarrow p)$' and the second subproof entitles us to derive '$(p \rightarrow r)$', thus illustrating the fact that $\leftrightarrow$**Intro** requires us to derive both directions of the desired bi-conditional being justified.

There are no limitations on the subproof structures that one could introduce into one's derivation. As long as one does not violate the seal on a subproof, only citing lines within the subproof in conjunction with one of the four complex inference rules, one can introduce absolutely any assumption one likes into one's derivation, deriving further sentences from that introduced assumption inside the subproof, without corrupting one's derivation. However, there is only a point to introducing a new temporary hypothesis into one's derivation when one plans to use that assumption and its associated subproof structure in union with a complex inference rule like $\rightarrow$**Intro**, as the complex inference rules provide the only way to legally use any of the lines inside the subproof to justify

sentences outside the subproof and so be of any use in justifying the ultimate conclusion of the derivation. So, it is never a good idea to introduce an assumption just to "see where it goes." Such random and unguided introductions of assumptions will not help one achieve one's ultimate goal of deriving the conclusion from one's premises. Instead, one should *only* introduce a temporary assumption and a subproof into one's derivation in union with the use of a complex inference rule, in which case there are *always* specific controls both on the temporary assumption one should make and the goal of the subproof. For example, whenever one plans to use →**Intro** to justify a sentence, one assumes the antecedent of that conditional and derives as the last line of that subproof that conditional's consequent. So, in our example at the beginning of this section, we planned to use →**Intro** to justify the sentence '$(p \rightarrow (q \vee s))$'. So, we introduced a subproof immediate above the line containing that sentence, assumed the antecedent of that conditional, 'p', as the first line of that subproof, and attempted to derive that conditional's consequent, '$(q \vee s)$', as the last line of the subproof. Once the consequent is derived and the subproof cited to justify the conditional '$(p \rightarrow (q \vee s))$', the subproof is sealed and we no longer cite any of the lines in that subproof.

We said in the above paragraph that one should *only* introduce a temporary assumption and a subproof structure into one's derivation in conjunction with the use of a complex inference rule like →**Intro**. This is why the LogicSpaces Proof-Builder application does not allow you to add a temporary hypothesis or subproof directly. Instead, a subproof is created automatically and only when a complex inference rule is first selected as the justification for a line of one's derivation and is removed when that justification is changed or that line deleted. This behavior is intended to enforce the idea that the only times one should be introducing temporary hypotheses and subproofs into one's derivation is in concert with the specific use of a complex inference rules like →**Intro**. This control is, we think, very useful when learning how to construct derivations. However, you should be reflective about the limitations the Proof-Builder application is enforcing so that when you are set free, as you are, for example, when you construct derivations on blank pieces of paper all on your own, you can effect the same controls on yourself.

Consider now the fact that $\models (p \rightarrow (p \rightarrow p))$, where this is a claim of the form $\Gamma \models B$, where Γ is the empty set, which we can prove with the following truth table.

p	$(p$	$\rightarrow$	$(p$	$\rightarrow$	p	$))$
T	T	T	T	T	T	
F	F	T	F	T	F	

The sentence is true under every interpretation and so is a logical truth. Thus, regardless of what sentence, if any, we begin with, we are guaranteed that we will not be moving from truth to falsity to infer $(p \rightarrow (p \rightarrow p))$. So, recalling our earlier discussion of the connections between logical consequence and provability, we know that $\vdash (p \rightarrow (p \rightarrow p))$. The following derivation establishes this last

claim.

Derivation establishing that $\vdash (p \rightarrow (p \rightarrow p))$

0.		
1. p		**Hyp** (for →**Intro**)
2. p		**Hyp** (for →**Intro**)
	[goal : p]	
3. (p → p)		→**Intro**:2,2
4. (p → (p → p))		→**Intro**:1,3

This derivation illustrates several points that are worth highlighting.

The first point to notice is that the derivation does not contain initial premises. So, the initial Fitch bar does not have any sentences above it; line 0 is empty. The derivation is thus an example of a *premiseless derivation*. Every (and only) logical truth has a legal premiseless derivation, which makes sense when one recalls the connections between the notions of logical truth and logical consequence (namely, that every logical truth is a logical consequence of every set of sentences, including the empty set) and the semantic notion of logical consequence and the proof-theoretic notion of provability (namely, that there is a corresponding true claim of provability for every true claim of logical consequence) that we mentioned in the previous paragraph. Note as well: Every premiseless derivation of $\mathcal{F}$ requires a use of one of our complex inference rules (in particular, either →**Intro**, ↔**Intro**, or ¬**Intro**), as every derivation requires some initial hypoheses, even if they are only temporary assumptions introduced in a subproof, to get the derivation ball rolling.

The second point the above derivation illustrates is that subproofs can embed inside other subproofs. We have already seen, in our discussion of ↔**Intro**, that a single derivation might contain multiple subproofs. But those earlier examples did not involve embedded multiple subproofs. The two subproofs needed to employ ↔**Intro** in a derivation involve a pair of subproofs that occur one after the other. In the example above, the subproof beginning and ending on line 2 occurs *within* the subproof beginning on line 1 and ending on line 3. This is perfectly acceptable and even common. Indeed, with more complex derivations, you will embed several subproofs within one another. In these cases, it is very important that you are clear about when the lines in a subproof are "sealed off" and when they are available for citation. This is aided by the Fitch format, as you can see, for example, that the subproof on line 2 is closed by line 3, while the subproof on line 1 continues to line 3. (If you want to create an even more vivid visual aid on paper, you can draw a rectangle around each subproof, the length of the Fitch line, and then remember that you cannot cross back into a sealed rectangle. If you do that with the derivation above, you will draw one rectangle, around line 2 and the empty space for derived lines, inside another rectangle, from line 1 to 3.) Keeping track of the duration that a temporary assumption is in play will help you avoid the temptation to refer to a line inside

a subproof from outside that subproof, after it has been "sealed off" and the temporary assumption has been discharged.

The third and final point illustrated by this derivation is what we can call a "degenerate" use of a rule. Notice that the citation for line 3 contains two references to line 2. Every legal use of →**Intro** involves citing a line that contains the assumed antecedent of the conditional being derived and a line containing the consequent of that conditional. The conditional on line 4 has the same sentence, 'p', as both its antecedent and consequent. So, we can cite the same line for both of these roles, although in that case, one must cite the line twice, indicating that it contains both the assumed antecedent and the consequent. (In the Proof-Builder application on LogicSpaces, one cites the same line twice by clicking that line twice with the citation menu open.) Because the goal of that embedded subproof, namely, the consequent 'p', has already been reached on line 2, there is no need to bring that sentence down inside the subproof, as one already has it as an earlier line available for citation. (Indeed, the easiest way to bring it down inside the subproof, given the nonexistence of a reiteration rule that allows you to copy an earlier line as a later line in $\mathcal{F}$, is to first use ∧**Intro** to justify a line with the sentence on the earlier line as a conjunct and use ∧**Elim** to then rewrite the original sentence beneath the horizontal Fitch bar, thus adding two unnecessary lines to your derivation.)

The above derivation illustrates one kind of "degenerate" (but perfectly legal!) use of a rule like →**Intro**, where one does not need to derive the consequent from the assumption of the antecedent. As an example of another kind of degenerate use, consider the following derivation establishing $q \vdash (p \rightarrow (r \rightarrow q))$.

Derivation establishing that $q \vdash (p \rightarrow (r \rightarrow q))$

1. q	**Hyp**
2. p	**Hyp** (for →**Intro**)
3. r	**Hyp** (for →**Intro**)
[goal : q]	
4. (r → q)	→**Intro**:3,1
5. (p → (r → q))	→**Intro**:2,4

The goal of the second, embedded subproof is to derive 'q', which is the consequent of the conditional on line 5 to be justified by the use of →**Intro** associated with the subproof beginning on line 3. But 'q' already occurs as an earlier line that is available (i.e., on a line that is not inside the subproof that is sealed at line 4) on line 1. So, we can simply cite line 1 directly on line 5; there is no need to (re)derive 'q' inside the embedded subproof.

→Intro and ↔Intro Derivation Problems

Each of the following claims of provability can be established by constructing a derivation that only uses only the seven Basic Inference rules and →**Intro**, where

each of the derivations turns on the use of →**Intro**. The problems are dividing in to difficulty crudely based on both the shortest length derivation that establishes the claim and the number of embedded subproofs that derivation requires. Each problem is followed by a pair of numbers (x, y). x represents the number of lines in the optimal derivation establishing the claim of provability and y the number of depth of embedded subproofs, where 1 means that the subproofs go to a depth of 1 and so there are no embedded subproofs, 2 that the subproofs go to a depth of 2 and so there is one subproof embedded inside another, and so on. Note: A derivation with a subproof depth of 1 may require more than one subproof, although none of those subproofs will be embedded within the scope of another. Depth measures embeddedness, not just the straight number of subproofs.

Easy

1. $q \vdash \neg p \rightarrow q$ (3,1)
2. $p \vdash q \rightarrow (p \wedge q)$ (4,1)
3. $p \rightarrow r \vdash (p \wedge q) \rightarrow r$ (5,1)
4. $(p \vee q) \rightarrow r \vdash p \rightarrow r$ (5,1)
5. $p \rightarrow r \vdash p \rightarrow (q \vee r)$ (5,1)
6. $p \rightarrow q \vdash (p \wedge r) \rightarrow q$ (5,1)

Medium Easy

7. $p \rightarrow r, r \rightarrow q \vdash p \rightarrow q$ (6,1)
8. $r \rightarrow (s \rightarrow t), s \vdash r \rightarrow t$ (6,1)
9. $p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$ (6,1)
10. $p \rightarrow \neg q, r \rightarrow p \vdash r \rightarrow \neg q$ (6,1)
11. $p \rightarrow q, q \rightarrow r \vdash p \rightarrow (q \wedge r)$ (7,1)
12. $p \rightarrow \neg q, \neg q \rightarrow r \vdash (p \rightarrow r) \vee (s \rightarrow t)$ (7,1)
13. $p \rightarrow q \vdash (p \wedge r) \rightarrow (q \wedge r)$ (7,1)

Medium

14. $(p \rightarrow q) \rightarrow r, \neg s \rightarrow q, p \rightarrow \neg s \vdash r$ (8,1)
15. $(p \rightarrow q) \rightarrow r, \neg s \rightarrow q, p \rightarrow \neg s \vdash r$ (8,1)
16. $p \rightarrow q, r \rightarrow p, (r \rightarrow q) \rightarrow s \vdash s$ (8,1)
17. $(p \rightarrow q) \rightarrow (r \rightarrow s), r \wedge (q \wedge p) \vdash s$ (9,1)
18. $p \rightarrow (q \wedge r) \vdash (p \rightarrow q) \wedge (p \rightarrow r)$ (10,1)
19. $s \rightarrow p, s \leftrightarrow q \vdash (r \wedge q) \rightarrow ((r \vee r) \wedge p)$ (10,1)

20. $p \to (q \wedge r), s \wedge q \vdash p \to ((q \wedge (s \wedge r)) \vee t)$ (11,1)

    Medium Hard

21. $\vdash (p \wedge p) \to (q \to p)$ (5,2)

22. $(p \wedge q) \to r \vdash p \to (q \to r)$ (7,2)

23. $\neg p \to \neg q \vdash q \to p$ (7,2)

24. $p \to (q \to r) \vdash (p \to q) \to (p \to r)$ (8,2)

25. $p \to q, q \to (r \to s) \vdash r \to (p \to s)$ (9,2)

26. $((p \wedge q) \wedge r) \to s \vdash p \to ((r \wedge q) \to s)$ (10,2)

    Hard

27. $p \to (q \to r), (q \to r) \to s, (p \to s) \to (t \to p) \vdash t \to p$ (11,2)

28. $(p \to q) \to (p \to r), p \to s, s \to q \vdash p \to r$ (11,2)

29. $((p \wedge q) \wedge r) \to s \vdash r \to (p \to (q \to s))$ (10,3)

30. $p \to (q \to (r \to s)) \vdash r \to (p \to (q \to s))$ (10,3)

31. $p \to (\neg q \to \neg r), q \to (s \wedge t) \vdash p \to (r \to s)$ (13,3)

    Use the Basic Inference rules, $\to$**Intro**, and $\leftrightarrow$**Intro** to construct derivations establishing the following claims of provability, where each derivations centers on the use of $\leftrightarrow$**Intro**.

32. $p \leftrightarrow q, q \leftrightarrow r \vdash p \leftrightarrow r$ (9,1)

33. $s \wedge \neg\neg\neg\neg q, p \leftrightarrow \neg r \vdash p \leftrightarrow (q \wedge \neg r)$ (12,1)

34. $(p \leftrightarrow q) \leftrightarrow r \vdash p \to (q \leftrightarrow r)$ (14,3)

35. $\vdash (p \to (q \to r)) \leftrightarrow ((p \wedge q) \to r)$ (15,3)

36. $(p \leftrightarrow q) \leftrightarrow r \vdash p \leftrightarrow (q \leftrightarrow r)$ (30,4) **challenge

4.3 Subproofs and the More Complex Inference Rules: ¬Intro

There are two remaining inference rules to discuss before the full system $\mathcal{F}$ is in place: ¬**Intro** and ∨**Elim**. If we rank the inference rules of $\mathcal{F}$ on a scale of complexity, these two rules are certainly on the far reaches of that scale, with the later being the most difficult. Using them correctly requires both memorizing their forms, but also understanding why they are set up as they are and the informal reasoning they are based on, the understanding of which will help guide you in the formal use of the rule itself. So, before you begin to use the rules

in your derivations, be sure you understand the form of reasoning those rules encode, as that will guide you in using the rules correctly.

We begin with ¬**Intro**, the abstract form of which is as follows.

¬**Intro**	i. A	
	⋮	
	j. B	
	k. ¬B	
▷	l. ¬A	¬**Intro**:i, (j, k)

This inference rule is the formal equivalent to the form of argumentation we called *reductio ad absurdum* in Chapter 1 and that you used in your semantic proof form chapter 3.9. The idea behind the rule is that we are justified in the negation of a sentence A, i.e., the sentence $\lceil \neg A \rceil$, provided we can derive a contradiction, in the form of both a sentence and its negation, from the assumption that sentence A is true. The key idea behind this line of argumentation (and hence the inference rule itself) is that, because contradictions are never true, anything that entails a contradiction is false. So, if we can show that a contradiction follows from the assumption that a sentence A is true, then we are entitled to the claim that A is false and so that its negation, $\lceil \neg A \rceil$, is true. (This is, strictly speaking, too strong. What we are instead entitled to is that we will not be moving from truth to falsity, and hence our inference will preserve truth, in the sense outlined earlier, if we infer $\lceil \neg A \rceil$ under these conditions. We won't dwell on the significance of the difference, but it is significant.) This is an extremely powerful form of argumentation. It is, together with the principle of bivalence or the Law of Excluded Middle (i.e., the principle that every instance of $\lceil (A \vee \neg A) \rceil$ is logically true), and the validity of the ¬**Elim** inference rule, a hallmark of classical logic. A comparison of classical and nonclassical logics, including logics that allow for true contradictions, so-called *paraconsistent logics*, and logics that countenance more than two truth values, so-called *many-valued logics*, is beyond the scope of this textbook, which is unapologetically a presentation of a classical propositional logic.

Let's begin with a very simple derivation employing ¬**Intro** that establishes $\vdash \neg(p \wedge \neg p)$. (The conclusion is an instance of what is called the *Law of Non-contradiction*, every instance being logically true in a classical logic.)

Derivation establishing that $\vdash \neg(p \wedge \neg p)$

0.		
1.	(p ∧ ¬p)	**Hyp** (for ¬**Intro**)
2.	p	∧**Elim**:1
3.	¬p	∧**Elim**:1
4.	¬(p ∧ ¬p)	¬**Intro**:1,(2,3)

The subproof from lines 1 to 3 involves assuming, on line 1, that the *unnegated form* of the sentence on line 4 to be justified using ¬**Intro** is true. That as-

sumption is then reduced to an absurdity by deriving a contradictory pair of sentences (the sentence 'p' on line 2 and its negation, the sentence '$\neg p$', on line 3) from that assumption. Because we have derived a contradiction from the assumption of the unnegated form of the sentence on line 4, i.e., the assumption on line 1, we are entitled to the falsity of that assumption and so the truth of its negation.

Consider now the following derivation establishing $\{\neg q, (p \rightarrow q)\} \vdash \neg p$, corresponding to an inference rule of some systems (not $\mathcal{F}$!) called *Modus Tollens*.

Derivation establishing that $\{\neg q, (p \rightarrow q)\} \vdash \neg p$

1. $\neg$q	**Hyp**
2. (p $\rightarrow$ q)	**Hyp**
3. p	**Hyp** (for $\rightarrow$**Intro**)
4. q	$\rightarrow$**Elim**:2,3
5. $\neg$p	$\neg$**Intro**:2,(1,4)

In this derivation, notice that only one member of the pair of contradictory sentences ('q', on line 4) occurs within the subproof itself. The other member of the pair occurs earlier in the derivation but still accessible, because not "sealed off," from both the subproof itself and the line where the complex rule $\neg$**Intro** is used, on line 5. This is perfectly legal. Notice also that, because this derivation exists, we have proven that there is no need to include Modus Tollens as a basic form of inference in our proof theory; it is instead an example of a (very useful indeed) *derivative* inference rule to be discussed in the last section of this chapter.

As already remarked, $\neg$**Intro** is an extremely useful inference rule. When other strategies for constructing one's derivation fail, one should suspect that $\neg$**Intro** is needed. Using $\neg$**Intro** will *never* lead one astray, although it is often redundant, as using it to justify a sentence may require deriving within the associated *reductio* subproof the very sentence one is trying to establish in order derive the needed contradiction, thus adding several steps. So, it is a good idea to look to use $\neg$**Intro** when one hits a roadblock in one's attempt to construct a derivation.

$\neg$**Intro** can *only* be used to immediately justify a negation. That is, one cannot use the rule to justify a sentence whose main operator is not a negation. However, one can always *indirectly* use the rule to derive such a sentence. Learning this technique is incredibly important to successfully constructing the derivations that you will be assigned.

Suppose that one wants to derive a positive sentence A, whose main operator is *not* '$\neg$'. One cannot directly justify A using $\neg$**Intro**, where A's leftmost symbol is not '$\neg$' (i.e., where A is not a negation), as one is not *introducing* a negation if '$\neg$' isn't the main operator of the sentence being justified. In such cases, one much instead employ $\neg$**Intro** *indirectly* by justifying the double negated form of A, i.e., the sentence $\ulcorner\neg\neg A\urcorner$, directly by $\neg$**Intro**, by assuming

for *reductio* $\lceil \neg A \rceil$, deriving (or citing) a contradiction, using ¬**Intro** to justify $\lceil \neg\neg A \rceil$, and then using ¬**Elim** to justify A, one's initial goal, as the following derivation structure illustrates.

	i. ¬A	
	⋮	
	j. B	
	k. ¬B	
l. ¬¬A		¬**Intro**:$i, (j, k)$
m A		¬**Elim**:l

While requiring an extra step, it is far more elegant, we think, to formulate a single, unvarying form for the inference rule ¬**Intro** that always only justifies negations, rather than positing an exception in the form of the rule itself to allow for a positive sentence to be directly justified using ¬**Intro** when that sentence's negation leads to a contradiction. (It is perfectly possible to construct a perfectly adequate proof theory that allows this form of inference, thus allowing what is in the above schema line m to be justified directly by ¬**Intro** and deleting line l. It is a preference for elegantly, streamlined inference rules over shorter derivations that leads us to formulate $\mathcal{F}$ the way we have.)

We will prove $\vdash (p \vee \neg p)$ in order to illustrate the above described indirect use of ¬**Intro**. '$(p \vee \neg p)$' is an instance of the *Law of Excluded Middle*, every instance of which is logically true in a classical logic. This derivation is extremely useful to study and master, as there are many more complex derivations to be assigned that are variants of its basic structure. Your time spent with this derivation, and learning to reproduce it on your own without notes, understanding how the derivation works, is worth the investment.

Derivation establishing that $\vdash (p \vee \neg p)$

0.		
1.	$\neg(\mathsf{p} \vee \neg\mathsf{p})$	**Hyp** (for ¬**Intro**)
2.	p	**Hyp** (for ¬**Intro**)
3.	$(\mathsf{p} \vee \neg\mathsf{p})$	∨**Intro**:2
4.	$\neg\mathsf{p}$	¬**Intro**:2,(1,3)
5.	$(\mathsf{p} \vee \neg\mathsf{p})$	∨**Intro**:4
6.	$\neg\neg(\mathsf{p} \vee \neg\mathsf{p})$	¬**Intro**:1,(1,5)
7.	$(\mathsf{p} \vee \neg\mathsf{p})$	¬**Elim**:6

The reasoning behind the above derivation can be described as follows. We assume, for the purposes of *reductio*, that the instance of the law of excluded middle, '$(p \vee \neg p)$', is false and so its negation is true. We derive from that assumption an absurdity, i.e., a sentence and its negation, the contradiction being the original assumption itself and the unnegated form of that assumption, which is '$(p \vee \neg p)$', the very sentence we are aiming to derive. We justify the sentence, within the scope of the primary ¬**Intro** subproof, by first assuming that the left disjunct of the disjunction '$(p \vee \neg p)$', i.e., the sentence 'p', is true, from which we derive the disjunction '$(p \vee \neg p)$' on line 3, producing a contradiction with the original assumption and allowing us to derive '$\neg p$' by ¬**Intro** on line 4. As '$\neg p$' is the right disjunct of that same disjunction '$(p \vee \neg p)$', we can again derive that disjunction by ∨**Intro**, on line 5. We now again have a contradiction with the original assumption on line 1. Now, because that original assumption takes the form of assuming that the negation of '$(p \vee \neg p)$' is true, what we can directly derive on line 6 is the *double* negation '$\neg\neg(p \vee \neg p)$' and then, by an application of ¬**Elim** our ultimate desired conclusion '$(p \vee \neg p)$' on line 7. One important thing to notice about this derivation: One of the members of the contrary pair involved in the use of ¬**Intro** on line 6 is the very initial assumption at the beginning of the associated subproof beginning on line 1. That is why we cite line 1 twice in the justification for line 6. The first citation to line 1 is as the initial assumption of the unnegated form of the sentence being justified on line 6 and the second citation to to line 1 is as one of the members of the contradictory pairs that reduce that assumption to an absurdity.

A successful use of ¬**Intro** involves deriving or citing a contrary pair of sentences; the use of the rule is alway guided, then, by the aim finding a contradiction. However, suppose that one already has, earlier in one's derivation, a pair of sentences of the form B and $\ulcorner\neg B\urcorner$. One is then in a position to derive *any* sentence C whatsoever by the following abstract inference schema.

$i.$ B	
$j.$ ¬B	
$\quad k.$ ¬C	
$\quad\vdash$	
$l.$ ¬¬C	¬**Intro**:$k,(i,j)$
$m.$ C	¬**Elim**:l

Use this pattern whenever you notice that you have contrary sentences on earlier, available lines of your derivation, as you can use it to quickly derive any sentence whatsoever. Furthermore, if you notice that the earlier available lines are contradictory, even if the contradiction is not overt in the form of two contrary pairs of sentences on separate lines, then, once you explicitly draw the contradiction out, you can again use the above form of inference to derive any sentence whatsoever. (This power of the ¬**Intro** rule, in combination with the ¬**Elim** rule, is again one of the hallmarks of classical logic, as it shows that countenancing contradictions leads to trivialization, in which every sentence whatsoever is provable.)

Here is a concrete example that illustrates the proof structure suggested in the previous paragraph that establishes $(p \wedge q), (q \rightarrow \neg p) \vdash r$.

Derivation establishing that $(p \wedge q), (q \rightarrow \neg p) \vdash r$

1. (p ∧ q)	**Hyp**
2. (q → ¬p)	**Hyp**
⊢	
3. p	∧**Elim**:1
4. q	∧**Elim**:1
5. ¬p	→**Elim**:2,4
$\quad$6. ¬r	**Hyp** (for ¬**Intro**)
$\quad\vdash$	
7. ¬¬r	¬**Intro**:6,(3,5)
8. r	¬**Elim**:7

Notice that lines 3–5 explicitly draw out the contradiction implicit in the original premises on lines 1 and 2. Lines 6-8 then involve the use of ¬**Intro**. Notice that the temporary assumption on line 6 is not used in the derivation of the contradiction, because the contradiction follows from the premises themselves. Notice also that the sentence on the last line of the derivation, the ultimate conclusion of the derivation, is a simple atomic sentence-letter that does not occur anywhere in the original premises. This may seem surprising.

You can convince yourself that 'r' should in fact be provable from the contradictory '$(p \wedge q)$' and '$(q \rightarrow \neg p)$' by reflecting on the fact that $\{(p \wedge q), (q \rightarrow \neg p)\} \models r$, the proof of which you are, after chapter 3, in a position to give yourself. Our proof theory would be incomplete, then, if it weren't the case that $\{(p \wedge q), (q \rightarrow \neg p)\} \vdash r$. Furthermore, the above derivation serves as a pattern for proving that *any* sentence whatsoever of the form A, replacing all of

the occurrences of 'r' in the above derivation with A, is provable from premises 1 and 2. This too is as it should be, as any sentence whatsoever is a logical consequence of the inconsistent set $\{(p \wedge q), (q \rightarrow \neg p)\}$.

¬Intro Derivation Problems

Each of the following claims of provability can be established by constructing a derivation that only uses only the seven Basic Inference rules, →**Intro**, ↔**Intro**, and ¬**Intro**, where the derivation centers on the use of the last rule. That is, all of the inference rules *except* ∨**Elim**.

Easy

1. $p \rightarrow (q \wedge r), \neg(q \wedge r) \vdash \neg p$ (5,1)
2. $p \wedge q \vdash \neg(p \rightarrow \neg q)$ (6,1)
3. $\neg\neg q \rightarrow p, \neg p \vdash \neg q$ (6,1)
4. $\neg(p \vee q), r \rightarrow p \vdash \neg r$ (6,1)
5. $\neg p \rightarrow \neg q, q \vdash p$ (6,1)
6. $\neg\neg q \rightarrow p, \neg p \vdash \neg q$ (6,1)
7. $q \rightarrow (p \rightarrow r), \neg r, q \vdash \neg p$ (7,1)
8. $p \rightarrow q, q \rightarrow r, \neg r \vdash \neg p$ (7,1)
9. $p \wedge q, r \rightarrow \neg q \vdash \neg r$ (7,1)
10. $p \rightarrow (q \rightarrow r), p, \neg r \vdash \neg q$ (7,1)
11. $p, \neg q \wedge r \vdash \neg(q \wedge r) \wedge p$ (7,1)

Medium Easy

12. $(p \vee q) \rightarrow \neg r, s \rightarrow r, p \vdash \neg s$ (8,1)
13. $(p \vee q) \rightarrow \neg r, s \rightarrow r, p \vdash \neg s$ (8,1)
14. $q \rightarrow (p \rightarrow r), \neg r \wedge q \vdash \neg p$ (8,1)
15. $\neg\neg r \rightarrow (\neg p \rightarrow q), \neg r \rightarrow p, \neg p \vdash q$ (8,1)
16. $p \rightarrow q, q \rightarrow (r \wedge s), s \rightarrow \neg r \vdash \neg p$ (10,1)
17. $p \wedge \neg s, q \rightarrow s, (p \wedge \neg q) \rightarrow t \vdash t$ (10,1)
18. $p \rightarrow q, \neg q, \neg p \rightarrow (r \wedge s) \vdash r \wedge s$ (7,1)
19. $p \rightarrow q, p \rightarrow (r \wedge s), (s \wedge r) \rightarrow t, \neg t \vdash \neg p$ (11,1)

Medium

20. $p \to q \vdash \neg q \to \neg p$ (6,2)
21. $p \to \neg q \vdash q \to \neg p$ (6,2)
22. $\neg p \to q \vdash \neg q \to p$ (7,2)
23. $\vdash p \vee \neg p$ (7,2)
24. $\neg p \to q \vdash \neg q \to p$ (7,2)
25. $p \to (q \wedge r) \vdash \neg q \to \neg(p \wedge r)$ (8,2)
26. $\neg p \to \neg q, (s \to q) \to q \vdash \neg p \to \neg(s \to q)$ (8,2)
27. $p \to (q \wedge r) \vdash \neg q \to \neg(p \wedge r)$ (8,2)
28. $\vdash p \vee \neg(p \wedge q)$ (8,2)
29. $\neg(p \vee q) \vdash \neg(\neg p \to q)$ (9,2)
30. $p \to q \vdash \neg p \vee q$ (9,2)
31. $\neg r \to p, r \to q \vdash q$ (9,2)
32. $p \leftrightarrow q \vdash \neg q \leftrightarrow \neg p$ (10,2)

Medium Hard

33. $\neg q \to p, \neg\neg q \to r \vdash r \vee p$ (11,2)
34. $p \to q, q \to r, \neg r, \neg p \to s \vdash s$ (11,2)
35. $\vdash (p \wedge q) \vee (\neg p \vee \neg q)$ (15,2)
36. $p \leftrightarrow q \vdash (\neg p \vee q) \wedge \neg(q \wedge \neg p)$ (15,2)
37. $p \to q, r \to q \vdash (q \vee \neg p) \wedge (q \vee \neg r)$ (19,2)

Hard

38. $\neg(p \to \neg q) \vdash \neg(q \to \neg p)$ (8,3)
39. $(p \to \neg q) \to \neg p \vdash p \to q$ (9,3)
40. $\neg(p \to \neg q) \vdash p \wedge q$ (16,3)
41. $\neg(p \to q), \neg(q \to r) \vdash p \to q$ (13,4)
42. $p \to (q \vee r) \vdash (p \to q) \vee (r \to p)$ (14,4)
43. $p \to q, \neg p \to q \vdash ((\neg p \wedge q) \vee (q \wedge \neg r)) \vee ((p \wedge q) \wedge r)$ (21,2) **challenge

4.4 Subproofs and the More Complex Inference Rules: ∨Elim

We turn now to our final inference rule, ∨**Elim**. Based solely on the form of our previous elimination rules, one might wrongly think that ∨**Elim** allows us to "detach" either disjunct from a disjunction, allowing us to move from an instance of $\ulcorner(A \vee B)\urcorner$ as an earlier line of our derivation to one of the disjuncts A or B as a later line of our derivation. Reflection shows, however, that such a rule would fail to preserve truth. While the truth of a disjunction guarantees that at least one of the disjuncts are true, if we are told only that a disjunction is true, we have no grounds for saying which disjunct in particular is true and so no grounds for inferring either disjunct individually. So, any rule that allowed us to blindly detach disjuncts from a disjunction would not be truth preserving and so would not be an inference rule of a sound proof theory. One might make a better guess at the form of this rule by claiming that we can eliminate a disjunction of the form $\ulcorner(A \vee B)\urcorner$ as an earlier line of our derivation provided we also have the *negation* of one of the disjuncts, either $\ulcorner\neg A\urcorner$ or $\ulcorner\neg B\urcorner$, also as an earlier line, thus allowing us to "detach" the other disjunct. Unlike the first guess, this suggestion has the virtue of being at least truth preserving: If a disjunction is true, then we know that at least one of its disjuncts is true and so, if we are also told that one of the disjuncts is false, then we know that the other is true. This form of inference is called *Disjunctive Syllogism* and is a basic inference rule of some proof theories, but it is **not** a basic inference rule of our system and it is **not** our ∨**Elim** rule. This form of inference is instead a very useful derivate inference rule that we will discuss in the following section.

It is possible to take our proof theory $\mathcal{F}$ and replace ∨**Elim** with DS, which is the inference rule we can schematize as follows, and the resulting system is sound and complete with respect to the semantics for PL described in chapter 3 and so is logically equivalent to our proof theory $\mathcal{F}$. Call the alternative system $\mathcal{F}\prime$.

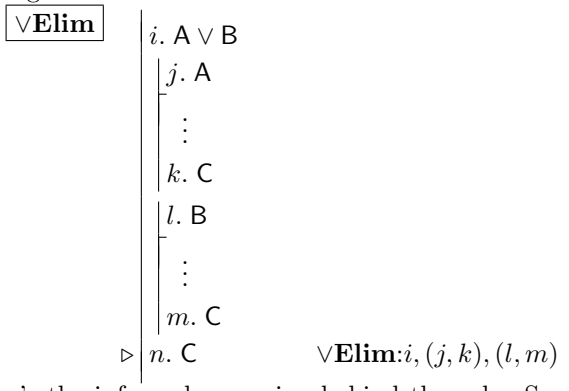
DS

	i. A ∨ B	
	j. ¬A/¬B	
▷	k. B/A	DS:i,j

The easiest way to quickly see the equivalence is to see how to derive instances of ∨**Elim** below in the alternative system $\mathcal{F}\prime$. It is somewhat tricky to figure out how to do that, but a worthwhile exercise that we will leave to the reader. A basic strategy is to think of a way to mimic the struct of ∨**Elim** using only DS and the other rules of $\mathcal{F}$.

We think that the added complexity of $\mathcal{F}$ compared to $\mathcal{F}\prime$, and so of ∨**Elim** below compared to DS, is worthwhile, as it is far easier to use the system and establish theorems with ∨**Elim** in one's tool belt.

Enough with the guessing game. The schematic form of ∨**Elim** is the following.



Here's the informal reasoning behind the rule. Suppose that we are given the truth of the disjunction $\ulcorner(A \vee B)\urcorner$. We then validly infer a sentence C from the assumption that the left disjunct A is true. We then seal that assumption and we assume that the right disjunct B is true, validly inferring the same sentence C from that assumption. Because at least one of the disjuncts A or B of our original disjunction is true, insofar as the disjunction $\ulcorner(A \vee B)\urcorner$ is true, and we know that, whichever is true, C follows, which is what we established by deriving C from each of the disjuncts individually, we are then in a position to assert that C is true as well. (More precisely, we are guaranteed that we will not transition from truth to falsity in inferring C.) We might call this form of argumentation *arguing by cases*. The form of argumentation will never, supposing the reasoning from A to C and the reasoning from B to C are both valid, lead us from truth to falsity and so preserves truth. (Proving this rigorously is significantly more difficult than proving the soundness of our other inference rules. You will be happy to hear that we will not force you through the proof, but you will also sleep well knowing that the proof exists and so you are entitled to rely on the soundness of the above inference rule.)

Let's give a simple example of this inference rule in use. Consider the following derivation establishing $((p \wedge r) \vee (q \wedge (q \rightarrow r))) \vdash r$.

Derivation establishing that $((p \wedge r) \vee (q \wedge (q \rightarrow r))) \vdash r$

1. $((\mathsf{p} \wedge \mathsf{r}) \vee (\mathsf{q} \wedge (\mathsf{q} \rightarrow \mathsf{r})))$	**Hyp**
2. $(\mathsf{p} \wedge \mathsf{r})$	**Hyp** (for $\vee$**Elim**)
3. r	$\wedge$**Elim**:2
4. $(\mathsf{q} \wedge (\mathsf{q} \rightarrow \mathsf{r}))$	**Hyp** (for $\vee$**Elim**)
5. q	$\wedge$**Elim**:4
6. $(\mathsf{q} \rightarrow \mathsf{r})$	$\wedge$**Elim**:4
7. r	$\rightarrow$**Elim**:5,6
8. r	$\vee$**Elim**:1,(2,3),(4,7)

In this derivation, we show that 'r' can be derived from each disjunct individually of the disjunction on line 1. In the first subproof, we show that 'r' is provable from '$(p \wedge r)$' and in the second subproof, we show that 'r' is provable from '$(q \wedge (q \rightarrow r)$'. Because we know that one of these disjuncts is true, insofar as we are entitled to the disjunction on line 1, we are thus entitled to 'r'.

One important point about $\vee$**Elim**. When constructing a derivation in which one suspects that $\vee$**Elim** will be used, because, for example, there is a disjunction as an earlier line of the derivation, one should set up the $\vee$**Elim** subproof structure early in the construction of the derivation, not employing our default strategy of working from the bottom up until one reaches atomic sentence-letters lacking logical complexity, forcing us to work from the top down. So, if an initial premise is a disjunction, one should suspect that one will use $\vee$**Elim** to justify the conclusion, regardless of the logical complexity of that conclusion. Waiting to use $\vee$**Elim**, thereby using it to justify a line earlier in the derivation, can lead to disaster. The following derivation establishing $\{((p \rightarrow r) \vee (q \leftrightarrow r)), \neg r\} \vdash (\neg p \vee \neg q)$ illustrates this. We will first present a correct derivation establishing the above result in which $\vee$**Elim** is used to justify the conclusion. We will then illustrate how the derivation systematically fails when $\vee$**Elim** is used earlier in the derivation, and thus later in the construction procedure, and introduction rules are used to derive the ultimate conclusion.

Derivation establishing that $\{((p \to r) \vee (q \leftrightarrow r)), \neg r\} \vdash (\neg p \vee \neg q)$

1. $((\mathsf{p} \to \mathsf{r}) \vee (\mathsf{q} \leftrightarrow \mathsf{r}))$	**Hyp**
2. $\neg\mathsf{r}$	**Hyp**
3. $(\mathsf{p} \to \mathsf{r})$	**Hyp** (for $\vee$**Elim**)
4. p	**Hyp** (for $\neg$**Intro**)
5. r	$\to$**Elim**:3,4
6. $\neg\mathsf{p}$	$\neg$**Intro**:4,(5,2)
7. $(\neg\mathsf{p} \vee \neg\mathsf{q})$	$\vee$**Intro**:6
8. $(\mathsf{q} \leftrightarrow \mathsf{r})$	**Hyp** (for $\vee$**Elim**)
9. q	**Hyp** (for $\neg$**Intro**)
10. r	$\leftrightarrow$**Elim**:8,9
11. $\neg\mathsf{q}$	$\neg$**Intro**:9,(10,2)
12. $(\neg\mathsf{p} \vee \neg\mathsf{q})$	$\vee$**Intro**:11
13. $(\neg\mathsf{p} \vee \neg\mathsf{q})$	$\vee$**Elim**:1,(3,7),(8,12)

This derivation is fairly involved and significantly more difficult than our earlier sample derivations. You can get a feel for its basic structure by seeing that the bulk of the derivation is a pair of subproofs associated with $\vee$**Elim**, with lines 3–7 being the subproof assuming the left disjunct of the disjunction on line 1 and lines 8–12 being the subproof assuming the right disjunct of that disjunction. Both of these subproofs end with the ultimate conclusion of the derivation, '$(\neg p \vee \neg q)$' as that conclusion is justified by $\vee$**Elim**. Each of those subproofs themselves contain a further subproof in conjunction with uses of $\neg$**Intro**.

While the above derivation has intrinsic interest and illustrates a number of important techniques and principles, and so pays dividends when you invest careful thought and study into it, our focus now concerns the importance of *first* using $\vee$**Elim** in constructing our derivation. We can more clearly see the importance of "using $\vee$**Elim** first" by seeing how our construction fails when we try to work from the bottom up in this case, employing the introduction rules associated with the main operator found in the ultimate conclusion to justify the last line of the derivation. So, suppose that we use $\vee$**Intro** to justify our conclusion, as follows.

1. $((\mathsf{p} \to \mathsf{r}) \vee (\mathsf{q} \leftrightarrow \mathsf{r}))$				**Hyp**
2. $\neg\mathsf{r}$				**Hyp**
3.	p	OR	q	**Hyp** (for $\neg$**Intro**)
goal : derive contradiction				
xx.	$\neg\mathsf{p}$	OR	$\neg\mathsf{q}$	$\neg$**Intro**:3,(-,-)
x. $(\neg\mathsf{p} \vee \neg\mathsf{q})$				$\vee$**Intro**:xx

We see that we have a choice at line xx: We can either derive '$\neg p$' or we can derive '$\neg q$'. And our plan for either of those options is to use $\neg$**Intro**. We will pursue both of the two paths side by side, as they each fail in analogous ways.

1. $((\mathsf{p} \rightarrow \mathsf{r}) \vee (\mathsf{q} \leftrightarrow \mathsf{r}))$	**Hyp**
2. $\neg\mathsf{r}$	**Hyp**
3. p	**Hyp**(for$\neg$**Intro**)
4. $(\mathsf{p} \rightarrow \mathsf{r})$	**Hyp**(for $\vee$ **Elim**)
5. r	$\rightarrow$ **Elim** : 3, 4
6. $(\mathsf{q} \leftrightarrow \mathsf{r})$	**Hyp**(for $\vee$ **Elim**)
z. r	???
y. r	$\vee$ **Elim** : 1, 4, 5, 6, z
xx. $\neg\mathsf{p}$	$\neg$**Intro** : 3, 2, y
x. $(\neg\mathsf{p} \vee \neg\mathsf{q})$	$\vee$ **Intro** : xx

1. $((\mathsf{p} \rightarrow \mathsf{r}) \vee (\mathsf{q} \leftrightarrow \mathsf{r}))$	**Hyp**
2. $\neg\mathsf{r}$	**Hyp**
3. q	**Hyp**(for$\neg$**Intro**)
4. $(\mathsf{p} \rightarrow \mathsf{r})$	**Hyp**(for $\vee$ **Elim**)
z. r	???
y. $(\mathsf{q} \leftrightarrow \mathsf{r}$	**Hyp**(for $\vee$ **Elim**)
x. r	$\leftrightarrow$ **Elim** : 3, 4
yy. r	$\vee$ **Elim**1, 4, 5, 6, z
xx. $\neg\mathsf{q}$	$\neg$**Intro** : 3, 2, y
zz. $(\neg\mathsf{p} \vee \neg\mathsf{q})$	$\vee$ **Intro** : xx

Whichever path we pursue, we hit a wall. When we try to derive '$\neg p$', we can generate a contradiction by deriving 'r' from the left disjunct of the disjunction on line 1, but we cannot derive it from the right disjunct. When we try to derive '$\neg q$', we have, as it were, the opposite problem: We find that we can generate a contradiction by deriving 'r' from the right disjunct of the disjunction on line 1, but we cannot derive it from the left disjunct (as we have assumed in this case 'q' at line 3 and not 'p'). In neither case can we derive a contradiction from *both* disjuncts. Indeed, we can see that both strategies are doomed to failure by reflecting on the fact that both the set $\{((p \rightarrow r) \vee (q \leftrightarrow r)), \neg r, p\}$ and the set $\{((p \rightarrow r) \vee (q \leftrightarrow r)), \neg r, q\}$ are perfectly consistent, as can be seen by constructing their joint truth tables, and so in both cases it is simply impossible to legally derive a contradiction. But that is exactly what we can do when we use $\vee$**Elim**, as we did in the first successful derivation given earlier. This illustrates, then, why it is good policy to first set up a plan to use $\vee$**Elim**, setting up the associated subproof structures, before pursuing the more general strategy of working from the bottom up whenever the initial premises include a disjunction.

We can draw the following lesson from the above discussion. *When a disjunction occurs as an earlier line of one's derivation, one should use* $\vee$**Elim** *to justify that last line as of the derivation.* While it is not always necessary to construct one's derivation in this order, using the introduction rule associated with the main operator of the conclusion inside the $\vee$**Elim** subproofs, using $\vee$**Elim** to justify the conclusion itself, it sometimes is necessary, as our previous case illustrates. Furthermore, we will never be led astray by following the practice of always setting up $\vee$**Elim** subproofs as early as possible in the construction of the derivation, whereas we will, as the case above illustrates, sometimes be led astray if we reverse that order. So, it is a good general strategy to always

set up the ∨**Elim** subproofs as early in the construction of one's derivation as one has a disjunction as an earlier line in that derivation. However, a downside of this general policy is that one's derivation will sometimes be longer than the derivation that results from reversing this order of operation. But that cost is balanced by the benefit of being guaranteed a strategy that will never lead one down the dead ends illustrated by our discussion above.

If you find yourself struggling with derivations that employ ∨**Elim**, we suggest that you simply employ the above policy without exception. However, we can formulate a more complex principle for constructing derivations using ∨**Elim** that produces shorter derivations than is produced through a blind allegiance to the simplistic principle formulated in the previous paragraph. The simplisitic principle tends to produce longer derivations when ∨**Elim** is used in conjunction with →**Intro** (and ↔**Intro**) and so derivations with a disjunction in the premises and a material conditional (or a bi-conditional) as a conclusion. (We emphasis, however, the policy described in the previous paragraph will never fail to deliver a derivation when a derivation is to be had; it is simply that the derivation delivered is not always the most elegant derivation of the derivations to be found.) The following example offers an illustration.

Suppose that we are asked to prove that $\{(p \rightarrow \neg q), (r \vee q)\} \vdash (\neg r \rightarrow \neg p)$. If we employed the policy of always using ∨**Elim** whenever there is a disjunction in an earlier line, the following would be our best derivation.

1. (p → ¬q)	**Hyp**
2. (r ∨ q)	**Hyp**
3. r	**Hyp** (for ∨**Elim**)
4. ¬r	**Hyp** (for →**Intro**)
5. p	**Hyp** (for ¬**Intro**)
6. ¬p	¬**Intro**:5,(3,4)
7. (¬r → ¬p)	→**Intro**:4,6
8. q	**Hyp** (for ∨**Elim**)
9. ¬r	**Hyp** (for →**Intro**)
10. p	**Hyp** (for ¬**Intro**)
11. ¬q	→**Elim**:1,10
12. ¬p	¬**Intro**:10,(8,11)
13. (¬r → ¬p)	→**Intro**:9,12
14. (¬r → ¬p)	∨**Elim**:2,(3,7),(8,13)

By contrast, if we use →**Intro** to justify the conclusion and use ∨**Elim** to justify the last line of the →**Intro** subproof, then we can offer the following derivation, which is two lines shorter and derives the conditional one fewer times than the above derivation.

1. $(\mathsf{p} \rightarrow \neg \mathsf{q})$	**Hyp**
2. $(\mathsf{r} \vee \mathsf{q})$	**Hyp**
3. $\neg \mathsf{r}$	**Hyp** (for $\rightarrow$**Intro**)
4. r	**Hyp** (for $\vee$**Elim**)
5. p	**Hyp** (for $\neg$**Intro**)
6. $\neg \mathsf{p}$	$\neg$**Intro**:5,(3,4)
7. q	**Hyp** (for $\vee$**Elim**)
8. p	**Hyp** (for $\neg$**Intro**)
9. $\neg \mathsf{q}$	$\rightarrow$**Elim**:1,8
10. $\neg \mathsf{p}$	$\neg$**Intro**:8,(7,9)
11. $\neg \mathsf{p}$	$\vee$**Elim**:2,(4,6),(7,10)
12. $(\neg \mathsf{r} \rightarrow \neg \mathsf{p})$	$\rightarrow$**Intro**:3,11

So, a general exception to the simple policy of using $\vee$**Elim** in the construction of a derivation when an earlier line in the derivation contains a disjunction is to use $\rightarrow$**Intro** first when the goal to be justified is a material conditional, in which case $\vee$**Elim** should instead be used within the $\rightarrow$**Intro** subproof. A similar exception should be carved for $\leftrightarrow$**Intro**. The differences between the simplistic policy for constructing derivations using $\vee$**Elim** (crudely, whenever an earlier line of one's derivation contains a disjunction, use $\vee$**Elim** to justify the last unjustified line) and the more complicated poliy (do as above, unless the latest line is a material conditional or material bi-conditional, in which case, first use the appropriate conditional introduction rule to justify the last line of the derivation and then use $\vee$**Elim** to justify the last line of the conditional subproof) is only one of elegance. The simplistic policy will always work, the primary difference being that the resulting derivation will be longer. So, if you are struggling, we suggest that you adopt the simplistic policy.

We will end this section with a very brief, general overview of the use of the $\vee$**Elim** inference rule, ignoring the above discussed complex policy. When an earlier line of one's derivation is a sentence whose main operator is '$\vee$', i.e., when one has a disjunction on an earlier line of the derivation, one should use $\vee$**Elim** to justify the last line of the constructed derivation without a justification. One then sets up a pair of subproofs, assuming the left disjunct in the first subproof and the right disjunct in the second subproof and then seeking to derive within each of those subproofs whatever sentence one is planning to justify in the derivation with the use of $\vee$**Elim**.

∨Elim Derivation Problems

Each of the following claims of provability can be established by constructing a derivation that use all of the inference rules. The derivations center on the use of ∨**Elim**.

Easy

1. $p \wedge (q \vee r) \vdash (p \wedge q) \vee (p \wedge r)$ (10,1)
2. $(p \vee q) \rightarrow s, s \rightarrow (s \leftrightarrow t), (p \vee q) \vee r \vdash r \vee (s \leftrightarrow t)$ (10,1)
3. $p \rightarrow q, \neg q, p \vee r \vdash r$ (10,2)
4. $(p \wedge q) \rightarrow (r \vee s), p \wedge q, \neg s \vdash r$ (10,2)

Medium Easy

5. $p \rightarrow q, q \rightarrow (r \rightarrow s), t \vee r, t \rightarrow u, \neg u, r \rightarrow p \vdash s$ (17,2)
6. $(p \rightarrow q) \rightarrow (r \vee s), \neg r \rightarrow (\neg r \rightarrow q), p \rightarrow \neg r, p \vdash s$ (16,2)
7. $\neg p \vee q, \neg p \rightarrow r, \neg r \vdash q \vee s$ (11,2)
8. $\neg(p \wedge q), (p \wedge q) \vee r), r \rightarrow s \vdash r \wedge s$ (11,2)
9. $(p \leftrightarrow q) \vee (q \rightarrow \neg r), (p \leftrightarrow r) \rightarrow s, q, \neg s \vdash \neg r$ (11,2)
10. $p \rightarrow q, \neg r \rightarrow s, p \vee \neg r, \neg q \vdash s$ (12,2)
11. $p \rightarrow (q \wedge \neg r), s \rightarrow \neg(q \wedge \neg r), t \vee s, \neg t \vdash \neg p$ (13,2)
12. $p \rightarrow q, r \rightarrow \neg q, s \vee r, \neg s \vdash \neg p$ (13,2)
13. $p \rightarrow q, q \rightarrow (r \vee s), p \wedge \neg r \vdash s$ (13,2)
14. $p \rightarrow (q \wedge r), \neg r \vee \neg q \vdash \neg p$ (13,2)
15. $\neg p \rightarrow (q \wedge r), r \rightarrow (s \rightarrow p), t \vee \neg p, \neg t \vdash \neg s$ (13,2)
16. $\neg q \rightarrow (q \leftrightarrow r), \neg q \wedge (p \vee r) \vdash p$ (13,2)
17. $p \vee (q \rightarrow r), \neg q \rightarrow (s \wedge \neg t), \neg p, \neg r \vdash \neg t$ (14,2)
18. $p \rightarrow q, \neg r \vee s, \neg s \wedge (q \rightarrow r) \vdash \neg p$ (14,2)
19. $(p \vee q) \wedge (r \vee s) \vdash (q \vee r) \vee (p \wedge s)$ (15,2)

Medium

20. $\neg p \rightarrow \neg q, \neg r \rightarrow \neg s, t \vee (\neg p \vee \neg r), \neg t \vdash \neg q \vee \neg s$ (17,2)
21. $(p \rightarrow q) \rightarrow (r \rightarrow (s \vee t)), (r \leftrightarrow \neg s) \rightarrow q, r, \neg s \vdash t$ (18,2)
22. $p \rightarrow (q \leftrightarrow \neg r), (r \leftrightarrow \neg q) \rightarrow (s \vee t), p \rightarrow \neg s, p \vdash t$ (23,2)

23. $p \vee q \vdash \neg p \rightarrow q$ (9,3)

24. $p \vee t, \neg q \rightarrow \neg p \vdash \neg q \rightarrow t$ (11,3)

25. $\neg p \wedge \neg q \vdash \neg(p \vee q)$ (11,3)

26. $\neg p \vee \neg q \vdash \neg(p \wedge q)$ (11,3)

27. $q \rightarrow (\neg p \vee r), \neg p \leftrightarrow r \vdash (\neg p \vee q) \rightarrow \neg p$ (12,3)

28. $p \vee t, r \leftrightarrow p, \neg q \rightarrow \neg p \vdash \neg q \rightarrow t$ (12,3)

29. $q \rightarrow (\neg p \vee r), r \rightarrow \neg p \vdash (\neg p \vee q) \rightarrow \neg p$ (12,3)

30. $(p \wedge q) \vee r, \neg p, s \rightarrow \neg r \vdash \neg s$ (13,3)

31. $(q \leftrightarrow p) \rightarrow r, q \vee \neg r, \neg q \wedge (\neg p \rightarrow (q \leftrightarrow p)) \vdash p$ (15,3)

Medium Hard

32. $(q \vee s) \rightarrow (p \vee r), t \vee (q \vee s), \neg r, \neg t \vdash p$ (17,3)

33. $q \rightarrow (\neg r \rightarrow s), p \vee q, \neg p, r \rightarrow p \vdash s$ (18,3)

34. $(p \vee q) \vee (s \vee \neg t), r \rightarrow (\neg p \wedge \neg q), (s \vee \neg t) \rightarrow \neg s, r \vdash \neg t$ (19,3)

35. $(p \wedge \neg q) \rightarrow r, r \rightarrow (q \vee s), t \vee (p \wedge \neg q), t \rightarrow q, \neg q \vdash s$ (20,3)

36. $p \rightarrow (\neg(q \rightarrow r) \rightarrow (q \vee \neg r)), (q \rightarrow r) \vee p, (q \rightarrow r) \rightarrow t, \neg t, \neg q \vdash \neg r$ (21,3)

37. $p \vee (q \vee (\neg r \vee s)), t \rightarrow \neg p, t \rightarrow \neg q, t \wedge r \vdash s$ (24,3)

Hard

38. $p \rightarrow \neg q, r \vee q \vdash \neg r \rightarrow \neg p$ (14,4)

39. $p \rightarrow (q \vee r), q \rightarrow (\neg r \rightarrow s) \vdash p \rightarrow (s \vee r)$ (18,4)

40. $p \rightarrow (r \vee q), q \rightarrow (\neg r \rightarrow s) \vdash p \rightarrow (s \vee r)$ (18,4)

41. $\vdash \neg p \vee \neg(\neg q \wedge (\neg p \vee q))$ (18,4) **challenge

42. $p \vee (q \wedge r), (\neg q \vee \neg r) \vee s) \vdash p \vee s$ (20,5)**challenge

For each of the following claims of logical consequence, decide whether or not it is true. (Hint: Use truth tables to do this!) If it is true true, prove that it is true with truth tables and then derive. If false, prove that it is false with truth tables and explicitly state a countermodel.

43. $p \rightarrow (q \wedge r), \neg(q \wedge r) \models \neg p$

44. $p \rightarrow (q \wedge r), q \wedge r \models p$

45. $\neg(p \vee \neg q), \neg(q \wedge r) \models r$

46. $(p \wedge \neg q) \rightarrow r, \neg q \wedge p \models r$

47. $(p \wedge \neg q) \rightarrow r, p \wedge r \models \neg q$
48. $(p \wedge \neg q) \rightarrow r, \neg(p \wedge r) \models \neg q$
49. $p \rightarrow q, q \rightarrow \neg s \models p \rightarrow \neg s$
50. $p \rightarrow q, \neg q \rightarrow \neg s \models p \rightarrow \neg s$
51. $p \rightarrow q, r \rightarrow \neg s, p \vee r \models q \wedge \neg s$
52. $p \rightarrow q, r \rightarrow q, p \vee r \models q$

4.5 More Advanced Derivation Techniques

The derivation system $\mathcal{F}$ consists of the eleven inference rules discussed above in the earlier sections of this chapter and collected together into a single group on the last page of this chapter below. Given a language with all five of our truth-functional operators as primitive symbols of the language and a classical semantics with the truth definitions described in chapter 3, this system is both complete, in the sense that, for any sentence B and set of sentences Γ in the language of PL, if $\Gamma \models B$, then $\Gamma \vdash_{\mathcal{F}} B$ (i.e., whenever B is a logical consequence of Γ, there is a legal derivation using only the inference rules of $\mathcal{F}$ with only Γ, or some finite subset of Γ, in the cases in which Γ has an infinite size, as premises and B as the last line), and sound, in the sense that, for any sentence B and set of sentences Γ in the language of PL, if $\Gamma \vdash_{\mathcal{F}} B$, then $\Gamma \models B$. Omitting any of the eleven inference rules of $\mathcal{F}$ would lead to an incomplete proof theory, in the sense that there would be logical consequences without corresponding derivations in the resulting theory. Furthermore, adding additional inference rules results in a system that is either a redundant system, in the sense that for any legal derivation in the extended system, there is also some (not necessarily the same) legal derivation in the original system $\mathcal{F}$, or an unsound system, in the sense that the system legalizes at least one derivation whose conclusion is not a logical consequence of its premises.

Establishing these completeness and soundness results is well beyond the scope of an introductory course, as is establishing the equivalence between $\mathcal{F}$ and other sound systems for PL. However, we can take these results for granted and use them to develop first a test for detecting fatal errors in our attempt to construct derivations and second strategies for constructing more difficult derivations, both of which count as more advanced techniques for constructing derivations, both of which count as more advanced techniques for constructing derivations.

Let's begin with a common mistake, particularly with students in the advanced beginner and intermediate levels of constructing derivations. You start off on a strategy for deriving a result that is in fact a hopeless dead end. Needless to say, your attempts fail. But how do you know whether your failure is just because you haven't tried hard enough or because you should scratch your

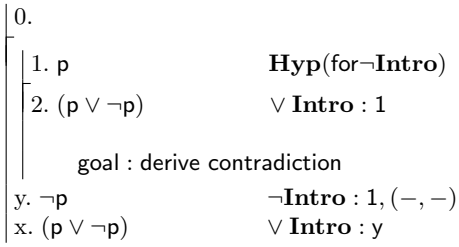
doomed strategy and begin afresh? Our first more advanced technique is intended to provide an escape from the traps of either abandoning a promising strategy too early or continuing with a hopeless strategy too long.

Recall the soundness result mentioned in the first paragraph of this section. That result guarantees that the only sentences that can be legally derived from a given set of sentences Γ in our proof theory $\mathcal{F}$ are all a logical consequence of Γ. An equivalent way of stating this same result is to say that, if $\Gamma \models B$, then $\Gamma \nvdash B$ and so there is not a legal derivation of B from Γ in the $\mathcal{F}$, whenever B is not a logical consequence of Γ. This is a useful fact as it is a purely mechanical task to use truth tables to test for whether or not a sentence of the language of PL is a logical consequence of a set of sentences of PL and so one can use truth tables to determine whether or not one's failure to successfully navigate one's way through a strategy is due to one's own shortcomings, in which case one should keep working through the problem walking down the same path, or is instead due to the fact that the result cannot be derived, in which case one should back up and pursue a different strategy that does not require deriving the underivable result identified. Notice that this can be used on *any* sentence within the derivation, whether it is the last line or a line embedded within subproofs, although this last raises issues that we will discuss below.

Here is a concrete illustration. Suppose we are given the assignment of establishing $\vdash (p \vee \neg p)$, which is an instance of the Law of Excluded Middle derived above in section 4.3. Having just learned of this first technique and eager to apply it, we check to see whether or not the result can be derived. Relying on the soundness of our proof theory, then, we know that $\vdash (p \vee \neg p)$ only if $\models (p \vee \neg p)$. (That is, we can derive '$(p \vee \neg p)$' from the null set and so from any set of initial premises whatsoever only if that sentence is a logical truth.) So, we use truth tables to check, as follows.

p	$(p$	$\vee$	$\neg$	$p)$
T	T	T	F	T
F	F	T	T	F

Our assignment of establishing $\vdash (p \vee \neg p)$, then, passes the "soundness test" and so we begin constructing our derivation. Noting that our conclusion contains logical complexity, the main operator being '$\vee$', and forgetting the discussion of the derivation from section 4.3 above, we naturally begin constructing our derivation by planning to use the introduction rule associated with the main operator of our conclusion, as follows.



0.	
1. ¬p	**Hyp**(for¬**Intro**)
2. (p ∨ ¬p)	∨ **Intro** : 1
goal : derive contradiction	
z. ¬¬p	¬**Intro** : 1, (−, −)
y. p	¬**Elim** : z
x. (p ∨ ¬p)	∨ **Intro** : y

Both of these derivations are hopeless dead ends. While in each case, we can derive our desired goal '$(p \vee \neg p)$' from the initial assumption, that is of little use, as it is not possible to bring that sentence outside of the initial subproof and it is not possible to legally derive the needed contradiction. But how can we be certain that the failure isn't our own shortcoming?

Start with the first derivation on the left. If we could legally complete that derivation, then we would have a derivation with the second to last line, outside any subproofs, containing the sentence '$\neg p$'. In that case, however, there would also be a legal derivation whose last line is '$\neg p$' and whose initial hypotheses are the null set; simply take that supposed completed derivation and delete the last line, '$(p \vee \neg p)$' and the result will be exactly that derivation, as the second to last line in the original derivation is not embedded in any subproofs. In that case, then, $\vdash \neg p$ and so, by the soundness of our proof theory, $\models \neg p$. But there is a countermodel, as follows.

$$\Rightarrow \left|\begin{array}{c||cc|} p & \neg & p \\ \hline \mathrm{T} & \mathrm{F} & \mathrm{T} \end{array}\right|$$

More generally, we can test the viability of our derivation strategy by testing to see if earlier the sentences on an earlier line in our derivation are indeed logical consequences of the earlier dependent lines of the derivation. When the sentence on a line, like the sentence on line y in our above imagined derivation, is not embedded in a subproof, then the sentence can be legally derived in $\mathcal{F}$ if and only if the sentence is a logical consequence of the initial premises of the derivation. If one discovers that a sentence is *not* a logical consequence of the premises, then one can infer that one's derivation strategy is a hopeless dead end. On the other hand, if one discovers that each of the earlier lines in one's derivation is a logical consequence of the earlier lines, then one can rest assured that there is indeed a derivation and so any problems one is encountering can in fact be overcome. (This fact relies on the completeness of our proof theory instead of its soundness which drives the negative test mentioned above. From the fact that our proof theory is sound and that $\Gamma \not\models B$, we can infer the $\Gamma \nvdash B$. To infer that there is a derivation, on the other hand, we do not need the assurance that the soundness result delivers that our proof theory cannot be legally used to derive invalidities, but instead the assurance that our proof theory is powerful enough to be capable of delivering a derivation for every validity.)

In the example discussed in the previous paragraphs, the problematic line of the derivation was not embedded in a subproof. In that case, as we said, we use our strategy by simply checking to see that the sentence on that unembedded line is a logical consequence of the initial hypotheses of the derivation. However, sometimes the problematic line is embedded in a subproof. In that case, our strategy requires that we check to see whether or not the sentence on the line in question is a logical consequence of the original hypotheses together with all of the temporary hypotheses introduced by any subproof the line in question is embedded inside. We'll start with a simple case to illustrate this point and then move to a more complex example.

Suppose we are asked to establish $(r \rightarrow (p \rightarrow q)) \vdash (r \rightarrow (\neg p \vee q))$ and we begin with the following derivation.

1. $(\mathsf{r} \rightarrow (\mathsf{p} \rightarrow \mathsf{q}))$	**Hyp**
2. r	**Hyp** (for $\rightarrow$**Intro**)
3. p	**Hyp** (for $\neg$**Intro**)
goal : derive contradiction	
z. $\neg\mathsf{p}$	$\neg$**Intro**:3,(-,-)
y. $(\neg\mathsf{p} \vee \mathsf{q})$	$\vee$**Intro**:z
x. $(\mathsf{r} \rightarrow (\neg\mathsf{p} \vee \mathsf{q}))$	$\rightarrow$**Intro**:2,y

We hit a wall when we try to derive the contradiction and wonder whether we should continue to struggle, the shortcoming being with the derivation's constructor, or pursue a different strategy, the fault being instead with the derivation's design. Our problem is with line z and so we check to see whether the sentence on that line is a logical consequence of the sentences on its dependent lines. Because line z is inside a subproof, its dependent lines include both the initial hypothesis and the introduced hypothesis on line 2. So, we check whether or not $\{(r \rightarrow (p \rightarrow q)), r\} \models \neg p$ by constructing a joint truth table and we see that there is a countermodel, as follows.

	p	q	r	$(r$	$\rightarrow$	$(p$	$\rightarrow$	$q))$	r	$\neg$	p
$\Rightarrow$	T	T	T	T	T	T	T	T	T	F	T

Because our proof theory is sound, then, the fact that $\{(r \rightarrow (p \rightarrow q)), r\} \not\models \neg p$, demonstrated by the interpretation above, entails that our derivation strategy above is hopeless and should be abandoned.

As a more complicated case, consider the following derivation strategy for establishing $(p \leftrightarrow (q \leftrightarrow (r \leftrightarrow s))) \vdash (((p \leftrightarrow q) \leftrightarrow r) \leftrightarrow s)$.

1. (p ↔ (q ↔ (r ↔ s)))	**Hyp**
2. ((p ↔ q) ↔ r)	**Hyp** (for ↔**Intro**)
3. ¬s	**Hyp** (for ¬**Intro**)
goal : derive contradiction	
zz. ¬¬s	¬**Intro**:3,(-,-)
z. s	¬**Elim**:zz
y. s	**Hyp** (for ↔**Intro**)
goal : derive ((p ↔ q) ↔ r)	
x. (((p ↔ q) ↔ r) ↔ s)	↔**Intro**:(2,z),(y,-)

Focus on the first half of the derivation, the left-right direction of the two ↔**Intro** subproofs. We can verify its plausibility of deriving a contradiction inside the subproof that tarts on line 3 by using truth tables, checking whether or not the dependent sentences, the sentences on lines 1, 2, and 3, are consistent, as follows.

p	q	r	s	$(p$	↔	$(q$	↔	$(r$	↔	s	$)))$	$((\ p$	↔	$q)$	↔	$r)$	¬	s
T	T	T	T	T	T	T	T	T	T	T		T	T	T	T	T	F	T
T	T	T	F	T	F	T	F	T	F	F		T	T	T	T	T	T	F
T	T	F	T	T	F	T	F	F	F	T		T	T	T	F	F	F	T
T	T	F	F	T	T	T	T	F	T	F		T	T	T	F	F	T	F
T	F	T	T	T	F	F	F	T	T	T		T	F	F	F	T	F	T
T	F	T	F	T	T	F	T	T	F	F		T	F	F	F	T	T	F
T	F	F	T	T	T	F	T	F	F	T		T	F	F	T	F	F	T
T	F	F	F	T	F	F	F	F	T	F		T	F	F	T	F	T	F
F	T	T	T	F	F	T	T	T	T	T		F	F	T	F	T	F	T
F	T	T	F	F	T	T	F	T	F	F		F	F	T	F	T	T	F
F	T	F	T	F	T	T	F	F	F	T		F	F	T	T	F	F	T
F	T	F	F	F	F	T	T	F	T	F		F	F	T	T	F	T	F
F	F	T	T	F	T	F	F	T	T	T		F	T	F	T	T	F	T
F	F	T	F	F	F	F	T	T	F	F		F	T	F	T	T	T	F
F	F	F	T	F	F	F	T	F	F	T		F	T	F	F	F	F	T
F	F	F	F	F	T	F	F	F	T	F		F	T	F	F	F	T	F

As the above joint truth table demonstrates, there is no interpretation that makes all three of the sentences in the set $\{(p \leftrightarrow (q \leftrightarrow (r \leftrightarrow s))), ((p \leftrightarrow q) \leftrightarrow r), \neg s\}$ all true together and so the set is inconsistent. By the completeness of our proof theory, then, a contradiction is provable from that set and so our strategy of using ¬**Intro** on line yy is viable. Notice that we are not really given much indication how to carry the derivation out by this method; we are only

given an assurance that there is a derivation to be discovered.

Finally, suppose that we wonder whether or not the second subproof is viable. We can use the first version of our method to see. We can derive our goal in the second primary subproof provided '$((p \leftrightarrow q) \leftrightarrow r)$' is a logical consequence of the sentences on its dependent lines: Namely, lines 1 and y. (Remember: We only take the hypotheses in the subproof structures that the line(s) in question lie within.) So, we check whether or not $\{(p \leftrightarrow (q \leftrightarrow (r \leftrightarrow s))), s\} \models (p \leftrightarrow q) \leftrightarrow r)$. We can do that by again completing the joint truth table for those sentences and checking for the existence of a countermodel, as follows.

p	q	r	s	$(p$	$\leftrightarrow$	$(q$	$\leftrightarrow$	$(r$	$\leftrightarrow$	s	$)))$	s	$(p$	$\leftrightarrow$	$q)$	$\leftrightarrow$	$r)$
T	T	T	T	T	T	T	T	T	T	T		T	T	T	T	T	T
T	T	T	F	T	F	T	F	T	F	F		F	T	T	T	T	T
T	T	F	T	T	F	T	F	F	F	T		T	T	T	T	F	F
T	T	F	F	T	T	T	T	F	T	F		F	T	T	T	F	F
T	F	T	T	T	F	F	F	T	T	T		T	T	F	F	F	T
T	F	T	F	T	T	F	T	T	F	F		F	T	F	F	F	T
T	F	F	T	T	T	F	T	F	F	T		T	T	F	F	T	F
T	F	F	F	T	F	F	F	F	T	F		F	T	F	F	T	F
F	T	T	T	F	F	T	T	T	T	T		T	F	F	T	F	T
F	T	T	F	F	T	T	F	T	F	F		F	F	F	T	F	T
F	T	F	T	F	T	T	F	F	F	T		T	F	F	T	T	F
F	T	F	F	F	F	T	T	F	T	F		F	F	F	T	T	F
F	F	T	T	F	T	F	F	T	T	T		T	F	T	F	T	T
F	F	T	F	F	F	F	T	T	F	F		F	F	T	F	T	T
F	F	F	T	F	F	F	T	F	F	T		T	F	T	F	F	F
F	F	F	F	F	T	F	F	F	T	F		F	F	T	F	F	F

On each row, either the sentence to the right is true or one of the sentences to the left is false; so there are no counterexamples and so $\{(p \leftrightarrow (q \leftrightarrow (r \leftrightarrow s))), s\} \models (p \leftrightarrow q) \leftrightarrow r)$. So, by the completeness of $\mathcal{F}$, we are assured that there are legal transitions from 1 and y to the goal of the subproof that begins on line y. (This derivation is very challenging. We will leave it as an exercise to the reader to complete the derivation having offered assurances that the general strategy outlined above bear fruit.)

This completes our discussion of our first more advanced technique. The technique can be quite labor intensive, as the last example shows. So we suggest using it sparingly. Understanding why the method works, however, involves understanding key connections between the semantic methods of truth tables and proof-theoretic methods of derivation, which is very important in gaining a deeper understanding of logic.

Our second set of more advanced strategies concern a set of very useful patterns of inference, which we will call derivative inference rules. Unlike the 11 inference rules comprising $\mathcal{F}$, these inference patterns do not need to be taken as basic inference rules, as they can be derived using just the rules of $\mathcal{F}$. But they are useful in the sense that they can be employed as intermediate steps in constructing more difficult derivations, allowing the construction of those derivations to be broken into a series of smaller stages and allowing very useful transformations of sentences, in particular, sentences whose main connective is a negation. Once one is in a position to prove an arbitrary instance of the derivative inference rule using only the official rules of $\mathcal{F}$, one can rely upon it and use it to fill in the steps connecting those stages and justifying those transformations as one constructs more difficult derivations. So, regardless of whether or not one is allowed to perform the transitions these derivative rules license in a single step, as one can if $\mathcal{F}$ is extended with these derivative rules, or if one is restricted to only using the inference rules in $\mathcal{F}$ in one's derivations,

having these useful derivative inference rules in your tool belt makes you better at constructing derivations.

We will begin with **Modus Tollens**, which allows us to infer $\lceil \neg A \rceil$ from $\lceil (A \rightarrow B) \rceil$ and $\lceil \neg B \rceil$ as earlier lines. Below we will derive an arbitrary instance of this pattern of inference. The idea behind this level of abstraction is that, for any sentences A and B, whether simple or complex, if we have a derivation with sentences of the form $\lceil (A \rightarrow B) \rceil$ and $\lceil \neg B \rceil$ as earlier lines, then, by replacing all occurrences of A and of B in the derivation schema below, we can legally transition to the corresponding instance of $\lceil \neg A \rceil$. (Note: In the schematic version, these two lines are taken as hypotheses. It may be, however, that in the concrete instance of interest to you, one or both of the lines at which the instances of these sentences occur are justified by one of the other inference rules. That difference will not affect the transition itself from those earlier lines to our ultimate goal $\lceil \neg A \rceil$.)

Schematic derivation of **Modus Tollens**

1. $(\mathsf{A} \rightarrow \mathsf{B})$	**Hyp**
2. $\neg\mathsf{B}$	**Hyp**
3. A	**Hyp** (for $\neg$**Intro**)
4. B	$\rightarrow$**Elim**:1,3
5. $\neg\mathsf{A}$	$\neg$**Intro**:3,(2,4)

What this schematic derivation illustrates is how, in three easy steps, one can infer the negation of a conditional's antecedent from that conditional and the negation of its consequent.

The next series of useful derivation inference rules are often call *DeMorgan Laws* after the 19^{th}-century British mathematician and logician Augustus De Morgan. The "laws" concern the equivalences between disjunction and conjunction. For the purposes of a proof theory, however, it is more useful to think of these principles as distribution principles for the negation operator '$\neg$'. The equivalences should be familiar from our discussion of logical equivalence and truth tables in earlier chapters. We repeat them here, where '$A \equiv B$' is short for 'A and B are equivalent'.

De Morgan Equivalences

- $(A \wedge B) \equiv \neg(\neg A \vee \neg B)$
- $(A \vee B) \equiv \neg(\neg A \wedge \neg B)$

By the completeness of $\mathcal{F}$, then, we are assured that there are derivations justifying each of the following inferences.

De Morgan Derivative Inference Rules

- **DM**$\wedge$: From $\ulcorner(A \wedge B)\urcorner$, infer $\ulcorner\neg(\neg A \vee \neg B)\urcorner$.
- **DM**$\neg\vee$: From $\ulcorner\neg(A \vee B)\urcorner$, infer $\ulcorner(\neg A \wedge \neg B)\urcorner$
- **DM**$\vee$: From $\ulcorner(A \vee B)\urcorner$, infer $\ulcorner\neg(\neg A \wedge \neg B)\urcorner$.
- **DM**$\neg\wedge$: From $\ulcorner\neg(A \wedge B)\urcorner$, infer $\ulcorner(\neg A \vee \neg B)\urcorner$

Of particular interest here are the inference patterns called 'DM$\neg\vee$' and 'DM$\neg\wedge$', as they allow us to "push" a negation across conjunction and disjunction operators, thereby transforming a negation into either a disjunction or a conjunction, where the latter is often times easier to manipulate given our proof theory.

We will offer a schematic derivation of one of these inference patterns, DM$\neg\wedge$, in $\mathcal{F}$, saving the others for exercises.

Schematic derivation of **DM**$\neg\wedge$

1.	$\neg(A \wedge B)$	**Hyp**
2.	$\neg(\neg A \vee \neg B)$	**Hyp** (for $\neg$**Intro**)
3.	$\neg A$	**Hyp** (for $\neg$**Intro**)
4.	$(\neg A \vee \neg B)$	$\vee$**Intro**:3
5.	$\neg\neg A$	$\neg$**Intro**:3,(2,4)
6.	A	$\neg$**Elim**:5
7.	$\neg B$	**Hyp** (for $\neg$**Intro**)
8.	$(\neg A \vee \neg B)$	$\vee$**Intro**:7
9.	$\neg\neg B$	$\neg$**Intro**:7,(2,8)
10.	B	$\neg$**Elim**:9
11.	$(A \wedge B)$	$\wedge$**Intro**:6,10
12.	$\neg\neg(\neg A \vee \neg B)$	$\neg$**Intro**:2,(1,11)
13.	$(\neg A \vee \neg B)$	$\neg$**Elim**:12

With this schematic derivation in our pocket, we can now rest assured that we have the means to transform the negation of a conjunction into a disjunction, where the latter is, given our inference rules, often much easier to deal with as an earlier line in our derivation than the former. Thus, if we have the negation of a disjunction as an earlier line of our derivation, we should realize that we can turn that into a disjunction and use that disjunction, then, as an intermediate step from our earlier lines to our conclusion, helping us frame a strategy and break a complicated derivation up into easier steps.

While De Morgan focused on the interrelations between the truth-functions expressed by negation, conjunction, and disjunction operators, our language of PL contains vocabulary for conditionals as well. We will, then, extend first the set of truth-functional equivalences and then the corresponding derivative inference rules to include our two conditional operators as well. (We will break the

equivalences into three groups: The first simply adding the material conditional to our original list of De Morgan equivalences, the second concerning the material conditional directly, and the third concerning the material bi-conditional.)

Extended De Morgan Equivalences

- $(A \wedge B) \equiv \neg(\neg A \vee \neg B) \equiv \neg(A \rightarrow \neg B)$
- $(A \vee B) \equiv \neg(\neg A \wedge \neg B) \equiv (\neg A \rightarrow B)$
- $\neg(A \leftrightarrow B) \equiv ((A \wedge \neg B) \vee (\neg A \wedge B))$

By the completeness of $\mathcal{F}$, then, we are assured that there are derivations justifying each of the following inferences patterns. We will continue to focus only on the negation distribution rules, as they are the most useful for present purposes.

Extended De Morgan Derivative Inference Rules

- $\mathbf{DM}_E\neg\wedge$: From $\lceil\neg(A \wedge B)\rceil$, infer $\lceil(\neg A \vee \neg B)\rceil$ or infer$\lceil(A \rightarrow \neg B)\rceil$.
- $\mathbf{DM}_E\neg\,\rightarrow$: From $\lceil\neg(A \rightarrow B)\rceil$, infer $\lceil(A \wedge \neg B)\rceil$.
- $\mathbf{DM}_E\neg\,\leftrightarrow$: From $\lceil\neg(A \leftrightarrow B)\rceil$, infer $\lceil((A \wedge \neg B) \vee (B \wedge \neg A))\rceil$.

Notice that we do not bother with negation pushing derivative rules involve '$\vee$' and '$\rightarrow$' as those equivalences do not warrant turning a negative sentence into a positive sentence but instead license the transition from one negation into another negation, which is rarely of much use in constructing derivations. We will leave as exercises the justification of these transitions in $\mathcal{F}$.

This completes our discussion of the derivation system $\mathcal{F}$. We end by collecting all eleven rules of system $\mathcal{F}$ in a single place, on the following page.

Inference Rules for $\mathcal{F}$

Hyp

$i.$ A

$\wedge$Intro

$i.$ A
$j.$ B
$\triangleright$ $k.$ A $\wedge$ B $\wedge$**Intro**: i, j

$\wedge$Elim

$i.$ A $\wedge$ B
$\triangleright$ $j.$ A/B $\wedge$**Elim**: i

$\vee$Intro

$i.$ A/B
$\triangleright$ $j.$ A $\vee$ B $\vee$**Intro**: i

$\vee$Elim

$i.$ A $\vee$ B
  $j.$ A
  ⋮
  $k.$ C
  $l.$ B
  ⋮
  $m.$ C
$\triangleright$ $n.$ C $\vee$**Elim**:$i, (j, k), (l, m)$

$\neg$Intro

  $i.$ A
  ⋮
  $j.$ B
  $k.$ $\neg$B
$\triangleright$ $l.$ $\neg$A $\neg$**Intro**:$i, (j, k)$

$\neg$Elim

$i.$ $\neg\neg$A
$\triangleright$ $j.$ A $\neg$ **Elim:** i

$\rightarrow$Intro

  $i.$ A
  ⋮
  $j.$ B
$\triangleright$ $k.$ A $\rightarrow$ B $\rightarrow$ **Intro:** i,j

$\rightarrow$Elim

$i.$ A $\rightarrow$ B
$j.$ A
$\triangleright$ $k.$ B $\rightarrow$ **Elim:** i, j

$\leftrightarrow$Intro

  $i.$ A
  ⋮
  $j.$ B
  $k.$ B
  ⋮
  $l.$ A
$\triangleright$ $m.$ A $\leftrightarrow$ B $(i, j), (k, l)$

$\leftrightarrow$Elim

$i.$ A $\leftrightarrow$ B
$j.$ A/B
$\triangleright$ $k.$ B/A $\leftrightarrow$**Elim**: i, j