CHAPTER 1

INFORMAL LOGIC

This chapter introduces logic from an intuitive, or **informal,** point of view. In later chapters, as we examine not specific arguments but argument forms, we will look back at the concepts introduced here from various formal viewpoints. The informal stance, however, is fundamental. It is the milieu out of which the study of logic emerges and to which, ultimately, it must return—on pain of losing its roots and becoming irrelevant to the concerns that produced it.

1.1 WHAT IS LOGIC?

Logic is the study of reasoning. Reasoning is a process of thought, but there exists no uncontroversial method for studying thought. As a result, contemporary logic, which likes to think of itself as founded on hard (though perhaps not empirical) facts, has nothing to say about thought. Instead, logicians study certain excrescences of thought: verbalized bits of reasoning—arguments.

An **argument** is a sequence of declarative sentences, one of which is intended as a **conclusion;** the remaining sentences, the **premises,** are intended to prove or at least provide some evidence for the conclusion. The premises and conclusion express **propositions**—which may be true or false—as opposed to questions, commands, or exclamations. Nondeclarative sentences may sometimes suggest premises or conclusions, but they never *are* premises or conclusions.

Declarative sentences are not themselves propositions. Some theorists have held that propositions are assertions made by sentences in particular contexts; others, that they are the meanings of sentences or the thoughts sentences express. But it is generally agreed that between sentences and propositions there is an important difference. The sentence "I am a woman" uttered by me expresses a different proposition than the same sentence uttered by you. When I utter the sentence, the proposition I assert is false; if you are a woman, when you utter the

itself. Still, bad as it is, it's an argument; the author intended the first two propositions (sentences) to be taken as evidence for or proof of the third, and that's all that being an argument requires.

Let's now consider a good one. I'll begin with a claim. The claim is that in certain matters your will is not free. In fact there is one act you cannot initiate no matter how strong your will. The act is this: to criticize all and only those people who are un-self-critical. For example, consider yourself. Are you going to criticize yourself or not? If you do, then you will criticize someone who is self-critical (namely, you)—and so you're not criticizing only the un-self-critical. On the other hand, if you don't criticize yourself, then you fail to criticize someone who is un-self-critical (namely, you again)—and so you don't criticize all the un-self-critical. So either way you fail.[1]

Now consider your thoughts as you read the previous paragraph. (I assume you read it with comprehension; if not, now might be a good time to try again.) When I first made the claim, unless you had read this sort of thing before, you were probably puzzled. You wondered, among other things, what I was up to. At a certain point (or maybe not a certain point—maybe slowly), a light went on and you saw it. The dawning of that light is insight.

A good argument, when it works, gives you insight. It enables you to see why the conclusion is true—not "see" in a literal sense, of course, but "in your mind's eye." What was wrong with the bad argument given above was that it didn't yield any insight at all. It puzzled us and offered no resolution to our puzzlement.

Here I am talking about thought (insight, puzzlement, dawning lights, and so on), when I said just a few paragraphs back that we were going to talk about symbol systems. That's because I want to make vivid a certain contrast. There is much to be noticed about the experience—the phenomenology—of argumentation. But contemporary logicians try to explain as much as possible of what makes an argument good or bad without using mentalistic jargon, which they view with suspicion. They prefer to talk about symbols.

The previous argument showed us something about insight, but it's rather flashy for an introductory illustration; let's consider a more mundane and time-worn example:

All men are mortal.
Socrates is a man.
∴ Socrates is mortal.[2]

---

[1] The reasoning here is identical to the reasoning of Russell's barber paradox and to the core of the argument by which we will prove the halting problem unsolvable (see Section 10.5).

[2] The origin of this argument is a mystery to me. It appears in many logic textbooks, going way back in history, so presumably it has a classical source. The obvious source would be Aristotle, since Aristotle invented formal logic, but an Aristotle scholar assures me that this argument is nowhere to be found among the Philosopher's works.

sentence you assert a true proposition. Even if you are not a woman, the proposition you assert by uttering this sentence is different than the one I assert by uttering it; your proposition is about you, mine about me.

Logicians, however, tend in practice to ignore the differences between sentences and propositions, studying the former as if they were the latter. This practice presupposes that each argument we study is uttered in a fixed context (a given speaker in a given circumstance), since only relative to such a fixed context does each sentence in the argument express a unique proposition. To illustrate, consider the following argument:

All women are mortal.
I am a woman.
∴ I am mortal.

The symbol '∴' means "therefore" and is used to mark the conclusion. The speaker might be you now, or me at age 13, or Queen Victoria reflecting on her imminent death. It doesn't matter who the speaker is, but we do presuppose that there is a *single* speaker, not several, so that, for example, 'I' in the second premise refers to the same person as 'I' in the conclusion. We also keep fixed some other presuppositions about the context: that, for example, the speaker is consistently using the English language—not some Alice-in-Wonderland tongue in which familiar words have unfamiliar meanings—and that demonstrative words like 'this' or 'that' have clear and unambiguous reference.

Having by fiat frozen these aspects of context, we have obliterated the difference between sentences and propositions and can proceed to treat sentences as if they were the propositions they express. (In this book we shall sometimes use the word 'statement' to designate sentences whose context has thus been frozen.) In this way we shift our focus from such elusive entities as assertions, meanings, or thoughts, to sentences, which can be pinned down on paper and dissected neatly into discrete components.

Contemporary logic thus replaces thoughts, meanings, or acts with symbols—letters, words, phrases, and sentences. Whether that is an illuminating or useful strategy, you will be able to judge for yourself by the time you finish this book. But this much is undeniable: Logicians have learned a great deal about systems of symbols—and much that is astonishing, unexpected, or useful, as we shall see.

Our definition of 'argument' stipulated that an argument's premises must be *intended* to give evidence for the conclusion. But an argument need not actually give evidence. There are bad arguments as well as good ones. Consider this:

Humans are the only rational beings.
Rationality alone enables a being to make moral judgments.
∴ Only humans are ends-in-themselves.

Now this is an argument, but it's bad. (Of course, no famous Western philosopher would ever *really* have reasoned this way!) Intuitively, the reason it's bad is that we can't see what the capacity for moral judgment has to do with being an end-in-

itself. Still, bad as it is, it's an argument; the author intended the first two propositions (sentences) to be taken as evidence for or proof of the third, and that's all that being an argument requires.

Let's now consider a good one. I'll begin with a claim. The claim is that in certain matters your will is not free. In fact there is one act you cannot initiate no matter how strong your will. The act is this: to criticize all and only those people who are un-self-critical. For example, consider yourself. Are you going to criticize yourself or not? If you do, then you will criticize someone who is self-critical (namely, you)—and so you're not criticizing only the un-self-critical. On the other hand, if you don't criticize yourself, then you fail to criticize someone who is un-self-critical (namely, you again)—and so you don't criticize all the un-self-critical. So either way you fail.[1]

Now consider your thoughts as you read the previous paragraph. (I assume you read it with comprehension; if not, now might be a good time to try again.) When I first made the claim, unless you had read this sort of thing before, you were probably puzzled. You wondered, among other things, what I was up to. At a certain point (or maybe not a certain point—maybe slowly), a light went on and you saw it. The dawning of that light is insight.

A good argument, when it works, gives you insight. It enables you to see why the conclusion is true—not "see" in a literal sense, of course, but "in your mind's eye." What was wrong with the bad argument given above was that it didn't yield any insight at all. It puzzled us and offered no resolution to our puzzlement.

Here I am talking about thought (insight, puzzlement, dawning lights, and so on), when I said just a few paragraphs back that we were going to talk about symbol systems. That's because I want to make vivid a certain contrast. There is much to be noticed about the experience—the phenomenology—of argumentation. But contemporary logicians try to explain as much as possible of what makes an argument good or bad without using mentalistic jargon, which they view with suspicion. They prefer to talk about symbols.

The previous argument showed us something about insight, but it's rather flashy for an introductory illustration; let's consider a more mundane and time-worn example:

All men are mortal.
Socrates is a man.
∴ Socrates is mortal.[2]

---

[1] The reasoning here is identical to the reasoning of Russell's barber paradox and to the core of the argument by which we will prove the halting problem unsolvable (see Section 10.5).

[2] The origin of this argument is a mystery to me. It appears in many logic textbooks, going way back in history, so presumably it has a classical source. The obvious source would be Aristotle, since Aristotle invented formal logic, but an Aristotle scholar assures me that this argument is nowhere to be found among the Philosopher's works.

This is good too, if not so good as our last example. It could, I suppose, convey some insight to a sheltered three-year-old. Its virtues, according to hoary tradition, are these:

1. Its premises are true.
2. Its reasoning is valid.

Now obviously, these virtues don't by themselves add up to a prize-winning argument. There are other things we'd like—such as significance, substance, relevance to some larger context—but the two listed above *are* virtues. Arguments that lack them are not likely to convey insight into true conclusions. So they are as good a place as any to start if we want to understand what makes an argument good.

Virtue 1, however, is the business of just about everybody but the logician. To tell whether or not a given premise is true (except for logically true or logically false propositions, cases to which we will later return), we must turn to science, conscience, or common sense—not to logic.

1.2 VALIDITY AND COUNTEREXAMPLES

That leaves us with virtue 2, the one that generally interests logicians. Most logicians have belonged to a school of thought known as the **classical tradition.** In the first four parts of this book we will consider logic from the classical perspective, though in the fifth we shall step outside of it. To say that an argument is **valid** is, according to the classical tradition, to say that there is no way for the conclusion not to be true while the premises are true. We'll sometimes put this in terms of "possible situations": There is no possible situation in which the premises are true but the conclusion isn't.

The Socrates argument is valid, for there is no possible situation in which all men are mortal, Socrates is a man, and Socrates is not mortal; we can't even coherently think such a thing.

The end-in-itself argument is **invalid** (i.e., not valid), for there is a possible situation in which the premises are true and the conclusion isn't. That is, it is possible that humans are the only rational beings and that rationality alone enables a being to make moral judgments but that humans are not the only ends-in-themselves. One way this is possible is if being an end-in-itself has nothing to do with the ability to make moral judgments, but rather is linked to some more general capacity, such as sentience or the ability to live and flourish. Thus perhaps other critters are also ends-in-themselves even if the argument's premises are true.

A possible situation in which an argument's premises are true and its conclusion is not true is called a **counterexample** to the argument. We may define validity more briefly simply by saying that a valid argument is one without a counterexample.

When we speak of possible situations, the term 'possible' is to be understood in a very broad sense. To be possible, a situation need not be something we can

bring about; it doesn't even have to obey the laws of physics. It just has to be something we can **coherently conceive**—that is, it has to be thinkable and describable without self-contradiction.

Thus, intuitively,[3] to tell whether or not an argument is valid, we try to conceive or imagine a possible situation in which its premises are true and conclusion is untrue. If we succeed (i.e., if we can describe a counterexample), the argument is invalid. If we fail, then either we have not been imaginative enough or the argument is valid. This makes logicians nervous; they'd like to have a test that doesn't rely on human ingenuity; much of this book will be devoted to explaining what they do about this anxiety and how their efforts fare.

But most people are not so skittish. We appeal to counterexamples almost unconsciously in everyday life. Consider this mundane argument:

> They said on the radio that it's going to be a beautiful day today.
> ∴ It *is* going to be beautiful today.

One natural (albeit cynical) reply is, "They could be wrong." This reply demonstrates the invalidity of the argument by describing a counterexample—that is, a possible situation in which the conclusion ('It's going to be a beautiful day today') is untrue even though the premise ('They said so on the radio') is true: namely, the situation in which the forecasters are wrong.

A counterexample need not be an actual situation, though it might; it is enough that the situation be conceptually possible. Thus it need not be *true* that the forecasters are wrong; to see the invalidity of the argument, we need only realize that it is *possible* they are wrong.

To give a counterexample, then, is merely to tell a kind of story. The story needn't be true, but it must be conceptually coherent. The cynical respondent to our argument above hints at such a story with the remark "They could be wrong."

That's enough for casual conversation. But for logical analysis it's useful to be more explicit. A well-stated description of a counterexample should contain three elements:

1. Affirmations of all the argument's premises.
2. A denial of the argument's conclusion.
3. An explanation of how this can be—that is, how the conclusion can still be untrue while the premises are all true.

If we flesh out the cynic's counterexample to make all of these elements explicit, the result might be something like this:

> They said on the radio that it's going to be a beautiful day today. But they are wrong. A cold front is moving in unexpectedly and will bring rain instead of a beautiful day.

---

[3] When I say 'intuitively', I mean from an informal point of view. We are still talking about thoughts here, not symbols. This is typical of informal logic. The formal, symbolic approach begins with the next chapter.

All three elements are now present. The first sentence of this "story" affirms the premise. The second denies the conclusion. The third explains how the conclusion could be untrue even though the premise is true.

This is not, of course, the only possible situation that would make the premises but not the conclusion true. I made up the idea of an unexpected cold front more or less arbitrarily. There are other counterexamples as well. Maybe an unexpected *warm* front will bring rain. Or maybe there will be an unexpected dust storm. Or maybe the radio announcer knew it was going to be an awful day and flat out lied. Each of these scenarios is a counterexample. This is typical; invalid arguments usually have indefinitely many counterexamples, each of which is by itself sufficient to show that the argument is invalid.

Let's consider another example. Is the following argument valid or invalid?

> All philosophers are freethinkers.
> Al is not a philosopher.
> ∴ Al is not a freethinker.

To answer, we try to imagine a counterexample. Is there a way for the conclusion not to be true while the premises are true? (To say that the conclusion is not true, of course, is to say that Al *is* a freethinker.) A moment's thought should reveal that this is quite possible. Here's one counterexample:

> All philosophers are freethinkers and Al is not a philosopher, but Al is nevertheless a freethinker, because there are some freethinking bricklayers who are not philosophers, and Al is one of these.

Again all three elements of a well-described counterexample are present. The statement 'All philosophers are freethinkers and Al is not a philosopher' affirms both of the premises. The statement 'Al is nevertheless a freethinker' denies the conclusion, and the remainder of the story explains how this can be so. The story is perfectly coherent, and thus it shows us how the conclusion could be untrue even if the premises were true.

Notice again that the counterexample need not be an actual situation. It's just a story, a scenario, a fiction. In fact, it isn't true that all philosophers are freethinkers, and maybe it isn't true that Al (whoever Al is) is a freethinker, either. That doesn't matter; our story still provides a counterexample, and it shows that the argument is invalid, by showing how it *could be* that the conclusion is untrue while the premises are true.

Notice, further, that we needn't have said that Al is a bricklayer; for purposes of the example, he could have been an anarcho-communist or some other species of freethinker—or an unspecified kind of freethinker. The details are flexible; what counts, however we formulate the details, is that our "story" is coherent and that it makes the premises true and the conclusion untrue.

Let's consider another argument:

> All philosophers are freethinkers.
> Al is a philosopher.
> ∴ Al is a freethinker.

This has no counterexample. If we affirm the premises, then we cannot without lapsing into incoherence deny the conclusion. If all philosophers are freethinkers and Al is one of the philosophers, then he must be a freethinker. This argument is valid.

That, of course, doesn't mean it's a good argument in all respects. On the contrary, some philosophers are dogmatically religious, so the first premise is false, which makes the argument unconvincing. But still the reasoning is valid.

Sometimes what appears to be a counterexample turns out on closer examination not to be. Unless the mistake is trivial (e.g., the story fails to make all the premises true or fails to make the conclusion untrue), the problem is often that the alleged counterexample is subtly incoherent and hence impossible. To return to the argument about Socrates, suppose someone said

> The argument is invalid because we can envision a situation in which all men are mortal and Socrates is a man, but Socrates is nevertheless immortal because he has an immortal soul.

This story does seem to make the premises of the argument true and the conclusion false. But is it really intelligible? If having an immortal soul makes one immortal and the man Socrates has an immortal soul, then not all men are mortal. The story is incoherent; it contradicts itself. It is therefore not a genuine counterexample, since a counterexample is a *possible* situation; that is, its description must be conceptually coherent.

Some additional invalid arguments with accompanying counterexamples are listed below. Keep in mind that invalid arguments generally have many counterexamples so that the counterexamples presented here are not the only ones. Note also that each counterexample contains all three elements (though sometimes more than one element may be expressed by the same sentence). The three elements, once again, are

1. Affirmations of all the argument's premises.
2. A denial of the argument's conclusion.
3. An explanation of how this can be—that is, how the conclusion can be untrue while the premises are all true.

In each case, the counterexample is a logically coherent story (not an argument) that shows how the conclusion could be untrue while the premises are true, thus proving that the argument is invalid. Notice how each of the counterexamples below performs this function:

Invalid Argument

Sandy is not a man.
∴ Sandy is a woman.

Counterexample

Sandy is neither a man nor a woman but a hamster.

Invalid Argument

If the TV is unplugged, it doesn't work.
The TV is not working.
∴ It's unplugged.

Counterexample

If the TV is unplugged it doesn't work, and it's not working. However, it is plugged in. The reason it's not working is that there's a short in the circuitry.

Invalid Argument

All charged particles have mass.
Neutrons are particles that have mass.
∴ Neutrons are charged particles.

Counterexample

All charged particles have mass, but so do some uncharged particles, including neutrons.

Invalid Argument

The winning ticket is number 540.
Beth holds ticket number 539.
∴ Beth does not hold the winning ticket.

Counterexample

The winning ticket is number 540; Beth is holding both ticket 539 and ticket 540.

Invalid Argument

There is nobody in this room taller than Amy.
Bill is in this room.
∴ Bill is shorter than Amy.

Counterexample

Bill and Amy are the only ones in this room, and they are the same height.

Invalid Argument

Sally does not believe that Eve ate the apple.
∴ Sally believes that Eve did not eat the apple.

Counterexample

Sally has no opinion about the story of Eve. She doesn't believe that Eve ate the apple, but she doesn't disbelieve it either.

Invalid Argument

Some people smoke cigars.
Some people smoke pipes.
∴ Some people smoke both cigars and pipes.

Counterexample

There are pipe-smokers and cigar-smokers, but nobody smokes both pipes and cigars, so the two groups don't have any members in common.

Invalid Argument

Some people smoke cigars.
∴ Some people do not smoke cigars.

Counterexample

There are people, and all of them smoke cigars. (If everybody does, then some people do and so the premise is true!)

Invalid Argument

We need to raise some money for our club.
Having a bake sale would raise money.
∴ We should have a bake sale.

Counterexample

We need to raise money for the club, and having a bake sale would raise money, but so would other kinds of events, like holding a car wash or a telethon. Some of these alternative fund-raising ideas better suit the needs of the club and the abilities of its members, and so they are what should be done instead of a bake sale.

Invalid Argument

Kate hit me first.
∴ I had to hit her back.

Counterexample

Kate hit the (obviously immature) arguer first. But the arguer could have turned the other cheek or simply walked away; there was no need to hit back.

Let's take stock. What launched our discussion of counterexamples was talk of validity, and what led us to validity was a look at the two virtues of a good argument, namely:

1. The premises are true.
2. The reasoning is valid.

Logicians sometimes suggest that these two virtues are sufficient for a good argument. I have already expressed doubts about this. But we can see why someone might believe it if we consider the two virtues together. To say that the reasoning is valid is to say that there is no counterexample—that is, there is no way for the conclusion not to be true while the premises are true. Now, if we add virtue 1—namely, that the premises *are* true—we see that the two virtues together add up to a guarantee of the truth of the conclusion. An argument that has both virtues—true premises and valid reasoning—is said to be **sound.** Sound reasoning certifies that its conclusion is true.

If that's all we want from reasoning, then virtues 1 and 2 are all we need. In the classical logical tradition, it has been customary to ask for no more. But I think we generally want more. We want insight, significance, cogency . . . well, at least we want **relevance.** Virtues 1 and 2 don't even give us that—as we shall see in the next section.

Exercise 1.2

Classify the following arguments as valid or invalid. For those that are invalid, describe a counterexample, making sure that your description includes all three elements of a well-described counterexample. Take each argument as it stands; that is, don't alter the problem by, for example, adding premises.

1. No plants are sentient.
   All morally considerable things are sentient.
   ∴ No plants are morally considerable.
2. All mathematical truths are knowable.
   All mathematical truths are eternal.
   ∴ All that is knowable is eternal.
3. Most geniuses have been close to madness.
   Blake was a genius.
   ∴ Blake was close to madness.

4. Most of the sentences in this book are true.
   Most of the sentences in this book are about logic.
   ∴ There are true sentences about logic in this book.
5. A high gasoline tax is the most effective way to reduce the trade deficit.
   We need to reduce the trade deficit.
   ∴ We need a high gasoline tax.
6. Some angels are fallen.
   ∴ Some angels are not fallen.
7. To know something is to be certain of it.
   We cannot be certain of anything.
   ∴ We cannot know anything.
8. The surface area of China is smaller than the surface area of Russia.
   ∴ The surface area of Russia is larger than the surface area of China.
9. Some men are mortal.
   ∴ Some mortals are men.
10. The witnesses said that either one or two shots were fired at the victim.
    Two bullets were found in the victim's body.
    ∴ Two shots were fired at the victim.
11. People do climb Mount Everest without oxygen tanks.
    ∴ It is possible to climb Mount Everest without oxygen tanks.
12. Some fools are greedy.
    Some fools are lecherous.
    ∴ There are some fools who are both lecherous and greedy.
13. No one has ever lived for 200 years.
    ∴ No one ever will.
14. DNA contains the code of life.
    Life is sacred.
    ∴ It is wrong to manipulate DNA.
15. There are fewer than a billion people in the whole United States.
    New York is only a part of the United States.
    ∴ There aren't a billion people in New York.

1.3 RELEVANCE

Consider the following rather lyrical argument:

I've heard of Wartburg, Tennessee.
∴ There's no tree that's not a tree.

Pretty bad—but it has both of the virtues discussed in the previous section: The premise is true, and the reasoning is valid. Of course the conclusion doesn't *follow* from the premise. But that wasn't how we defined validity—following from the premises. We defined it as the absence of a counterexample. And there is no counterexample here.

The queerness resides in the conclusion: 'There's no tree that's not a tree'. This conclusion can't be untrue; it's true in any possible situation, no matter what the world is like. Hence there is no possible situation in which the premise is true and the conclusion is not (simply because, regardless of the premises, there is no possible situation in which the conclusion is not true). So the argument has no counterexample; it is valid. Further, since the premise is true, it is sound. Still, it is a dumb argument.

Not that it leads to an incorrect conclusion. The conclusion *can't* be untrue; it's true in all possible situations. So here as elsewhere, soundness guarantees truth. What's wrong with this argument is that the conclusion derives no support from the premise. The premise is irrelevant; the conclusion could stand on its own.

The conclusion is a **logical truth**, a statement true in all possible situations. And the argument is an illustration of the general rule that *any argument whose conclusion is logically true is automatically valid, no matter what the premises.* A logical truth must be true no matter what we assume; so it is a valid conclusion from anything.

But since this argument is bad nevertheless, we may infer that at least one additional virtue is required for good reasoning: relevance. But what is relevance? In recent years a whole field of logic, relevance logic, has emerged to attempt to answer this question. We shall consider it in some detail in Section 16.3. Unfortunately, there seem to be as many relevance logics as relevance logicians, so the discipline is in disarray. But the need for relevance is clear.

There is another kind of inference in which the need for relevance stands out starkly: an argument with inconsistent premises. A set of propositions (or a single proposition) is **inconsistent** if there is no possible situation in which they are all true (or in which it is true). The proposition

> There is a tree that's not a tree.

is inconsistent. So is this set of propositions:

> Albert is a pirate.
> Albert is not a pirate,

and the more complex set:

> He's either here or in Chicago.
> He's not here.
> He's not in Chicago.

Including any such inconsistent proposition or set of propositions among the premises of an argument makes the argument automatically valid. For example, the argument

> Albert is a pirate.
> Albert is not a pirate.
> ∴ Albert plays golf.

is valid. Since there is no possible situation in which both premises are true, there is no possible situation in which both premises are true and the conclusion is not

true; hence there is no counterexample. Any attempt to describe a situation in which both premises are true will result in incoherence. More generally, *any argument with inconsistent premises is valid.*

This sounds disastrous, but it isn't. It doesn't mean you can prove whatever you want just by assuming an inconsistency. To prove in the fullest sense means (at least) to have a sound argument—to reason validly from *true* premises. But inconsistent premises cannot all be true. So arguments with inconsistent premises never prove anything and are therefore harmless. Yet they are odd because, though valid, they may lack relevance.

Relevance logicians reject the classical tradition's definition of validity, arguing that validity should by definition imply relevance so that we can reject such perverse examples as those recently contemplated. This is an appealing idea. The problem comes, as I noted before, in the attempt to work out the details. No one has hit upon a relevance-preserving definition of validity that has gained widespread acceptance.

It might be useful, however, to take a stab at saying what relevance is. One criterion that might be taken as an indicator of relevance is this: *Any idea that occurs in the conclusion also occurs in at least one of the premises.* In other words, every idea in a conclusion must "come from" somewhere, that is, from one or more of the premises. Conclusions should be "summations" of the premises. They should consist of elements of the premises recombined in a way that, ideally, produces insight. Consider, for example, this typical deductive argument:

All courses numbered less than 400 are undergraduate courses.
No undergraduate course can be taken for graduate credit.
∴ No course numbered less than 400 can be taken for graduate credit.

The fundamental ideas in the conclusion are 'course numbered less than 400' and 'being taken for graduate credit'. The first of these ideas comes from the first premise and the second from the second. Each has its origin in a premise, and this accounts, at least in part, for the conclusion's relevance.

Notice that I did not list the terms 'no' and 'can' as expressing ideas. These terms represent logical relationships, and they belong to a class of words that we shall call **logical operators.** (Some authors call them **syncategorematic** terms.) Roughly, a word is a logical operator if it expresses, not a specific idea itself, but a way of modifying or combining ideas. Some common logical operators are 'all', 'some', 'most', 'no', 'not', 'if . . . then', 'or', 'unless', and 'and'. Thus, though a relevant conclusion may not introduce ideas not contained in the premises, it may use logical operators to recombine the premises' ideas in new ways. (Precisely which such combinations preserve relevance is one of the controversial issues in relevance logic.)

Contrast the previous relevant and valid argument with this argument, which is both fallacious and irrelevant:

Smoking is harmful.
∴ Smoking should be illegal.

There are two fundamental ideas in the conclusion—and perhaps a third. The two obvious ones are 'smoking' and the notion of being 'illegal'. Depending on how we count, we might also treat 'should' as an idea, though some logicians would consider it a logical operator. No matter. The inference is clearly irrelevant, because although the term 'smoking' in the conclusion has its origin in the premise, the term 'illegal' comes from nowhere—and that's a hallmark of irrelevance.

Notice that we could make the inference relevant by adding a premise connecting the idea of harm to the idea of illegality:

Anything that is harmful should be illegal.
Smoking is harmful.
∴ Smoking should be illegal.

The conclusion is now relevantly drawn. The terms 'illegal' and 'should' (if we want to count the latter as expressing an idea) come from the first premise, and the term 'smoking' comes from the second. We have also strengthened the reasoning; the argument is now valid. The added premise, however, is false. And so, though valid, the argument is unsound.

Arguments with logically true conclusions or inconsistent sets of premises provide the most glaring examples of validity (in the classical sense) without relevance. But logically true statements and inconsistent premise sets are relatively rare in actual reasoning. Most of the statements with which we reason are **contingent**—that is, true in some possible situations and false in others. When we reason with contingent statements, there is less dissonance between classical logic and relevance logic. Still, differences remain, as we shall see in Section 16.3.

Exercise 1.3

Classify the following arguments as valid or invalid, using the informal concept of validity. For those that are invalid, describe a counterexample. Then discuss whether or not the argument's premises are relevant to its conclusion.

1. Joe is a mathematician.
   ∴ Joe is a mathematician.
2. Joe is a mathematician.
   Joe is not a mathematician.
   ∴ Joe is weird.
3. Joe is a mathematician.
   ∴ Joe is not both a mathematician and not a mathematician.
4. Olaf has been vaporized into incandescent plasma.
   ∴ Olaf is dead.
5. All men are mortal.
   Socrates is Greek.
   Socrates is a man.
   ∴ Socrates is mortal.

1.4 ARGUMENT INDICATORS

We have defined an argument as a sequence of declarative sentences, one of which is *intended* as a conclusion that the others, the premises, are *intended* to support. In this section we consider the grammatical cues by which speakers of English communicate such intentions. The most important of these are **argument indicators,** words or phrases that signal the presence and communicate the structure of arguments. These fall into two classes: premise indicators and conclusion indicators. A **premise indicator** is an expression such as 'for', 'since', and 'because' that connects two statements, signifying that the one to which it is immediately attached is a premise from which the other is inferred as a conclusion. So, for example, in the sentence

> The soul is indestructible because it is indivisible.

the premise indicator 'because' signals that the statement 'it is indivisible' (where 'it' refers to the soul) is a premise supporting the conclusion 'the soul is indestructible'. Premise indicators can also occur at the beginnings of sentences, but the rule still holds: The statement to which the premise indicator is attached is the premise; the other is the conclusion. Hence, for example, in the sentence

> Since numbers are nonphysical, nonphysical objects exist.

the word 'since' shows that the statement 'numbers are nonphysical' is a premise leading to the conclusion 'nonphysical objects exist'.

Conclusion indicators are words or phrases that signify that the statement to which they are attached is a conclusion that follows from previously stated premises. English is rich in conclusion indicators. Some of the most common are 'therefore', 'thus', 'so', 'hence', 'then', 'it follows that', 'in conclusion', 'accordingly', and 'consequently'. In the following argument, for example, 'hence' indicates that the third statement, 'God exists', is a conclusion from the first two:

> Without God, there can be no morality. Yet morality exists. *Hence* God exists.

But the same thing can be signaled by a premise indicator:

> God exists, *for* without God there can be no morality, and morality exists.

or by a mix of premise and conclusion indicators:

> Without God there can be no morality. *Then* God exists, *since* morality exists.

These are three different expressions of the same argument. There are many others. Notice that the conclusion (in this case, 'God exists') may occur at the end, or at the beginning, or in the middle of the argument, depending on the arrangement of argument indicators. All three positions are common in ordinary speech and writing. But for logical analysis it is customary to list the premises first and

the conclusion, prefixed by '∴', last, as we have been doing. This is called **standard form.**

Arguments may also be stated without indicators, in which case we must rely on subtler clues of context, intonation, or order to discern their structure. Most often when argument indicators are lacking the conclusion is given first, followed by the premises. Here is an example:

> There is no truth without thought. Truth is a correspondence between thought and reality. And a correspondence between two things cannot exist unless the things themselves exist.

Here the first statement is a conclusion from the remaining two.

Like most English words, many of the terms we use for argument indicators have more than one meaning. So not every occurrence of 'since', 'because', 'thus', and so on is an argument indicator. If someone says, "I got down on my hands and knees and *thus* I escaped beneath the smoke," it is unlikely that she is offering an argument. 'Thus' here means "in this way," not "it follows that." The speaker is not attempting to prove that she escaped. Similarly, in the sentence '*Since* the summer began, there hasn't been a drop of rain', the word 'since' indicates temporal duration, not a logical relationship between premise and conclusion. Neither sentence is an argument.

Sometimes arguments are not completely stated. A premise may be omitted because it is so obvious that it need not be stated (or, more sinisterly, because the arguer is trying to get listeners to take it for granted without thinking). Likewise, a conclusion may be omitted because it is very obvious, or because the arguer wants listeners to draw it for themselves and thus perhaps be more inclined to accept it. This argument, for example, has an implicit premise:

> The moon has no atmosphere and therefore cannot support life.

The unstated premise is, of course, that an atmosphere is needed to support life. (Notice also that the conclusion is only partly stated; its subject, having been mentioned already in the premise, is not repeated.) The argument, stated in full, is

> An atmosphere is needed to support life.
> The moon has no atmosphere.
> ∴ The moon cannot support life.

Here is an argument with an implicit conclusion:

> Ailanthus trees have smooth bark, but the bark of this tree is rough.

The full argument is

> Ailanthus trees have smooth bark.
> The bark of this tree is rough.
> ∴ This tree is not an ailanthus tree.

Arguments are sometimes confused with conditional statements. A **conditional statement** is an assertion that one thing is the case if another thing is, for example:

If three is an even number, then it is divisible by two.

A conditional asserts neither of its components. This statement, for example, asserts neither that three is even nor that it is divisible by two, but the latter is the case *if* the former is. In this way the statement differs significantly from the following argument, which is formed from the same components:

Since three is an even number, it is divisible by two.

In an argument, both the premises and the conclusion are categorically asserted. A person who utters these words is saying (absurdly) that three is even and that three is divisible by two.

The point here is that 'if' and certain related terms, such as 'unless' and 'only if', are *not* premise indicators. Instead, they form compounds that function as single statements. We shall have more to say about them in the next chapter.

Exercise 1.4

Some of the following passages are arguments, some are not. Some of the arguments are incomplete, lacking a premise, or a conclusion, or both. Rewrite each argument in standard form, supplying implicit premises or conclusions. For those passages that are not arguments, write 'not an argument'.

1. Uranium is heavier than iron, because gold is heavier than iron and uranium is heavier than gold.
2. Since anyone under 18 is a juvenile and juveniles are not allowed on the premises, Sally is not allowed on the premises.
3. If there is a storm warning, the siren sounds. So there is no storm warning, since the siren is not sounding.
4. Savage could not have been the thief. The thief was over six feet tall. But Savage is only 5'8".
5. We went to Indianapolis; then we went to Chicago.
6. The water froze, and when water freezes the temperature must be at or below zero degrees Celsius.
7. Different cultures have different conceptions of rationality. Hence rationality itself takes many forms, for what a culture *conceives* as rational *is* rational for that culture.
8. Alice has a National Rifle Association sticker on her windshield. It is likely, therefore, that she opposes gun control.
9. Because all things other than pleasure are valued only for the pleasure they produce, but pleasure is valued for its own sake, only pleasure is intrinsically valuable. For a thing is intrinsically valuable if and only if it is valued for its own sake.
10. I lied because I was afraid you would hate me if I told the truth.

1.5 USE AND MENTION

Before we move on to formal logic, it will be useful to explain a convention used throughout this text. Since contemporary logic studies systems of symbols, logicians must often talk about specific symbols or strings of symbols. To do this, we need names for these symbols or strings. We shall form names for symbols or strings by enclosing them in single quotation marks.

If we take an ordinary name, for example, and flank it with single quotation marks, the result is a new name—one that names the old name. Thus, for example, the following sentence is true:

'Smog' is a four-letter word.

But this is false:

Smog is a four-letter word.

Smog is not a word at all; it is a form of air pollution. In the first sentence the word 'smog' is mentioned but not used; in the second it is used but not mentioned. In logic we usually mention specific symbols by using their quotation names in the manner of the first sentence. This is why single quotation marks have appeared and will continue to appear so frequently in this book.

Failure to observe the use/mention distinction can lead to confusion or nonsense. Consider

The King refers to Elvis.

Which king? And why would he want to do that? But, of course, what is intended is

'The King' refers to Elvis.

That is, the phrase 'The King' or, more completely, 'The King of Rock and Roll' is used to refer to the man Elvis Presley. The confusion arises from a failure to indicate that the phrase 'The King' is merely being mentioned, not used.

Here's an example that is purely nonsensical:

Contains four words does not contain four words

—at least until appropriate quotation marks are added, when it becomes this simple and obvious truth:

'Contains four words' does not contain four words.

Indeed not; it contains only three.

Quotation marks are likewise needed for mentioning letters, numerals, and other symbols. Thus we may (correctly) write

'10' is a numeral that names the number 10.

Numerals are symbols. They can be written or printed. The numeral '10', for example, consists of a vertical stroke followed by a circle. But numbers are some-

thing else again. They are not shapes or marks and cannot themselves be written—though they may be named. The names of the number ten are legion. They include not only the Arabic numeral '10' but also the Roman numeral 'X', the formula '8 + 2', the English word 'ten', and so on. Ten is not any of these things, but the unique thing that they all name.

To summarize, when you see single quotation marks, look between them: The word or phrase, symbol or formula that you see written there is, precisely as written, the thing being mentioned. Where there are no quotation marks, the words in the sentence are being *used*, and you generally have to look elsewhere to find what is being *mentioned*.

When we are using one language to study another, the one we use is called the **metalanguage** and the one being studied—i.e., mentioned—is called the **object language.** For example, when native speakers of English study Hebrew, they usually converse about Hebrew grammar, style, wording, and so on in English. Here English is the metalanguage and Hebrew is the object language. In the succeeding chapters, we shall study various logical languages. Each, as we study it in turn, will become our object language; but the metalanguage will always be English. We will, however, from time to time import exotica, such as Greek letters for variables, or some notation from mathematics into our metalanguage, so it will be a specialized or technical form of English.

Exercise 1.5

One or more of the following sentences is true as it stands; others are not true unless quotation marks are added. Supply quotation marks where necessary to make them true.

1. Logic begins with an L.
2. The numbers 2 and -2 are both solutions to the equation $x^2 = 4$.
3. The argument some dogs are collies therefore some collies are dogs is valid.
4. The number four has many names.
5. Instead of the word therefore we may write the symbol $\therefore$.

PART II

CLASSICAL PROPOSITIONAL LOGIC

CHAPTER 2

CLASSICAL PROPOSITIONAL LOGIC: SYNTAX

An argument form is a pattern of reasoning common to many arguments. Studying forms enables us to understand whole classes of arguments at once. In this chapter we introduce a symbolic language capable of exhibiting simple argument forms: the language of propositional logic. We investigate its **syntax** (grammar) in this chapter and its **semantics** (meaning-structures) in the next in order to develop mathematically rigorous methods to check for validity and related properties.

2.1 ARGUMENT FORMS

In Chapter 1 we approached logic from an **informal** point of view, considering arguments as they occur in natural language. Here we begin the study of **formal logic,** whose subject matter is **argument forms**—patterns of reasoning shared by many different arguments. Here is a simple argument form:

If P, then Q
P
∴ Q

This form is known by its medieval Latin name, ***modus ponens.***[1] The letters 'P' and 'Q' function as place-holders for declarative sentences. We shall call such letters **sentence letters.** We say that an argument is an **instance** of a form comprised of sentence letters (or, simply, that it *has* that form) if it is obtainable from the form by replacing the sentence letters with sentences, each occurrence of the same

[1] This is an abbreviation of the longer term '*modus ponendo ponens*', which, loosely translated, means the mode of proving an assertion by assuming an assertion (the nonconditional premise).

letter being replaced by the same sentence.[2] The following argument, for example, is an instance of the form modus ponens in which 'P' is replaced by 'the fetus is a person' and 'Q' by 'abortion is murder':

> If the fetus is a person, then abortion is murder.
> The fetus is a person.
> ∴ Abortion is murder.

Since the number of declarative sentences is potentially infinite, the form represents infinitely many different arguments, all with the same structure. Another example of the form modus ponens is

> If Clara won't get the raise, then she'll quit.
> She won't get the raise.
> ∴ She'll quit.

What is significant about this form is that any argument that has it is valid. Since there are infinitely many such arguments, to know that they all are valid is to possess knowledge that is in a sense infinite. But can we really *know* that each instance of modus ponens is valid?

One possible approach to such knowledge is to insert random sentences in place of 'P' and 'Q' and check the resulting arguments for validity by trying to formulate counterexamples. But this case-by-case approach, though it might ultimately convince us, is logically inconclusive. Even in an entire human lifetime we could check only a finite number of instances; there will always be infinitely many that we have never examined. Maybe some of these, so strange and complex that we would never think to check them, are *in*valid. Maybe even some of the ones we have checked are invalid, and failures of imagination have prevented us from noticing!

A more sophisticated approach is to move to a more abstract level of thought. We might, for example, reason this way: No matter which sentences 'P' and 'Q' stand for, whenever 'if P, then Q' is true and 'P' is true, then 'Q' has to be true as well. Here we focus on the general pattern of the reasoning, on the form, rather than the form's instances. In this way we might be able to "see" that the form itself guarantees the validity of its instances. But this sort of "seeing," or intuition, is still fallible and hence still subject to doubt.

---

[2] Notice, however, that this definition does not rule out replacing different letters with the same sentences. Just as in the mathematical equation '$x + y = y + x$' we may legitimately replace both 'x' and 'y' by '2' to obtain the instance '$2 + 2 = 2 + 2$', so in the form modus ponens, for example, we might replace both 'P' and 'Q' by the same sentence, say 'People have souls'. The result

> If people have souls, then people have souls.
> People have souls.
> ∴ People have souls.

is, like '$2 + 2 = 2 + 2$', trivial and uninteresting. But, like all instances of modus ponens, it is valid.

These doubts can be *wholly* dispelled, but not without a deeper understanding of the grammatical structure (**syntax**) and meaning (**semantics**) of argument forms. This understanding will enable us to see why some arguments are valid and others not, and it will yield rigorous techniques for settling questions of validity and related issues.

We begin with some syntactic fundamentals. First, the order of the premises and minor variations in wording that do not alter meaning are irrelevant to an argument's form. Thus with regard to the previous argument, for example, we may omit the 'then', reverse the premises, and adjust the wording a bit without altering the meaning. The result

Clara won't get the raise.
If she won't get the raise, she'll quit.
∴ She'll quit.

still counts as an instance of modus ponens.

However, the order of components within a sentence *does* matter. To illustrate this, it will be useful to introduce some terminology. An "if . . . then" statement, such as 'if she won't get the raise, she'll quit', is called a **conditional statement,** or often just a **conditional.** The sentence following the 'if' is the **antecedent,** and the sentence following the 'then' (or simply the remainder if the 'then' is missing) is the **consequent** ('consequen*t*' with a 't', not consequen*ce*). If we exchange the antecedent and the consequent within the conditional, as in this argument:

If she quits, Clara won't get the raise.
She won't get the raise.
∴ She'll quit.

the result is no longer modus ponens. Instead, this argument has the form

If Q, then P
P
∴ Q

Because the second premise 'P' affirms the consequent of the conditional premise 'If Q, then P', this form is called **affirming the consequent.** (For parallel reasons, modus ponens is sometimes called **affirming the antecedent.**)

Moreover, the argument itself, unlike the two previous arguments, is *in*valid. Here is a counterexample: The boss is a scrooge; Clara won't get the raise, period, whether she quits or not. But she won't quit, because she needs the job to feed her kids.

At least one instance of affirming the consequent, then, is invalid. A form like modus ponens all of whose instances are valid is called a **valid form.** Any form that, like affirming the consequent, has at least one invalid instance is called an **invalid form.**

But not every instance of an invalid form need be invalid. Here is a valid instance of affirming the consequent:

If some men are saints, then some saints are men.
Some saints are men.
∴ Some men are saints.

This is valid, however, not *because* it is an instance of affirming the consequent, but rather because the second premise and the conclusion say essentially the same thing. The conclusion therefore follows from the second premise alone. The first premise is superfluous.

Because invalid forms may have valid instances, knowing that a form is invalid tells us little about the validity of a particular instance of that form. We may safely assume that having that form does not make the instance valid, but it may be valid nevertheless in virtue of structure not represented in the form (such as the equivalence of the second premise and conclusion in the previous example).

In algebra, the equation '$x + y = y + x$' has precisely the same meaning as the equation '$z + w = w + z$'. Likewise in logic, the use of different sentence letters makes no difference to the form. We could also write modus ponens, for example, as

If R, then S
R
∴ S

Similarly, we might express affirming the consequent as

If P, then Q
Q
∴ P

Hence, although exchanging letters within the conditional alone changed modus ponens into affirming the consequent, consistent replacement of letters throughout a form does not alter the form. The preceding form, for example, results from our original version of affirming the consequent by replacing 'P' by 'Q' and 'Q' by 'P' everywhere they occur. Thus it still counts as affirming the consequent.

We need to be cautious, however, if we obliterate distinctions between letters. If in the preceding form, we replace 'P' by 'Q', and make no other changes, we get

If Q, then Q
Q
∴ Q

This form is still affirming the consequent, but it is also affirming the antecedent. We might call it "affirming both the consequent and the antecedent"!

For brevity, logicians have invented symbols to represent logically significant terms, that is, **logical operators.**[3] We shall represent 'if . . . then', for example, by the symbol '→'. So instead of 'if P then Q', from now on we may write 'P → Q'. To save space, we often write an entire form on one line. In this format we shall

[3] This is not, of course, an exact definition, but as we shall see later (especially in Section 9.4), the exact definition of 'logical operator' is a matter of dispute.

use the symbol '⊢', often called the **turnstile,** instead of '∴', and separate the premises with commas. Thus modus ponens, for example, is written as 'P → Q, P ⊢ Q'. An argument form written in this format is called a **sequent.**

'If . . . then' is not the only logical operator. In this chapter we consider four others. These correspond roughly to the English expressions 'it is not the case that', 'and', 'or', and 'if and only if', which we represent respectively by the symbols '~', '&', '∨', and '↔'.[4] (Often 'and' is accompanied by the word 'both' and 'or' by the word 'either'; these additions are usually for clarity or emphasis and do not affect the logical meaning.) What is common to all five operators is that they apply to propositions to produce more complex propositions. That is why the form of logic considered in this chapter is known as **propositional logic.** In later chapters we will consider operators that apply to entities of other types.

The operators 'if . . . then', 'and', 'or', and 'if and only if' are called **binary** or **dyadic operators,** because they combine two statements into a new statement. We may, for example, use the operator 'and' to combine the separate statements 'it is Wednesday' and 'it is hot' into a new statement 'it is Wednesday and it is hot'. The operator 'it is not the case that', by contrast, applies to only one proposition at a time and hence is **monadic** or **unary.** We may, for example, affix it to the sentence 'it is Wednesday' to form the new sentence 'it is not the case that it is Wednesday'.

The operator expressed by the symbol '~' is called the **negation operator,** and a proposition resulting from its application is called a **negative proposition** or **negation.** The symbol '&' is the **conjunction operator;** it combines two propositions, called **conjuncts,** into a **conjunction.** Similarly, '∨' is the **disjunction operator;** it combines two propositions, called **disjuncts,** into a **disjunction.** The **biconditional operator** '↔' is the least familiar of the five. It combines two propositions, which we shall call **constituents,** into a **biconditional.** A biconditional asserts the equivalence of its components in the sense that if either one is true, so is the other.

These five operators can be combined with sentence letters to produce an infinity of argument forms. The central aim of formal propositional logic is to establish methods for deciding which of these forms are valid and which are not.

As a first step toward this goal, we might attempt to evaluate forms informally, by the method of counterexamples explained in Section 1.2. That is, given a sequent, we might by trial and error attempt to produce invalid instances—instances for which there is a possible situation that makes the premises true but

---

[4] Just as mathematicians use both '·' and '×' to represent multiplication, so logicians sometimes use other symbols to represent these operators:

Logical Operator	Alternative Symbol(s)
It is not the case that	¬
And	• ∧
Or	(none)
If . . . then	⊃
If and only if	≡

the conclusion untrue. The hope would be that either we find an invalid instance, thus showing the sequent to be invalid, or we fail to find an invalid instance, but as a result of our search become familiar enough with the sequent to see that it is valid.

Consider, for example, the sequent '$P \vee Q \vdash P \leftrightarrow Q$'. To test its validity informally, we consider instances, more or less at random. Suppose we take this instance:

Either it is a skunk or it is a badger.
∴ It is a skunk if and only if it is a badger.

Now it is not difficult to formulate a counterexample. Consider a possible situation in which the animal referred to by the word 'it' is a skunk but not a badger. Then the premise is certainly true, but the conclusion is false. For the conclusion asserts that if it's a skunk it's also a badger, and vice versa, but in the situation we are envisioning (which is perfectly possible) it is a skunk but not a badger.

If we try to find an invalid instance of the sequent '$P \rightarrow Q, \sim Q \vdash \sim P$', by contrast, we meet with repeated failure. (This sequent, incidentally, is called ***modus tollens***—i.e., mode of denying, or **denying the consequent.**) Consider this instance:

If you press the accelerator, the engine speeds up.
It is not the case that the engine speeds up.
∴ Therefore, you are not pressing the accelerator.

We might at first attempt a counterexample along these lines: Maybe the engine is malfunctioning or simply turned off so that even though you are pressing the accelerator it is not speeding up. This, of course, is a possible situation. But it is not a counterexample, because it is a situation in which the first premise is false. Or maybe the engine is not speeding up, though whenever you press the accelerator it does speed up. But then you are certainly not pressing the accelerator. This, once again, is a possible situation, but it is not a counterexample because it is a situation in which the conclusion is true.

By repeated failures to find a counterexample, both with this instance and with other instances of modus tollens, we might eventually gain confidence in the validity of the form itself and maybe even see why it is valid. This method, however, is intuitive and imprecise. It relies heavily on inventiveness and the powers of imagination. And since for all of us these powers are limited, it does not guarantee a correct answer—or any answer at all. There are better methods, as we shall soon see.

Exercise 2.1.1

Check the following forms for validity informally by attempting to construct an instance that has an obvious counterexample. If you can do so, write out the instance and describe the counterexample that shows it to be invalid. If not, or if you see that the argument is valid, simply write 'valid' for that form.

1. $P \rightarrow Q, \sim P \vdash \sim Q$

2. $P \vdash P \,\&\, Q$
3. $P \,\&\, Q \vdash P$
4. $P \vee Q \vdash P$
5. $P \vdash P \vee Q$
6. $P \rightarrow Q \vdash Q \rightarrow P$
7. $P \rightarrow Q \vdash P \rightarrow \sim\sim Q$
8. $P \leftrightarrow Q \vdash Q \leftrightarrow P$
9. $P \leftrightarrow Q \vdash P \,\&\, Q$
10. $P, \sim P \vdash Q$

Exercise 2.1.2

Given that modus ponens is also called affirming the antecedent and modus tollens is also called denying the consequent, what is the name of the sequent in problem 1 of Exercise 2.1.1?

2.2 FORMALIZATION

In this section we present the syntax (grammar) of the language of propositional logic. Fundamental to an understanding of syntax is the notion of the *scope* of a logical operator. The **scope** of a particular occurrence of an operator consists of that occurrence of the operator itself, together with whatever it is operating on. Consider, for example, the following pair of English sentences:

> It is not the case that boron is both a compound and an element.
>
> Boron is a compound, and it is not the case that it is an element.

In the first sentence—which, incidentally, is true—the negation operator applies to the entire conjunction 'boron is both a compound and an element', or, more explicitly, 'boron is a compound and boron is an element'. Thus the scope of the negation operator is the entire sentence. In the second sentence, which is false, the negation operator applies only to the subsentence 'it [boron] is an element'. Its scope is thus only the second conjunct of the second sentence—that is, the subsentence 'it is not the case that it [boron] is an element'.

In representing the forms of these two sentences in the language of propositional logic, we need some conventions for indicating scope. For this purpose, we borrow from algebra the idea of using brackets as punctuation. Using brackets, we can represent the form of the first of the two sentences above as '~(C & E)'. The second is then symbolized as 'C & ~E'.

In algebra, the negative sign '−' is presumed to apply just to the term it immediately prefixes, unless brackets are used to extend its scope. Thus the expression '−3 + 5' represents the number 2, because '−' applies just to the numeral '3'. But in the expression '−(3 + 5)', the '−' applies to '(3 + 5)' so that this expression represents the number −8. We use brackets similarly in logic. The

negation sign is presumed to apply to whatever formula it immediately prefixes, unless we extend its scope with brackets.

Brackets are also needed to determine scope when two or more binary operators occur in the same formula. Suppose you receive the following announcement in the mail:

> You have won ten thousand dollars and a Caribbean cruise or a dinner for two.

You might well be puzzled, for this announcement is ambiguous. Using 'T' for 'you have won ten thousand dollars', 'C' for 'you have won a Caribbean cruise', and 'D' for 'you have won a dinner for two', we may symbolize the sentence either as 'T & (C ∨ D)' or as '(T & C) ∨ D'. There is a big difference. If the first formula represents what is meant, you have won ten thousand dollars, plus a cruise or a dinner. If (as is most likely) the second formula represents what is meant, then, even if the announcement is true, you probably have won only a dinner.

This sort of multiplicity of meaning is called **scope ambiguity.** In the first formula, the scope of the conjunction operator is the whole formula and the scope of the disjunction operator is just the second conjunct. In the second formula, the scope of the disjunction operator is the whole formula and the scope of the conjunction operator is just the first disjunct. Without brackets, the scopes of the two operators are indeterminate and, as with the English sentence, it is not clear what is meant.

Because one of the purposes of propositional logic is to clarify thought, its grammatical rules prohibit expressions such as 'T & C ∨ D', which are ambiguous in just the way the contest announcement is, because of the absence of brackets. To prevent such ambiguities, each binary operator in a grammatical formula must be accompanied by a pair of brackets that indicate its scope. There is only one exception: We may omit a pair of brackets that surround everything else in a formula, since brackets in this position are not needed to prevent ambiguity. Thus, instead of '(T & (C ∨ D))', which is strictly correct, having a pair of brackets for each of the formula's two binary operators, we may, if we like, write 'T & (C ∨ D)', as we did in the preceding example, dropping the outermost brackets, the ones that indicate the scope of the '&'.

Negation requires no brackets of its own. We can negate any part of the formula '(C ∨ D)', for example, simply by appropriate placement of the negation operator. The possible locations for a single negation operator are as follows: '~(C ∨ D)', '(~C ∨ D)', '(C ∨ ~D)'. The brackets that come with the '∨' (which we have kept here in the second and third formulas, even though we could have omitted them), together with the placement of '~' suffice to define the scopes of both '~' and '∨'. Even when we iterate negations, as in '~~P' ("it is not the case that it is not the case that P"), no brackets are needed. Because '~' applies to the smallest whole formula to its right, the scope of the leftmost occurrence of '~' is the whole formula and the scope of the rightmost occurrence of '~' is '~P'.

All formulas of propositional logic contain sentence letters as their ultimate constituents. Thus sentence letters are called **atomic** formulas, and, by analogy, formulas consisting of more than just a single sentence letter are called **complex,** or **molecular,** formulas.

Recall that the scope of an occurrence of a logical operator is that occurrence of the operator together with all the parts of the formula to which it applies. More precisely, it is the smallest formula containing that occurrence of the operator (which often is only a part of a larger formula). Each molecular formula has one and only one operator whose scope is that entire formula. This operator is called the formula's **main operator,** and it defines the formula's fundamental form. The main operator of the formula '(P & (Q ∨ R))', for example, is '&'. The formula as a whole is thus a conjunction, though its second conjunct is a disjunction. In the formula '–(P → (Q ∨ R))' the main operator is '–'. This formula is therefore negative; more specifically, it is the negation of a conditional whose consequent is a disjunction. Recognition of the main operator in a formula is crucial in constructing semantic trees or planning proof strategies, procedures that are discussed in the next two chapters.

The task of representing the forms of arguments in propositional logic is complicated by the fact that there are many ways of expressing the logical operators in natural language. English has, for example, many ways of expressing negation. Usually, of course, instead of saying 'it is not the case that', we simply append 'not' to the sentence's verb. 'It is not the case that I am going' sounds better as 'I am not going'. But we use the more awkward wording to emphasize that from a logical point of view negation is an operation that applies to a whole sentence, not just to a verb.

Prefixes such as 'non-', 'im-', 'in-', 'un-', 'a-', 'ir-', and so on may also express negation. But not always. 'He is incompetent' is arguably synonymous with 'it is not the case that he is competent', but 'gasoline is inflammable' does not mean the same thing as 'it is not the case that gasoline is flammable'! Likewise, 'she uncovered the dough' does not mean the same thing as 'it is not the case that she covered the dough'. Negation is not the only kind of opposition.

To determine whether we are dealing with true negation or some other form of opposition, we must ask whether what we are dealing with can be adequately expressed by the phrase 'it is not the case that'. If so, it is negation. If not, it is something else.

Conjunction may be expressed not only by the terms 'and' or 'both . . . and', but also by 'but', 'nevertheless', 'furthermore', 'moreover', 'yet', 'still', and so on. These terms connote differing nuances of contrast or connection, yet like 'and' they all perform the logical operation of linking two sentences into a compound sentence that affirms them both. Even a semicolon between two sentences may express conjunction in English.

English sentences with compound subjects or predicates are usually treated in propositional logic as conjunctions of two complete sentences. Hence we think of the sentence 'Sal and Jeff were here' as abbreviating 'Sal was here and Jeff was here' and the sentence 'Sal danced and sang' as abbreviating 'Sal danced and Sal sang'.

Often where two sentences are linked by a word that expresses conjunction, we may question whether to treat them as a conjunction or as separate sentences. From a logical point of view it makes little difference. The argument 'He's big and he's mean, so he's dangerous' is equally well rendered into propositional logic as 'B & M ⊢ D' or as 'B, M ⊢ D'. Neither sequent is valid.

Conditionals also have important variants. They can, for example, be presented in reverse order, provided that the antecedent remains attached to the 'if'. The statement 'if it rains, it pours', for example, can also be expressed as 'it pours if it rains'. The form is in each case the same: 'R → P'. In either order, the antecedent is always the clause prefixed by 'if'.

There is one exception. Where 'if' is preceded by the term 'only', what it prefixes is the consequent, not the antecedent. The following four sentences, for example, all assert the same (true) conditional proposition:

If you are pregnant, then you are female.

You are female if you are pregnant.

Only if you are female are you pregnant.

You are pregnant only if you are female.

In each case, 'you are pregnant' is the antecedent and 'you are female' the consequent. The form of all four is the same: 'P → F'. If we reverse antecedents and consequents, we get four sentences of the form 'F → P'. These, too, all affirm the same proposition, but it is a different proposition from that affirmed by the first group of four, a proposition that is (fortunately) not in all cases true.

If you are female, then you are pregnant.

You are pregnant if you are female.

Only if you are pregnant are you female.

You are female only if you are pregnant.

Many people find it difficult to keep the meanings of 'if' and 'only if' distinct. Keep in mind that 'if' always prefixes antecedents and 'only if' always prefixes consequents, and you should have no trouble.

Given these remarks about 'if' and 'only if', it ought to be clear that the biconditional operator 'if and only if' may be understood as a conjunction of two conditionals, one expressed by 'if', the other by 'only if'. Hence 'P ↔ Q' just means '(P → Q) & (Q → P)'. We could therefore dispense with the symbol '↔' and treat all biconditionals as conjunctions of two conditionals in this way. But we retain '↔', partly in deference to tradition, partly because it saves writing.

Apart from the optional addition of 'either', the term 'or' has few variants in English. We may regard it, however, as a component of the important term 'neither . . . nor'. Etymologically, this is a contraction of 'not either . . . or'; it thus expresses negated disjunction. The sentence 'it will neither snow nor rain', for example, may be symbolized as '~(S ∨ R)'. It is also acceptable to symbolize this statement as '~S & ~R', which is logically equivalent to '~(S ∨ R)', though this symbolization has the disadvantage of failing to reflect the English etymology.

The term 'unless' may be thought of as expressing another two-operator combination, a conditional with a negated antecedent. 'We will starve unless we eat' says the same thing as 'if we do not eat, we will starve'. We may thus symbolize

the sentence as '~E → S'. Alternatively, 'unless' may be understood simply as expressing disjunction, in which case it prefixes the first disjunct. So we may also symbolize 'we will starve unless we eat' as 'E ∨ S'. These two symbolizations are equally correct.

Exercise 2.2.1

Formalize each of the sentences below, using the following interpretation scheme:

P — the peasants revolt
Q — the queen hesitates
R — the revolution will succeed
S — the slaves revolt

1. Either the peasants will revolt or the slaves will revolt.
2. Both the peasants and the slaves will revolt.
3. The peasants and the slaves will not both revolt.
4. If the peasants revolt, then the revolution will not succeed.
5. The peasants revolt if and only if they don't fail to revolt.
6. Only if the peasants revolt will the slaves revolt.
7. The revolution will succeed only if the queen hesitates.
8. If the peasants revolt and the queen hesitates, the revolution will succeed.
9. If the peasants revolt, then the revolution will succeed if the queen hesitates.
10. The revolution will not succeed unless the queen hesitates.
11. The peasants will revolt whether or not the queen hesitates.
12. The revolution will succeed if the slaves and the peasants both revolt.
13. If either the peasants or the slaves revolt and the queen hesitates, then the revolution will succeed.
14. If the peasants revolt but the slaves don't, the revolution will not succeed, and if both the peasants and the slaves revolt, the revolution will succeed.
15. If the peasants revolt if and only if the slaves revolt, then neither will revolt.

Exercise 2.2.2

Use premise and conclusion indicators to determine the premises and conclusions of the following arguments, then symbolize them in the formal notation of propositional logic using the sentence letters whose interpretation is specified below. (The forms of all of these arguments, incidentally, are valid in classical logic.)

Sentence Letter	Interpretation
B	Descartes believes that he thinks
E	Descartes exists
K_1	Descartes knows that he thinks
K_2	Descartes knows that he exists
J	Descartes is justified in believing he thinks
T	Descartes thinks

1. If Descartes thinks, then he exists; for he doesn't both think and not exist.

2. If Descartes thinks, then he exists. Hence he does not think, because he does not exist.
3. Descartes is justified in believing that he thinks if he knows that he thinks. But he is not justified in believing that he thinks, so he does not know that he thinks.
4. If Descartes knows that he thinks, then he exists. For if he knows that he thinks, then he thinks; and if he thinks, then he exists.
5. Descartes does not exist. For either he knows that he exists or he doesn't exist; and he doesn't know that he exists.
6. Descartes believes that he thinks. If he does not think, he does not believe that he thinks. Therefore Descartes thinks.
7. If Descartes thinks, then he knows that he exists, and if he knows that he exists, then he exists. Therefore, if Descartes thinks, then he both knows that he exists and really does exist.
8. If Descartes does not exist, then he doesn't think; so if he thinks, it is not the case that he does not exist.
9. Descartes neither exists nor does not exist. Therefore Descartes thinks.
10. Descartes knows that he thinks if and only if (1) he believes that he thinks, (2) he is justified in believing that he thinks, and (3) he does in fact think. Therefore, if Descartes does not think, then he does not know that he thinks.

2.3 FORMATION RULES

The formulas of propositional logic have a grammar, and that grammar (or syntax) may be precisely articulated as **formation rules.** Formation rules define what counts as a formula by giving general directions for assembling formulas out of simple symbols, or characters. They are the rules of grammar for a formal language. In order to state the formation rules for propositional logic, we need first to define the **character set** for propositional logic—that is, the alphabet and punctuation marks from which the formulas of its language are constructed. We stipulate that a **character** for the language of propositional logic is anything belonging to one of the following four sets:

Sentence letters:	Capital letters from the English alphabet
Numerals:	0 1 2 3 4 5 6 7 8 9
Logical operators:	~ & ∨ → ↔
Brackets:	( )

The only novelty here is the numerals. These are used to form subscripts for sentence letters when we want to use the same letter for two different sentences and need a means to keep the letters distinct. Moreover, without subscripts we could symbolize no more noncompound sentences than we have capital letters—that is, twenty-six. And though we are unlikely in practice to need more than

twenty-six letters at once, a system of logic should not be subject to such arbitrary restrictions.

With these ideas in mind, we are ready to state the **formation rules**—the rules of grammar for the language of propositional logic; they define the notion of a grammatical formula by telling how to construct such formulas, starting with sentence letters, and combining them with the operators and brackets.

Formation Rules for Propositional Logic

1. Any sentence letter, with or without a sequence of numerals as a subscript, is a formula.
2. If Φ is a formula, then so is $\neg\Phi$.
3. If Φ and Ψ are formulas, then so are $(\Phi \,\&\, \Psi)$, $(\Phi \vee \Psi)$, $(\Phi \rightarrow \Psi)$ and $(\Phi \leftrightarrow \Psi)$.

Anything that is not a formula by finitely many applications of these rules is not a formula. Notice that in stating the formation rules, we use Greek letters (which belong to the metalanguage (see Section 1.5), *not* to the language of propositional logic). They are variables that stand for formulas of propositional logic. The Greek indicates generality. For example, 'Φ' and 'Ψ' in rule 3 stand for *any* formulas, no matter how simple or complex. When they are combined with operators and brackets into a complex expression, this expression stands for any formula obtainable by replacing the Greek letters with formulas. Thus, for example, the expression '$(\Phi \,\&\, \Psi)$' stands for '$(P \,\&\, Q)$', '$(\neg R \,\&\, S)$', '$((P \vee R) \,\&\, (Q \rightarrow \neg\neg S))$', and so on. Use of English letters here would be inappropriate, since they would too easily be confused with individual expressions of the object language.[5] In contrast to such expressions as '$(P \,\&\, Q)$', expressions containing Greek letters, such as '$(\Phi \,\&\, \Psi)$', are not formulas. Rather, they are metalinguistic devices used for referring to whole classes of formulas.

Repeated (recursive) application of the formation rules enables us to construct a great variety of formulas. So, for example, 'P' and 'Q' are formulas by rule 1. Hence by rule 3, '$(P \vee Q)$' is a formula. Now by rule 1 again 'R' is a formula, from which it follows by rule 2 that '$\neg R$' is a formula and again by rule 2 that '$\neg\neg R$' is a formula. Hence, since both '$(P \vee Q)$' and '$\neg\neg R$' are formulas, by rule 3 '$((P \vee Q) \rightarrow \neg\neg R)$' is a formula. And since this is a formula, by rule 2 again, '$\neg((P \vee Q) \rightarrow \neg\neg R)$' is also a formula. And so on! In this way we can build up formulas as complex as we like.

---

[5] To see this more clearly, suppose that instead of rule 2 we wrote:

2′ If 'P' is a formula, then so is '$\neg$P'.

Then the rule would tell us only how to generate this one formula '$\neg$P'. It would not tell us how to generate '$\neg\neg$P' or '$\neg$Q'. If, by contrast, we put the rule this way:

2″ If P is a formula, then so is $\neg$P

we would be mixing the object language and the metalanguage confusingly. It's not clear what this means. The Greek says exactly what we want while avoiding these problems.

Notice that the only formation rule that introduces brackets is rule 3. This means that the only legitimate function of a pair of brackets is to delineate the scope of some binary operator. In particular, brackets are not used to indicate the scopes of either sentence letters or the negation operator. Thus, for example, none of the following expressions count as formulas:

(P) ~(P) (~P) ~(~P) **(All wrong!)**

Exercise 2.3

Some of the following expressions are formulas of propositional logic. Others are not. For those that are, explain how they are built up by the formation rules. For those that aren't, explain why they can't be built up by the formation rules.

1. $(P) \vee (Q)$
2. $(P \,\&\, Q)$
3. $P \,\&\, Q$
4. $\sim\sim P$
5. $\sim(P \vee (Q \,\&\, S))$
6. $(\Phi \rightarrow \Psi)$
7. $((P \,\&\, Q) \vee (R \,\&\, S))$
8. P
9. $(P \rightarrow P)$
10. $(P \,\&\, Q \,\&\, R)$

CHAPTER 3

CLASSICAL PROPOSITIONAL LOGIC: SEMANTICS

3.1 TRUTH CONDITIONS

In this chapter we examine *semantics* of classical propositional logic. **Semantics** is the study of meaning. The logical meaning of an expression is usually understood as its contribution to the truth or falsity of sentences in which it occurs. By rigorously characterizing the meanings (in this sense) of the logical operators, we deepen our understanding of validity and related concepts.

Logicians have traditionally defined meaning in terms of possible truth. To know the meaning of a sentence, they have assumed, is to know which possible situations or circumstances make it true and which make it false. For example, if we wished to check a student's understanding of the sentence 'The government is an oligarchy', we might describe various possible political arrangements, asking each time whether the government described was an oligarchy. The pattern of the student's responses to these scenarios would quickly reveal whether she knows what 'The government is an oligarchy' means.

Or, to take a more sophisticated example, philosophers sometimes debate what is meant by such sentences as 'James knows that God exists'. To clarify their understanding, they ask whether or not the sentence would be true in various possible situations. Suppose, for example, that James has been brought up from earliest childhood to believe in God. Would that make it true that he knows God

exists? Suppose he has had a mystical vision in which it seemed to him that God gave him a message. Would that make it true? Suppose that he has had such a vision and that God really did give him the message. The point of these queries is to clarify the meaning of the sentence 'James knows that God exists'—or, more broadly, to clarify the meaning of the predicate 'knows' in application to religious assertions. And the general assumption of the inquiry is that to know the meaning of a sentence or term is to know which possible situations make that sentence true or that term truly applicable.

This assumption is often expressed by saying that the meaning of a term is its **truth conditions.** The truth conditions for a term are rules that specify the possible situations in which sentences containing that term are true and the possible situations in which sentences containing that term are false.

In this section we give a truth-conditional semantics for the five logical operators introduced in Chapter 2. That is, we explain their meanings in terms of the possible situations in which sentences containing them are true and the possible situations in which sentences containing them are false.

In doing so, we shall employ the concept of *truth value*. A **truth value** is a kind of semantic quantity that characterizes propositions. For now, we assume that there are only two truth values: **true,** or **T,** and **false,** or **F.** A true proposition has the value T and a false proposition the value F. Moreover, we assume that in each possible situation each proposition has one, and only one, of these truth values. This assumption is called the **principle of bivalence.** Logics based on the principle of bivalence and the assumption that meaning is truth conditions are called **classical.** The dominant logics in Western thought have been classical.

Some philosophers have held that classical logic is universally the best form of logic, or even the only true logic. This book dissents from that view. In Part V we shall explore reasons for thinking that the principle of bivalence, though appropriate for some applications of logic, is less appropriate for others. We shall consider truth values other than T and F and the possibility that sentences may have more than one truth value, or none at all. And in Section 16.2, we question even the idea that meaning has anything to do with truth. There we explore a semantics that defines the meanings of terms, not as their truth conditions, but as their **assertibility conditions**—the conditions under which statements containing these terms are confirmable by adequate evidence. And beyond that we shall glimpse still more radical ways of departing from the classical tradition. Each of these novel semantic assumptions alters our conception of what valid reasoning is. For now, however, we present the semantics of the logical operators in the classical way, as bivalent truth conditions.

We begin with classical logic for two reasons. First, it is highly established in the Western logical tradition—our tradition. Second, it is, from a semantic viewpoint at least, the simplest logic, and it is best to start with what is simple.

Our immediate task, then, is to define the meanings of the five logical operators in terms of their truth conditions. We begin with the conjunction operator. If we conjoin two sentences—say, 'it is Wednesday' and 'it is hot'—we obtain a single sentence ('it is Wednesday and it is hot') that is true if and only if both

original sentences were true, and false otherwise. Hence the truth conditions for conjunction may be stated as follows:

> The truth value of a conjunction is T in a given situation iff the truth value of each of its conjuncts is T in that situation.

and

> The truth value of a conjunction is F in a given situation iff one or both of its conjuncts does not have the value T in that situation.

(The term 'iff' is a commonly used abbreviation for 'if and only if'.) To understand these truth conditions is, by the lights of classical logic, to understand what conjunction means.

All of this is well and good if we aim to state the truth conditions for conjunction when applied to statements of natural language. But we have taken a step of abstraction into formal logic. We are no longer concerned primarily with sentences, like 'it's Wednesday and it's hot', but with formulas, like 'W & H'. Simple sentences have been replaced with sentence letters. But a sentence letter, such as 'W', has in itself no meaning and is neither true nor false. Of course we can *give* it a meaning by associating it with a particular statement of natural language. But this we do differently in different contexts. For one problem 'W' may mean "it's Wednesday," for another "Water is H_2O." So what can it mean to talk about a situation that makes a mere formula like 'W & H' true?

Two things are needed to make talk about possible situations intelligible in formal propositional logic. The first is an interpretation of the sentence letters. **Interpretations** are given by associating sentence letters with statements of natural language. The interpretation of a sentence letter may vary from problem to problem, but within a given problem we keep the interpretation fixed. Thus we may stipulate, for example, that (for the duration of this example) 'W' means "it's Wednesday" and 'H' means "it's hot." Let us now, in fact, stipulate this. The second thing we need to make sense of the notion of a possible situation is the concept of a valuation:

> DEFINITION A **valuation** of a formula or set of formulas of propositional logic is an assignment of one and only one of the truth values T and F to each of the sentence letters occurring in that formula or in any formula of that set.

For a formula, such as 'W & H', that contains two sentence letters, there are four valuations, as shown in the following table:

W	H
T	T
T	F
F	T
F	F

That is, both 'W' and 'H' might be true, 'W' might be true and 'H' false, 'W' might be false and 'H' true, and both 'W' and 'H' might be false. Given an interpretation of the sentence letters, each valuation defines a situation. For example, given the interpretation stipulated above, the valuation that assigns T to both 'W' and 'H' defines a possible situation in which it is both Wednesday and hot. Similarly, the valuation that assigns T to 'W' and F to 'H' defines a possible situation in which it is Wednesday but not hot, and so on.

Stipulation of the interpretation, however, though obviously essential for applying logic to natural language, is inessential from a purely formal point of view. To state truth conditions for the logical operators, we need only to say how the truth values of sentences containing them depend on the truth values of their components, not what the components themselves have been interpreted to mean. Hence truth conditions for formal logic need concern themselves only with valuations, not with interpretations. A valuation alone is not a possible situation, but merely a pattern of truth values—the empty form, as it were, of a possible situation. It has the advantage, however, of being an entity definable with mathematical precision. If we disregard particular interpretations of sentence letters and think of abstract valuations rather than possible situations, we enter a realm of formal thought where everything is sharply defined and clear. The truth conditions for conjunction now look like this:

> The truth value of a conjunction is T on a given valuation iff the truth value of each of its conjuncts is T on that valuation.

and

> The truth value of a conjunction is F on a given valuation iff one or both of its conjuncts does not have the value T on that valuation.

Because these more abstract truth conditions are stated in terms of valuations, they are often referred to as **valuation rules.**

Valuation rules will appear so often from now on that it will be useful to abbreviate them. We shall use the script letter '$\mathcal{V}$' to stand for valuations, and, as in the previous section, Greek capital letters will stand for formulas. Instead of the cumbersome phrase 'the value assigned to Φ by $\mathcal{V}$', we shall write '$\mathcal{V}(\Phi)$'. Thus, to say that $\mathcal{V}$ assigns the value T to Φ, we write simply '$\mathcal{V}(\Phi) = T$'. Using this notation, we may state the valuation rules for conjunction more compactly as follows:

$\mathcal{V}(\Phi \,\&\, \Psi) = T$ iff both $\mathcal{V}(\Phi) = T$ and $\mathcal{V}(\Psi) = T$.
$\mathcal{V}(\Phi \,\&\, \Psi) = F$ iff either $\mathcal{V}(\Phi) \neq T$ or $\mathcal{V}(\Psi) \neq T$, or both.[1]

---

[1] We could also state the second rule this way:

$\mathcal{V}(\Phi \,\&\, \Psi) = F$ iff either $\mathcal{V}(\Phi) = F$ or $\mathcal{V}(\Psi) = F$, or both.

But then a question might arise regarding the truth value of '$\Phi \,\&\, \Psi$' if somehow Φ or Ψ lacked truth value or had some value other than T or F. Defining the falsity of the conjunction in terms of the untruth, rather than the falsity, of its components makes conjunctions bivalent even if their components are not. Bi-

The same idea may be expressed in tabular form, listing the four possible combinations of truth value for Φ and Ψ on the left and the resulting truth value for Φ & Ψ on the right:

Φ	Ψ	Φ & Ψ
T	T	T
T	F	F
F	T	F
F	F	F

This is called a **truth table.** Truth tables are, perhaps, easier to read than valuation rules. But, unlike the rules, they have the disadvantage of not being generalizable to more advanced forms of logic. Rules will prove more useful in the long run, which is why we emphasize them here.

Let's now examine the truth conditions for the negation operator. If we attach it to a sentence—say, 'It's snowing'—we get a negated sentence: for example, 'It is not the case that it's snowing'. If the sentence is true, its negation is false. If the sentence is false, its negation is true. This is vividly apparent when negation is iterated. The sentence 'It is not the case that it is not the case that it's snowing', for example, is just an elaborate way of saying 'It's snowing'; any situation in which one sentence is true is a situation in which the other is true, and in any situation in which one sentence is false, the other is false as well. Two negations "cancel out," producing a statement with the same meaning as the original. By the same principle, three negations have the same effect as one, four likewise cancel out, and so on. Negation, then, is simply an operation that inverts truth value. Hence the truth conditions for negation may be stated precisely as follows:

$\mathcal{V}(\sim\Phi) = \text{T}$ iff $\mathcal{V}(\Phi) \neq \text{T}$.
$\mathcal{V}(\sim\Phi) = \text{F}$ iff $\mathcal{V}(\Phi) = \text{T}$.

This, according to classical logic, is the meaning of negation. We can also represent these rules in a truth table. Since the negation operator is monadic, applying to a single formula rather than to two, there are only two cases to consider instead of four: the case in which that formula is true and the case in which it is false. The table shows that $\sim\Phi$ has the value listed in the right column when Φ has the value listed to the left:

---

valence is thus built into the valuation rules themselves. In fact, throughout this book I consistently define all semantic ideas in terms of truth and untruth, rather than truth and falsehood. This saves a good bit of trouble in the metatheoretic work of Chapter 5 and facilitates a smooth transition to nonclassical logics in Part V.

Often both valuation rules are stated together in a very compact fashion, as follows:

$\mathcal{V}(\Phi \,\&\, \Psi) = \text{T}$ iff both $\mathcal{V}(\Phi) = \text{T}$ and $\mathcal{V}(\Psi) = \text{T}$; otherwise, $\mathcal{V}(\Phi \,\&\, \Psi) = \text{F}$.

Our formulation says exactly this, but it is more explicit about what "otherwise" means.

Φ	$\neg\Phi$
T	F
F	T

Let's now consider the truth conditions for 'or'. 'Or' has two meanings. It can mean "either . . . or . . . and possibly both" or "either . . . or . . . and not both." The first meaning is called **inclusive disjunction** and the second **exclusive disjunction.** This ambiguity is unfortunate. Suppose, for example, that on a true–false quiz you find the statement

Four is either an even number or a square number.

What should you answer? Four is both even and square. So, you might argue, it is not *either* even or square—that is, not just one of these two things—it's *both*. In that case you would mark the statement false. On the other hand, you might think that since four is even (and also since it's square), it's true that it is even or square. In that case you would mark the statement true.

In neither case would you be wrong, and in neither case would you have misunderstood anything. But if you marked the statement false, that would mean you understood the 'or' exclusively, and if you marked it true, that would mean you understood it inclusively.

This problem might be less acute if we were speakers of Latin. In Latin there are two words for 'or': '*vel*' and '*aut*'. In most contexts, '*vel*' more naturally expresses the idea "either . . . or . . . and possibly both" (the inclusive sense), and '*aut*' tends to mean "either . . . or . . . and not both" (the exclusive sense of 'or'). In English we sometimes resolve the ambiguity by using the compound term 'and/or' for the inclusive sense. But both 'or' by itself and 'either . . . or' generally admit of both readings. When we use them, we or our listeners may not know exactly what we mean.

This situation would be intolerable in a formal logical language. Formal logic aims at precision. Its operators must have clear and unambiguous meanings. Therefore, when we introduce an operator like '$\vee$' we must stipulate precisely what it means. Logicians usually have found the inclusive sense of 'or' more useful, and so, by convention, that is the sense they have given to the operator '$\vee$'. In fact, '$\vee$' is just an abbreviation for '*vel*'.

Apart from cases in which both disjuncts are true (the cases on which the inclusive and exclusive senses of 'or' disagree), the truth conditions for 'or' are clear. If one disjunct is true and the other false (e.g., 'either the sun is a star or the moon is'), then the whole disjunction is true. And if both disjuncts are false (e.g., 'either the moon is a star or the earth is'), then the disjunction is false. Hence the valuation rules for the operator '$\vee$' are as follows:

$\mathcal{V}(\Phi \vee \Psi) = \text{T}$ iff either $\mathcal{V}(\Phi) = \text{T}$ or $\mathcal{V}(\Psi) = \text{T}$, or both.
$\mathcal{V}(\Phi \vee \Psi) = \text{F}$ iff both $\mathcal{V}(\Phi) \neq \text{T}$ and $\mathcal{V}(\Psi) \neq \text{T}$.

The corresponding truth table is:

Φ	Ψ	$\Phi \vee \Psi$
T	T	T
T	F	T
F	T	T
F	F	F

The logical operator '$\vee$', then, accurately symbolizes the English 'or' only when 'or' is used in the inclusive sense. In spite of this, we need not introduce a special symbol for exclusive disjunction, since 'P or Q', where 'or' is intended in the exclusive sense, may be symbolized in our notation as '$(P \vee Q) \,\&\, \sim(P \,\&\, Q)$'—that is, "either P or Q, but not both P and Q."

In formalizing arguments involving disjunction, we will for the sake of simplicity and consistency treat the disjunctions as inclusive, except when there is strong reason not to. But on those fairly frequent occasions when the meaning of 'or' is unclear, we should keep in mind that this policy is essentially arbitrary.

We now turn to the truth conditions for conditional statements. Under what conditions is $\Phi \rightarrow \Psi$ true? Let's consider the case in which the antecedent Φ is false (we assume nothing about the consequent Ψ). Now, though Φ is false, the conditional invites us to consider what is the case *if* Φ, hence to suppose Φ true. This, however, yields a contradiction, from which (as we saw in Section 1.3) any proposition validly follows. Take a specific instance: The statement 'Napoleon conquered Russia' is false. Given this, it is (in a certain sense) true, for example, that if Napoleon conquered Russia, then Caesar conquered the universe. Indeed, if Napoleon conquered Russia, then anything you like is true—because the fact is that Napoleon didn't conquer Russia. Thus it appears that when Φ is false, $\Phi \rightarrow \Psi$ is true, regardless of the truth value of Ψ.

Let us next consider the case in which the consequent Ψ is true. Then, whether or not Φ is true, Ψ is true (trivially). Take a specific instance: The statement 'Iron is a metal' is true. Then any conditional containing 'Iron is a metal' as its consequent is true. For example, 'If today is Tuesday, then iron is a metal' would be true—because whether or not it is Tuesday (i.e., regardless of the truth value of the antecedent), iron *is* a metal. Thus in general we may infer that $\Phi \rightarrow \Psi$ is true when Ψ is true, regardless of the truth value of Φ.

We have now concluded that $\Phi \rightarrow \Psi$ is true whenever Φ is false or whenever Ψ is true. Together these conclusions account for three of the four truth combinations for Φ and Ψ; that is, $\Phi \rightarrow \Psi$ is true whenever Φ and Ψ are both true, or Φ is false and Ψ is true, or Φ and Ψ are both false. The only remaining case is the one in which Φ is true and Ψ false. But in this case the conditional is clearly false. If, for example, it is Tuesday and the weather is not hot, then the conditional 'if it is Tuesday, then it is hot' is obviously false.

To summarize, we have concluded that $\Phi \rightarrow \Psi$ is true if Ψ is true (regardless of the truth value of Φ), $\Phi \rightarrow \Psi$ is also true if Φ is not true (regardless of the truth value of Ψ), and $\Phi \rightarrow \Psi$ is false if Φ is true and Ψ is not true. This covers all possible cases. Hence the truth conditions for '$\rightarrow$' are as follows:

$\mathcal{V}(\Phi \rightarrow \Psi) = \text{T}$ iff either $\mathcal{V}(\Phi) \neq \text{T}$ or $\mathcal{V}(\Psi) = \text{T}$, or both.
$\mathcal{V}(\Phi \rightarrow \Psi) = \text{F}$ iff both $\mathcal{V}(\Phi) = \text{T}$ and $\mathcal{V}(\Psi) \neq \text{T}$.

The corresponding truth table is

Φ	Ψ	$\Phi \rightarrow \Psi$
T	T	T
T	F	F
F	T	T
F	F	T

The conditional defined by these truth conditions is called the **material conditional.**

If you were unconvinced by the reasoning that led us to the truth conditions for the material conditional, you are not alone. Many logicians (your author among them) are troubled by this reasoning.

Consider, once again, the last line on the truth table, the case in which Φ and Ψ are both false. I argued that the conditional was true in that case, since its antecedent contradicts the facts, and from a contradiction anything follows. Now surely conditionals are *sometimes* true when their antecedents and consequents are both false. The statement

(A) If you are less than an inch tall, then you are less than a foot tall.

for example, is uncontroversially true, though (taking 'you' as referring to you) its antecedent and consequent are both false. But the antecedent and consequent of the following statement are also both false, and yet, unlike statement (A), this statement seems false:

(B) If you have no lungs, then you can breathe with your eyeballs.

If, as these examples suggest, English conditionals are sometimes true and sometimes false when their antecedents and consequents are both false, then the truth value of an English conditional must not be determined solely by the truth values of its components. Something else must figure into the truth conditions.

An operator which forms compounds whose truth value is strictly a function of the truth values of the components is said to be **truth-functional.** The symbols '&', '~', 'v', and the material conditional as defined by the truth conditions above are truth-functional. But we have seen evidence that suggests that 'if . . . then' is not a truth-functional operator and hence is not the material conditional.

Intuitively, what makes statement (A) true is not that its antecedent contradicts the facts, but that it is necessary, given that you are less than an inch tall, that you are also less than a foot tall. Correspondingly, what makes statement (B) false seems to be the lack of just such a necessary connection: It is not necessary, given that you have no lungs, that you can breathe with your eyeballs. This suggests that an English statement of the form 'if P then Q' is true if and only if such a necessary connection exists, regardless of the truth values of the components.

The truth conditions for the material conditional take into account only the truth values of the antecedent and consequent, not the presence or lack of such a necessary connection. This leads to anomalies not only in the case in which the antecedent and consequent are both false, but also in the case in which the antecedent is false and the consequent true. In that case a material conditional is true. But consider this statement:

> If there are no people, then people exist.

Once again, contrary to the truth conditions for the material conditional, this seems false, and once again, the necessary connection is lacking. This case, in fact, is especially anomalous, since here the antecedent does not merely fail to necessitate the consequent—it actually necessitates the negation of the consequent.

Further anomalies occur in the case in which the antecedent and consequent are both true. Consider this example:

> If the Mississippi contains more than a thimbleful of water, then it is the greatest river in North America.

The Mississippi contains considerably more than a thimbleful of water and it is the greatest river in North America, so both the antecedent and consequent are true. If it is a material conditional, it is therefore true. Yet many English speakers would say that this conditional is false. It seems false, once again, because it is not *necessary* given merely that the Mississippi contains more than a thimbleful of water that it is the greatest river in North America.

Thus English conditionals seem to be true only when there is a necessary connection between antecedent and consequent, whereas the truth conditions for material conditionals ignore all such connections, taking into account only the truth values of the antecedent and consequent.

What, then, of the reasoning by which I arrived at the truth conditions for the material conditional in the first place? It is sound—as applies to the material conditional, but not to English conditionals. I claimed, for example, that when the consequent Ψ of $\Phi \rightarrow \Psi$ is true, then whether or not Φ is true, Ψ is true. But this claim tacitly ignores the possibility that the truth or falsity of Φ might necessitate the falsity of Ψ so that (taking this necessary connection into account) it would be wrong to conclude that Ψ is true whether or not Φ is true. Thus I arrived at the truth conditions for the material conditional by implicitly assuming that such necessary connections do not affect the conditional's truth value.

A similar assumption underlies my reasoning in the case in which the antecedent is false. I claimed that when Φ is false, the supposition that Φ is true yields a contradiction, from which any proposition validly follows. Thus, given that Φ is false, if Φ then Ψ, that is, $\Phi \rightarrow \Psi$ is true—for any proposition Ψ. Thus I assumed that what determines the truth value of the conditional is simply the contradiction of its antecedent with the facts, rather than a necessary connection, or lack thereof, between the antecedent and consequent.

It was, therefore, by assuming that such necessary connections do not affect the truth value of the conditional that I arrived at the truth conditions for the material conditional. But this assumption seems false for at least some English

conditionals. The material conditional is therefore not just another way of writing the English 'if . . . then'.[2]

But then if our aim is to evaluate arguments, which we normally formulate in English, why bother with the material conditional?

Part of the answer is historical. Beginning with the work of the Scottish philosopher David Hume (1711–1776), many thinkers, among them some of the founders of contemporary logic, have doubted the intelligibility of this idea of necessary connection. As a result, many have found the material conditional (whose truth conditions, though odd, are at least exact) preferable to English conditionals, whose truth conditions seem bound up with the suspect notion of necessity. Indeed, logicians have long dreamed of an ideal language, free of all ambiguity, unclarity, and dubious metaphysics; and the replacement of the English conditional by the material conditional offered hope of progress toward that goal. Early in this century, Bertrand Russell, Ludwig Wittgenstein, and other prominent philosophers held that with the creation of such a language the perennial philosophical problems, which they regarded as linguistic confusions, would simply dissolve. There is indeed much to be said for the replacement of murkier notions by clearer ones, but in the end we may be left wondering whether we have really solved the problems or merely changed the subject.

In any case, these early thinkers had little choice but to embrace the material conditional. They needed some sort of conditional operator, and it was not until midcentury that logicians began to formulate rigorous and illuminating truth conditions involving ideas of necessary connection. Moreover, the material conditional does mimic English conditionals fairly well in many cases. Like English conditionals, it is always false when its antecedent is true and its consequent false; and in the other cases its truth value sometimes agrees and sometimes disagrees with that of English conditionals. Certainly, it offers the best approximation to English conditionals among truth-functional operators. Moreover, its truth conditions are simple and precise. For these reasons, the material conditional has become the standard conditional of logic and mathematics.

Lately, however, logicians have formulated a variety of alternative truth conditions that seem to reflect more adequately the meanings of English conditionals. We shall consider some of these in later chapters. Unfortunately, none of these alternatives has won universal acclaim as the true meaning of 'if . . . then'. That is why we still bother with the material conditional.

This having been said, however, it must be admitted that the common textbook practice of symbolizing 'if . . . then' in English as the material conditional (a practice in which this textbook too has indulged) is not wholly defensible. The material conditional is at best a rough approximation to 'if . . . then', and some patterns of reasoning valid for the one are not valid for the other. We shall be more

[2] That, at least, is my view. Some logicians still insist that English conditionals *are* material conditionals. They say unabashedly that statements like 'if you have no lungs, then you can breathe with your eyeballs' are true. We can see why they say this by understanding what they mean by 'if . . . then', but I do not think that that is what the rest of us usually mean by 'if . . . then'.

careful about the difference between the material conditional and English conditionals for the remainder of this chapter. When considering instances of argument forms containing the material conditional, we shall not translate the material conditional back into English as 'if . . . then', but instead retain the symbol '$\rightarrow$' as a reminder that its meaning lies solely in its truth conditions, not in what we normally mean by 'if . . . then'.

It remains to discuss the truth conditions for the biconditional operator '$\leftrightarrow$', which we have associated with the English expression 'if and only if'. Since, as we saw in Section 2.2, 'if' prefixes antecedents and 'only if' prefixes consequents, the statement form

Φ if Ψ

may be symbolized as $\Psi \rightarrow \Phi$, and the form

Φ only if Ψ

as $\Phi \rightarrow \Psi$. Thus the biconditional, as its name implies, can be understood as a pair of conditionals—more precisely, as a conjunction of two conditionals. As a conjunction, it is true if both conditionals are true, and it is false if either or both are untrue. Now if Φ and Ψ are either both true or both untrue, both conditionals are true (by the valuation rules for '$\rightarrow$'). But if Φ is true and Ψ untrue, then $\Phi \rightarrow \Psi$ is untrue; and if Φ is untrue and Ψ true, then $\Psi \rightarrow \Phi$ is untrue. In either of these cases, the biconditional is false. Thus the truth conditions for the biconditional are as follows:

$\mathcal{V}(\Phi \leftrightarrow \Psi) = \text{T}$ iff either $\mathcal{V}(\Phi) = \text{T}$ and $\mathcal{V}(\Psi) = \text{T}$, or $\mathcal{V}(\Phi) \neq \text{T}$ and $\mathcal{V}(\Psi) \neq \text{T}$.
$\mathcal{V}(\Phi \leftrightarrow \Psi) = \text{F}$ iff either $\mathcal{V}(\Phi) = \text{T}$ and $\mathcal{V}(\Psi) \neq \text{T}$, or $\mathcal{V}(\Phi) \neq \text{T}$ and $\mathcal{V}(\Psi) = \text{T}$.

These rules yield the following truth table:

Φ	Ψ	$\Phi \leftrightarrow \Psi$
T	T	T
T	F	F
F	T	F
F	F	T

The biconditional, in other words, is true if the two constituents have the same truth value and false if they differ in truth value.

Because the truth conditions for '$\leftrightarrow$' are just those for a conjunction of two material conditionals, '$\leftrightarrow$' is often called the **material biconditional** operator, and where $\Phi \leftrightarrow \Psi$ is true, Φ and Ψ are called **material equivalents.** Two formulas, then, are **materially equivalent** on a valuation if and only if they have the same truth value on that valuation.

The material biconditional shares with the material conditional the oddity of ignoring necessary connections between its components. If its components are either both true or both untrue a material biconditional is true, regardless of the existence or lack of existence of necessary or relevant connections between the

TABLE 3.1
Valuation Rules for Propositional Logic

For any formulas Φ and Ψ and any valuation $\mathcal{V}$:

1. $\mathcal{V}(\sim\Phi) = T$ iff $\mathcal{V}(\Phi) \neq T$;
   $\mathcal{V}(\sim\Phi) = F$ iff $\mathcal{V}(\Phi) = T$.

2. $\mathcal{V}(\Phi \,\&\, \Psi) = T$ iff both $\mathcal{V}(\Phi) = T$ and $\mathcal{V}(\Psi) = T$;
   $\mathcal{V}(\Phi \,\&\, \Psi) = F$ iff either $\mathcal{V}(\Phi) \neq T$ or $\mathcal{V}(\Psi) \neq T$, or both.

3. $\mathcal{V}(\Phi \vee \Psi) = T$ iff either $\mathcal{V}(\Phi) = T$ or $\mathcal{V}(\Psi) = T$, or both;
   $\mathcal{V}(\Phi \vee \Psi) = F$ iff both $\mathcal{V}(\Phi) \neq T$ and $\mathcal{V}(\Psi) \neq T$.

4. $\mathcal{V}(\Phi \rightarrow \Psi) = T$ iff either $\mathcal{V}(\Phi) \neq T$ or $\mathcal{V}(\Psi) = T$, or both;
   $\mathcal{V}(\Phi \rightarrow \Psi) = F$ iff both $\mathcal{V}(\Phi) = T$ and $\mathcal{V}(\Psi) \neq T$.

5. $\mathcal{V}(\Phi \leftrightarrow \Psi) = T$ iff either $\mathcal{V}(\Phi) = T$ and $\mathcal{V}(\Psi) = T$, or $\mathcal{V}(\Phi) \neq T$ and $\mathcal{V}(\Psi) \neq T$;
   $\mathcal{V}(\Phi \leftrightarrow \Psi) = F$ iff either $\mathcal{V}(\Phi) = T$ and $\mathcal{V}(\Psi) \neq T$, or $\mathcal{V}(\Phi) \neq T$ and $\mathcal{V}(\Psi) = T$.

components. Thus (importing '$\leftrightarrow$' into English) the following statements, however odd, are both true:

Life evolved on earth $\leftrightarrow$ Ronald Reagan was president of the United States.

Grass is purple $\leftrightarrow$ grass is colorless.

The first statement is true because both of its components are true, the second because both of its components are false. Obviously, then, '$\leftrightarrow$' differs from 'if and only if', just as '$\rightarrow$' differs from 'if . . . then'. The material biconditional and English biconditionals do agree, however, when one component is true and the other false; here the biconditional itself is surely false.

To summarize: The meanings of the five truth-functional operators of propositional logic are given by the rules in Table 3.1. These rules constitute the complete semantics for classical propositional logic. For each operator there is a rule telling when formulas of which it is the main operator are true and a rule telling when those formulas are false. Together this pair of rules implies each such formula is false if and only if it is not true. Hence collectively the valuation rules embody the principle of bivalence—the principle that each formula is either true or false, but not both, on all valuations.

The rules are numbered 1–5. We will use this numbering for future reference.

Exercise 3.1

The five operators discussed in this section are a somewhat arbitrary selection from among many possible truth-functional operators, some of them expressible by common words or phrases of English. Invent symbols and formulate truth tables and a valuation rule for binary operators expressible by these English terms (you may find it easier to do the truth tables first):

1. exclusive 'or'
2. '. . . unless . . .'
3. 'neither . . . nor . . .'
4. 'not both . . . and . . .'; this is sometimes called the nand operator.

3.2 TRUTH TABLES

The valuation rules tell us what the operators of propositional logic mean. But they do more than that. They also enable us to understand why in some cases one formula must be true if others are true; that is, they enable us to understand why some argument forms are valid—and not merely to understand, but to confirm our understanding by calculation. Because propositional logic makes such calculations possible, it is sometimes called the **propositional calculus.**

In Chapter 1 we said that an argument is valid iff it has no counterexample—that is, iff there is no possible situation in which its premises are true but its conclusion is untrue. In this chapter we have shifted our attention from specific arguments to argument forms. For forms, too, we may define a notion of counterexample:

> DEFINITION A **counterexample** to a sequent or argument form is a valuation on which its premises are true and its conclusion is not true.

This notion of a counterexample is closely related to the earlier one. Given a counterexample to a sequent, we can always convert it to a counterexample to an argument that is an instance of that sequent by giving an appropriate interpretation to the form's sentence letters.

Consider, for example, the invalid sequent 'P ∨ Q ⊢ Q'. The valuation $\mathcal{V}$ such that $\mathcal{V}$('P') = T and $\mathcal{V}$('Q') = F is a counterexample to this sequent. For since $\mathcal{V}$('P') = T, by the valuation rule for disjunction $\mathcal{V}$('P ∨ Q') = T. But $\mathcal{V}$('Q') = F. That is, $\mathcal{V}$ is a valuation that makes the premise 'P ∨ Q' of this sequent true and its conclusion 'Q' untrue. Now any interpretation that correlates 'P' with a true statement and 'Q' with a false one produces an instance of this sequent that has a counterexample. And, indeed, this counterexample will describe an *actual* situation. (An actual situation is, of course, a kind of possible situation; anything actual can be coherently described.) For example, suppose we interpret 'P' by the true statement 'People are mammals' and 'Q' by the false statement 'Quail are mammals'. Then (retaining the '∨' symbol) this interpretation yields the following instance of the sequent:

People are mammals ∨ Quail are mammals.
∴ Quail are mammals.

And in the actual situation the premise is true but the conclusion isn't.

Conversely, given a counterexample to an instance of a sequent, we can always construct a counterexample to the sequent by assigning to its sentence letters the truth values of the corresponding sentences in the counterexample to the instance. Consider, for example, this argument, which is an instance of the sequent 'P ⊢ P & Q':

Bill is a prince.
∴ Bill is a prince & Jill is a queen.

Here is a counterexample to this argument:

Bill is a prince, but Jill, the miller's daughter, is a poor but honest maiden, not a queen.

This counterexample makes 'Bill is a prince' true and 'Jill is a queen' false. We can turn it into a counterexample to the sequent by ignoring our interpretation of the sentence letters and assigning these truth values directly to the corresponding sentence letters themselves. The result is the valuation $\mathcal{V}$ such that $\mathcal{V}$('P') = T and $\mathcal{V}$('Q') = F, which is a counterexample to the sequent.

In this way we can always convert a counterexample to an instance of a sequent into a counterexample to the sequent itself, and vice versa. This realization leads us to a new understanding of the concept of a valid argument form. In Section 2.1, we defined a valid argument form as a form all of whose instances are valid arguments. Thus an argument form is valid iff none of its instances have counterexamples. But we have just seen that for each possible situation that is a counterexample to an instance there is a valuation that is a counterexample to the form, and vice versa. Therefore, to say that no instances of the form have counterexamples is equivalent to saying that the form itself has no counterexamples. Thus we may equally well define validity for an argument form as follows:

DEFINITION A sequent or argument form is **valid** iff there is no valuation on which its premises are true and its conclusion is not true.

Likewise, since an invalid form is just one that has an instance with a counterexample, and since there is a counterexample to some instance iff the form has a counterexample, we may likewise redefine the concept of invalidity for an argument form:

DEFINITION A sequent or argument form is **invalid** iff there is at least one valuation on which its premises are true and its conclusion is not true.

We shall rely on these new definitions from now on. Central to both is the concept used by the valuation rules to define truth conditions: the concept of a

valuation. Thus these definitions illuminate the relationship between the concepts of validity and invalidity and the valuation rules. The remainder of this chapter shows how to utilize this relationship to develop computational tests for validity and other semantic properties.

As a first step in this direction we note that, given a valuation, the valuation rules enable us to calculate the truth value of a formula or set of formulas from the truth values assigned to their component sentence letters. For example, given the valuation $\mathcal{V}$ such that $\mathcal{V}$('P') = T, $\mathcal{V}$('Q') = T, and $\mathcal{V}$('R') = F, we can calculate the truth value of the formula 'P → (Q & ~R)' as follows. Since $\mathcal{V}$('R') ≠ T, by the valuation rule for negation, $\mathcal{V}$('~R') = T. And since both $\mathcal{V}$('Q') = T and $\mathcal{V}$('~R') = T, by the valuation rule for conjunction $\mathcal{V}$('Q & ~R') = T. And, finally, since both $\mathcal{V}$('P') = T and $\mathcal{V}$('Q & ~R') = T, by the valuation rule for the material conditional, $\mathcal{V}$('P → (Q & ~R)') = T.

We may list the results of such calculations for all the valuations of a formula or set of formulas on a truth table. If we do this for the set of formulas that comprises a sequent, the table will display all the possible valuations of the premises and conclusion, each as a single horizontal line. We can then, simply by scanning down the table, check to see if there is a line (i.e., a valuation) on which the premises are true and the conclusion is false. If so, that line represents a counterexample to the sequent and the sequent is invalid. If not, then (since all the valuations of the sequent are displayed on the table) there is no counterexample and the sequent is valid. Here at last is a simple, mathematically rigorous test for validity, one that relies on neither intuition nor imagination!

Let's try it out. Our example will be a sequent expressing modus ponens, where '→' is now explicitly understood as the material conditional. This sequent contains only two sentence letters, so it has four possible valuations (both 'P' and 'Q' true, 'P' true and 'Q' false, 'P' false and 'Q' true, both 'P' and 'Q' false), which we list in the two leftmost columns of the table. Then, beneath each formula of the sequent, we write its truth value on each of those valuations, like this:

P	Q	P → Q,	P	⊢ Q
T	T	T	T	T
T	F	F	T	F
F	T	T	F	T
F	F	T	F	F

Each horizontal line represents a single valuation. For example, the second line from the bottom represents the valuation $\mathcal{V}$ such that $\mathcal{V}$('P') = F and $\mathcal{V}$('Q') = T. To the right, below each formula of the sequent, is listed the truth value of that formula on $\mathcal{V}$. On this valuation, for example, 'P → Q' is true. Since the truth table is a complete list of valuations, if there is a valuation on which the premises are true and the conclusion is not, it will show up as a line on the table. In this case, there is no such line, that is, no counterexample. (The only valuation on which both premises are true is the first one listed, the valuation $\mathcal{V}$ such that $\mathcal{V}$('P') = T and $\mathcal{V}$('Q') = T. But on this valuation the form's conclusion 'Q' is true.) Thus modus ponens is valid for the material conditional.

Affirming the consequent, which is sometimes confused with modus ponens, is, as we saw in Section 2.1, intuitively *in*valid. Its truth table confirms our intuitions:

P	Q	P → Q,	Q	⊢ P
T	T	T	T	T
T	F	F	F	T
F	T	T	T	F
F	F	T	F	F

On the third valuation listed in the table the premises 'P → Q' and 'Q' are both true, but the conclusion 'P' is false. This valuation is therefore a counterexample to the sequent, proving it invalid.

We can use the counterexample displayed in the truth table to construct instances of the sequent that are invalid arguments. Since the valuation which makes 'P' false and 'Q' true provides a counterexample, we need merely substitute any sentence that is actually false for 'P' and any sentence that is actually true for 'Q' to obtain an invalid instance. Since these are the truth values these sentences have in the actual situation, a description of the actual situation constitutes a counterexample to that instance. Let 'P', for example, be the false sentence 'Logic is a kind of biology' and 'Q' the true sentence 'Logic is an intellectual discipline'. Then we obtain this instance:

Logic is a kind of biology → Logic is an intellectual discipline.
Logic is an intellectual discipline.
∴ Logic is a kind of biology.

The premises of this argument are true and its conclusion false in the actual situation.

This technique sometimes yields puzzling instances whose conditional premise, though actually true, seems false if we confuse '→' with 'if . . . then'. But keeping the truth conditions for '→' distinctly in mind resolves the puzzlement.

To obtain the values listed under the formulas in the preceding tables, we just copied them from one of the leftmost columns (if the formula was a sentence letter) or derived them directly from the valuation rules (in the case of the conditional formulas). With more complex formulas, however, we may need to apply the valuation rules successively, a step at a time, to calculate truth values for whole formulas.

Consider the sequent '(P & Q) ∨ (¬P & ¬Q) ⊢ ¬P ↔ ¬Q'. (To give this some intuitive content, we might interpret 'P' as the statement 'The princess dines' and 'Q' as the statement 'The queen dines'. This makes the argument: Either the princess and queen both dine or neither dines; therefore the princess does not dine if and only if the queen does not dine.) We begin the table for this sequent as before by listing the four possible valuations of the two sentence letters 'P' and 'Q' in the two leftmost columns. Now we use the valuation rules to calculate truth values, starting with the sentence letters and working our way up to more and more

complex formulas. The first step is to copy the 'P' column at the left of the table under each occurrence of the sentence letter 'P' in the formulas and the 'Q' column under each occurrence of 'Q'. Where 'P' or 'Q' are directly preceded by '~', however, we know that their truth values will be reversed, so in these cases we reverse each truth value in the column we copy. The table now lists the truth values for all sentence letters or negated sentence letters in the formulas:

P	Q	(P	&	Q)	∨	(~P	&	~Q)	⊢	~P	↔	~Q
T	T	T		T		F		F		F		F
T	F	T		F		F		T		F		T
F	T	F		T		T		F		T		F
F	F	F		F		T		T		T		T

Notice that we have written the columns for negated sentence letters under the negation signs. In general, whenever we are listing the truth values for a complex formula, we write them under the operator whose scope is that formula.

The next step is to use the valuation rules to calculate the truth values for the formulas directly joining those whose truth values we have already identified. In the case of the premise, these are the two conjunctions 'P & Q' and '~P & ~Q'. Remembering that a conjunction is true iff both conjuncts are true, we place a 'T' beneath '&' on lines where the sentence letters it joins are both true and an 'F' in all other cases. In the case of the conclusion, the operator joining the '~P' and '~Q' is the biconditional, and its scope is the entire conclusion. This biconditional is true on those lines in which '~P' and '~Q' have the same truth value and false where they differ in truth value. We write these values beneath the symbol '↔'. The truth table now looks like this:

P	Q	(P	&	Q)	∨	(~P	&	~Q)	⊢	~P	↔	~Q
T	T	T	T	T		F	F	F		F	T	F
T	F	T	F	F		F	F	T		F	F	T
F	T	F	F	T		T	F	F		T	F	F
F	F	F	F	F		T	T	T		T	T	T

One step in our calculation remains. The premise is a disjunction of the two conjunctions whose truth values we have just determined. A disjunction is true if and only if one or both of its disjuncts are true; otherwise, it is false. Using this rule, we write the appropriate truth values beneath the symbol '∨':

P	Q	(P	&	Q)	∨	(~P	&	~Q)	⊢	~P	↔	~Q
T	T	T	T	T	T	F	F	F		F	T	F
T	F	T	F	F	F	F	F	T		F	F	T
F	T	F	F	T	F	T	F	F		T	F	F
F	F	F	F	F	T	T	T	T		T	T	T

We also circle the column under the main operator of each formula. Only the

circled values, the ones listing the truth values for whole formulas, matter. The other columns of truth values on the table are merely part of the calculation by which the circled values were obtained.

As before we read the table by scanning down the columns of truth values for whole formulas (the circled columns), looking for a valuation on which the premise but not the conclusion is true. There isn't any; that is, there is no counter-example. So the sequent is valid.

When a sequent contains two sentence letters, there are four valuations of that sequent and hence four lines on the truth table. But not all argument forms contain two sentence letters. Some contain only one, some three or more. Where the number of sentence letters in a sequent is n, the number of valuations of the sequent, and the number of lines on its truth table, is 2^n. Thus a sequent containing only one sentence letter has $2^1 = 2$ valuations, a sequent containing three has $2^3 = 8$ valuations, a sequent containing four has $2^4 = 16$ valuations, and so on.

It is useful to list the valuations in a standard order. Our convention is as follows: List all the sentence letters of the sequent horizontally in the top left corner of the table *in alphabetical order.* (Where the letters have superscripts the order remains alphabetical, but letters with lower superscripts are written before those with higher superscripts, and letters with no superscripts come first of all.)

Then list the valuations under these letters as follows. Beneath the *rightmost* letter, write a column of alternating 'T's and 'F's, beginning with 'T', continuing downward until you have written 2^n 'T's and 'F's, where n is the total number of sentence letters. Then under the next rightmost letter, write another column of 'T's and 'F's, beginning with 'T', but doubling the number alternated. In other words, write two 'T's, two 'F's, and so on, downward until again you have written 2^n 'T's and 'F's. Now moving to the next rightmost letter (if one remains) and beginning, as always, with 'T' (because, after all, truth deserves priority over falsehood), write another column of 2^n T's and F's, doubling the alternation again (four 'T's, four 'F's, and so on). Keep moving to the left, doubling the number of the alternation between T's and F's until each letter has a column of 'T's and 'F's beneath it. For the three letters 'P', 'Q', and 'R', for example, the listing of valuations looks like this:

P	Q	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

Argument forms containing many sentence letters are tested for validity in just the same way as argument forms containing only one or two. Consider, for

example, the sequent '(P & Q) → R ⊢ (P → R) & (Q → R)'. Once again, we simply recopy the columns for the sentence letters from the columns to the left, then calculate the values for formulas of successively larger scope by using the valuation rules, and finally circle the columns under the main operators. Here is the result:

P	Q	R	(P	&	Q)	→	R	⊢	(P	→	R)	&	(Q	→	R)
T	T	T	T	T	T	T	T		T	T	T	T	T	T	T
T	T	F	T	T	T	F	F		T	F	F	F	T	F	F
T	F	T	T	F	F	T	T		T	T	T	T	F	T	T
T	F	F	T	F	F	T	F		T	F	F	F	F	T	F
F	T	T	F	F	T	T	T		F	T	T	T	T	T	T
F	T	F	F	F	T	T	F		F	T	F	F	T	F	F
F	F	T	F	F	F	T	T		F	T	T	T	F	T	T
F	F	F	F	F	F	T	F		F	T	F	T	F	T	F

The table shows that two valuations are counterexamples to this sequent: the valuation on which 'P' is true and 'Q' and 'R' are false, and the valuation on which 'P' is false, 'Q' true, and 'R' false. Thus the sequent is invalid.

The significance of these counterexamples can be made clearer by considering a specific interpretation. Let 'P' stand for 'The match is lighted', 'Q' for 'You drop the match into a can of gasoline', and 'R' for 'An explosion occurs'. Then the first counterexample represents a situation in which the match is lighted but you don't drop it into a can of gasoline so that no explosion occurs, and the second counterexample represents a situation in which the match is not lighted and you do drop it into a can of gasoline but once again no explosion occurs.

Let's now consider the truth table for the sequent 'Q ⊢ P → P'. Like some of the examples discussed in Section 1.3, this sequent lacks relevance; yet it is clearly valid, as its truth table shows:

P	Q	Q	⊢	P	→	P
T	T	T		T	T	T
T	F	F		T	T	T
F	T	T		F	T	F
F	F	F		F	T	F

The sequent is valid because there is no valuation on which the premise is true but the conclusion is not. This, however, is due to a peculiarity of the conclusion: It is true on all valuations; it cannot in any way be false. In Section 1.3 we noted that a logical truth, that is, a *statement* true in all possible situations, validly follows from any set of premises. 'P → P' is, of course, a symbolic formula, not a statement. But if we supply it with an interpretation by assigning to 'P' some specific statement (*any* statement will do), then 'P → P' expresses a logical truth. In a sense, then, 'P → P' itself, though a formula and not a statement, is logically true. A formula that is logically true in this sense is said to be **valid:**

DEFINITION A **valid formula** is a formula true on all of its valuations.

Up until now we have applied the term 'valid' exclusively to arguments or argument forms, not to formulas. Since it is important not to confuse the formulas with argument forms, perhaps we ought to use some other term here to avoid confusion. Yet there is a substantial justification for applying the term 'valid' to both. For a valid formula is in effect a valid sequent with no premises. That is, we may think of a valid formula as a formula which may legitimately function as a conclusion without any premises at all. Since there is no valuation on which this "conclusion" is not true, there is no valuation on which the (nonexistent) premises are true and the "conclusion" is not true, and hence no counterexample. A valid formula may thus be regarded as a limiting case of a valid sequent.

Actually, in propositional logic we do have a term that substitutes nicely for 'valid formula'. The term is 'tautology':

DEFINITION A **tautology** is a formula whose truth table displays a column consisting entirely of 'T's under its main operator.

In elementary propositional logic, a tautology (or **tautologous formula**) and a valid formula are the same thing. But valuations in more advanced forms of logic are not always expressible by truth tables. Thus in later chapters we shall encounter valid formulas that are not tautologies. 'Valid formula' is the more general and widely applicable of the two terms.

Truth tables make it graphically clear why a tautology validly follows from any set of premises. Such an inference is valid because it has no counterexample. It can't have a counterexample (that is, a valuation on which the premises are true but the conclusion is not) because a tautologous (valid) conclusion is true on all valuations.

In Section 1.3 we also noted that any argument with an inconsistent set of premises is valid. The corresponding notions of inconsistency for a formula or set of formulas are as follows:

DEFINITION A *formula* is **inconsistent** iff there is no valuation on which it is true.

DEFINITION A *set of formulas* is **inconsistent** iff there is no valuation on which all the formulas in the set are true.

Inconsistent formulas may also be called **self-contradictory formulas** or **contradictions.** With regard to truth tables, a formula is inconsistent if and only if the

column under its main operator contains only 'F's, and a set of formulas is inconsistent if and only if there is no horizontal line on which all formulas show a 'T' beneath their main operators.

In Section 1.3 we observed that any conclusion follows from an inconsistent set of premises. The medievals called this principle *ex falso quodlibet*—"from a falsehood, anything you please." (Actually, this is misleading; the phrase better fits the relation between the antecedent and consequent of the material conditional; a false antecedent materially implies whatever you please. But a merely *false* premise or premise set does not validly imply all conclusions. From the premise 'The Earth is flat', for example, we cannot validly deduce whatever we please. We can, however, do so from an *inconsistent* premise or premise set.) Ex falso quodlibet may be plainly depicted on a truth table. The premises of the sequent 'P, ~P ⊢ Q', for example, constitute an inconsistent set:

P	Q	P,	~P	⊢	Q
T	T	T	F		T
T	F	T	F		F
F	T	F	T		T
F	F	F	T		F

By scanning down the table, we see that there is no horizontal line (valuation) on which both premises are true so that the set consisting of the two premises is inconsistent. Obviously, then, there is no valuation on which both premises are true while the conclusion is not true. So the sequent is valid.

Corresponding to the two notions of inconsistency defined above are the following two notions of consistency:

DEFINITION A *formula* is **consistent** iff it is true on at least one valuation.

DEFINITION A *set of formulas* is **consistent** iff there is at least one valuation on which all the formulas in the set are true.

The truth table of a consistent formula has at least one 'T' in the column beneath its main operator. The truth table of a consistent set of formulas contains at least one horizontal line on which there is a 'T' beneath the main operator of every formula of the set. Some authors use the term '**satisfiable**' instead of 'consistent'.

Formulas which are consistent but not valid are said to be **contingent:**

DEFINITION A formula is **contingent** iff it is true on some of its valuations and not true on others.

TABLE 3.2
Semantic Classification of Formulas

Type of Formula	Definition	Truth-Table Indication
Valid (Tautologous)	True on all valuations	Column under main operator contains only 'T's
Contingent	True on at least one but not all valuations	Column under main operator contains both 'T's and 'F's
Inconsistent	True on no valuations	Column under main operator contains only 'F's

TABLE 3.3
Semantic Classification of Sets of Formulas

Type of Set	Definition	Truth-Table Indication
Consistent	There is at least one valuation on which all formulas in the set are true	Horizontal line on which all formulas in the set have 'T's under their main operators
Inconsistent	There is no valuation on which all formulas in the set are true	There is no horizontal line on which all formulas in the set have 'T's under their main operators

All formulas fall into one of the three categories: valid, contingent, or inconsistent. These are summarized in Table 3.2. A consistent formula, of course, is just one that is not inconsistent. It is therefore either valid or contingent. We could also, therefore, have divided all formulas into the two classifications, "consistent" and "inconsistent," instead of into the three categories described in Table 3.2. For *sets* of formulas this twofold classification is generally the most useful. See Table 3.3.

Truth tables are useful for detecting one other semantic relationship that is of great importance—logical equivalence:

> DEFINITION Two formulas are **logically equivalent** iff they have the same truth value on every valuation of both.

With respect to truth tables, two formulas are logically equivalent if and only if the columns under their main operators are identical. Consider, for example, the formulas '~(P ∨ Q)' and '~P & ~Q', which as we saw in Section 2.2 are both ways

of symbolizing 'neither P nor Q'. They are logically equivalent, as we can see by placing them both on the same truth table:

P	Q	~	(P	∨	Q)		~P	&	~Q
T	T	F	T	T	T		F	F	F
T	F	F	T	T	F		F	F	T
F	T	F	F	T	T		T	F	F
F	F	T	F	F	F		T	T	T

The truth table reveals their equivalence in that the columns under their main operators (the circled columns) are identical.

Logical equivalence is significant for several reasons. For one thing, logically equivalent formulas validly imply one another. That is, an inference from either as premise to the other as conclusion is valid. There can be no counterexample because since the two formulas have the same truth values on all valuations, there is no valuation on which the premise but not the conclusion of such an inference is true.

Further, since in classical logic meaning is truth conditions and since logically equivalent formulas have identical truth conditions, it follows that logically equivalent formulas have, for the purposes of classical logic, the same meaning. Thus '~(P ∨ Q)' and '~P & ~Q' are equally adequate symbolizations for 'neither P nor Q' because from the viewpoint of classical logic they are synonymous. 'Neither . . . nor', in other words, is not ambiguous. Rather, these formulas are two ways of expressing the single meaning that 'neither . . . nor' has in English. The logical meaning of 'neither . . . nor' is, in other words, simply the pattern of truth values the table for each of these formulas displays.

The material conditional formula 'P → Q' is equivalent to both '~P ∨ Q' and '~(P & ~Q)', as the following table shows:

P	Q	P	→	Q		~P	∨	Q		~	(P	&	~Q)
T	T	T				F	T	T		T	T	F	F
T	F	F				F	F	F		F	T	T	T
F	T	T				T	T	T		T	F	F	F
F	F	T				T	T	F		T	F	F	T

This means that so far as logic is concerned 'P → Q', '~P ∨ Q', and '~(P & ~Q)' all have the same meaning. Thus, whenever we symbolize a statement as 'P → Q', we could as well symbolize it as '~P ∨ Q' or as '~(P & ~Q)'. It follows that the material conditional is a needless redundancy. We could exclude it from the language of propositional logic and still be able (using '&' or '∨', and '~') to say everything we could say before. This is true even for very complex formulas. If, for example, we consistently replace formulas of the form Φ → Ψ with formulas of the form ~Φ ∨ Ψ, then instead of writing

P → (Q → R)

we would write

$$\neg P \vee (\neg Q \vee R)$$

Likewise, we could eliminate '$\leftrightarrow$' by taking formulas of the form $\Phi \leftrightarrow \Psi$ as abbreviations for more complex but equivalent formulas, say, those of the form $(\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)$ (see problem 1 of Exercise 3.2.4). This would leave us with only three operators: '$\neg$', '&', and '$\vee$'.

We could reduce our vocabulary still further, either by rewriting $\Phi \,\&\, \Psi$ as $\neg(\neg\Phi \vee \neg\Psi)$ or by rewriting $\Phi \vee \Psi$ as $\neg(\neg\Phi \,\&\, \neg\Psi)$ (see problems 2 and 3 of Exercise 3.2.4), thus eliminating either '&' or '$\vee$'. Only two of the five operators, then, are really needed—either '$\neg$' and '&' or '$\neg$' and '$\vee$'. And either of these pairs can be further reduced to a single operator, though not to one of the familiar five. This final reduction requires either the operator 'neither . . . nor', which is written as a downward pointing arrow, '$\downarrow$', or as the operator 'not both . . . and', sometimes called 'nand', and written simply as '$|$'. (Neither symbol belongs to the language of propositional logic we defined it in Section 2.3; thus to do this reduction we would have to adopt a new set of formation rules.) The truth tables for these operators are as follows:

Φ	Ψ	$\Phi \downarrow \Psi$
T	T	F
T	F	F
F	T	F
F	F	T

Φ	Ψ	$\Phi \mid \Psi$
T	T	F
T	F	T
F	T	T
F	F	T

Here we consider the reduction of '&' and '$\neg$' to '$|$', leaving the reduction to '$\downarrow$' as an exercise. This reduction depends on the equivalence of $\neg\Phi$ to $\Phi|\Phi$ and the equivalence of $\Phi \,\&\, \Psi$ to $\neg(\Phi|\Psi)$, that is, $(\Phi|\Psi)|(\Phi|\Psi)$, which the following truth tables illustrate:

Φ	$\neg\Phi$	$\Phi\|\Phi$
T	F	F
F	T	T

Φ	Ψ	$\Phi \,\&\, \Psi$	$(\Phi \mid \Psi)$	$\mid$	$(\Phi \mid \Psi)$
T	T	T	F	T	F
T	F	F	T	F	T
F	T	F	T	F	T
F	F	F	T	F	T

(In each case we represent two separate formulas on the same table to show their equivalence.) Thus we can see that any formula has an equivalent whose sole

operator is '|'. We can express the conditional 'P → Q', for example, first in terms of '–' and '&':

$$\neg(P \,\&\, \neg Q)$$

Then the negation sign prefixing 'Q' may be eliminated in terms of '|':

$$\neg(P \,\&\, (Q|Q))$$

(The order in which we eliminate particular occurrences of '–' and '&' is arbitrary.) Next we eliminate the '&' in terms of '|':

$$\neg((P \mid (Q|Q)) \mid (P \mid (Q|Q)))$$

And finally, we eliminate the initial negation:

$$((P \mid (Q|Q)) \mid (P \mid (Q|Q))) \mid ((P \mid (Q|Q)) \mid (P \mid (Q|Q)))$$

The material conditional! In principle, we could by such means express every formula of propositional logic solely in terms of the operator '|'. The unreadability of the result explains why in practice we don't. Thus we will stick with the traditional five operators, chiefly because, though redundant, they give us a reasonably comprehensible language.

Yet here we have learned something remarkable: We can say much with paltry means if we are willing to tolerate long formulas. Humans generally are not; but, as we shall see in Chapter 10, computers have a different opinion.

Exercise 3.2.1

Use truth tables to test the following argument forms for validity. Write either 'valid' or 'invalid' beside the table to indicate the answer.

1. $P \vee Q, P \vdash Q$
2. $P \vee Q, \neg P \vdash Q$
3. $P \rightarrow Q \vdash \neg Q \rightarrow \neg P$
4. $P \rightarrow Q \vdash Q \rightarrow P$
5. $P \rightarrow Q, \neg P \vdash \neg Q$
6. $P \rightarrow Q, \neg Q \vdash \neg P$
7. $P \vdash Q \rightarrow P$
8. $\neg Q \vdash Q \rightarrow P$
9. $P \vdash P \rightarrow Q$
10. $\neg(P \rightarrow Q) \vdash P \,\&\, \neg Q$
11. $P \vee Q \vdash P \leftrightarrow Q$
12. $P \,\&\, Q \vdash P \leftrightarrow Q$
13. $P \,\&\, Q \vdash Q \,\&\, P$
14. $P \vee Q, P \rightarrow R, Q \rightarrow R \vdash R$
15. $P \vee Q, Q \vee R \vdash P \vee R$
16. $P \rightarrow Q, Q \rightarrow P \vdash P \leftrightarrow Q$

17. $P \vdash \neg(P \,\&\, \neg P)$
18. $\neg(P \,\&\, Q) \vdash \neg P \,\&\, \neg Q$
19. $\neg(P \vee \neg P) \vdash Q$
20. $P \vdash (P \rightarrow Q) \rightarrow (P \,\&\, Q)$

Exercise 3.2.2

Use truth tables to determine whether the following formulas are valid, contingent, or inconsistent. Write your answer beside the table.

1. $P \rightarrow \neg P$
2. $P \rightarrow \neg\neg P$
3. $P \leftrightarrow \neg\neg P$
4. $P \leftrightarrow \neg P$
5. $(P \vee Q) \vee (\neg P \,\&\, \neg Q)$
6. $P \,\&\, \neg\neg P$
7. $P \vee P$
8. $(P \,\&\, (P \rightarrow Q)) \rightarrow Q$
9. $(P \vee Q) \leftrightarrow (Q \vee P)$
10. $(P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$

Exercise 3.2.3

Use truth tables to determine whether the following sets of formulas are consistent or inconsistent. Write your answer beside the table.

1. $P \rightarrow Q, Q \rightarrow \neg P$
2. $P \leftrightarrow Q, Q \leftrightarrow \neg P$
3. $P \vee Q, \neg P, \neg Q$
4. $P \,\&\, Q, \neg P$
5. $\neg(P \,\&\, Q), P \vee Q$

Exercise 3.2.4

1. Use a truth table to verify that '$P \leftrightarrow Q$' is logically equivalent both to '$(P \rightarrow Q) \,\&\, (Q \rightarrow P)$' and to '$(P \,\&\, Q) \vee (\neg P \,\&\, \neg Q)$'.
2. Use a truth table to verify that '$P \,\&\, Q$' is logically equivalent to '$\neg(\neg P \vee \neg Q)$'.
3. Use a truth table to verify that '$P \vee Q$' is logically equivalent to '$\neg(\neg P \,\&\, \neg Q)$'.
4. Use a truth table to verify that '$Q \vee P$' and '$\neg Q \rightarrow P$', which are both ways of symbolizing 'P unless Q', are equivalent.
5. Find equivalents for the forms $\neg\Phi$ and $\Phi \,\&\, \Psi$ in terms of '$\downarrow$', and show that they are logical equivalents by constructing the appropriate truth tables.
6. Find a logical equivalent for $\Phi \vee \Psi$ in terms of '$\mid$', and demonstrate the equivalence with a truth table. Do the same thing in terms of '$\downarrow$'.

3.3 SEMANTIC TREES

A semantic tree is a device for displaying all the valuations on which the formula or set of formulas is true. Since classical logic is bivalent, the valuations on which the formula or set of formulas is false are then simply those not displayed. Thus trees do the same job as truth tables. But they do it more efficiently; especially for long problems, a tree generally requires less computation and writing than the corresponding truth table. A truth table for a formula or sequent containing n sentence letters has 2^n lines. For $n = 10$, for example, $2^n = 1024$—a good many more lines than we are likely to want to write. But a tree for a formula or sequent with ten sentence letters (or even more) may fit easily within a page. Moreover, as we shall see in Section 7.4, trees have the advantage of being straightforwardly generalizable to predicate logic, which truth tables are not.

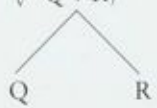
Suppose, for instance, that we want a list of the valuations on which the formula '~P & (Q ∨ R)' is true. To obtain this by the tree method, we write the formula and then begin to break it down into those smaller formulas which, according to the valuation rules, must be true in order to make '~P & (Q ∨ R)' true. Now '~P & (Q ∨ R)' is a conjunction, and a conjunction is true iff both of its conjuncts are true. So we write '~P & (Q ∨ R)', then check it off (to indicate that it has been analyzed), and write its two conjuncts beneath it, like this:

√ ~P & (Q ∨ R)
~P
Q ∨ R

A formula which has been checked off is in effect eliminated. We need pay no further attention to it. What remains, then, are the two formulas '~P' and 'Q ∨ R'. '~P' is true on just those valuations on which 'P' is false. But we still need to analyze 'Q ∨ R'.

Now, whereas a conjunction can be true in only one way (both conjuncts are true), there are two ways in which a disjunction can be true: Either the first disjunct is true or the second disjunct is true (or both—but this possibility is in effect already included in the other two, as will be explained shortly). Hence to analyze 'Q ∨ R', we check it and split our list into two branches, the first representing valuations in which 'Q' is true, the second representing valuations on which 'R' is true, as follows:

√ ~P & (Q ∨ R)
~P
√ Q ∨ R)



It is because lists may "branch" in this way that the structures we create by this procedure are called semantic *trees*. (But these trees grow downward!) When all

formulas other than sentence letters or negated sentence letters have been checked off, as they have here, the tree is finished. This tree contains two "branches," or *paths*, one running from '~P & (Q ∨ R)' to 'Q', the other from '~P & (Q ∨ R)' to 'R'.

> DEFINITION A **path** through a tree (in any stage of construction) is a complete column of formulas from the top to the bottom of the tree.

Now we scan along each path, looking for sentence letters or negated sentence letters. Along the first path we find '~P' and 'Q'. This shows that '~P & (Q ∨ R)' is true on those valuations which make both '~P' and 'Q' true, that is, those valuations which make 'P' false and 'Q' true. But 'R' does not appear either by itself or negated along the first path. This indicates that if 'P' is false and 'Q' true, '~P & (Q ∨ R)' is true, regardless of whether 'R' is true or false. The first path, then, represents these two valuations:

P	Q	R
F	T	T
F	T	F

Checking the second path for sentence letters or negated sentence letters, we find '~P' and 'R'. This means that '~P & (Q ∨ R)' is also true on valuations in which 'P' is false and 'R' is true, regardless of the truth value of 'Q', which does not appear, either alone or negated, along that path. Hence the second path represents these valuations:

P	Q	R
F	T	T
F	F	T

Notice that there is some redundancy here. Both paths represent the valuation on which 'P' is false and both 'Q' and 'R' are true. This is what I meant when I said a few paragraphs back that the possibility of both disjuncts being true is in effect already included in the possibilities of either disjunct being true. Thus together the two paths represent three valuations, not four:

P	Q	R
F	T	T
F	T	F
F	F	T

These are precisely the lines of the truth table on which '~P & (Q ∨ R)' is true; they represent all the valuations which make this formula true. In this way the tree procedure accomplishes exactly what truth tables do.

Sometimes as we are constructing a tree, we find that a formula and its negation both appear on the same path. Since no valuation can make both a formula and its negation true, this means that the path does not represent any valuation. It is merely a failed attempt to find valuations that make the formulas of the initial list true. Such a path is considered "blocked" or "closed," and this is indicated by writing an 'X' beneath it.

Consider, for example, the tree for the formula '~(P & Q) & P'. Since this formula is a conjunction, after writing it we check it and analyze it into its two conjuncts, like this:

```
√ ~(P & Q) & P
  ~(P & Q)
  P
```

Now '~(P & Q)' is true iff 'P & Q' is false. By valuation rule 2, 'P & Q' is false iff either 'P' or 'Q' or both are untrue—that is, by valuation rule 1, iff either '~P' or '~Q' or both are true. Thus we may check '~(P & Q)' and split our list into two branches, the first representing valuations on which '~P' is true, the second representing valuations on which '~Q' is true:

```
√ ~(P & Q) & P
  ~(P & Q)
  P
  /\
~P  ~Q
X
```

We have placed an 'X' at the bottom of the left branch because the path it represents—the path extending from '~(P & Q) & P' to '~P'—contains both 'P' and '~P' and must be closed. This path represents no valuations. The path that follows the right branch, however, does not close. On it we find 'P' and '~Q'. Since these are the only letters in our initial formula, '~(P & Q) & P', that formula is true on only one valuation, namely, the valuation on which 'P' is true and 'Q' is false.

Sometimes all paths close. This indicates that the initial formula or set of formulas is not true on any valuations, that is, is inconsistent. Consider, for example, the formulas 'P ∨ Q', '~P', and '~Q', which together form an inconsistent set. We may set them down in a vertical list and apply the same procedure as above:

```
√ P ∨ Q
  ~P
  ~Q
  /\
 P   Q
 X   X
```

When we check 'P ∨ Q' and analyze it into its two truth possibilities, 'P' and 'Q', we see that each of the resulting paths contains both a formula and its negation: 'P' and '~P' on the path that branches to the left, 'Q' and '~Q' on the path that branches to the right. So we place an 'X' at the bottom of each path to indicate that it is closed. With both paths closed, there is nothing more to be done. The tree is complete, and it shows that there are no valuations on which the formulas of our initial list ('P ∨ Q', '~P', and '~Q') are all true.

In summary, then, to construct a semantic tree, list the formula or set of formulas to be tested in a single column. Then check off a complex formula on the list, writing at the bottom of the list simpler formulas that would have to be true if this complex formula were true. If the complex formula can be true in more than one way, split the list and display formulas representing each of these ways on a separate "branch." Then repeat this procedure for other complex formulas in the list. Eventually along each path of the tree one of two things will occur. Either (1) all the formulas along that path of the tree will be simplified into sentence letters or negations of sentence letters or (2) some formula and its negation both will appear. In the first case, the path displays one or more valuations on which the initial formula or set of formulas is true—namely, those valuations on which isolated sentence letters along that path are assigned the letter T, negated sentence letters along that path are assigned the value F, and sentence letters not appearing either negated or unnegated along the path are assigned either T or F. In the second case, the path is a dead end and represents only a failed attempt to construct a valuation. We close it with an 'X'.

The following terminology will be helpful:

DEFINITION A *path* is **finished** if it is closed or if the only unchecked formulas it contains are sentence letters or negations of sentence letters so that no more rules apply to its formulas. A *tree* is **finished** if all of its paths are finished.

DEFINITION An **open path** is a path that has not been ended with an 'X'.

DEFINITION A **closed path** is a path that has been ended with an 'X'.

DEFINITION A formula **occurs on** a path if (1) it is on that path and is not merely a subformula of some other formula on that path and (2) it is unchecked.

In order to apply the tree procedure formally, we need to specify exact rules by which complex formulas are to be analyzed into simpler components. There is nothing surprising here. The tree rules simply mimic the valuation rules. This is why trees are called *semantic* trees. They are simply a perspicuous way of displaying the semantics for any formula or (finite) set of formulas.

There are ten tree rules, two for each of the valuation rules (that is, one for the truth clause and one for the falsity clause of each rule). Each tree rule is listed,

along with the clause of the valuation rule to which it corresponds, in Table 3.4. The trees we have done so far have exemplified the negation, conjunction, negated conjunction, and disjunction rules.

The conjunction rule, for example, was the one we used to analyze '~(P & Q) & P' into its components:

1.	√	~(P & Q) & P	Given
2.		~(P & Q)	1 &
3.		P	1 &

Here we have repeated the first step of the tree for '~(P & Q) & P', annotating it by numbering and labeling the lines—a procedure which, having named the rules, we will follow from now on. Line 1 is marked 'given' to indicate that it is the given formula. Lines 2 and 3 are marked '1 &' to show that they are obtained from line 1 by the conjunction rule.

In the second step of this tree, we used the negated conjunction rule to show that there are two ways in which 'P & Q' can be false—namely, if 'P' is false or if 'Q' is false.

1.	√	~(P & Q) & P	Given
2.		~(P & Q)	1 &
3.		P	1 &
4.	~P	~Q	2 -&

Finally, we complete the tree by closing the left branch with the negation rule. From now on we will annotate the 'X' by writing the line numbers on which we found the formula and its negation, which closed the path—in this case, lines 3 and 4:

1.	√	~(P & Q) & P	Given
2.		~(P & Q)	1 &
3.		P	1 &
4.	~P	~Q	2 -&
5.		X 3, 4	

As noted previously, this tree shows that '~(P & Q) & P' is true on only one valuation—namely, the valuation on which 'P' is true and 'Q' is false.

Because a tree for a formula, a set of formulas, or a sequent displays all the information contained in a truth table for that formula, set of formulas, or sequent, any test that can be performed by a truth table can also be performed by a tree. To test a sequent for validity, for example, we must determine whether there is a valuation that makes its premises but not its conclusion true. This would, of course, be a valuation on which both the sequent's premises and the negation of

TABLE 3.4
The Ten Tree Rules

Valuation Rule	Corresponding Tree Rule
1 $\mathcal{V}(\sim\Phi) = \text{T}$ iff $\mathcal{V}(\Phi) \neq \text{T}$.	**Negation** ($\sim$) If an open path contains both a formula and its negation, place an 'X' at the bottom of the path.
$\mathcal{V}(\sim\Phi) = \text{F}$ iff $\mathcal{V}(\Phi) = \text{T}$.	**Negated Negation** ($\sim\sim$) If an open path contains an unchecked formula of the form $\sim\sim\Phi$, check it and write Φ at the bottom of every open path that contains this newly checked formula.
2 $\mathcal{V}(\Phi \,\&\, \Psi) = \text{T}$ iff both $\mathcal{V}(\Phi) = \text{T}$ and $\mathcal{V}(\Psi) = \text{T}$.	**Conjunction** (&) If an open path contains an unchecked formula of the form $(\Phi \,\&\, \Psi)$, check it and list Φ above Ψ at the bottom of every open path that contains this newly checked formula.
$\mathcal{V}(\Phi \,\&\, \Psi) = \text{F}$ iff either $\mathcal{V}(\Phi) \neq \text{T}$ or $\mathcal{V}(\Psi) \neq \text{T}$, or both.	**Negated Conjunction** ($\sim$&) If an open path contains an unchecked formula of the form $\sim(\Phi \,\&\, \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first write $\sim\Phi$, and at the end of the second write $\sim\Psi$.
3 $\mathcal{V}(\Phi \vee \Psi) = \text{T}$ iff either $\mathcal{V}(\Phi) = \text{T}$ or $\mathcal{V}(\Psi) = \text{T}$, or both.	**Disjunction** ($\vee$) If an open path contains an unchecked formula of the form $(\Phi \vee \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first write Φ, and at the end of the second write Ψ.
$\mathcal{V}(\Phi \vee \Psi) = \text{F}$ iff both $\mathcal{V}(\Phi) \neq \text{T}$ and $\mathcal{V}(\Psi) \neq \text{T}$.	**Negated Disjunction** ($\sim\vee$) If an open path contains an unchecked formula of the form $\sim(\Phi \vee \Psi)$, check it and list $\sim\Phi$ above $\sim\Psi$ at the bottom of every open path that contains this newly checked formula.

its conclusion are true. We may construct a tree to search for just such valuations by starting with the argument's premises and the negation of its conclusion.

Let's test the sequent '$P \rightarrow (Q \,\&\, (R \vee S)) \vdash P \rightarrow Q$'. Since it contains four sentence letters, the truth table would take $2^4 = 16$ lines—a laborious task. The tree method is much more efficient. We list the premise and the negation of the conclusion, labeling them as such to the right, and then analyze them by mechanically applying the tree rules. Here is the result:

TABLE 3.4
The Ten Tree Rules (*continued*)

Valuation Rule	Corresponding Tree Rule
4 $\mathcal{V}(\Phi \rightarrow \Psi) = \mathrm{T}$ iff either $\mathcal{V}(\Phi) \neq \mathrm{T}$ or $\mathcal{V}(\Psi) = \mathrm{T}$, or both.	**Conditional** ($\rightarrow$) If an open path contains an unchecked formula of the form $(\Phi \rightarrow \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first write $\sim\Phi$, and at the end of the second write Ψ.
$\mathcal{V}(\Phi \rightarrow \Psi) = \mathrm{F}$ iff both $\mathcal{V}(\Phi) = \mathrm{T}$ and $\mathcal{V}(\Psi) \neq \mathrm{T}$.	**Negated Conditional** ($\sim\rightarrow$) If an open path contains an unchecked formula of the form $\sim(\Phi \rightarrow \Psi)$, check it and list Φ above $\sim\Psi$ at the bottom of every open path that contains this newly checked formula.
5 $\mathcal{V}(\Phi \leftrightarrow \Psi) = \mathrm{T}$ iff either $\mathcal{V}(\Phi) = \mathrm{T}$ and $\mathcal{V}(\Psi) = \mathrm{T}$, or $\mathcal{V}(\Phi) \neq \mathrm{T}$ and $\mathcal{V}(\Psi) \neq \mathrm{T}$.	**Biconditional** ($\leftrightarrow$) If an open path contains an unchecked formula of the form $(\Phi \leftrightarrow \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first list Φ above Ψ, and at the end of the second list $\sim\Phi$ above $\sim\Psi$.
$\mathcal{V}(\Phi \leftrightarrow \Psi) = \mathrm{F}$ iff either $\mathcal{V}(\Phi) = \mathrm{T}$ and $\mathcal{V}(\Psi) \neq \mathrm{T}$, or $\mathcal{V}(\Phi) \neq \mathrm{T}$ and $\mathcal{V}(\Psi) = \mathrm{T}$.	**Negated Biconditional** ($\sim\leftrightarrow$) If an open path contains an unchecked formula of the form $\sim(\Phi \leftrightarrow \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first list Φ above $\sim\Psi$, and at the end of the second list $\sim\Phi$ above Ψ.

1.		$\surd$ $P \rightarrow (Q \,\&\, (R \vee S))$	Premise
2.		$\surd$ $\sim(P \rightarrow Q)$	Negation of conclusion
3.		P	2 $\sim\rightarrow$
4.		$\sim Q$	2 $\sim\rightarrow$
5.	$\sim P$	$\surd$ $Q \,\&\, (R \vee S)$	1 $\rightarrow$
6.	X 3, 5	Q	5 &
7.		$R \vee S$	5 &
8.		X 4, 6	

Both paths close, so there is no valuation which makes the premises but not the conclusion true. Hence the sequent is valid.

Notice that we closed the right branch without analyzing 'R ∨ S'. This is permissible. Any path may be closed as soon as a formula and its negation both appear on it. Analyzing 'R ∨ S' would have split this path into two new paths, but each of these still would have contained both 'Q' and '–Q' and hence each still would have closed. Closing a path as soon as possible saves work.

It also saves work to apply nonbranching rules first. When I began the tree, I had the choice of analyzing either 'P → (Q & (R ∨ S))' or '–(P → Q)' first. I chose the latter, because it is a negated conditional, and the negated conditional rule does not branch. If I had analyzed 'P → (Q & (R ∨ S))', which is a conditional, first, then I would have had to use the conditional rule, which does branch. Then when I analyzed '–(P → Q)', I would have had to write the results twice, once at the bottom of each open path. Analyzing 'P → (Q & (R ∨ S))' first is not wrong, but it requires more writing, as can be seen by comparing the resulting tree with the previous tree:

1.	√ P → (Q & (R ∨ S))		Premise
2.	√ –(P → Q)		Negation of conclusion
3.	–P	√ Q & (R ∨ S)	1 →
4.	P	P	2 –→
5.	–Q	–Q	2 –→
6.	X 3, 4	Q	3 &
7.		R ∨ S	3 &
8.		X 5, 6	

Yet this tree, though more complicated, gives the same answer as the first. There is, then, some flexibility in the order of application of the rules. But in general it is best where possible to apply nonbranching rules before the branching ones.

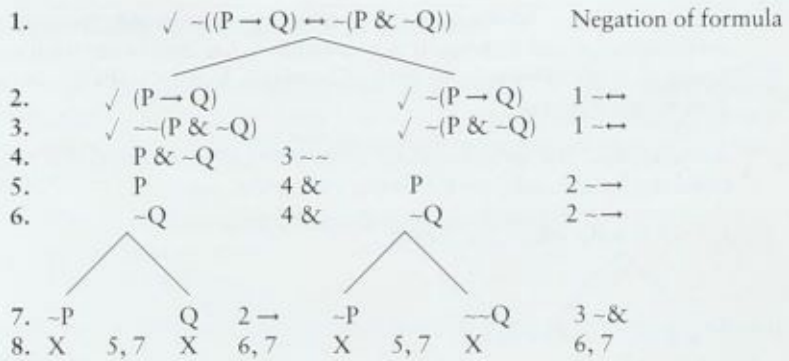
Let's next test the sequent '(P ↔ Q) ⊢ –(P ↔ R)' for validity. Once again we list the premises and the negation of the conclusion so that the tree searches for counterexamples. In this case the conclusion is a negation, so its negation is a double negative. Here is the tree:

1.		√ (P ↔ Q)			Premise
2.		√ ––(P ↔ R)			Negation of conclusion
3.		√ (P ↔ R)			2 ––
4.	P		–P		1 ↔
5.	Q		–Q		1 ↔
6.	P	–P	P	–P	3 ↔
7.	R	–R	R	–R	3 ↔
8.		X 4, 6	X 4, 6		

The leftmost and rightmost branches remain open. The leftmost branch reveals that the premise '(P ↔ Q)' is true and the conclusion '~(P ↔ R)' false (because its negation is true) on the valuation on which 'P', 'Q', and 'R' are all true. This valuation, in other words, is a counterexample to the sequent. The rightmost branch reveals that the valuation on which 'P', 'Q', and 'R' are all false is also a counterexample. Thus the sequent is invalid.

If we begin a tree, not with premises and a negated conclusion, but with a single formula or set of formulas, as we did in the first examples of this section, the tree tests this formula or set of formulas for consistency. If all paths close, there is no valuation on which the formula or set of formulas is true, and so that formula or set is inconsistent. If one or more paths remain open after the tree is finished, these represent valuations on which the formula or all members of the set are true, and so the formula or set is consistent.

Trees may also be used to test *formulas* for validity. The easiest way to do this is to search for valuations on which the formula is not true. If no such valuations exist, then the formula is valid. Thus we begin the tree with the formula's negation. If all paths close, there are no valuations on which its negation is untrue so that the original formula is true on all valuations. Consider, for example, the formula '(P → Q) ↔ ~(P & ~Q)'. When we negate it and do a tree, all paths close:



Therefore, since there is no valuation on which '~((P → Q) ↔ ~(P & ~Q))' is true, '(P → Q) ↔ ~(P & ~Q)' is true on all valuations, that is, valid.

Notice that I closed the rightmost path before '~~Q' was fully analyzed. This is a legitimate use of the negation rule. If the path had not closed, however, the tree would not be finished until the negated negation rule was applied to '~~Q'.

Here is a list of some of the ways in which trees may be used to test for various semantic properties:

To determine whether a sequent is valid, construct a tree starting with its premises and the negation of its conclusion. If all paths close, the sequent is valid. If not, it is invalid and the open paths display the counterexamples.

To determine whether a formula or set of formulas is consistent, construct a tree starting with that formula (or set of formulas). If all paths close, that formula (or set of formulas) is inconsistent. If not, it is consistent, and the open paths display the valuations that make the formula (or all members of the set) true.

To determine whether a formula is valid, construct a tree starting with its negation. If all paths close, the formula is valid. If not, then the formula is not valid, and the open paths display the valuations on which it is false.

To determine whether a formula is contingent, construct two trees, one to test it for consistency and one to test it for validity. If the formula is consistent but not valid, then it is contingent.

Constructing trees is just a matter of following the rules, but there are a few common errors to avoid. Keep these in mind:

The rules for constructing trees apply only to whole formulas, not to their parts. Thus, for example, the use of ~~shown below is not permissible:

1. √ P → ~~Q Given
2. √ P → Q 1 ~~ **(Wrong!)**

Although using ~~ on subformulas does not produce wrong answers, it is never necessary and technically is a violation of the double negation rule. Trying to apply other rules to parts of formulas, however, often does produce wrong answers.

A rule applied to a formula cannot affect paths not containing that formula. Consider, for example, the following incomplete tree:

1. √ P ∨ (Q ∨ R) Given

2. P 1 ∨ Q ∨ R 1 ∨

Here the formula 'Q ∨ R' at the end of the right-branching path remains to be analyzed. The next step is to apply the disjunction rule to this formula. In doing so, we split this right-branching path but add nothing to the path at the left, for it does not contain 'Q ∨ R' and is in fact already finished.

The negation rule applies only to formulas on the same path. In the following tree, for example, both 'P' and '~P' appear, but neither path closes because the formulas don't appear on the same path:

1. √ P → P Given

2. ~P 1 → P 1 →

To summarize: A finished tree for a formula or a set of formulas displays all the valuations on which that formula or all members of that set are true. Thus trees do the same work as truth tables, but in most cases they do it more efficiently. Moreover, as we shall see in later chapters, they may be used for some logics to which truth tables are inapplicable.

Exercise 3.3.1

Redo Exercises 3.2.1, 3.2.2, and 3.2.3 using trees instead of truth tables.

Exercise 3.3.2

1. How might trees be used to prove that two formulas are logically equivalent? Explain.
2. To prove a formula valid using trees, we construct a tree from its negation. Is there a way to prove a formula valid by doing a tree on that formula without negating it? Explain.

3.4 VALUATIONS AND POSSIBLE SITUATIONS

We saw in Section 2.1 that while any instance of a valid form is a valid argument, not every instance of an invalid form is an invalid argument. We noted, for example, that this instance of the invalid sequent affirming the consequent is in fact a valid argument:

If some men are saints, then some saints are men.
Some saints are men.
∴ Some men are saints.

How can this be? The answer lies in the distinction between valuations and possible situations.

Suppose we let 'S_1' stand for 'Some men are saints' and 'S_2' for 'Some saints are men'. Then we can represent the form of the argument as '$S_1 \rightarrow S_2, S_2 \vdash S_1$'. Here is its truth table:

S_1	S_2	$S_1 \rightarrow S_2$,	S_2	$\vdash S_1$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	F
F	F	T	F	F

The valuation in which 'S_1' is false and 'S_2' true is a counterexample to the *sequent* or *argument form*. But the corresponding situation—the one in which 'Some men

are saints' is false and 'Some saints are men' is true—isn't a counterexample to the *argument* because it isn't a *possible* situation. The very idea of a situation in which some men are saints but it is not the case that some saints are men (i.e., no saints are men) is nonsense. Of course we can easily find other interpretations of 'S_1' and 'S_2'—and consequently other instances of this form—to which the valuation that makes 'S_1' false and 'S_2' true provides a genuine counterexample, even an *actual* counterexample. But on this particular interpretation, that valuation corresponds to an *im*possible situation. (Incidentally, so does the valuation that makes 'S_1' true and 'S_2' false.) An *im*possible situation, if it even makes sense to talk about such a thing, cannot be a counterexample.

Depending on how we interpret the sentence letters, then, a particular valuation may or may not correspond to a possible situation. For many interpretations, all valuations correspond to possible situations. For example, if we let 'S_1' stand for 'It is sunny' and 'S_2' for 'It is Sunday', every line on the truth table above (every valuation) represents a possible situation, and the valuation on which 'S_1' is false and 'S_2' true represents a counterexample to the argument as well as to the form. In such cases, the statements corresponding to the sentence letters are said to be **logically independent.** But where the statements corresponding to the sentence letters logically imply one another or exclude one another, in various combinations, some valuations represent impossible situations.

Such nonindependent statements as 'Some men are saints' and 'Some saints are men' have interrelated semantic structures that are not represented in propositional argument forms in which they are symbolized simply as sentence letters. (In this case, the semantic structures in question are relationships among the logical meanings of the words 'some', 'men', and 'saints'.) In Chapter 6 we shall begin to formalize the semantic structures of such statements, and we shall redefine the notion of a valuation so that it reflects more of these semantic structures and yields a more powerful and precise logic. Later we shall explore ways of creating logics that are more powerful and precise still. But at no point shall our concept of a valuation become so sophisticated that a valuation may never represent an impossible situation—which is to say that at no point do we ever achieve a formal semantics or formal logic that reflects all the logical dependencies inherent in natural language.

Certain consequences of this disparity between valuations and possible situations, between formal and informal logic, will haunt us throughout this book:

An invalid sequent may have valid instances. The reason for this we have already seen. The counterexamples to the sequent may on some interpretations represent impossible situations so that there are no possible situations which make the corresponding argument's premises true while its conclusion is untrue. No argument is valid *because of* having an invalid form, but an argument may be valid *in spite of* having an invalid form, because of elements of its semantic structure not represented in the form.

A contingent formula may have valid or inconsistent instances. A contingent formula is true on some valuations and false on others. But on some

interpretations either the valuations on which the contingent formula is true or those on which it is false may all correspond to impossible situations. In the former case, the interpretation yields an inconsistent instance. In the latter, provided that at least one of the valuations on which the formula is true corresponds to a possible situation, the interpretation yields a valid instance. Example: 'P & Q' is a contingent formula, but the instance 'Some women are mortal & Nothing is mortal' is an inconsistent statement, and the instance 'Every woman is a woman & Every mortal is a mortal' is a valid statement (logical truth). (Check the truth table of 'P & Q' to see which valuations correspond to impossible situations in each case.)

A consistent formula or set of formulas may have inconsistent instances. That is, though there is a valuation that makes the formula or set of formulas true, there may not be a possible situation that makes a particular instance of that formula or set of formulas true, again because the situation corresponding to that valuation may be impossible. Example: The set consisting of the formulas 'P' and 'Q' is consistent, but if we interpret 'P' as 'Smoking is permitted' and 'Q' as 'Smoking is forbidden', the set of statements for which these letters stand is inconsistent.

All of this sounds discouraging. Nevertheless:

All instances of a valid sequent are valid arguments. A valid sequent has no valuation on which its premises are true but its conclusion is not true. Some valuations may on a particular interpretation correspond to impossible situations. Yet since a valid sequent has no valuations representing situations (possible or impossible) in which the premises are true and the conclusion is false, none of its valuations represents a possible situation that is a counterexample to the instance. Hence, if a sequent is valid, all of its instances must be valid as well. Valid sequents are, in other words, perfectly reliable patterns of inference.

All instances of a valid formula are logical truths. A valid formula is a formula true on all valuations. Even if on a given interpretation some valuations of such a formula do not represent possible situations, the formula is still true on all the others and hence true in all the situations that are possible. Therefore any statement obtained by interpreting a valid formula must be true in all possible situations. That is, it must be a logical truth.

All instances of an inconsistent formula are inconsistent statements. An inconsistent formula is true on no valuations. Hence, even if on a given interpretation some of its valuations represent impossible situations, still the formula is true on none of the remaining valuations which represent possible situations. Therefore any statement obtained by interpreting an inconsistent formula is not true in any possible situation.

Under the same interpretation, logically equivalent formulas have as their instances logically equivalent statements. Logically equivalent formulas are formulas whose truth value is the same on all valuations. Once again, even

if a given interpretation rules out some of these valuations as impossible, still the formulas will have the same truth value in the remaining valuations—the ones representing possible situations. Hence any two statements obtained by interpreting them will have the same truth values in all possible situations. That is, they will be equivalent statements.

To summarize: Formal validity (for both formulas and sequents), inconsistency, and equivalence are reliable indicators of their informal counterparts. Formal invalidity, contingency, and consistency are not.

CHAPTER 4

CLASSICAL PROPOSITIONAL LOGIC: INFERENCE

4.1 CHAINS OF INFERENCE

Most people can at best understand arguments that use about three or four premises at once. For more complicated arguments, we generally break the argument down into more digestible chunks. Beginning with one or two or three premises, we draw a subconclusion, which functions as a stopping point on the way to the main conclusion the argument aims to establish. This subconclusion summarizes the contribution of these premises to the argument so that they may henceforth be forgotten. This subconclusion is then combined with a few more premises to draw a further conclusion, and the process is repeated, step by small step, until the final conclusion emerges. The following example illustrates the utility of breaking complex inferences down into smaller ones:

> The meeting must be held on Monday, Wednesday, or Friday.
> At least four of these five people must be there: Al, Beth, Carla, Dave, and Em.
> Em can't come on Monday or Wednesday.
> Carla and Dave can't both come on Monday or Friday, though either of them could come alone on those days.
> Al can come only on Monday and Friday.
> ∴ The meeting must be held on Friday.

may prove a sequent to be valid, but proofs are not designed to reveal invalidity. If we try to prove a sequent and fail, that does not show the sequent is invalid. Maybe we did not try hard enough. Thus, given a sequent whose validity is in question, trying to prove it settles the question only if the sequent is valid and we do in fact find a proof.

Moreover, because proof rules can be applied or axioms introduced in any order, they may be applied repeatedly without ever reaching the desired conclusion, even if that conclusion validly follows from the premises. Proof, in other words, may elude us, even though proofs exist.

Because of these disadvantages, this book does not emphasize proofs, though they are the most familiar and widely recognized way of doing logic. We shall concentrate instead on the more powerful semantic techniques—valuation rules, truth tables, and trees—and their generalizations for more advanced logical systems.

Exercise 4.5.1

Prove instances of Ax1, Ax2, and Ax3 as theorems within our natural deduction system.

Exercise 4.5.2

Prove the following sequents within the axiom system presented in this section:

1. $\vdash P \rightarrow P$
2. $\vdash (\neg P \rightarrow P) \rightarrow P$
3. $P, \neg P \vdash Q$
4. $P \rightarrow (Q \rightarrow R), Q \vdash P \rightarrow R$

CHAPTER 5

CLASSICAL PROPOSITIONAL LOGIC: METATHEORY

5.1 INTRODUCTION TO METALOGIC

Metalogic is the logical study of formal logical systems. In metalogic we study one or more formal languages (e.g., the language of propositional logic) which, being the objects of our study, are called **object languages.** We must, of course, use language to talk about an object language, but the language we use is usually not the object language itself. We call it the **metalanguage.** Virtually everything said in a logic textbook, except for the problems and exercises, is formulated in the metalanguage. Our metalanguage is English augmented with an assortment of variables (e.g., Greek letters) and other technical devices.

Chapter 4 covered proofs formulated and carried out in an object language, the language of propositional logic. The proofs we shall construct in this chapter will be proofs about the object language, formulated and carried out in the metalanguage. The conclusions of these proofs are called **metatheorems,** and the proofs themselves are called **metaproofs.** Reasoning in metaproofs often mirrors reasoning in the object language. Similar inference rules (e.g., modus ponens) may be used, though usually they are used without comment, rather than being explicitly cited and annotated. Over the years, logicians have developed a peculiar style and rhetoric for expressing metatheorems. Part of what you will be learning here is that style and rhetoric.

To prove a metatheorem, we begin with a set of premises and construct a proof linking them to some desired conclusion, just as in a formal language. In a typical exercise in propositional logic, however, both the premises and the conclusion are explicitly given. For metatheorems, only the conclusion is given; you are not told exactly which premises to use. Part of your work is to decide which premises are needed. Premises for metatheorems are typically definitions or previously proved metatheorems. Occasionally some principles of arithmetic or algebra are also used. It's usually easy to tell which definitions to use; they will nearly always be the definitions of the concepts employed in the metatheorem. The definitions we will need to prove metatheorems in this chapter are the ones introduced in Chapters 2–4.

Consider, for example, how to prove the elementary metatheorem that '~~P' is a formula (of the language of propositional logic). The main concept used in this metatheorem is the concept of a formula. So we need to locate the definition of a formula, which will function as a premise in the proof. The appropriate definition is embodied in the formation rules (see Section 2.3). From rule 1 it follows that 'P' is a formula. And since 'P' is a formula, by rule 2 '~P' is also a formula. Once we have shown that '~P' is a formula, it follows, again by rule 2, that '~~P' is a formula—so we have our conclusion. Notice that the only premises used were rules 1 and 2, and that rule 2 was used twice. Here's the proof in good metatheoretic style:

> **METATHEOREM:** '~~P' is a formula.
>
> **PROOF:** By formation rule 1, 'P' is a formula, whence it follows by rule 2 that '~P' is a formula, and again by rule 2 that '~~P' is a formula. QED

The letters 'QED' at the end stand for *quod erat demonstrandum*, a Latin phrase meaning "which was to be proved." This is the logician's equivalent of a high five. When you finish proving a metatheorem, writing these letters gives you a little rush.

Metatheorems differ greatly in form and content. Here is another simple metatheorem that uses the valuation rules, rather than the formation rules:

> **METATHEOREM:** 'P → ~P' is consistent.
>
> **PROOF:** Since (by the definition of a valuation), a valuation is simply an assignment of one of the values T or F to the sentence letters of a formula, there is a valuation $\mathcal{V}$ of the formula 'P' such that $\mathcal{V}$('P') = F and hence $\mathcal{V}$('P') ≠ T. By valuation rule 4 (the rule for the conditional), if $\mathcal{V}$('P') ≠ T, then $\mathcal{V}$('P → ~P') = T. Hence there is a valuation (namely

$\mathcal{V}$) on which 'P → ~P' is true. It follows (by the definition of consistency) that 'P → ~P' is consistent. QED

This metatheorem simply puts into words what a truth table or tree would reveal. But for that very reason it may be useful as an illustration of metatheoretic style. Simple metatheorems may just be summaries or reminders of what we already know; that is the case, too, with the next one.

But this next metatheorem is more general in scope. It combines the definition of a valuation, the formation rules, and the valuation rules to get the conclusion that all formulas have exactly one of the values T or F on each of their valuations:

METATHEOREM (Bivalence): Each formula of propositional logic is either true or false, but not both, on each of its valuations.

PROOF: Consider any formula Φ of propositional logic and any valuation $\mathcal{V}$ of Φ. Since Φ is a formula, Φ is either atomic or complex. If Φ is atomic, then the definition of a valuation stipulates that $\mathcal{V}$ assigns it one, but not both, of the values T or F. If Φ is complex, then by formation rules 2 and 3 it must have one of five forms: $\sim\Phi$, $(\Phi \,\&\, \Psi)$, $(\Phi \vee \Psi)$, $(\Phi \rightarrow \Psi)$, or $(\Phi \leftrightarrow \Psi)$. Now the valuation rule for each of these forms stipulates that $\mathcal{V}$ assigns Φ the value F iff $\mathcal{V}$ does not assign Φ the value T.[1] No matter whether Φ is atomic or complex, then, $\mathcal{V}$ assigns to Φ one, but not both, of the values T or F. QED

Though simple, this metatheorem is important for it confirms that our semantics is bivalent—that is, classical.

Exercise 5.1

Prove the following metatheorems:

1. '(P → (Q ∨ (R ∨ ~S)))' is a formula.
2. '(P → ~P)' is true on the valuation in which 'P' is false.

[1] The valuation rule for conjunction, for example, is:

$\mathcal{V}(\Phi \,\&\, \Psi) = \text{T}$ iff both $\mathcal{V}(\Phi) = \text{T}$ and $\mathcal{V}(\Psi) = \text{T}$;
$\mathcal{V}(\Phi \,\&\, \Psi) = \text{F}$ iff either $\mathcal{V}(\Phi) \neq \text{T}$ or $\mathcal{V}(\Psi) \neq \text{T}$, or both.

The conditions under which $\mathcal{V}(\Phi \,\&\, \Psi) = \text{F}$ are precisely those under which it is not the case that $\mathcal{V}(\Phi \,\&\, \Psi) = \text{T}$. (Check that the other rules imply that complex formulas of other forms too are false iff they are not true.)

5.2 CONDITIONAL PROOF

In formal logic, different kinds of proofs require different strategies. The same is true in metalogic. Indeed, the same principles of strategy apply to both kinds of reasoning. In both cases, strategy is governed mainly by the structure of the conclusion. For example, if the conclusion is a conditional statement (i.e., a statement of the form $\Phi \rightarrow \Psi$), then the best strategy is usually **conditional proof.** The object language version of conditional proof was covered in Section 4.3, where it is called conditional introduction ($\rightarrow$I). Here we discuss conditional proof in the metalanguage.

In a conditional proof we suppose the antecedent Φ for the sake of argument and use it, perhaps together with other assumptions, to derive the consequent Ψ. The argument in which we derive Ψ is **hypothetical,** in the sense that it depends on the supposition of the antecedent Φ, which we need not assert to be true. Because of its hypothetical character, I like to think of this argument as a kind of fiction. However, if we succeed in validly deriving the consequent Ψ from the antecedent Φ, then we certainly know this:

> *If* Φ is true, then Ψ is true.

That is, we know that $\Phi \rightarrow \Psi$. Since we know this whether or not Φ is true, the conclusion $\Phi \rightarrow \Psi$, unlike the hypothetical conclusion Ψ, does not depend on the truth of the supposition Φ. Hence we are entitled to **discharge** this supposition—that is, not to regard it as one of the assumptions on which the proof of $\Phi \rightarrow \Psi$ rests. (It was, after all, made only "for the sake of argument.")

Formal systems usually have some notational device to indicate the discharging of suppositions, and some even have ways of setting off the hypothetical argument. In the system of Chapter 4, the discharging of suppositions is indicated by the ending of the line to the left of the hypothetical derivation. Metatheory dispenses with these devices. I find it helpful, however, to adopt the convention of indenting the proof when a supposition is introduced and ending the indentation when it is discharged. The hypothetical argument is thus set off clearly from the rest of the argument. This convention is consistently employed below.

Let's consider a simple example of a metatheorem employing a conditional proof strategy. The metatheorem is this:

> If the set of premises of a valid sequent is consistent, then so is the conclusion.

This metatheorem is a conditional statement, so to prove it we will use conditional proof. We begin by supposing the antecedent:

> (A) The set of premises of a valid sequent is consistent.

Our immediate goal is to prove the consequent:

> (C) The conclusion of the sequent is consistent.

If we can do so, then we can discharge the supposition (A) and assert the desired conditional statement (A) → (C).

The problem now becomes how to derive (C) from (A). The argument is not immediate, so we need some additional assumptions.[2] As noted above, these are likely to be definitions of major terms used in the metatheorem. Two major terms used here are 'valid' and 'consistent'. The latter is used in two senses; statement (A) concerns the consistency of a set of object language formulas, and statement (C) concerns the consistency of a single object language formula, the conclusion. So the next step is to look up the definitions of these terms. These definitions are as follows:

(1) A sequent is valid iff there is no valuation on which its premises are true and its conclusion is not true.
(2) A set of formulas is consistent iff there is at least one valuation in which all members of the set are true.
(3) A single formula is consistent iff there is at least one valuation in which it is true.

(These definitions are statements in the metalanguage concerning sequents or formulas of the object language.)

Now the path of reasoning from (A) to (C) is easy to see. Since the sequent in question is valid, by (1) there is no valuation on which its premises are true and its conclusion is not true. But since the set of its premises is consistent, by (2) there is at least one valuation on which these premises are all true. Hence on that valuation (if no other) the sequent's conclusion is not untrue—that is, it is true. But, then by definition (3), the sequent's conclusion is consistent—which is the conclusion (C) that we were trying to prove.

That, of course, was just the hypothetical argument. Having completed the hypothetical argument, we still need to discharge supposition (A) and assert our conclusion (A) → (C). Here's what it looks like when we assemble the pieces:

METATHEOREM: If the set of premises of a valid sequent is consistent, then so is the conclusion.

PROOF: Suppose (for conditional proof) that the set $\Phi_1, \ldots, \Phi_n$ of premises of some valid sequent with conclusion Ψ is consistent. Then (by the definition of consistency for a set) there is at least one valuation $\mathcal{V}$ on which $\Phi_1, \ldots, \Phi_n$ are all true. But (by the definition of validity) there is no valuation on which $\Phi_1, \ldots, \Phi_n$ are all true and Ψ is not true. Thus Ψ is not untrue

[2] By the way, be sure to keep straight which argument we are talking about at which time. We are talking about two arguments: a metatheoretical one that we are constructing and a formal one, technically a sequent, in the object language that the metatheoretical argument is about. Metatheory is always working on two levels like this, and that's one of the things that makes metatheory difficult.

on $\mathcal{V}$ and so must be true on $\mathcal{V}$. Hence (by the definition of consistency) Ψ is consistent.
Therefore, if the set of premises of a valid sequent is consistent, then so is the conclusion. QED

The indented part is the hypothetical argument, the "logical fiction." We need not know or care whether there is any such object language sequent as we are supposing here (though in this case, of course, there are many). The hypothetical argument takes us from the antecedent (A) to the consequent (C) of our conclusion. Then the antecedent is discharged (indicated by ending the indentation) and the conditional conclusion is asserted to complete the proof.

I have used parenthetical remarks to indicate where definitions are invoked in the proof. In most metatheoretical writing, these remarks would be omitted; it is assumed that the reader is sophisticated enough to realize that the argument is by definition. But it helps when you are learning metatheory to remind yourself of what you are doing by incorporating such remarks.

Notice how I used Greek letters as variables in the metalanguage. Such variables provide clear reference and help to condense the prose. I used Greek because I didn't want the variables that belong to the metalanguage to be confused with the P's, Q's, and so on that are part of the formal object language we are talking about. If, for example, I had used the letter

P

instead of

Ψ

in the metatheorem, someone might have thought the conclusion was meant specifically to be the atomic formula 'P'. The metatheorem, however, does not specify the form of the conclusion. To designate object language formulas without specifying their identity, we need special variables in our metalanguage. And since we don't want to confuse these **metavariables**—that is, variables of the metalanguage—with object language formulas, it's best to use a wholly distinct alphabet. That's the reason for the Greek. When no such confusion could arise, however, we will sometimes use the more familiar English letters as metavariables.

The purpose of the notation '$\Phi_1, \ldots, \Phi_n$' should be clear. This is just a way of designating a list of some unspecified number (n) of formulas. We stipulate that this notation allows the possibility that the list has no members so that n might be 0, unless otherwise specified. If $n = 0$, then by convention every valuation makes $\Phi_1, \ldots, \Phi_n$ true, since there is nothing to make true. (The convention in question is a convention that governs universally quantified statements—that is, statements about *all* members or *every* member of a class—in classical logic generally. In this case the particular statement at issue—'Every valuation makes $\Phi_1, \ldots, \Phi_n$ true'—is a statement of the metalanguage. Further explanation of this convention must await a full treatment of the semantics of quantifiers, which is given in Section 7.2.)

One final point about this metatheorem: Its hypothetical argument follows a pattern that is very common in metalogical reasoning:

Unpacking — Logical manipulation — Repacking

Unpacking means replacing the terms given in the metatheorem (in this case in its antecedent) with their definitions. The following statements in the proof constituted the unpacking:

> Then (by the definition of consistency for a set) there is at least one valuation $\mathcal{V}$ on which $\Phi_1, \ldots, \Phi_n$ are all true. But (by the definition of validity) there is no valuation on which $\Phi_1, \ldots, \Phi_n$ are all true and Ψ is not true.

When you begin to prove metatheorems, it is best to look up relevant definitions as you unpack and to word your unpacking as nearly like the definition as is possible. This prevents errors.

One common error in unpacking is to introduce the symbol $\mathcal{V}$ without specifying whether it stands for some particular valuation (as in the preceding example) or for all valuations. Always be sure to specify what it stands for.

Moreover, remember that there is no such thing as truth per se in formal logic. That is, a formula is never merely true or false; it is true or false on *some* valuation or, perhaps, on *all* valuations. Make sure as you unpack that with each mention of truth or falsity you specify the valuation(s) to which it applies.

Once the given terms are unpacked, a conclusion is drawn from them by logical inference. This is the stage of logical manipulation:

> Thus Ψ is not untrue on $\mathcal{V}$ and so must be true on $\mathcal{V}$.

The logical step here is simply the elimination of a double negation. The final stage, repacking, puts the newly derived conclusion back into defined terms:

> Hence (by the definition of consistency) Ψ is consistent.

Watch for this pattern, and imitate it where appropriate. It is common not only in conditional proof but in many other forms of metalogical reasoning as well.

Some metatheorems are biconditional in form. Since a biconditional is, in effect, simply two conditionals, proofs of biconditional metatheorems often take the form of two conditional proofs. That is the case in the next metatheorem:

METATHEOREM: A sequent is valid if and only if the set containing its premises and the negation of its conclusion is inconsistent.

PROOF: Suppose (for conditional proof) that $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is a valid sequent. Then (by the definition of validity) there is no valuation in which its premises $\Phi_1, \ldots, \Phi_n$ are all true and its conclusion Ψ is not true. From this it follows by valuation rule 1 that there is no valuation on which $\Phi_1, \ldots, \Phi_n$ and $\neg\Psi$ are all

uses negation → go into this more

true—that is, (by the definition of consistency) that the set $\{\Phi_1, \ldots, \Phi_n, \sim\Psi\}$ is inconsistent.

Hence we have shown that if a sequent is valid, then the set consisting of its premises and the negation of its conclusion is inconsistent.

Now suppose (again for conditional proof) that the set $\{\Phi_1, \ldots, \Phi_n, \sim\Psi\}$ is inconsistent. This means that there is no valuation on which $\Phi_1, \ldots, \Phi_n$ and $\sim\Psi$ are all true. Hence (by valuation rule 1) there is no valuation on which $\Phi_1, \ldots, \Phi_n$ are true and Ψ is not true, which is to say that the sequent $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is valid.

Thus, if the set containing a sequent's premises and the negation of its conclusion is inconsistent, then that sequent is valid. In summary, we have shown that a sequent is valid *if and only if* the set containing its premises and the negation of its conclusion is inconsistent. QED

The proof consists of two conditional proofs, whose conclusions are assembled into the biconditional at the end. Again, I am saying more here than the usual sparse metalogical style permits. At most what is actually written when this kind of proof appears in a journal article is the two hypothetical arguments (indicated here by the indentations). The rest would be understood as implicit. With this example it is also possible to combine the two proofs into a series of biconditional inferences—something like this:

PROOF: $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is a valid sequent iff there is no valuation in which its premises $\Phi_1, \ldots, \Phi_n$ are all true and its conclusion Ψ is false. But this is the case iff there is no valuation on which $\Phi_1, \ldots, \Phi_n$ and $\sim\Psi$ are all true, that is, iff $\{\Phi_1, \ldots, \Phi_n, \sim\Psi\}$ is inconsistent. QED

This style of proof is fairly common. The curly brackets '{' and '}' are the conventional marks used to indicate the members of a set. We will employ this convention from now on. For more on sets, see Section 7.1.

Exercise 5.2

Prove the following metatheorems by conditional proof:

1. If formula Φ is valid, then $\sim\Phi$ is inconsistent.
2. If Φ and Ψ are formulas and Φ is inconsistent, then $(\Phi \,\&\, \Psi)$ is inconsistent.
3. If $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is a valid sequent whose premises $\Phi_1, \ldots, \Phi_n$ are all valid, then its conclusion Ψ is also a valid formula.
4. If the set of premises of a sequent is inconsistent, then that sequent is valid.
5. A formula Φ is inconsistent if and only if $\sim\Phi$ is a valid formula.

5.3 REDUCTIO AD ABSURDUM

Conditional proof is a very common proof strategy in metatheorems. It may be used whenever a metatheorem is conditional in form. However, not all metatheorems are conditionals, and so some require other strategies. Another common strategy is **reductio ad absurdum**—also called **indirect proof.** The object language version of this strategy, which is embodied in the negation introduction rule (–I), was covered in Section 4.3. This is a powerful technique, which is used primarily in proving negative conclusions but may be used with conclusions of any form.

The trick of a reductio is to suppose the denial of the conclusion you want to prove and then show that that supposition validly implies a contradiction. Typically the contradiction will be a metalinguistic statement of the form (Φ and not-Φ), but occasionally other sorts of contradictions are used—for example, the arithmetic statement '0 = 1' or a statement to the effect that a thing is not identical to itself. Now any supposition which—perhaps together with other statements that are given as true—validly implies a contradiction must be false. For by the definition of validity it is impossible for all the premises of a valid inference to be true while its conclusion is false. But a contradictory conclusion is certainly false. Hence at least one of the premises that validly implies it must be false. Therefore, given that the other premises used to derive the contradiction are true, we may discharge the supposition (which, recall, was the denial of our intended conclusion) and assert this intended conclusion itself.

As in a conditional proof, the argument from the supposition to the contradiction is hypothetical; it is a fiction based on a supposition which we needn't believe. (In a reductio we certainly don't believe our supposition, since what we are trying to prove is precisely its opposite!) And, as in conditional proof, we will set off this fiction by indenting it. Here is a simple metatheorem whose proof uses a reductio strategy:

METATHEOREM: There is no invalid sequent with an inconsistent set of premises.

PROOF: Suppose for reductio that there is an invalid sequent with an inconsistent premise set $\{\Phi_1, \ldots, \Phi_n\}$. Since the sequent is invalid, there is (by the definition of invalidity) some valuation $\mathcal{V}$ on which $\Phi_1, \ldots, \Phi_n$ are all true and the argument's conclusion is not true. But since $\Phi_1, \ldots, \Phi_n$ are true on $\mathcal{V}$, $\{\Phi_1, \ldots, \Phi_n\}$ is consistent (by the definition of consistency for a set), which contradicts our supposition.

Consequently, there is no invalid argument with an inconsistent set of premises. QED

The hypothetical argument begins with the supposition of the denial of the intended conclusion. This is shown by the hypothetical argument to lead to the contradiction that $\{\Phi_1, \ldots, \Phi_n\}$ is both consistent and inconsistent. So the supposition is discharged (ending the indentation), and the desired conclusion is asserted as proved.

Exercise 5.3

Prove the following metatheorems by reductio ad absurdum:

1. There is no valid sequent with a consistent set of premises and an inconsistent conclusion.
2. There is no formula Φ such that Φ and $\neg\Phi$ are each inconsistent.
3. There is no invalid sequent with a valid conclusion.
4. The formula '(P & ~P)' is inconsistent.
5. The form 'P ⊢ P ∨ Q' is valid.
6. '(P)' is not a formula. (Hint: Suppose for reductio that '(P)' is a formula; then it must have been constructed only by successive application of the three formation rules so that one of these rules must be the last to have been applied. You can then contradict this supposition by showing for each of the formation rules that it could not have been the last rule used to construct something of the form '(P)'.)
7. Any sequent of the form $\Phi \vdash \Phi$ is valid.

5.4 MIXED STRATEGIES

The metatheorems we have considered so far are extremely simple. More interesting metatheorems use several different strategies, one nested inside another. For example, in deriving the consequent from the antecedent in the hypothetical argument of a conditional proof, we might need to use a reductio strategy so that we nest a reductio argument inside a conditional proof. Here is an example:

METATHEOREM: If the conclusion of one valid sequent is Φ and the conclusion of a second valid sequent is $\neg\Phi$, then the set consisting of all the premises of both sequents is inconsistent.

PROOF: Suppose for conditional proof that the conclusion of one valid sequent is Φ and the conclusion of a second valid sequent is $\neg\Phi$.

> Now suppose for reductio that the set consisting of all the premises of both sequents is consistent. That is (by the definition of consistency for sets), there is some valuation $\mathcal{V}$ which makes each member of this set true.

> Then all the premises of both sequents are true on $\mathcal{V}$; and, since both sequents are valid, it follows by the definition of validity that neither the conclusion Φ nor the conclusion $\neg\Phi$ is untrue on $\mathcal{V}$. Therefore both $\mathcal{V}(\Phi) = T$ and $\mathcal{V}(\neg\Phi) = T$. But since $\mathcal{V}(\Phi) = T$, by valuation rule 1, $\mathcal{V}(\neg\Phi) \neq T$, and so we have a contradiction.
>
> Thus, contrary to our reductio supposition, the set consisting of all the premises of both sequents is inconsistent.
>
> So, if the conclusion of one valid sequent is Φ and the conclusion of a second valid sequent is $\neg\Phi$, then the set consisting of all the premises of both sequents is inconsistent. QED

The trick in proving this metatheorem is to pay careful attention to the form of the conclusion, that is, the metatheorem itself. The metatheorem is a conditional whose antecedent is

(A) The conclusion of one valid sequent is Φ and the conclusion of a second valid sequent is $\neg\Phi$.

and whose consequent is

(C) The set consisting of all the premises of both sequents is inconsistent.

So to prove this conditional, we suppose (A) for conditional proof (thus beginning the proof with an indentation to indicate that we are engaged in a logical fiction) and from (A) derive (C). Then we discharge (A) and assert the conditional conclusion (the last statement of the proof).

But how can we derive (C) from (A)? The clue to follow here is that (C) is a negative statement; it says that a certain set is inconsistent, that is, not consistent. Negative conclusions are usually best proved by reductio. So within the hypothetical argument of the conditional proof, we use a reductio strategy. We thus suppose the denial of (C) for reductio. This is our second supposition, so we indent a second time; we are now engaged in a "fiction within a fiction"—something like the play performed inside Shakespeare's comedy *A Midsummer Night's Dream*. We then proceed from (A) by simple definition, as in the previous examples, and a contradiction follows quickly. This contradiction brings the "inner" fiction to an end. We discharge the reductio supposition and conclude that its negation is true. But this negative conclusion is precisely the conclusion (C) that we were aiming for.

Exercise 5.4

Prove the following metatheorems by using reductio arguments inside conditional proofs:

1. If the conclusion of a valid sequent is inconsistent, then the set of premises is inconsistent as well.

2. If the conclusion Ψ of a sequent $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is valid, then the sequent itself is valid.
3. If new premises $\Phi_{n+1}, \ldots, \Phi_m$ are added to a valid sequent $\Phi_1, \ldots, \Phi_n \vdash \Psi$, then the resulting sequent $\Phi_1, \ldots, \Phi_n, \Phi_{n+1}, \ldots, \Phi_m \vdash \Psi$ is valid.
4. If $\Phi_1, \ldots, \Phi_m \vdash \Psi$ and $\Psi, \Psi_1, \ldots, \Psi_n \vdash \Theta$ are valid sequents, then $\Phi_1, \ldots, \Phi_m, \Psi_1, \ldots, \Psi_n \vdash \Theta$ is a valid sequent.

5.5 MATHEMATICAL INDUCTION

The final metatheoretic strategy that we will consider is mathematical induction. The name is really a misnomer; mathematical induction is a form of deductive reasoning, not inductive reasoning. In fact, it's just an iterated form of modus ponens. Mathematical induction is used when we want to prove that each member of a series of items has a certain property. A series is a linear list, in which there is a first item, a second item, and so on. Mathematical induction works on both finite series (which have a last item) and infinite series (which do not). An infinite series is a series ordered like the natural numbers (the whole numbers beginning with 1); that is, it has a first item and each item of the series has a successor (e.g., after the second item there is a third), but the series itself never ends. Many items that concern us in logic can be arranged into series. For example, rule 2 of the formation rules generates an infinite series of negated formulas for each sentence letter so that from rule 2 alone we can see that there are infinitely many formulas. Using the sentence letter 'P', for example, we have the series

P, ~P, ~~P, ~~~P, . . .

(the dots indicate that the series continues infinitely). Let us call this series *S*.

Now suppose that we want to prove that each item of *S* has the property of being a formula. Of course, that's obvious from the formation rules, but we are concerned with how to give a proper proof of it. The proof requires mathematical induction. To prove by mathematical induction that each item of a series has a given property F, we prove two things:

(1) That the first item of the series has F and
(2) That for any n, if the nth item of the series has F, then so does the $(n+1)$st.

The proof of (1) is called the **basis case** of the induction; the proof of (2) is called the **inductive step.** If we can prove these two things, then our work is done, for together they logically imply the conclusion that every object in the series has the property, even if the series is infinite. To see this, note that (2) is a universal statement which implies each of the following instances:

If item 1 has F, then so does item 2.

If item 2 has F, then so does item 3.

If item 3 has F, then so does item 4.

. . . and so on.

But (1) tells us that item 1 has F. So by modus ponens, together with the first statement, item 2 has F. But then by modus ponens, together with the second statement, item 3 has F, and so on. Thus by infinitely many steps of modus ponens it follows that each item in the series has F. Of course we can't actually carry out infinitely many steps of modus ponens. That's why we have the special principle of mathematical induction. (We wouldn't need it if all we ever had to worry about were finite series.) This principle stipulates that if we have proved (1) and (2), we can conclude straightaway that each item of the series has F; we needn't bother with modus ponens. The validity of the principle is obvious.

In proofs by mathematical induction, the basis case is usually trivial. The inductive step justifies the universally quantified conditional (2). The strategy is always conditional proof. We suppose for conditional proof that some arbitrary *n*th item of the series has F (this supposition is called the **inductive hypothesis**) and prove from this supposition that the $(n + 1)$st item has F as well. That proves the conditional, and since the item considered was arbitrary, we can universally generalize the conditional.

Now in the problem we are considering, F is the property of being a formula. We want to prove that every item of series *S* has this property. The basis case must establish that 'P' is a formula (which follows immediately from formation rule 1), and the inductive step must show that if one item in the series is a formula, the result of prefixing it with a negation sign is also a formula (which follows immediately from formation rule 2). Hence the proof is easy. Here it is in proper metatheoretic form:

METATHEOREM: Each item of series *S* is a formula.

PROOF:

Basis Case: The first item of *S* is 'P', which (by formation rule 1) is a formula.

Inductive Step: Suppose that the *n*th item of *S* is a formula. (This is the inductive hypothesis; it initiates the conditional proof.) Now the $(n + 1)$st item is the result of prefixing the *n*th with a negation sign. Therefore (by formation rule 2 and the inductive hypothesis) the $(n + 1)$st item of *S* is a formula.

Thus (by conditional proof) it follows that if the *n*th item of *S* is a formula, then so is the $(n + 1)$st. Hence (by mathematical induction) each item of *S* is a formula. QED

Again, this is more explicit than the usual metatheoretic style. In professional writing, the labels 'Basis Case' and 'Inductive Step' and the parenthetical remarks would be omitted, as would the last two sentences, which explicitly use conditional proof and mathematical induction to draw the conclusions.

Mathematical induction enables us to prove that each item in a sequence has a given property F. In the previous example, F was the property of being a formula.

In the next example, F is the property of being logically equivalent to 'P', and the sequence is:

$$P, \quad (P \,\&\, P), \quad ((P \,\&\, P) \,\&\, P), \quad (((P \,\&\, P) \,\&\, P) \,\&\, P), \ldots$$

We shall call this series T. What we want to show, in other words, is that each member of T has the property of being logically equivalent to 'P'. In this sequence, for each number n ($n > 0$), the $(n + 1)$st item is a conjunction whose first conjunct is the nth item and whose second conjunct is 'P'.

We begin by recalling that two formulas are logically equivalent iff they have the same truth value on every valuation of both. Since no formula of T contains any sentence letter other than 'P', there are only two valuations to consider: the valuation on which 'P' is true and the valuation on which 'P' is false. The proof proceeds as follows:

METATHEOREM: Each item of series T is logically equivalent to 'P'.

PROOF:

Basis Case: The first item of T is 'P', which (trivially) has the same truth value as 'P' on any valuation. Hence the first item of T is logically equivalent to 'P'.

Inductive Step: Suppose that the nth item Φ of T is logically equivalent to 'P'. That is, Φ is true on any valuation on which 'P' is true and false on any valuation on which 'P' is false. Now the $(n + 1)$st item is of the form $(\Phi \,\&\, P)$. On any valuation on which 'P' is true, therefore, both conjuncts of $(\Phi \,\&\, P)$ are true; similarly, on any valuation on which 'P' is false, both conjuncts of $(\Phi \,\&\, P)$ are false. Thus, by the valuation rule for conjunction, $(\Phi \,\&\, P)$ is true on any valuation on which 'P' is true and false on any valuation on which $(\Phi \,\&\, P)$ is false. Thus 'P' has the same truth value as $(\Phi \,\&\, P)$ on every valuation of both, and so $(\Phi \,\&\, P)$, which is the $(n + 1)$st item in the series, is logically equivalent to 'P'.

Thus (by conditional proof) it follows that if the nth item of T is logically equivalent to 'P', then so is the $(n + 1)$st. Hence (by mathematical induction) each item of T is logically equivalent to 'P'. QED

Let's consider one more metatheorem that uses mathematical induction. In this case, we will be concerned with the following sequence of formulas, which we shall call T:

$$P_1, \quad (P_1 \vee P_2), \quad ((P_1 \vee P_2) \vee P_3), \quad (((P_1 \vee P_2) \vee P_3) \vee P_4), \ldots$$

For each number n greater than 0, the $(n + 1)$st item of this sequence is obtained from the nth by disjoining it with the letter 'P' subscripted with the numeral for $n + 1$. Our problem is to prove that for each such n, the tree constructed using the

*n*th item as its initial list contains exactly *n* paths. That is, F is the rather complex property of *being a disjunction whose tree contains the number of paths designated by the numeral that subscripts its second disjunct*. Despite the complexity of this property, mathematical induction operates in precisely the same way as in the previous metatheorem. Here is the proof:

METATHEOREM: For all *n*, the tree constructed by using the *n*th item of *T* as its initial list has exactly *n* paths.

PROOF:

Basis Case: The first item of *T* is 'P_1'. Since 'P_1' is atomic, the tree constructed by using it as the initial list is finished as soon as 'P_1' is written, and it contains one path.

Inductive Step: Suppose (inductive hypothesis) that the tree constructed by using the *n*th item of *T* as its initial list has exactly *n* paths. Now the $(n + 1)$st item is obtained from the *n*th by disjoining it with 'P' subscripted by the numeral for $n + 1$. Thus, when the $(n + 1)$st item is used as the initial list of a tree, the only possible first move is to check it and branch to the *n*th item on the left and to 'P' subscripted by the numeral for $n + 1$ on the right. The right path is then finished, since the initial formula is checked and 'P' with its subscript is atomic. And the left path below the initial formula will consist simply of the tree for the *n*th item of *T*, which by hypothesis has exactly *n* paths. Hence the whole tree must contain exactly $n + 1$ paths.

Thus we have shown (by conditional proof) that if the tree constructed by using the *n*th item of *T* as its initial list has exactly *n* paths, then the tree constructed by using the $(n + 1)$st item of *T* as its initial list has exactly $n + 1$ paths. So (by mathematical induction) for all *n*, the tree constructed by using the *n*th item of *T* as its initial list has exactly *n* paths. QED

To summarize: We have considered three important strategies for metalinguistic proofs: conditional proof, reductio ad absurdum, and mathematical induction. These strategies may be nested within one another in various combinations, but each always produces an argument of the same form. Which form to use is determined by the structure of the conclusion: For a conditional conclusion, use conditional proof; for a negative conclusion and some conclusions of other forms, use reductio; for a conclusion about a series of things, use mathematical induction. The essentials of these forms are expressed in the following templates. These templates may be used quite literally and mechanically in setting up proofs, but filling in the arguments (represented in each case by a box containing a sketchy outline of the argument) may require creativity.

Template for Conditional Proof

METATHEOREM: If [ANTECEDENT], then [CONSEQUENT]

PROOF: Suppose for conditional proof that [ANTECEDENT]

Unpacking
Logical Manipulation
Repacking

Therefore [CONSEQUENT]

Hence (by conditional proof) if [ANTECEDENT], then [CONSEQUENT].

Template for Reductio

METATHEOREM: [CONCLUSION]

PROOF: Suppose for reductio that [DENIAL OF CONCLUSION]

Unpacking
Logical Manipulation

Therefore [CONTRADICTION]

Hence (by reductio) [CONCLUSION].

Template for Mathematical Induction

METATHEOREM: All members of [SERIES] have property F

PROOF:

Basis Case:

(Style of argument here varies but is often trivial.)

Therefore the first member of [SERIES] has property F.

Inductive Step:

Suppose that the nth member of [SERIES] has property F

Unpacking
Logical Manipulation
Repacking

Therefore the $(n+1)$st member of [SERIES] has property F.

Hence (by conditional proof) we have shown that for any n, if the nth member of [SERIES] has property F, so does the $(n+1)$st. Consequently (using mathematical induction to combine this conclusion with the conclusion of the basis case) all members of [SERIES] have property F.

Exercise 5.5

Prove the following metatheorems:

1. Every member of the following sequence is a formula:

   $(P \rightarrow P)$, $(P \rightarrow (P \rightarrow P))$, $(P \rightarrow (P \rightarrow (P \rightarrow P)))$, . . .

2. Every member of the sequence of problem 1 is valid.
3. Every member of the following sequence is contingent:

   P, $\neg$P, $\neg\neg$P, $\neg\neg\neg$P, . . .

4. Each member of the following sequence of formulas

   P_1, $(P_2 \rightarrow P_1)$, $(P_3 \rightarrow (P_2 \rightarrow P_1))$, $(P_4 \rightarrow (P_3 \rightarrow (P_2 \rightarrow P_1)))$, . . .

   is true in any valuation on which 'P_1' is true. (Hint: For each n, the $(n + 1)$st member of the series is a conditional whose antecedent is 'P' subscripted by the numeral for $n + 1$, and whose consequent is the nth member.)
5. If Ψ is a valid formula, then every member of the following sequence is a valid formula:

   Ψ, $(\Psi \vee P_1)$, $((\Psi \vee P_1) \vee P_2)$, $(((\Psi \vee P_1) \vee P_2) \vee P_3)$, . . .

   (Hint: Use mathematical induction inside a conditional proof.)

5.6 ALGORITHMS

An **algorithm**[3] is a fully determinate computational procedure. A **fully determinate** procedure is one that leaves nothing to chance or human discretion; any two people (or computers) carrying out the procedure (on the same symbols) would carry out the same steps in the same order. Algorithms are **computational** in the sense that they are operations on symbols; that is, an algorithm takes a sequence of symbols and converts it into a sequence of symbols. One simple algorithm is the familiar procedure for adding a column of numbers. The process begins, for instance, with a sequence of symbols that looks like this:

```
  27
  82
+ 13
```

[3] The word "algorithm" (occasionally spelled "algorism") is a corruption of the name of the ninth-century Arabic mathematician Al-Khawarazmi. Al-Khawarazmi is most noted for bringing what we now call "Arabic" numerals from India to the Arab world, whence they were later transmitted to the West. He also wrote a famous textbook illustrating many algorithms (or "Al-Khawarazmisms"!).

This initial symbol sequence is called the **input** to the algorithm. Then you perform a series of precise, well-defined operations that yield a new sequence of symbols, namely,

122

This is the **output,** or answer. The steps comprising the algorithm consist of adding the individual digits (starting in the rightmost column), carrying the appropriate numbers, and so on. This procedure, of course, works with any finite column of numbers as input so that once you learn the algorithm, you can, at least in principle, add any column of numbers.

The symbols or characters that an algorithm operates on need not be numerals. The automatic "search and replace" operations available on most word processors, for example, are simple algorithms that operate on the character set of a computer (which includes the English alphabet), rather than just on numerals. Say you want to replace all occurrences of the word 'Milton' in a document with the word 'Shakespeare'. You put the cursor at the beginning of the document and invoke the algorithm. The computer then runs through the entire document from beginning to end, making the replacements you indicated. In this case, the input to the algorithm consists of three symbol sequences: 'Milton', 'Shakespeare', and the initial document. The output is the revised document, in which the word 'Shakespeare' has replaced the word 'Milton'. Here again the algorithm is a *general* procedure; it operates not only on these three symbol sequences, but (in principle at least—ignoring the memory limitations of computers) on any three sequences of letters.[4] This is why mathematicians sometimes refer to algorithms as **general procedures.**

The concept of an algorithm carries with it some important presuppositions, which are not always explicitly recognized. First, *each algorithm is defined only over a prescribed character set, that is, a specified alphabet of symbols.* Though both sequences of numerals and strings of ordinary text are symbol sequences, you can do addition only on sequences of numerals, not on strings of text. That is, the generality of an algorithm is not absolute; it is limited by the kinds of symbols the algorithm is designed to deal with. More specifically, each algorithm presupposes a fixed character set upon which it works.

A **character set** is simply a finite set of discrete symbols. It may be as simple as the binary alphabet of a computer (which has only two fundamental symbols, often represented as 0 and 1)[5] or as complex as the typographical system of English

[4] When the input to an algorithm consists of more than one sequence of symbols, as it does here, these may be regarded as a single sequence in which the three elements are listed in some conventional order (for example: term to be replaced, term to replace it, document). We might in practice need additional symbols (such as spaces, commas, or other special symbols) between successive members of the sequence so that we can tell where one ends and the next begins. But by this means, any finite set of sequences can be treated as a single sequence. Therefore we lose no generality by thinking of an algorithm as operating always on a *single* sequence of symbols.

[5] Simpler still are character sets containing only one character; the abaci discussed in Section 10.1 use in effect only a single character type: the counters that are manipulated in their registers.

(which includes letters, both upper and lower case, numerals, punctuation marks, etc.). Logicians, of course, are most interested in the character sets of logical languages. The character set for propositional logic, for example, consists of the twenty-six capital letters of the English alphabet, the ten numerals 0–9 (for subscripts), right and left parentheses, and the five characters for the logical operators. Among the most prominent algorithms applied to sequences of these symbols (formulas or lists of formulas) are truth tables and trees.[6]

The character set presupposed by any particular algorithm must be finite; that is, it must not contain an infinite number of fundamental symbols. We shall sometimes talk about infinite *sequences* of characters, but the character set itself must be finite. (Thus these infinite sequences always contain repeated characters.) This is a genuine limitation, though it might not seem so at first, for there are symbol systems that can be interpreted as having infinitely many characters. Consider, for example, the dial of a nondigital watch. If the hands move continuously—instead of in discrete jumps or ticks—then each configuration of the hands might be thought of as a character representing a time. But between any two distinct positions of a given hand, there is always an intermediate position so that there are infinitely many of these "characters." Such characters (or sequences of them) would not, therefore, be appropriate input for an algorithm. We'd have to digitize them—that is, represent them in a symbol system with a finite character set, before we could apply anything that could legitimately be called an algorithm.

The character set must not only be finite; *its symbols must be distinct* as well. This does not preclude some variation. For example, all the following constitute tokens of the letter 'U' in the character set of English:

U ***U*** $\mathcal{U}$ **u** $_u$ U u U u

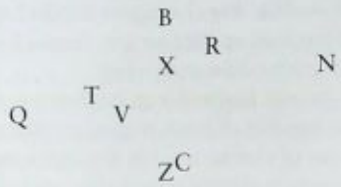
But what about this?

Is it a 'U', or an 'O' that didn't quite get closed at the top? In reality, there are borderline cases—symbol tokens that could be classified either of two or more ways. But the conception of an algorithm presupposes that such things don't happen, that each individual symbol is distinct and uniquely classifiable.

In addition, inputs to and outputs from algorithms must be ***sequences*** *of symbols.* That is, they must be arrayed in a distinct linear order, like the text you are now reading. By convention the sequential order of this text is from left to

[6] Actually, truth tables and trees are not quite algorithms as we use them, since we allow some choice as to which rule to apply next in a tree or which subformula to analyze next in a truth table. Only if we adopted rules that rigidly determined the order of these operations would trees and truth tables be algorithms, strictly speaking. We could adopt such rules, just to make truth tables and trees conform to the definition of an algorithm, but that would be more trouble than it would be worth.

right and from the top to the bottom of the page, but other conventions could be used, as they are in some languages. Below are some symbols from the English character set that are not arranged in any clear sequential order:



This sort of thing could not be input or output for an algorithm, unless we established some convention that would impose a sequential order on it.

Moreover, all input sequences are presumed to be finite. Infinite sequences, like the sequence of numerals used for counting,

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, . . .

cannot be input to an algorithm (though each individual member of the sequence, being itself a *finite* string of symbols, could be).

Finally, *an algorithm need not be defined over all the finite sequences of symbols from its prescribed character set.* Many algorithms work with only certain very specific sorts of sequences. In the case of truth tables and trees, for example, the sequences must be formulas or lists of formulas, that is, sequences generated by the formation rules. There are infinitely many ill-formed or nonsensical sequences of characters of propositional logic, such as

)) → P (

for which the truth-table algorithm is not defined. In general, we designate those symbol sequences for which an algorithm *is* defined as its **permissible** symbol sequences.

Noting the limitations of algorithms gives us a clearer conception of their nature. An algorithm can apply only to finite, linear sequences of absolutely distinct symbols—and only to those that count as permissible sequences of the prescribed character set.[7]

A final and crucial feature of algorithms is that they may be either terminating or nonterminating. A **terminating** algorithm is one that, given any permissible input, will always yield its output after a finite number of steps. A **nonterminating** algorithm is one that for at least one permissible input does not yield its entire output after any finite number of steps. The procedure for adding columns of numbers, for example, is a terminating algorithm. But the algorithm for counting

[7] Philosophers—especially those who want to identify thought with algorithmic process—have not always kept these presuppositions in mind. The human mind operates with sensory input and behavioral output that, at least on the face of it, seems not to satisfy all of these presuppositions.

(using a given numeral, usually '1', as the starting point or input) is nonterminating. It's a fully definite computational procedure, but it never achieves completion.

Logicians are interested in algorithms because they want to know how much of logic can be reduced to mechanical computational procedures. Early in this century, some philosophers and mathematicians hoped that all the theses of logic and mathematics could eventually be brought within the reach of terminating algorithms—that the truth or falsity of any statement of logic or mathematics could be decided by finite calculations. They thought this could be accomplished by completely *formalizing* logic and mathematics—that is, expressing them in symbol systems—and then devising the appropriate algorithms to operate on these symbol systems. Accordingly, this line of research was called **formalism.** Formalists hoped to encode various fields of logic and mathematics in **axioms** (fundamental assumptions) expressed in a logical language, and then to apply finite computational procedures (usually envisioned at the time as rules of inference) to these axioms to determine the truth or falsity of any question expressible in the system.

But formalism failed. We now know that it is impossible to answer all logical questions by finite computations, and we shall prove this when we consider the undecidability of predicate logic. Nevertheless, for some restricted systems of logic, the formalist dream can be realized. Propositional logic is one such system. For all questions of validity (for both formulas and sequents), invalidity, consistency, and so on, we have terminating algorithms (truth tables and trees) which give the answers.

Here we are concerned with trees. It is obvious that the tree test is—or can easily be transformed into—an algorithm (see footnote 6). What is not so obvious is that it terminates for all inputs consisting of any finite list of sentences whatsoever. That requires proof. Our task in the next section is to construct a metatheorem that proves this.

5.7 DECIDABILITY

What we want to prove ultimately is that propositional logic is decidable. What it means to say that a logic is **decidable** is that there exists a terminating algorithm which determines for each sequent of the logic whether or not it is valid. Such an algorithm is called a **decision procedure** or a **solution to the decision problem** for the logic. What we shall show is that the tree test is a decision procedure for propositional logic.[8] To do so, we need to establish that

(1) the tree test for propositional logic is in fact a *terminating* algorithm,
(2) if the tree test classifies the sequent as valid (i.e., all paths of its finished tree close), then that sequent *is* valid, and

[8] We could have shown this for the truth-table test as well. In fact, for the truth-table test it's obvious. We focus on trees, however, because the tree test can be straightforwardly generalized to predicate logic; the truth-table test can't be.

(3) if a sequent *is* valid, the tree test classifies that sequent as valid (i.e., all paths of its finished tree close).

We will prove proposition (1) in this section, proposition (2) in Section 5.8, and proposition (3) in Section 5.9. Proposition (2) expresses the **soundness** of the tree test; the test is sound in the sense that if it classifies a sequent as valid, the sequent is in fact valid. Proposition (3) expresses the test's **completeness;** the test is complete in that it does not fail to classify any valid sequents as valid. Uniting the conditionals (2) and (3) into a single biconditional, we get a statement that expresses the full accuracy of the tree test:

(4) The tree test classifies a sequent as valid (i.e., all paths close) if and only if that sequent *is* valid.

Our first task is to prove that the tree test always terminates. The reason it terminates is that whenever we apply a rule to a formula, each of the new formulas produced by the rule is shorter (i.e., contains fewer characters) than the original formula. Now formulas are not infinitely divisible; like material substances they have smallest units, or atoms—namely, atomic formulas. Hence this shortening process cannot go on forever. Eventually it has to stop.

This is the right idea, but it misses something: Although formulas grow shorter and shorter within each path, the *number of paths* increases each time we apply a branching rule. In applying branching rules, might we not spawn so many new paths that we create more work than we complete and hence never finish? In other words, even though no single path may ever grow infinitely long, might we not generate so many new paths that the tree continues to grow—perhaps by becoming "bushier and bushier"—without end?

In fact this cannot happen. But that is not so obvious; it requires proof.

Our reasoning will fall into several parts, which we will express as **lemmas** (short proofs preliminary to a major result). In lemma 1 we will show that because of the shortening of formulas, each individual path in a tree must come to an end. Then, to alleviate the concern that the tree might continue to grow forever anyway (e.g., by multiplication of paths), we will prove in lemma 3 that in order to grow endlessly it would have to have an unending path, which lemma 1 will have shown to be impossible. Before proving lemma 3, we shall prove another lemma, lemma 2, which provides a fact needed in proving lemma 3. Finally, we'll combine lemmas 1 and 3 into a metatheorem that proves the tree test always terminates. Before embarking on our proofs, we need some definitions.

DEFINITION 1 The **character count** of an open path is the total number of characters (logical operators, sentence letters, and parentheses—numerical subscripts don't count) contained in unchecked formulas on that path. The character count of a closed path is zero.

NOTE: For purposes of calculating the character count, the formation rules must be followed strictly. This means that outer brackets may not be dropped; they are included in the count.

DEFINITION 2 A path P_2 is a **one-step extension** of a path P_1 iff P_2 is obtained from P_1 by applying a single tree rule to some unchecked formula or (in the case of the negation rule) pair of unchecked formulas of P_1.

Each application of a branching rule produces two one-step extensions of a path, but application of a nonbranching rule produces only one one-step extension. One-step extensions created by some rules (for example, the disjunction or double negation rules) contain only one more formula than the path they extend. But one-step extensions created by others (for example, the conjunction or biconditional rules) contain two more formulas than the path they extend.

We can now launch our proofs. We first prove lemma 1, which shows in effect that if an initial list has a finite character count (which is always the case), then any path it generates must be finitely long as well.

Lemma 1: If the character count of the tree's initial list is n, then each path of the tree must be finished after at most n applications of the tree rules to formulas on that path.

PROOF: Suppose the character count of the tree's initial list is n. Now when any of the rules is applied to a formula on a path P, each of the resulting one-step extensions of P has a character count at least one less than the character count of P (check this for each of the rules).[9] Further, the minimum character count for any path is zero.[10] Thus, since the character count of the initial list is n, and each application of a rule decreases the character count of the resulting one-step extensions by at least one, at most n applications of the rules can be made to formulas on a path before that path is finished.

[9] The biconditional rule, for example, allows us to check a formula of the form $(\Phi \leftrightarrow \Psi)$ and create one one-step extension to which we add Φ and Ψ and another to which we add $-\Phi$ and $-\Psi$. The formula $(\Phi \leftrightarrow \Psi)$ has three characters in addition to those in Φ and Ψ, namely, '(', '$\leftrightarrow$', and ')'. (Outer brackets are included in the character count!) But the first one-step extension omits all three of these characters, keeping only Φ and Ψ, whereas the second adds only two characters in addition to those in Φ and Ψ, namely, two occurrences of '–'. Thus, since a checked formula no longer counts, application of the biconditional rule reduces the character count along the first one-step extension by three and along the second one-step extension by one. Similar reductions of the character count occur with all the other rules.

[10] The character count of a path drops to zero if the path closes; if it doesn't close, the path must nevertheless be finished by the time its character count reaches zero, since by that time all formulas are checked and so no further rules can be applied. Actually, the character count of an open path cannot drop as far as zero, since some unchecked atomic formulas or negations of atomic formulas remain on the path.

Hence, if the character count of the tree's initial list is n, then each path of the tree must be finished after at most n applications of the tree rules to formulas on that path. QED

Having shown that all paths must be finite, we must still prove that the tree can't grow forever by endless proliferation of these finite paths. To make these ideas precise, we add two more definitions.

DEFINITION 3 A path P is **infinitely prolongable** iff there exists an infinite series $P_0, P_1, \ldots$ of paths such that $P_0 = P$ and, for each n, P_{n+1} is a one-step extension of P_n.

That is, a path is infinitely prolongable iff the tree rules can be applied to make it grow endlessly longer. This is just what lemma 1 rules out; that is, so long as the initial list has a finite character count, lemma 1 tells us that it cannot produce an infinitely prolongable path. But we are still concerned about a tree growing endlessly in some other way—for example, by becoming infinitely "bushy." The next definition gives precision to this worry. It captures the idea of infinite growth in general.

DEFINITION 4 A path P is **nonterminating** iff there exists an infinite series $T_0, T_1, \ldots$ such that $T_0 = P$ and, for each n, T_{n+1} is the result of applying a single rule to an unchecked formula or (in the case of the negation rule) pair of unchecked formulas somewhere in T_n.

The infinite series $T_0, T_1, \ldots$ is a series of trees (or partial trees) generated by applying rules starting with P, but not confining application of the rules to only one path. Thus a path is nonterminating iff starting with that path we can apply tree rules to formulas (perhaps among various branches into which that path splits) forever. Nontermination is an apparently broader concept than infinite prolongability; a path might, it seems, be nonterminating by being able to grow endlessly more "bushy," as well as by growing endlessly longer. Actually, however, this apparent difference is illusory, as we'll prove in lemma 3. But to prove lemma 3, we first need to prove lemma 2:

Lemma 2: If P is a nonterminating path, then P has a nonterminating one-step extension.

PROOF: Suppose (for conditional proof) that P is a nonterminating path. That is, there is an infinite series $T_0, T_1, \ldots$ such that $T_0 = P$ and, for each n, T_{n+1} is the result of applying a single

rule somewhere in T_n. Thus in particular T_1 is the result of applying a single rule to a formula or pair of formulas of P. Since no single application of a rule can split a path into more than two paths, T_1 contains at most two paths—maybe only one.

Now suppose for reductio that P does not have a nonterminating one-step extension. This means that no path of T_1 is nonterminating. Hence there can't be an infinite succession of rule-applications starting with any path of T_1. But since T_1 has at most two paths, it follows that there can't be an infinite succession of rule-applications to T_1 itself, since the total number of rule-applications for T_1 is just the total number for its paths, and this number, being the sum of at most two finite quantities, is finite. Hence there is no infinite series $T_0, T_1, \ldots$ such that $T_0 = P$ and, for each n, T_{n+1} is the result of applying a single rule somewhere in T_n, in contradiction to what we concluded earlier.

Hence, contrary to our supposition, P does have a nonterminating one-step extension.

Thus we have shown that if P is a nonterminating path, then P has a nonterminating one-step extension.

The next lemma, which shows that a path can't grow infinitely in any sense without being infinitely prolonged, is historically known as König's lemma:

Lemma 3 (König's Lemma): If L is a nonterminating path, then L is infinitely prolongable.

PROOF: Suppose (for conditional proof) that L is a nonterminating path. This means that there exists an infinite series $T_0, T_1, \ldots$ of trees such that $T_0 = L$ and, for each n, T_{n+1} is the result of applying a single rule somewhere in T_n. We now define an infinite series $P_0, P_1, \ldots$ of paths such that $P_0 = L$ and, for each n, P_{n+1} is a one-step extension of P_n. First let $P_0 = L$. Now by lemma 2, P_0 has at least one nonterminating one-step extension. Call it P_1. (If there is more than one, let P_1 be the leftmost.) Now again by lemma 2, since P_1 is nonterminating, it must have a nonterminating one-step extension P_2, and so on ad infinitum. Clearly, then, $P_0, P_1, \ldots$ is an infinite series of paths such that $P_0 = L$ and, for each n, P_{n+1} is a one-step extension of P_n. But this means that L is infinitely prolongable.

Thus we have shown that if L is a nonterminating path, then L is infinitely prolongable.

We are ready at last to combine lemmas 1 and 3 into the major result of this section:

METATHEOREM (Decidability): Any tree for any finite list of formulas of propositional logic is finished after some finite number of applications of the tree rules.

PROOF: Suppose for reductio that this is not the case—that there is a finite list L of formulas of propositional logic which yields a tree that is not finished after any finite number of applications of the rules. This means that, starting with L, rules can be applied infinitely so that there is an infinite series $T_0, T_1, \ldots$ such that $T_0 = L$ and, for each n, T_{n+1} is the result of applying a single rule somewhere in T_n. L, that is to say, is nonterminating. Hence by lemma 3, L is infinitely prolongable. But since L is a finite list of formulas, it must have a finite character count, n. Hence by lemma 1 each path of L's tree must be finished after at most n applications of tree rules to formulas on that path, where n is finite. So L is not infinitely prolongable, and we have a contradiction.

Therefore the tree for any finite list of formulas of propositional logic is finished after some finite number of applications of the tree rules. QED

Since sequents are always finite lists, and they remain finite when we negate their conclusions, it follows that the tree test peformed on a sequent will always finish in a finite number of steps. The tree test is, in other words, a terminating algorithm.

5.8 SOUNDNESS OF THE TREE TEST

A test for validity is said to be **sound** if whenever that test classifies a sequent as valid, it is in fact valid. In this section, we will show that the tree test is sound.[11] We shall do this proof in two stages. First we will prove as a metatheorem that any tree constructed from a consistent initial list has an open path. Then we will derive

[11] This is a different use of the term than when we speak of a *sound* argument—that is, a valid argument with true premises. The soundness of the tree rules (or of the rules of inference; see Section 5.10) implies nothing at all about the truth or falsity of the premises. It is unfortunate that the same word 'sound' is used in both of these ways, but the usage is so firmly established in the logical literature that there is no point in bucking it.

the soundness result explicitly as a **corollary**—that is, a result that follows easily from something previously established.

To prove the conditional that if an initial list is consistent, then it always yields an open path, we suppose for conditional proof that we have a consistent initial list. We then unpack the notion of consistency as truth on some valuation, which we shall call $\mathcal{V}$. The heart of the argument is to show that each time we apply a tree rule, the resulting tree contains a path P whose formulas are all true on $\mathcal{V}$. But since all of P's formulas are true on $\mathcal{V}$, P cannot be closed because it cannot contain both a formula and its negation (since these could not both be true on $\mathcal{V}$). Hence P must be an open path.

That's the proof in a nutshell. Here it is in greater detail:

METATHEOREM: If an initial list of formulas is consistent, then there is an open path through any (finished or unfinished) tree obtainable from that list by the tree rules.

PROOF: Suppose (for conditional proof) that some initial list of formulas, call it L, is consistent.

This means that there is some valuation $\mathcal{V}$ on which all the members of L are true. Now let T be any tree obtainable from L by the tree rules. To create T, a series $T_1, \ldots, T_z$ of trees was successively constructed, whose first member T_1 was L, whose final member T_z is T, and whose $(n + 1)$st member T_{n+1}, for each n $(1 < n \leq z)$, was obtained from the nth, T_n, by the application of a single tree rule. We shall prove that each member of this series contains an open path, whence it follows that T itself (i.e., T_z) contains an open path. To prove this, it suffices to show that every tree in the sequence contains a path whose formulas are all true on $\mathcal{V}$. For if all formulas of a path are true on $\mathcal{V}$, then (by valuation rule 1) that path cannot contain both a formula and its negation, and hence must be open. To prove that each tree in the series contains a path all formulas of which are true on $\mathcal{V}$, we use mathematical induction:

Basis Case: The first member of the series is T_1, which is L itself, and by hypothesis each member of L is true on $\mathcal{V}$.[12]

Inductive Step: Suppose (inductive hypothesis) that the nth item T_n of the series (where $n < z$) contains a path P all of whose formulas are true on $\mathcal{V}$. Now the $(n + 1)$st item, T_{n+1}, is formed by a single application of a rule

[12] When I say that this is true "by hypothesis," I refer to the fact that we are operating under the supposition (for conditional proof) that L is consistent, which means there must be a valuation on which all of its members are true. We have labeled that valuation "$\mathcal{V}$". Thus each member of L is true on $\mathcal{V}$.

to T_n. There are two possibilities concerning the point of application of this rule: Either the formula or formulas to which the rule is applied are on P or they are not. If the rule is applied to formulas on P, then by the inductive hypothesis these formulas are true on $\mathcal{V}$. Hence the rule used can't have been the negation rule, which closes paths, since a formula and its negation cannot both be true on $\mathcal{V}$. So it must have been one of the other nine rules. Now, when applied to a formula on a path whose formulas are all true on some valuation, each of these rules produces at least one one-step extension of that path whose formulas are all true on that valuation. (It is easy to check this for each rule, and you should do so.[13]) So at least one of the one-step extensions of P is a path of T_{n+1} whose formulas are all true on $\mathcal{V}$. If, on the other hand, the formula or formulas to which the rule is applied are not on P, then nothing will be added to P in moving from T_n to T_{n+1}. Hence in this case P itself is a path of T_{n+1} whose formulas are all true on $\mathcal{V}$. So either way T_{n+1} contains a path whose formulas are all true on $\mathcal{V}$.

Thus (by conditional proof) we have shown that for any n ($n < z$), if T_n contains a path whose formulas are all true on $\mathcal{V}$, so does T_{n+1}. So (by mathematical induction) each tree in the sequence $T_1, \ldots, T_z$ contains a path whose formulas are all true on $\mathcal{V}$. Hence T (T_z) must itself contain a path whose formulas are all true on $\mathcal{V}$. Hence (as explained above) T contains an open path.

Hence we have shown (by conditional proof) that if L is consistent, then there is an open path through any (finished or unfinished) tree T obtainable from L by the tree rules. QED.

We now use this metatheorem to prove that the tree test is sound. A test for validity is sound, once again, if whenever that test classifies a sequent as valid, that sequent is in fact valid. What it means for the tree test to **classify a sequent as valid** is that, given the premises and the negation of the conclusion as an initial list, we

[13] Take, for example, the disjunction rule $\vee$. Suppose this is applied to a formula $\Phi \vee \Psi$ which occurs on a path all of whose formulas are true on some valuation $\mathcal{V}$. Then, since $\Phi \vee \Psi$ itself is true on $\mathcal{V}$, by the valuation rule for disjunction either Φ or Ψ must be true on $\mathcal{V}$. Therefore, since the rule $\vee$ produces two one-step extensions of the path, one of which appends Φ to it and the other of which appends Ψ, all the formulas of at least one of these one-step extensions of the path must be true on $\mathcal{V}$.

get a tree all of whose paths close. The argument from our metatheorem to soundness is relatively simple:

COROLLARY (Soundness): If the tree test classifies a sequent as valid, it is in fact valid.

PROOF: Suppose (for conditional proof) that the tree test classifies a sequent $\Phi_1, \ldots, \Phi_n \vdash \Psi$ as valid. This means that the tree's initial list is $\Phi_1, \ldots, \Phi_n, \sim\Psi$ and that all paths of the tree close. But by the previous metatheorem, if that initial list is consistent, then there is an open path through any tree constructed from it. Since all paths close, there is no open path through the finished tree. So (by modus tollens), the initial list is inconsistent. But then by the metatheorem proved at the end of Section 5.2, the sequent is valid.

Therefore, if the tree test classifies a sequent as valid, it is in fact valid. QED

A radical skeptic might wonder what this soundness proof really proves. If, for example, someone had doubts about the validity of sequents proved by $\rightarrow$I, those doubts would hardly be allayed by a metatheorem established by conditional proof—that is, the same pattern of reasoning at a different linguistic level. As a response to such a person, the metaproof would be in some sense circular—assuming the validity of one of the very patterns whose validity it purported to prove.

Metatheoretic proofs are not, however, intended as responses to radical skeptics. To a person who doggedly doubts the elementary rules of logic, there is no effective *logical* response. Rather, the point of this soundness proof is to show, given a prior understanding of logic, that any sequent judged valid by the tree test is valid on classical semantics—the semantics described in Chapter 3. The soundness proof provides not a wholesale assurance that classical reasoning is irrefutable, but the more modest assurance that the tree rules validate only sequents they ought to validate, given classical semantics. It presupposes, moreover, a willingness to use what are in effect the rules of classical logic in the metalanguage. But what alternative is there? If we did not grant the validity of some sort of reasoning somewhere, we would never accept any conclusion and never come to systematic insight about anything. That is the radical skeptic's game, but it is a game that precludes much intellectual adventure.

Exercise 5.8.1

In the proof of the main metatheorem of this section, it is necessary to verify for each tree rule other than ~ that when applied to a path whose formulas are all true

on some valuation $\mathcal{V}$, it yields at least one one-step extension of that path whose formulas are all true on $\mathcal{V}$ or on some expansion of $\mathcal{V}$. I did this for the $\vee$ rule in footnote 13. Write out the necessary verifications for the rules $--$, $\&$, $-\&$, $-\vee$, $\rightarrow$, $-\rightarrow$, $\leftrightarrow$, and $-\leftrightarrow$.

Exercise 5.8.2

Using the metatheorem of this section as an assumption, prove the following corollaries:

1. If the tree test classifies a formula as inconsistent, it is in fact inconsistent.
2. If the tree test classifies a formula as valid, it is in fact valid.
3. If a formula is contingent, the tree test classifies it as contingent.

5.9 COMPLETENESS OF THE TREE TEST

In this section we prove the completeness of the tree test. A test for validity of sequents is **complete** iff it classifies all the valid sequents as valid; in other words, if a sequent *is* valid, the test classifies that sequent as valid. To prove that the tree test is complete, we proceed as we did in proving that it is sound. That is, we first prove a general metatheorem and then append a simple corollary that proves the completeness result.

The general metatheorem is expressed as the following conditional:

> If there is an open path through a finished tree, then its initial list is consistent.

To prove this conditional, the first step is to suppose for conditional proof that there is an open path P through a finished tree. We then show how to construct a valuation $\mathcal{V}$ on which all the members of P are true. But given that all members of P—including the initial list itself—are true on $\mathcal{V}$, by the definition of consistency the initial list is consistent.

In showing how to construct $\mathcal{V}$, we first define the notion of formula length. The **length** of a formula is the number of characters it contains, excluding subscripts. Thus, for example, the length of '$-(P \& Q)$' is 6 and the length of '$--P_{12}$' is 3. Here is the proof in full regalia:

> METATHEOREM: If there is an open path through a finished tree, then its initial list is consistent.
>
> PROOF: Suppose (for conditional proof) that there is an open path P through a finished tree. Since this tree is finished, any formula

of P having length 3 or more has been checked.[14] Now consider the valuation $\mathcal{V}$ which makes formulas of length 1 (sentence letters) occurring on P true and all other sentence letters false.[15] Since P is open and the tree is finished, P cannot contain both an atomic formula and its negation. So all formulas of P having length 1 or 2 are true on $\mathcal{V}$.[16] We use this fact to show that all formulas of P are true on $\mathcal{V}$ and hence that all members of the initial list are true on $\mathcal{V}$ (since these are formulas of any path, including P). To do so, we proceed by reductio:

> Suppose for reductio that some formula of P is not true on $\mathcal{V}$. Then there must be some formula Φ of P which is not true on $\mathcal{V}$, such that all formulas of P shorter than Φ are true on $\mathcal{V}$. Now the length of Φ is at least 3, for otherwise Φ would be true on $\mathcal{V}$, as noted above. And since the tree is finished and P is open, some rule has been applied to Φ.[17] Except for the negation rule (which closes paths and hence could not have been applied on P), applying any rule to any formula yields only shorter formulas. Hence P contains at least one formula shorter than Φ that is obtained from Φ by one of the rules. But since Φ is the shortest formula on P that is not true on $\mathcal{V}$, all formulas shorter than Φ on P are true on $\mathcal{V}$. Hence all formulas on P obtained from Φ by the rules are true on $\mathcal{V}$. But it is a property of each of the rules that if they yield only true formulas on some path, then the formula to which they were applied is true. (Check this for each

[14] That is, only atomic formulas or negations of atomic formulas remain unchecked. This must be the case, since otherwise more rules could be applied on P, which is open, and so the tree would not be finished.

[15] Recall that formulas count as occurring on a path only if they are listed there as whole formulas, not if they are merely parts of other formulas.

The point of making all sentence letters that do not occur on P false is that some sentence letters in the initial list may not occur either negated or unnegated along the path. This signifies that, given the other truth values determined by the path, their truth value does not affect the truth of formulas of the initial list. For the sake of definiteness, we define $\mathcal{V}$ in a way that makes those sentence letters false, but this choice is arbitrary. Our definition also, of course, makes sentence letters whose negations occur on P false, and this is not arbitrary.

[16] Formulas of length 1 (sentence letters) are true because they are assigned the value T by $\mathcal{V}$ directly. And formulas of length 2 (negated sentence letters) are true because, since their sentence letters do occur on path P, these sentence letters have been assigned the value F by $\mathcal{V}$.

[17] We know this because a finished path by definition contains no unchecked sentences of length 3 or more. Any checked sentence is a sentence to which a rule has been applied.

rule.[18]) Hence Φ itself must be true on $\mathcal{V}$. But by hypothesis Φ is not true on $\mathcal{V}$, and so we have a contradiction. This contradiction shows that our reductio hypothesis (namely, that some formula of P is not true on $\mathcal{V}$) is false; hence all formulas of P are true on $\mathcal{V}$, whence it follows (as noted above) that the initial list was consistent.

But all of this was proved under the assumption (for conditional proof) that the finished tree contains an open path. Hence in sum what we have proved is that if a finished tree contains an open path, then the tree's initial list was consistent. QED

COROLLARY (Completeness): If a sequent is valid, the tree test classifies that sequent as valid.

PROOF: Suppose that $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is a valid sequent. It follows by the metatheorem proved at the end of Section 5.2 that the set $\{\Phi_1, \ldots, \Phi_n, \sim\Psi\}$ is inconsistent. But by the previous metatheorem, if there is an open path through the finished tree whose initial list is this set, then that initial list is consistent. Hence (by modus tollens) there is no open path through the tree constructed by using $\Phi_1, \ldots, \Phi_n, \sim\Psi$ as the initial list. But this is to say that the tree test classifies the sequent $\Phi_1, \ldots, \Phi_n \vdash \Psi$ as valid.

Therefore, if a sequent is valid, then the tree test classifies that sequent as valid. QED

Exercise 5.9

Using the metatheorem of this section as an assumption, prove the following corollaries:

1. If a formula is inconsistent, then the tree test classifies it as inconsistent.
2. If a formula is valid, then the tree test classifies it as valid.
3. If the tree test classifies a formula as contingent, then it is contingent.

[18] If, for example, the rule & is applied to a conjunction Φ & Ψ, it will produce the formulas Φ and Ψ on each path below it. But by the valuation rule for conjunction, if these two formulas are true, then Φ & Ψ itself must be true. Similarly, if $\vee$ is applied to a disjunction $\Phi \vee \Psi$, each new path it produces will contain either Φ or Ψ. But by the valuation rule for disjunction, if either of these two formulas is true, then $\Phi \vee \Psi$ is true. Thus, if either of these rules yields only true formulas along a given path, then the formula to which it is applied must be true. The same result holds for the other rules.

5.10 SOUNDNESS AND COMPLETENESS OF THE NATURAL DEDUCTION RULES

Having shown that the tree test is sound and complete—that it classifies a sequent as valid iff it is in fact valid—our next task is to show that the ten natural deduction rules presented in Sections 4.2 and 4.3 are also sound and complete. A system of *inference rules* is **sound** iff each sequent provable by these rules is valid, and it is **complete** iff each valid sequent is provable by the rules. The soundness of these ten basic inference rules provides a guarantee that no sequent provable by these rules has a counterexample. The completeness of these rules guarantees that they alone suffice to prove every valid sequent expressible in the language of propositional logic.

The ten rules by themselves, however, do not constitute a decision procedure, since applications of the rules need not terminate. Consider, for example, the following infinite "proof" that uses only the rules &I and &E:[19]

1.	P & Q	A
2.	P	1 &E
3.	Q	1 &E
4.	P & Q	2, 3 &I
5.	P	4 &E
6.	Q	4 &E
7.	P & Q	5, 6 &I
8.	P	7 &E
9.	Q	7 &E
	.	
	.	
	.	

This pattern can clearly be iterated ad infinitum. Nobody would do this in practice, of course, but it is a common experience in working with inference rules to reason in circles, repeatedly deriving what you have already proved or assumed. This simple example shows that in principle you could do so forever.

So inference rules by themselves are not a decision procedure in the way truth tables or trees are. It is, however, possible to design a terminating algorithm for generating proofs by rigorously specifying the order of application of the inference rules. One way to do this is to make proofs mimic trees. If a sequent is valid, we know from the completeness of the tree test that all the paths of the tree for that sequent close. The tree procedure is closely akin to a reductio (~I) proof in which the negation of the conclusion is shown to lead to contradiction. If we had a terminating algorithm for reliably converting trees for valid sequents into such proofs, then by using it we could avoid the kind of infinite regress illustrated

[19] I put the word 'proof' in quotation marks because an infinite structure like this is not a genuine proof. A proof always has a final line on which its conclusion is displayed.

above. The whole procedure would, however, be parasitic upon the tree test. That is, we would need to construct a tree first to determine whether or not the sequent was valid; then, if all paths closed, we would convert that tree into a proof. For invalid sequents, whose paths do not all close, prior performance of the tree test would prevent us from attempting a proof. Thus an algorithm for converting trees of valid sequents to proofs would have little value in itself, since once we have determined the validity of a sequent by the tree test, it is redundant to construct a proof.

Yet such an algorithm would have at least one valuable implication: It would demonstrate the completeness of the ten basic rules. For if we could convert any tree for a valid sequent into a proof of that sequent, then the following would be true:

(1) If the tree test classifies a sequent as valid, then that sequent can be proved using only the ten basic inference rules.

Putting this together with the completeness of the tree test (which we proved in the previous section):

(2) If a sequent is valid, the tree test classifies that sequent as valid,

we obtain the conclusion:

(3) If a sequent is valid, then that sequent can be proved using only the ten basic inference rules.

But (3) asserts that the rules are complete. Since we have already proved (2), to prove the completeness of the inference rules, then, we need only prove (1)—that is, to show how to convert a tree for a valid sequent (a tree all of whose paths close) into a proof that uses only the ten rules. Before defining a general method for doing this, let's do the conversion for some specific examples.

Our first example is the valid sequent '$P \& Q \vdash P \vee R$'. Our aim is to show how to convert the tree for this sequent into a proof. We shall do this by constructing the tree and the corresponding proof side by side. The first step of the tree test is to write the initial list, consisting of the premises and the negation of the conclusion. The tree test reveals the inconsistency of this initial list (assuming the sequent it represents is valid) by showing that each possible way in which all of its formulas might be true leads to contradiction. We may think of this as a reductio strategy in which the negation of the conclusion is hypothesized in order to show that, given the premises, it leads to absurdity. Thus we can begin to construct a proof that mimics the tree test by assuming the sequent's premises and hypothesizing the negation of its conclusion for indirect proof. Thus, for the sequent '$P \& Q \vdash P \vee R$', the tree and the corresponding proof begin as follows:

Tree		Corresponding Proof		
1. $P \& Q$	Premise	1. $P \& Q$		A
2. $\sim(P \vee R)$	Neg. Concl.	2. |	$\sim(P \vee R)$	H (for $\sim$I)

In the tree, both formulas require nonbranching rules so that the order of appli-

cation is unimportant. Let's begin by analyzing 'P & Q'. The corresponding move in the proof is to apply &E twice:[20]

3.	P	1 &	3.	P	1 &E
4.	Q	1 &	4.	Q	1 &E

The occurrence of 'P & Q' in the tree should now be checked off. The next step in constructing the tree is to check '~(P ∨ R)' and analyze it into '~P' and '~R' using the ~∨ rule. In the proof these same formulas may be deduced by converting '~(P ∨ R)' into '~P & ~R' by De Morgan's law (DM) and then using two steps of &E:

5.	~P	1 ~∨	5.	~P & ~R	2 DM
6.	~R	1 ~∨	6.	~P	5 &E
			7.	~R	5 &E

DM, of course, is a derived rule, not one of the ten basic rules. We saw in Section 4.4, however, that derived rules are merely abbreviatory devices; anything proved with derived rules can also be proved with the ten basic rules so that the use of derived rules here is legitimate.

Both 'P' and '~P' now appear on the tree's one path. The next step in the tree, therefore, is to close this path using the negation rule. We shall think of the 'X' that closes the path as representing a contradictory formula—specifically, 'P & ~P'. The proof also contains both 'P' and '~P'; thus the corresponding move in the proof is to derive 'P & ~P' by &I:

7.	X	3, 5	8.	P & ~P	3, 6 &I

The tree, which represents only the derivation of a contradiction from the premises and the hypothesis of the negated conclusion, is now complete. But to finish the proof we need to end the hypothetical derivation and deduce the conclusion. This takes two more steps, one of ~I and one of ~E:

9.	~~(P ∨ R)	1–8 ~I
10.	P ∨ R	9 ~E

We have now converted the tree for the sequent 'P & Q ⊢ P ∨ R' into a proof of that sequent.

This example was unusually simple, since the tree did not branch. When trees branch, the corresponding proofs involve uses of ∨E within the overall ~I strategy, and things become more complicated. Take, for example, the valid sequent 'P ∨ Q, ~P ⊢ Q', which expresses one version of disjunctive syllogism. As before, we begin the tree by listing the premises and negation of the conclusion, and we begin the corresponding proof by assuming the premises and hypothesizing the negation of the conclusion:

[20] Only the first application of &E, the one at line 3, is essential to the proof, but we are concerned here with proofs that mimic trees, not with proofs that are maximally compact. The extra steps are harmless.

Tree			Corresponding Proof			
1.	P ∨ Q	Premise	1.	P ∨ Q		A
2.	~P	Premise	2.	~P		A
3.	~Q	Neg. Concl.	3.	|	~Q	H (for ~I)

The next step of the tree is to check the disjunction and break it into its components, drawing out the consequences of each component separately along distinct paths. However, if the initial list is inconsistent (as this one is), then each path closes because each contains a contradiction. Thus, if a disjunction $\Phi \vee \Psi$ occurs in a tree with an inconsistent initial list, the tree will show that Φ and Ψ each lead to contradiction, which shows that the disjunction itself, together with the other statements on its path, implies a contradiction.

To mimic this in a proof, we need to prove $\Phi \rightarrow \Theta$ and $\Psi \rightarrow \Theta$, for some contradiction Θ, then use ∨E to derive Θ directly from $\Phi \vee \Psi$. Thus a single application of the ∨ rule in the tree becomes a disjunction elimination (∨E) with two subsidiary conditional proofs in the corresponding proof (here Φ is 'P', Ψ is 'Q', and Θ is 'P & ~P'):

4.	P	Q	1∨
5.	X 2, 4	X	3, 4

4.	|	P	H (for →I)
5.	|	P & ~P	2, 4 &I
6.		P → (P & ~P)	4–5 →I
7.	|	Q	H (for →I)
8.	|	P & ~P	3, 7 EFQ
9.		Q → (P & ~P)	7–8 →I
10.		P & ~P	1, 6, 9 ∨E

The two conditional proofs constructed in preparation for ∨E represent the two branches of the tree. The hypothetical derivation at lines 4–5 of the proof represents the left branch of the tree. The 'X' on the left branch of the tree corresponds to the contradiction 'P & ~P' at line 5 of the proof. The hypothetical derivation at lines 7–8 represents the right branch of the tree, and the contradiction 'P & ~P' in the proof (line 8) represents 'X' that ends the right branch of the tree. But here 'P & ~P' is obtained, not by &I from 'P' and '~P', but by the derived rule EFQ (ex falso quodlibet; see Section 4.4) from 'Q' and '~Q'.

This use of EFQ is important. Though different formulas (such as 'P' and '~P' or 'Q' and '~Q') may lead us to close paths in the tree, each occurrence of 'X' must be represented by the same contradictory formula in the proof. This is because each of the two conditionals used in ∨E (here the conditionals appearing at lines 6 and 9) must have the same consequent. So to apply ∨E (which we do here at line 10) we need to derive the same contradiction from each hypothesis. EFQ will always enable us to do this. (EFQ is, of course, not one of the ten basic rules, but we saw above that use of derived rules here is legitimate.) *In fact, to standardize our procedure, we shall arbitrarily stipulate that 'X' in any tree always represents the formula 'P & ~P' in the proof.*

The conditionals on lines 6 and 9 of the proof and the conclusion derived by ∨E at line 10 do not correspond to any particular formulas in the tree. They are, rather, part of the apparatus of disjunction elimination, which ensures that contradictions derived along different branches of the tree also follow from the formula from which those branches stem.

As in the previous example, the tree represents only the derivation of a contradiction from the hypothesized negation of the sequent's conclusion. To complete the proof, we must end this hypothetical derivation and apply final steps of –I and –E:

11.	——Q	3–10 –I
12.	Q	11 –E

Indeed, any proof derived from a tree by the method illustrated here must end in this way.

The full algorithm for converting trees for valid sequents into proofs may be stated as follows:

1. List the premises of the sequent as assumptions, then hypothesize the negation of the conclusion.
2. Derive 'P & –P' from this hypothesis by converting each step in the tree into a series of proof steps as described in Table 5.1.
3. Deduce the double negation of the sequent's conclusion by –I, and then deduce the conclusion itself by –E.

Table 5.1, as noted in step 2, provides instructions for converting each application of a tree rule into a series of steps in the proof. It remains only to verify that this algorithm performs as advertised.

TABLE 5.1
Instructions for Converting Trees for Valid Sequents into Proofs

Tree Rule	Corresponding Step(s) in Proof
Negation (–) If an open path contains both a formula Φ and its negation, place an 'X' at the bottom of the path.	Deduce from Φ and –Φ the contradiction 'P & –P', either directly by &I (if Φ is 'P') or by EFQ.
Negated Negation (––) If an open path contains an unchecked formula of the form ––Φ, check it and write Φ at the bottom of every open path that contains this newly checked formula.	Deduce Φ from ––Φ by –E.
Conjunction (&) If an open path contains an unchecked formula of the form (Φ & Ψ), check it and list Φ above Ψ at the bottom of every open path that contains this newly checked formula.	Apply &E twice to (Φ & Ψ) to obtain Φ and Ψ on separate lines.

(continued)

TABLE 5.1
Instructions for Converting Trees for Valid Sequents into Proofs (*continued*)

Tree Rule	Corresponding Step(s) in Proof
Negated Conjunction (–&) If an open path contains an unchecked formula of the form $\neg(\Phi \,\&\, \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first write $\neg\Phi$, and at the end of the second write $\neg\Psi$.	Apply DM to $\neg(\Phi \,\&\, \Psi)$ to obtain $\neg\Phi \vee \neg\Psi$, then apply the directions for disjunction to $\neg\Phi \vee \neg\Psi$.
Disjunction ($\vee$) If an open path contains an unchecked formula of the form $(\Phi \vee \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first write Φ, and at the end of the second write Ψ.	Hypothesize Φ, aiming to derive '$P \,\&\, \neg P$' and then deduce $\Phi \rightarrow (P \,\&\, \neg P)$ by $\rightarrow$I. Next, hypothesize Ψ, again derive '$P \,\&\, \neg P$', and then obtain $\Psi \rightarrow (P \,\&\, \neg P)$ by $\rightarrow$I. Finally, use $\vee$E to deduce '$P \,\&\, \neg P$' from $(\Phi \vee \Psi)$, $\Phi \rightarrow (P \,\&\, \neg P)$, and $\Psi \rightarrow (P \,\&\, \neg P)$. This procedure always works if the sequent being tested on the tree is valid, since in that case all the paths below $(\Phi \vee \Psi)$ must close, and each closed path is converted into a derivation of '$P \,\&\, \neg P$' in the proof (see negation rule). If there are further applications of branching rules below $(\Phi \vee \Psi)$, these will also be converted into derivations of '$P \,\&\, \neg P$' by further applications of the procedure for disjunction.
Negated Disjunction ($\neg\vee$) If an open path contains an unchecked formula of the form $\neg(\Phi \vee \Psi)$, check it and list $\neg\Phi$ above $\neg\Psi$ at the bottom of every open path that contains this newly checked formula.	Apply DM to $\neg(\Phi \vee \Psi)$ to obtain $\neg\Phi \,\&\, \neg\Psi$, then use &E twice to obtain $\neg\Phi$ and $\neg\Psi$ on separate lines.
Conditional ($\rightarrow$) If an open path contains an unchecked formula of the form $(\Phi \rightarrow \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first write $\neg\Phi$, and at the end of the second write Ψ.	Apply MI to $(\Phi \rightarrow \Psi)$ to obtain $\neg\Phi \vee \Psi$, then apply the directions for disjunction to $\neg\Phi \vee \Psi$.

(continued)

TABLE 5.1
Instructions for Converting Trees for Valid Sequents into Proofs (*continued*)

Tree Rule	Corresponding Step(s) in Proof
Negated Conditional ($\neg\rightarrow$) If an open path contains an unchecked formula of the form $\neg(\Phi \rightarrow \Psi)$, check it and list Φ above $\neg\Psi$ at the bottom of every open path that contains this newly checked formula.	From $\neg(\Phi \rightarrow \Psi)$, reason as follows (line numbers are represented by letters, starting with 'a'): a $\neg(\Phi \rightarrow \Psi)$ b $\quad \neg\Phi \vee \Psi$ — H (for $\neg$I) c $\quad \Phi \rightarrow \Psi$ — b MI d $\quad (\Phi\rightarrow\Psi)\,\&\,\neg(\Phi\rightarrow\Psi)$ — a, c &I e $\neg(\neg\Phi \vee \Psi)$ — b–d $\neg$I f $\neg\neg\Phi \,\&\, \neg\Psi$ — e DM g $\neg\neg\Phi$ — f &E h Φ — g $\neg$E i $\neg\Psi$ — f &E (Here Φ is proved at line h and $\neg\Psi$ at line i.)
Biconditional ($\leftrightarrow$) If an open path contains an unchecked formula of the form $(\Phi \leftrightarrow \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first list Φ above Ψ, and at the end of the second list $\neg\Phi$ above $\neg\Psi$.	From $(\Phi \leftrightarrow \Psi)$, reason as follows: a $\Phi \leftrightarrow \Psi$ b $\quad \neg((\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi))$ — H (for $\neg$I) c $\quad\quad \Phi$ — H (for $\neg$I) d $\quad\quad \Psi$ — a, c $\leftrightarrow$MP e $\quad\quad (\Phi \,\&\, \Psi)$ — c, d &I f $\quad\quad (\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)$ — e $\vee$I g $\quad\quad (\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)\,\&$ $\quad\quad \neg((\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi))$ — b, f &I h $\quad \neg\Phi$ — c–g $\neg$I i $\quad \neg\Psi$ — a, h $\leftrightarrow$MT j $\quad \neg\Phi \,\&\, \neg\Psi$ — h, i &I k $\quad (\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)$ — j $\vee$I l $\quad (\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)\,\&$ $\quad\quad \neg((\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi))$ — b, k &I m $\neg\neg((\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi))$ — b–l $\neg$I n $(\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)$ — m $\neg$E Then apply the directions for disjunction to $(\Phi \,\&\, \Psi) \vee (\neg\Phi \,\&\, \neg\Psi)$.
Negated Biconditional ($\neg\leftrightarrow$) If an open path contains an unchecked formula of the form $\neg(\Phi \leftrightarrow \Psi)$, check it and split the bottom of each open path containing this newly checked formula into two branches; at the end of the first list Φ above $\neg\Psi$, and at the end of the second list $\neg\Phi$ above Ψ.	From $\neg(\Phi \leftrightarrow \Psi)$, reason as follows: a $\neg(\Phi \leftrightarrow \Psi)$ b $\quad \neg((\Phi \,\&\, \neg\Psi) \vee (\neg\Phi \,\&\, \Psi))$ — H (for $\neg$I) c $\quad \neg(\Phi \,\&\, \neg\Psi) \,\&\, \neg(\neg\Phi \,\&\, \Psi)$ — b DM d $\quad \neg(\Phi \,\&\, \neg\Psi)$ — c &E e $\quad \Phi \rightarrow \Psi$ — d MI f $\quad \neg(\neg\Phi \,\&\, \Psi)$ — c &E g $\quad \neg\neg\Phi \vee \neg\Psi$ — f DM h $\quad \neg\Psi \vee \neg\neg\Phi$ — g COM i $\quad \neg(\Psi \,\&\, \neg\Phi)$ — h DM j $\quad \Psi \rightarrow \Phi$ — i MI

(continued)

TABLE 5.1
Instructions for Converting Trees for Valid Sequents into Proofs (*continued*)

Tree Rule	Corresponding Step(s) in Proof
	k | $\Phi \leftrightarrow \Psi$ — e, j $\leftrightarrow$I l | $(\Phi \leftrightarrow \Psi)$ & $-(\Phi \leftrightarrow \Psi)$ — a, k &I m $--((\Phi$ & $-\Psi)$ v $(-\Phi$ & $\Psi))$ — b–l –I n $(\Phi$ & $-\Psi)$ v $(-\Phi$ & $\Psi)$ — m –E Then apply the directions for disjunction to $(\Phi$ & $-\Psi)$ v $(-\Phi$ & $\Psi)$.

METATHEOREM: If the tree test classifies a sequent as valid, then that sequent can be proved using only the ten basic inference rules.

PROOF: Suppose that the tree test classifies a sequent $\Phi_1, \ldots, \Phi_n \vdash \Psi$ as valid. Then all paths of the tree whose initial list is $\Phi_1, \ldots, \Phi_n, -\Psi$ close. To construct a proof of $\Phi_1, \ldots, \Phi_n \vdash \Psi$ using only the ten basic inference rules, apply the algorithm described earlier. Now either the tree contains applications of branching rules or it does not. If it does not, then it is evident by inspection of the algorithm that each formula in the tree obtained by the tree rules is deduced from the initial assumptions and the hypothesis in the proof, and that where the final 'X' appears in the tree, the corresponding formula derived in the proof is 'P & ~P'. Hence the portion of the proof corresponding to the tree is just a straightforward derivation of 'P & ~P' from the assumptions $\Phi_1, \ldots, \Phi_n$ and the hypothesis $-\Psi$. If, on the other hand, the tree employs branching rules, the formulas that begin new branches of the tree constitute additional hypotheses in the proof.[21] Yet, since each path of the tree closes, each of these hypothetical derivations still ends with 'P & ~P'. So, in accordance with the procedures for disjunction and for the other branching rules, each time a branching rule is applied to some formula Θ in the tree, the portion of the proof generated by the algorithm is a derivation by vE of 'P & ~P' from Θ. Even if the branches themselves branch, the result is the same, since 'P & ~P' will be derived in the portion of the proof corresponding to each branch and hence

[21] In the case of the two rules for the biconditional, $\leftrightarrow$ and $-\leftrightarrow$, which produce branches beginning with two formulas each, the hypothesis corresponding to each branch in the proof is a single formula—the conjunction of these two formulas.

(by ∨E) from the formula from which the subbranches originate. Thus, whether or not the tree branches, the portion of the proof corresponding to the tree as a whole is a derivation of 'P & ~P' from the assumptions $\Phi_1, \ldots, \Phi_n$ and the hypothesis ~Ψ. Therefore we may apply step 3 of the algorithm (obtain ~~Ψ by ~I and then Ψ by ~E), completing the proof of $\Phi_1, \ldots, \Phi_n \vdash \Psi$. Though portions of the proof may use derived rules, these can be replaced as explained in Section 4.4 by derivations using only the ten basic rules. In this way we obtain a proof of $\Phi_1, \ldots, \Phi_n \vdash \Psi$ using only the ten basic rules.

Therefore, if the tree test classifies a sequent as valid, then that sequent can be proved using only the ten basic inference rules. QED

COROLLARY (Completeness of the Inference Rules): If a sequent is valid, then that sequent can be proved using only the ten basic inference rules.

PROOF: From the completeness of the tree test, we know that if a sequent is valid, then the tree test classifies it as valid. Together with the previous metatheorem, this implies that if a sequent is valid, then that sequent can be proved using only the ten basic rules. QED

Finally, we shall show that the system consisting of the ten basic inference rules is sound—that is, that any sequent provable by these rules is valid. We have already seen in Sections 4.2 and 4.3 that each rule individually is valid—that is, that there is no counterexample to any instance of any of these rules. To prove a sequent, however, we apply these rules successively. We must, then, show that invalidity does not somehow creep into a proof as a result of this succession. In order to show this, it will be useful to define the notion of a **corresponding sequent** to a line of a proof.

DEFINITION The **corresponding sequent** for a given line of a proof is the sequent whose conclusion is the formula on that line, and whose premises are all the assumptions and all the hypotheses whose derivations have not yet ended that are listed on that line or at any previous lines.

The corresponding sequent for a given line is in effect what is proved at that line. To illustrate, consider this proof of 'P → Q, ~Q ⊢ ~P'. Corresponding sequents are listed to the right.

Line of Proof			Corresponding Sequent
1.	$P \rightarrow Q$	A	$P \rightarrow Q \vdash P \rightarrow Q$
2.	$\neg Q$	A	$P \rightarrow Q, \neg Q \vdash \neg Q$
3.	P	H (for $\neg$I)	$P \rightarrow Q, \neg Q, P \vdash P$
4.	Q	1, 3 $\rightarrow$E	$P \rightarrow Q, \neg Q, P \vdash Q$
5.	Q & $\neg Q$	2, 3 &I	$P \rightarrow Q, \neg Q, P \vdash Q \,\&\, \neg Q$
6.	$\neg P$	3–5 $\neg$I	$P \rightarrow Q, \neg Q \vdash \neg P$

Since all hypothetical derivations must end before a proof is finished, the corresponding sequent for the last line of any proof is just the sequent whose premises are the proof's assumptions and whose conclusion is the proof's conclusion—that is, the sequent to be proved. Thus, if we can show that the corresponding sequent for *any* line of any proof is valid, it will follow that the corresponding sequent for *the last line* of any proof is valid, and hence that any sequent provable by the ten basic inference rules is valid. Actually, since inference rules may apply to any earlier lines, it is easier to prove something apparently a little stronger than this—namely, that each line and all lines preceding it correspond to valid sequents. This can be done by mathematical induction on the number of lines in the proof. The induction appeals frequently to the following lemma, which was problem 4 of Exercise 5.4:

LEMMA: If $\Phi_1, \ldots, \Phi_m \vdash \Psi$ and $\Psi, \Psi_1, \ldots, \Psi_n \vdash \Theta$ are valid sequents (where $m \geq 0$ and $n \geq 0$), then $\Phi_1, \ldots, \Phi_m, \Psi_1, \ldots, \Psi_n \vdash \Theta$ is a valid sequent.

Here is the induction itself:

METATHEOREM: Let P be any proof using only the ten basic inference rules; then for each line of P, the corresponding sequents for all lines up to and including that line are valid.

PROOF: The lines of P form a series, so we may proceed by mathematical induction. The property which we show belongs to each line is rather convoluted. It is the property of being a line of P such that the corresponding sequents for it and all previous lines are valid.

Basis Case: The first line of any proof is always the assumption or hypothesis of some formula Ψ. Since there are no lines previous to this first line, the corresponding sequent is $\Psi \vdash \Psi$. This is clearly valid (see problem 7 of Exercise 5.3). Hence the first line of P has the property of being such that the corresponding sequents for it and all previous lines are valid.

Inductive Step: Suppose that the corresponding sequents of all lines up to and including the nth line are valid. We must show that the

corresponding sequents of all lines up to and including the $(n + 1)$st line are valid. To do this, it suffices to show just that the corresponding sequent for the $(n + 1)$st line is valid. Now in a proof using only the ten basic inference rules there are only twelve ways in which the $(n + 1)$st line can be obtained: It may be an assumption, or a hypothesis, or a conclusion obtained by one of the ten basic rules. If it is an assumption or hypothesis Ψ, then the corresponding sequent must also have Ψ as both premise and conclusion (though it may have other premises as well), and so this sequent is clearly valid. (This follows from problem 7 of Exercise 5.3 together with problem 3 of Exercise 5.4.) Now we must show for each of the ten rules that when applied to lines whose corresponding sequents are valid this rule produces a line whose corresponding sequent is valid. We shall do this for $\sim$E, $\rightarrow$E, and $\sim$I, leaving the remaining seven cases as exercises. *First, we show that if $\sim$E is applied to a line whose corresponding sequent is valid, the resulting conclusion is a line whose corresponding sequent is valid.* To do this, we proceed by conditional proof.

> Suppose $\sim$E is applied to a line whose corresponding sequent is valid. Now since $\sim$E is applied to this line, the formula it contains must be of the form $\sim\sim\Theta$. Since the conclusion of the corresponding sequent for this line must be the formula that appears on this line, the corresponding sequent must have the form $\Phi_1, \ldots, \Phi_m \vdash \sim\sim\Theta$, where $m \geq 0$. Now $\sim$E has been applied to $\sim\sim\Theta$ to obtain Θ. Therefore the sequent corresponding to the line obtained by $\sim$E is of the form $\Phi_1, \ldots, \Phi_m, \Psi_1, \ldots, \Psi_n \vdash \Theta$, where $n \geq 0$. (Here $\Psi_1, \ldots, \Psi_n$ are any hypotheses or assumptions that may have been introduced after the line at which $\sim\sim\Theta$ appears.[22]) Now since $\Phi_1, \ldots, \Phi_m \vdash \sim\sim\Theta$ is valid, and we saw in Section 4.2 that $\sim\sim\Theta \vdash \Theta$ is valid, it follows by the lemma that $\Phi_1, \ldots, \Phi_m \vdash \Theta$ is valid. (In terms of the lemma, Ψ is $\sim\sim\Theta$ and $n = 0$.) And since $\Phi_1, \ldots, \Phi_m \vdash \Theta$ is valid, by problem 7 of Exercise 5.3, $\Phi_1, \ldots, \Phi_m, \Psi_1, \ldots, \Psi_n \vdash \Theta$ is valid.

Hence we have shown that if $\sim$E is applied to a line whose corresponding sequent is valid, the resulting conclusion is a line whose corresponding sequent is valid. *Next, we shall show, again by conditional proof, that if $\rightarrow$E is applied to a pair of lines, each of whose corresponding sequents is valid, the result is a line whose corresponding sequent is valid.*

[22] Notice that none of the original hypotheses or assumptions $\Phi_1, \ldots, \Phi_m$ can be dropped, since that would indicate that $\sim\sim\Theta$ appears in a hypothetical derivation that has ended, so that neither $\sim$E nor any other rule may be applied to it.

Suppose that →E is applied to two lines each of whose corresponding sequents is valid. The formulas on these lines are therefore of the forms $\Phi \rightarrow \Psi$ *and* Φ*, and their corresponding sequents have the forms* $\Phi_1, \ldots, \Phi_m \vdash \Phi \rightarrow \Psi$ and $\Delta_1, \ldots, \Delta_n \vdash \Phi$, where $m \geq 0$ and $n \geq 0$. The line obtained by the application of →E is of the form Ψ, and its corresponding sequent has the form $\Psi_1, \ldots, \Psi_p \vdash \Psi$, where $p \geq 0$ and $\Phi_1, \ldots, \Phi_m$ and $\Delta_1, \ldots, \Delta_n$ are included among $\Psi_1, \ldots, \Psi_p$.[23] Now we saw in Section 4.2 that $\Phi \rightarrow \Psi, \Phi \vdash \Psi$ is a valid form. Hence, since $\Phi_1, \ldots, \Phi_m \vdash \Phi \rightarrow \Psi$ is valid, it follows by the lemma that $\Phi_1, \ldots, \Phi_m, \Phi \vdash \Psi$ is valid. Given this and the fact that $\Delta_1, \ldots, \Delta_n \vdash \Phi$ is valid, it follows again by the lemma that $\Phi_1, \ldots, \Phi_m, \Delta_1, \ldots, \Delta_n \vdash \Psi$ is valid. But since $\Phi_1, \ldots, \Phi_m$ and $\Delta_1, \ldots, \Delta_n$ are included among $\Psi_1, \ldots, \Psi_p$, from this by problem 3 of Exercise 5.4 we may infer that $\Psi_1, \ldots, \Psi_p \vdash \Psi$ is valid. But this is the corresponding sequent for the line obtained by →E.

Therefore, if →E is applied to a pair of lines, each of whose corresponding sequents is valid, the result is a line whose corresponding sequent is valid. Finally, we show that *if ¬I is applied to a series of lines all of whose corresponding sequents are valid, the result is a line whose corresponding sequent is valid.*

Suppose that ¬I is applied to a series of lines all of whose corresponding sequents are valid. For ¬I to be applicable, the first such line must contain a hypothesized formula Φ and the last must contain a contradiction $\Psi \,\&\, \neg\Psi$ that is derived from Φ. The corresponding sequent of this last line must therefore have the form $\Phi, \Phi_1, \ldots, \Phi_m \vdash \Psi \,\&\, \neg\Psi$, where $m \geq 0$. Since we have supposed this sequent to be valid, there is no valuation on which $\Phi, \Phi_1, \ldots, \Phi_m$ are all true and $\Psi \,\&\, \neg\Psi$ is untrue. But since $\Psi \,\&\, \neg\Psi$ is untrue on all valuations, there is no valuation on which $\Phi, \Phi_1, \ldots, \Phi_m$ are all true. Hence by valuation rule 1, there is no valuation on which $\Phi_1, \ldots, \Phi_m$ are all true and $\neg\Phi$ is not true. Therefore the sequent $\Phi_1, \ldots, \Phi_m \vdash \neg\Phi$ is valid. But since application of ¬I ends the hypothetical derivation from Φ, leaving only $\Phi_1, \ldots, \Phi_m$ as the assumptions or hypotheses whose derivations have not ended, this sequent is just the corresponding sequent for the line obtained by ¬I.

[23] The reason for this inclusion is explained in the previous footnote.

Therefore, if –I is applied to a series of lines all of whose corresponding sequents are valid, the result is a line whose corresponding sequent is valid. Similar results may be obtained for each of the remaining seven rules: &I, &E, ∨I, ∨E, →E, ↔I, and ↔E. Hence no matter by which of the twelve possible ways the $(n + 1)$st line is obtained, the corresponding sequent for the $(n + 1)$st line is valid.

Therefore, if the corresponding sequents of all lines of P up to and including the nth line are valid, then so are the corresponding sequents for all lines up to and including the $(n + 1)$st. Hence by mathematical induction, for each line of P, the corresponding sequents for all lines up to and including that line are valid. QED

From this result, the soundness of the ten rules follows as a corollary:

COROLLARY (Soundness of the Ten Basic Inference Rules): If a sequent can be proved using only the ten basic inference rules, then that sequent is valid.

PROOF: Let $\Phi_1, \ldots, \Phi_n \vdash \Psi$ be a sequent provable using only the ten basic rules. Then there exists a proof of this sequent whose assumptions are $\Phi_1, \ldots, \Phi_n$ and whose last line contains the formula Ψ. Since all hypothetical derivations used in this proof must have ended before the last line, the corresponding sequent for this last line is just $\Phi_1, \ldots, \Phi_n \vdash \Psi$ (see definition of corresponding sequent). But by the previous metatheorem, the corresponding sequent for any line of any proof is valid. Hence $\Phi_1, \ldots, \Phi_n \vdash \Psi$ is valid.

Hence, if a sequent can be proved using only the ten basic inference rules, then that sequent is valid. QED

Having shown that the ten basic inference rules are sound and complete (i.e., that they enable us to prove a sequent of propositional logic iff it is valid), we now know that they enable us to prove exactly the sequents we should be able to prove, given the classical notion of validity.

Exercise 5.10.1

Prove for each of the rules &I, &E, ∨I, ∨E, →E, ↔I, and ↔E that if applied to lines whose corresponding sequents are valid, they yield a conclusion whose corresponding sequent is valid.

Exercise 5.10.2

1. A set of rules is **consistent** iff there is no formula Φ such that both Φ and $\neg\Phi$ are provable as theorems from these rules. Use the soundness of the ten basic inference rules to prove that these rules are consistent.
2. Use the soundness and completeness of the ten basic inference rules to prove that a formula of propositional logic is valid (tautologous) iff it is a theorem.

PART III

CLASSICAL PREDICATE LOGIC