Symbolic Logic

Theory and Applications

Randy Ridenour

Contents

Chapter 1

Basic Concepts

1.1 What is Logic?

I don't know that there is any accepted definition of logic. Here are four that I've heard:

1. The study of reasoning
2. The analysis and evaluation of arguments
3. The study of formal languages
4. The study of patterns of truth

Although I don't think that any of the four are successful definitions, it is certainly the case that they each represent some important part of the study of logic. We will see all four of these topics, in various forms, in later chapters. This chapter introduces some important concepts that are fundamental to the study of logic. The first is truth preservation.

The study of logic provides a set of tools for reasoning. As you continue in the study of logic, you will find that there are different systems of logic, each system being a different tool that can be used. There are two obvious, but important, things to note — first, the particular tool that should be used is relative to the context or task at hand. Hammers and saws are both tools used in carpentry. Which one should you use? Is your goal to drive a nail or cut a board? We'll see that the system of logic that should be used depends on the context in which one is reasoning. A good rule of thumb is that one should use the simplest tool that is adequate for the task. For example, a

doctor shouldn't use an electrocardiogram or MRI when a stethoscope will do. Systems of logic also differ in degrees of complexity, and we should use the simplest system that will be effective in the circumstances.

Second, a tool that leaves you in a worse situation than you were in before you used it is certainly *not* good. The purpose of a toaster is to make a nicely toasted piece of bread. A toaster that produces a smoldering, inedible, piece of charcoal is not a good toaster. You were better off just eating the non-toasted bread. A central purpose of a logic system is to preserve truth. When the tools of logic are applied to truths, the results should still be true. A system of logic that turns truth into falsehoods is a failure as a tool of reasoning. Another way to state this is that the rules of our logic systems should be truth-preserving.

1.2 Arguments

One fundamental concept in logic is that of the argument. For a good example of what we are not talking about, consider a bit from a famous sketch by *Monty Python's Flying Circus* (Cleese and Chapman 1980):

> Man: (Knock)
> Mr. Vibrating: Come in.
> Man: Ah, Is this the right room for an argument?
> Mr. Vibrating: I told you once.
> Man: No you haven't.
> Mr. Vibrating: Yes I have.
> Man: When?
> Mr. Vibrating: Just now.
> Man: No you didn't.
> Mr. Vibrating: Yes I did.
> Man: You didn't!
> Mr. Vibrating: I did!
> Man: You didn't!
> Mr. Vibrating: I'm telling you I did!
> Man: You did not!!
> Mr. Vibrating: Oh, I'm sorry, just one moment. Is this a five minute argument or the full half hour?

People often use "argument" to refer to a dispute or quarrel between people. For our purposes, an argument is defined as

Argument A set of statements, one of which is taken to be supported by the remaining sentences.

Conclusion The statement in the argument that is being supported.

Premises The statements in the argument that provide the support for the conclusion.

There are three important things to remember here:

1. Arguments contain statements.
2. They have a conclusion.
3. They have at least one premise

This might get more complicated later, but for now, think of a statement as a sentence that has a truth-value. Also, for now, we will assume that there are only two truth values: true and false, which we will abbreviate with **T** and **F**. Think of a statement as an attempt to describe the world; when it gets it right, the statement is true, otherwise, it is false. Since George Washington was in fact the first President of the United States, the statement 'George Washington was the first President of the United States' has the truth value of **T**. Likewise, the statement 'John Adams was the first President of the United States' has the truth value of **F**.

Some sentences don't have truth values, that is to say that they are neither true or false. Examples are questions ('What did you do last summer?') and commands ('Please take a seat.'). Such sentences can be neither premises nor conclusions of arguments. It is important to note that, even though a statement has a truth value, we may not necessarily know what that truth value is. The rules of logic will still apply, though.

An inference is the process of reasoning from the premises to the conclusion. That is, inference is the psychological process by which one draws a conclusion from some given premises. The argument just is the premises and conclusion.

A distinction needs to be made between deductive arguments and inductive arguments. It's very hard to make this distinction precise, but it's not

hard to have an informal understanding of the difference. Think of the difference in terms of what the argument is intended to establish. If the argument is only intended to establish the probable truth of the conclusion, then it is inductive. If the argument is intended to guarantee the truth of the conclusion, then it is deductive.

Every argument has exactly one conclusion. Very complex arguments may have sub-conclusions, which are themselves inferred from premises. These sub-conclusions then serve as premises for the main conclusion of the argument. Let's keep things simple for now. Consider this argument:

> Calculus II will be no harder than Calculus I. Susan did well in Calculus I. So, Susan should do well in Calculus II.

Here the conclusion is that Susan should do well in Calculus II. The other two sentences are premises. These premises are the reasons offered for believing that Susan should do well in the course..

Now, to make the argument easier to evaluate, we will put it into what is called "standard form." To put an argument in standard form, write each premise on a separate, numbered line. Draw a line underneath the last premise, the write the conclusion underneath the line.

1. Calculus II will be no harder than Calculus I.
2. Susan did well in Calculus I.

∴ Susan will do well in Calculus II.

Now that we have the argument in standard form, we can talk about premise 1, premise 2, and all clearly be referring to the same statement.

Unfortunately, when people present arguments, they rarely put them in standard form. So, we have to decide which statement is intended to be the conclusion, and which are the premises. Don't make the mistake of assuming that the conclusion comes at the end. The conclusion is often at the beginning of the passage, but could even be in the middle. A better way to identify premises and conclusions is to look for indicator words. Indicator words are words that signal that statement following the indicator is a premise or conclusion. The example above used a common indicator word for a conclusion, 'so.' The other common conclusion indicator, as you can probably guess, is 'therefore.' This table lists the indicator words you might encounter:

Conclusion	**Premise**
Therefore	Since
So	Because
Thus	For
Hence	Is implied by
Consequently	For the reason that
Implies that	
It follows that	

Table 1.1: Indicator words.

Each argument will likely use only one indicator word or phrase. When the conclusion is at the end, it will generally be preceded by a conclusion indicator. Everything else, then, is a premise. When the conclusion comes at the beginning, the next sentence will usually be introduced by a premise indicator. All of the following sentences will also be premises.

For example, here's our previous argument rewritten to use a premise indicator:

> Susan should do well in Calculus II, because Calculus II will be no harder than Calculus I, and Susan did well in Calculus I.

Sometimes, an argument will contain no indicator words at all. In that case, the best thing to do is to determine which of the premises would logically follow from the others. If there is one, then it is the conclusion. Here is an example:

> Spot is a mammal. All dogs are mammals, and Spot is a dog.

The first sentence logically follows from the others, so it is the conclusion. When using this method, we are forced to assume that the person giving the argument is rational and logical, which might not be true.

One thing that complicates our task of identifying arguments is that there are many passages that, although they look like arguments, are not arguments. The most common types are:

1. Explanations
2. Mere assertions

3. Conditional statements
4. Loosely connected statements

Explanations can be tricky, because they often use one of our indicator words. Consider this passage:

> Abraham Lincoln died because he was shot.

If this were an argument, then the conclusion would be that Abraham Lincoln died, since the other statement is introduced by a premise indicator. If this is an argument, though, it's a strange one. Do you really think that someone would be trying to prove that Abraham Lincoln died? Surely everyone knows that he is dead. On the other hand, there might be people who don't know how he died. This passage does not attempt to prove that something is true, but instead attempts to explain why it is true. To determine if a passage is an explanation or an argument, first find the statement that looks like the conclusion. Next, ask yourself if everyone likely already believes that statement to be true. If the answer to that question is yes, then the passage is an explanation.

Mere assertions are obviously not arguments. If a professor tells you simply that you will not get an A in her course this semester, she has not given you an argument. This is because she hasn't given you any reasons to believe that the statement is true. If there are no premises, then there is no argument.

Conditional statements are sentences that have the form "If…, then…" A conditional statement asserts that *if* something is true, then something else would be true also. For example, imagine you are told, "If you have the winning lottery ticket, then you will win ten million dollars." What is being claimed to be true, that you have the winning lottery ticket, or that you will win ten million dollars? Neither. The only thing claimed is the entire conditional. Conditionals can be premises, and they can be conclusions. They can be parts of arguments, but that cannot, on their own, be arguments themselves.

Finally, consider this passage:

> I woke up this morning, then took a shower and got dressed. After breakfast, I worked on chapter 1 of the logic text. I then took a break and drank some more coffee.…

This might be a good description of my day, but it's not an argument. There's nothing in the passage that plays the role of a premise or a conclusion. The passage doesn't attempt to prove anything. Remember that arguments need a conclusion, there must be something that is the statement to be proved. Lacking that, it simply isn't an argument, no matter how much it looks like one.

1.3 Deductive Validity

Deductive arguments are intended to be fully truth-preserving. A deductively valid argument is one that is in fact completely truth-preserving. That is, a deductively valid argument will never have all true premises and a false conclusion. It is important to understand how strong this claim is. It is not merely the case a valid argument happens to have true premises and a true conclusion. The relationship between the premises and the conclusion is so strong that it is not possible for the premises to be true and the conclusion false.

Deductive validity An argument is deductively valid if and only if it is not possible for the premises to be true and the conclusion to be false.

Deductive invalidity An argument is deductively invalid if and only if it is not deductively valid.

Here is an example of a deductively valid argument:

1. All dogs are mammals.
2. Lucy is a dog.

∴ Lucy is a mammal.

Since the first premise is true, being a dog is enough to guarantee that the animal is a mammal. So, the only way that the conclusion could be false is if Lucy is not a dog. It is impossible that the premises be true and the conclusion false.

Compare the previous argument to this one:

1. All dogs are mammals.
2. Lucy is a mammal.

∴ Lucy is a dog.

Since Lucy is the name of my dog, both the premises and the conclusion are in fact true. Note, though, that the premises are not enough to guarantee the truth of the conclusion. Lucy could have been the name of a cat. If so, the premises would have been true and the conclusion false. This argument is therefore invalid.

If we know that an argument is invalid, then we know that there is a special logical relationship between the premises and the conclusion such that, if the premises are true, then the conclusion must also be true. It is important to understand that validity alone does not mean that the premises and conclusion are true. If I know only that an argument is valid, I know that the *if* the premises are true, *then* the conclusion must also be true. Valid arguments cannot have a combination of true premises and false conclusion, but they can have any other combination. It can be reasonable to doubt that a conclusion is true, even if the argument is valid. What is not reasonable is to grant the argument is valid and has true premises and still doubt that the conclusion is true. This means that validity alone is not enough to guarantee that a conclusion is true. What guarantees that a conclusion is true is deductive validity along with true premises. This is called deductive soundness.

Deductive soundness An argument is deductively sound if and only if it is deductively valid and has all true premises.

1.4 Inductive Arguments

Later chapters will cover inductive arguments. For now it is enough to say that inductive arguments are not intended to be deductive valid. That is, inductive arguments are those whose premises do not guarantee the truth of the conclusion. For inductive arguments, it is always possible that the premises be true and the conclusion false. Here is an example of an inductive argument:

> A random sample of 100 students at the university unanimously reported preferring traditional classes to online instruction. Therefore, the majority of all students at the university prefer traditional classes to online instruction.

The truth of the premise does not guarantee the truth of the conclusion. It is certainly possible that the sample managed to include the only students that don't prefer online courses. Still, though, it seems that, if the premise is in fact true, then the conclusion should be highly likely to be true. So, this is a good inductive argument, one that we call inductively strong.

Inductive strength An argument is inductively strong to the extent that the conclusion is probably true given the truth of the premises.

Another difference between inductive and deductive arguments is that inductive strength is a matter of degree. The argument above is inductively strong, but doubling the sample size would make it even stronger.

1.5 Logical Consistency and Logical Truth

Consistency is a property of sets of statements:

Logical consistency A set is logically consistent if and only if it is possible for all of the members of the set to be true at the same time.

Logical inconsistency A set is logically inconsistent if and only if it is not logically consistent.

It is not necessary for all of the statements to be true in order for the set to be consistent. Here is an example:

Oklahoma is south of Texas. There are 125 members of the U.S. Senate.

Neither of the statements in this set are true. The set is consistent, though, because there is nothing about either sentence that prevents the other from possibly being true. Here is an example of an inconsistent set:

No student will make an A in Logic this semester. At least one student will make an A in Logic this semester.

The truth of one of those statements is incompatible with the truth of the other. So, the set containing both is inconsistent. Logical consistency is a very important concept, because, once defined, other logical concepts can be defined in terms of consistency. We'll do this in a later chapter.

For the most part, logic alone is not enough to determine if a statement is true. It is not logic that makes it true that Topeka is the capital of Kansas. What makes that true is something about the political structure and history

of Kansas. There are two important exceptions to this, though. There are some statements that are true simply because of their logical structure. An important example is something like this: Either Susan will pass logic this semester or Susan will not pass logic this semester. No matter how well Susan does in the class, it must be true that she either passes or not. Sentences like these are called logical truths, or tautologies. Logical truths *must* be true. On the other hand, there are sentences that *cannot* be true. For example, Susan will both pass logic this semester and not pass logic this semester. Sentences like these are called logical falsehoods, or contradictions. Most statements are neither logical truths nor logical falsehoods. Such statements are logically indeterminate, or contingent.

Logical truth A statement is logically true if and only if it is not possible for the statement to be false.

Logical falsity A statement is logically false if and only if it is not possible for the statement to be true.

Logical indeterminacy A statement is logically indeterminate if and only if it is neither logically true nor logically false.

Finally, there are sentences that are related in such a way so that, if one is true, the second must also be true, and vice versa. These sentences are called logically equivalent. Here is a simple example:

Susan will pass.
Susan will not fail.

Since failing is just not passing, then any situation in which Susan passes is a situation in which she does not fail. So, the two statements are true in exactly the same situations, and false in exactly the same situations. They always have the same truth values.

Logical equivalence Two sentences are logically equivalent if and only if it is impossible for one to be true and the other to be false.

Chapter 2

Categorical Logic

Now we turn to some structured logic systems. The first, categorical logic, is one of the oldest. It dates back at least to Aristotle (384–322 BCE). Categorical logic is a fairly simple logic of categories or classes. A class is a group of things that we designate with a common noun: students, teachers, dogs, politicians, etc. Each sentence will use two different classes. One is the subject class, and the other is the predicate class. In this logic, we can say something about all members of a class, called a universal sentence, or we can say something about some members of a class, called a particular sentence. We can also make a positive claim, called an affirmation, or we can make a negative claim, called a negation.

With these two distinctions, universal/particular and affirnation/negation, we can make four kinds of sentences. S and P stand for the subject class and the predicate class, respectively.

A : All S are P (universal affirmation)

E : No S are P (universal negation)

I : Some S are P (particular affirmation)

O : Some S are not P (particular negation)[1]

Here are some examples of categorical statements, some true and some false.

[1]The letters A, E, I, and O, are thought to come from the first two vowels of the Latin words *affirmo* and *nego*, meaning "I affirm" and "I deny."

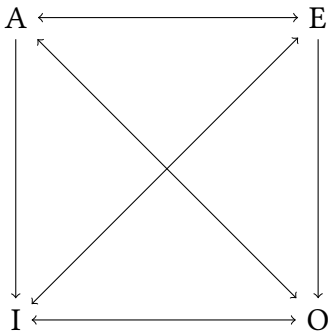
1. All dogs are mammals.
2. All mammals are dogs.
3. No reptiles are dogs.
4. No politicians are honest people.
5. Some politicians are honest people.
6. Some cats are amphibians.
7. Some dogs are not beagles.
8. Some beagles are not dogs.

Look at the sentences carefully. You should be able to tell that the odd-numbered ones are true and the even-numbered ones are false.

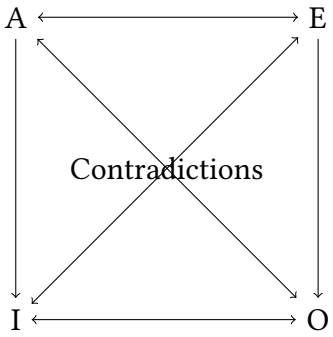
2.1 The Square of Opposition

We can visualize interesting logical relationships between these four types of sentences with something called "The Square of Opposition."

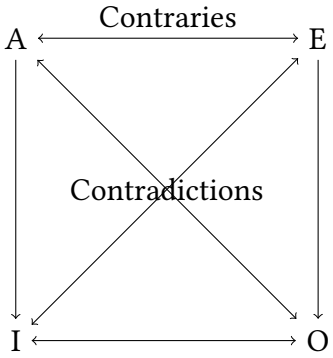
The first step is to place the sentence types in the corners of an imaginary square. A is at the upper left; E, the upper right; I, the lower left, and O, the lower right. Next, draw arrows on the diagonals, pointing to the sentences in the corners. Then, draw an arrow between the two at the top, and another one between the two at the bottom. Finally, draw an arrow on each side, going from top to bottom. When finished, you should have something like this:



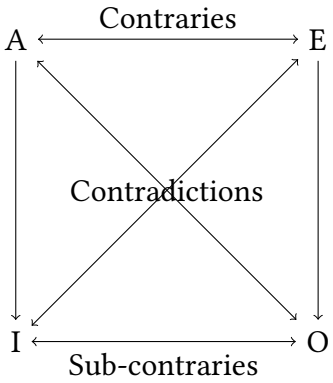
The next step is to note the relationship between the diagonals. The diagonals are contradictories, meaning they always have opposite truth values. They can't both be true, and they also can't both be false. If the A sentence is true, the O sentence must be false—if it is true that all dogs are mammals, it cannot be true that some dogs are not mammals. If the O sentence is true, then the A sentence must be false. It is the same for the E and the I.



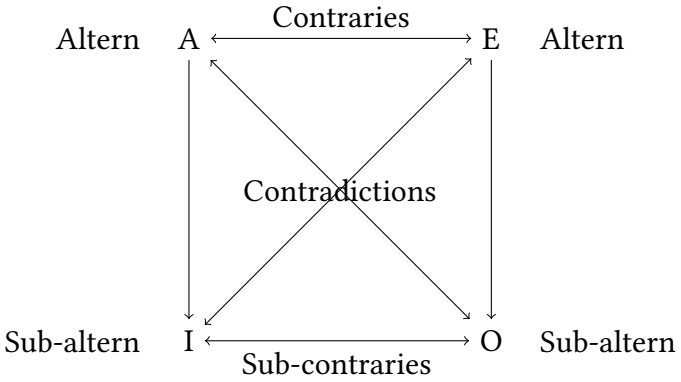
Next, note the relationship between the A sentences and the E sentences, called contraries. Like the contradictories, they cannot both be true. Unlike the contradictories, they can both be false. If it's true that all critical thinking students are good students, then it must be false that no critical thinking students are good students. If it's false that all critical thinking students are good students, then it can be false that critical thinking students are good students. In fact, they are both false, because some critical thinking students are good and others are not.



At the bottom, we have sub-contraries. They can both be false, but cannot both be true.



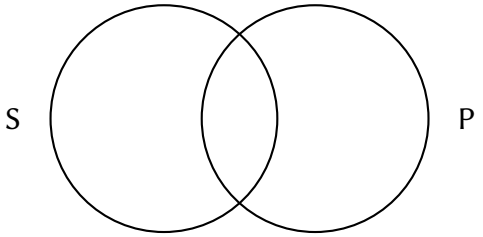
Finally, we have the relationship between the top level sentences and the bottom level sentences on the same side. This is called alternation. The universal is called the superaltern and the particular is called the subaltern. If the superaltern is true, then the subaltern must also be true. If the superaltern is false, then the subaltern can be either true or false. If the subaltern is false, then the superaltern must be false. If the subaltern is true, then the superaltern can be either true or false. It is easy to remember this way: truth goes down, falsity goes up.



2.2 Diagramming Sentences

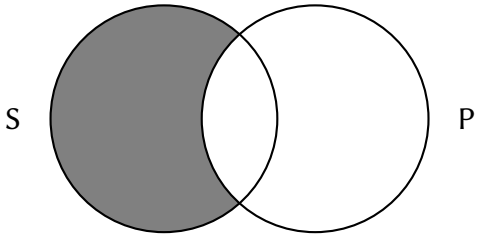
We diagram sentences and arguments in categorical logic using Venn diagrams. You've probably used these in a math class at some time. Before we can use these to evaluate arguments in categorical logic, we first have to learn how to diagram individual sentences.

The first step is to draw two interlocking circles. Label the left circle with an "S" and the right circle with "P"—standing for the subject term and predicate term, respectively.



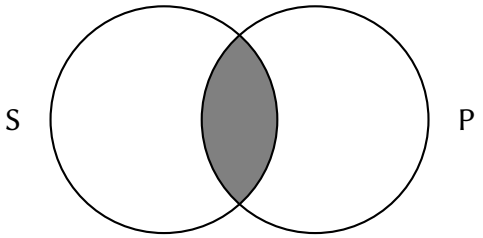
A-Sentences

Remember that the A-sentence has the form All S are P. That means that everything that is in the S circle must also be in the P circle. To diagram this, we shade the region of the S circle that is not contained in the P circle. If a region is shaded, that means that nothing is in that region.



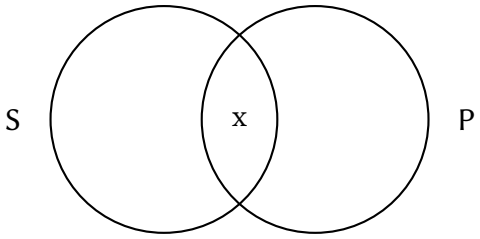
E-Sentences

To shade the universal negation, we shade the region that is shared by both S and P:



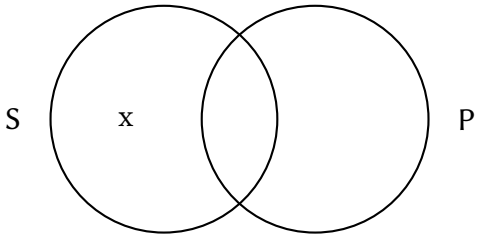
I-Sentences

To diagram a particular affirmation, we place an x in the region shared by S and P:



O-Sentences

Finally, to diagram an O-sentence, we place an x in S, but not in P:



Evaluating Categorical Syllogisms

A syllogism is an argument that has two premises and a conclusion. A categorical syllogism is a syllogism that contains only categorical sentences. Here is an example:

1. All Dogs are mammals.
2. All mammals are animals.

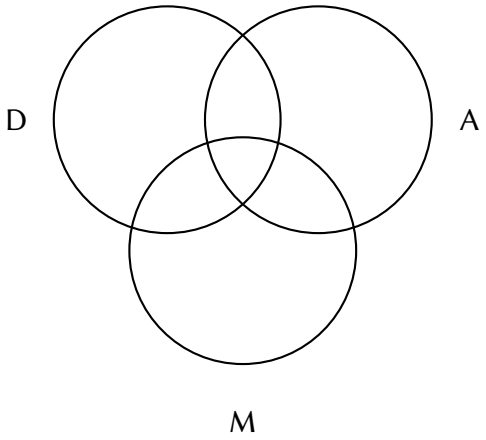
∴ All dogs are animals

Both premises and the conclusion are A-sentences. Notice that we have three terms in the argument: dogs, mammals, and animals. Every categorical syllogism, in proper form, has three terms. Each term occurs in two sentences. Two of those terms will be found in the conclusion, and one term is only in the premises. The predicate term of the conclusion is called the major term. The subject of the conclusion is called the minor term. The term that is not in the conclusion is called the middle term.

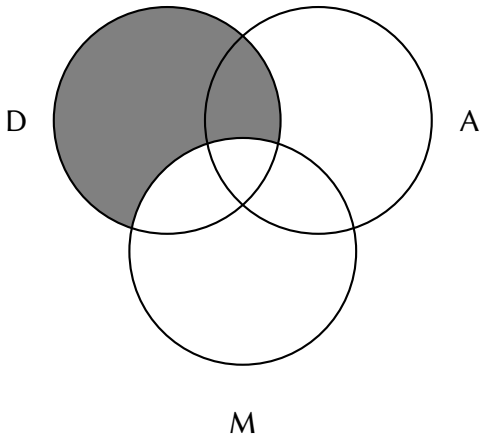
There are two ways to determine if a categorical syllogism is valid. One way uses Venn diagrams, and the other involves applying some simple rules.

Diagram Method

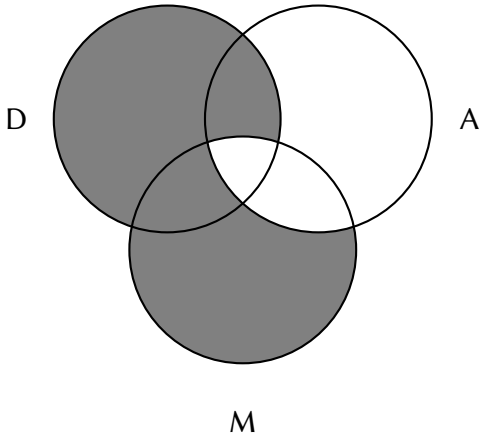
Since we have three terms in the argument, we'll need three intersecting circles. We'll start by drawing two circles for the conclusion, just as we did before. Then, in the middle and below, we'll draw another circle for the middle term. For labels, use letters that correspond to the classes in the argument. Here, we'll use D for dogs, M for mammals, and A for animals.



Next, we finish diagramming the premises by shading or placing an x. Since our first premise is "All dogs are mammals", we need to shade everything in the D circle that is not in the M circle.



Next, we diagram the second premise by shading everything that is in the M circle but not in the A circle.

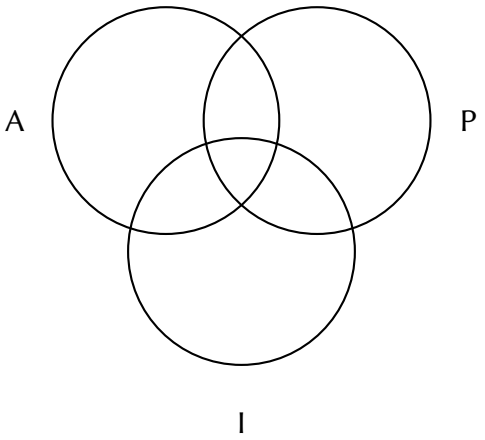


If there is any circle that has only one region left unshaded, you can place an 'X' in that region. This is because categorical logic assumes that there are no empty categories, meaning that every category has at least one thing in it. This is really only important for arguments that have an I or an O-sentence for a conclusion. In this case, we won't worry about it. Now that the premises are diagrammed, check to see if the conclusion has also been diagrammed, which in this case means that everything in the D circle that is not also in the A circle is shaded out. If so, then the argument is valid. This shows that making the premises true was enough to make the conclusion true also.

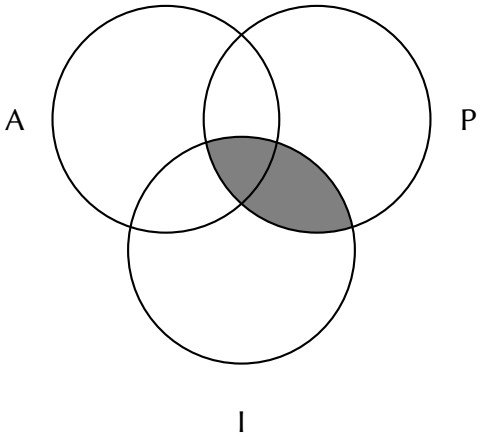
Let's try to diagram this argument:

1. No introverts are politicians
2. All artists are introverts
3. No artists are politicians

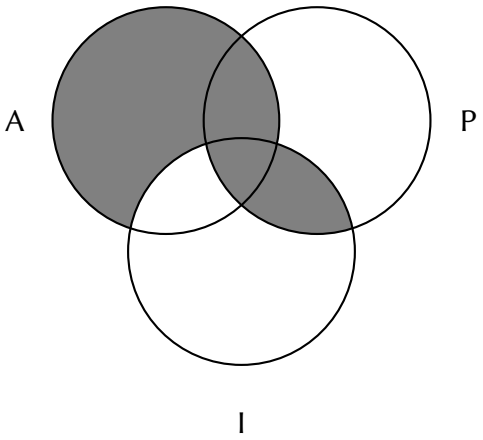
First, we draw and label the circles:



Then we diagram the premises, always doing the universals before any particulars. In this case, we have two universal premises, so we will just begin with the first premise:



Now, we'll diagram the second premise:

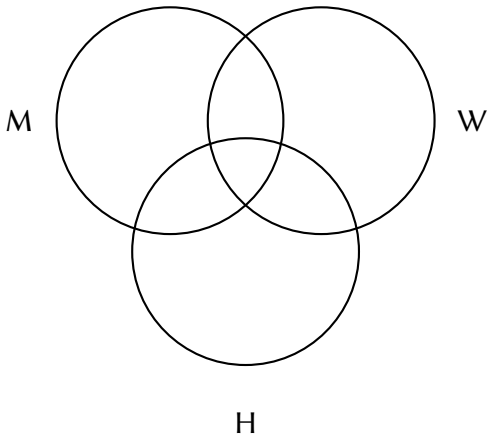


Diagramming the conclusion would require the intersection of A and P to be shaded. Notice, though, that the region between A and P has already been shaded by just diagramming the premises. That means that making the premises true was enough to guarantee that the conclusion would also be true, and the argument is valid.

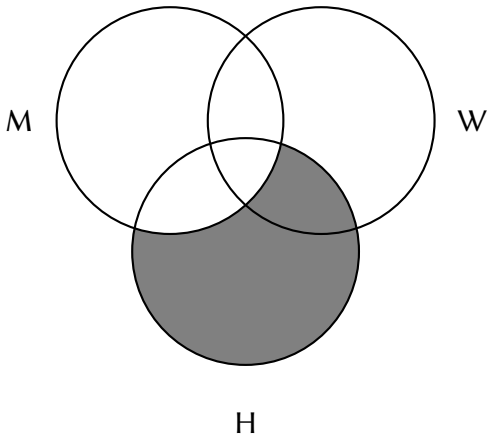
Let's try one more argument.

1. Some horses are things that weigh over 2,000 pounds.
2. <u>All horses are mammals.</u>
3. Some mammals are things that weigh over 2,000 pounds.

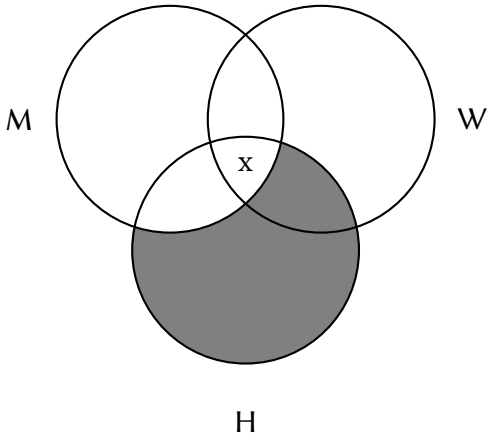
Again, we begin by drawing and labeling the circles.



The we diagram any universal premises, which, in this case, is the second premise.



Then, we diagram any particular premises.



Finally, we check to see if diagramming the premises was enough to make the conclusion also diagrammed. In this case, it was, so the argument is valid.

Hints for Diagramming Categorical Syllogisms

1. Diagram universals before particulars (shade before making an x.)
2. If it is not clear where the x goes, then put it on the line.

2.3 Rules for Categorical Syllogisms

There is another way to determine validity for categorical syllogisms. Every valid syllogism must meet three conditions:

1. There must be the same number of negations in the conclusion as in the premises.
2. The middle term must be distributed at least once.
3. Any term distributed in the conclusion must be distributed in the premises.

Before these rules can be applied, we'll have to explain what distribution is. Every categorical statement says something about a category or class. A statement distributes a term just in case what it says about that class is true of every subset of the class. For example, it is true that all dogs are mammals. It's also true that all members of any subset of the set of dogs are mammals—all dogs in Oklahoma are mammals, and all dogs in Greece are mammals, and so on. All dogs are not necessarily members of every subset of the class of mammals, however. The class of cats is a subset of the class of mammals, and no dog is a cat. So, the subject of an A-sentence is distributed, but the predicate is not. To remember when something is distributed, keep this in mind:

1. Universals distribute subjects, and
2. Negations distribute predicates.

So, A-sentences distribute the subject, E-sentences distribute both terms, I-sentences don't distribute anything, and O sentences distribute the predicate.

The rules are easy to apply. First, put the argument in standard form:

1. All A are B.
2. All B are C.

∴ All A are C.

Then, circle all of the distributed terms.

1. All Ⓐ are B.
2. All Ⓑ are C.

∴ All Ⓐ are C.

Now, just check to see if there are any violations of the rules:

1. Are there the same number of negations in the conclusion as in the premises? Yes, since there are no negations at all.

2. Is the middle term distributed at least once? Yes, the middle term is B and it is distributed in the second premise.
3. Is any term that distributed in the conclusion also distributed in the premises? Yes, A is distributed in the conclusion, but it is also distributed in the first premise.

So, since the argument breaks none of the rules, it is valid.

2.4 Relations of Equivalence

Properly formed categorical syllogisms have only three terms. Unfortunately, some arguments that you will encounter won't always be in proper form. One common way this happens is for a person to use a term like "Americans" in one premise, but use "non-Americans" in another. This can result in a syllogism with four or more terms, making it impossible to evaluate using either of our two methods. What we then need to do is to convert the sentence using one of the terms into a logically equivalent sentence that uses the other term.

There are three operations that can be applied to categorical sentences: conversion, obversion, and contraposition. It is important to know both how to apply them and in what cases does an operation result in an equivalent sentence. We're particularly interested in the conditions that those different operations are *truth-preserving*. An operation is truth preserving when, applied to a true sentence, it always results in a true sentence.

Conversion

Conversion is the simplest of the three. The converse of a sentence simply exchanges the subject and predicate terms of the original sentence. Conversion applied to A-sentences is *not* truth-preserving. "All dogs are mammals" is true, but "All mammals are dogs" is not. Conversion is truth-preserving for E-sentences and I-sentences. If it is true that no dogs are reptiles, it must be true that no reptiles are dots. Likewise, if it is true that some dogs are brown things, it must be true that some brown things are dogs.

Another way to think about this is to consider what the diagrams would like before the change and after the change. Before the change, the diagram looks like figure below, with the intersection of the S and P circles shaded.

After the change, the diagram looks like figure , with the intersection of the S and P circles shaded. Essentially, there's been no change. Imagine what it would like to view the first diagram from behind, or upside-down. In either case, what you would see is the same as the first diagram.

Obversion

Take another look at the square of opposition in figure 4.1. Note that the A and the E are straight across from each other, as are the I and the O. The first step in forming the obverse is to first change the sentence into the type that is straight across the square of opposition. That is, if you started with an A-sentence, then make it into and E. The O becomes and I, and so on.

Once you've changed the sentence type, the next step is to change predicate into its complement. The complement of a class C is the class of everything that is not in C. The easiest way to form a complement is to prefix the class with 'non'. For example, the complement of the class of students is the class of non-students.

So, the obverse of all dogs are mammals is no dogs are non-mammals. The obverse of no OBU students are martians is all OBU students are non-martians. Obversion is truth-preserving in all cases.

Contraposition

The last of our three relations is contraposition. To form the contrapositive of a sentence, first form the converse, then exchange both terms for their complements.

The contrapositive of all dogs are mammals is all non-mammals are non-dogs. Contraposition is truth-preserving for A-sentences and O-sentences only.

Original	Converse	Obverse	Contrapositive
All S are P	All P are S	**No S are non-P**	**All non-P are non-S**
No S are P	**No P are S**	**All S are non-P**	No non-P are non-S
Some S are P	**Some P are S**	**Some S are not non-P**	Some non-P are non-S
Some S are not P	Some P are not S	**Some S are non-P**	**Some non-P are not non-S**

Here's a table to help keep this straight (operations that are truth-preserving are in bold type):

Example

Look at the following argument:

1. All Catholics are non-Protestants.
2. All Lutherans are Protestants.
3. No Catholics are Lutherans.

Note that this argument has four terms:

1. Catholics
2. Non-Protestants
3. Lutherans
4. Protestants

To evaluate the argument, we will first have to either change "non-Protestants" to "Protestants" in the first premise, or "Protestants" to "non-Protestants" in the second premise and conclusion. To minimize errors, we should probably try the option requiring the fewest changes. The only two truth-preserving operations on A-sentences are obversion and contraposition. The contrapositive of "All Catholics are non-Protestants" is "All non-non-Protestants are non-Catholics." The double-non will cancel out, which will fix our original problem, but it will leave us with a new term, "non-Catholic." So, let's try the obverse. The obverse of "All Catholics are non-Protestants" is "No Catholics are Protestants." So, using that for our first premise, the argument becomes:

1. No Catholics are Protestants.
2. All Lutherans are Protestants.

3. No Catholics are Lutherans.

Now, we can check for validity — I'll leave that for you.

Chapter 3

Sentential Logic: Syntax

Sentential logic (*SL*) is a system of logic that treats statements, or propositions, as fundamental units.[1] *SL* is not just a system of logic, however, it is also a language. There are two very general types of languages, natural and formal. Natural languages are the languages that we ordinarily use to communicate with each other. Examples of natural languages include English, French, German, Mandarin, etc. Computer programming languages like Python and Java, on the other hand, are formal languages.

There are certain features that every language, natural or formal, must have. Those features govern how expressions are formed in the language, and determine what those expressions mean. Let's see how those features work in the English language. It begins with a character set, which includes the twenty-six characters of the Latin alphabet and various punctuation marks. The characters are put together to form words; the meaning of a word is determined by its definition. These words are combined in various ways to form sentences. Not just any string of English words can be a meaningful sentence in English. There are certain rules that must be followed—articles must precede nouns, for instance. The characters and the formation rules are the syntax of the language, the rules that determine meaning are the semantics. *SL* is a language, and, like any other language, it has a syntax and

[1] Informally, we use 'proposition' and 'statement' interchangeably. Strictly speaking, the proposition is the content, or meaning, that the statement expresses. When different sentences in different languages mean the same thing, it is because they express the same proposition. Sentential logic is also called 'propositional logic.'

semantics. This chapter will focus on the syntax; the semantics of *SL* will be the subject of the next chapter.

Natural languages have many advantages over formal languages, the greatest of which is their expressive power. This expressive power, however, can also be a disadvantage. It is possible to express more than one thought with exactly the same sentence using a natural language. For example, consider the following English sentence:

> "I don't beat my dog."

Now, imagine that same sentence uttered several times, stressing a different word each time. "I don't beat my *dog*" means something very different from "I don't beat *my* dog." This is an example of ambiguity, a word or an expression having more than one meaning. Another feature of natural languages is vagueness; an expression is vague when it doesn't have a precise meaning. Think of vague expressions as those having fuzzy boundaries that make it impossible to draw a precise line between the times that the expression is true of something and the times when it is not. Examples of vague terms are rich, bald, and young. Formal languages eliminate, as much as possible, ambiguity and vagueness. Every term in a formal language should have exactly one meaning and that meaning should be precise.

Every language has ways of using sentences in the language to create other sentences in the language. This is done by applying sentence formation operators or connectives. One way to do this is by joining two sentences with terms like 'and', 'or', 'but', 'however', or 'unless'. So, we can make the sentence

> All of Plato's know works are dialogues, but none of Aristotle's known works are dialogues.

by putting 'but' between 'All of Plato's know works are dialogues' and 'None of Aristotle's known works are dialogues.'

Although we call these sentence-forming operators "connectives," it is important to understand that some of them don't actually connect different sentences. Some of them apply to just one sentence. Examples of these are 'it is not the case that', and 'it is possible that'.

All of the connectives in *SL* will be truth-functional connectives. A connective is truth-functional if the truth value of the resulting sentence is determined completely by the truth values of the connected sentences and the

definition of the connective. So, if we know the truth values of the connected sentences and the operator used to connect them, we will always be able to determine the truth value of the resulting sentence.

Formal languages are used for many things, but an important use in logic is analyzing the logical relationships between sentences and sets of sentences in natural languages, in our case, English. Doing this with *SL* will require translating sentences from English into *SL* and vice versa. Later, I'll state the syntax of *SL* in a very precise way. For now, let me introduce the language by discussing how English is translated into *SL*.

3.1 *SL* Translations

Upper-case letters of the Latin alphabet are used to symbolize simple, or atomic, English sentences. Remember that a simple sentence is one that contains no other sentence as a component. So, 'Socrates is a philosopher' is a simple sentence, but 'Socrates is a philosopher, but Cicero is an orator' is not, since it contains two sentences as components.

Any letter can be used to symbolize a sentence, but we should pick one that will help us to remember the English sentence that is translated. For example, the English sentence

> Socrates is a philosopher

could be symbolized with

> S

Of course, we only have twenty-six upper-case letters, but we need a potentially infinite supply. So, we'll allow letters with positive integer subscripts to be used as sentences in *SL*. So, these are all simple sentences in *SL*:

> A, B, C, P, Q, Z, D_1, F_{412}

One could use a sentence letter to symbolize a complex, or molecular, sentence of English, but it's not a good idea. By doing so, we would be hiding some of the logical structure of the sentence, run the risk of translating a valid argument into an invalid argument in *SL*. So, be sure to only use letters for the simple sentences.

Conjunctions

Our first kind of complex sentence is the conjunction. Conjunctions are sentences formed by combining two sentences with a conjunction connective. The sentences that are joined are called 'conjuncts'. Conjunctions are true whenever both of the conjuncts are true. Although the most common conjunction operator in English is 'and', any operator that implies that both of the connected sentences are true is a conjunction operator. The list of English conjunction operators includes

and

but

also

however

yet

still

moreover

although

nevertheless

both

The two most common ways of symbolizing a conjunction operator are '&' (ampersand), '·' (dot), and '∧' (wedge). We'll use '&'.

Now, we can translate our sentence

Socrates is a philosopher, but Cicero is an orator

as

S & C

and

Plato is from Athens and Aristotle is from Macedonia

can be translated as

A & M

Most of the time, these translations are simple. Unfortunately, the complexity of natural languages can sometimes result in some trick translation issues. There are times when the conjunction operator doesn't appear to be joining two sentences. Here is an example:

Plato and Aristotle are philosophers.

The best thing to do in these cases is to paraphrase the English sentence into a sentence that joins two statements with the conjunction operator. If the resulting paraphrase is true under exactly the same conditions as the original, then they are equivalent. If so, then simply translate the paraphrase into *SL*.

The paraphrase of the preceding sentence would be

Plato is a philosopher and Aristotle is a philosopher.

and the *SL* translation is

P & A

On the other hand, consider this sentence:

Abbot and Costello made a good team.

Paraphrasing this results in

Abbot made a good team and Costello made a good team.

This can't be right though. Neither one, individually, made a team at all. So, this sentence, even though it contains the word 'and', is a simple sentence.

The list of English words that gets translated as a conjunction shows that the *SL* translations often fail to capture the full sense of the original English sentences. When I say, "Although John studied hard for his logic exam, he failed", I don't just mean to say that John studied hard *and* he failed, which is how we would paraphrase it before symbolizing the sentence in *SL*. Instead, I mean to say that *in spite* of his studying, he failed. Some nuance is lost in the translation. That nuance is not important however, for our purposes. The conjunction captures everything that is important for logical analysis.

Disjunctions

Another type of complex sentence is the disjunction, commonly expressed in English with 'or', as in this sentence:

> Either the Democrat will win the election or the Republican will win.

Disjuncts are true whenever at least one of the connected sentences, called disjuncts, are true. They are only false when both disjuncts are false. The symbol '$\vee$' (wedge) is used for disjunctions in *SL*. So, the sentence above is translated

> D $\vee$ R

As was the case with the conjunction, disjunctions in English are not always two complete sentences joined with an 'or'. The example above is more likely to be stated, "Either the Democrat or the Republican will win." Sometimes, disjunctions in English don't use 'or' at all. Here's an example:

> Alice and Bob are running a race tomorrow. At least one of them will finish.

That second sentence can be paraphrased

> Alice will finish the race, or Bob will finish the race.

It can then be symbolized

> A $\vee$ B

A very important thing to remember when translating disjunctions into *SL*, is that the English word 'unless' express a disjunction. The sentence,

> Charlie will fail logic unless he drops the course before the deadline

is equivalent to

> Either Charlie will fail logic or he drops the course before the deadline

and symbolized as

$$F \vee D$$

One thing that is tricky about disjunctions is distinguishing inclusive disjunctions from exclusive disjunctions. An inclusive disjunction is true when *at least one* of the disjuncts is true. An exclusive disjunction is true when one, *and only one*, of the disjuncts is true. The *SL* symbol '$\vee$' is always an inclusive disjunction. In English, however, the word 'or' is often used to express exclusive disjunctions. For example, you might see this in a menu:

Your meal comes with either baked potato or french fries.

The probably don't mean "Your meal comes with either baked potato, french fries, or both." So, this sentence should *not* be translated as

$$P \vee F$$

Later, we will see how to translate exclusive disjunctions in *SL*.

Negations

The English phrase 'it is not the case that' generates a kind of complex sentence called a negation. Negations are true if the negated sentences are false. For example,

It is not the case that Aristotle was from Athens.

This sentence is true just in case it is false that Aristotle was from Athens, and it would be true if it were the case that Aristotle was not from Athens.

The symbol for negations is '$\neg$'. A sentence is negated by placing the negation operator in front of the sentence. This is the one logical connective that doesn't connect two sentences. So, the negation is known as a "unary connective", and the others are called "binary connectives." The phrase 'it is not the case at' is convenient to use when paraphrasing negations because it precedes negated sentences in the same way that the negation symbol in *SL* does. Most of the time, though, 'not' will be somewhere in the English sentence, like this:

Aristotle was not from Athens.

This is naturally equivalent to 'It is not the case that Aristotle was from Athens' and is translated in *SL* as

¬A

Other ways that negations are expressed in English besides 'it is not the case that' and 'not' are prefixes like 'non', 'in', and 'un'. These, however, require some care. The sentence

John is unmarried

is equivalent to

It is not the case that John is married

and is symbolized as '¬J'. On the other hand,

Some students are unmarried

is *not* equivalent to

It is not the case that some students are married.

Instead, it is equivalent to

It is not the case that *all* students are married.

Combining Logical Connectives

Before I introduce the last two connectives in *SL*, let's look at how the connectives are combined to form more complex sentences. Imagine that I want to symbolize this sentence:

Either I will stay home tonight, or I will both go out for dinner and go to the movie theater.

Let's start with this:

H ∨ D & M

There's a problem, though. Remember that a goal of formal languages is to eliminate ambiguity, and this sentence is definitely ambiguous. Is it a disjunction that has a conjunction as one of its disjuncts, or is a conjunction with a disjunction as one of its conjuncts? It it's a disjunction, then my staying home would be enough to make it true. If it's a conjunction, then it's being true requires that I go to the movies. In *SL*, the ambiguity is cleared up with parentheses. If we decide that the English sentence is a conjunction, then we'll symbolize it like this:

$$(H \vee D) \,\&\, M$$

And if it is a disjunction, we'll symbolize it like this:

$$H \vee (D \,\&\, M)$$

In *SL*, and all of the other systems that we will encounter, every sentence has a main connective. The main connective of a sentence determines what kind of sentence it is, and will always be outside of the parentheses—in this case, should we make the main connective the '$\vee$' or the '&'?

There are two things to look for when trying to determine the main logical operator in an English sentence:

1. Puncation, such as commas and semicolons.
2. Words like 'either... or', 'both... and', and 'if... then'.

The first strategy is to group things together that are on the same side of a punctuation mark. Since 'I will go out for dinner' and 'I will go to the movie theater' are both on the right side of the comma, we should group them together with parentheses, like this:

$$H \vee (D \,\&\, M)$$

The second strategy is that anything that is grouped together with parentheses should read naturally as a complete English sentence. If we translated the sentence as

$$(H \vee D) \,\&\, M$$

then 'Either I will stay home tonight, or I will both go out for dinner' should make sense as a stand-alone sentence, but it doesn't because of the 'both' that is missing an 'and'. 'I will both go out for dinner and go to the movie theater' does make sense, though. So this strategy also requires us to translate the sentence as

$$H \vee (D \ \& \ M)$$

These two strategies should be enough to help us correctly translate any well-formed, well-punctuated English sentence. Unfortunately, natural languages are often simply ambiguous, and, in those cases, you must simply do your best. Here are some more examples.

> It's not the case that both Plato and Aristotle are from Athens.

First, we'll paraphrase the sentence to make the connected sentences clear:

> It's not the case that both Plato is from Athens and Aristotle is from Athens.

Since we need to keep the 'both... and' together, we should symbolize this as

$$\neg(P \ \& \ A)$$

Since the only connective that is outside the parentheses is the '$\neg$', it is the main operator and the sentence is a negation.

> Plato is from Athens, but Aristotle is not.

The 'but' is a conjunction, so paraphrasing this results in

> Plato is from Athens, and it is not the case that Aristotle is from Athens

and is symbolized like this:

$$P \ \& \ \neg A$$

Parentheses are not needed here, since there's no ambiguity. The '¬' modifies what it immediate precedes, which in this case is the simple sentence 'A'. The '&' joins the sentence is immediately before it with the one that is immediately after it, which in this case are 'P' and '¬A'. Intuitively, the main connective is the '&', and the sentence is a conjunction.

Here is a rule for determining the main connective of a sentence in symbolic logic: The main connective is the one outside the parentheses, unless there is more than one connective outside the parentheses, then the main connective is not a negation. Just keep in mind that the only time that there can be more than one connective outside the parentheses is when negations are involved, and in that case, the main connective will never be a negation. The main connective can be a negation only when it is the only one outside the parentheses.

Here's one a bit more complex:

> Both either Plato was the greatest metaphysician or Aristotle was the greatest logician, and either Plato was the greatest epistemologist or Aristotle was the greatest moral philosopher.

I don't think there's any need for paraphrasing here. The comma tells us that the 'and' is the main connective, and it joins two disjunctions. So, the translation is

> $(M \vee L) \,\&\, (E \vee P)$

Now, here's something that can be a little trick:

> Neither Aristotle nor Epictetus were from Athens.

This can be paraphased as Both it is not the case that Aristotle is from Athens and it is not the case that Epictetus is from Athens. So, the translation would be

> $\neg A \,\&\, \neg E$

An equally acceptable paraphase is this:

> It is not the case that either Aristotle is from Athens or Epictetus is from Athens.

This would naturally be translated as

$$\neg(A \vee E)$$

It should be easy to see that these are equivalent. The first says that both of the sentences represented by 'A' and 'E' are false. The second says that it is not the case that either one is true. We'll prove that they are equivalent in the next chapter.

Material Conditionals

A conditional is an 'if... then' sentence. There are many ways to express conditionals in English, including

- if
- if… then
- only if
- whenever
- when
- only when
- implies
- provided that
- means
- entails
- is a sufficient condition for
- is a necessary condition for
- given that
- on the condition that
- in case

A conditional claims that something is true, if something else is also. Examples of conditionals are

> "If Sarah makes an A on the final, then she will get an A for the course."
>
> "Your car will last many years, provided you perform the required maintenance."
>
> "You can light that match only if it is not wet."

The symbol that we will use in *SL* for conditionals is called a horseshoe, '$\supset$'. Another commonly used symbol is the right arrow, '$\rightarrow$'. We can translate the above examples like this:

> $\mathsf{F} \supset \mathsf{C}$
>
> $\mathsf{M} \supset \mathsf{L}$
>
> $\mathsf{L} \supset \neg\mathsf{W}$

One big difference between conditionals and other sentences, like conjunctions and disjunctions, is that order matters. Notice that there is no logical difference between the following two sentences:

> Albany is the capital of New York and Austin is the capital of Texas.
>
> Austin is the capital of Texas and Albany is the capital of New York.

They essentially assert exactly the same thing, that both of those conjuncts are true. So, changing order of the conjuncts or disjuncts does not change the meaning of the sentence, and if meaning doesn't change, then truth value doesn't change either. That's not true of conditionals. Note the difference between these two sentences:

> If you drew a diamond, then you drew a red card.
>
> If you drew a red card, then you drew a diamond.

The first sentence *must* be true. if you drew a diamond, then that guarantees that it's a red card. The second sentence, though, could be false. Your drawing a red card doesn't guarantee that you drew a diamond, you could have drawn a heart instead. So, we need to be able to specify which sentence goes before the arrow and which sentence goes after. The sentence before the

arrow is called the antecedent, and the sentence after the arrow is called the consequent.

Look at those three examples again:

> If Sarah makes an A on the final, then she will get an A for the course.
>
> Your car will last many years, provided you perform the required maintenance.
>
> You can light that match only if it is not wet.

The antecedent for the first sentence is 'Sarah makes an A on the final'. The consequent is 'She will get an A for the course'. Note that the 'if' and the 'then' are not parts of the antecedent and consequent.

In the second sentence, the antecedent is 'You perform the required maintenance'. The consequent is 'Your car will last many years'. This tells us that the antecedent won't always come first in the English sentence.

The third sentence is tricky. The antecedent is 'You can light that match'. Why? The explanation involves understanding the difference between necessary and sufficient conditions.

A sufficient condition is something that is enough to guarantee the truth of something else. For example, getting a 95 on an exam is sufficient for making an A on the exam, assuming the usual grading scale and that exam is worth 100 points. A necessary condition is something that must be true in order for something else to be true. Making a 95 on an exam is not necessary for making an A—a 94 would have still been an A. Taking the exam is necessary for making an A, though. You can't make an A if you don't take the exam, or, in other words, you can make an A only if you enroll in the course.

Here are some important rules to keep in mind:

> 'If' introduces an antecedent, but 'only if' introduces a consequent.
>
> If A is a sufficient condition for B, then $A \rightarrow B$.
>
> If A is a necessary condition for B, then $B \rightarrow A$.

All of the connectives in *SL* are truth-functional. That is, the truth value of the complex sentence is a function solely of the simple sentences and the

logical connectives contained in the sentence. The truth-functional conditional is called the material conditional, and it has some puzzling consequences. Consider this sentence again:

> If Sarah makes an A on the final, then she will get an A for the course.

Intuitively, this sentence is false when Sarah makes an A on the final, but does *not* get an A for the course, and true when Sarah makes an A on the final and also makes an A for the course. What if Sarah does not get an A on the final? In that case, the material conditional is true no matter whether she gets an A for the course or not. So far, this doesn't seem to be a problem. Let's change the consequent of the conditional, however.

> If Sarah makes an A on the final, then she will get an F for the course.

If this is a material conditional, it will be false only when the antecedent is true and the consequent is false, true whenever either the antecedent is false or the consequent is true. In any case in which Sarah does not make an A on the final, the sentence is true. Now, imagine that the final is worth 100 of the 1,000 points possible in the course. Also, imagine that Sarah has made a perfect score on everything in the course before the final. Sarah then takes the final and makes a score of 85. That lowers her perfect average of 100 to an almost perfect 98.5, which is surely still deserving of an A for the course. Since she didn't make an A on the final, however, the material conditional 'If Sarah makes an A on the final, then she will get an F for the course' is still true. This is called the paradox of the material conditional, something that we will revisit in later chapters.

Most of the time, when we assert a conditional in English, we don't intend it to be a material conditional. Unfortunately, incorporating other kinds of conditionals would complicate our logic. The good news, though, is that when it comes to analyzing logical relations between sentences, we can usually just pretend that every conditional is a material conditional. For now, that just what we'll do.

Material Biconditionals

Our final connective is expressed by the English phrases 'if and only if', 'when and only when', and 'just in case'. These are used to express conditions that are both necessary and sufficient, such as making at least a 90 and getting an A. So this sentence,

> Students make an A if and only if they average at least 90% on all work

can be paraphrased as

> If students make an A, then they average at least 90% on all work; and students average at least 90% on all work only if they make an A

and symbolized as a conjunction of two conditionals like this:

> $(A \supset W) \,\&\, (W \supset A)$

Although we could certainly translate biconditionals this way, they occur often enough that it is useful to have a symbol dedicated for that purpose. It's common to use either '$\equiv$' (triple bar) or '$\leftrightarrow$' (double arrow). We'll use the triple bar, and translate the previous example like this:

> $A \equiv W$

Material biconditionals are true whenever the two sentences joined by the biconditional operator have the same truth value, and false whenever they have different truth values. For biconditionals, as for conjunctions and disjunctions, the order of the connected sentences doesn't matter.

Complex Symbolizations

Now that we've introduced the five truth-functional connectives, let's practice some more complex translations into *SL*. We'll use these simple sentences:

F: The Federal Reserve will raise interest rates.
C: Congress will increase spending.
I: The inflation rate goes up.
U: There is increased unemployment.
P: Prices of consumer goods increase.

Here's the first one:

> If the inflation rate goes up, then either the Federal Reserve will raise interest rates or Congress will increase spending.

We start by putting in the sentence letters and the connective symbols.

$$I \supset F \vee C$$

Then, we put in the parentheses, making sure to keep the sentences on the same side of the comma together.

$$I \supset (F \vee C)$$

The second example is

> Both the Federal Reserve will raise interest rates and Congress will increase spending if and only if either prices of consumer goods increase or there is increased unemployment.

Filling in the sentence letters and connectives gives us this:

$$F \,\&\, C \equiv P \vee U$$

We don't have any punctuation to help us here, but we do have a 'both... and' and an 'either... or' that will guide our punctuation, resulting in this:

$$(F \,\&\, C) \equiv (P \vee U)$$

One more:

> Congress will not increase spending, unless neither prices of consumer goods increase nor there is increased unemployment; but if the inflation rate goes up, then the Federal Reserve will raise interest rates.

The semicolon tells us that the 'but' is the main connective, and the second conjunct is pretty simple:

$$I \supset F)$$

The first conjunct is a disjunction. The first disjunct is

$$\neg C$$

The second disjunct, is a 'neither, nor' sentence, which is best translated as a negation of a disjunction. Putting the two disjuncts together, we have

$$\neg C \vee \neg(P \vee U)$$

So, we just need to put those two conjuncts together with an '&', being sure to put brackets around the first to avoid any ambiguity:

$$[\neg C \vee \neg(P \vee U)] \ \& \ (I \supset F)$$

3.2 Rules of Syntax

Use and Mention

We use words to talk about other things. For instance, in the sentence, 'The 46th President of the United States is from Delaware', the phrase 'The 46th President of the United States' refers to Joe Biden. In the sentence 'Red is the color at the long wavelength end of the visible spectrum', the word 'red' refers, unsurprisingly to the color red. When we use the word 'red', we refer to the color red. When we use the name 'Joe Biden', we refer to President Biden.

Sometimes, though, we want to talk about the words themselves. What should we do then? Just as we use names of persons to talk about the persons, we need names for words and expressions to be able to talk about those words and expressions. We form the names of expressions by placing those expressions in single quotes. Without the quotes, we are using the expresion. When the expression is enclosed in quotes, it is being mentioned, not used. For example,

'Red' has three letters.

In this case, the word ‘red’ is being mentioned, not used. In the sentence,

> Red is often used to signify passion.

the word ‘red’ is being used. The use-mention fallacy is committed whenever a word is used when it should have been mentioned, or vice versa. An example is the sentence,

> Red has three letters.

This sentence asserts that the color red has three letters, but colors aren’t the kind of things that can have letters. Instead it should be

> ‘Red’ has three letters.

The first is nonsense, the second is a perfectly meaningful, and true, English sentence. When we use a word, we are referring to the thing that the word stands for, when we mention a word, we are referring to the word itself.

Object Language and Metalanguage

The ability to form names of expressions by enclosing them in single quotes enables us to talk about the expressions of on language while using another language. Here is an example:

> ‘Ich’ is the German first-person singular pronoun.

This is a sentence in English, so all of its components must be in English. By enclosing the German word in single quotes, we have formed the English name for that German expression. So, we are using English to talk about an expression in German. The language that is being discussed is called the *object language*. The language that is being used to talk about the object language is called the *metalanguage*. In this case, the object language is German, and the metalanguage is English. As we formulate the rules for *SL*, the metalanguage will be English, but the object language will be *SL*.

Metalinguistic Variables

To refer to specific expressions of *SL*, we can simply enclose them in single quotes, as we have been doing. The sentence

> 'A & B' is a conjunction.

claims that a particular expression of *SL* is a conjunction. It is an English sentence using the English name for an expression of *SL*. To formulate the rules of *SL*, however, we will need to be able to discuss general types of expressions of *SL*, not just particular expressions of *SL*. We do this using *metalinguistic variables*, or *metavariables*, for short.

A metalinguistic variable is an expression in the metalanguage that stands for *any* expression in the object language. We will use lowercase Greek letters, such as 'α', 'β', and 'γ', for metavariables. The preceding sentence encloses the metavariables in single quotes because it is talking about the metavariables. When the metavariables are being used, they won't need to be enclosed in quotes since they are expressions in the metalanguage.

The Syntax of *SL*

We now have everything we need to carefully define the rules of syntax of *SL*. These rules will tell us what counts as a proper expression of *SL*. Not every string of English words will be a proper English sentence. For instance 'ball the red is blue and' is composed of perfectly good English words, but any English speaker will know that the sentence is nonsense because it violates the conventions for making proper English expressions. We need to formulate rules for making proper expressions of *SL* that are rigorous and precise, such that, for any string of symbols of *SL*, will be able to determine if that string forms a proper expression of *SL*. We'll call a proper expression a 'well-formed formula', or 'wff' for short.

Here are our rules of syntax for *SL*:

1. Every uppercase letter of the Latin alphabet, with or without a numerical subscript, is a well-formed formula.
2. If α is a well-formed formula, then ¬α is a well-formed formula.
3. If α and β are well-formed formulas, then so is (α & β).
4. If α and β are well-formed formulas, then so is (α ∨ β).
5. If α and β are well-formed formulas, then so is (α ⊃ β).

6. If α and β are well-formed formulas, then so is $(\alpha \equiv \beta)$.
7. Nothing is well-formed formula unless it can be formed by repeated application of 1–6.

These are *recursive* rules. That is, they specify what counts as the simplest type of well-formed formula in *SL*, then show how increasingly more complex formulas are generated by applying the same rules again and again.

If you have paid close attention to the previous sections, you might have recognized a problem with the rules. Consider the expression formed with the negation symbol and a metalinguistic variable:

$\neg\alpha$

It consists of a metalinguistic variable and a symbol of *SL*. Thus, it's a mixture of the object language and the metalanguage. That means it is a well-formed formula of neither language. We're committing the use-mention fallacy—the '$\neg$' is being used, when instead it should be merely mentioned. So, we need to enclose it in single quotes, but where should the quotes go? It looks like we have two options, but neither are good. The first is to enclose the whole formula, including the metavariable, in quotes:

'$\neg\alpha$'.

If we do that, then, we are no longer using 'α' as a metavariable. Instead, we would be treating it as a symbol in *SL*, but it's not a symbol in *SL*. So, we haven't avoided the use-mention fallacy; we've merely committed another instance of the use-mention fallacy.

Another option is to enclose only the negation symbol in single quotes, like this:

'$\neg$'α

This is better, but it's not clear what it means. A precise way to state rule 2 without committing the use-mention fallacy is this:

> 2*: If α is a well-formed formula, then the formula expressed by concatenating '$\neg$' and α is a well-formed formula.

That's not bad, but fixing rules 3–7 in the same way would be somewhat awkward. The solution is to use something called "quasi-quotation".[2] Quasi-quotation encloses the string formed by the symbols of the object-language and the metavariables in corner-quotes:

$\ulcorner\neg\alpha\urcorner$

Quasi-quotation is just shorthand for a statement about the concatenation of symbols and formulas. So $\ulcorner\neg\alpha\urcorner$ is shorthand for

The formula expressed by concatenating '$\neg$' and α.

To make things simpler, we will adopt this as a convention: whenever we use expressions that consist of both metavariables and formulas of an object language, we will understand the entire expression as being enclosed in corner-quotes. So, if we were to write this,

$(\alpha \mathbin{\&} (\neg\beta \vee \gamma))$

we would understand it as expressing this:

$\ulcorner(\alpha \mathbin{\&} (\neg\beta \vee \gamma))\urcorner$

and when we quote an expression consisting of a mixture of object language symbols and metavariables, we will understand that quotation to be quasi-quotation. So, when we write '$\alpha \mathbin{\&} \beta$', we will understand it as $\ulcorner\alpha \mathbin{\&} \beta\urcorner$.

There are two more conventions regarding the use of parentheses in *SL*. First, notice that the only punctuation defined in the rules of syntax are parentheses; there are no brackets or braces. This is because we need to be able to form indefinitely large expressions of *SL*, which requires nested expressions. If we demanded a different punctuation character at each level, we would quickly run out of punctuation symbols. This can, though, make things difficult to read. Since we will never, in this book, encounter nesting of more than three levels, we will use braces and brackets for levels two and three. So, instead of writing

$\neg((\alpha \supset (\beta \mathbin{\&} \gamma)) \equiv ((\alpha \vee \delta) \vee (\gamma \supset \varepsilon)))$

[2] This is also sometimes called "Quine-quotation", because it was first introduced by W.V.O. Quine in his book *Mathematical Logic*.

we will write

$$\neg\{[\alpha \supset (\beta \,\&\, \gamma)] \equiv [(\alpha \vee \delta) \vee (\gamma \supset \varepsilon)]\}$$

which should be easier to read.

The second convention is that we will drop the outermost set of parentheses that encloses the entire complex formula, since those parentheses aren't needed to avoid ambiguities. So, instead of

$$(\alpha \,\&\, \beta)$$

we'll simply write

$$\alpha \,\&\, \beta$$

and instead of

$$[\alpha \,\&\, (\neg\beta \vee \gamma)]$$

we can write

$$\alpha \,\&\, (\neg\beta \vee \gamma)$$

Now, we'll use the rules to determine if an expression of *SL* is a well-formed formula. We start with the sentence letters used in the expression, then show, by using repeated instances of the rules of syntax, that the expression can be generated. Here's a simple example:

> Show that '$\neg(A \,\&\, B) \supset C$' is a well-formed formula, or sentence, in *SL*.

First, by rule 1, both 'A' and 'B' are wff's. By rule 3, 'A & B' is a wff. By rule 2, '$\neg$(A & B)' is a wff. Finally, by rule 5, '$\neg(A \,\&\, B) \supset C$' is a wff.

Finally, here are some expression that are *not* well-formed formulas of *SL*. See if you can determine what is wrong with each one.

$A \,\&\, B \vee C$

$(P \supset Q)R$

$\neg \equiv Z$

$\neg[(E \,\&\, F) \supset D$

Chapter 4

Sentential Logic: Semantics

The syntax for a language is concerned with the vocabulary and the rules for what counts as a well-formed expression of the language. The semantics is concerned with the meaning and truth conditions of the expressions of the language. The meanings of sentences in *SL* are always truth values, and every sentence has one, and only one, of two truth values, either 1 (true) or 0 (false).

4.1 Truth Tables for Connectives

Sentences of *SL* can be analyzed using truth-tables. Truth-tables are extremely powerful tools—they provide a foolproof test for all those concepts introduced in chapter 1, like validity, consistency, and logical truth. We'll start with the most basic truth-tables, which are the *characteristic truth-tables* for the logical connectives of *SL*.

The simplest is the negation. A negation is true whenever the negated sentence is false, and false whenever the negated sentence is true.

α	$\neg\alpha$
1	0
0	1

Conjunctions are true when and only when both conjuncts are true. So, the characteristic truth table for the conjunction looks like this:

α	β	$\alpha \,\&\, \beta$
1	1	1
1	0	0
0	1	0
0	0	0

Disjunctions are true whenever at least one of the disjuncts are true:

α	β	$\alpha \vee \beta$
1	1	1
1	0	1
0	1	1
0	0	0

A conditional is false only when the antecedent is true and the consequent is false.

α	β	$\alpha \supset \beta$
1	1	1
1	0	0
0	1	1
0	0	1

Finally, biconditionals are true whenever the sentences joined by the biconditional operator have the same truth value, and false when they have different truth values.

α	β	$\alpha \equiv \beta$
1	1	1
1	0	0
0	1	0
0	0	1

4.2 Truth Tables for Complex Sentences

The characteristic truth-tables for the logical connectives show us how to determine the truth value for any complex sentence, given the truth values of its constituent simple sentences. The truth values of those simple sentences are provided by an interpretation, also called a valuation or *model*. A model contains all of the information that is required to determine the truth values of every formula in a language. How complex a model needs to be depends on the language; models in *SL* are very simple. Remember that the logical connectives are truth-functional, which means that the truth-values of the complex sentences formed using those connectives are completely determined by the truth-values of the constituent simple sentences. So, to determine the truth-values of all of the formulas, however complex, of *SL*, we only need to assign truth-values to all of the simple sentences of *SL*.[1]

A model is a function that maps atomic formulas of *SL* to truth-values. Thus, a model determines the truth-values of every sentence in *SL*, not just the simple sentences. A partial model assigns truth-values to only some of the sentences in *SL*. Whenever we use the term 'model', we will almost always mean a partial model, since we aren't generally concerned about the truth values of all of the sentences in *SL*, but only those that are contained in the sentence or argument that we are analyzing.

Here are two ways to express models. The first uses standard function notation, using 'v' for 'valuation':

$$v_1(A) = 1, v_1(B) = 0, v_1(C) = 1$$

This says that valuation v_1 assigns 1 to **A** and **C**, and 0 to **B**. The same information can be expressed in a table:

	A	B	C
v_1	1	0	1

A truth-table for a sentence shows its truth-value for every partial model. Let's begin by constructing a very simple truth table for a sentence containing one simple sentence, 'A ⊃ A'. Begin simply by writing down the sentence.

[1] Models in *SL* are often called "truth-value assignments."

A ⊃ A

Then, to the left of the sentence, write all of the simple sentence letters in alphabetical order, like this:

A A ⊃ A

Now draw a line underneath all of that.

A A ⊃ A

The next thing to do is to determine how many rows are needed. We need one row for each partial model. In this case, since we only have one simple sentence, there are only two models; one in which 'A' is true and another in which it is false.

Now that we know we will need two rows, we draw a vertical line between the complex sentence and the simple sentence letters. The line should extend down for however many rows are needed.

A	A ⊃ A

Now, we list our truth value assignments for the simple sentences. In this case, underneath the 'A' that is to the left of the vertical line, we'll write '1' and '0', like this:

A	A ⊃ A
1	
0	

Next, copy the column under each sentence letter at the left to every occurrence of that sentence letter in the complex sentence. That's simple in this case, since we only have one simple sentence.

A	A	⊃	A
1	1		1
0	0		0

Now, beginning with the lowest level of parentheses, determine the value of the complex sentences. On the top row, we have a conditional with a true antecedent and a true consequent. So, the conditional is true.

A	A	⊃	A
1	1	1	1
0	0		0

On the second row, we have a conditional with a false antecedent and a false consequent, which still results in a true conditional.

A	A	⊃	A
1	1	1	1
0	0	1	0

When everything is filled in, the truth-table is complete.

Now, let's do one containing two simple sentences, '(P ∨ Q) & ¬P'.

First, write the complex sentence, then, to the left, write the simple simple sentence letters found in that complex sentence in alphabetical order.

P Q (P ∨ Q) & ¬ P

Before we draw the lines, we need to determine how many rows are needed. Now, there are four partial models, one in which 'P' and 'Q' are both true, one in which they are both false, one where 'P' is true and 'Q' is false, and one in which 'Q' is true and 'P' is false. The general rule is that, if there are n simple sentences, the truth-table will have 2^n rows. So, draw the lines, leaving room for four rows.

P	Q	(P	∨	Q)	&	¬	P

Now, we need to fill in the partial models. We'll do this systematically so that we will be sure to get every one. The best thing to do is to start with the rightmost simple sentence, in this case, 'Q'. Underneath that, alternate 'T' and 'F' until you reach the bottom of the truth table.

P	Q	(P	∨	Q)	&	¬	P
	1						
	0						
	1						
	0						

Then, under the next sentence to the left, write '1' twice, then '0' twice. If there were another column, it would have '1' four times, followed by '0' four times, and so on. The top row will always be all 1's, and the bottom row will always contain all 0's. The rightmost column will always alternate 1 and 0, and the leftmost column will have all 1's in the top half and 0's in the bottome half. Our table should look lie this:

P	Q	(P	∨	Q)	&	¬	P
1	1						
1	0						
0	1						
0	0						

Now, copy those columns to their respective letters in the complex sentence.

P	Q	(P	∨	Q)	&	¬	P
1	1	1		1			1
1	0	1		0			1
0	1	0		1			0
0	0	0		0			0

Next, we have to determine the truth-values of 'P ∨ Q' and '¬P' for each model. Let's start with '¬P'. It's true whenever 'P' is false and false whenever 'P' is true.

P	Q	(P	∨	Q)	&	¬	P
1	1	1		1		0	1
1	0	1		0		0	1
0	1	0		1		1	0
0	0	0		0		1	0

Now, 'P ∨ Q' is false only when both 'P' and 'Q' are false. So, the column under the disjunction symbol looks like this:

P	Q	(P	∨	Q)	&	¬	P
1	1	1	1	1		0	1
1	0	1	1	0		0	1
0	1	0	1	1		1	0
0	0	0	0	0		1	0

Now, we're ready to do the column under the main connective of the sentence. This sentence is a conjunction of two sentences, one is a disjunction and the other is a negation. So, we're only concerned about the columns under the "∨" and the '¬'. Remember, conjunctions are true only when both conjuncts are true. On the first two rows, the second conjunct is false, so

P	Q	(P	∨	Q)	&	¬	P
1	1	1	1	1	0	0	1
1	0	1	1	0	0	0	1
0	1	0	1	1		1	0
0	0	0	0	0		1	0

Both conjuncts are true on the third row:

P	Q	(P	∨	Q)	&	¬	P
1	1	1	1	1	0	0	1
1	0	1	1	0	0	0	1
0	1	0	1	1	1	1	0
0	0	0	0	0		1	0

Finally, the first conjunct is false on the fourth row. So, the completed truth table looks like this:

P	Q	(P	∨	Q)	&	¬	P
1	1	1	1	1	0	0	1
1	0	1	1	0	0	0	1
0	1	0	1	1	1	1	0
0	0	0	0	0	0	1	0

Now, let's try one with three simple sentences. So, we'll need eight rows.

F	G	H	¬	[(H	&	F)	≡	G]

Filling in the models gets us this:

F	G	H	¬	[(H	&	F)	≡	G]
1	1	1						
1	1	0						
1	0	1						
1	0	0						
0	1	1						
0	1	0						
0	0	1						
0	0	0						

Copying those columns over produces this:

F	G	H	¬	[(H	&	F)	≡	G]
1	1	1		1		1		1
1	1	0		0		1		1
1	0	1		1		1		0
1	0	0		0		1		0
0	1	1		1		0		1
0	1	0		0		0		1
0	0	1		1		0		0
0	0	0		0		0		0

Now, we start with the conjunction. It's true on any TVA in which both 'H' and 'F' are true.

F	G	H	¬	[(H	&	F)	≡	G]
1	1	1		1	1	1		1
1	1	0		0	0	1		1
1	0	1		1	1	1		0
1	0	0		0	0	1		0
0	1	1		1	0	0		1
0	1	0		0	0	0		1
0	0	1		1	0	0		0
0	0	0		0	0	0		0

Now, biconditionals are true whenever the two connected sentences have the same truth value. So, we just need to compare the truth value of the conjunction that we just completed with the truth-value of 'G'

F	G	H	¬	[(H	&	F)	≡	G]
1	1	1		1	1	1	1	1
1	1	0		0	0	1	0	1
1	0	1		1	1	1	0	0
1	0	0		0	0	1	1	1
0	1	1		1	0	0	0	1
0	1	0		0	0	0	0	1
0	0	1		1	0	0	1	0
0	0	0		0	0	0	1	0

Finally, the negation will flip the truth-value of the biconditional.

F	G	H	¬	[(H	&	F)	≡	G]
1	1	1	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	0	0	0	1	1	1
0	1	1	1	1	0	0	0	1
0	1	0	1	0	0	0	0	1
0	0	1	0	1	0	0	1	0
0	0	0	0	0	0	0	1	0

4.3 Truth Table Shortcuts

If you understand the characteristic truth-tables for the logical connectives, you can complete the most complex truth-table. There is one major problem with truth-tables, however, they get very big very quickly. With every additional simple sentence letter, the truth-table doubles in size. A truth-table for a sentence with seven simple sentences will have 128 rows! Sometimes, though, we can take some shortcuts. Let's go back and look at some of our earlier examples.

First, we don't need to always copy the columns for the simple sentences. Sometimes it helps to do so for clarity's sake. In this case, there's really no need.

P	Q	(P	∨	Q)	&	¬P
1	1					
1	0					
0	1					
0	0					

First, we know that the column for '¬P' will just be the opposite of 'P'. It's just as easy to switch the truth-values as copy the column in the first place. So, we'll just write this:

P	Q	(P	∨	Q)	&	¬P
1	1					0
1	0					0
0	1					1
0	0					1

Always keep in mind that what we're really interested in is the column under the main connective, which, in this case, is the conjunction symbol. Conjunctions are true only when both conjuncts are true, and since '¬P' is false on the first two lines, the conjunction will be false there also:

P	Q	(P	∨	Q)	&	¬P
1	1				0	0
1	0				0	0
0	1					1
0	0					1

Now, all that we need to do is find the truth-value of the 'P ∨ Q' on the last two lines. We can easily see that it will be false only on the last line, where both disjuncts are false. So, we now have this:

P	Q	(P	∨	Q)	&	¬P
1	1				0	0
1	0				0	0
0	1		1			1
0	0		0			1

Now, we can complete the column for the main connective on the last two rows. Both conjuncts are true on the third row, but the first is false on the fourth row, resulting in this:

P	Q	(P	∨	Q)	&	¬P
1	1				0	0
1	0				0	0
0	1		1		1	1
0	0		0		0	1

Sometimes, we're only asked to determine the truth-value for a sentence on a single model. That is just asking what the truth-value of the sentence would be on one particular row of a truth-table. Here's an example:

> Determine the truth value of '¬(Q & R) ⊃ (S ∨ ¬T)' on this model:
>
> Q = 0
> R = 1
> S = 0
> T = 0

So, we build a single row truth-table.

Q	R	S	T	¬	(Q	&	R)	⊃	(S	∨	¬T)
0	1	0	0								

Since **Q** is false, so is **Q & R**.

Q	R	S	T	¬	(Q	&	R)	⊃	(S	∨	¬T)
0	1	0	0			0					

The negation, then, will be true.

Q	R	S	T	¬	(Q	&	R)	⊃	(S	∨	¬T)
0	1	0	0	1		0					

¬T is true, so the disjunction is also true.

Q	R	S	T	¬	(Q	&	R)	⊃	(S	∨	¬T)
0	1	0	0	1		0				1	1

So, now we have a conditional with a false antecedent and a true consequent, making the conditional true.

Q	R	S	T	¬	(Q	&	R)	⊃	(S	∨	¬T)
0	1	0	0	1		0		1		1	1

Note that we needed to determine the truth-value of only one of the antecedent or the consequent. Either one would have been enough to guarantee that the conditional was true.

4.4 Logical Truth, Falsehood, and Indeterminacy

Now that we know how to complete truth tables, let's see what can be done with them. Truth-tables provide a systematic means for determining those things that were discussed in chapter 1—validity, consistency, logical truth and falsehood, and logical equivalency.

To check the logical status of a sentence, simply complete the truth table the same way that we have done so far. After completing it, if there are only 1's under the main connective of the sentence, then the sentence is a logical truth. If there are only 0's, then the statement is a logical falsehood. If there is a mixture of 1's and 0's, then it is logically indeterminate. Let's work a few examples.

P ⊃ (Q ⊃ P)

Since there are two simple sentences, the truth-table will have four rows.

P	Q	P	⊃	(Q	⊃	P)
1	1					
1	0					
0	1					
0	0					

Whenever the antecedent of a conditional is false, the conditional must be true. So, the bottom two rows must both be true.

P	Q	P	⊃	(Q	⊃	P)
1	1					
1	0					
0	1	0	1			
0	0	0	1			

Now, the consequent is itself a conditional, 'Q ⊃ P'. So, when **P** is true, that conditional is true.

P	Q	P	⊃	(Q	⊃	P)
1	1			1		
1	0			1		
0	1	0	1			
0	0	0	1			

Since a true consequent is enough to make a conditional true, the sentence is also true on the top two rows.

P	Q	P	⊃	(Q	⊃	P)
1	1		1	1		
1	0		1	1		
0	1	0	1			
0	0	0	1			

So, since this sentence is true on every model, it is a logical truth, or tautology.

$\neg(A \supset B) \,\&\, (B \vee \neg A)$

This is also a four-row truth table.

A	B	¬	(A	⊃	B)	&	(B	∨	¬A)
1	1								
1	0								
0	1								
0	0								

The left conjunct contains a simple conditional, so let's start there.

a	b	¬	(a	⊃	b)	&	(b	∨	¬a)
1	1			1					
1	0			0					
0	1			1					
0	0			1					

Now, that conditional is negated to form the first conjunct.

A	B	¬	(A	⊃	B)	&	(B	∨	¬A)
1	1	0		1					
1	0	1		0					
0	1	0		1					
0	0	0		1					

That's enough to make the conjunction false on rows 1, 3, and 4.

A	B	¬	(A	⊃	B)	&	(B	∨	¬A)
1	1	0		1		0			
1	0	1		0					
0	1	0		1		0			
0	0	0		1		0			

So, we really just need to find the truth-value of the second conjunct on row 2. Since, on that row, **B** is false and **A** is true, the disjunction B $\vee$ $\neg$A is false, which makes the conjunction false on the second row also.

A	B	$\neg$	(A	$\supset$	B)	&	(B	$\vee$	$\neg$A)
1	1	0		1		0			
1	0	1		0		0		0	
0	1	0		1		0			
0	0	0		1		0			

So, the truth-table show that the sentence is false on all models, and is therefore a logical falsehood or contradiction.

One more example:

(X $\equiv$ Y) $\vee$ (W $\vee$ Z)

This truth table, containing four simple sentences, will have sixteen rows. That makes it slightly painful to complete. So, let's think about how we can make things easier. Note that before we can conclude that a sentence is a tautology or contradition, we have to complete all of the rows. On the other hand, to conclude that a sentence is logically indeterminate or contingent, we only need one row that is 1, and one row that is 0. Thinking shrewdly about which rows should have which values may make our job much easier.

W	X	Y	Z	(X	≡	Y)	∨	(W	∨	Z)
1	1	1	1							
1	1	1	0							
1	1	0	1							
1	1	0	0							
1	0	1	1							
1	0	1	0							
1	0	0	1							
1	0	0	0							
0	1	1	1							
0	1	1	0							
0	1	0	1							
0	1	0	0							
0	0	1	1							
0	0	1	0							
0	0	0	1							
0	0	0	0							

X and **Y** are both true on the first row, so the biconditional is true. That's enough to guarantee that the disjunction is also true.

W	X	Y	Z	(X	≡	Y)	∨	(W	∨	Z)
1	1	1	1	1	1	1	1			
1	1	1	0							
1	1	0	1							
1	1	0	0							
1	0	1	1							
1	0	1	0							
1	0	0	1							
1	0	0	0							
0	1	1	1							
0	1	1	0							
0	1	0	1							
0	1	0	0							
0	0	1	1							
0	0	1	0							
0	0	0	1							
0	0	0	0							

Now, we just need a row that is false. Since our sentence is a disjunction, we need a row on which both disjuncts are false. Let's start with the second disjunct. For this to be false, both **W** and **Z** must be false. That is the case on rows, 10, 12, 14, and 16. Now, we also need it to be a row on which the first disjunct is false, which will be row on which **X** and **Y** have different truth values. That's true on rows 12 and 14. So, either row 12 or 14 will provide what we need, a row on which the sentence is false.

W	X	Y	Z	(X	≡	Y)	∨	(W	∨	Z)
1	1	1	1	1	1	1	1			
1	1	1	0							
1	1	0	1							
1	1	0	0							
1	0	1	1							
1	0	1	0							
1	0	0	1							
1	0	0	0							
0	1	1	1							
0	1	1	0							
0	1	0	1							
0	1	0	0							
0	0	1	1							
0	0	1	0	0	0	1	0	0	0	0
0	0	0	1							
0	0	0	0							

We now have at least one row that is true and at least one row that is false, so the sentence is logically indeterminate.

4.5 Logical Equivalence

Two sentences are logically equivalent if and only if they have the same truth-value on every model. To show logical equivalence, put the two sentences on the same truth-table. If they are logically equivalent, the columns under the main connectives of the sentences will be exactly the same.

Two logical truths will always be logically equivalent, as will two logical falsehoods. Logical indeterminacies may or may not be equivalent.

Let's check these two sentences for logical equivalence (We'll separate them with a forward slash for clarity):

$$F \supset G \qquad \neg G \supset \neg F$$

F	G	F	⊃	G	/	¬G	⊃	¬F
1	1		1				1	
1	0		0				0	
0	1		1				1	
0	0		1				1	

See that the two sentences have the same truth-value on every model, and are therefore logically equivalent.

Now, consider these two sentences:

A ∨ B (B ∨ C) ∨ A

A	B	C	A	∨	B	/	(B	∨	C)	∨	A
1	1	1		1				1		1	1
1	1	0		1				1		1	1
1	0	1		1				1		1	1
1	0	0		1				0		1	1
0	1	1		1				1		1	0
0	1	0		1				1		1	0
0	0	1		0				1		1	0
0	0	0		0				0		0	0

Note that it is not until line 7 that the two sentences differ in truth-value. Since there is at least one model on which they have different truth-values, the two sentences are not logically equivalent.

4.6 Logical Consistency

A set of sentences is logically consistent if and only if there is at least one model on which all of the members of the set are true. If there is no model on which all of the members are true, then the set is logically inconsistent.

Any set that contains a logical falsehood is inconsistent. Sets that do not contain logical falsehoods *can* be inconsistent. We can check for logical consistency using one truth-table. If the set is consistent, then there will be at least one row on which all of the sentences are true.

$\{\neg A \,\&\, \neg B, A \vee \neg C\}$

A	B	C	¬A	&	¬B	/	A	∨	¬C
1	1	1	0	0	0		1	1	0
1	1	0	0	0	0		1	1	1
1	0	1	0	0	1		1	1	0
1	0	0	0	0	1		1	1	1
0	1	1	1	0	0		0	0	0
0	1	0	1	0	0		0	1	1
0	0	1	1	1	1		0	0	0
0	0	0	1	1	1		0	1	1

This truth table shows that the set of consistent. Not that it is only on the last row on which both sentences are true. We cannot conclude that a set is inconsistent without examining all of the rows of a truth-table. That doesn't mean that we necessarily have to complete each row completely, though. Let's look at this example again to see the shortcuts we could have taken.

A	B	C	¬A	&	¬B	/	A	∨	¬C
1	1	1							
1	1	0							
1	0	1							
1	0	0							
0	1	1							
0	1	0							
0	0	1							
0	0	0							

Keep in mind that we're trying to find a row on which both sentences are true. The first sentence will be true only when A and B are both false, which is only on the last two rows.

A	B	C	¬A	&	¬B	/	A	∨	¬C
1	1	1							
1	1	0							
1	0	1							
1	0	0							
0	1	1							
0	1	0							
0	0	1		1					
0	0	0		1					

The truth-value of the second sentence won't matter on the first six rows since the first sentence is false on those rows. So, we just need to determine the truth value of the second sentence on the last two rows. It is true whenever A is true or C is false. So, it is false on row 7, but true on row 8.

A	B	C	¬A	&	¬B	/	A	∨	¬C
1	1	1							
1	1	0							
1	0	1							
1	0	0							
0	1	1							
0	1	0							
0	0	1		1				0	
0	0	0		1				1	

So, it's only on that last row that both of the sentences are true, but that is enough to make the set consistent.

Here's another example:

$\{P \vee \neg P, P \supset Q, P \,\&\, \neg Q\}$

We know that $P \vee \neg P$ is true on all rows.

P	Q	P	∨	¬P	/	P	⊃	Q	/	P	&	¬Q
1	1		1									
1	0		1									
0	1		1									
0	0		1									

Unfortunately, we can't draw any conclusions from that alone. The second sentence is true on all rows except for the second.

P	Q	P	∨	¬P	/	P	⊃	Q	/	P	&	¬Q
1	1		1				1					
1	0		1				0					
0	1		1				1					
0	0		1				1					

The third sentence is true only when P is true and Q is false, which is just the second row.

P	Q	P	∨	¬P	/	P	⊃	Q	/	P	&	¬Q
1	1		1				1				0	
1	0		1				0				1	
0	1		1				1				0	
0	0		1				1				0	

So, there is no row on which all of the sentences are true. The set is then inconsistent.

4.7 Logical Entailment and Validity

Entailment is a relation between a set of sentences of a language and a sentence of that language.

A set Γ logically entails a sentence α if and only if there is no model on which every member of Γ is true and α is false.

The symbol for logical entailment is the double turnstile '⊨'

This expression

$$\Gamma \models \alpha$$

means that Γ entails α, and the expression

$$\Gamma \not\models \alpha$$

means that Γ does not entail α.

When we test for logical entailment using a truth table, we'll single slashes between each member of Γ and a double slash between the members of Γ and the purportedly entailed sentence. Here's a simple example of a classic argument form called *Modus Ponens*:

$$\{P, P \supset Q\} \models Q$$

P	Q	P	/	P	$\supset$	Q	//	Q
1	1	1			1			1
1	0	1			0			1
0	1	0			1			0
0	0	0			1			0

We now check to make sure that there are no rows on which all of the premises are true and the conclusion is false. The only row on which all of the premises are true is the first one, but on that row, the conclusion is also true. So, there is no row on which all of the premises are true and the conclusion is false, so the set containing the premises logically entails the conclusion, which is to say that the argument is valid.

Compare that to this:

P	Q	P	$\supset$	Q	/	Q	//	P
1	1		1			1		1
1	0		0			0		1
0	1		1			1		0
0	0		1			0		0

The third row demonstrates that $\{P \supset Q, Q\}$ does not logically entail P.

Entailment and validity are closely related concepts. An argument is logically valid if and only if there is no model on which all of the premises are

true and the conclusion is false. This means that an argument is logically valid just in case the set containing the premises logically entails the conclusion.

Let's check this argument for validity:

1. $H \equiv I$
2. $\underline{\neg H \vee \neg I}$

$\therefore \neg H \,\&\, \neg I$

H	I	H	$\equiv$	I	/	$\neg$H	$\vee$	$\neg$I	//	$\neg$H	&	$\neg$I
1	1		1				0				0	
1	0		0				1				0	
0	1		0				1				0	
0	0		1				1				1	

The fourth row is the only row on which both premises are true, but the conclusion is also true on that row. So, the argument is logically valid.

Now, let's look at this argument:

1. $W \supset (X \vee Y)$
2. $X \supset (\neg Y \vee Z)$
3. $\underline{\quad W \quad}$

$\therefore Z$

With four simple sentences, this is a long truth-table. So, let's take a shortcut or two. Notice first that the third premise is only true on the last eight rows of the truth-table. That means that there won't be any lines in the upper half of the truth-table where the premises are all true and the conclusion false.

W	X	Y	Z	W	⊃	(X	∨	Y)	/	X	⊃	(¬Y	∨	Z)	/	W	//	Z
1	1	1	1															
1	1	1	0															
1	1	0	1															
1	1	0	0															
1	0	1	1															
1	0	1	0															
1	0	0	1															
1	0	0	0															
0	1	1	1													1		
0	1	1	0													1		
0	1	0	1													1		
0	1	0	0													1		
0	0	1	1													1		
0	0	1	0													1		
0	0	0	1													1		
0	0	0	0													1		

Now, the conclusion is false on only half of those rows.

W	X	Y	Z	W	⊃	(X	∨	Y)	/	X	⊃	(¬Y	∨	Z)	/	W	//	Z
1	1	1	1															
1	1	1	0															
1	1	0	1															
1	1	0	0															
1	0	1	1															
1	0	1	0															
1	0	0	1															
1	0	0	0															
0	1	1	1													1		1
0	1	1	0													1		0
0	1	0	1													1		1
0	1	0	0													1		0
0	0	1	1													1		1
0	0	1	0													1		0
0	0	0	1													1		1
0	0	0	0													1		0

That means that we only need to check the truth-value of the other two premises on rows, 10, 12, 14, and 16. The first premise is true on rows 8–12, since W is false on those rows. So, it really depends on the second premise. Since X is false on the bottom four rows, making the second premise true there.

W	X	Y	Z	W	⊃	(X	∨	Y)	/	X	⊃	(¬Y	∨	Z)	/	W	//	Z
1	1	1	1															
1	1	1	0															
1	1	0	1															
1	1	0	0															
1	0	1	1															
1	0	1	0															
1	0	0	1															
1	0	0	0															
0	1	1	1		1											1		1
0	1	1	0		1											1		0
0	1	0	1		1											1		1
0	1	0	0		1											1		0
0	0	1	1		1						1					1		1
0	0	1	0		1						1					1		0
0	0	0	1		1						1					1		1
0	0	0	0		1						1					1		0

We shown that the premises are true and the conclusion is false on at least rows 14 and 16. That's one more than needed to prove that the argument is invalid.

4.8 Short Truth Tables

If done correctly, truth-tables are a foolproof way of answering questions about truth-functional properties. They do have a significant drawback, though—they can quickly get very large and cumbersome. Although we have seen how we can take some shortcuts, we still often have to complete large portions of a truth table before we can draw a conclusion. It would nice if we were able to go straight the row, if there is one, that proves that an argument is invalid or a set is consistent, or else easily prove that there is no such row.

In this section, I'll introduce a method called the short truth-table that can do that, and that works very well *most* of the time.

Let's try it out on the following argument:

1. $A \vee B$
2. $B \supset C$
3. $\underline{\neg A}$

$\therefore$ C

We start by listing the premises and conclusion at the top of a truth table, but we only need to leave room for one line:

A	B	C	A	$\vee$	B	/	B	$\supset$	C	/	$\neg$	A	//	C

Now, we will assume that the argument is invalid. That is, we assume that there is at least one row on which the premises are true and the conclusion is false. So, we'll put a 1 under the main connective of each premises and a 0 under the main connective of the conclusion.

A	B	C	A	$\vee$	B	/	B	$\supset$	C	/	$\neg$	A	//	C
				1				1				1		0

Now, we fill in what we know. The same sentence letter must have the same truth-value no matter where it occurs in the argument. So, 'C' must be false in the second premise.

A	B	C	A	$\vee$	B	/	B	$\supset$	C	/	$\neg$	A	//	C
				1				1	0			1		0

We can't really do anything with the first premise yet, since there are three different ways that a disjunction can be true. We do know, however, that 'A' must be false in the third premise, which would make it false in the first premise as well.

A	B	C	A	$\vee$	B	/	B	$\supset$	C	/	$\neg$	A	//	C
			0	1				1	0			1	0	0

Now, since we have a true disjunction with one false disjunct, we know the other disjunct must be true.

A	B	C	A	∨	B	/	B	⊃	C	/	¬	A	//	C
			0	1	1			1	0		1	0		0

Finally, we fill in the truth-value for B in the second premise.

A	B	C	A	∨	B	/	B	⊃	C	/	¬	A	//	C
			0	1	1		1	1	0		1	0		0

Now, we have a problem. Assuming that the argument was invalid led to a contradiction. In this case, the contradiction is a true conditional with a true antecedent and a false consequent. So, assumption must have been false, and the argument is therefore valid. There are two kinds of contradictions that can occur. The first is having a simple sentence that is true in one place but false in another. The second is having a complex sentence with a truth-value that is incompatible with the truth-values of the component simple sentences.

Let's back up a few steps to see another way that it could have gone, although it won't change anything about the validity of the argument.

A	B	C	A	∨	B	/	B	⊃	C	/	¬	A	//	C
			0	1				1	0		1	0		0

Instead of completing the first premise, we could have gone to the second premise. The only way that a conditional can be true if it has a false consequent is that the antecedent is false. So, we'll put 0 under the 'B'.

A	B	C	A	∨	B	/	B	⊃	C	/	¬	A	//	C
			0	1			0	1	0		1	0		0

Now, we'll have to put 0 under the 'B' in the first premise.

A	B	C	A	∨	B	/	B	⊃	C	/	¬	A	//	C
			0	1	0		0	1	0		1	0		0

But that gives us a true disjunction with two false disjuncts, another contradiction. There is no way that we can construct a line where this valid argument has all true premises and a false conclusion. Let's try another one:

1. $A \equiv (B \vee C)$

2. $\underline{\neg B \vee \neg D}$
$\therefore\ \neg A$

We'll start by making the heading row of the truth table.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A

Fill in the truth values under the main connectives like this:

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
					1							1				0	

There are too many ways that biconditional and disjunctions can be true, so the only place we can start is the conclusion.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
					1							1				0	1

Now, we can fill in something in the first premise.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
				1	1							1				0	1

Since the first half of that biconditional is true, we know the second half must be also.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
				1	1		1					1				0	1

Now, the only thing we can do is to take a guess. Let's assume that both 'B' and 'C' are true in the first premise. That will make the bicondional true.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
				1	1	1	1	1				1				0	1

And if 'B' is true, then '¬B' must be false in the second premise.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
				1	1	1	1	1		0	1	1				0	1

So, now we have a true conditional with one false disjunct, so the other disjunct must be true.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
				1	1	1	1	1		0	1	1	1			0	1

This means that 'D' must be false.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
				1	1	1	1	1		0	1	1	1	0		0	1

We have now filled in the truth-table, and our assumption that the argument was invalid has led to no contradictions. Not only do we know that the argument is invalid, we can state one of the rows on which it would have true premises and a false conclusion. Our last step is to fill in the model on the left by copying the truth-values underneath the simple sentences.

A	B	C	D	A	≡	(B	∨	C)	/	¬	B	∨	¬	D	/	¬	A
1	1	1	0	1	1	1	1	1		0	1	1	1	0		0	1

Remember that I said that this works very well *most* of the time. We start by assuming the argument is invalid. If that's the only assumption that we have to make, then we should get our answer fairly easily. In the second example, though, I had to make some further assumptions—I assumed that both

'B' and 'C' were true. If, on those assumptions, we do *not* get any contradictions, then we have successfully proved the argument to be invalid. If we do get a contradiction, however, we have only proved that at least one of our assumptions was false. It need not have been the assumption that the argument was invalid. So, if there had been a contradiction, we would have needed to assume that B was true and C was false to see if we still got a contradiction. If we did, then we would have to assume that C was true and B was false and tried again. The bottom line is that the more assumptions we make, the more difficult it will be to fill everything in.

4.9 Semantics for SL

Finally, let's formally state the semantics of *SL*:

- If α is a propositional constant, α is true in v if and only if $v(\alpha) = 1$.
- $\neg\alpha$ is true in v if and only if α is not true in v.
- $\alpha \,\&\, \beta$ is true in v if and only if α is true in v and β is true in v.
- $\alpha \vee \beta$ is true in v if and only if either α is true in v or β) is true in v, or both.
- $\alpha \vee \beta$ is true in v if and only if either α is not true in v or β is true in v.
- $\alpha \equiv \beta$ is true in v if and only if either both α and β are true in v or neither α nor β are true in v.

These clauses determine the truth value for any well-formed formula of *SL* that uses any of the five logical connectives defined.

We can also carefully define some important logical concepts using the notion of a model:

- A formula is a logical truth if and only if it is true on every model.
- A formula is a logical false hood if and only if it is false on every model.
- A formula is logically contingent if and only if it is neither a logical truth nor a logical falsehood.
- Two formulas are equivalent if and only if they have the same truth value on every model.
- A set of formulas is consistent if and only if there is at least one model in which they are all true.
- A formula α entails another formula β if and only if there is no model in which α is true and β is false.

- An argument is valid if and only there is no model in which all of its premises are true and its conclusion is false.

Chapter 5

Sentential Logic: Truth Trees

Truth-tables provide a mechanical means of determining the logical status of sentences, sets, and arguments. You simply follow the rules, and the table will give you the answer. As I mentioned before, though, they have a significant drawback. With every additional propositional constant, the truth-table doubles in size. Truth-trees provide another means for determining consistency, validity, etc., and in many cases truth-trees are far more efficient than truth-tables.

Strictly speaking, truth-trees are methods for determining consistency of sets. Since the concepts of tautology, contradiction, consistency, logical equivalence, and validity can all be defined in terms of consistency, truth trees can be very powerful tools for logical analysis.

Remember that a set is consistent if and only if its members are all true in at least one model. A set is inconsistent if and only if there is no model in which all of its members are true. Now, let's define our other logical concepts in terms of consistency.

Logical Falsehood : α is logically false if and only if $\{\alpha\}$ is inconsistent.

Logical Truth : α is logically true if and only if $\{\neg\alpha\}$ is inconsistent.

Logical Indeterminacy : α is logically indeterminate if and both $\{\alpha\}$ and $\{\neg\alpha\}$ are consistent.

Logical Equivalence : α and β are logically equivalent if and only if $\{\neg(\alpha \equiv \beta)\}$ is inconsistent.

Logical Entailment : $\Gamma \models \alpha$ if and only if $\Gamma \cup \{\neg\alpha\}$ is inconsistent.

Logical Validity : An argument is logically valid if and only the set containing the premises and the negation of the conclusion is inconsistent.

The trees work by breaking down complex sentences into increasingly simpler sentences, until nothing is left except for simple sentences and the negations of simple sentences. Complex sentences are broken down by using the following decomposition rules.

5.1 Decomposition Rules

Conjunctions

Conjunction Decomposition

$\alpha \,\&\, \beta$ ✓
α
β

Negated Conjunction Decomposition

$\neg(\alpha \,\&\, \beta)$ ✓
$\neg\alpha$ $\quad$ $\neg\beta$

Note that the tree on the right splits, or has branches. A tree branches when there is more than one way that a sentence could be true. In order for a conjunction to be true, both conjuncts must be true. So, the tree for a conjunction does not branch. There are two ways, however, that conjunctions could be false — either the first conjunct is false or the second conjunct is false.

Knowing the truth conditions for disjunctions, conditionals, and biconditionals, you should be able to see what the trees for those sentences and their negations will look like. Disjunctions and conditionals will both branch, but their negations will not. The trees for biconditionals and their negations will both branch.

Disjunctions

Disjunction Decomposition

$\alpha \vee \beta$

$\alpha \quad \beta$

Negated Disjunction Decomposition

$\neg(\alpha \vee \beta)$

$\neg\alpha$

$\neg\beta$

Conditionals

Conditional Decomposition

$\alpha \supset \beta$ ✓

$\neg\alpha \quad \beta$

Negated Conditional Decomposition

$\neg(\alpha \supset \beta)$ ✓

α

$\neg\beta$

Biconditionals

Biconditional Decomposition

$\alpha \equiv \beta$ ✓

$\alpha \,\&\, \beta \quad \neg\alpha \,\&\, \neg\beta$

Negated Biconditional Decomposition

$\neg(\alpha \equiv \beta)$ ✓

$\alpha \,\&\, \neg\beta \quad \neg\alpha \,\&\, \beta$

Double Negations

Finally, we need a rule for double negations. As you can guess, it's rather simple:

Double Negation Decomposition

$\neg\neg\alpha$ ✓

α

5.2 Logical Analysis with Trees

A branch includes all of the sentences acquired by reading from the bottom of the tree upwards to the top. If a branch contains both a simple sentence and its negation, then that branch is closed. If all of the complex sentences on a

branch have been completely decomposed, and that branch does not contain a simple sentence and the negation of that same sentence, then it is a completed open branch. A tree is completed when every branch is either a closed branch or a completed open branch. A tree is closed if all of its branches are closed. It is open if it contains at least one completed open branch. Sentences that have been decomposed are marked with a checkmark; branches that are closed are marked with an "x."

Strategically, the goal is to close branches as quickly as possible. That will keep the tree manageable. Sentences may be decomposed in any order. The best strategy is to first decompose any sentence that does not branch, then decompose sentences that will result in a closed branch.

Consistency

To use a tree to determine the consistency of a set, start by listing the members of the set at the top of the tree on separate, numbered lines.

1. $P \supset Q$
2. $\neg(Q \vee R)$
3. $R \vee \neg P$

The first thing to do is to decompose line 2, since it doesn't branch. Just as in derivations, we'll list what line these came from and what rule was used. We'll also check off line 2, so that we'll know that it has been decomposed already.

1. $P \supset Q$
2. $\neg(Q \vee R)$ ✓
3. $R \vee \neg P$
4. $\neg Q$ 2 $\neg \vee$ D
5. $\neg R$ 2 $\neg \vee$ D

Lines 1 and 3 will both branch, so we should see if either will result in a closed branch. In this case, both will, so it doesn't matter which we do first.

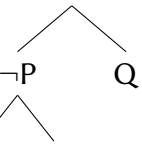
1. $P \supset Q$ ✓
2. $\neg(Q \vee R)$ ✓
3. $R \vee \neg P$
4. $\neg Q$ — 2 ¬ ∨ D
5. $\neg R$ — 2 ¬ ∨ D

6. $\neg P$ | Q ✗ — 1 ⊃ D

Note that the branch on the right now contains Q and ¬Q, so it is closed. We will mark it as closed with some symbol, an "X" is common, especially when writing trees out by hand. If a branch is closed, then nothing more needs to be done on that branch.

Now, we just have line 3 remaining to finish.

1. $P \supset Q$ ✓
2. $\neg(Q \vee R)$ ✓
3. $R \vee \neg P$ ✓
4. $\neg Q$ — 2 ¬ ∨ D
5. $\neg R$ — 2 ¬ ∨ D



6. $\neg P$ | Q — 1 ⊃ D

7. R ✗ | $\neg P$ ✗

An open tree contains at least one open branch; a closed tree contains no open branches.

So, we finished decomposing all of the complex sentences, but we still have one open branch. That means our original set is consistent. The tree also gives us a model on which all of the set's members are true. We just read up an open branch and assign F to any negated simple sentence and T to any non-negated simple sentence. In this case, the members of the set are all true when P, Q, and R are all false.

Trees for single sentences can be used to determine if a sentence is a tautology, a contradiction, or a contingency. If the tree for a sentence is closed, then it is a contradiction. If the tree for the negation of a sentence is closed,

then the sentence is a tautology. If neither the tree for the sentence nor the tree for its negation is closed, then the sentence is contingent.

5.3 Contradictions and Tautologies

To determine whether a sentence is a contradiction, we do a tree for that sentence. If the tree is closed, then there is no model in which the sentence is true, and it is therefore a contradiction. Here's an example:

P & (¬(Q ⊃ P))

We start by writing the sentence on line 1.

1. P & ¬(Q ⊃ P)

Then, we will decompose it using the conjunction decomposition rule, and checking it off showing that we're finished with it.

1.	P & ¬(Q ⊃ P) ✓	
2.	P	1 & D
3.	¬(Q ⊃ P) ✓	1 & D

Line 2 is already a simple sentence, so there's nothing we need to do with it. Decomposing the negated conditional on line 3 produces '¬P', which closes the branch.

1.	P & ¬(Q ⊃ P) ✓	
2.	P	1 & D
3.	¬(Q ⊃ P) ✓	1 & D
4.	Q	3 ¬ ⊃ D
5.	¬P	3 ¬ ⊃ D
	✗	

Since the tree has only one branch and it is closed, there is no model on which the original sentence is true. So, it is a contradiction.

To determine if a sentence is a tautology, we do a tree for the negation of the sentence. Since the negation of a tautology is false on every model, the tree will be closed. Here's an example:

P ⊃ (Q ⊃ P)

We start by putting the negation of the sentence on line 1.

1. ¬[P ⊃ (Q ⊃ P)]

After that, decompose line 1 with negated conditional decomposition.

1. ¬[P ⊃ (Q ⊃ P)] ✓
2. P
3. ¬(Q ⊃ P)

Now, decompose line 3, again using negated conditional decomposition

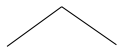
1.	¬[P ⊃ (Q ⊃ P)] ✓	
2.	P	1 ¬ ⊃ D
3.	¬(Q ⊃ P) ✓	1 ¬ ⊃ D
4.	Q	3 ¬ ⊃ D
5.	¬P	3 ¬ ⊃ D
	✗	

This tree tells us that '¬[P ⊃ (Q ⊃ P)]' is a contradiction, which means that our original sentence, 'P ⊃ (Q ⊃ P)' is tautology.

If a tree for a sentence is open, we only know that it is not a contradiction, *even if every branch is open*. Likewise, if tree for a negated sentence is open, we only know that the original sentence is not a tautology. Let's see a simple example:

P ∨ (Q & R)

1. P ∨ (Q & R) ✓



2.	P	Q & R ✓	1 ∨ D
3.		Q	2 & D
4.		R	2 & D

Note that both branches are open. Now, we can easily state a model on which 'P ∨ (Q & R)' is true. The left-hand branch tells us that any model in which 'P' is true will be a model in which 'P ∨ (Q & R)' is true. We can fill in any values for 'Q' and 'R' that we like. Here's one such model:

P = 1
Q = 1
R = 1

Note, though, that even though every branch is open, our sentence is definitely not a tautology. 'P ∨ (Q & R)' is false whenever both P and at least

one of Q or R are false. We can show this by doing a tree for the negation of the sentence. Here's the completed tree:

1. $\neg[P \vee (Q \,\&\, R)]$ ✓
2. $\neg P$ $1\neg \vee D$
3. $\neg(Q \,\&\, R)$ ✓ $1\neg \vee D$

4. $\neg Q$ $\neg R$ $3\neg \,\&\, R$

Now, since we have an open tree for both '$P \vee (Q \,\&\, R)$' and its negation, we have shown that the sentence is logically indeterminate, or contingent.

5.4 Equivalence

Remember that a biconditional is true whenever the two sentences joined by the biconditional operator have the same truth value. So, if the two sentences have the same truth value in every model, then the biconditional will be a tautology. This means that we can test whether two sentences are logically equivalent by joining them with a triple bar, negating the resulting sentence, then working a truth-tree for it. If the tree is closed, then the two original sentences are logically equivalent.

Let's test it with a simple example:

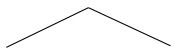
$P \supset Q \quad \neg P \vee Q$

First, we will make a negated biconditional using the two sentences.

1. $\neg[(P \supset Q) \equiv (\neg P \vee Q)]$

Now, we can decompose it using the negated biconditional rule.

1. $\neg[(P \supset Q) \equiv (\neg P \vee Q)]$ ✓



2. $P \supset Q$ $\neg(P \supset Q)$ $1\neg \equiv D$
3. $\neg(\neg P \vee Q)$ $\neg P \vee Q$ $1\neg \equiv D$

Now, we have to finish each branch; let's work on the left side first. Too keep things as simple as possible, remember to always decompose non-branching sentences first. In this case, we should decompose the negated disjunction.

1.	$\neg[(P \supset Q) \equiv (\neg P \vee Q)]$ ✓		
2.	$P \supset Q$	$\neg(P \supset Q)$	1¬ ≡ D
3.	$\neg(\neg P \vee Q)$ ✓	$\neg P \vee Q$	1¬ ≡ D
4.	$\neg\neg P$		3¬ ∨ D
5.	$\neg Q$		3¬ ∨ D

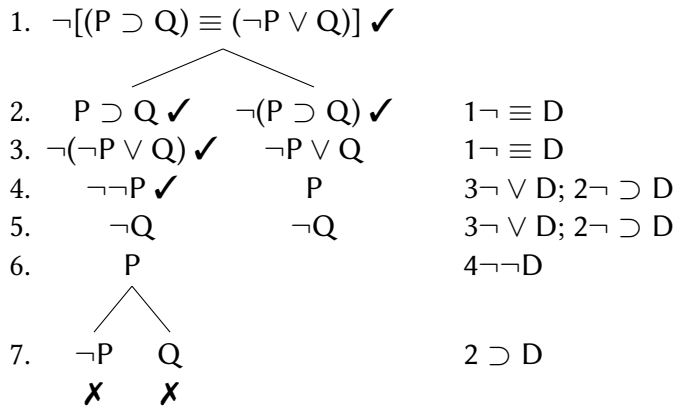
The next step is to take care of the double negation.

1.	$\neg[(P \supset Q) \equiv (\neg P \vee Q)]$ ✓		
2.	$P \supset Q$	$\neg(P \supset Q)$	1¬ ≡ D
3.	$\neg(\neg P \vee Q)$ ✓	$\neg P \vee Q$	1¬ ≡ D
4.	$\neg\neg P$ ✓		3¬ ∨ D
5.	$\neg Q$		3¬ ∨ D
6.	P		4¬¬D

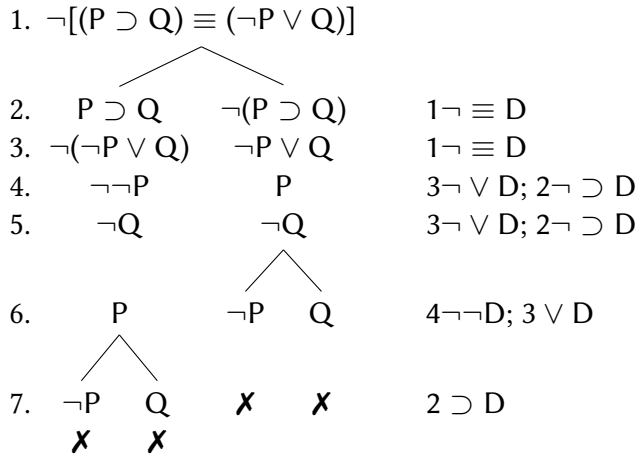
Now, we can decompose the conditional on line 2, which will result in two closed branches.

1.	$\neg[(P \supset Q) \equiv (\neg P \vee Q)]$ ✓		
2.	$P \supset Q$ ✓	$\neg(P \supset Q)$	1¬ ≡ D
3.	$\neg(\neg P \vee Q)$ ✓	$\neg P \vee Q$	1¬ ≡ D
4.	$\neg\neg P$ ✓		3¬ ∨ D
5.	$\neg Q$		3¬ ∨ D
6.	P		4¬¬D
7.	$\neg P$ ✗	Q ✗	2 ⊃ D

Now, we'll decompose the negated conditional on the main right hand branch, since it doesn't branch.



Finally, we decompose the disjunction on line 3, which gives us two new closed branches.



5.5 Validity

Since validity can be defined in terms of consistency, truth trees can be used to determine if an argument if valid. List the premises on numbered lines at the top, followed by the negation of the conclusion. Then, complete the tree using the same rules and methods as above. If the tree is closed, then there is no way for the premises to be true and the conclusion false, and the argument is valid. If the tree is open, then there is at least one model on which the

premises are true and the conclusion false, and the argument is invalid. A truth-value assignment showing invalidity can be read from any open branch.

Here's a simple tree on a version of DeMorgan's Law, P & Q $\vdash \neg(\neg P \vee \neg Q)$. First, we list the premises and the negated conclusion.

1. P & Q
2. $\neg\neg(\neg P \vee \neg Q)$

Then, we decompose any non-branching sentences.

1.	P & Q ✓	Premise
2.	$\neg\neg(\neg P \vee \neg Q)$ ✓	Negated Conclusion
3.	P	1 &D
4.	Q	1 &D
5.	$\neg P \vee \neg Q$	2 $\neg\neg$ D

We then continue by decomposing any branching sentences, beginning with those that will result in closed branches. In this case, we only have line 5 remaining.

1.	P & Q ✓	Premise
2.	$\neg\neg(\neg P \vee \neg Q)$ ✓	Negated Conclusion
3.	P	1 &D
4.	Q	1 &D
5.	$\neg P \vee \neg Q$ ✓	2 $\neg\neg$ D
6.	$\neg P$ ✗ $\quad$ $\neg Q$ ✗	5 $\vee$ D

Since all branches are closed, the argument is valid. Note that a tree for logical entailment works exactly the same way, simply put all of the members of the set and the negation of the entailed sentence on the tree.

Here is a more complex example:

(G & H) $\supset \neg I,\ I \vee G,\ H \supset \neg G,\ H \equiv I \vdash \neg H \vee G$

As before, we begin by list the premises and the negation of the conclusion.

1. $(G \ \& \ H) \supset \neg I$
2. $I \vee G$
3. $H \supset \neg G$
4. $H \equiv I$
5. $\neg(\neg H \vee G)$

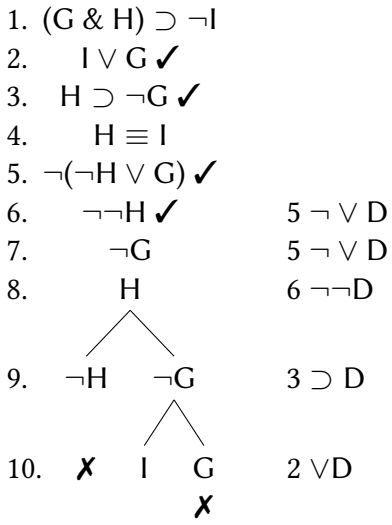
Unfortunately, there is only line that doesn't branch. So, we'll decompose line 5, followed by decomposing the double negation that will result.

1. (G & H) ⊃ ¬I
2. I ∨ G
3. H ⊃ ¬G
4. H ≡ I
5. ¬(¬H ∨ G) ✓
6. ¬¬H ✓ 5 ¬ ∨ D
7. ¬G 5 ¬ ∨ D
8. H 6 ¬¬D

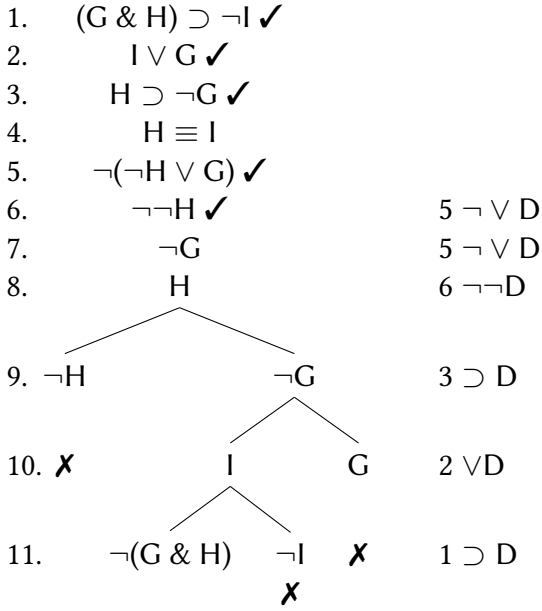
Now, we have a choice. Both line 2 and line 3 will branch, but they will both also result in an immediately closed branch. Let's start with line 3.

1. (G & H) ⊃ ¬I
2. I ∨ G
3. H ⊃ ¬G ✓
4. H ≡ I
5. ¬(¬H ∨ G) ✓
6. ¬¬H ✓ 5 ¬ ∨ D
7. ¬G 5 ¬ ∨ D
8. H 6 ¬¬D
9. ¬H | ¬G 3 ⊃ D
   ✗

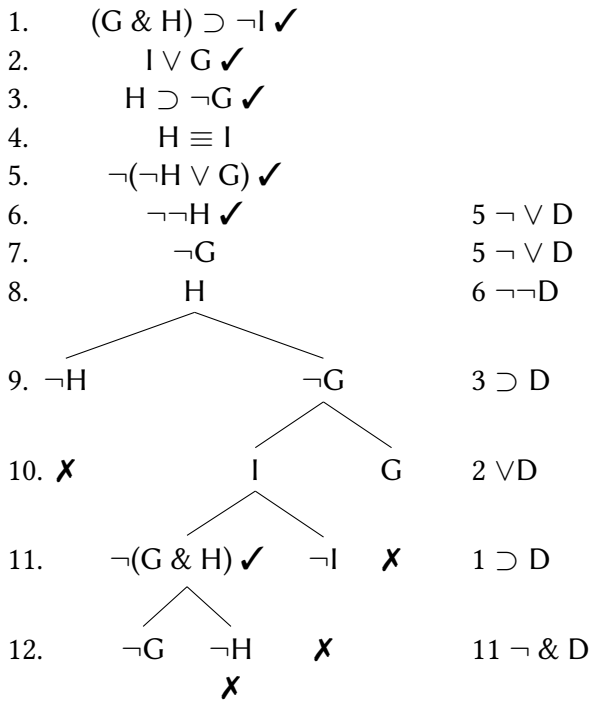
We'll next do line 2.



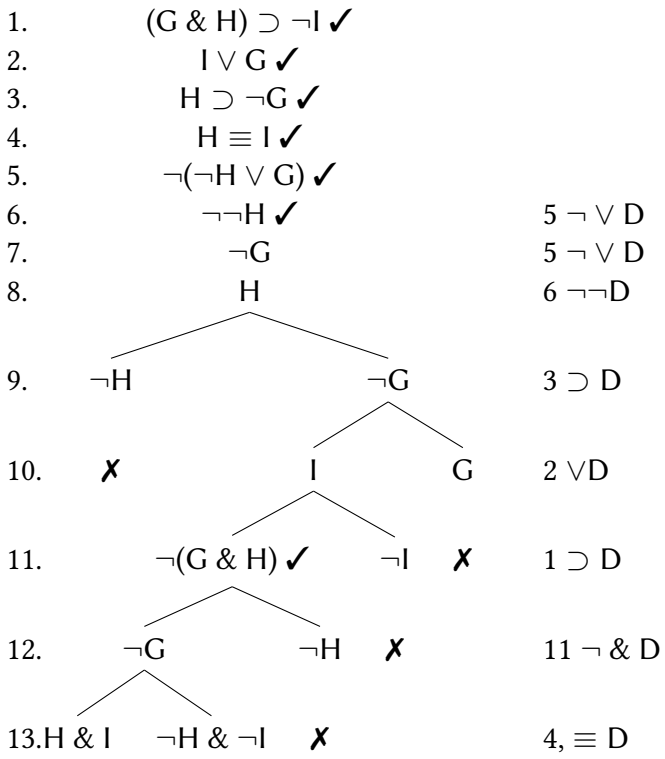
That just leaves two lines. I usually try to avoid biconditionals until the end, so I'll decompose line 1.



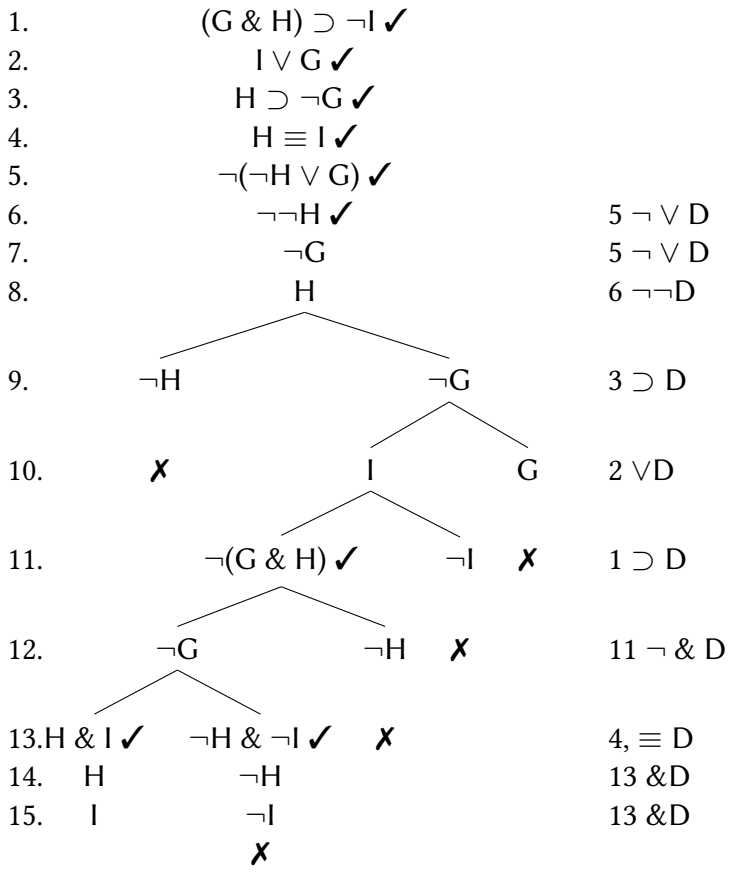
Now, I'll decompose the negated conjunction in line 11.



The right branch closes with H and ¬H. Now I'm forced to decompose the biconditional in line 4.



All that remains now is to decompose the two conjunctions in line 13.



The tree is finally finished. The right branch is closed, but the left branch is not. It tells us that the premises of the original argument will be true and the conclusion will be false whenever G is false, H is true, and I is true. That would be the fifth line of the truth table, and would look like this:

G	H	I	(G	&	H)	⊃	¬I	/	I	∨	G	/	H	⊃	¬G	/	H	≡	I	//	¬H	∨	G
F	T	T	F	F	T	**T**	F		T	**T**	F		T	**T**	T		T	**T**	T		F	**F**	F

Chapter 6

Sentential Logic: Derivations

6.1 The Derivation System SD

The first derivation rule of *SD* is *Reiteration*

Reiteration (R)

	α
$\triangleright$	α

The symbol '$\triangleright$' indicates the formula we are allowed to derive using the rule. Reiteration does nothing more than simply allowing us to derive a sentence from itself. Here's an example of a a derivation that uses the Reiteration rule.

1	$P \supset (Q \vee R)$	
2	$\neg(Q \,\&\, S)$	
3	$P \supset (Q \vee R)$	R 1

The first two lines of the derivation are assumptions. Assumptions are set off by a horizontal line from the derived sentences that follow. The vertical line is called a scope line. Notice that the lines of the proof are numbered. To the right of each derived formula will be the rule that was used to derive that formula, and the line numbers to which the rule was applied.

Along with Reiteration, *SD* includes rules for introducing and eliminating each of the five logical connectives in *SL*. The conjunction rules are the simplest, so let's start there.

Conjunction Introduction (&I)

	α			α
	β	or		β
▷	α & β		▷	β & α

This rule allows us to form a conjunction of any two formulas occurring on previous lines. The order that those formulas appear in the proof doesn't matter. Even if β occurred before α, we could still derive α & β.

Conjunction Elimination (&E)

	α & β			α & β
		or		
▷	α		▷	β

We have two different versions of the same rule—this allows us to derive either of the two conjuncts using the single conjunction elimination rule.

Let's see an example derivation using the conjunction rules.

Derive: W & (X & Y)

1	W & X
2	Y & Z

A good strategy is to start with the sentence that is to be derived and then think backwards. To form 'W & (X & Y)' using the &I rule, we will need to have 'W' on one line and 'X & Y' on another. We can get 'W' from line 1, so let's start there.

1	W & X	
2	Y & Z	
3	W	&E 1

To form ‘X & Y’, we will need to have a line with each formula. One will come from line 1 and the other from line 2.

1	W & X	
2	Y & Z	
3	W	&E 1
4	X	&E 1
5	Y	&E 2

Now, we can from ‘X & Y’ using conjunction introduction on lines 4 and 5.

1	W & X	
2	Y & Z	
3	W	&E 1
4	X	&E 1
5	Y	&E 2
6	X & Y	&I 4, 5

Finally, we use lines 1 and 6 to form ‘W & (X & Y)’

1	W & X	
2	Y & Z	
3	W	&E 1
4	X	&E 1
5	Y	&E 2
6	X & Y	&I 4, 5

Conditional Elimination ($\supset$ E)

	$\alpha \supset \beta$
	α
$\triangleright$	β

The conditional elimination rule, also called 'Modus Ponens', tells us that the consequent of a conditional can be derived whenever the conditional and its antecedent occur on previous lines. Here's a derivation using conditional elimination.

Derive: H

1	F /H	
2	G	
3	(F & G) $\supset$ H	
4	F & G	&I 1, 2
5	H	$\supset$E 3, 4

Conditional Introduction ($\supset$ I)

	α	
	β	
$\triangleright$	$\alpha \supset \beta$	

Conditional introduction is often called 'conditional proof'. Notice that it uses a *subderivation*. The subderivation begins with α as an assumption, and then derives β. Since β can be derived from α, the conditional $\alpha \supset \beta$ must be true.

Look at this example of a derivation using conditional proof:

1. If Alan passes Organic Chemistry, then he will graduate.
2. If Alan graduates, then he will move to Kansas.

$\therefore$ If Alan passes Organic Chemistry, then he will move to Kansas.

1	P ⊃ G	
2	G ⊃ K	
3	| P	
4	| G	⊃E 1, 3
5	| K	⊃E 2, 4
6	P ⊃ K	⊃I 3–5

The subderivation is on lines 3–5. It has vertical and horizontal lines just like the derivations that we have been doing so far. The horizontal line distinguishes assumptions from derived formulas; the assumptions of the main derivation are called primary assumptions, and the assumptions of any subderivations are called auxiliary assumptions. The vertical line is the scope line. The scope line shows the scope of any assumptions. The subderivation shows that, given lines 1 and 2, if we assume 'P', we can derive 'P ⊃ K'. We complete the subderivation by ending the scope line of the subderivation and moving the scope line that is immediately to the left of the subderivation. By ending the subderivation, we discharge the assumption. After the assumption is discharged, the lines in the subderivation are no longer accessible, and thus cannot be used later in the proof. For example, this is not allowed:

1	P ⊃ G	
2	G ⊃ K	
3	| P	
4	| G	⊃E 1, 3
5	| K	⊃E 2, 4
6	K	MISTAKE! R 5

This claims that, given lines 1 and 2, we can derive 'K', but we cannot do that without making the further assumption of line 3. What we can do, though, is appeal to the entire subderivation, as in our original example:

1	P ⊃ G	
2	G ⊃ K	
3	| P	
4	| G	⊃E 1, 3
5	| K	⊃E 2, 4
6	P ⊃ K	⊃I 3–5

Note that the justification for line 6 cites lines 3–5, which is the entire subderivation.

Disjunction Introduction (∨I)

	α			α
		or		
▷	α ∨ β		▷	β ∨ α

This is an unusual rule for two reasons. First, it starts with a definite claim, α, and derives a vague claim, α ∨ β. Second, β might be completely new to the derivation, giving us a feeling of deriving something from nothing. The rule is obviously truth-preserving, though, since α is true, it has to be the case that α ∨ β is true. Disjunction Introduction is particularly useful when a formula required to complete a derivation contains a sentence-letter that is not found in any of the premises. For example,

1	F & G	
2	(F ∨ H) ⊃ E	
3	F	&E 1
4	F ∨ H	∨I 3
5	E	⊃E 4, 3
▷	E ∨ I	∨I 5

Disjunction Elimination (∨E)

1	$\alpha \vee \beta$
2	$\quad \alpha$
3	$\quad \gamma$
4	$\quad \beta$
5	$\quad \gamma$
$\triangleright$	γ

We can derive γ from $\alpha \vee \beta$ if we can both derive γ from α and derive γ from β. In other words, if we have sentence of the form $\alpha \vee \beta$, a subderivation that begins with the assumption α and ends with γ, and another subderivation that begins with the assumption β and ends with γ, we can derive the sentence γ.

Negation Introduction ($\neg$I)

$\quad \alpha$
$\quad \beta$
$\quad \neg\beta$
$\neg\alpha$

$\neg\alpha$ can be derived by using a subderivation that assumes α and derives β and $\neg\beta$ from α Both β and $\neg\beta$ must be immediately to the right of same scope line as α. Intuitively, anything that entails a contradiction must be false.

Negation Elimination $\neg$E

Negation Elimination works the same way:

$\quad \neg\alpha$
$\quad \beta$
$\quad \neg\beta$
α

Biconditional Introduction

1		α
2		β
3		β
4		α
▷	α ≡ β	

This rule makes sense if we remember that a biconditional is logically equivalent to the conjunction of two conditionals, that is, $\alpha \equiv \beta$ is equivalent to $(\alpha \supset \beta)\ \&\ (\beta \supset \alpha)$.

Biconditional Elimination (≡ E)

	α ≡ β			α ≡ β	
	α	or		β	
▷	β		▷	α	

Note that Biconditional Elimination works just like Conditional Elimination, except that either term can be derived from the biconditional and the other term.

6.2 Using the Rules of SD

Here are some important things to keep in mind when using *SD*. First, all of the rules of *SD* are rules of inference, and rules of inference apply only to complete lines of a derivation. For example, consider this simple derivation:

Derive: A

	(A & B) & C

We are given a conjunction, and we know that all three of those conjuncts have to be true for that conjunction to be true. So, it is tempting to do this:

1	(A & B) & C	
2	A	&E 1

NO!

The rule of inference can only be applied to the entire line, not part of the line. So, we first have to use Conjunction Elimination to break down the first line, then use it again to finish the proof.

1	(A & B) & C	
2	A & B	&E 1
3	A	&E 2

Second, the only lines that can be used for the rules of *SD* are lines that are accessible at that stage of the proof. A formula is accessible at line *n* if and only if it is not within the scope of an assumption that has been discharged before line *n*.

Notice this mistake:

1	F ⊃ G	
2	G ⊃ H	
3	| F	
4	| G	⊃E 1, 3
5	| H	⊃E 2, 4
6	F ⊃ H	⊃I 3–5
7	F	R 3
8	H	⊃E 6, 7

NO!

Lines 1–6 are correct. The subderivation assumes F, and derives G. It is important to remember that when a subderivation is used, the entire subderivation must be cited. Here, line 6 cites Conditional Introduction and lines 3–5. The mistake occurs in line 7. Line 7 reiterates line 3, but line 3 is not accessible at 7. Once the assumption of the subderivation is discharged, none of the lines within that subderivation can be used in the remaining part of the proof.

6.3 SD: Basic Concepts

SD is the basic derivation system for *SL*. Stated precisely, A derivation in *SD* is a series of sentences of *SL*, in which each sentence is either an assumption with a line indicating its scope or is justified by one of the rules of *SD*. Now that we have a basic understanding of the rules of *SD*, it's time to define a few basic concepts. The first is a derivation:

> A **derivation in *SD*** is a series of sentences of *SL*, in which each sentence is either an assumption with a line indicating its scope or is justified by one of the rules of *SD*.

We define the concept of derivability like this:

> A sentence α of *SL* is **derivable in *SD*** from a set Γ of sentences of *SL* if and only if there is a derivation in *SD* in which all of the primary assumptions are members of Γ and α occurs in the scope of only those primary assumptions.

We can state that a sentence is derivable from a set using the single turnstile symbol like this:

$$\Gamma \vdash \alpha$$

We can say that a sentence is not derivable from a set like this:

$$\Gamma \nvdash \alpha$$

Derivability is the fundamental concept of *SD*; other concepts will be defined in terms of derivability. For instance, validity and invalidity in *SD* are defined like this:

> An argument of *SL* is **valid in *SD*** if and only if the conclusion of the argument is derivable in *SD* from the set consisting of the premises.
>
> An argument of *SL* is **invalid in *SD*** if and only if it is not valid in *SD*.

It is possible to derive some sentences from no premises at all, which is to say that those sentences can be derived from the empty set. To do that, we begin with a subderivation. Here is an example:

Derive $P \supset (Q \supset P)$

1	P	
2	Q	
3	P	R 1
4	$Q \supset P$	$\supset$I 2–3
5	$P \supset (Q \supset P)$	$\supset$I 1–3

These sentences are called theorems of *SD*.

> A sentence of *SL* is a **theorem of *SD*** if and only if it is derivable in *SD* from the empty set.

We can also define equivalence in *SD*:
Two sentences α and β of *SL* are **equivalent in *SD*** if and only if β is derivable in *SD* from $\{\alpha\}$ and α is derivable in *SD* from $\{\beta\}$

> A set Γ of *SL* is **inconsistent in *SD*** if and only if both a sentence α of *SL* and its negation are derivable in *SD* from Γ.
>
> A set Γ of *SL* is **consistent** if and only if it is not inconsistent.

In section 4.9, we defined many of those same concepts for *SL*. There, we defined them in terms of truth in a model. Here, they are defined in terms of derivability, not truth. The derivation system operates solely at the level of syntax, and truth is a semantic concept. Nevertheless, we want the concepts of the derivation *SD* to parallel the semantic concepts of *SL*:

- A sentence α of *SL* is derivable in *SD* from a set of sentences Γ in *SL* if and only if α is truth-functionally entailed by Γ.
- An argument of *SL* is valid in *SD* if and only if the argument is truth-functionally valid.
- A sentence α of SL is a theorem in *SD* if and only if α is truth-functionally true.

- Sentences α and β or *SL* are equivalent in *SD* if and only if they are truth-functionally equivalent.
- A set of sentences of *SL* is consistent in *SD* if and only if it is truth-functionally consistent.

This means that if an argument is truth-functionally valid in *SL*, then it will be derivable in *SD*. Unfortunately, this does not mean that every student of logic will be able to construct a derivation for the argument. Constructing a successful derivation is much more likely if approached strategically, rather than haphazardly applying the derivation rules.

6.4 Strategies for Derivations

The key to building derivations in *SD* is to focus on goals and sub-goals. The goal is the sentence that we have been tasked to derive. Strategizing derivations often looks something like this. Say that we have been tasked to derive some sentence α. Looking at the initial assumptions, we see that we can derive α if we can first derive β. We then see that deriving β requires deriving γ, and so on. So, α is our goal, but β abd γ become sub-goals. Let's work through an example:

Show that $\{\neg A, (\neg A \supset B) \,\&\, (C \equiv \neg A)\} \vdash (B \vee D) \,\&\, C$ in *SD*.

1	$\neg A$
2	$(\neg A \supset B) \,\&\, (C \equiv \neg A)$
⋮	⋮
n	$(B \vee D) \,\&\, C$

Now, note that the conclusion is a conjunction. We can use the Conjunction Introduction to form it, if we can derive each conjunct. The second conjunct, C, has to come from the right side of premise 2, which is $C \equiv \neg A$. So, our goal now becomes getting that on a line by itself. Since it's a conjunct, we can just use Conjunction Elimination.

1	¬A	
2	(¬A ⊃ B) & (C ≡ ¬A)	
3	C ≡ ¬A	&E 2
⋮	⋮	
n	(B ∨ D) & C	

Now, we can use Biconditional Elimination to produce C.

1	¬A	
2	(¬A ⊃ B) & (C ≡ ¬A)	
3	C ≡ ¬A	&E 2
4	C	≡E 3, 4
⋮	⋮	
n	(B ∨ D) & C	

That takes care of half of the conclusion. Now, we need to get B ∨ D. There's no D in the premises, which means that we'll have to introduce it using Disjunction Introduction. To do that, we first need to get B. B is the consequent of a conditional, ¬A ⊃ B, so we can get B if we have the antecedent, ¬A. Fortunately, ¬A is the first premise. So our completed proof looks like this:

1	¬A	
2	(¬A ⊃ B) & (C ≡ ¬A)	
3	C ≡ ¬A	&E 2
4	C	≡E 3, 4
5	¬A ⊃ B	&E 2
6	B	⊃E 5, 1
7	B ∨ D	∨I 6
8	(B ∨ D) & C	&I 7, 4

6.5 The Derivation System SD+

So far, *SD* has only eleven rules, Reiteration and an introduction and elimination rule for each of the five logical connectives. These eleven rules are enough to derive the conclusion from the premises of every valid argument in *SL*. There is a trade-off, however—the simpler the derivation system in terms of the number of available derivation rules, the more complex the derivations tend to be. For example, consider this argument:

1. $\alpha \vee \beta$
2. $\underline{\neg\alpha}$

$\therefore\ \beta$

The argument is obviously valid. The first premise tells us that least one of α and β are true, the second tells us that α is false, so β must therefore be true. Proving this with the basic rules of *SD*, however, is trickier than it seems it ought to be:

1	$\alpha \vee \beta$	
2	$\neg\alpha$	
3	α	
4	$\neg\beta$	
5	α	R 3
6	$\neg\alpha$	R 2
7	β	$\neg$E 4–6
8	β	
9	β	R 7
10	β	$\vee$E 3–7, 8–9

We have now proved, using the basic derivation rules of *SD*, that a disjunct can be derived from a disjunction and the negation of the other disjunct. So, we can add it as a new derivation rule to *SD*. There is no theoretical limit

to the number of rules that could be added. We could, in principle, add a new rule for every derivation that we successfully complete. The resulting system, however, would be practically unusable. We need a good balance between the simplicity of the system itself and the simplicity of the derivations when using the system. There are two criteria that we can use to determine if an additional rule should be added to the system. First, the circumstances to which the proposed rule applies should occur fairly often, thus saving steps in many different derivations. Second, those circumstances are easily recognizable. Having new rules won't help, if we can't tell when we need to use them.

Rules of Inference

Here are the rules of inference that we will add:

Disjunctive Syllogism (DS)

	$\alpha \vee \beta$			$\alpha \vee \beta$
	$\neg\alpha$	or		$\neg\beta$
$\triangleright$	β		$\triangleright$	α

Hypothetical Syllogism (HS)

	$\alpha \supset \beta$
	$\beta \supset \gamma$
$\triangleright$	$\alpha \supset \gamma$

Modus Tollens (MT)

	$\alpha \supset \beta$
	$\neg\beta$
$\triangleright$	$\neg\alpha$

Constructive Dilemma (CD)

$\alpha \supset \gamma$
$\beta \supset \delta$
$\alpha \vee \beta$
$\triangleright$ $\gamma \vee \delta$

Destructive Dilemma (DD)

$\alpha \supset \gamma$
$\beta \supset \delta$
$\neg\gamma \vee \neg\delta$
$\triangleright$ $\neg\alpha \vee \neg\beta$

Biconditional Modus Tollens (BMT)

$\alpha \equiv \beta$
$\neg\alpha$
$\triangleright$ $\neg\beta$

Biconditional Hypothetical Syllogism (BHS)

$\alpha \equiv \beta$
$\beta \equiv \gamma$
$\triangleright$ $\alpha \equiv \gamma$

Rules of Replacement

In addition to rules of inference, *SD+* contains rules of replacement. Rules of replacement let us derive some sentences from other sentences by replacing parts of sentences. Unlike rules of inference, which can be applied only to whole lines, rules of replacement can be applied to parts of lines. For example, using a rule called Contraposition, from the sentence

$(A \vee B) \equiv (A \supset C)$

we can infer

$(A \vee B) \equiv (\neg C \supset \neg A)$

Since every sentence is a part of itself, the rules of replacement can also be applied to entire lines. The rules of replacement are also bidirectional. Using Contraposition, I can not only go from

$A \supset C$

to

$\neg C \supset \neg A$

I can also go from

$\neg C \supset \neg A$

to

$A \supset C$

Here are the rules of replacement of *SD+*:

De Morgan's (DM)

$\neg(\alpha \,\&\, \beta) \Leftrightarrow \neg\alpha \vee \neg\beta$
$\neg(\alpha \vee \beta) \Leftrightarrow \neg\alpha \,\&\, \neg\beta$

Association (Ass)

$\alpha \vee (\beta \vee \gamma) \Leftrightarrow (\alpha \vee \beta) \vee \gamma$
$\alpha \,\&\, (\beta \,\&\, \gamma) \Leftrightarrow (\alpha \,\&\, \beta) \,\&\, \gamma$

Distribution (Dis)

$\alpha \,\&\, (\beta \vee \gamma) \Leftrightarrow (\alpha \,\&\, \beta) \vee (\alpha \,\&\, \gamma)$
$\alpha \vee (\beta \,\&\, \gamma) \Leftrightarrow (\alpha \vee \beta) \,\&\, (\alpha \vee \gamma)$

Commutativity (Com)

$\alpha \vee \beta \Leftrightarrow \beta \vee \alpha$
$\alpha \,\&\, \beta \Leftrightarrow \beta \,\&\, \alpha$

Double Negation (DN)

$\alpha \Leftrightarrow \neg\neg\alpha$

Contraposition (Cont)

$\alpha \supset \beta \Leftrightarrow \neg\beta \supset \neg\alpha$

Material Implication (Imp)

$\alpha \supset \beta \Leftrightarrow \neg\alpha \vee \beta$

Material Equivalence (ME)

$\alpha \equiv \beta \Leftrightarrow (\alpha \supset \beta)\ \&\ (\beta \supset \alpha)$
$\alpha \equiv \beta \Leftrightarrow (\alpha\ \&\ \beta) \vee (\neg\alpha\ \&\ \neg\beta)$

Exportation (Exp)

$\alpha \supset (\beta \supset \gamma) \Leftrightarrow (\alpha\ \&\ \beta) \supset \gamma$

Tautology (Tau)

$\alpha \Leftrightarrow \alpha\ \&\ \alpha$
$\alpha \Leftrightarrow \alpha \vee \alpha$

Biconditional De Morgan's (BDM)

$\neg(\alpha \equiv \beta) \Leftrightarrow \neg\alpha \equiv \beta$

Biconditional Commutativity (BCom)

$\alpha \equiv \beta \Leftrightarrow \beta \equiv \alpha$

Biconditional Inversion (BI)

$$\alpha \equiv \beta \Leftrightarrow \neg\alpha \equiv \neg\beta$$

Biconditional Association (BAss)

$$\alpha \equiv (\beta \equiv \gamma) \Leftrightarrow (\alpha \equiv \beta) \equiv \gamma$$

Chapter 7

Predicate Logic: Syntax

In the last few chapters we developed rules for the syntax and semantics of the language *SL*. We learned how to use two powerful methods, truth-tables and truth-trees, for testing sentences and sets of sentences to analyze the logical properties that we defined for the language. Finally, we developed a system that enables us to derive new sentences of *SL* from given sentences of *SL*.

An important feature of *SL* is that it is *decidable*. Truth-tables and truth-trees are mechanical methods for verifying truth-functional properties like validity, consistency, etc., in the sense that each step of the method is determined by a rule and the previous steps. To say that *SL* is decidable is to say that, for questions about truth-functional concepts, our methods will always give us a definite yes or no answer in a finite number of steps. So, we can, simply by following a set of rules, determine the validity of *any* argument in sentential logic. There will never be an instance for which the validity of an argument in *SL* cannot be determined.

Sentential logic is a powerful tool, but there are times when it fails. Consider this argument:

1. All dogs are mammals.
2. All cats are mammals.
3. Either Lola is a dog or Lola is a cat.

∴ Lola is a mammal.

Symbolizing this argument in *SL* results in something like this:

1. D
2. C
3. $\underline{\text{E} \vee \text{F}}$

$\therefore$ G

The natural language argument above is obviously valid, but the corresponding argument in *SL* is not. The problem is that *SL* cannot capture the logical relations between "All dogs are mammals", "Lola is a dog", and "Lola is a mammal." Those relations are determined by the internal structures of the sentences, and the internal structures of sentences are invisible to *SL*, because the smallest logical unit in *SL* is an entire sentence.

In this chapter, we will begin to develop a new language, *PL* (for predicate logic), that will allow us to express some of the internal structures of sentences. The argument above has a valid symbolization in *PL*. Unfortunately, this power comes with a cost. *PL* is not a decidable system. There is no method that we can use that is guaranteed to always tell us if an argument in *PL* is valid or a set of sentences in *PL* is consistent.

7.1 Singular Terms and Predicates

Predicate logic identifies three important components of sentences in natural languages: individual constants, predicates, and quantity terms. Individual constants are a type of singular term. A singular term is one that refers to (or designates or denotes), or purports to refer to, some particular object, called the referent of the term. There are two kinds of singular terms, proper names and definite descriptions. Examples of proper names are 'Aristotle' and 'Grace Hopper'. Examples of definite descriptions are 'the teacher of Alexander the Great' and 'the inventor of COBOL'. The referent is often determined by context. For example, there may be several people with the name 'Grace Hopper', but in a conversation about the history of computing, it's clear that the intended referent is Grace Hopper, the inventor of COBOL.

The referent of a pronoun is also determined by context. In the sentence, "Grace Hopper was a naval officer, and she invented COBOL", 'Grace Hopper' and 'she' refer to the same person. In this case, 'she' is being used as a singular term. There are some tricky cases, though. In the sentence

> If someone wins the lottery, then he will be very happy.

Here, 'he' isn't used as a singular term. Substituting some name for 'he' results in a non-equivalent sentence:

If someone wins the lottery, then Joe Biden will be very happy.

Some sentences can contain more than one singular term. For instance,

Oklahoma City is north of Dallas.

We get predicates by deleting one or more singular terms from a sentence. Form this sentence, we can get the predicates

_____ is north of Dallas

and

_____ is north of _____.

So, think of predicates as strings of words with blanks such that when the blanks are filled in with singular terms, we a complete sentences. A predicate with just one blank is a one-place, or monadic, predicate. A predicate with two blanks is a two-place, or dyadic, predicate, and so on. Predicates with more than one place are collectively called polyadic predicates. Instead of using blanks, we'll use the lowercase letters 'w', 'x', 'y', and 'z'.

Singular terms and predicates allow us to say things like "Josh is present" and "Sarah doesn't like root canals." With these alone, though, we can't say simple things like "everyone is present" and "no one likes root canals." For statements like these, we also need quantity terms. These are terms like 'all', 'some', and 'none'. It should be clear that quantity terms are not singular terms. 'Everyone' does not refer to a particular thing. Instead, the sentence 'everyone is present' means that the predicate 'is present' is true of everyone, and what 'everyone' means will of course be determined by context.

7.2 PL: Singular Terms and Predicates

The formal language *PL* contains the sentential connectives of *SL*, along with components that correspond to the singular terms, predicates, and quantity terms of English and other natural languages. For singular terms, *PL* uses

individual constants, which are lowercase letters of the Roman alphabet 'a' through 'v', with subscripts if necessary. The predicates of *PL* are uppercase letters of the Roman alphabet. As in English, predicates in *PL* have blanks that are filled in with individual constants to form sentences. A sentence is formed by writing the predicate letter first, followed by the singular terms in the order given by the translation key. A translation key will specify a universe of discourse (UD), which is the set of things that we're talking about. It will also assign individual constants to some, or all, of the members of the universe of discourse. Finally, it will assign predicate letters to the relevant natural language predicates. Here is an example of a translation key:

UD: The Smith family pets

Cx: x is a cat

Dx: x is a dog

Lxy: x likes y

a: Abby

b: Bailey

c: Cookie

Using this translation key, we can translate these English sentences

1. Abby is a cat.
2. Bailey and Cookie are dogs.
3. Bailey likes Abby.
4. If Abby is a cat, then Cookie doesn't like Abby.

like this in *PL*:

1. Ca
2. Db & Da
3. Lba
4. Ca ⊃ ¬Lca

We could also use the same translation key to go from *PL* to English. Translating these sentences in *PL*

1. Bc

2. Bd ∨ ¬Lcb
3. Lab & ¬Laa
4. (Lab & Lac ⊃)¬(Db ∨ Dc)

like this into English:

1. Either Bailey is a dog or Cookie doesn't like Bailey.
2. Abby likes Bailey, but doesn't like herself.
3. Abby likes both Bailey and Cookie only if neither Bailey nor Cookie are dogs.

7.3 Quantifiers

When we establish a universe of discourse, we are essentially pretending, for the sake of discussion, that those are the only things that exist. Using the above translation key, we could translate

Bailey likes everything

as

(Lba & Lbc) & Lbb

Note that since Bailey is one of the three things that are in the universe of discourse, then saying that Bailey likes everything is to imply that Bailey likes himself. Likewise, we could translate

Bailey likes something

as

(Lba ∨ Lbc) ∨ Lbb

This is only practical, however, when the universe of discourse is *very* small. Even with a small universe of discourse, things can quickly get out of hand. For example, to translate

Something likes something

we would have to do something like this:

{[(Laa ∨ Lab) ∨ (Lac ∨ Lba)] ∨ [(Lbb ∨ Lbc) ∨ (Lca ∨ Lcb)]} ∨ Lcc

Imagine what translating 'Someone likes someone else' would require if the universe of discourse were all of the students at the university!

This is why we need something analogous to the quantity terms that was discussed earlier. For this, we will use the quantifier symbols '$\forall$', usually read as "for all", and '$\exists$', usually read as "there exists" or "there is". A quantifier consists of one of these symbols followed by a variable, which in *PL* are the lowercase Roman letters 'w' through 'z'.

Let's us the same translation key that we used in the preceding section and look at a few examples. To say that everything in the universe of discourse is a dog, we simply write

$\forall xDx$

This is how we would symbolize 'There is at least one cat':

$\exists xCx$

We can symbolize 'Bailey likes everything' as

$\forall xLbx$

and 'Bailey likes something' would be

$\exists xLbx$

We can think of quantifiers as providing a way of interpreting variables in formulas of *PL*. In '$\forall xDx$', the universal quantifier tells us that the variable in "Dx" stands for every object in the universe of discourse, while an existential quantifier would tell us that the variable stands for at least one object in the universe of discourse.

Quantifiers are like negations in two important senses. First, remember that a negation operates only on the sentence that immediately follows it. In '$\neg P \supset Q$', the tilde negates only the sentence 'P', while in '$\neg(P \supset Q)$', the tilde negates the entire conditional '$P \supset Q$'. In the same way, quantifiers interpret only the formulas that they immediately precede. In the sentence

$\forall xDx \,\&\, Cx$

the universal quantifier only provides an interpretation of the formula 'Dx', but in

$\forall x(Dx \,\&\, Cx)$

it allows us to interpret both variables in the conjunction 'Dx & Cx'.

Second, just like '$\neg(P \vee Q)$' is a negation, while '$\neg P \vee \neg Q$' is a disjunction of two negations, '$\forall x(Dx \vee Cx)$' is a universal quantification, while '$\forall xDx \vee \forall xCx$' is a disjunction of two universal quantifications.

We've reserved the letters 'w' through 'z' for variables. For the most part, which letter you choose to use does not matter. For example,

$$\forall xLxb$$

and

$$\forall yLyb$$

are the same sentence. They are both perfectly good symbolizations of 'Everything likes Bailey'. Although we used 'x' for the translation key, we don't have to use the same variable when doing the actual translations. Note that we do have to use the same variable for the quantifier and the quantified formula. For example, this isn't allowed:

$$\forall xDy$$

The 'y' isn't in the scope of any quantifier. There are times when it's helpful, although not necessary, to use different variables. For example, here's a translation of 'There is at least one dog, and there is at least one cat.'

$$\exists xDx \ \& \ \exists xCx$$

Although, this is a perfectly good translation, using different variables would help to prevent the sentence from being read as this:

$$\exists x(Dx \ \& \ Cx) \text{ (Mistake!)}$$

I'll leave it up to you to decide what would be wrong with that. So, here's a good way to translate the sentence:

$$\exists xDx \ \& \ \exists yCy$$

Which quantifier needs to be used is usually fairly easy to determine. Words like 'some', 'something', or 'someone' will use the existential quantifier, while words like 'each', 'every', 'any', and 'all' will ordinarily use the universal quantifier. There are times when this can get a bit tricky, however. For example, 'Bailey likes everything' and 'Bailey likes anything' mean the same thing, and are both translated this way.

$$\forall x Lbx$$

Notice the difference between 'Bailey doesn't like everything' and 'Bailey doesn't like anything', however. Those sentence don't mean the same thing, because they have different truth conditions. The first is false only if Bailey likes everything, while the second only requires that there be one thing that Bailey likes to be false. So, we can translate the first as a negated universal quantification and the second as a negated existential quantification.

Actually, anything that we can translate with a existential quantifier could be translated with a universal quantifier, and vice versa. That is, we only really need one quantifier. Remember that we didn't really need both of '&' and '$\vee$' because

$$A \,\&\, B$$

is equivalent to

$$\neg(\neg A \vee \neg B)$$

In the same way, 'Everything is F' has exactly the same truth conditions as 'It's not the case that there is something that is not F.' So,

$$\forall x Fx$$

is equivalent to

$$\neg\exists x \neg Fx$$

Just keep in mind that as a negation is moved across a quantifier, the quantifier changes.

7.4 Formal Syntax of PL

Now that we have a basic understanding of the features of *PL*, we're in a better position to understand its syntax. The vocabulary of *PL* consists of:

Predicates: Capital Roman letters with or without positive integer subscripts followed by zero or more primes. (Exactly *n* primes indicate an *n*-place predicate. A zero-place predicate is equivalent to a sentence letter of *SL*.)

Individual constants: Lowercase Roman letters 'a' through 'v' with or without positive integer subscripts.

Individual variables: Lowercase Roman letters 'w' through 'z' with or without positive integer subscripts.

Individual term: An individual constant or individual variable.

Truth-functional connectives: $\neg$ $\&$ $\vee$ $\supset$ $\equiv$

Quantifier symbols: $\forall$ $\exists$

Punctuation marks: ( )

In practice, we will omit the primes from the predicate letters. Adding zero-place predicates means that every sentence that is symbolized in *SL* can also count as a symbolization in *PL*.[1]

Expression of *PL*: a sequence of elements of the vocabulary of *PL*.

Quantifier of *PL*: An expression of *PL* of the form $\forall$x (a universal quantifier) or $\exists$x (an existential quantifier). A quantifier is said to *contain* a variable. For example, '$\forall$x' contains the variable 'x' and '$\exists$z' contains the variable 'z'. We'll call a quantifier containing 'x' an 'x quantifier', a quantifier containing 'y' a 'y quantifier', and so on.

Atomic formula of *PL*: An expression of *PL* that is an *n*-place predicate of *PL* followed by *n* individual terms of *PL*.

[1] In this course, however, we won't be using sentence letters in *PL*.

Now, we can give a recursive definition of 'formula of *PL*', using ϕ, ψ, and χ as metavariables for expressions of *PL* and α as a metavariable for individual variables of *PL*:

1. Every atomic formula of *PL* is a formula of *PL*.
2. If ϕ is a formula of *PL*, then so is $\neg\phi$.
3. If ϕ and ψ are formulas of *PL*, then so are $(\phi \,\&\, \psi)$, $(\phi \vee \psi)$, $(\phi \supset \psi)$, and $(\phi \equiv \psi)$.
4. If ϕ is a formula of *PL* that contains at least one occurrence of α and no α-quantifier, then $\forall\alpha\phi$ and $\exists\alpha\phi$ are both formulas or *PL*.
5. Nothing is a formula of *PL* unless it can be formed by repeated applications of 1–4.

Logical operator of *PL*: An expression of *PL* that is either a quantifier or a truth-functional connective.

Now, we need the concept of a subformula and a main logical operator:

1. If ϕ is an atomic formula of *PL*, then ϕ contains no logical operator, and ϕ is the only subformula of ϕ.
2. If ϕ is a formula of *PL* of the form $\neg\psi$, then $\neg$ is the main logical operator of ϕ and ψ is the immediate subformula of ϕ.
3. If ϕ is a formula of *PL* of the form $(\psi\&\chi)$, $(\psi\vee\chi)$, $(\psi \supset \chi)$, or $(\psi \equiv \chi)$ then the binary connective between ψ and χ is the main logical operator of ϕ and ψ and χ are the immediate subformulas of ϕ.
4. If ϕ is a formula of *PL* of the form $\forall\alpha\psi$ or $\exists\alpha\psi$, then the quantifier that occurs before ψ is the main logical operator of ϕ and ψ is the immediate subformula of ϕ.
5. If ϕ is a formula of *PL*, then every subformula of a subformula of ϕ is a subformula of ϕ, and ϕ is a subformula of itself.

Scope of a quantifier: The scope of a quantifier in a formula ϕ of *PL* is the subformula ψ of which that quantifier is the main logical operator.

Bound variable: An occurrence of a variable α in a formula ϕ of *PL* is bound if and only if that occurrence is within the scope of an α-variable.

Free variable: An occurrence of a variable α in a formula ϕ of *PL* is free if and only if it is not bound.

Sentence of *PL*: A formula ϕ of *PL* is a sentence of *PL* if and only if contains no free variables.

7.5 PL: Translations

In 2, we encountered the A, E, I, and O statements of categorical logic:

A: All S are P.

E: No S are P.

I: Some S are P.

O: Some S are not P.

We are able to express all of these claims, and more, in *PL*. Let's look at some examples using this translations scheme:

UD: living things

Sx: x is a snake.

Rx: x is a reptile.

Px: x is poisonous.

Wx: x is warm-blooded.

Now, consider the sentence:

All snakes are reptiles.

How should we translate that in *PL*? An initial inclination might be to translate it this way:

$\forall xSx \ \& \ \forall xRx$

This, though, is clearly wrong, since it means that everything in the universe of discourse is a snake *and* everything in the universe of discourse is a reptile. That would be true only if every living thing were a snake. We can begin to translate it correctly by first paraphrasing it in a way that captures the truth conditions of the sentence. Basically, it is saying, for each living thing, if that thing is snake, then it is also a reptile. So, an equivalent sentence of *PL* is

$\forall x(Sx \supset Rx)$

The sentence 'No snakes are warm-blooded' asserts that everything in the universe of discourse that is a snake is not warm-blooded, so this is an appropriate translation:

$\forall x(Sx \supset \neg Wx)$

Of course, when one says that no snakes are warm-blooded, one is essentially denying that there are any warm-blooded snakes. So, another perfectly good translation would be this:

$\neg\exists x(Sx \mathbin{\&} Wx)$

The claim that some snakes are poisonous is not a claim about everything, so we shouldn't use the universal quantifier. 'Some snakes are poisonous' is true just in case there is at least one thing in the universe of discourse that is both a snake and is poisonous. So, this I sentence is naturally translated as:

$\exists x(Sx \mathbin{\&} Px)$

O sentences are translated the same way, with the addition of a negation. 'Some snakes are not poisonous' is

$\exists x(Sx \mathbin{\&} \neg Px)$

Keep in mind that for sentences of more than the simplest degree of complexity, there will always be more than one acceptable translation. Any universal quantification can be rephrased as a negated existential quantification, every existential quantification is equivalent to a negated universal quantification, every conjunction can be expressed as the negation of a disjunction, and so on. Even so, there is a natural way to translate the categorical statements, using the universal quantifier for the universal A and E sentences, and an existential quantifier for the particular I and O sentences. If ϕ and ψ are formulas with the variable x, then

A: $\forall x(\phi \supset \psi)$

E: $\forall x(\phi \supset \neg\psi)$

I: $\exists x(\phi \mathbin{\&} \psi)$

O: $\exists x(\phi \mathbin{\&} \neg\psi)$

Let's practice with some sentences using this translation scheme:

UD: All birds

Fx: x can fly

Rx: x is a raptor

Hx: x is a hawk

Px: x is a penguin

Sx: x can swim

Vx: x is a vertebrate

Wx: x is warm-blooded

1. All raptors can fly.
2. No penguins can fly.
3. Some raptors are hawks.
4. Penguins can swim but can't fly.
5. If any penguin swims, then all raptors are warm-blooded.
6. If any penguin swims, then it is warm blooded.
7. All birds are vertebrates.

The first three are simple:

1. $\forall x(Rx \supset Fx)$
2. $\forall x(Px \supset \neg Fx)$
3. $\exists x(Rx \,\&\, Hx)$

One way to understand the fourth would be as a conjunction: 'All penguins swim and no penguins fly.' So, the translation could be

4. $\forall x(Px \supset Sx) \,\&\, \forall y(Py \supset \neg Fy)$

Another way to paraphrase this would 'All penguins both swim and not fly'. This means that the third sentence can also be translated this way:

4. $\forall x[Px \supset (Sx \,\&\, \neg Fx)]$

This simpler translation should be preferred.

The fifth sentence says that if there is a penguin that swims, then all raptors are warm-blooded. This is a conditional sentence that has a existential quantification for an antecedent and a universal quantification for a consequent.

5. $\exists x(Px \,\&\, Sx) \supset \forall y(Rx \supset Wx)$

The antecedent and the consequent of the sixth sentence share the same subject. It says that for any penguin, if that penguin swims then it is warm-blooded. This means the same as saying that all penguins that swim are warm blooded. So, we should translate it like this:

6. $\forall x[(Px \,\&\, Sx) \supset Wx]$

The key difference between the fifth and sixth sentences is that the fifth contains two quantity terms, 'any' and 'all', and the sixth has only one quantity term. So, the fifth sentence should require two quantifiers, but only one quantifier for the sixth.

Since the universe of discourse is all birds, the seventh simply says that everything is a vertebrate, or

7. $\forall xVx$

Had the universe of discourse been broader, then it would need to be translated something like

$$\forall x(Bx \supset Vx)$$

For the rest of the examples in this section, we will pick predicate letters and individual constants that make sense for the sentence to be translated.

PL can be used to translate much more complex sentences than just the four standard sentence types in categorical logic.

Assuming a universe of discourse as all people, then

> Everyone who likes Alfred also likes Beth and Carlos.

This is paraphrased as, 'For all people, if a person likes Alfred, then they also like Beth and like Carlos.' It's then translated as

$\forall x[Lxa \supset (Lxb \ \& \ Lxc)]$

Now consider

Beth likes anything that Alfred and Carlos like.

This can be paraphrased as "Everything is such that, if both Alfred and Carlos like it, then Beth does too." So, a natural translation would be

$\forall x[(Lax \ \& \ Lcx) \supset Lbx]$

Here are some tips for translating some tricky sentences. We know how to translate a sentence like this:

Some dogs are Labradors.

In *PL*, it is

$\exists x(Dx \ \& \ Lx)$

Sometimes, though, we will squeeze two predicates together so that one modifies the other. For example,

Some black dogs are Labradors.

The general rule is that, in cases like this, split the predicates to form two open sentences, like this:

$Bx \ \& \ Dx$

So, the correct translation of the sentence above is

$\exists x[(Bx \ \& \ Dx) \ \& \ Lx)]$

To translate

Some intelligent students are lazy

we would do this:

$\exists x[(Ix \ \& \ Sx) \ \& \ Lx]$

There are exceptions to the general rule, though. Don't split the predicates when doing so would result in a false sentence. Something can be a small elephant, but not be uncategorically small. So, treat phrases like 'short skyscraper', and 'large mouse' like a single predicate.

Unless otherwise stated, we will assume a domain of discourse that is the set of all existing things. So, when a sentence is making a claim only about certain kinds of things, we need to be careful to make that clear. Sometimes, we really are talking about anything:

Something here is not right: $\exists x(Hx \,\&\, \neg Rx)$

Other times, we intend to be referring only to people. In those cases, simply add another conjunction.

Someone here is not right

is translated

$\exists x[(Px \,\&\, Hx) \,\&\, \neg Rx]$

'Only' always signifies a universal, but translating it can be tricky. As a general rule, the word 'only' introduces the consequent of a conditional.

All emeralds are green things.

is

$\forall x(Ex \supset Gx)$

If we instead say

Only green things are emeralds

we would translate it the same way:

$\forall x(Ex \supset Gx)$

Here is the rule to remember:
All A's are B's: $\forall x(Ax \supset Bx)$
Only A's are B's: $\forall x(Bx \supset Ax)$
That's simple enough. Be especially careful when more than one predicate follow the 'only'. For example,

Only intelligent students are good logicians.

is *not*

$\forall x[Gx \supset (Ix \ \& \ Sx)]$ **NO!**

This would mean that if someone is a good logician, then they are both intelligent and a student. I may be willing to grant that they are intelligent, but surely there are some good logicians who are not students. Follow this as a general rule:

'Only PQ's are R' is equivalent to 'All RQ's are P.'

So, 'Only intelligent students are good logicians.' is translated

$\forall x[Gx \ \& \ Sx \supset Ix]$

7.6 Multiple Quantifiers

7.7 Identity

Bibliography

Cleese, John, and Graham Chapman. 1980. “The Argument Clinic.” In *Monty Python Live at the Hollywood Bowl.* Columbia Pictures.