Symbolic Logic

John W. Snow

Contents

Preface

These are notes for a course in Mathematical Logic for Liberal Arts Students. The notes are designed for students with little to no mathematical background. In making these notes I had these specific goals in mind:

- The class should be a math class involving theorems and proofs.
- The class should expose students to the process of doing mathematics and the manner in which Logic in particular has developed over time.
- The class should address these ideas: formalization of language, sentential and propositional logic, logical equivalence, implication, proof, completeness, soundness, computability, Gödel's Incompleteness Theorems, cardinality, the Halting Problem.
- The class should communicate that Logic is interesting from mathematical, utilitarian, aesthetic, and philosophical perspectives.
- The class should be accessible to motivated undergraduates who are unaware of the definition of "function."

I considered many texts which are already in print for this class. The reason that these notes exist is that the texts I previewed all fell into one of these categories:

- **"Real" Mathematical Logic Texts:** There are a variety of good Logic books for mathematics majors. I needed a book which assumed no background.
- **Handbooks on Proof:** There are several books that do a good job of describing how to use rules of inference. Some of these

even discuss completeness and soundness in understandable ways. None of the books I considered in this category covered all of the material I wanted in the class.

- **General Audience Books on "How to Think":** There are some very good survey books on Logic. I found them to be largely non-mathematical discussions of mathematical topics.

The mathematician who reads these notes will probably find them inadequate at best, but they are appropriate for my intended audience (in my humble opinion). Some statements in the notes have been weakened from their full mathematical strength. Others are necessarily somewhat imprecise due to the target audience. Proofs of theorems are largely outlined or discussed in vague details.

I decided that I would begin the study of Logic by addressing sentential logic in some detail. The processes we use in sentential logic are then repeated in much less detail for first order predicate logic. The discussion becomes less precise as the topics become more involved.

As with most texts, there is nothing original mathematically in these notes. Most of this material can be found in most textbooks on mathematical logic (though usually at at higher level.) Some specific suggestions for further reading include:

- Barwise, Jon, and H. Jerome Keisler. *Handbook Of Mathematical Logic.* Amsterdam: North-Holland Pub. Co., 1977.

- Chang, Chen Chung, and H. Jerome Keisler. *Model Theory.* Amsterdam: North-Holland Pub. Co., 1973.

- Enderton, Herbert B. *A Mathematical Introduction to Logic.* San Diego, CA: Academic Press, 2001.

- Halmos, Paul R., and Steven R. Givant. *Logic As Algebra.* Washington, D.C.: Mathematical Association of America, 1998.

- Lyndon, Roger C. *Notes On Logic.* Princeton, NJ: Van Nostrand, 1966.

- Mates, Benson. *Elementary Logic.* New York: Oxford University Press, 1965.

- Tarski, Alfred. *Introduction To Logic and to the Methodology of Deductive Sciences.* New York: Oxford University Press, 1965.

Chapter 1

Introduction

Logic is the study of how we reason and draw conclusions. At the heart of logic are the questions: What is true? How does one truth follow from another? How can we decide what is or is not true? To address questions like these, we first need to address what types of *things* can be true. We assign the quality of being true or false to sentences or statements. Our natural language has rules which govern what is or is not a grammatically correct sentence. Among these grammatically correct sentences, some can be given a truth value while others cannot. Among those that have truth values, some are true while others are false.

The rules governing natural language are complex and ever-changing. To study logic and how we think, mathematicians create formal languages governed by clearer and more explicit rules. These formal languages are a shadow of the natural language and can be used to describe how we think and communicate in the natural language.

In the sections that follow, we will use a variety of symbols to approximate and study language and truth. We will let capital letters such as P, Q, A, and B represent declarative sentences which may have truth values. We will also use symbols for special words which are used in English to form compound sentences:

symbol	meaning
$\vee$	or
$\wedge$	and
$\neg$	not
$\rightarrow$	if...then..

These symbols are called logical connectives. We will use Greek letters

such as α, β, and γ to represent compound sentences formed by combining our simple sentence symbols with these logical connectives. We use these sentences to build formal languages in which we can analyze how to reason. Within these languages, we will address questions such as:

- How do the logical connectives affect truth?
- How do the truth values of simple sentences affect the truth value of a compound sentence in which they appear.
- How can a collection of sentences which are truth force another sentence to be true?
- How can we determine and demonstrate mechanically that the truth of a collection of sentences forces another sentence to be true.
- Can we mechanically determine what is and what is not true?

In the process of this analysis, we will sometimes view our formal languages from within – as if we live in a small, precise world in which all reasoning is absolutely rigorous. We will also sometimes view the languages from without. In this case, we will discuss the formal language in our less precise natural language. Sometimes, we will be able to prove things about a language from within the language. Sometimes, we will be able to give convincing arguments in our natural language for properties of the formal language. The tools we develop here for proof and verification will be quite powerful in many respects, but they are limited. We will discuss some of their mathematical limitations at the end of the course in the sections on Incompleteness and Computability.

The first part of these notes will work in the realms of *sentential logic*. Sentential logic provides very restrictive, precise formal languages where we can study reasoning quite thoroughly. This precision comes at a price – the descriptive power of sentential logic (how easy it is to communicate and what can be described) appears quite limited compared to our natural languages. The basic building blocks of sentential logic are whole sentences. These are put together with the words and, or, not, and implies.

In the second part of these notes, we move up from sentential logic to *predicate logic*. Predicate logic expands the tools of sentential logic

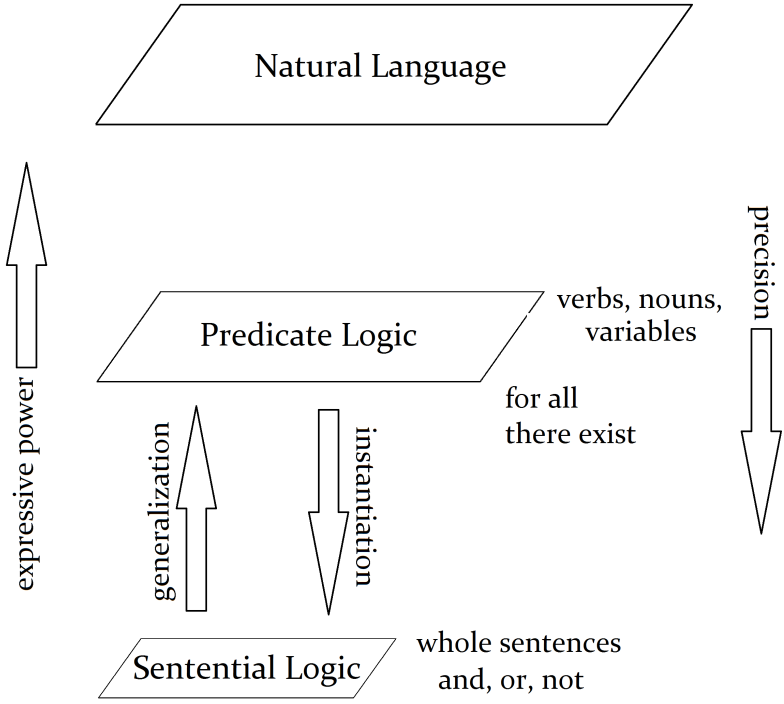


Figure 1.1: Formal languages are used to study how we reason in natural language. Sentential languages offer precision at a compromise of expressive power. Natural language offers a great deal of expressive power but can be terribly vague.

to include mechanisms that mimic verbs and nouns. It also adds quantifiers to act as the words "for all" and "there exist." Predicate logic provides languages with (apparently) more descriptive power which are more complex than sentential languages. We will have tools called instantiation and generalization that allow us to move between predicate and sentential logic when reasoning. All deduction (drawing of conclusions) and most computation of truth values actually occur in sentential logic. Many formal descriptions of mathematical ideas occur in predicate logic. Discussions about these ideas and conclusions occur in an environment closer to our natural language. These levels of language are depicted in Figure 1.1.

1.1 Examples for Thought

We begin by considering a handful of logical statements and arguments to whet your appetite.

The Liar's Paradox

The Liar's Paradox was formally introduced by Eubulides in the fourth century BC. If a man says, "I am lying to you," then what are we to think? If we think he is telling the truth, then that means he is lying. But then he cannot be telling the truth. If we think that he is not telling the truth, then what he says is says is false. This means that he is not lying and is in fact telling the truth. This problem has been a fascination of logicians for almost two and a half millenia.

Eubulides is not the first author to record statements like the Liar's Paradox. Around the sixth century BC, Epimenedes wrote that, "Cretans are always liars." The Apostle Paul calls attention to this in Titus 1:12. The odd thing here is that Epimenedes was a Cretan. Had Epimenedes said, "All Cretans always lie," we would have a clear version of the Liar's Paradox.

About the time of Epimenedes, the Psalmist wrote in Psalm 116:11, "All men are liars." This again comes close to the Liar's Paradox.

The Resurrection of the Dead

The next argument we consider is offered by Jesus in Mark 12 as an argument in favor of the resurrection of the dead.

> And as for the dead being raised, have you not read in the book of Moses, in the passage about the bush, how God spoke to him, saying, 'I am the God of Abraham, and the God of Isaac, and the God of Jacob'? He is not God of the dead, but of the living. You are quite wrong." (Mark 12:26-27, ESV)

Apparently, because God is the God of Abraham, Abraham must be alive at the time this is spoken. However, Abraham had previously died (Genesis 5:28). Thus he must have been raised from the dead.

Marcion

The second century Christian heretic Marcion believed the God of the Old Testament and the God of the New Testament were two

different Gods. Here is an argument to support this stance[1]:

> The God of the New Testament cannot be the God of the Old Testament. In John 1:18, Jesus says, "No one has ever seen God." However, according to Exodus, Moses and others did see God:
>
> > Then Moses and Aaron, Nadab, and Abihu, and seventy of the elders of Israel went up, and they saw the God of Israel. There was under his feet as it were a pavement of sapphire stone, like the very heaven for clearness. (Exodus 24:9-10, ESV)

Is there a God?

Our next argument is an argument for the existence of God. Variants of this argument are known as the ontological (existence) argument. The argument goes something like this: God is that beyond which nothing greater can be imagined. If God did not exist, then we could imagine another God, God 2.0, that did exist. This imagined God 2.0 would then be even greater than God (now rendered to the obsolete designation God 1.0). But then we have imagined God 2.0 to be greater than that beyond which nothing greater can be imagined. Thus, we would have a contradiction if God did not exist.

Omniscience

Next, we address a proof that God does not exist. We prove that no one can be omniscient. First, we prove that Tyler is not omniscient by giving an example of a sentence that Tyler cannot correctly know to be true or false. The sentence is

> Tyler knows that this sentence is false.

The "this sentence" refers to the sentence itself.

Suppose that Tyler correctly knows that this sentence is true. Since he is correct, the sentence is true, and we should believe what it says - that Tyler knows it is false. This is a contradiction since we assumed that Tyler knows the sentence is true. This contradiction implies that Tyler cannot correctly know the sentence to be true.

[1]This was probably not Marcion's argument since it is based on John. Marcion used Luke.

Suppose that Tyler correctly knows the sentence to be false. Since Tyler is correct, the sentence must be false. But then what the sentence says is false - Tyler does not know that the sentence is false. This is again a contradiction. Tyler cannot know this sentence to be false.

Since Tyler cannot correctly know whether this sentence is true or false, Tyler is not omniscient. There is nothing special about the name "Tyler." The same argument works with the insertion of any name. Thus God is not omniscient, or there cannot be an omniscient God. Therefore, if omniscience is one of our defining features of God, there can be no God.

Proof and Truth

Finally, consider this sentence:

> This sentence cannot be proven true.

The "this" in the sentence refers to the sentence itself. Now, suppose that this sentence is false. Then, it cannot be that the sentence cannot be proven true, so it must be that the sentence *can* be proven true. However, this means that the sentence is both true and false – which simply cannot be. We conclude then that the sentence must be true. Thus, we have a sentence which is true but (according to the sentence itself) cannot be proven. But wait! Did we just prove the sentence to be true?

Chapter 2

An Incomplete History

This section should probably be skimmed when one first reads these notes and then be read in more detail after one has thoroughly read through all of the notes. We do not pretend the history listed below is exhaustive. We only intend to provide an outline to mention some of the main figures and concepts that have helped to mold mathematical logic. We note that this is intended to be a brief summary of the history of *mathematical* logic. The realms of mathematics, science, philosophy, and logic are at times difficult to separate historically.

The first written discussions of logic seem to be from the *Dissoi Logoi* (Double Arguments) from around 400 B.C. This writing includes a debate over whether truth is a temporal or atemporal quality. It also includes discussions that witness to the awareness of complications involved with self-referential statements. This issue with self-referential statements is a recurring theme throughout the history of logic.

Zeno of Elea (b. 490 B.C.) and Socrates (470-399 B.C.) were famous for their debating abilities. Their tactics on occasion took advantage of arguments similar to reductio ad absurdum (or proof by contradiction). Zeno virtually used the rule of inference modus tollens in arguments of the form "This implies that, but not that; so not this."

Eubulides – a grand-student of Socrates – seems to be the first to introduce the Liar's Paradox. The man who says "I am lying" is neither lying nor telling the truth. If he is lying, then we should not believe him and must conclude he is telling the truth. But if he is telling the truth, then he must be lying. Self-referential sentences such as these have plagued logicians for over two thousand years. They were exploited by twentieth century mathematicians to prove deep theorems

about the power of logic and computation.

Plato (429-347 B.C.) instituted dialectic competitions – verbal debates governed by rules – at the Academy. Patrons of these competitions seemed to have at least studied the patterns of argument.

The first great logician by all accounts was Aristotle (384-322 B.C.). Aristotle summarized the tools of the dialecticians and created his own logic, which is usually called a "logic of terms" because it deals with the relationships between terms such as "man" and "white." The statements in his syllogisms or arguments were originally all universal or particular and affirmative or negative. For example:

Universal Affirmative	All men are white.
Particular Affirmative	Some men are white.
Universal Negative	Not every man is white.
Particular Negative	No man is white.

These types of statements are similar to those involving quantifiers in predicate logic in the second part of these notes. Aristotle's syllogisms contained exactly two premises and a conclusion and up to three terms such as:

> For if A is predicated of (holds for) all B and B of all C, it is necessary for A to be predicated of all C.

Here, the letters A, B, and C are variables for terms. This is the first known use of variables in mathematics or science. Aristotle also introduce modal logic. He considered that truth can be necessary, possible, contingent, or impossible – paving the way for multi-valued logic. Aristotle's logic was essentially the leading logic until the nineteenth century. His school included the Peripatetics who wrote much but basically preserved, refined, and expanded the ideas of Aristotle.

A follower of Socrates by the name of Euclid (not the geometer) began the Megarian School. One of his students was Eubulides (father of the Liar). A great-grand student was Zeno (336-264), the founder of Stoicism. While Aristotle is heralded as the pre-eminent logician by modern historians, ancient historians gave that stature to the Stoic logician Chrysippus (280-205 B.C.). Chrysippus and the Stoics invented much of the propositional/sentential logic which we will discuss in the first part of these notes. Their achievements and completeness in these respects were not matched until Frege in the nineteenth century. They developed the basic rules of inference such as Modus Ponens. A Stoic description of such a rule would read something like:

If **the first**, then **the second**;
The first;
Therefore, **the second**.

Notice the clear use of first, second, etc. as variables. These variables differ from those of Aristotle in that the Stoic variables are *complete propositions* rather than just *terms*. The Stoics also seem to have been obsessed with paradoxes such as the Liar. Chrysippus apparently wrote whole volumes on them. Chrysippus was legendary in his own time for his skills of deductions. He is reported as having written to a colleague, "Just send me the theorems; I'll find the proofs for myself." He claimed that rules of inference such as disjunctive syllogism were so natural that even dogs are aware of them. Ancient historians declared, "If there is any logic in Heaven, it is that of Chrysippus."

One of the offshoots of the Megarian school was Diodorus Cronus and his student Philo. They used truth assignments (as we will in the first part of these notes) to define logical connectives. They are credited with inventing the "material implication" – that a statement of the form "if A then B" is false exactly when A is true and B is false. This is an approach which modern mathematicians take and which was debated by early logicians.

For the next millennium, historians declare that logicians did little except to preserve the heritage of Aristotle and Chrysippus. This is the first "Dark Age" of logic (if the time before Aristotle is to be ignored). The next distinguished logician came in the form of indomitable debater and scholar Peter Abelard (1079-1142 A.D.). Abelard created his own system of propositional logic based on truth assignments complete with rules of inference, logical implication, and a discussion of the negation of quantifiers. In his logic, he focused much on the word "is." Many (most, all) statements which he considered could be phrased as "A is B." This is similar to Aristotle's term relationships, but is closer to predicate logic and set theory which would not take the stage until the nineteenth century. Abelard is perhaps most well known for his method of presenting an idea through a series of questions and answers both for and against the idea (*Sic et Non*).

The most distinguished logicians of the fourteenth century were William of Ockham (1295-1349), Jean Buridan (1300-1358), and an anonymous writer called Pseudo Scotus. These writers developed a theory of *consequentiae* (basic arguments or rules of inference). A consequentia was defined to be sound if and only if it was not possible

for the antecedent to be true and the the consequent to be false. The writing of these logicians masked the mathematical form of their rules of inference. A simple rule of inference which employs variables could be expressed as "P implies P or Q." The same rule for these fourteenth century logicians may read

> There is a sound consequentia from either part of an affirmative disjunction to the affirmative disjunction of which it is a part.

The Renaissance in the west and the rediscovery of ancient Greek writings ironically ushered in a second dark age for logic. The clarity of the the ancients was such that Renaissance scholars saw the works of "modern" logicians as inferior. With the exception of Leibniz, there seems not to have been a truly innovative logician for four hundred years.

Gottfried Wilhelm von Leibniz (1647-1716 A.D.) was one of the founding fathers of the Calculus. At the heart of twenty-first century culture is technology. Technology is built upon science – largely physics. Science and physics are dependent upon Calculus. With this in mind, it is safe to say that Leibniz has made unmeasurable contributions to the world today. His work in logic is at a tangent to these magnificent contributions. As a teenager, Leibniz envisioned an artificial language which could mirror thought and the process of implication. This language could streamline communication, derivation, and proof. After building an adding machine, Leibniz imagined a machine which would use his language and mechanical rules to perform arithmetic with propositions to separate mechanically between truth and falseness. Leibniz worked on developing his language, but it never fully emerged. Later mathematicians would take up his concept of a logical calculating machine. This, like the Liar's Paradox, is a theme which, once introduced, would remain.

The modern era of logic began with George Boole (1815-1864 A.D.) and Augustus De Morgan (1806-1871 A.D.). These mathematicians introduced the idea that one can use logical connectives (and, or, not, if...then...) to do "arithmetic" with sentences and relations. Logic then took on a more mechanical and mathematical tone than it had enjoyed at any time in the previous two millennia.

Georg Cantor (1845-1918 A.D.), beginning with the word "is" (as Peter Abelard) formally laid out the theory of *sets* – on which all of

modern mathematics would be built. In the process, he formalized the notion of the *infinite*.

Gottlob Frege (1848-1925) can arguably be described as the greatest modern logician – perhaps the foremost logician of all-time (but for our affinity for Aristotle and that due to Chrysippus). In his *Begriffsschrift* Frege essentially laid out the concepts of modern mathematical logic. Using a formalized, axiomatic process, he developed sentential logic, quantifiers and first order predicate logic, and he laid the foundation for theory of positive integers. Moreover, he described how all of arithmetic can be reduced to logic. While Boole and DeMorgan showed us that logic *really is* mathematics, Frege showed us that mathematics (or, at least arithmetic) *really is logic*. Many of the ideas of Frege were simultaneously developed by Charles Sanders Peirce (1839-1914) in a less systematic way.

About the same time that Frege was laying down the foundations for modern mathematical logic Giuseppe Peano (1858-1932) was building off of the work Boole to construct his own system of mathematical logic. One of his goals was to lay down rigid and complete axioms from which the natural numbers (0,1,2,...) and their properties could be derived. His system – usually called the Peano Axioms – form the standard axiom system used today for the natural numbers.

The theory of sets laid out by Georg Cantor was not entirely acceptable. Bertrand Russell showed that it was susceptible to certain paradoxes which we will entertain later. Russell and Alfred North Whitehead (1861-1947) set out to remedy this situation in their *Principia Mathematica* in which they sought to fulfill Frege's program of deriving all of mathematics from logic.

Kurt Gödel (1906-1978) proved in 1930 that the deductive logic of Frege was complete – any logical implication could be proven in this logical system. This could be read as a declaration of the power of proof. However, Gödel also proved in 1931 that there are statements true in the logic of the positive integers which cannot be proven from the usual axioms. In fact, the statements true about positive integers cannot be listed or described in any "nice" way. His proofs heavily take advantage of self-reference. Related to these results, Alfred Tarski (1901-1983) proved that in sufficiently complex logical systems, truth cannot be defined or easily described.

In 1928, David Hilbert posed the *Entscheidungsproblem* (decision problem), which asks if there is an algorithm which, for any math-

ematical proposition, would decide if that proposition were true or false. Alan Turing (1912-1954) invented a theoretical computing machine which allowed him to formulate a definition of what it meant to *compute*. He was able to prove that there is no algorithm to decide if a program on one of his machines would run forever or eventually stop. This answered Hilbert's's question. Turing's solution (like Gödel's) relies on self-reference. Turing's solution to the Entscheidungsproblem came just on the heels of another solution by Alonzo Church (1903-1995). The result of Alonzo Church proved that no algorithm exists to decided the truthness of a proposition in arithmetic or in set theory.

Chapter 3

Sentential Languages

In this chapter, we discuss how to build sentences in a formal language. Sentences will begin with symbols (usually letters A, $B, \ldots$) that represent what can be thought of as simple sentences. We combine these with logical connectives to form compound sentences. After we have said how to build sentences, we will discuss how the truth values of the simple sentence symbols affect the truth value of a compound sentence.

We will use a variety of words for the idea of a sentence in this chapter. The words sentence, statement, and proposition all refer to a declarative sentence which must be either true or false and not both. We will also use the words formula and well-formed formula for the mathematical counterpart (or representation) of a sentence.

3.1 Statements

We first address the question, "What types of things can be true?" A chair cannot be true. An apple cannot be true. Truth is a quality which is assigned to sentences – and not to every sentence at that. We isolate our attention to those sentences which may be said to be true or false. A **statement** is a declarative sentence which must be either true or false and not both. Statements are also called **propositions**. Here are some examples of statements (not all of these are true):

"The grass is green."
"George W. Bush is the American president."
"The number 2 is less than the number 1."
"1+1=2"
"The sun will rise tomorrow."

Here are some examples of things which are not statements:

> "Go to bed." (an imperative or command)
> "The house on the hill" (not even a sentence)
> "Is this a statement?" (an interrogative or question)
> "Paul is tall." (Since "tall" is relative, this might seem true to some people and false to others. It is not strictly true or false.)

3.2 Exercises

Which of the following are statements?

1. The brown and white dog ran down the long winding road.
2. True is spelled t-r-u.
3. This sentence is true.
4. This sentence is neither true nor false.
5. The old white house on the lonesome hill outside of town.
6. Feed the lazy dog on the porch once every day.
7. The clock is slow.
8. The car is slow.
9. The sentence "This sentence is false" is not a statement. Explain why. Hint: What would happen if the sentence were true? What would happen if it were false?

3.3 Assumptions

There are two underlying assumptions in our definition of a statement. First, every statement must be either true or false. Second, no statement is both true and false. That any statement must be either true or false but not both is called the **law of the excluded middle**. The choice of two truth values is almost arbitrary, but it is the way we perceive the world to work. We will experiment with more truth values later in Chapters 12 and 13.

3.4 Symbols

The work of mathematicians is largely to determine what is true and what is false. The ambiguous nature of our spoken language can make this task difficult. In order to avoid ambiguity, mathematicians use symbols. As our first use of symbols we will be letting capital letters represent statements. For example, we could let the letter P be the statement "It is raining." Then whenever we see a P, we think "It is raining." Of course, there are only twenty six capital letters in our alphabet and many, many statements, so we will often use the same letter to represent different statements in different problems.

3.5 Compound Statements

A single letter representing a statement will be called an **atomic statement**. We can join atomic statements together with the words "and," "or," "not," and "implies." These words will be called **logical operators**, and the more complex statements which are formed will be called **compound statements**. To make compound statements, we will not actually use the words, but symbols for the words. This is outlined in the next few sections.

3.5.1 Conjunction

We will use the symbol $\wedge$ to mean "and." If P and Q are two statements, then $P \wedge Q$ is the new statement "P and Q." For example, if P is "It is raining," and Q is "The grass is green," then $P \wedge Q$ is "It is raining, and the grass is green." The statement $P \wedge Q$ is called the **conjunction** of the statements P and Q. The statement $P \wedge Q$ should be true when both of the statements P and Q are true. Otherwise, it should be false. We can sum this up in this **truth table**:

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

The first two columns of the table list all of the possible combinations of truth values for P and Q. The third column gives the corresponding

truth value for $P \wedge Q$. As we noted above, the only time when $P \wedge Q$ is true is when P is true and Q is true.

In the English language, there are many ways of expressing $P \wedge Q$. Any statement which communicates that both P and Q are true expresses $P \wedge Q$. If P is "It is raining," and Q is "The grass is green," then each of the following communicate $P \wedge Q$:

"It is raining, and the grass is green."
"It is raining, but the grass is green."
"It is raining; however, the grass is green."
"Even though it is raining, the grass is green."
"While it is raining, the grass is green."
"The grass is green, and it is raining."

This last one is interesting; $Q \wedge P$ seems to communicate the same thing as $P \wedge Q$.

3.5.2 Disjunction

We will use the symbol $\vee$ to mean "or." If P and Q are two statements, then the statement $P \vee Q$ is "P or Q." This is called the **disjunction** of the statements P and Q. The statement $P \vee Q$ will be true when P is true, Q is true, or both are true. This can be expressed in a truth table:

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

Again, the first two columns of the truth table list the all possible combinations of truth values for P and Q (note that these are the same as the first two columns for $\wedge$ above). The last column gives the corresponding truth value for $P \vee Q$.

3.5.3 Negation

We will use the symbol $\neg$ to mean "It is not the case that..." If P is any statement then $\neg P$ means "It is not the case that P." For example, if P is "It is raining," then $\neg P$ is "It is not the case that it is raining." In English, there are simpler ways of expressing this. The best choice may be "It is not raining." For simplicity, we will most often read $\neg P$

as "Not P." A truth table for $\neg$ is easy to draw. If P is true, then $\neg P$ should be false. If P is false, then $\neg P$ should be true:

P	$\neg P$
T	F
F	T

The statement $\neg P$ is called the **negation of** P.

3.5.4 Implication (Conditional)

We will use the symbol $\rightarrow$ to mean "implies." If P and Q are statements, then $P \rightarrow Q$ is "P implies Q." We will often read this as "If P, then Q." For example, if P is "I left my hat at home," and Q is "It will rain," then $P \rightarrow Q$ could be read as "If I left my hat at home, then it will rain." To determine the truth values for this new logical operator, it is useful to think of $P \rightarrow Q$ as a promise. Let P be the statement "You win," and let Q be the statement "We will go out to eat." Then $P \rightarrow Q$ is "If you win, then we will go out to eat." Think of this as a promise. The statement will be true when the promise is kept and false if it is broken. There is only one way in which the promise may be broken - if you win and we do not go out to eat. This is the case where P is true and Q is false. Thus if P is true and Q is false, then $P \rightarrow Q$ is false. Otherwise, the promise is not broken, so the statement should be true. Here is the truth table:

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

As with conjunction, there are many ways of expressing implication. Here are a few common ways of expressing $P \rightarrow Q$:

"If P, then Q."
"P implies Q."
"Q, if P."
"P only if Q."
"Q follows from P."
"Whenever P, Q."
"Q, whenever P."

"Not P unless Q."
"P is sufficient for Q."
"Q is necessary for P."

The part of an implication which comes before the arrow is called the **antecedent** or **hypothesis**. That which comes after the arrow is the **consequent** or **conclusion**. Thus in $B \to K$, B is the antecedent, and K is the consequent. Notice in the last two statements on our list that the antecedent is the sufficient part and the consequent is the necessary part.

3.5.5 Bi-Implication

Sometimes we may want to express both $P \to Q$ and $Q \to P$. We could write this as $(P \to Q) \wedge (Q \to P)$, but instead we introduce the special symbol $P \leftrightarrow Q$. This is called the **bi-implication** or the **biconditional** and is usually read as "P if and only if Q" or "P is necessary and sufficient for Q." Sometimes, this is abbreviated as "P iff Q."

3.6 Translations

Our logical operators can be applied repeatedly to atomic statements to form more complex compound statements. Let L be "The lights are on." Let O be "The oven is on," and let D be "The door is open." We can combine these statements with our logical operators to create a multitude of compound statements. In doing so, we will use parenthesis as punctuation to indicate order of operations.

For example, we could first form $L \wedge O$ - "The lights are on and the oven is on." Then we could negate this statement to get $\neg(L \wedge O)$ - "It is not the case that the lights are on and the oven is on." In English, the statement is not as clear as in symbols. In symbols, it is clear that the $\neg$ applies to all of $L \wedge O$. In English, it sounds as if it may just apply to the L - as in $(\neg L) \wedge O$. In order make this clear, we could translate $\neg(L \wedge O)$ as "It is not the case that both the lights are on and the oven is on." The "both...and..." act to join the two simpler statements together. Now, we could take this new statement and combine it with D with an "or" to get $(\neg(L \wedge O)) \vee D$ - "It is not the case that both the lights are on and the oven is on, or the door

is open." Notice that we placed parenthesis around the $\neg(L \wedge O)$ to communicate that the $\neg$ applies only to this part of the statement.

The best strategy for translating from symbols to words is use parenthesis to locate the smallest compound statements within a statement. Form these, then combine them with the appropriate logical operators, using parenthesis as clues to punctuation and sentence structure. At each step along the way, it may be helpful to rephrase the statements so they will be more readable in English.

For example, consider the statement $(L \vee O) \rightarrow (\neg D)$. The simplest compound statements are $L \vee O$ - "The lights are on or the oven is on" - and $\neg D$ - "It is not the case that the door is open." The first of these may sound better as "The lights or the oven are on." The second may better be written as "The door is not open," or even "The door is closed." We combine these with an implication to get "If the lights or oven are on, then the door is closed."

3.7 Order of Operations

For the most part, we will always use parenthesis to indicate order of operations in compound statements. The one exception we will make to avoid too many parenthesis is to let $\neg$ take precedence over all other operations. This means that unless a set of parenthesis is in the way, we apply all $\neg$s first. For example, rather than writing $((\neg P) \wedge (\neg Q)) \rightarrow (\neg(R \wedge S))$, we can write $(\neg P \wedge \neg Q) \rightarrow \neg(R \wedge S)$.

3.8 More Translations

Let L, O, and D be as above Here are some more examples of translations from words to symbols:

$L \wedge O$	The lights are on and the oven is on.
$\neg(L \wedge O)$	It is not the case that both the lights are on and the oven is on.
$\neg(L \wedge O) \vee D$	Either it is not the case that both the lights are on and the oven is on, or the door is open.
$(L \vee O) \rightarrow \neg D$	If the lights or oven are on, then the door is closed.

$L \to (O \vee D)$	If the lights are on then either the oven is on or the door is open.
$\neg(L \to D)$	It is not the case that if the lights are on then the door is open.
$(L \wedge D) \vee (L \wedge O)$	Either the lights are on and the door is open or the lights are on and the oven is on.
$\neg D \vee (L \to O)$	The door is closed, or if the lights are on, then the oven is on.

3.9 Exercises

Let M be "The moon is full." Let A be "The alarm is set for 4:00 AM," and let F be "Fred is going fishing in the morning." Below are compound statement using A, F, M. Translate each into words. (Try to be creative).

1. $F \vee \neg A$
2. $\neg F \wedge \neg A$
3. $\neg(F \vee A)$
4. $F \wedge (M \vee A)$
5. $A \wedge M \wedge F$
6. $(F \wedge M) \vee (F \wedge \neg M)$
7. $(M \wedge A) \to F$
8. $\neg A \vee (M \to F)$

3.10 Translating from words to symbols

We can also translate statements from words to symbols. Let S be "The sun will rise in the morning." Let C be "Candace leaves a candle in her window," and let D be "Doug passes his math test."

Example 3.1: Translate this sentence into symbols:
"If Candace leaves a candle in her window or Doug passes his math test, then the sun will rise in the morning."

We can identify the atomic statements C, D and S in the statement:

$$\text{“If } \overbrace{\text{Candace leaves a candle in her window}}^{C} \text{ or } \overbrace{\text{Doug passes his math class}}^{D} \text{ then } \overbrace{\text{the sun will rise in the morning.}}^{S}$$

We notice the "or" and the "if...then..." in the statement and can label them (notice we place the $\rightarrow$ over the "then"):

$$\text{If } \overbrace{\text{Candace leaves a candle in her window}}^{C} \overset{\vee}{\text{or}} \overbrace{\text{Doug passes his math class}}^{D} \overset{\rightarrow}{\text{then}} \overbrace{\text{the sun will rise in the morning.}}^{S}$$

Finally, we can use the structure of the sentence and punctuation to determine placement of parenthesis. This gives:

$$\text{If } \overbrace{\text{Candace leaves a candle in her window}}^{(C} \overset{\vee}{\text{or}} \overbrace{\text{Doug passes his math class}}^{D)} \overset{\rightarrow}{\text{then}} \overbrace{\text{the sun will rise in the morning.}}^{S}$$

Thus our statement is $(C \vee D) \rightarrow S$.

The process is not always this straightforward since there are many ways of expressing the logical operators in words.

Example 3.2: Translate this sentence into symbols:
"If Doug fails his math test, then in order for the sun to rise tomorrow, it is sufficient that Candace leaves a candle in her window."

We notice the occurrence of $\neg D$, S, and C:

$$\overbrace{\text{If Doug fails his math test,}}^{\neg D} \text{ then in order for } \overbrace{\text{the sun to rise tomorrow,}}^{S}$$
$$\text{it is sufficient that } \overbrace{\text{Candace leaves a candle in her window.}}^{C}$$

We notice the "if...then..." and place an $\rightarrow$ over the "then" and group it by itself. The rest of the sentence seems to be a single unit, so we place it in parenthesis:

$$\overbrace{\text{If Doug fails his math test,}}^{\neg D} \overset{\rightarrow}{\text{then}} \text{ in order for } \overbrace{\text{the sun to rise tomorrow,}}^{(S}$$
$$\text{it is sufficient that } \overbrace{\text{Candace leaves a candle in her window.}}^{C)}$$

What we have in parenthesis - "In order for S, it is sufficient that C" - is an implication. The sufficient part is C, and we recall that the sufficient part of an implication is the antecedent - what comes before the arrow. Thus, what is in parenthesis is $C \rightarrow S$. We draw the arrow backwards here ($\leftarrow$) to maintain the sentence structure.

$$\overbrace{\text{If Doug fails his math test,}}^{\neg D} \overset{\rightarrow}{\text{then}} \text{ in order for } \overbrace{\text{the sun to rise tomorrow,}}^{(S}$$
$$\text{it is } \overset{\leftarrow}{\text{sufficient}} \text{ that } \overbrace{\text{Candace leaves a candle in her window.}}^{C)}$$

Our statement is $\neg D \rightarrow (C \rightarrow S)$. More examples are saved for the exercises.

3.11 Exercises

Let F be the statement "The fox is more clever than the rabbit." Let R be "The rabbit is quicker than the fox," and let C be "The fox will catch the rabbit." Write each of the following compound statements using symbols:

1. The rabbit is quicker than the fox; however, the fox is more clever than the rabbit and will catch the rabbit.

2. The fox is not more clever than the rabbit, and the rabbit is quicker than the fox.

3. The rabbit is quicker than the fox, but the fox will catch the rabbit anyway.

4. Although the fox is more clever than the rabbit, the fox will not catch the rabbit.

5. If the fox is more clever than the rabbit or the rabbit is not quicker than the fox, then the fox will catch the rabbit.

6. In order for the fox to catch the rabbit, it is sufficient that the rabbit is not quicker than the fox.

7. For the fox to catch the rabbit, it is necessary that the rabbit is not quicker than the fox.

8. While the fox is more clever than the rabbit, the rabbit is quicker than the fox; hence, the fox will not catch the rabbit.

3.12 Sentential Languages

Our definition above of *statement* as a declarative sentence which must be either true or false but not both is inadequate. To know what this definition means, we would need first to define the notions of sentence, declarative sentence, true, and false, and we would have to define what it means for a sentence to be true or false. In this section, we outline how to define the notion of statement in a rigorous way that avoids these problems. To emphasize the formality, we will usually call a statement a well-formed formula or a wff.

A sentential (propositional, statement) language is built on a set L of *(atomic) sentence symbols* along with the symbols

$$\wedge, \vee, \neg, \rightarrow, \leftrightarrow,), ($$

The symbols $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ are called *logical connectives* and have intended interpretations in the English language. The parenthesis are to be viewed as punctuation to force grouping. The sentence symbols in L may have different interpretations in different environments. In general, we can only be sure that they represent declarative sentences. These symbols will usually be capital letters (sometimes adorned with subscripts): A, B, P, Q, A_1, A_2, and so on. The intended interpretations of the logical connectives are given in Table 3.1.

connective	name	interpretation
$\wedge$	conjunction	and
$\vee$	disjunctions	or
$\neg$	negation	not
$\rightarrow$	conditional	if...then...
$\leftrightarrow$	biconditional	if and only if

Figure 3.1: Translations of the logical connectives.

An *expression* over L is a finite sequence (list) of symbols. Not all expressions are meaningful. The expression

$$) \leftrightarrow ($$

is meaningless. The expression

$$(P \wedge Q)$$

can be read as "P and Q."

Any meaningful expression – that is, any legal combination of sentence symbols and logical connectives – we will call a *sentence* or *proposition* or (most often) *well-formed formula*. Our first task is to define formally what such an expression is. The description below is designed specifically so that a machine could easily generate well-formed formulas and could easily check whether or not an expression is a well-formed formula. We will abbreviate well-formed formula as wff and will often use Greek letters to represent wffs. The wffs of a language with sentence symbols L are defined as:

Well-Formed Formulas over L

1. Every sentence symbol in L is a wff.
2. If α and β are wffs, then so are
$$(\neg\alpha), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), \text{ and } (\alpha \leftrightarrow \beta).$$
3. No expression can be a wff except for these reasons.

We call the sentence symbols in part 1 *atomic* wffs or atomic sentences (or statements or propositions), while the wffs in part 2 of the

definition are called *compound* wffs or compound sentences (or statements or propositions).

Notice that our definition of a wff over L refers to wffs. A definition such as this is called a *recursive definition*. Recursive definitions often allow for simple descriptions of complex systems that are built systematically from a given foundation. Recursive definitions also provide us with a powerful proof technique call induction which we will explore in Chapter 13.

Example 3.3: Suppose that L contains only the symbols P and Q. List several wffs over L.

According to (1) in the definition of wff, the symbols P and Q are wffs over L. These are the atomic wffs in the language. According to (2), our language should also contain these wffs:

$$(\neg P),\ (\neg Q),\ (P \wedge Q),\ (Q \wedge P),\ (P \vee Q),\ (Q \vee P),\ (P \to Q),\ (Q \to P),\ (P \leftrightarrow Q),\ (Q \leftrightarrow P)$$

Each of these would be considered compound wffs. To form more wffs in the language, we can take each of the wffs we have listed so far (including P and Q) and combine them with the operators to find wffs such as these:

$$((\neg P) \wedge (P \to Q)),\ (\neg(P \to Q)),\ (P \vee (P \leftrightarrow Q))$$

We can then repeat the process combining more and more complex wffs.

3.13 Order of operations and abuse of notation

Using this formal definition seems to leave us with a lot of parenthesis as in

$$(((\neg A) \vee (\neg B)) \to (\neg C)).$$

The order of operations we introduced before helps to alleviate this problem. We declare that $\neg$ takes precedence over all other operations, and we will omit the parentheses at the beginning and end of compound wffs. With these conventions, the wff above could be written in the much more manageable form

$$(\neg A \vee \neg B) \to \neg C.$$

This is technically an abuse of notation, but it is a convention which will make our lives a bit easier.

Chapter 4

Truth Values

In the previous chapter we informally declared how logical operators should interact with truth values. In this chapter, we will make the assignment of truth values more rigorous and show how to draw truth tables for compound statements. First, we assume that we have two values T and F. These represent the words true and false, but we will never declare what those words mean. If we begin with a set L of basic sentence symbols, we can assign a T or an F to each symbol. We then will have rules (listed in the last chapter and listed again below in another form) which will allow us to extend this truth assignment to any wff in the sentential language over L.

4.1 Truth Assignments

A *truth assignment* for a sentential language built from the sentence symbols in a set L is an assignment of either T or F to each of the symbols in L. For each symbol P in L, $s(P)$ denotes the truth value assigned to P.

A truth assignment s on L can be extended to all wffs recursively. Suppose that α and β are wffs whose truth values have been assigned. Then

- $s(\neg\alpha) = \begin{cases} T & \text{if } s(\alpha) = F \\ F & \text{else} \end{cases}$
  Interpret this as saying that "not true" is "false" and "not false" is "true." This can be expressed by the equations $\neg T = F$ and $\neg F = T$.

- $s(\alpha \wedge \beta) = \begin{cases} T & \text{if } s(\alpha) = s(\beta) = T \\ F & \text{else} \end{cases}$
  The only way a conjunction can be true is for both parts of the conjunction to be true. That is, $T \wedge T = T$ but $T \wedge F = F$ and $F \wedge T = F$ and $F \wedge F = F$.
- $s(\alpha \vee \beta) = \begin{cases} F & \text{if } s(\alpha) = s(\beta) = F \\ T & \text{else} \end{cases}$
  The only way a disjunction can be false is if both halves are false. That is, $F \vee F = F$ but $T \vee T = T$ and $T \vee F = T$ and $F \vee T = T$. Note that our "or" is an *inclusive or*. The statement $\alpha \vee \beta$ means for us, "α or β *or both*." Some (computer scientists for example) prefer an *exclusive or* which would mean "α or β *but not both*."
- $s(\alpha \to \beta) = \begin{cases} F & \text{if } s(\alpha) = T \text{ and } s(\beta) = F \\ T & \text{else} \end{cases}$
  These truth values are easiest to follow if we think of implication as a promise. Think of $\alpha \to \beta$ as the promise, "I promise that if α is true then so is β." The implication is true as long as the promise is not broken. There is only one way for the promise to be broken – if α is true but β is false. In particular, if α is false, then the promise cannot be broken (the implication is true). If β is true, the promise cannot be broken (the implication is true).
- $s(\alpha \leftrightarrow \beta) = \begin{cases} T & \text{if } s(\alpha) = s(\beta) \\ F & \text{else} \end{cases}$
  The biconditional $\alpha \leftrightarrow \beta$ means $(\alpha \to \beta) \wedge (\beta \to \alpha)$. It is true when α and β have the same truth value.

These definitions can be summarized in operation tables as in Figure 4.1. Our five logical connectives, their interpretations, and their effects on truth values are summarized in Figure 4.2.

Example 4.1: Suppose that L contains the symbols A, B, C, and D and that s is a truth assignment on L so that $s(A) = s(C) = s(D) = T$ and $s(B) = F$. Calculate

$$s(\neg((A \to B) \vee \neg(C \wedge B))).$$

We will perform this calculation two ways. First, we will follow the recursive definitions above. Then we will calculate the truth values by performing arithmetic on truth values. The second approach is probably simplest, but we need to first to see how the definitions apply.

$\vee$	T	F
T	T	T
F	T	F

$\wedge$	T	F
T	T	F
F	F	F

$\rightarrow$	T	F
T	T	F
F	T	T

$\leftrightarrow$	T	F
T	T	F
F	F	T

Figure 4.1: The effects of the five logical connectives on truth values.

Since $s(A) = s(C) = s(D) = T$ and $s(B) = F$, then $s(A \rightarrow B) = F$ and $s(C \wedge B) = F$. Since $s(C \wedge B) = F$, then $s(\neg(C \wedge B)) = T$. We can combine $s(A \rightarrow B) = F$ and $s(\neg(C \wedge B)) = T$ to see that $s((A \rightarrow B) \vee \neg(C \wedge B)) = T$. Finally, we can negate this to get $s(\neg((A \rightarrow B) \vee \neg(C \wedge B))) = F$.

Now we perform some arithmetic to arrive at the same conclusion. We will first treat the basic symbols as variables and substitute their truth values. We will then calculate with the truth values one step at a time as outlined by the tables in Figure 4.1. At each step in the following computation, we will underline the portion on which we are about to compute to make the arithmetic easier to follow.

$$\begin{aligned}
\neg((A \rightarrow B) \vee \neg(C \wedge B)) &= \neg((T \rightarrow F) \vee \neg\underline{(T \wedge F)}) \\
&= \neg((T \rightarrow F) \vee \underline{\neg F}) \\
&= \neg(\underline{(T \rightarrow F)} \vee T) \\
&= \neg\underline{(F \vee T)} \\
&= \neg T \\
&= F.
\end{aligned}$$

4.2 Exercises

Find the truth values of the following statements.

1. $(P \vee Q) \wedge (P \vee R)$ if P is false, Q is true, and R is true.

2. $\neg(\neg P \wedge (Q \vee \neg R))$ if P is false, Q is false, and R is true.

3. $(\neg P \wedge \neg Q) \vee (P \vee Q)$ if P is false and Q is true.

4. $\neg(P \vee Q) \wedge (P \vee \neg Q)$ if P is true and Q is false.

5. If the sun rises in the east, then it sets in the west.

6. In order for the sun to set in the west, it is necessary for it to rise in the east.

7. In order for the sun to set in the north, it is sufficient for it to rise in the west.

8. If I clap three times, the sun will rise tomorrow.

9. The sun rises in the east only if it sets in the north.

10. The sun rises in the south if and only if the sun rises in the north.

11. Suppose P and Q are statements (it does not matter what they are). Write a compound statement using P and Q which is always **true**. (You do not have to use both P and Q if you do not need to.)

12. Suppose P and Q are statements (it does not matter what they are). Write a compound statement using P and Q which is always **false**. (You do not have to use both P and Q if you do not need to.)

13. Suppose P and Q are statements. Use logical operators to write the statement: "P is true, or Q is true, but not both."

14. Suppose P, Q, and R are statements. Use logical operators to write the statement: "Exactly two of P, Q, and R are true."

4.3 Truth Tables for WFFs

If our set L of sentence symbols is small we can display the effects of all truth assignments on a particular wff at once in a table we call a *truth table*. We domonstrate this with an example.

Example 4.2: Draw a truth table for the wff $(P \vee Q) \wedge (\neg P \vee Q)$.

The first columns of the table will be labeled P and Q. After this, we will have a column for each logical connective that appears in our wff. The columns should be ordered so that there is a column for each argument of the connective before the column of that connective. Our table should have columns labeled by P, Q, $\neg P$, $P \vee Q$, $\neg P \vee Q$, and $(P \vee Q) \wedge (\neg P \vee Q)$. One might call these the *subformulas* of our wff.

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

The formulas are single letters on the extreme left. They get more complicated as we move toward the right until we reach the entire statement. Now, the first two columns will list all possible assignments of truth values for P and Q. Each row corresponds to a different truth assignment.

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T				
T	F				
F	T				
F	F				

We next fill in the column for $\neg P$. These truth values should be exactly opposite those for P. The column for $P \vee Q$ should have an F where P and Q are both F. Otherwise, there is a T in this column. Filling in these two columns gives

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T	F	T		
T	F	F	T		
F	T	T	T		
F	F	T	F		

Fill in the column for $\neg P \vee Q$ by applying $\vee$ to the columns for $\neg P$ and Q.

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T	F	T	T	
T	F	F	T	F	
F	T	T	T	T	
F	F	T	F	T	

Finally, apply $\wedge$ to the last two columns to get the truth values for the entire statement. $T \wedge T = T$, and everything else is false.

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T	F	T	T	T
T	F	F	T	F	F
F	T	T	T	T	T
F	F	T	F	T	F

Each row of this table corresponds to a truth assignment. For example, the second row corresponds to the assignment s where $s(P) = T$ (P is true) and $s(Q) = F$ (Q is false). With this assignment, $s((P \vee \neg Q) \wedge (\neg P \vee Q)) = F$ (that is, $(P \vee \neg Q) \wedge (\neg P \vee Q)$ is false).

Example 4.3: In the language with sentence symbols A, B, and C, draw a truth table for the wff $(A \wedge B) \rightarrow \neg C$.

We need columns for A, B, C, $A \wedge B$, $\neg C$, and the whole formula. To get every possible combination of truth values for A, B, and C, we need eight rows. We then fill in the other columns as we did above to get

A	B	C	$A \wedge B$	$\neg C$	$(A \wedge B) \rightarrow \neg C$
T	T	T	T	F	F
T	T	F	T	T	T
T	F	T	F	F	T
T	F	F	F	T	T
F	T	T	F	F	T
F	T	F	F	T	T
F	F	T	F	F	T
F	F	F	F	T	T

4.4 Exercises

Draw truth tables for each of these

1. $\neg P \vee \neg Q$
2. $\neg(P \wedge Q)$
3. $\neg P \vee Q$
4. $\neg B \rightarrow \neg A$
5. $A \wedge (B \vee C)$
6. $(A \wedge B) \vee (A \wedge C)$

name	symbol	truth table	meaning
negation	$\neg\alpha$	$\begin{array}{c\|c} \alpha & \neg\alpha \\ \hline T & F \\ F & T \end{array}$	not α
conjunction	$\alpha \wedge \beta$	$\begin{array}{c\|c\|c} \alpha & \beta & \alpha \wedge \beta \\ \hline T & T & T \\ T & F & F \\ F & T & F \\ F & F & F \end{array}$	α and β
disjunction	$\alpha \vee \beta$	$\begin{array}{c\|c\|c} \alpha & \beta & \alpha \vee \beta \\ \hline T & T & T \\ T & F & T \\ F & T & T \\ F & F & F \end{array}$	α or β
conditional	$\alpha \rightarrow \beta$	$\begin{array}{c\|c\|c} \alpha & \beta & \alpha \rightarrow \beta \\ \hline T & T & T \\ T & F & F \\ F & T & T \\ F & F & T \end{array}$	if α then β
biconditional	$\alpha \leftrightarrow \beta$	$\begin{array}{c\|c\|c} \alpha & \beta & \alpha \leftrightarrow \beta \\ \hline T & T & T \\ T & F & F \\ F & T & F \\ F & F & T \end{array}$	α iff β

Figure 4.2: Logical connectives and their effects on truth values.

Chapter 5

Implication

There are a variety of manners in which we might define the notion of one sentence α implying another sentence β. Some options are:

- Whenever a truth assignment makes α true, then it also makes β true.
- The wff $\alpha \rightarrow \beta$ is a always true.
- β can be derived from α via some pre-defined algebraic rules of proof.

We begin a look at the first two options in this chapter. We will consider the other option and the relationship between these options later.

5.1 Tautology

A compound statement which is always true regardless of the truth values of the atomic statements involved is called a **tautology**. The standard example of a tautology is $P \vee \neg P$. Any statement P is either true or false. This means that one of P and $\neg P$ must always be true. Hence, $P \vee \neg P$ must be true. We can draw a truth table to verify this:

P	$\neg P$	$P \vee \neg P$
T	F	T
F	T	T

We see that the last column consists of only T's. This is the tell-tale sign of a tautology. Another less obvious example of a tautology is

$(\neg A \vee B) \rightarrow (A \rightarrow B)$. To show this statement is a tautology, we can simply draw a truth table and see that the final column contains only T's.

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

5.2 Contradiction

A compound statement which is always false regardless of the truth values of the atomic statements involved is called a **contradiction**. The standard example of a contradiction is $P \wedge \neg P$. Since P and $\neg P$ will always have opposite truth values, they can never both be true, so $P \wedge \neg P$ must be false. Here is the truth table

P	$\neg P$	$P \wedge \neg P$
T	F	F
F	T	F

To show that any other statement is a contradiction, you may draw a truth table for the statement and see that the final column is all F's.

5.3 Logical Implication

Suppose that Σ is a set of wffs in a sentential language and that α is any wff in the same language. If a truth assignment s forces α to be true, then we will write $s \models \alpha$ (so, $s \models \alpha$ means the same as $s(\alpha) = T$). If a truth assignment s forces every wff in Σ to be true, then we write $s \models \Sigma$. This is read as "s *satisfies* Σ."

We say that Σ *logically implies* α if whenever s is a truth assignment so that $s \models \Sigma$, then also $s \models \alpha$. In symbols, we write this as $\Sigma \models \alpha$. This is read as "Σ *implies* α" or "Σ *entails* α." If Σ contains only a single sentence β (so $\Sigma = \{\beta\}$), we will write $\beta \models \alpha$. If Σ along with a wff α together imply β, then we write $\Sigma, \alpha \models \beta$.

If α is true for every truth assignment, then α is called a *tautology*, and we write $\models \alpha$ (meaning the empty set of sentences implies α). An example of a tautology is $P \vee \neg P$.

At the opposite extreme from a tautology is a contradiction. A wff α is a *contradiction* if $s(\alpha) = F$ for every truth value assignment s. An example of a contradiction of $P \wedge \neg P$.

Suppose that α and β are wffs in a sentential language. Below are some examples of entailment.

Example 5.1: Modus Ponens: $\{\alpha \to \beta, \alpha\} \models \beta$.

This is our most important example of entailment. To see that this is true, simply look at the truth table in Figure 4.2 for $\alpha \to \beta$. There is only one truth assignment in which α and $\alpha \to \beta$ are both true (the first row). This assignment also makes β true. Of course, this entailment is almost obvious. The implication $\alpha \to \beta$ can be translated as saying, "If α is true, then so is β." Of course if this is true and if α is true, then β should be.

Example 5.2: $\{\alpha \vee \beta, \neg\beta\} \models \alpha$

This entailment should seem reasonable. If either α or β is true but β is not true, then α must be true. To verify this formally, we draw a truth table including α, $\alpha \vee \beta$, $\neg\beta$ and α.

α	β	$\alpha \vee \beta$	$\neg\beta$
T	T	T	F
T	**F**	**T**	**T**
F	T	T	F
F	F	F	T

Notice that there is only one row (i.e. one truth assignment) in which both $\alpha \vee \beta$ and $\neg\beta$ are both true. In this row, α is also true.

If some truth assignment makes everything in Σ true and α false, then Σ does not imply α. We write this as $\Sigma \not\models \alpha$.

Example 5.3: $\{\alpha \to \beta, \beta\} \not\models \alpha$

To see that this logical implication does not hold, we can again look at the truth table in Figure 4.2 for $\alpha \to \beta$. In the third row we see that it is possible for $\alpha \to \beta$ and β to be true while α is false.

Example 5.4: $\{\alpha, \beta\} \models \gamma$ if and only if $\alpha \wedge \beta \models \gamma$.

To see this, we will look at a truth table for each logical implication. Since we do not know the relationship between γ and α and β, the best we can do in the truth tables is to imagine that all eight combinations of truth values are possible. These are the tables.

α	β	γ
T	T	T
T	**T**	**F**
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

α	β	γ	$\alpha \wedge \gamma$
T	T	T	T
T	**T**	**F**	**T**
T	F	T	F
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

Since we do not know the relationship between these sentences, we do not know which of these rows are actually possible. Things turn out to be easier if we focus on the rows in which the logical implications **fail**. We isolate our attention to the rows in which the logical implications **fail**. In the first table, there is one row (the second) in which α and β are both true but γ is false. In this row, the first logical implication would fail. In the second table, there is one row (the second) in which $\alpha \wedge \beta$ is true but γ is false. In this row, the implication would fail. We see that the two logical implications fail at the same time, so they must also hold at the same time.

Example 5.5: $\{\sigma, \alpha\} \models \beta$ if and only if $\sigma \models (\alpha \to \beta)$

We again approach this by drawing two truth tables and seeing where the implications may fail.

σ	α	β
T	T	T
T	**T**	**F**
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

σ	α	β	$\alpha \to \beta$
T	T	T	T
T	**T**	**F**	**F**
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

In the first table, there is one row in which σ and α are both true and β is false. This is the only row in which the first implication fails. In the second table, there is only one row in which σ is true and $\alpha \to \beta$ is false. Note that this is the same row as in the first table.

Example 5.6: If $\Sigma = \{\gamma_1, \ldots, \gamma_n\}$ then

$$\Sigma, \alpha \models \beta \text{ if and only if } \Sigma \models (\alpha \to \beta)$$

Let

$$\sigma = \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n.$$

Then this entailment follows from the previous example because a truth assignment will satisfy Σ if and only if it satisfies σ.

5.4 Exercises

For each of these implications, either explain why the implication holds or explain why it does not hold. You may want to draw truth tables.

1. $\{\alpha \to \beta, \neg\beta\} \models \neg\alpha$.
2. $\{\alpha \to \beta, \neg\alpha\} \models \neg\beta$.
3. $\{\alpha, \beta\} \models \alpha \wedge \beta$.
4. $\{\alpha \wedge \beta\} \models \alpha$.
5. $\{\alpha \vee \beta\} \models \alpha$.
6. $\{\alpha\} \models \alpha \vee \beta$.
7. $\{\alpha\} \models \alpha \wedge \beta$.
8. $\{\alpha, \neg\alpha\} \models \beta$.
9. $\{\alpha \to \beta, \beta \to \gamma\} \models \alpha \to \gamma$.
10. $\models (\beta \vee \neg\beta)$.

11. Explain why $\alpha \models \beta$ if and only if $\models (\alpha \to \beta)$ (that is, if and only if $\alpha \to \beta$ is a tautology).
12. Explain why $\{\alpha, \beta\} \models \gamma$ if and only if $\alpha \wedge \beta \models \gamma$.
13. Explain why $\{\alpha, \beta\} \models \gamma$ if and only if $(\alpha \wedge \beta) \to \gamma$ is a tautology.
14. Explain why $\{\alpha, \beta\} \models \gamma$ if and only if $\alpha \to (\beta \to \gamma)$ is a tautology.

Chapter 6

Logical Equivalence

Closely related to logical implication is the idea of logical equivalence. If you look closely at the truth table we drew for $(P \vee \neg Q) \wedge (\neg P \vee Q)$ in the last section, you will notice that the truth values for $(P \vee \neg Q) \wedge (\neg P \vee Q)$ are identical to those for Q in the same table. If all we care about is truth values, then these statements should in some way be equivalent.

Suppose we are working in a sentential language with only two atomic sentence symbols. If we draw a truth table for a wff in this language, then there are only sixteen possible final columns of the table:

F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T

If the truth values of a wff capture all the information that we care about, then *essentially* there are only 16 different things that a wff can mean in this language – different wffs can have the same *meaning*. This motivates the following definition.

Suppose that α and β are wffs in the same sentential language. We say that α and β are *logically equivalent* and write $\alpha \equiv \beta$ if $s(\alpha) = s(\beta)$ for every truth assignment s. That is, α is equivalent to β if α and β have the same truth value for every truth assignment.

The definitions of logical equivalence and logical implication immediately imply this theorem:

Theorem 6.1: $\alpha \equiv \beta$ if and only if both $\alpha \models \beta$ and $\beta \models \alpha$.

For an example of a logical equivalence,

Example 6.2: If α and β are wffs in the same language, then

$$\alpha \to \beta \equiv \neg\alpha \vee \beta.$$

To see this, we simply draw two truth tables:

α	β	$\alpha \to \beta$
T	T	T
T	F	F
F	T	T
F	F	T

and

α	β	$\neg\alpha$	$\neg\alpha \vee \beta$
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	T

Notice that the final columns of the tables are identical.

6.1 Basic Equivalences

There are only eight types of equivalences you need to remember. All other logical equivalences can be derived from these. They are listed below.

6.1.1 Commutative Laws

The English words "and" and "or" do not care about order. Saying "The grass is green, or the sky is blue" communicates the same thing as "The sky is blue, or the grass is green." The same is true with "and." Thus our first pair of equivalences should make sense:

$$P \wedge Q \equiv Q \wedge P \text{ and } P \vee Q \equiv Q \vee P$$

Notice that these resemble the commutative laws for multiplication and addition.

6.1.2 Associative Laws

Our next pair of equivalences resembles the associative laws for multiplication and addition. The two statements "Jill and Jane passed math, and Janet passed math" and "Jill passed math, and Jane and Janet passed math" communicate the same thing - all three women

passed. The word "and" does not care how statements are grouped together, and neither does "or." Thus

$$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R) \text{ and } (P \vee Q) \vee R \equiv P \vee (Q \vee R)$$

Because of this equivalence, we will usually just write $P \wedge Q \wedge R$ or $P \vee Q \vee R$ and dispense with the parenthesis.

6.1.3 Idempotent Laws

Our next set of equivalences again reflect the English language. All three of these statements

"It is not the case that it is not raining."
"It is raining, and it is raining."
"Either it is raining, or it is raining."

communicate the same thing - "It is raining." Thus we have three equivalences called the idempotent laws

$$\neg(\neg P) \equiv P \text{ and } P \wedge P \equiv P \text{ and } P \vee P \equiv P$$

6.1.4 Absorption Laws

The next pair of equivalences are perhaps the least intuitive and least reflect a situation in English. For now, we will justify them by truth tables. They will become much more intuitive when we discuss switching networks. The equivalences are

$$P \wedge (P \vee Q) \equiv P \text{ and } P \vee (P \wedge Q) \equiv P$$

Here is a truth table for $P \wedge (P \vee Q)$

P	Q	$P \vee Q$	$P \wedge (P \vee Q)$
T	T	T	T
T	F	T	T
F	T	T	F
F	F	F	F

Notice that when P is true this statement is true. When P is false, this statement is false. Hence they are equivalent.

6.1.5 Distributive Laws

Consider the statement "It is raining, and either Hal forgot his hat or he forgot his coat." If this is true, what do we know? We know it is raining. We know that Hal forgot either his hat or his coat. In the first case, it is raining and he forgot his hat. In the second, it is raining and he forgot his coat. The statement seems to say "It is raining and Hal forgot his coat, or it is raining and Hal forgot his hat." This reflects our next pair of equivalences:

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

and

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

These resemble the way in which we distribute multiplication over addition.

6.1.6 DeMorgan's Laws

Consider the sentence "It is not true that Sam passed math and English." What does this mean? Let P be "Sam passed math," and let Q be "Sam passed English." The statement we are looking at is $\neg(P \wedge Q)$. In order for this to be true, $P \wedge Q$ needs to be false. This happens if at least one of P and Q is false. Thus our sentence appears to be $\neg P \vee \neg Q$ - "Either Sam did not pass math, or sam did not pass English." This is an example of one of DeMorgan's Laws:

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q \text{ and } \neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

You can think of DeMorgan's Laws as distributing negation over $\wedge$ and $\vee$ - except that the negation applies to everything, even the $\wedge$ and the $\vee$.

6.1.7 Disjunctive Implication

We already saw this equivalence as an example earlier. Here it is

$$P \to Q \equiv \neg P \vee Q$$

This reflects what was said in the first chapter that $P \to Q$ can be phrased as "Not P unless Q."

6.1.8 Contrapositive

Suppose you know this statement is true "If Sam won his game, he is going to play in the championship game." If someone tells you that Sam is not going to play in the championship game, then you immediately conclude that Sam did not win his game. You are intuitively aware of this final logical equivalence

$$P \to Q \equiv \neg Q \to \neg P$$

The statement $\neg Q \to \neg P$ is called the **contrapositive** of $P \to Q$. You should be careful not to confuse it with the statement $\neg P \to \neg Q$ known as the **inverse** of $P \to Q$ or with $Q \to P$ known as the **converse** of $P \to Q$. These statements are not equivalent to $P \to Q$. Here are all of the basic equivalences together:

Basic Logical Equivalences

$\alpha \wedge \beta \equiv \beta \wedge \alpha$	commutative law
$\alpha \vee \beta \equiv \beta \vee \alpha$	commutative law
$\alpha \wedge (\beta \wedge \gamma) \equiv (\alpha \wedge \beta) \wedge \gamma$	associative law
$\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \vee \gamma$	associative law
$\neg(\neg\alpha) \equiv \alpha$	idempotent law
$\alpha \wedge \alpha \equiv \alpha$	idempotent law
$\alpha \vee \alpha \equiv \alpha$	idempotent law
$\alpha \equiv \alpha \vee (\alpha \wedge \beta)$	absorption law
$\alpha \equiv \alpha \wedge (\alpha \vee \beta)$	absorption law
$\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$	distributive law
$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$	distributive law
$\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$	DeMorgan's Law
$\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$	DeMorgan's Law
$\alpha \to \beta \equiv \neg\alpha \vee \beta$	disjunctive implication
$\alpha \to \beta \equiv \neg\beta \to \neg\alpha$	contrapositive
$\alpha \leftrightarrow \beta \equiv (\alpha \to \beta) \wedge (\beta \to \alpha)$	biconditional

In all of these equivalences, remember that α, β, and γ are wffs and may represent any wff in the appropriate language.

6.2 Special Equivalences

Suppose that α is a contradiction and that β is any wff in the same language as α. If we draw a table to illustrate the possible truth values of $\alpha \vee \beta$, it would look like this:

α	β	$\alpha \vee \beta$
F	T	T
F	F	F

Notice that $\alpha \vee \beta$ always has the same truth value as β, so $\alpha \vee \beta \equiv \beta$. We summarize this by saying that $F \vee \beta \equiv \beta$. A similar result holds for tautologies and conjunctions:

Theorem 6.3: (Equivalences, Tautologies, Contradictions)

$$F \vee \beta \equiv \beta \text{ and } T \wedge \beta \equiv \beta.$$

6.3 Algebra with equivalences

The equivalences above can be used to demonstrate more complicated equivalences. For example, the sentence $\neg(A \wedge \neg B)$ is equivalent to $A \rightarrow B$. We can use the basic equivalences to show this. Notice that $\neg(A \wedge \neg B)$ is of the form of one of DeMorgan's Laws, so we can "distribute" the negation to get $\neg(A \wedge \neg B) \equiv \neg A \vee \neg(\neg B)$. Next, notice the double negation. We can use one of the idempotent laws to get $\neg A \vee \neg(\neg B) \equiv \neg A \vee B$. This last wff looks just like part of disjunctive implication, which tells us that $\neg A \vee B \equiv A \rightarrow B$. This is what we were looking for. Here is our work all together:

$$\begin{array}{rcll} \neg(A \wedge \neg B) & \equiv & \neg A \vee \neg(\neg B) & \text{DeMorgan's Law} \\ & \equiv & \neg A \vee B & \text{Idempotent Law} \\ & \equiv & A \rightarrow B & \text{Disjunctive Implication} \end{array}$$

Notice how we set up our work here. To show that $\neg(A \wedge \neg B) \equiv A \rightarrow B$, we begin with one formula on the left of an equivalence sign (here we use $\neg(A \wedge \neg B)$, but we could very well have started with the other statement). We then apply equivalences to this sentence, listing the results to the right of equivalence signs until we arrive at $A \rightarrow B$. Here are more examples:

Example 6.4: Show $A \to (P \vee R) \equiv (A \to P) \vee (A \to R)$

Solution:

$$\begin{array}{lcll}
(A \to P) \vee (A \to R) & \equiv & (\neg A \vee P) \vee (\neg A \vee R) & \text{(disjunctive implication)} \\
& \equiv & \neg A \vee (P \vee (\neg A \vee R)) & \text{(associative law} \\
& \equiv & \neg A \vee ((P \vee \neg A) \vee R) & \text{(associative law)} \\
& \equiv & \neg A \vee ((\neg A \vee P) \vee R) & \text{(commutative law)} \\
& \equiv & \neg A \vee (\neg A \vee (P \vee R)) & \text{(associative law)} \\
& \equiv & (\neg A \vee \neg A) \vee (P \vee R) & \text{(associative law)} \\
& \equiv & \neg A \vee (P \vee R) & \text{(idempotent law)} \\
& \equiv & A \to (P \vee R) & \text{(disjunctive implication)}
\end{array}$$

This looks a little confusing with all of the associative law applications. It is actually much simpler. If we abuse a little notation, the work looks like:

$$\begin{array}{lcll}
(A \to P) \vee (A \to R) & \equiv & (\neg A \vee P) \vee (\neg A \vee R) & \text{(disjunctive implication)} \\
& \equiv & \neg A \vee P \vee \neg A \vee R & \text{(associative law)} \\
& \equiv & \neg A \vee \neg A \vee P \vee R & \text{(commutative law)} \\
& \equiv & \neg A \vee P \vee R & \text{(idempotent law)} \\
& \equiv & \neg A \vee (P \vee R) & \text{(associative law)} \\
& \equiv & A \to (P \vee R) & \text{(disjunctive implication)}
\end{array}$$

Usually, we will abuse notation like this and ignore parenthesis when associativity allows it. In fact, we will usually apply associativity and commutativity without calling attention to it.

Example 6.5: Show $(A \vee B) \vee (A \wedge P) \vee (B \wedge P) \equiv (A \vee B)$

Solution:

$$(A \vee B) \vee (A \wedge P) \vee (B \wedge P) \equiv$$

$$\begin{array}{lcll}
& \equiv & (A \vee B) \vee (P \wedge A) \vee (P \wedge B) & \text{(commutative law)} \\
& \equiv & (A \vee B) \vee (P \wedge (A \vee B)) & \text{(distributive law)} \\
& \equiv & (A \vee B) \vee ((A \vee B) \wedge P) & \text{(commutative law)} \\
& \equiv & (A \vee B) & \text{(absorption law)}
\end{array}$$

The second step in this example can be thought of as "factoring" or "undistributing" a common P from the last two terms. Commutativity actually lets us distribute from both directions, so we could shorten this to

$$(A \vee B) \vee (A \wedge P) \vee (B \wedge P) \equiv$$

$$\begin{array}{lcll}
& \equiv & (A \vee B) \vee ((A \vee B) \wedge P) & \text{(commutative law)} \\
& \equiv & (A \vee B) & \text{(absorption law)}
\end{array}$$

Example 6.6: Show $(P \to B) \wedge (Q \to B) \equiv (P \vee Q) \to B$

Solution:

$$\begin{array}{rcll}(P \to B) \wedge (Q \to B) & \equiv & (\neg P \vee B) \wedge (\neg Q \vee B) & \text{(disjunctive implication)} \\ & \equiv & (\neg P \wedge \neg Q) \vee B & \text{(distributive law)} \\ & \equiv & \neg(P \vee Q) \vee B & \text{(DeMorgan's Law)} \\ & \equiv & (P \vee Q) \to B & \text{(disjunctive implication)}\end{array}$$

6.4 Exercises

Use the basic logical equivalences to show that these statements are equivalent.

1. $(P \to R) \vee (Q \to R) \equiv (P \wedge Q) \to R$
2. $\neg(A \vee B) \vee P \equiv (A \to P) \wedge (B \to P)$
3. $(A \vee B) \wedge (C \vee D) \equiv (A \wedge C) \vee (B \wedge C) \vee (A \wedge D) \vee (B \wedge D)$
4. $A \vee (B \vee C) \equiv (A \vee B) \vee (A \vee C)$

6.5 Simplification

Logical equivalences often can be used to simplify statements. For example,

Example 6.7: Use equivalences to simplify the statement "It is not true that I passed and you did not."

This sounds a little confusing. Let I be "I passed," and let Y be "You passed." The statement we are considering is $\neg(I \wedge \neg Y)$. Using DeMorgan's Law and then the idempotent law, we see $\neg(I \wedge \neg Y) \equiv \neg I \vee \neg\neg Y \equiv \neg I \vee Y$. Thus the original statement is equivalent to the simpler statement "Either I did not pass, or you did."

Logical equivalences can be used to simplify instructions or conditions. For example, suppose you are building a machine with a warning light and you are told that "The warning light should come on if either the temperature is high while the pressure is high and the door is open or the temperature is high while it is not the case that both the pressure is not high and the door is closed." This is a baffling condition.

Example 6.8: Simplify the statement, "The warning light should come on if either the temperature is high while the pressure is high and the door is open or the temperature is high while it is not the case that both the pressure is not high and the door is closed."

Let T be "The temperature is high." Let P be "The pressure is high," and let D be "The door is open." Our condition for the warning light to come

on is $(T \wedge P \wedge D) \vee (T \wedge \neg(\neg P \wedge \neg D))$. This is confusing, and the circuitry to build the warning light could be quite complicated. However, notice

$$(T \wedge P \wedge D) \vee (T \wedge \neg(\neg P \wedge \neg D)) \equiv$$

$$\begin{array}{lll} \equiv & T \wedge ((P \wedge D) \vee (P \vee D)) & \text{(distributive law)} \\ \equiv & T \wedge (((P \wedge D) \vee P) \vee ((P \wedge D) \vee D)) & \text{(distributive law)} \\ \equiv & T \wedge (P \vee D) & \text{(absorption law)} \end{array}$$

so the condition is equivalent to the much simpler statement "The temperature is high and either the pressure is high or the door is open."

6.6 Exercises

Convert each of the following statements into symbols using logical connectives. You will have to introduce symbols (letters) for parts of the statments to do so. Use logical equivalences to simplify each statement. Then translate each statement back into words.

1. It is not true that both it is not cold and it is raining or snowing.
2. The possible combinations of toppings on the sandwich are meat and pickles and cheese, or meat and onions and cheese, or meat and pickles and tomatoes.
3. You will pass if either you pass both the midterm and the final, or if you do not fail both the major project and the final.
4. If we beat the "Cats" and the "Dogs" but not the "Penguins," or if we beat the "Cats" and the "Penguins" but not the "Dogs," or if we beat all three, then we will go to the playoffs.

6.7 Disjunctive Normal Form

Consider the following list of equivalences:

$$P \wedge (A \to Q) \wedge \neg R \equiv$$

$$\begin{array}{lll} \equiv & P \wedge (\neg A \vee Q) \wedge \neg R & \text{(disjunctive implication)} \\ \equiv & ((P \wedge \neg A) \vee (P \wedge Q)) \wedge \neg R & \text{(distributive law)} \\ \equiv & (P \wedge \neg A \wedge \neg R) \vee (P \wedge Q \wedge \neg R) & \text{(distributive law)} \end{array}$$

This last wff is in a special form. It is the disjunction ($\vee$) of statements which are conjunctions ($\wedge$) of atomic statements and negated atomic statements. Such a wff is said to be in *disjunctive normal form*. Every wff is equivalent to a statement in disjunctive normal form.

There is a simple strategy for converting any statement to disjunctive normal form (the strategy is simple, but applying it can get messy). First,

apply disjunctive implication to get rid of any implications. Second, apply DeMorgan's Laws along with the idempotent law repeatedly until the only statements which are negated are atomic statements. Next, distribute $\wedge$s over $\vee$s repeatedly. Along the way, you may simplify things with absorption and the special equivalences from Theorem 6.3. Here is an example.

Example 6.9: Place $R \wedge \neg((A \wedge B) \rightarrow (P \wedge Q))$ in disjunctive normal form.

$$R \wedge \neg((A \wedge B) \rightarrow (P \wedge Q)) \equiv$$

$$\begin{array}{lll}
\equiv & R \wedge \neg(\neg(A \wedge B) \vee (P \wedge Q)) & \text{(disjunctive implication)} \\
\equiv & R \wedge (\neg\neg(A \wedge B) \wedge \neg(P \wedge Q)) & \text{(DeMorgan's Law)} \\
\equiv & R \wedge ((A \wedge B) \wedge \neg(P \wedge Q)) & \text{(idempotent law)} \\
\equiv & R \wedge ((A \wedge B) \wedge (\neg P \vee \neg Q)) & \text{(DeMorgan's Law)} \\
\equiv & R \wedge ((A \wedge B \wedge \neg P) \vee (A \wedge B \wedge \neg Q)) & \text{(distributive law)} \\
\equiv & (R \wedge A \wedge B \wedge \neg P) \vee (R \wedge A \wedge B \wedge \neg Q) & \text{(distributive law)}
\end{array}$$

It is possible to end up with a disjunction of more than two statements. For example, you may have $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$. It is also possible to end up with simply a conjunction of atomic statements and negations of atomic statements.

6.8 Exercises

Write each of the following wffs in disjunctive normal form.

1. $A \rightarrow \neg(B \vee C)$
2. $(A \rightarrow B) \wedge \neg(A \rightarrow C)$
3. $A \wedge (\neg B \rightarrow (C \wedge D))$
4. $(A \rightarrow B) \wedge (\neg A \rightarrow \neg B)$

6.9 Building Statements

We can build wffs to have any desired set of truth values we like. For example,

Example 6.10: Find a statement which has a truth table that looks like

P	Q	$\cdots$	?
T	T		F
T	F		T
F	T		T
F	F		F

To construct the statement, locate the rows where we want truth. In this case, these are the middle two rows. These two rows give two conditions. The first row would require P to be true and Q to be false. A statement which would give truth here is $P \wedge \neg Q$. The second true row requires P to be false and Q to be true. A statement which has truth in this instance is $\neg P \wedge Q$. To construct the statement we need, we simply join these two statements with an $\vee$: $(P \wedge \neg Q) \vee (\neg P \wedge Q)$. This statement will have the desired truth values.

The strategy is this: For each true row in the truth table form a conjunction. Each letter involved should appear once in the conjunction. If in that row the letter is assigned a value of F, negate it in the conjunction. Form a conjunction like this for each true row. Then join these together with $\vee$.

Example 6.11: Find a wff which would have these truth assignments:

P	Q	R	$\cdots$	?
T	T	T		T
T	T	F		F
T	F	T		T
T	F	F		T
F	T	T		F
F	T	F		T
F	F	T		T
F	F	F		F

A solution is:

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

6.10 Exercises

1. Find a statement which has this truth table

P	Q	R	$\cdots$	?
T	T	T		F
T	T	F		F
T	F	T		F
T	F	F		T
F	T	T		T
F	T	F		F
F	F	T		F
F	F	F		T

2. Find a statement which has this truth table

P	Q	R	$\cdots$	?
T	T	T		F
T	T	F		T
T	F	T		F
T	F	F		F
F	T	T		F
F	T	F		F
F	F	T		F
F	F	F		T

3. Find a statement which has this truth table

P	Q	R	$\cdots$	?
T	T	T		T
T	T	F		T
T	F	T		F
T	F	F		F
F	T	T		F
F	T	F		F
F	F	T		T
F	F	F		T

4. Find a compound statement using four atomic statements which is true when three or more of the atomic statements is true.
5. Find a compound statement using four atomic statements which is true when exactly two of the atomic statements is true.
6. Find a wff for each of the 16 final truth table columns listed at the beginning of this chapter.

6.11 Implications

There are many cases in our natural language when an implication may be expressed without the words "if...then.." For example, the sentence

> The square of any even integer is even.

is logically the same as the implication

> If n is an even integer, then n^2 is even.

The statement

> All kittens are cute.

can be expressed as

If it is a kitten, then it is cute.

The sentence

When it rains, it pours.

can be expressed

If it is raining, then it is pouring.

Using disjunctive implication, the contrapositive, and varying English translations, we can rewrite these sentences in a variety of ways:

Original: When it rains, it pours.

Implication: If it is raining, then it is pouring.

Contrapositive: If it is not pouring, then it is not raining.

Disjunction: Either it is not raining, or it is pouring.

Necessary: In order for it to rain, it is necessary that it pours.

Sufficient: In order for it to pour, it is sufficient for it to rain.

6.12 Exercises

Translate each of these sentences into an implication. Then rewrite each with the contrapositive, as a disjunction, using "necessary," and using "sufficient."

1. All men are liars.
2. When the sun shines, she dances.
3. She cries when it rains.
4. All primes greater than 2 are odd. (Hint: "If n is...")
5. If you build it, he will come.

6.13 Fewer Logical Connectives

Not all of our logical connectives are necessary. The biconditional can be expressed with conjunction and implication. Disjunctive implication lets us express the implication with $\neg$ and $\vee$. In fact, DeMorgan's Laws let us write $\vee$ with $\neg$ and $\wedge$:

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q).$$

We can also express $\wedge$ with $\neg$ and $\vee$:

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q).$$

We could write all of our well-formed formulas using fewer logical connectives. However, having all of our connectives makes it easier to translate between the wffs and English (and easier to think in terms of the wffs).

6.14 Exercises

1. Use the comments above to express every connective in terms of $\neg$ and $\wedge$.
2. Use the comments above to express every connective in terms of $\neg$ and $\vee$.
3. Express every connective in terms of $\neg$ and $\rightarrow$.
4. Define a new connective (called the Sheffer Stroke) by $P|Q = \neg(P \wedge Q)$. Express all of the logical connectives using $|$. For example, $\neg P = P|P$ and $P \wedge Q = (P|Q)|(P|Q)$. (Check these.) This means that we could do all of our logic with only one logical connective.
5. Is the Sheffer stroke operation associative?
6. Is the Sheffer stroke operation commutative?

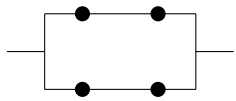
Chapter 7

Switching Networks

Switching networks were initially inspired by electronic applications; however, it is not necessary to know anything about electronics to study switching networks. The notions we employ here apply in any environment in which some current (such as electricity, water, or light) is flowing along a path (such as a wire, a pipe, or optic fiber) and is controlled by some type of switching mechanisms or valves which can be turned on and off. The ultimate goal of this chapter is to gain more insight about logic and to see an application of what we have learned so far. We will see that the design of such things as electronic circuits is tied to the rules of logic we have been studying.

7.1 Switches and Switching Networks

A *switch* is a device with two states, "on" and "off." A switch controls the flow of a current. When a switch is on, a current may flow through the switch. When a switch is off, it may not. Our currents will flow along things called paths. When switches are placed on a collection of paths to regulate the flow of the current, what results is called a *switching network*. All of our switching networks have a single path through which the current enters the network. This is the *input path*. Each network has a single path through which the current may leave – the *output path*. If current can flow from the input to the output path, then the network is on. Otherwise, the network is off. In this chapter, we will be drawing pictures of switching networks. In the pictures, switches will be symbolized by filled circles, and paths will be symbolized by line segments. For example



An example of a switching network.

If a line segment passes through a circle it symbolizes that the corresponding switch controls the flow of the current along that path.

7.2 Switches and Logical Operators

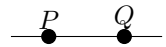
We can draw switching networks to represent logical statements in such a way that a network is on precisely when the logical statement is true. Let P be the symbol of a logical statement. We can label switches in a switching network with the symbol P. When we do so, we assume that when P is true, then the switches labeled P are on. When P is false, they are off.

Here is a very small switching network.

A switching network for the statement P.

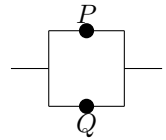
When P is true, this network is on. When P is false, this network is off.

We can draw networks for compound statements also. This is a network for $P \wedge Q$.



A network for $P \wedge Q$.

The only way that current can flow through this network is for both switches to be on. That happens when both statements are true, so $P \wedge Q$ is true. If current does not flow, one of the switches is off. This means either P or Q is false, so $P \wedge Q$ is false. The switches in the network for $P \wedge Q$ are said to be connected in *series*. We will always associate series with $\wedge$. Here is a network for $P \vee Q$.



A network for $P \vee Q$.

Current will flow through this network exactly when one of the switches is on, which is when one of the statements is true, which is exactly when $P \vee Q$ is true. The switches in the network for $P \vee Q$ are said to be connected in *parallel*. We will always associate parallel with $\vee$.

In order to draw a network for negation, we cheat. We can label switches by the negation of a symbol for a statement. For example, if P is a statement, we can label a switch as $\neg P$. This switch would be on when P is false and off when P is true. The network would look like this.

A switching network for the statement $\neg P$.

To be able to draw switching networks for more complicated compound statements, we will assume that we can give more than one switch the same label. For instance, a network may have many switches labeled by P. All of these switches would be on when P is true. All would be off when P is false.

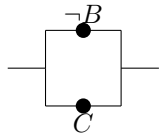
7.3 Networks for Compound Statements

We can draw switching networks for any wff.

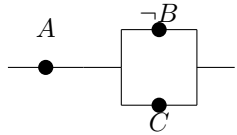
Example 7.1: Draw a switching network for $A \wedge (B \rightarrow C)$.

To handle $\rightarrow$, we will use the logical equivalence $B \rightarrow C \equiv \neg B \vee C$. Thus we will draw a network for $A \wedge (\neg B \vee C)$. First, we list the subformulas of our wff (as we did when we drew truth tables): A, B, C, $\neg B$, $\neg B \vee C$, $A \wedge (\neg B \vee C)$. We draw networks for each subformula progressing from simpler to more complex.

A Network for $\neg B$



$\neg B \vee C$ is formed by placing $\neg B$ and C in parallel.



$A \wedge (\neg B \vee C)$ is formed by placing A and $\neg B \vee C$ in series.

Example 7.2: Draw a network for the statement

$$\neg(A \wedge B) \wedge (P \rightarrow (A \vee B)).$$

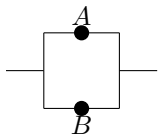
First, we must use disjunctive implication to rewrite this as

$$\neg(A \wedge B) \wedge (\neg P \vee (A \vee B)).$$

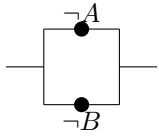
Next, DeMorgan's Law gives

$$\neg(A \wedge B) \wedge (\neg P \vee (A \vee B)) \equiv (\neg A \vee \neg B) \wedge (\neg P \vee (A \vee B)).$$

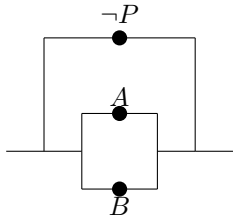
We now make a list of subformulas: A, B, P, $\neg A$, $\neg B$, $\neg P$, $A \vee B$, $\neg A \vee \neg B$, $\neg P \vee (A \vee B)$. Finally, we can start drawing. We will not draw networks for A, B, P, $\neg A$, $\neg B$, or $\neg P$.



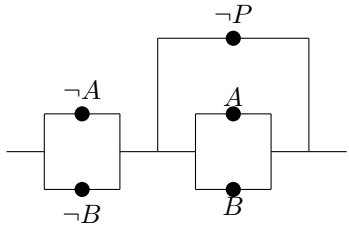
$A \vee B$



$\neg A \vee \neg B$



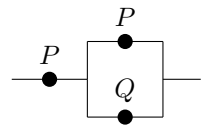
To make $\neg P \vee (A \vee B)$ we place $\neg P$ and $A \vee B$ in parallel.



We put $\neg A \vee \neg B$ and $\neg P \vee (A \vee B)$ in series to arrive at the entire sentence.

7.4 Absorption Laws

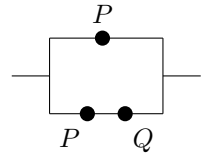
We commented earlier that the absorption laws would make more sense with switching networks. The network for $P \wedge (P \vee Q)$ looks like



A network for $P \wedge (P \vee Q)$.

If P is true, then the switches labeled P are on and the current can flow through the top half of this network (regardless of Q). If P is false, then the first switch labeled P prevents the current from flowing (again, regardless of Q). The network is on precisely when P is true. It seems reasonable, then, that this statement should be equivalent to P.

The network for $P \vee (P \wedge Q)$ looks like



A network for $P \vee (P \wedge Q)$.

If P is true, current can flow through the top of the network. If P is false, it cannot flow through either half of the network. Again, the network is on exactly when P is true.

7.5 Designing a Network

Example 7.3: Design a switching network to operate a hallway light with two switches.

If the light is on and either switch is flipped, then the light should go off. If the light is off and either switch is flipped, the light should come on. We are going to construct a logical statement which mimics the behavior of the desired network, and then we will draw the network.

We will have two sentence symbols P and Q since there are to be two physical switches in the hallway. We assume that the light is on if both switches are on. If we flip the switch corresponding to Q off (so P is true and Q is false), the lights should turn off (the statement is false). Similarly, if we start with both on and turn the switch for P off (so P is false and Q is true), then the lights should go off (the statement is false). At this point, if we turn

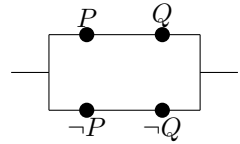
the switch for Q off (P and Q are now both false), the lights should turn on (the statement is true). This gives the desired truth table

P	Q	$\cdots$	?
T	T		T
T	F		F
F	T		F
F	F		T

Using our earlier technique, we construct the statement

$$(P \wedge Q) \vee (\neg P \wedge \neg Q).$$

This should have the appropriate truth values. We now have a logical statement which mimics the desired switching network. We need to draw the network for $(P \wedge Q) \vee (\neg P \wedge \neg Q)$. Since the statement is in disjunctive normal form, drawing it is simple.



A switching network to operate a hall light.

7.6 Exercises

Draw switching networks for each of the following statements.

1. $(P \wedge \neg Q) \vee (\neg P \wedge Q)$
2. $((\neg P \vee Q) \wedge R) \vee \neg Q$
3. $P \rightarrow (A \wedge B)$
4. $\neg(A \vee (B \wedge C)) \wedge (P \vee Q)$
5. $A \vee (B \wedge (C \vee D))$
6. $(A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)$
7. $(A \wedge \neg B) \vee (B \wedge \neg C) \vee (C \wedge \neg D) \vee E$
8. $(A \wedge B) \vee (\neg A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge \neg B)$
9. A light is to be operated by three switches. The light is to be on when two or more of the switches are on. Draw a switching network to control such a light.
10. A light is to be operated by three switches labeled A, E, and T. If the letters corresponding to the switches which are on can be used to spell an English word, the light should be on. Otherwise, the light is to be off. Draw a switching network to accomplish this.

Chapter 8

Deduction

In Chapter 5 we saw one way of concluding the truth of a wff α from a collection Σ of other wffs. In that chapter, we said that Σ logically implies α if every truth assignment that makes every statement in Σ true also makes α true. In this chapter, we investigate a more mechanical way in which to construct α from Σ when Σ implies α. It should not be obvious that the notion of implication we define here corresponds to the notion in Chapter 5. We will prove the concepts are equivalent later.

8.1 Modus Ponens and Proofs

The most fundamental entailment example from Chapter 5 was Example 5.1:

$$\{\alpha \to \beta, \alpha\} \models \beta.$$

This is the basis for our most important rule of inference:

Modus Ponens: From $\alpha \to \beta$ and α, infer β.

Modus Ponens is the basis for our notion of proof.

Proof Suppose that Σ is a set of wffs and α is a wff. A *proof* or *deduction* of α from Σ is a list $\gamma_1, \gamma_2, \ldots, \gamma_n$ so that $\gamma_n = \alpha$ and each sentence γ_i in the list

1. is a tautology or
2. is in Σ or
3. follows from two earlier sentences by the rule Modus Ponens.

If there is a proof of α from Σ, then we say that α is *provable* or *deducible* from Σ. We express this in symbols as $\Sigma \vdash \alpha$.

When listing the sentences in a proof of a wff α from a set Σ of wffs, we list the statements in an organized fashion so it is clear where the statements in the list come from. This organized list will consist of two columns – one containing the sentences, and one containing reason for the sentences. In these proofs, we will call the statements in Σ *premises* – statements that are being assumed to establish α.

Example 8.1: $\{(P \vee Q) \to R, P\} \vdash R$

1. $(P \vee Q) \to R$	Premise
2. P	Premise
3. $P \to (P \vee Q)$	Tautology
4. $P \vee Q$	Modus Ponens 3,1
5. R	Modus Ponens 1,4

Of course, to write this proof, we need to know that the sentence in line 3 is a tautology. We could draw a truth table to verify this (though it should almost be clear by thinking of the meaning of the sentence). Our definition of proof allows for all tautologies. We do not need to allow all tautologies, but doing so simplifies proofs. Most of the tautologies we use will be of the form $\alpha \to \beta$ where $\alpha \equiv \beta$ is one of the equivalences in our list of basic equivalences in Chapter 5.

Example 8.2: Modus Tollens: $\{\alpha \to \beta, \neg\beta\} \vdash \neg\alpha$

1. $\alpha \to \beta$	Premise
2. $\neg\beta$	Premise
3. $(\alpha \to \beta) \to (\neg\beta \to \neg\alpha)$	Tautology
4. $\neg\beta \to \neg\alpha$	Modus Ponens 1, 3
5. $\neg\alpha$	Modus Ponens 4,2

8.2 Derived Rules of Inference

If the only rule of inference we have is Modus Ponens, then our proofs will be somewhat complicated. We will derive some additional rules of inference from Modus Ponens. Because these will be derived using generic wffs and Modus Ponens, any inference we make in a proof based on these derived rules will be valid.

We already have Modus Ponens and Modus Tollens:

Modus Ponens: $\{\alpha \to \beta, \alpha\} \vdash \beta$ (Our main inference.)

Modus Tollens: $\{\alpha \to \beta, \neg\beta\} \vdash \neg\alpha$ (derived above)

Many of the rest of our rules of inference will come in pairs based on the commutativity of $\vee$ and $\wedge$.

Disjunctive Syllogism (or $\vee$-Elimination): $\{\alpha \vee \beta, \neg\alpha\} \vdash \beta$

1. $\alpha \vee \beta$	Premise
2. $\neg\alpha$	Premise
3. $(\alpha \vee \beta) \rightarrow (\neg\alpha \rightarrow \beta)$	Tautology
4. $\neg\alpha \rightarrow \beta$	Modus Ponens 3,1
5. β	Modus Ponens 4,2

Because of the commutativity of $\vee$, we also have this form of Disjunctive Syllogism:

Disjunctive Syllogism (or $\vee$-Elimination): $\{\alpha \vee \beta, \neg\beta\} \vdash \alpha$

Simplification (or $\wedge$-elimination): $\{\alpha \wedge \beta\} \vdash \alpha$

1. $\alpha \wedge \beta$	Premise
2. $(\alpha \wedge \beta) \rightarrow \alpha$	Tautology
3. α	Modus Ponens 2,1

Because of the commutativity of $\wedge$, we also have this form of Simplification:

Simplification (or $\wedge$-elimination): $\{\alpha \wedge \beta\} \vdash \beta$

Addition (or $\vee$-Introduction): $\alpha \vdash \alpha \vee \beta$

1. α	Premise
2. $\alpha \rightarrow (\alpha \vee \beta)$	Tautology
3. $\alpha \vee \beta$	Modus Ponens 2,1

Because of the commutativity of $\vee$, we also have this form of Addition:

Addition (or $\vee$-Introduction): $\beta \vdash \alpha \vee \beta$

Conjunction (or $\wedge$-Introduction): $\{\alpha, \beta\} \vdash \alpha \wedge \beta$

1. α	Premise
2. β	Premise
3. $\alpha \to (\beta \to (\alpha \wedge \beta))$	Tautology
4. $\beta \to (\alpha \wedge \beta)$	Modus Ponens 1,3
5. $\alpha \wedge \beta$	Modus Ponens 2,4

Equivalence: If $\alpha \equiv \beta$, then $\alpha \vdash \beta$.

1. α	Premise
2. $\alpha \to \beta$	Tautology
3. β	Modus Ponens 1,2

Transitivity: $\{\alpha \to \beta, \beta \to \gamma\} \vdash \alpha \to \gamma$

1. $\alpha \to \beta$	Premise
2. $\beta \to \gamma$	Premise
3. $\neg\alpha \vee \beta$	Equivalent 1
4. $\neg\beta \vee \gamma$	Equivalent 2
5. $\neg\alpha \vee \beta \vee \gamma$	Addition 3
6. $\neg\alpha \vee \neg\beta \vee \gamma$	Addition 4
7. $(\neg\alpha \vee \beta \vee \gamma) \wedge (\neg\alpha \vee \neg\beta \vee \gamma)$	Conjunction 5,6
8. $(\neg\alpha \vee \gamma) \vee (\beta \wedge \neg\beta)$	Equivalent 7
9. $\neg\alpha \vee \gamma$	Equivalent 8
10. $\alpha \to \gamma$	Equivalent 9

For your convenience, here is a list of our rules of inference. We will derive more proof techniques later in Chapter 8

Rules of Inference

Modus Ponens ..... $\{\alpha \to \beta, \alpha\} \vdash \beta$
Modus Tollens ..... $\{\alpha \to \beta, \neg\beta\} \vdash \neg\alpha$
Disjunctive Syllogism (or $\vee$-Elimination) ..... $\{\alpha \vee \beta, \neg\beta\} \vdash \alpha$
Simplification (or $\wedge$-elimination) ..... $\{\alpha \wedge \beta\} \vdash \alpha$
Addition (or $\vee$-Introduction) ..... $\alpha \vdash \alpha \vee \beta$
Conjunction (or $\wedge$-Introduction) ..... $\{\alpha, \beta\} \vdash \alpha \wedge \beta$
Equivalence ..... If $\alpha \equiv \beta$, then $\alpha \vdash \beta$.
Transitivity ..... $\{\alpha \to \beta, \beta \to \gamma\} \vdash \alpha \to \gamma$

Remember that the commutativity of $\wedge$ and $\vee$ gives us symmetric versions of some of these.

8.3 Applying Rules of Inference

We demonstrate how the rules of inference can be applied repeatedly to a list of premises to eventually arrive at a desired conclusion. The list we form will be in three columns. The first column will be statements. These will be numbered so that we can refer to them. The second column will be the rules of inference we applied to get those statements. The third column will be the numbers of the statements to which the rules were applied. We begin with two premises.

1. $(P \vee Q) \rightarrow (R \wedge \neg S)$

2. $\neg R \vee S$

From these two premises we will build a chain of inferences concluding eventually with $\neg P$.

To begin with, premise (2) is logically equivalent via DeMorgan's Law to $\neg(R \wedge \neg S)$. Thus from the rule of inference **E**, we can infer $\neg(R \wedge \neg S)$. We now have a list of three statements:

	Statement	**Rule**	**Specifics**
1.	$(P \vee Q) \rightarrow (R \wedge \neg S)$	premise	
2.	$\neg R \vee S$	premise	
3.	$\neg(R \wedge \neg S)$	Equivalence	2

We can now apply Modus Tollens to statements (1) and (3) in our list to infer $\neg(P \vee Q)$. This now gives us four statements:

	Statement	**Rule**	**Specifics**
1.	$(P \vee Q) \rightarrow (R \wedge \neg S)$	premise	
2.	$\neg R \vee S$	premise	
3.	$\neg(R \wedge \neg S)$	Equivalence	2
4.	$\neg(P \vee Q)$	Modus Tollens	$1, 3$

Next, we can apply DeMorgan's Law to (4) to infer $\neg P \wedge \neg Q$ via Equivalence:

	Statement	**Rule**	**Specifics**
1.	$(P \vee Q) \rightarrow (R \wedge \neg S)$	premise	
2.	$\neg R \vee S$	premise	
3.	$\neg(R \wedge \neg S)$	Equivalence	2
4.	$\neg(P \vee Q)$	Modus Tollens	$1, 3$
5.	$\neg P \wedge \neg Q$	Equivalence	4

Statement (5) is more than we are looking for. We only want half of the

conjunction, so we apply Simplification to get the half we want.

	Statement	Rule	Specifics
1.	$(P \vee Q) \to (R \wedge \neg S)$	premise	
2.	$\neg R \vee S$	premise	
3.	$\neg(R \wedge \neg S)$	Equivalence	2
4.	$\neg(P \vee Q)$	Modus Tollens	$1, 3$
5.	$\neg P \wedge \neg Q$	Equivalence	4
6.	$\neg P$	Simplification	5

8.4 Exercises

1. Apply Modus Tollens to these premises:
   $P \to (Q \wedge R)$
   $\neg(Q \wedge R)$
2. Apply Simplification and then Modus Ponens to these two premises:
   $P \wedge Q$
   $P \to R$
3. Apply Modus Ponens and then Disjunctive Syllogism to these three premises:
   $P \to (Q \vee R)$
   P
   $\neg R$
4. Apply Addition and then Modus Ponens to these premises
   $(P \vee R) \to (S \wedge T)$
   P

8.5 Examples of Proofs

Here are a few exmaples of how to write proofs.

Example 8.3: Write a proof to show that

$$\{P \wedge Q, P \to R\} \vdash R$$

First, we write down our premises:

	Statement	Rule	Specifics
1.	$P \wedge Q$	premise	
2.	$P \to R$	premise	

Now, how can we get R? We have $P \to R$, so if we had P, we could apply Modus Ponens to get R. We can get P from statement (1) by Simplification. Thus we first add P to our list of statements using simplification:

	Statement	**Rule**	**Specifics**
1.	$P \wedge Q$	premise	
2.	$P \to R$	premise	
3.	P	Simplification	1

We can then add R by applying Modus Ponens to statements (2) and (3):

	Statement	**Rule**	**Specifics**
1.	$P \wedge Q$	premise	
2.	$P \to R$	premise	
3.	P	Simplification	1
4.	R	Modus Ponens	$2, 3$

This completes our proof. Notice how we arrived at this proof. We looked at the conclusion and decided what we might need in order to infer the conclusion. We then looked at the premises and tried to build what we needed from these using rules of inferences. We worked from both ends to arrive at a proof. This process is typical of how proofs are written.

Example 8.4: Write a proof to show that

$$\{P \to (Q \vee R), P, \neg R\} \vdash Q$$

We again begin by listing our premises:

	Statement	**Rule**	**Specifics**
1.	$P \to (Q \vee R)$	premise	
2.	P	premise	
3.	$\neg R$	premise	

We want to end up at Q. The only Q is in statement (1) after the arrow. Modus Ponens is the only rule of inference which gives us statements which appear after an arrow. The first two statements are set up for Modus Ponens, so we try it.

	Statement	**Rule**	**Specifics**
1.	$P \to (Q \vee R)$	premise	
2.	P	premise	
3.	$\neg R$	premise	
4.	$Q \vee R$	Modus Ponens	$1, 2$

Remember that we are looking for Q. This new statement says that either Q is true or R is true. If we can exclude R, then we will be there by Disjunctive

Syllogism. Statement (3) does the trick. So:

	Statement	Rule	Specifics
1.	$P \to (Q \vee R)$	premise	
2.	P	premise	
3.	$\neg R$	premise	
4.	$Q \vee R$	Modus Ponens	$1, 2$
5.	Q	Disjunctive Syllogism	$4, 3$

Do not let the fact that (4) and (3) do not come in the right order for Disjunctive Syllogism bother you. Knowing (3) and (4) is the same as knowing (4) and (3).

Example 8.5: Write a proof to show that

$$\{(P \vee R) \to (S \wedge T), P\} \vdash T$$

We need T. The only T is in the second half of the implication in the first premise. Thus if we can make the first half of the implication $(P \vee R)$ true we can get to the second half by Modus Ponens. We could then apply simplification to get to T. We know that P is true from the second premise. This is enough to get $P \vee Q$ by Addition. Thus we arrive at this proof:

	Statement	Rule	Specifics
1.	$(P \vee R) \to (S \wedge T)$	premise	
2.	P	premise	
3.	$P \vee R$	Addition	2
4.	$S \wedge T$	Modus Ponens	$1, 3$
5.	T	Simplification	4

Again notice how we worked from both ends of the proof.

Example 8.6: Write a proof to show that

$$\{P \to (Q \wedge R), \neg Q\} \vdash \neg P$$

We want $\neg P$. P shows up as the first half of an implication in the first premise. Thus we might try to use Modus Tollens. In order to do this, we need $\neg(Q \wedge R)$. We can get this by beginning with the premise $\neg Q$, adding $\neg R$ to get $\neg Q \vee \neg R$ and then applying DeMorgan's Law. Hence we have:

	Statement	Rule	Specifics
1.	$P \to (Q \wedge R)$	premise	
2.	$\neg Q$	premise	
3.	$\neg Q \vee \neg R$	Addition	2
4.	$\neg(Q \wedge R)$	Equivalence	3
5.	$\neg P$	Modus Tollens	$1, 4$

Example 8.7: Write a proof to show that

$$\{P \to Q, P \to R, P\} \vdash Q \wedge R$$

It should be clear that we can get Q and R individually from Modus Ponens. To get $Q \wedge R$, we simply need to use Conjunction:

	Statement	**Rule**	**Specifics**
1.	$P \to Q$	premise	
2.	$P \to R$	premise	
3.	P	premise	
4.	Q	Modus Ponens	$1, 3$
5.	R	Modus Ponens	$2, 4$
6.	$Q \wedge R$	Conjunction	$4, 5$

Our next example emphasizes the importance of beginning with premises that are true. From this we learn that if we assume a contradiction is true, anything can happen.

Example 8.8: Write a proof to show that

$$\{P \wedge \neg P\} \vdash Q$$

This will make more sense if we show the proof and then discuss where it came from:

	Statement	**Rule**	**Specifics**
1.	$P \wedge \neg P$	premise	
2.	P	Simplification	1
3.	$\neg P$	Simplification	1
4.	$P \vee Q$	Addition	2
5.	Q	Disjunctive Syllogism	$4, 3$

Knowing both P and $\neg P$ sets up up for using Disjunctive Syllogism on any disjunction involving P. Thus all we need to do is create a disjunction involving P and Q. This is easily done with Addition. Premises such as the ones in this argument which either contain or can be used to prove a contradiction are called **inconsistent**. Premises which cannot prove a contradiction are called **consistent**.

Example 8.9: Write a proof to show that

$$\{\alpha \to (\gamma \to \beta), \alpha \to \gamma\} \vdash \alpha \to \beta$$

1.	$\alpha \to (\gamma \to \beta)$	Premise
2.	$\alpha \to \gamma$	Premise
3.	$\neg\alpha \vee \neg\gamma \vee \beta$	Equivalent 1
4.	$\neg\alpha \vee \gamma$	Equivalent 2
5.	$(\neg\alpha \vee \neg\gamma \vee \beta) \wedge (\neg\alpha \vee \gamma)$	Conjunction 3,4
6.	$\neg\alpha \vee ((\neg\gamma \vee \beta) \wedge \gamma)$	Equivalent 5
7.	$\neg\alpha \vee ((\neg\gamma \wedge \gamma) \vee (\beta \wedge \gamma))$	Equivalent 6
8.	$\neg\alpha \vee (\beta \wedge \gamma)$	Equivalent 7
9.	$(\neg\alpha \vee \beta) \wedge (\neg\alpha \vee \gamma)$	Equivalent 8
10.	$\neg\alpha \vee \beta$	Simplification 9
11.	$\alpha \to \beta$	Equivalent 10

8.6 Exercises

Prove the following deductions:

1. $\{(\neg P \vee R) \to \neg Q, Q\} \vdash P \wedge \neg R$
2. $\{\neg P \to (Q \vee R), \neg P \wedge S, \neg R\} \vdash Q$
3. $\{P \to (R \wedge S), \neg R\} \vdash \neg P$
4. $\{P \to Q, \neg(Q \vee R)\} \vdash \neg P$
5. $\{P \to (Q \wedge R), R \to S, P\} \vdash S$
6. $\{(P \vee R) \to Q, S \to P, S\} \vdash Q$
7. $\{P \vee Q) \to R, \neg R, T \to Q\} \vdash \neg T$
8. $\{(P \vee \neg Q) \to R, S \to (T \wedge U), S \wedge P\} \vdash R \wedge U$
9. $\{(P \wedge \neg Q) \to R, S \to P, Q \to \neg U, U \wedge S\} \vdash R$
10. $\{(P \to Q) \to R, \neg(Q \vee R)\} \vdash P$
11. $\{(P \to Q) \to (P \to R), Q \wedge P\} \vdash R$
12. $\{P \to (Q \to R), Q\} \vdash P \to R$
13. $\{A \to B, C \to D, A \vee C\} \vdash B \vee D$
14. $\{A \vee B, \neg A \vee C\} \vdash B \vee C$
15. In this section, we used Modus Ponens as our fundamental rule of inference. This is (almost) arbitrary. In this exercise, you are to prove that we could instead have used $\wedge$-introduction and $\wedge$-elimination.

    Suppose that we had defined a deduction in this manner: A deduction of α from Σ is a sequence of wffs so that α is the last member of the sequence and so that each wff in the sequence

    - Is in Σ, or

- Is a tautology, or
- Is **equivalent** to an earlier wff in the sequence, or
- Follows from two earlier wffs in the sequence by $\wedge$-introduction, or
- Follows from an earlier wff by $\wedge$-elimination.

Provide such a deduction for $\{\alpha \to \beta, \alpha\} \vdash \beta$.

16. Is it possible to derive Modus Ponens from $\vee$-introduction and $\vee$-elimination as we did in the previous exercise for $\wedge$?

Chapter 9

Soundness, Completeness, and Compactness

We have introduced two possible manners in which one may conclude the truth of a sentence α from the truth of a collection Σ of sentences.

In Chapter 5 we saw the idea of logical implication and said that $\Sigma \models \alpha$ if every truth assignment that makes the sentences in Σ true also makes α true. This is a *semantic* notion based on the meanings of the sentences involved and on truth.

In Chapter 8, we studied the idea of deducibility and proof. We defined $\Sigma \vdash \alpha$ to mean that there is a proof of α from Σ based on the single rule of inference Modus Ponens. This is a *syntactic* notion based on mechanical rules of derivation.

In this chapter, we address the relationship between these two notion. On the surface, the two ideas seem quite different. Aside from the fact that one idea is about meaning and one is about mechanical rules, proof is a finite notion – a proof is a finite list of wffs each involving a finite number of symbols – while logical implication may involve infinitely many statements. In this chapter, we discuss the somewhat surprising fact that these two different notions of implication are equivalent. This means that our notion of proof is *sound* in the sense that any statement proven from true premises must also be true. It also means that our notion of proof is *complete* in the sense that every logical implication can be supported by a proof. For mathematicians, who spend their time trying to establish the truth of theorems which are usually logical implications, this means that proof is the proper tool to use.

9.1 Soundness and Completeness

Recall the definition of proof from Chapter 8

Proof Suppose that Σ is a set of wffs and α is a wff. A *proof* or *deduction* of α from Σ is a list $\gamma_1, \gamma_2, \ldots, \gamma_n$ so that $\gamma_n = \alpha$ and each sentence γ_i in the list

1. is a tautology or
2. is in Σ or
3. follows from two earlier sentences by the rule Modus Ponens.

We can use this definition to prove that if every sentence in Σ is true, and if $\Sigma \vdash \alpha$, then α must also be true. An informal way of saying this is that "proof preserves truth." This is called the *Soundness Theorem*. Suppose that we have a proof $\gamma_1, \gamma_2, \ldots, \gamma_n$ from a set Σ of wffs and that all of the wffs in Σ are true. If some γ_i is in the proof because of reason (1), then it is a tautology and is true. If some γ_i is in the proof because of reason (2), then it is in Σ and is true because everything in Σ is true. Finally, if some γ_i is in the proof because of reasosn (3), then it is true because Modus Ponens preserves truth, as in example 5.1 in Chapter 5. Thus every statement in the proof – including the conclusion γ_n – must be true. We have:

Theorem 9.1: Soundness Theorem:

If $\Sigma \vdash \beta$, then $\Sigma \models \beta$.

A rigorous proof of this theorem would require induction based on a recursive characterization of provability as discussed in Chapter 13.

The converse of the Soundness Theorem is known as the Completeness Theorem. This theorem tells us that our deductive system is "complete" or strong enough to prove all valid logical implications. We can observe a restricted version of the Completeness Theorem in which Σ is finite easily.

Theorem 9.2: Finite Completeness Theorem:

If $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \models \beta$, then $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vdash \beta$.

If $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \models \beta$, then $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \rightarrow \beta$ is a tautology by exercise 13 in 5.4, so we can offer this proof of β from $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$.

1. α_1	Premise
2. α_2	Premise
$\vdots$	$\vdots$
n. α_n	Premise
$n+1$. $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n)$	Conjunction
$n+2$. $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \rightarrow \beta$	Tautology
$n+3$. β	Modus Ponens

9.2 The Deduction Theorem

The most fundamental proof technique in mathematics is Direct Proof, which will be discussed in Chapter 10. The basis for this proof technique is the Deduction Theorem, which we can now establish with the help of the Soundness Theorem. You should compare this theorem to Example 5.6.

Theorem 9.3: Deduction Theorem:

$\Sigma, \alpha \vdash \beta$ if and only if $\Sigma \vdash \alpha \to \beta$.

Since proof is a finite notion, it is adequate to consider the case when Σ is finite:

$\{\gamma_1, \gamma_2, \ldots, \gamma_n, \alpha\} \vdash \beta$ if and only if $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} \vdash \alpha \to \beta$.

For this discussion, let $\sigma = \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n$.

- If $\{\gamma_1, \gamma_2, \ldots, \gamma_n, \alpha\} \vdash \beta$, then $(\sigma \wedge \alpha) \vdash \beta$ by the rule of inference Simplification.
- If $(\sigma \wedge \alpha) \vdash \beta$, then $(\sigma \wedge \alpha) \models \beta$ by the Soundness Theorem.
- If $(\sigma \wedge \alpha) \models \beta$, then $(\sigma \wedge \alpha) \to \beta$ is a tautology (see Exercise 11 in Section 5.4).
- If $(\sigma \wedge \alpha) \to \beta$ is a tautology, then $\sigma \to (\alpha \to \beta)$ is a tautology because these statements are equivalent.
- If $\sigma \to (\alpha \to \beta)$ is a tautology, then $\sigma \vdash (\alpha \to \beta)$ by Modus Ponens.
- If $\sigma \vdash (\alpha \to \beta)$ then $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} \vdash (\alpha \to \beta)$ by the rule of inference Conjunction.

Applying the rule of inference Transitivity to these implications gives

If $\{\gamma_1, \gamma_2, \ldots, \gamma_n, \alpha\} \vdash \beta$ then $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} \vdash \alpha \to \beta$.

On the other hand, the implication

If $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} \vdash \alpha \to \beta$ then $\{\gamma_1, \gamma_2, \ldots, \gamma_n, \alpha\} \vdash \beta$

follows immediately from Modus Ponens.

The Deduction Theorem provides an important means for demonstrating that an implication $\alpha \to \beta$ is provable. To demonstrate such, simply assume α and use this to prove β. This is known as Direct Proof, and we will explore this technique in Chapter 10. Note that Direct Proof does not actually provide a proof of $\alpha \to \beta$, it simply argues that a proof exists.

9.3 Consistency and Satisfiability

The Completeness and Soundness Theorems so far tell us that if Σ is *finite* then $\Sigma \models \beta$ and $\Sigma \vdash \beta$ are equivalent. However, what if Σ is infinite? Is it possible that assuming infinitely many truths somehow could surpass the power of proof and imply logical consequences that cannot be directly proven? The answer is no, but we will have to wait for the Compactness Theorem 9.8 to see why. For the Compactness Theorem, we need the notions of consistency and satisfiability. We first look at an example which has a moral.

Example 9.4: Write a proof to show that

$$P \wedge \neg P \vdash Q$$

1. $P \wedge \neg P$	Premise
2. P	Simplification 1
3. $\neg P$	Simplification 1
4. $P \vee Q$	Addition 2
5. Q	Disjunctive Syllogism 4,3

The moral is: anything can be proven from a contradiction. For this reason, we make the following definition. A set Σ of wffs is *consistent* if no contradiction can be proven from Σ. Σ is *inconsistent* if a contradiction (such as $\alpha \wedge \neg\alpha$) can be proven from Σ. From the previous example, any wff can be proven from an inconsistent set of wffs.

Theorem 9.5: If Σ is inconsistent, then $\Sigma \vdash \alpha$ for any wff α.

Suppose that Σ is inconsistent. Then for any wff α in the same language, $\Sigma \vdash \alpha \wedge \neg\alpha$. This implies that $\Sigma \models \alpha \wedge \neg\alpha$. If a truth assignment s made every wff in Σ true, then Soundness would give us $s(\alpha \wedge \neg\alpha) = T$. This cannot happen. Thus, there is no truth assignment that will make all of the wffs in Σ true. Thus we define: A set Σ of wffs is *satisfiable* if there is a truth assignment s so that $s \models \Sigma$. What we just observed is that if Σ is not consistent, then Σ is not satisfiable. The contrapositive of this is:

Theorem 9.6: Soundness Theorem (Version 2): If Σ is satisfiable, then Σ is consistent.

We have named this as Version 2 of the Soundness Theorem because it happens to be equivalent to Version 1 of the Soundness Theorem above. We will not prove this equivalence.

We also have a second (and equivalent) version of the Completness Theorem. At this point, we can only address the finite version of this theorem.

Theorem 9.7: Finite Completeness Theorem (Version 2): If $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is consistent, then $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is satisfiable.

To see this, we will argue that if $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is not satisfiable, then $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is not consistent. Let α be any wff. Since $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is not satisfiable, then for any truth assignment s, it is not the case that $s \models \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Therefore, the implication

$$\text{If } s \models \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \text{ then } s \models (\alpha \wedge \neg\alpha)$$

is vacuously true (since it is of the form $F \rightarrow F$). This means that

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \models (\alpha \wedge \neg\alpha).$$

By Version 1 of the Finite Completeness Theorem,

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vdash (\alpha \wedge \neg\alpha).$$

But then, $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is inconsistent. We see that if $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is not satisfiable, then $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is not consistent. This is the contrapositive of Version 2 of the Finite Completeness Theorem.

Since proof is a finite notion, and since consistency is defined in terms of proof, it seems that this version of the Finite Completeness Theorem leaves open the possibility that an infinite set Σ of wffs might be consistent (no finite subset of them can prove a contradiction), but taken as a whole they may not be satisfiable. This is not possible. To prove this, we need the Compactness Theorem 9.8.

9.4 Exercises

At least in the finite case, consistency and satisfiability are equivalent. We can use this to determine if a set of wffs is consistent. If we can find truth values for the atomic symbols involved which make all of the wffs true, then the set of wffs is satisfiable and consistent. If a contradiction can be proven from the wffs, then the set of wffs is not consistent and not satisfiable.

Decide if each the following sets of wffs is or is not consistent/satisfiable. For those that are satisfiable, provide an assignment of truth values which makes all of the wffs true. For those that are not satisfiable, provide a proof from the wffs of a contradiction.

1. $\{P \rightarrow \neg Q, P \wedge Q\}$
2. $\{A \rightarrow B, C \rightarrow D, A \vee C, \neg B\}$
3. $\{A \vee B, \neg A \vee C, \neg(B \vee C)\}$
4. $\{A \rightarrow B, B \rightarrow C, A \wedge \neg C\}$

5. $\{(A \wedge \neg A) \vee B\}$
6. $\{A \rightarrow B, (A \wedge B) \rightarrow C, D \rightarrow C, \neg D\}$
7. $\{A \rightarrow B, B \rightarrow A, A \wedge \neg B\}$
8. $\{(A \wedge B) \rightarrow C, A \rightarrow \neg C, B \rightarrow A, B\}$
9. $\{(A \wedge \neg B) \rightarrow C, (\neg A \wedge B) \rightarrow \neg C, A \vee B\}$
10. $\{A \rightarrow \neg A, B \rightarrow \neg B, A \vee B \vee C\}$

9.5 Compactness

Compactness is a notion that essentially declares that logical implication and satisfiability (like proof and consistency) are finite concepts. The Compactness Theorem comes in two equivalent forms like the Soundness and Completeness Theorems. To keep our system of naming these theorems parallel, the first version of the Compactness Theorem we will see will actually be called Version 2. Call a set Σ of wffs *finitely satisfiable* if every finite set of wffs in Σ is satisfiable.

Theorem 9.8: Compactness Theorem (Version 2): If Σ is finitely satisfiable, then Σ is satisfiable.

We will not attempt a proof of this version of the Compactness Theorem here, but we will describe a rough approach. Suppose that Σ is a finitely satisfiable set of wffs in a sentential language L. List all of the wffs in the language as $\alpha_1, \alpha_2, \ldots$.[1] We consider each α_n one at a time. Either Σ, α_n is finitely satisfiable, or $\Sigma, \neg\alpha_n$ is finitely satisfiable.[2] If Σ, α_n is finitely satisfiable, then add α_n to Σ to form a larger set of wffs. Otherwise, add $\neg\alpha_n$ to Σ. Do this for all $\alpha_1, \alpha_n, \ldots$ to form a new finitely satisfiable set Γ of wffs which contains Σ so that for every wff α, Γ contains either α or $\neg\alpha$ (such a Γ is said to be *maximally consistent* because adding any α to Γ would make it inconsistent). Since Γ is a set of wffs, some of the atomic symbols in our language may be in Γ, while some may not be. Define a truth assignment s so that for any atomic symbol A, if A is in Γ then $s(A) = T$. Otherwise $S(A) = F$. It can be proven that for any wff α, $s(\alpha) = T$ if and only if α is in

[1] This approach relies on the language in question being countable. Zorn's Lemma can be used to avoid this, but that is beyond the intended scope of these notes.

[2] Suppose not. Then there are finite sets Γ_1 and Γ_2 of wffs from Σ_n so that Γ_1, α_n and $\Gamma_2, \neg\alpha_n$ are not satisfiable. If a truth assignment s made all of the wffs in Γ_1 and Γ_2 true, then $s(\alpha_n)$ or $s(\neg\alpha_n)$ would have to be true – thus satisfying one of Γ_1, α_n and $\Gamma_2, \neg\alpha_n$. Neither of these can happen, so Γ_1 and Γ_2 cannot be satisfied together. Therefore, Γ_1 and Γ_2 together would form a finite unsatisfiable set of wffs from Σ_n.

Γ. In particular, $s \models \Gamma$, so $s \models \Sigma$. With the right details, this construction will prove this version of the Compactness Theorem.

We can use the Satisfiability version of the Compactness Theorem to prove a version of the Compactness Theorem (which happens to be equivalent) that refers to logical implication. Suppose that $\Sigma \models \alpha$. This implies that $\Sigma, \neg\alpha$ is not satisfiable (since α and $\neg\alpha$ cannot be true at the same time). By Version 2 of the Compactness Theorem, there is some finite set Σ_0 of wffs in Σ so that $\Sigma_0, \neg\alpha$ is not satisfiable. But then $\Sigma_0, \neg\alpha \models \alpha$ vacuously. By Example 5.6 of Chapter 5, $\Sigma_0 \models \neg\alpha \rightarrow \alpha$. Since $(\neg\alpha \rightarrow \alpha) \equiv \alpha$, we have $\Sigma_0 \models \alpha$. This proves the following version of the Compactness Theorem.

Theorem 9.9: Compactness Theorem (Version 1): If $\Sigma \models \alpha$, then there is some finite set Σ_0 of wffs in Σ so that $\Sigma_0 \models \alpha$.

9.6 Completeness

We are now ready to use the Compactness Theorems to extend our Finite Completeness Theorems to arbitrary sets. First, if $\Sigma \models \beta$, then by Version 1 of the Compactness Theorem, $\Sigma_0 \models \beta$ for some finite set Σ_0 of wffs in Σ. By Version 1 of the Finite Completeness Theorem, $\Sigma_0 \vdash \beta$. Since every wff in Σ_0 is in Σ, this implies $\Sigma \vdash \beta$. We have

Theorem 9.10: Completeness Theorem (Version 1):
If $\Sigma \models \beta$, then $\Sigma \vdash \beta$.

The "consistent implies satisfiable" version of the Completeness Theorem is proven using Version 2 of Compactness. Suppose that Σ is consistent. Then every finite set of wffs in Σ is consistent (since proofs only involve finitely many sentences). By Version 2 of the Finite Completeness Theorem, this means that every finite set of wffs in Σ is satisfiable. Thus Σ is finitely satisfiable. By Version 2 of the Compactness Theorem, Σ is satisfiable. Thus we have

Theorem 9.11: Completeness Theorem (Version 2): If Σ is consistent, then Σ is satisfiable.

9.7 Summary

We now have proven two theorems each referring to Soundness, Completeness, and Compactness. We list the pairs of theorems together here.

Theorem 9.12: Soundness Theorems

- If $\Sigma \vdash \alpha$, then $\Sigma \models \alpha$.
- If Σ is satisfiable, then Σ is consistent.

Theorem 9.13: Completeness Theorems

- If $\Sigma \models \alpha$, then $\Sigma \vdash \alpha$.
- If Σ is consistent, then Σ is satisfiable.

Theorem 9.14: Compactness Theorems

- If $\Sigma \models \alpha$, then there is some finite set Σ_0 of wffs in Σ so that $\Sigma_0 \models \alpha$.
- If Σ is finitely satisfiable, then Σ is satisfiable.

The first Soundness Theorem and the first Completeness Theorem tell us that implication and provability are equivalent ideas. The second Soundness Theorem and second Completeness Theorem tell us that satisfiability and consistency are equivalent. The Compactness Theorems tell us that these are finite notions.

Chapter 10

Temporary Assumptions

The formal deduction scheme defined in Chapter 8 allowed only Modus Ponens as a rule of inference. We simplified writing proofs in this environment by deriving other rules of inference based on Modus Ponens. We will simplify deductions further in this chapter through a method called Direct Proof. This method is based on the Deduction Theorem (Theorem 9.3).

10.1 Direct Proof

Recall the Deduction Theorem:

Theorem 10.1: Deduction Theorem (9.3):

$$\Sigma, \alpha \vdash \beta \text{ if and only if } \Sigma \vdash \alpha \to \beta.$$

To prove an implication $\alpha \to \beta$, we can *assume* α and use this to establish β. By the Deduction Theorem, this is good enough to demonstrate that $\alpha \to \beta$ could have been derived with Modus Ponens. We call this method of proof *Direct Proof*

Direct Proof: To prove $\alpha \to \beta$, assume α and use this to establish β. Conclude $\alpha \to \beta$.

When we introduce a temporary assumption for direct proof, we will lead every line in our proof in which we use that assumption with a vertical line to indicate a temporary assumption is "in play." The statements which are bound by the vertical line cannot be referenced outside of the vertical line.

Example 10.2: Write a proof to show that

$$\{P \to Q, P \to R\} \vdash P \to (Q \wedge R)$$

1.	$P \to Q$	Premise
2.	$P \to R$	Premise
3.	$\quad P$	Temp. Assumption
4.	$\quad Q$	Modus Ponens 1,3
5.	$\quad R$	Modus Ponens 2,3
6.	$\quad Q \wedge R$	Conjunction 4,5
7.	$P \to (Q \wedge R)$	Direct Proof 3-6

We can even nest temporary assumptions:

Example 10.3: Write a proof to show that

$$\{P \to Q, (Q \wedge R) \to S\} \vdash P \to (R \to S)$$

1.	$P \to Q$	Premise
2.	$(Q \wedge R) \to S$	Premise
3.	$\quad P$	Temp. Assumption
4.	$\quad Q$	Modus Ponens 1,3
5.	$\quad\quad R$	Temp. Assumption
6.	$\quad\quad Q \wedge R$	Conjunction 4,5
7.	$\quad\quad S$	Modus Ponens 2,6
8.	$\quad R \to S$	Direct Proof 5-7
9.	$P \to (R \to S)$	Direct Proof 3-8

10.2 Proving a Disjunction

Suppose that we want to prove a disjunction $P \vee Q$. Note that by disjunctive implication, $P \vee Q \equiv \neg P \to Q$. We can prove $\neg P \to Q$ by Direct Proof. Thus, a strategy to prove $P \vee Q$ is

Disjunction Proof: To prove $\alpha \vee \beta$, assume $\neg\alpha$. Use this to establish β. Conclude $\alpha \vee \beta$.

For example:

Example 10.4: Write a proof to show that

$$\{A \to B, C \to D, A \vee C\} \vdash B \vee D$$

1.	$A \to B$	Premise
2.	$C \to D$	Premise
3.	$A \vee C$	Premise
4.	$\neg B$	Temp. Assumption
5.	$\neg A$	Modus Tollens 1,4
6.	C	Disjunctive Syllogism 3,5
7.	D	Modus Ponens 2,6
8.	$B \vee D$	Disjunction Proof 4-8

10.3 Cases

Suppose we want to prove a statement of the form $(P \vee Q) \to R$. From Example 6.6, we know that $(P \vee Q) \to R \equiv (P \to R) \wedge (Q \to R)$. Thus, it is good enough to prove that $P \to R$ and $Q \to R$. This is known as proof by cases.

Proof by Cases: To prove $(\alpha \vee \beta) \to \gamma$, Prove $\alpha \to \gamma$ and $\beta \to \gamma$. Conclude $(\alpha \vee \beta) \to \gamma$

For example:

Example 10.5: Write a proof to show

$$\{\neg R \to \neg(P \vee S), Q \to (R \wedge T)\} \vdash (P \vee Q) \to R$$

1.	$\neg R \to \neg(P \vee S)$	Premise
2.	$Q \to (R \wedge T)$	Premise
3.	P	Temp. Assumption
4.	$P \vee S$	Addition 3
5.	R	Modus Tollens 1, 4
6.	$P \to R$	Direct Proof 3-5
7.	Q	Temp. Assumption
8.	$R \wedge T$	Modus Ponens 2,7
9.	R	Simplification 8
10.	$Q \to R$	Direct Proof 7-9
11.	$(P \vee Q) \to R$	Cases 6,10

10.4 Contrapositive

When we are trying to prove an implication $P \to Q$, it is sometimes easier to prove the contrapositive $\neg Q \to \neg P$. For example, proving "If n^2 is even, then n is even" is difficult; however, it is easy to prove "If n is not even, then n^2 is not even." Note that we can use direct proof to establish the contrapositive.

Example 10.6: Write a proof to show that

$$\{\neg R \to \neg(P \vee S)\} \vdash P \to R$$

1.	$\neg R \to \neg(P \vee S)$	Premise
2.	$\neg R$	Temp. Assumption
3.	$\neg(P \vee S)$	Premise
4.	$\neg P \wedge \neg S$	Equivalent 3
5.	$\neg P$	Simplification 4
6.	$\neg R \to \neg P$	Direct Proof 2-5
7.	$P \to R$	Equivalent (Contrapositive) 6

10.5 Contradiction

Suppose that we are trying to prove a sentence α. Suppose further that by assuming $\neg\alpha$ we can prove a contradiction β (a statement which is false for every truth assignment). Since we have proven $\neg\alpha \to \beta$, we know the contrapositive $\neg\beta \to \alpha$ must also hold. Since $\neg\beta$ is a tautology, then it follows that α must hold. This is known as *indirect proof, reductio ad absurdum*, or *proof by contradiction.*

Proof by Contradiction: To prove α by contradiction, assume $\neg\alpha$. Use this to establish a contradiction such as $\beta \wedge \neg\beta$. Conclude α.

An example:

Example 10.7: Write a proof to show that

$$\{A \to B, C \to D, A \vee C\} \vdash B \vee D$$

1.	$A \to B$	Premise
2.	$C \to D$	Premise
3.	$A \vee C$	Premise
4.	$\quad \neg(B \vee D)$	Temp. Assumption (BWOC)
5.	$\quad \neg B \wedge \neg D$	Equivalent 4
6.	$\quad \neg B$	Simplification 5
7.	$\quad \neg D$	Simplification 5
8.	$\quad \neg A$	Modus Tollens 1,6
9.	$\quad \neg C$	Modus Tollens 2,7
10.	$\quad \neg A \wedge \neg C$	Conjunction 8,9
11.	$\quad \neg(A \vee C)$	Equivalent 10
12.	$\quad (A \vee C) \wedge \neg(A \vee C)$	Conjunction 3,11
13.	$(B \vee D)$	Contradiction 4-12

10.6 Exercises

Write proofs for these deductions.

1. $\{P \to \neg Q, Q \vee R, (R \vee S) \to T\} \vdash P \to T$
2. $\{P \to S, Q \to T, (S \wedge T) \to R\} \vdash (P \wedge Q) \to R$
3. $\{P, Q \to R\} \vdash (P \to Q) \to R$
4. $\{P \to R, \neg S \to P\} \vdash R \vee S$
5. $\{P \to R, \neg T \to Q, (R \wedge T) \to (S \wedge U)\} \vdash (P \to Q) \vee S$
6. $\{A, (B \wedge C) \to D\} \vdash (A \wedge \neg B) \vee (C \to D)$
7. $\{P \vee S, Q \vee U, (S \wedge U) \to R\} \vdash P \vee Q \vee R$
8. $\{\neg P \vee R, (Q \vee S) \to R\} \vdash (P \vee Q) \to R$
9. $\{\neg A \vee C, (A \vee E) \to F, F \to D, B \to F, \neg B \vee A\} \vdash (A \vee B) \to (C \wedge D)$
10. $\{\neg D \to \neg C, S \to T, (B \vee \neg S) \to D, \neg T, (A \wedge \neg S) \to D\} \vdash (A \vee B \vee C) \to D$

Chapter 11

Arguments

An *argument* is a list of statements called *premises* followed by a statement called the *conclusion*.

The argument with premises $P_1, P_2, \ldots, P_n$ and conclusion C can be written as

$$P_1, P_2, \ldots, P_n \therefore C$$

or as

$$\begin{array}{l} P_1 \\ P_2 \\ \vdots \\ \underline{P_n} \\ \therefore C \end{array}$$

The three dots $\therefore$ symbolize "therefore." This is not a logical symbol. It is a symbol to emphasize the conclusion.

An argument $P_1, P_2, \ldots, P_n \therefore C$ is *valid* if $\{P_1, P_2, \ldots, P_n\} \models C$. That is, the argument is valid if the premises logically imply the conclusion. Otherwise, the argument is *invalid*.

By the Completeness and Soundness Theorems, an argument is valid if and only if it has a proof. To demonstrate that an argument is valid, we can write a proof for the argument. To demonstrate that an argument is invalid, we can either draw a truth table or exhibit a truth assignment that makes the premises true and the conclusion false.

Example 11.1: Demonstrate that this argument is invalid:

$$\begin{array}{l} P \to Q \\ \underline{Q} \\ \therefore P \end{array}$$

We could use the truth table approach. For this approach, we draw a truth table that includes all of the premises and the conclusion of the argument.

We then note that there is a row in which all of the premises are true but the conclusion is false. A truth table that works for this argument is:

P	Q	$P \to Q$
T	T	T
T	F	F
F	**T**	**T**
F	F	T

Note that in the third line of this table, both premises Q and $P \to Q$ are true but the conclusion P is false. Thus the premises do not logically imply the conlcusion. The argument is invalid. We do not need to draw the entire truth table to show that this argument is invalid. It is enough to note that if P is F and Q is T then the premises are true but the conclusion is false.

11.1 Exercises

Show each of these arguments is invalid.

1. $P \to Q, \neg P \therefore \neg Q$
2. $P \to Q, P \to R \therefore Q \to R$
3. $(P \wedge Q) \to R, P \therefore R$
4. $(P \wedge Q) \to R, \neg R, \therefore \neg P$

For each of the arguments in words below, introduce appropriate atomic sentence symbols and translate the argument into sentential logic. Then either write a proof for the argument to show that it is valid or find truth values demonstrating that the argument is invalid.

5. Jeremy will buy either the red pillow or the blue pillow. The blue pillow is quite comfortable. If he buys it, he will sleep in and miss his meeting with the President. If Jeremy misses the meeting with the President, the Cabinet will be misled by Jeremy's adversaries. Either the Cabinet will not be misled, or we will go to war. Therefore, either Jeremy will buy the red pillow or we will go to war.

6. If Paul buys one more piece of candy, he will not be able to afford to buy both his math book and his history book. Paul must major in either math or government or he will lose his scholarship. If Paul loses his scholarship, he will drop out of college and will never become a lawyer or a doctor. If Paul does not buy his math book, he will fail math and will not be able to major in math. If he does not buy his history book, he will not be able to major in government. Therefore, if Paul does not buy one more piece of candy, he will either become a lawyer or a doctor.

7. Sue is good at basket weaving and at logic. If she majors in basket weaving, she will always be able to find a job in a remote village of South America. If she majors in logic, she will at least be able to work at Wal-Mart. Since Sue is good at logic, she is not good at basket weaving. Therefore, if Sue majors in basket weaving and in logic, then she will one day sell her baskets in a Wal-Mart in South America.

8. Either fuzzy glow worms eat yellow leaves, or they eat green leaves. Either fuzzy glow worms are green, or the leaves they eat are green. If fuzzy glow worms eat yellow leaves, then they turn orange. Either fuzzy glow worms are not orange, or they are eaten by large purple birds. If large purple birds eat fuzzy glow worms, then large purple birds exist. Therefore, either large purple birds exist, or fuzzy glow worms eat green leaves.

11.2 Fallacies

There are mistakes that are frequently made in arguments in which people invoke supposed rules of inference which look similar to our rules of inference above but which are actually invalid arguments. Short invalid arguments such as these are called **logical fallacies**. Some common fallacies are listed below.

11.2.1 Affirming the Consequent

This fallacy is also called the **fallacy of the converse**. It assumes an implication and the consequent of that implication and then invokes Modus Ponens not on the stated implication but on its converse.

$$\begin{array}{l} P \to Q \\ Q \\ \hline \therefore P \end{array}$$

11.2.2 Denying the Antecedent

This fallacy is also called the **fallacy of the inverse**. It assumes an implication and the negation of the antecedent of that implication and then invokes Modus Ponens not on the stated implication but on its inverse.

$$\begin{array}{l} P \to Q \\ \neg P \\ \hline \therefore \neg Q \end{array}$$

11.2.3 Affirming a Disjunct

This fallacy stems from confusing an inclusive or with an exclusive or.

$$\begin{array}{l} P \vee Q \\ P \\ \hline \therefore \neg Q \end{array}$$

The fields of philosophy and rhetoric provide a rich study of other fallacies which are arguments that are problematic for reasons other than their logical form.

11.3 Exercises

For each fallacy listed above, find or make two examples – one in which the conclusion is false and one in which the conclusion is true.

Chapter 12

Three Valued Logic

Before we close our look at sentential logic, we consider systems in which there are more than two truth values. Various forms of logic with three truth values have been suggested over the centuries beginning with Aristotle. The third truth value has been described somtimes as unknown, unknowable, or maybe. In these exercises, we select truth values to extend sentential logic to three truth values. Note that there is more than one way to do this!

12.1 Exercises

1. Draw truth tables for the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$ using three truth values T, M, and F. Your tables should agree with the tables in 4.2 in the rows with Ts and Fs. The first two columns of your truth tables with two letters should look like:

α	β
T	T
T	M
T	F
M	T
M	M
M	F
F	T
F	M
F	F

   Explain which truth values require some thought and why you made the decision you did.

2. Which logical equivalences from sentential logic still hold in your logic with three variables?

3. Does this inference hold in your logic?

If $\alpha \to \beta$ and α are T, then β is M.

4. Does this inference hold in your logic?

   If $\alpha \to \beta$ and α are M, then β is M.

5. Does this inference hold in your logic?

   If $\alpha \to \beta$ and α are M or T, then β is M or T.

6. Which of the above rules of inferences seems to you to be the most desireable extension of Modus Ponens?

7. - Is it true that $\alpha \vee \neg\alpha$ is always T in your logic?
   - Is it true that $\alpha \wedge \neg\alpha$ is always F in your logic?
   - Can you adjust your truth values to make these eqivalences hold?

8. Draw truth tables for the connectives $\neg$, $\wedge$, $\vee$, $\to$, and $\leftrightarrow$ using four truth values T, P, N, and F. Your tables should agree with the tables in 4.2 in the rows with Ts and Fs. The first two columns of your truth tables with two letters should look like:

α	β
T	T
T	P
T	N
T	F
P	T
P	P
P	N
P	F
N	T
N	P
N	N
N	F
F	T
F	P
F	N
F	F

   Explain which truth values require some thought and why you made the decision you did.

9. Which logical equivalences from sentential logic still hold in your logic with four variables?

10. Does this inference hold in your logic?

    If $\alpha \to \beta$ and α are P or T, then β is P or T.

11. Does this inference hold in your logic?

    If $\alpha \to \beta$ and α are not false, then β is not false.

12.
- Is it true that $\alpha \vee \neg\alpha$ is always T in your logic?
- Is it true that $\alpha \wedge \neg\alpha$ is always F in your logic?
- Can you adjust your truth values to make these eqivalences hold?

Chapter 13

Recursion and Induction

13.1 Recursive Definitions

In Chapter 3 we gave this definition of the well formed formulas in a sentential language with a set L of basic sentence symbols:

Well-Formed Formulas over L

1. Every sentence symbol in L is a wff.
2. If α and β are wffs, then so are
$$(\neg\alpha), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), \text{ and } (\alpha \leftrightarrow \beta).$$
3. No expression can be a wff except for these reasons.

This definition is *self-referential* or *recursive* because we use wff here to define what a wff is. This type of definition is useful for a number of reasons.

- The definition allows quite complex wffs while only referring to relatively simple wffs with at most two symbols.
- The definition makes it easy to program a machine to generate wffs:
  1. Make a list containing the basic sentence symbols.
  2. For each pair of entries α and β in the list, form
  $$(\neg\alpha), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), \text{ and } (\alpha \leftrightarrow \beta)$$
  and add these to the list.
  3. Goto 2
- The definition allows one to easily define notions such as truth assignments across all wffs by only referring to basic symbols and the logical connectives.

- The definition allows for a powerful proof technique called induction that we will discuss in 13.7.

13.2 Exercises

The following exercises begin an exploration of a few "toy" languages defined recursively.

1. We will call this type of language a TL1 language (for "Toy Language 1"). If L is any set of basic sentence symbols then the wffs in the TL1 language over L are defined so that:

   - Every sentence symbol in L is a wff.
   - If α is a wff and if x is any sentence symbol in L, then $x\alpha$ is a wff.
   - No expression can be a wff except for these reasons.

   (a) List several wffs in the TL1 language with basic sentence symbols A and B.

   (b) List several wffs in the TL1 language with basic sentence symbol A.

2. Let L be any set of basic sentence symbols. The wffs in the TL2 language over L are defined so that:

   - Every sentence symbol in L is a wff.
   - If α is a wff and if x is any sentence symbol in L, then $xx\alpha$ is a wff.
   - No expression can be a wff except for these reasons.

   (a) List several wffs in the TL2 language with basic sentence symbols A and B.

   (b) Find some expressions which cannot be wffs in the TL2 language with basic sentence symbols A and B.

   (c) List several wffs in the TL2 language with basic sentence symbol A.

3. Let L be any set of basic sentence symbols. The wffs in the TL3 language over L are defined so that:

   - Every sentence symbol in L is a wff.
   - If α and β are wffs, then so are $(\neg\alpha)$ and $(\alpha \wedge \beta)$.
   - No expression can be a wff except for these reasons.

   (a) List several wffs in the TL3 language with basic sentence symbols A and B.

(b) Find some expressions which cannot be wffs in the TL3 language with basic sentence symbols A and B.

(c) List several wffs in the TL3 language over with basic sentence symbol A.

4. TL4 will be a language built from the basic symbols A and B.

   - A and ABA are wffs.
   - If α is a wff, then so are $A\alpha A$ and $\alpha\alpha$.
   - No expression can be a wff except for these reasons.

   (a) List several wffs in TL4.

   (b) Find some expressions which cannot be wffs in TL4.

5. TL5 is a language which will use two symbols 1 and S.

   - 1 is a wff.
   - If α is any wff, then $S\alpha$ is a wff.
   - That's all folks.

   (a) List several wffs in this language.

   (b) List several wffs not in this language.

13.3 Truth Assignments

One of the benefits of our recursive definition of sentential wffs is that we are able to establish truth values for an entire language by simply assigning truth values to the basic sentences involved and defining how the logical connectives affect truth values. This allows a machine to generate a list of wffs in a language along with truth values for the wffs at the same time. This should be reminiscent of the machine that Leibniz dreamed of that could determine which statements about natural numbers are true and which are false.

13.4 Exercises

The following exercises explore recursively defining "assignments" on the toy languages defined in the exercises from Section 13.2.

1. Let L be a set of basic sentence symbols. We recursively define how to assign either the letter O or the letter E to each wff in the TL1 language over L. For each wff α, we will write $s(\alpha)$ for the letter assigned to α.

Suppose that α is a wff, that x is a basic sentence symbol, and that $s(\alpha)$ and $s(x)$ have been assigned. Then

$$s(x\alpha) = \begin{cases} E & s(x) = E \text{ and } s(\alpha) = E \\ E & s(x) = O \text{ and } s(\alpha) = O \\ O & otherwise \end{cases}$$

(a) Consider the TL1 language with basic sentence symbols A and B. Suppose that $s(A) = O$ and $s(B) = E$.

i. Find $s(AA)$, $s(ABA)$, $s(BBAA)$, $s(BAB)$, and $s(AAAB)$.

ii. Guess how to determine quickly what $s(\alpha)$ is for any α.

(b) Consider the TL1 language with basic sentence symbols A and B. Suppose that $s(A) = E$ and $s(B) = E$.

i. Find $s(AA)$, $s(ABA)$, $s(BBAA)$, $s(BAB)$, and $s(AAAB)$.

ii. Guess how to determine quickly what $s(\alpha)$ is for any α.

(c) Consider the TL1 language with basic sentence symbols A and B. Suppose that $s(A) = O$ and $s(B) = O$.

i. Find $s(AA)$, $s(ABA)$, $s(BBAA)$, $s(BAB)$, and $s(AAAB)$.

ii. Guess how to determine quickly what $s(\alpha)$ is for any α.

2. Let L be a set of basic sentence symbols. We recursively define how to assign a number from $\{0, 1, 2, 3, 4, 5, \ldots\}$ to each wff in the TL2 language over L. For each wff α, we will write $n(\alpha)$ for the number assigned to α. Suppose that α is a wff, that x is a basic sentence symbol, and that $n(\alpha)$ and $n(x)$ have been assigned. Then $n(xx\alpha) = n(x) + n(x) + n(\alpha)$.

(a) Consider the TL2 language with basic sentence symbols A and B Suppose that $n(A) = 1$ and $n(B) = 2$. Find each of $n(AAA)$, $n(BBA)$, $n(BBBBB)$, and $n(AABBA)$.

(b) Consider the TL2 language with basic sentence symbols A and B Suppose that $n(A) = 1$ and $n(B) = 0$.

i. Find each of $n(AAA)$, $n(BBA)$, $n(BBBBB)$, and $n(AABBA)$.

ii. Guess how to determine quickly what $n(\alpha)$ is for any α.

3. Let L be a set of basic sentence symbols. We recursively define how to assign a color from $\{red, orange, yellow\}$ to each wff in the TL3 language over L. For each wff α, we will write $c(\alpha)$ for the color assigned to α. If α and β are wffs for which $c(\alpha)$ and $c(\beta)$ have been defined then define

$$c(\alpha \wedge \beta) = \begin{cases} red & c(\alpha) = c(\beta) = red \\ yellow & c(\alpha) = c(\beta) = yellow \\ orange & else \end{cases}$$

and

$$c(\neg\alpha) = \begin{cases} red & c(\alpha) = yellow \\ orange & c(\alpha) = orange \\ yellow & c(\alpha) = red \end{cases}$$

(a) Consider the TL3 language with basic sentence symbols R, O, and Y. Suppose that $c(R) = red$, $c(O) = orange$, and $c(Y) = yellow$. Find each of $c(\neg(R \wedge Y))$, $c((\neg R \wedge \neg O) \wedge (\neg Y))$, $c(\neg(\neg R \wedge Y))$, $c(\alpha \wedge \neg\alpha)$.

(b) Draw a "color table" for $\wedge$ and for $\neg$. The tables should be shaped like this (we abbreviate the names of the colors):

α	β	$\alpha \wedge \beta$
r	r	
r	o	
r	y	
o	r	
o	o	
o	y	
y	r	
y	o	
y	y	

and

α	$\neg\alpha$
r	
o	
y	

(c) For TL3 languages, define $\alpha \vee \beta = \neg(\neg\alpha \wedge \neg\beta)$ and $\alpha \to \beta = \neg\alpha \vee \beta$. Draw color tables for $\vee$ and $\to$.

(d) Look back at the tables you drew in the previous exercises. Which logical equivalences appear to hold in TL3 languages?

4. Recursively assign either O or E to each wff in TL5 in the following way:

- $s(1) = O$
- if $s(\alpha)$ has been defined, then

$$s(S\alpha) = \begin{cases} O & s(\alpha) = E \\ E & s(\alpha) = O \end{cases}$$

(a) Calculate $s(1)$, $s(S1)$, $s(SS1)$, $s(SSS1)$, $s(SSSS1)$, $s(SSSSS1)$, and $s(SSSSSS1)$.

(b) Make a conjecture about how to quickly determine s.

13.5 Deduction

For truth assignments, we recursively define which wffs are true or false. We had two categories being defined at once. The notion of "provable" is actually

quite similar. For any set Σ of wffs, we can define a wffs to either be provable or not provable from Σ. Based on the definition of proof in Chapter 8, we can offer this recursive characterization of those wffs provable from a set Σ of wffs.

Recursive Characterization of Wffs Provable from Σ

1. Every wff in Σ is provable from Σ.
2. Every tautology is provable from Σ.
3. If $\alpha \to \beta$ and α are provable from Σ, then β is also provable from Σ.
4. The only wffs provable from Σ are provable for one of these reasons.

There really is nothing magical about the words true, false, or provable. We could have defined wffs to be good, bad, or ugly. In the exercises below, we explore recursively defining different categories of our toy languages.

13.6 Exercises

Here, we explore recursively defining different categories of our toy languages. So as not to confuse the toy notions here with provability, we will use the word "constructible" in a manner similar to "provable" above.

1. Let L be a set of basic sentence symbols. We give a recursive definition of which TL2 wffs are constructible from a set Σ of TL2 wffs.

   - Every wff in Σ is constructible from Σ.
   - If α and β are wffs so that α and $\alpha\alpha\beta$ are constructible from Σ, then so is β.

   List all wffs in the TL2 language with basic sentence symbols A and B which are constructible from

   $$\Sigma = \{A, AAB, AABBABBA, BBAABAABBBA\}.$$

2. Let L be a set of basic sentence symbols. This exercise refers to the TL3 language over L defined in 13.2 and colorings of such languages as defined in 4.4.

   Define two wffs α and β to be equivalent if for every coloring c, $c(\alpha) = c(\beta)$. Suppose that Σ is a set of TL3 wffs over L. We give a recursive definition of of which wffs are constructible from L.

   - Every wff in Σ is constructible from Σ.

- Every wff which is equivalent to a wff constructible from Σ is constructible from Σ.
- If α and β are constructible from Σ, then so is $\alpha \wedge \beta$.
- If $\alpha \wedge \beta$ is constructible from Σ, then so is α.

(a) List some wffs which are equivalent to each of $\alpha \wedge \alpha$ and $\alpha \wedge \beta$

(b) List some wffs constructible from $\Sigma = \{A, B, C \wedge \neg D\}$

13.7 Proof by Induction

Recursive definitions give us a powerful proof technique called *induction*.

Induction Suppose that S is a set of wffs in a language L. If these two conditions are satisfied:

- Every sentence symbol is in S.
- Whenever α and β are in S, then so are

$$(\neg\alpha), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), \text{ and } (\alpha \leftrightarrow \beta).$$

Then S is the set of all wffs of L.

We take the principle of induction as an axiom (an assumption) based on our recursive definition of wffs. Here is an example of using induction to prove a fact about a sentential language. Note that this is a proof *about* sentential languages, not a proof *in* a sentential language. We sometimes will refer to the second bullet by saying that S is closed under logical connectives. Induction can be simply described by saying that if S contains the basic sentence symbols in L and if S is closed under logical connectives, then S contains all wffs over L.

Theorem 13.1: Every wff in a sentential language is equivalent to a wff which involves at most the connectives $\neg$ and $\wedge$.

We will prove this with induction. Let S be the set of all wffs in a sentential language with basic symbols L which are equivalent to a wff which involves at most the connectives $\neg$ and $\wedge$. Note that every basic sentence symbol is in S since these involve no logical connectives. Now suppose that α and β are in S. There are wffs $\hat{\alpha}$ and $\hat{\beta}$ which involve at most the connectives $\neg$ and $\wedge$ so that $\alpha \equiv \hat{\alpha}$ and $\beta \equiv \hat{\beta}$. Then $\neg\alpha \equiv \neg\hat{\alpha}$, $\alpha \wedge \beta \equiv \hat{\alpha} \wedge \hat{\beta}$, $\alpha \vee \beta \equiv \neg(\neg\hat{\alpha} \wedge \neg\hat{\beta})$, $\alpha \rightarrow \beta \equiv \neg(\hat{\alpha} \wedge \neg\hat{\beta})$, and $\alpha \leftrightarrow \beta \equiv \neg(\hat{\alpha} \wedge \neg\hat{\beta}) \wedge \neg(\hat{\beta} \wedge \neg\hat{\alpha})$. Since each of these wffs contains only basics symbols, $\neg$ and $\wedge$, they are all in S. Thus S contains the basic sentence symbols and is closed under logical connectives. By induction, S contains all wffs over L. Thus, every wff over L is equivalent to a wff which involves at most the connectives $\neg$ and $\wedge$.

Note about the word "induction": Proof by induction should not be confused with the method of reasoning called inductive reasoning which is prevalent in the sciences. *Deductive reasoning* is reasoning through logical inferences as we have been doing in the past few chapters. When one begins with true premises, any conclusion drawn by deductive reasoning is necessarily true. Science relies on inductive reasoning. *Inductive reasoning* does not prove conclusions. Rather, it gathers evidence for conclusions. When one begins with true premises, any conclusion drawn by inductive reasoning may be true, but it may also be false. Science does not provide proof, it provides evidence. Proof by induction is actually a deductive technique. It does not suffer from the weaknesses of scientific inductive reasoning.

13.8 Exercises

1. Let L be a sentential language. Let S be the set of all wffs in L with the same number of left and right parenthesis. Use induction to prove that S contains all wffs over L.
2. Let Σ be a set of wffs in a sentential language with basic symbols L. The recursive characterization of wffs provable from Σ gives rise to an induction strategy for proving facts about those wffs provable from Σ:

   Induction for Wffs Provable from Σ
   Suppose that S is a set of wffs. If these conditions are met:
   - Every wff in Σ is in S.
   - Every tautology is in S.
   - If $\alpha \to \beta$ and α are in S, then β is in S.

   then S contains the set of all wffs provable from Σ.

   We will describe this by saying S must contain all of Σ and all tautologies and must be closed under Modus Ponens.

   Let s be a truth assignment over L and suppose that $s \models \Sigma$ (that is, s makes all of Σ true). Let S be the set of all wffs which s makes true. Use the induction strategy described here to prove that every wff provable from Σ is in S. As a result, any wff provable from a set of true premises is also true.
3. Every recursive definition gives an induction strategy. Give an induction strategy based on the recursive definition of TL2 languages.
4. Give an induction strategy for TL4 languages.
5. Use induction to prove that every wff in the TL2 language with basic sentence symbol A (from 13.2) has an odd number of A's.

6. Use induction to prove that every wff in the language TL4 begins with an A.
7. This exercise refers to coloring and constructibility in TL3 languages. Give an induction strategy for those TL3 wffs constructible from a set Σ of TL3 wffs.
8. Use induction to prove that if every TL3 wff in Σ is red, then every wff constructible from Σ is red.
9. Is it true that if every TL3 wff in Σ is orange, then every wff constructible from Σ is orange?

13.9 Recursively Enumerable Sets

The recursive definitions early in this chapter defined what we called languages, but we can use recursive definitions to define many types of sets. In this section, we will consider recursive definitions which list subsets of the natural numbers: $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$. We call such sets *recursively enumerable*[1]. Each of these definitions will include the ingredients of the recursive definitions from earlier in the chapter:

- A *basis* condition for inclusion in the set. (Such as, "The symbols in L are contained in S.")
- A *transition* or *step* condition for inclusion in the set. (Such as, "If α and β are in S then so is $\alpha \wedge \beta$.") This is the condition which is usually self-referential.
- A *terminal* or *closure* condition. (Such as, "No element is in the set except for one of these reasons.")

Example 13.2: Give a recursive definition of the set E of even natural numbers.

The set E of even natural numbers can be defined recursively by:

- The number 0 is in E.
- If a number n is in E, then so is $n + 2$.
- No number is in E except for one of these reasons.

Example 13.3: Give a recursive description of the set $\mathbb{N}$ of all natural numbers.

The set $\mathbb{N}$ can be described recursively by:

[1] OK. So a lot of mathematicians would have a problem with this definition, but it works for our purposes.

- The number 0 is in $\mathbb{N}$.
- If a number n is in $\mathbb{N}$, then so is $n + 1$.
- No number is in T except for one of these reasons.

Example 13.4: Give a recursive description of the set S of natural numbers which are not multiples of 3.

- 1 and 2 are in S.
- If n is in S then so is $n + 3$.
- That's all.

Example 13.5: Give a recursive description of this set:

$$\{1, 2, 3, 5, 8, 13, 21 \ldots\}$$

When arranged in increasing order, when two adjacent numbers are in the set, then so is their sum. This definition is not rigorous because the word *adjacent* does not mean anything within a set. However, we can recursively define a sequence whose entries are the numbers in this set:

- $f_1 = 1$.
- $f_2 = 1$.
- If f_n and f_{n+1} have been defined, then $f_{n+2} = f_n + f_{n+1}$. (This is the adjacency condition.)

Let us use this definition to calculate some values.

- Since f_1 and f_2 are defined for us, $f_3 = f_1 + f_2 = 1 + 1 = 2$.
- Since f_2 and f_3 are defined for us, $f_4 = f_2 + f_3 = 1 + 2 = 3$.
- Since f_3 and f_4 are defined for us, $f_5 = f_3 + f_4 = 2 + 3 = 5$.
- Since f_4 and f_5 are defined for us, $f_6 = f_4 + f_5 = 3 + 5 = 8$.
- Since f_5 and f_6 are defined for us, $f_7 = f_5 + f_6 = 5 + 8 = 13$.

These numbers are called the Fibonacci Numbers. They arise from a problem posed in 1202 about the growth of a rabbit population with these assumptions:

1. A newly born pair of rabbits, one male, one female, are put in a field.
2. Rabbits are able to mate at the age of one month so that at the end of its second month a female can and will produce another pair of rabbits.
3. Rabbits never die.
4. Rabbits will mate any chance they get but are monomogous.
5. A mating pair always produces one new pair (one male, one female) every month from the second month on.

The number f_n is the number of pairs in the population after n months.

13.10 Exercises

Provide recursive descriptions of each of the following sets of natural numbers.

1. The set of odd natural numbers.
2. The set of multiples of 3.
3. The set of powers of 2.
4. The set of numbers which are not multiples of 4.
5. The set $\{1, 3, 7, 15, 31, 62, \ldots\}$
6. The set $\{1, 2, 4, 7, 11, 16, 22, 29, \ldots\}$

13.11 Truth Machines

We can think of a recursive definition of a set as a design for a machine which will list the elements of the set. Suppose that I have a recursively enumerable set S and that I have built the machine which lists the elements of S. If I have any number n, I can wait while the machine generates numbers. If n is in the set, then the machine will eventually generate the number n. However, it may take a long time. While I am waiting, if I have not seen n come out of the machine, then either n is in S and I just have to wait a bit longer, or maybe n is not in S. I do not know which. One of two things happens. Either I eventually see n and know that n is in S, or I wait for ever not knowing whether or not n is in S.

Now suppose that S and the set of numbers not in S are both recursively enumerable. I can build two machines – one listing those numbers in S and one listing those numbers not in S. Now consider any number n. Either n is in S or it is not. This means that one of the two machines will eventually list the number n. All I have to do is wait. There is no danger that both machines will run forever without generating n. Thus, I can build a machine (the two together) which will tell me whether or not a number n is in S. In this case, we say that S is *computable* or *recursive*.

Recall that Leibniz dreamed of building a truth machine into which he could feed statements and which would output (correctly) whether the statement was true or false. Leibniz was asking whether or not the set of true statements (in some language) is computable (or recursive). We could just as well ask whether or not a set of true statements is recursively enumerable.

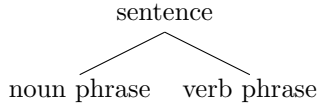
Chapter 14

Phrase Structure Grammars

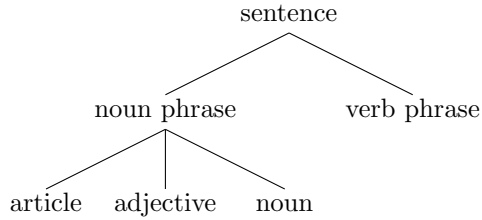
In the previous chapter we explored recursive definitions of wffs in a sentential language. In this chapter we investigate a manner of specifying grammatically correct sentences in a language in a way that more closely mimics the process we use in our spoken language.

14.1 Building a Sentence in English

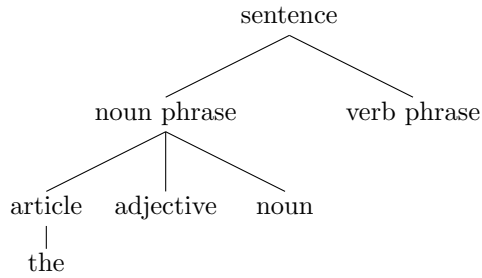
We begin with a question: What is a sentence? Well, there are many answers to this, but a first approximation is that a sentence is a noun phrase followed by a verb phrase. Pictures will be helpful in this chapter. A picture indicating that a sentence is a noun phrase followed by a verb phrase might look like:



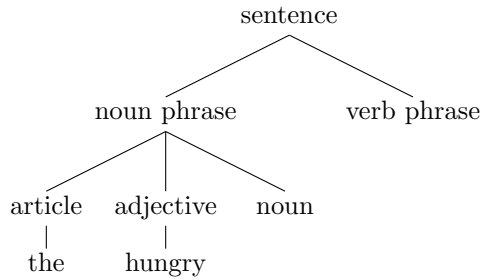
This is called a *tree*. This is all well and good, but what is a noun phrase? A noun phrase can be many things. One example is that a noun phrase can be an article followed by an adjective followed by a noun:



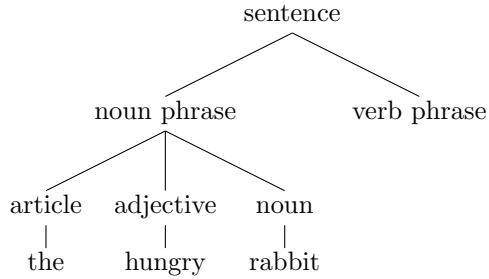
An article is a word like a, an, or the. We will replace the word article with one of these.



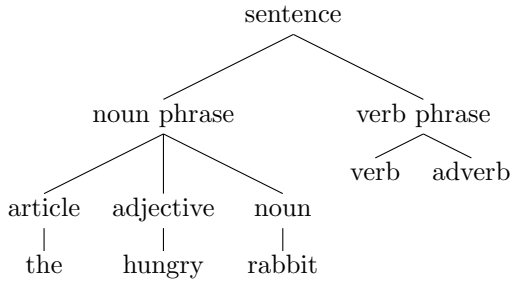
There are many adjectives. Some are tired, hungry, happy, sloppy. We can replace the word article with any of these.



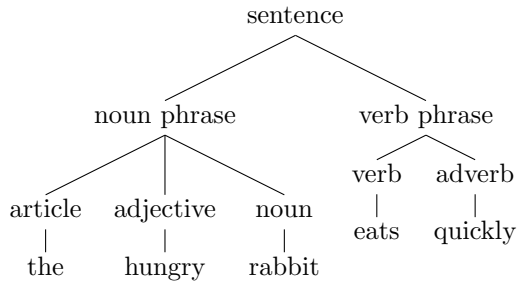
There are many nouns too. Some are rabbit, student, professor. Any of them can be used here.



We can now play the same game with the verb phrase. A verb phrase might be a verb followed by an adverb.



Here are some verbs: eats, jumps, sleeps. Here are some adverbs: quickly, lazily, violently. We choose one of each:



Any way we pick an article, an adjective, a noun, a verb, and an adverb from our lists will be guaranteed to make a grammatically correct sentence. Some of these will have meaning like the one above. Others will be nonsense such as.

A hungry student sleeps violently.

This tree which we drew to indicate how our sentence is built is called a *derivation tree*.

14.2 Phrase Structure Grammars

Let us formalize the process we followed in the last section to write a sentence. First, we have a set of symbols which we will call our *vocabulary*. For this example, the symbols involved were:

> sentence, noun-phrase, verb-phrase, article, adjective, noun, ver
> adverb, a, an, the, tired, hungry, happy, sloppy, rabbit, student,
> professor, eats, jumps, sleeps, quickly, lazily, violently

Notice how we connected noun-phrase and verb-phrase to indicate these pairs of words represent single symbols. A *string* or *expression* is just a list of symbols from our vocabulary. The symbol sentence has a special place in our discussion. It is our *start symbol*. We began in the last section with our start symbol – sentence. We then replaced this with a string of symbols – noun-phrase verb-phrase. We then replace each of these and then did some more replacing. Our replacing has to follow certain rules. This is what guarantees the grammatical correctness of our sentences. We call these rules *replacement rules* or *production rules*. The replacement rules we considered in the example can be written this way:

- sentence→noun-phrase verb-phrase
- noun-phrase→article adjective noun
- verb-phrase→verb adverb
- article→a, article→an, article→the
- adjective→tired, adjective→hungry, adjective→happy, adjective→sloppy

- noun→rabbit, noun→student, noun→professor
- verb→eats, verb→jumps, verb→sleeps
- adverb→quickly, adverb→lazily, adverb→violently

While constructing our sentence in the previous section, we replaced some symbols following these production rules, but we did not have to replace every symbol. For example, we did not replace the symbol eats. Those symbols in our vocabulary that do not need to be replaced are called *terminal symbols*. We have now identified all of the ingredients of a phrase structure grammar.

A *phrase structure grammar* consists of

- A set of symbols called the *vocabulary*.
- A designated symbol in the vocabulary called the *start symbol*.
- A set of symbols in the vocabulary called the *terminal symbols*.
- A set of *production or replacement rules*.

The expressions which are *derivable* from a phrase structure grammar are recursively defined by

- The start symbol is derivable.
- If α is derivable and if β is constructed by applying a production rule to α, then β is derivable.
- These are the only ways in which an expression is derivable.

The *language generated by a phrase structure grammar* is the set of all derivable expressions which contain only terminal symbols.

14.3 Some Examples

Example 14.1: Make up production rules for the phrase structure grammar which will generate all arithmetical expressions involving only the variables x and y, the operation $+$, and parenthesis. Then draw a derivation tree of the expression

$$((x + y) + (x + y))$$

We will use the word "expression" as a start symbol. Since the sum of two expressions should also be an expression, we must include the production rule

expression→(expression+expression)

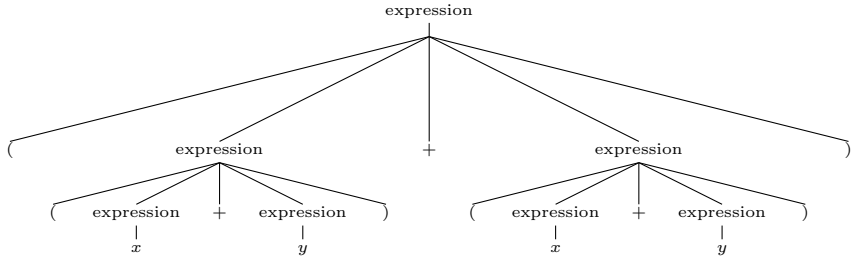
Notice how we build parenthesis into the production rule to avoid confusion about associativity. Next, inside of a sum, an expression might simply be a single variable, so we include

expression→ x and expression→ x

Our rules are

- expression→(expression+expression)
- expression→ x
- expression→ y

Here is a derivation tree for $((x + y) + (x + y))$



Example 14.2: Add production rules to the example in section 14.1 that will allow for compound sentences.

As in the last example, we need to allow two sentences to be joined together. In the arithmetic expression example, the sentences are joined with a +. In English, we need a conjunction. One rule will allow two sentences to be joined with a conjunction. The other rules will say what a conjunction is.

- sentence→sentence conjunction sentence
- conjunction→and, conjunction→but, conjunction→or

Example 14.3: Make up production rules for the language which consists of all strings of the form AB, $AABB$, $AAABBB$ where there are the same number of As and Bs.

We will approach this two ways. Each uses a different trick. The first approach is to replace the start symbol with AB and then to replace AB with $AABB$ to increase the number of As and Bs. That is right – we can replace a string of symbols.

- start→ AB
- $AB \rightarrow AABB$

The second approach uses an *empty string*. We first replace the start symbol with $A\text{start}B$ and repeat this until we have the desired number of As and Bs. Then we replace the start symbol with an empty string – thereby removing it. In our notation, to indicate the empty string, we will use λ (so, when you see λ there really is nothing there).

- start$\rightarrow A\text{start}B$
- start$\rightarrow \lambda$

14.4 Exercises

1. Make up two more sentences in the language generated by the grammar in section 14.1 and draw derivation trees for each.
2. Make up two sentences in the language generated by the grammar in Example 14.2 and draw derivation trees for each.
3. List three expressions in the language generated by the phrase structure grammar with vocabulary $\{start, A, B\}$ and production rules

   - start$\rightarrow AB$
   - $AB \rightarrow ABAB$
   - $AB \rightarrow B$

   The letters A and B are both terminal.
4. List three expressions in the language generated by the phrase structure grammar with vocabulary $\{start, A, B\}$ and production rules

   - start$\rightarrow ABA$
   - $A \rightarrow ABA$
   - $BAB \rightarrow ABA$

   The letters A and B are both terminal.
5. Make a phrase structure grammar that will generate all expressions of the form $AABBBB$, $ABBB$, $AAAABB$ with any positive number of As and Bs.
6. Make a phrase structure grammar for first order sentential logic.

Chapter 15

Predicate Logic

The sentential logic we have developed is powerful – it gives a reasonable picture of a large segment of natural reasoning and a model of many of the types of reasoning involved in doing mathematics. It has a natural deductive calculus which is complete in the sense that all logical implications can be proven from Modus Ponens and sound in the sense that proofs only establish sentences which are logical consequences of the assumptions.

However, sentential logic has a glaring weakness in that the notion of "sentence" is a bit too vague. *Real* reasoning requires us to say a bit about what makes up a sentence, what a sentence can "say," and how sentences can be related beyond constructions with logical connectives.

This argument

Bob is a man. All men are mortal. Therefore, Bob is mortal.

Makes perfect sense, but the best sentential logic could do with this is to identify three seemingly unrelated atomic sentences. To sentential logic, this argument looks like $A, B \therefore C$ – which is *invalid*. The problem is that the idea of basic sentence symbols is a bit too coarse for many forms of actual reasoning. Sentences (such as the ones here) can be related by what they say as well as how they are put together with connectives. In this chapter, we introduce the ideas of predicates and quantifiers to address this problem.

15.1 Predicates

We begin with the first sentence in the argument above, "Bob is a man." The sentences "Larry is a man," "Lola is a man," and "Glenda is a man" are all related to this sentence. They are all of the form "x is a man." Each has a different name (or person) substituted for the letter x. The sentence "x is a man" is an example of a predicate. A *predicate* is a sentence involving variables which takes on a truth value once specific objects are substituted for the variables.

We will use capital letters such as A, B, $C, \ldots$ to represent predicates. If a predicate has a variable x, and if we want to name the predicate P, we will usually refer to the predicate as $P(x)$ (read "P of x"). The same predicate with "Bob" substituted for x would be $P(\text{Bob})$. For example, if $P(x)$ is "x is a man" then $P(\text{Bob})$ would be "Bob is a man." $P(\text{Glenda})$ is "Glenda is a man."

Predicates can have more than one variable. If $Q(x, y)$ is "x is married to y," then $Q(\text{Bob}, \text{Glenda})$ is "Bob is married to Glenda." $Q(1, 2)$ is "1 is married to 2" (which makes no sense). Suppose that $B(x, y, z)$ is "y is between x and z." Then $B(1, 2, 3)$ is "2 is between 1 and 3." $B(\text{Bob}, \text{Frank}, \text{Hank})$ is "Frank is between Bob and Hank."

The number of variables in a quantifier will be called the *rank* of the predicate. Here, P has rank 1, Q has rank 2, and B has rank 3. We can also say that Q is a 2-place predicate or that B is a 3-place predicate.

15.2 Quantifiers

Now that we are aware of predicates, we move on to the second sentence in the argument from above: "All men are mortal." There is clearly the predicate "x is mortal" at play here. The difference between this and the first sentence is that here we try to substitute all men for x at the same time. We can rewrite this sentence in this way to account for all men:

> For all x, if x is a man, then x is mortal.

Here there are two predicates which have been combined: "x is a man" and "x is mortal." We also have an implication in the form of "if...then..." What is new is the "For all x." This is a quantifier. We will have two quantifiers, one to mean "For all" and one to mean "For some."

The *universal quantifier* is the symbol $\forall$. The expression $\forall x$ can be read as "For all x." If $P(x)$ is any predicate, then $\forall x P(x)$ can be read "For all x, $P(x)$."

The *existential quantifier* is the symbol $\exists$. The expression $\exists x$ can be read as "For some x." If $P(x)$ is any predicate, then $\exists x P(x)$ can be read "For some x, $P(x)$." (Note: Here "some" means "at least one.")

We make the following assumption:

Definition of the Existential Quantifier The existential quantifier $\exists x P(x)$ is defined to mean $\neg \forall x \neg P(x)$.

This should seem reasonable. For example, "Some men are mortal" should mean (about) the same thing as "It is not the case that all men are not mortal."

Let us return now to "All men are mortal." We have translated this into

For all x, if x is a man, then x is mortal.

Let $P(x)$ be "x is a man," and let $Q(x)$ be "x is mortal." We can express this symbolically as $\forall x(P(x) \to Q(x))$. The argument from the introduction to the chapter can now be expressed symbolically as

$$P(\text{Bob}), \forall x(P(x) \to Q(x)) \therefore Q(\text{Bob}).$$

As with our logical connectives, there are a variety of ways to translate quantifiers into English. Some translations of $\forall x P(x)$ are

For all x, $P(x)$.
For any x, $P(x)$.
$P(x)$, for all x.

Some translations of $\exists x P(x)$ are

For some x, $P(x)$.
For at least one x, $P(x)$.
There exists an x so that $P(x)$.
There is an x so that $P(x)$.
There is at least one x so that $P(x)$.
$P(x)$ for some x.
$P(x)$ for at least one x.

The task before us is to mimic the work we did with sentential logic now with predicate logic: We need to address:

- Which expressions are wffs for predicate languages?
- Truth values.
- Logical Equivalences.
- Logical implication and deduction.
- Completeness and Compactness.

Because of the work we have already done, this will not be as difficult as it may sound. We will begin with some translations to get used to our new symbols.

15.3 Exercises

Define these predicates:

- $C(x)$ is "x is a cat."
- $D(x)$ is "x is a dog."
- $B(x, y)$ is "x is bigger than y."

- $H(x, y)$ is "x chases y."
- $P(x)$ is "x is pink."
- $Y(x)$ is "x is yellow."
- $F(x)$ is "x has fleas."

Translate the following English sentences into symbols.

1. There is a yellow cat.
2. Every cat is either yellow or pink.
3. If it is pink, it is not a cat.
4. Every dog is bigger than every cat.
5. Every dog is bigger than some cat.
6. Some dog is bigger than some cat.
7. Some dog is bigger than any cat.
8. If it is a cat, then it is a dog.
9. Every yellow cat is not bigger than any pink dog.
10. If it is pink, then it is either a dog or a cat.
11. If it is yellow and it is bigger than some dog, then it is a cat.
12. All dogs have fleas.
13. Some dogs have fleas.
14. Not every dog has fleas.
15. Some dogs do not have fleas.
16. No dog has fleas.
17. Every dog chases some cat.
18. Every dog chases every cat.
19. Some dog chases every cat.
20. Some dog does not chase any cat.
21. No dog chases every cat.

15.4 Predicate Languages

In Chapter 2, we gave a recursive definition of what composed a grammatically correct sentence or well-formed formula in sentential logic. We mimic that process here with predicate logic.

The languages we build here are designed to exploit predicates and allow quantification over *elements*. This is *first order predicate logic*. If we were to allow quantification over *sets*, then we would be working in *second order logic*. We will often call our system "predicate logic." The "first order" is supposed to be understood.

To build a first order predicate language we need these logical symbols:

parenthesis	) and (
logical connectives	$\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$
quantifiers	$\forall$ and $\exists$
variables	v_1, v_2, $v_3 \ldots$

The new additions here are the quantifiers and the variables (which will be placed inside of predicates).

Suppose that L is a set of predicate symbols (each symbol in L represents a predicate and has a rank associated to it). Any list of logical symbols and symbols from L is an *expression*. We define the *well-formed formulas over* L recursively as follows

1. If v_i and v_j are variables, then $v_i = v_j$ is a wff.
2. If P is an n-place predicate symbol from L, and if $x_1, x_2, \ldots, x_n$ are variables, then $P(x_1, x_2, \ldots, x_n)$ is a wff.
3. If α and β are wffs, then so are
$$(\neg\alpha), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \to \beta), \text{ and } (\alpha \leftrightarrow \beta).$$
4. If v is a variable an α is a wff, then $\forall v\alpha$ is a wff.
5. If v is a variable an α is a wff, then $\exists v\alpha$ is a wff.
6. No expression can be a wff except for these reasons.

The wffs in (1) and (2) are *atomic formulas*. The intention in (2) is that $x_1, x_2, \ldots, x_n$ are among the v's. In (4) $\forall v\alpha$ is called a *generalization* of α.

In most practical exercises, this formal recursive definition of wffs can practically be ignored. As with sentential logic, the recursive definition of wffs gives us a framework for proof by induction. It also gives us an outline for making definitions in relation to wffs. The recursive definition also sets us up for a discussion of computability questions later.

Because of our assumption that $\exists x P(x)$ means the same thing as $\neg\forall x\neg P(x)$, and because of the logical equivalences for sentential logic, we note that we could have defined wffs only with $\forall$ and $\neg$ and $\wedge$ (or $\neg$ and $\vee$ or $\neg$ and $\to$).

15.5 Free Variables

Any wff contains variables. Some of these variables may be affected by quantifiers while others may not. For example, in $\forall x(P(x) \wedge Q(y))$, x is affected by the $\forall$, but y is not. We say that x is bound and y is free. We also say that x is in the scope of the $\forall$.

Formally: Suppose that α is a wff. Any x in $\forall x\alpha$ is *in the scope* of the $\forall x$. A variable which is in the scope of a quantifier is *bound*. Any variable which is not bound is *free*.

A wff with free variables is akin to a predicate with variables. Neither can have a truth value until some object is substituted for the variables. For this reason, when we start speaking of truth in predicate logic, we must speak

only about wffs with no free variables. We will call a wff with no free variables a *sentence*. A formula with free variables will be called an *open formula* or open statement (or even predicate sometimes).

15.6 Substitution in formulas

If we view wffs with free variables as predicates, then we will be wanting to substitute objects for the free variables in a wff. Suppose that α is a wff. We will write $\alpha(v_1, v_2, \ldots v_n)$ to indicate that the free variables in α are among $v_1, v_2, \ldots, v_n$. We will use $\alpha(x_1, x_2, \ldots, x_n)$ for the expression obtained by replacing in α every free occurrence of v_1 by an object x_1, and v_2 by x_2, and so on. For example, suppose α is

$$\exists v_1 \forall v_2 (P(v_1, v_3) \to Q(v_2, v_4)).$$

In α, v_1 and v_2 are bound but v_3 and v_4 are free. We could refer to α as $\alpha(v_1, v_2, v_3, v_4)$ (even though v_1 and v_2 cannot be "substituted for") or as $\alpha(v_3, v_4)$ (which is really an abuse of notation). We generally would not use both expressions in the same context. The expressions $\alpha(a, b, c, d)$ and $\alpha(c, d)$ would both would mean

$$\exists v_1 \forall v_2 (P(v_1, c) \to Q(v_2, d)).$$

Sometimes, we will really abuse notation and use $\alpha(v_i)$ for a formula with a free variable v_i even if there are other free variables. In these cases, we will assume that all substitution is for free occurrences of v_i.

15.7 Logical Axioms

In this section, we lay out basic assumptions that we make in order to give the desired meanings to our logical symbols. These assumptions are *logical axioms*. We call this set of axioms (for a fixed predicate language) Δ.

The set Δ of logical axioms is the set of all *generalizations* of these types of formulas.

Tautologies:

- If α is any tautology of sentential logic, then any formula obtained by replacing the sentence symbols in α by well-formed predicate formulas is an axiom of predicate logic.

For example, $(P \to Q) \leftrightarrow (\neg P \vee Q)$ is a tautology of sentential logic (Disjunctive Implication). If we replace the Ps in this formula with $\forall x R(x)$ and the Qs with $\exists y \forall z (R(y, z) \vee S(y))$ then we get

$$(\forall x R(x) \to \exists y \forall z (R(y, z) \vee S(y)) \leftrightarrow (\neg \forall x R(x) \vee \exists y \forall z (R(y, z) \vee S(y))).$$

This would be an axiom of predicate logic. These axioms are assumed to allow us to use the logical equivalences of sentential logic.

Quantifier Axioms:

- For any wff α and variable x, $\exists x\alpha \leftrightarrow \neg\forall x\neg\alpha$ is an axiom.
  (This axiom merely defines the existential quantifier in terms of the universal quantifier.)
- Suppose that α is a wff and that x is any symbol. Then $\forall v_i\alpha(v_i) \rightarrow \alpha(x)$ is an axiom.
  (This axioms says that if α is true everywhere, then α is true in one specific instance.)
- If α and β are wffs and v is a variable then $\forall v(\alpha \rightarrow \beta) \rightarrow (\forall v\alpha \rightarrow \forall v\beta)$ is an axiom.
  (This axioms says that if α always implies β, and if α is always true, then β is always true.)
- If a variable v does not occur free in α, then $\alpha \rightarrow \forall v\alpha$ is an axiom.
  (Basically, if v has no relationship to α, and if α is true, it follows that $\forall v\alpha$ is true.)

The first axiom defines the existential quantifier. It gives $\neg\forall v\alpha \leftrightarrow \exists v\neg\alpha$ and $\neg\exists v\alpha \leftrightarrow \forall v\neg\alpha$. That is, "not all" is "some not" and "not some" is "all not." The second axiom will be the basis for the rule of inference Universal Instantiation in Chapter 16. The third and fourth axioms will be used in establishing the rule of inference Universal Generalization in Chapter 16.

Equality Axioms:

- $x = x$
- $(x = y) \rightarrow (\alpha \leftrightarrow \alpha')$ where α' is obtained from α by replacing some occurrences of x by y.

15.8 Exercises

These exercises emphasize the relationships

$$\neg\forall v\alpha \leftrightarrow \exists v\neg\alpha \text{ and } \neg\exists v\alpha \leftrightarrow \forall v\neg\alpha.$$

That is, "not all" is "some not" and "not some" is "all not." They will also emphasize that truth for predicate logic is handled a little differently than truth for sentential logic. Rather than truth assignments, we must consider small universes or models. Here are eight sets of "holes." Some are filled (the dark ones). Some are not filled.

A. ○ ○ ○

B. ○ ○ ●

C. ○ ● ○

D. ○ ● ●

E. ● ○ ○

F. ● ○ ●

G. ● ● ○

H. ● ● ●

Let $F(x)$ be "x is filled." Translate the following statements into symbols and then decide which sets of holes satisfy the statement.

1. All holes are filled.
2. Some holes are filled.
3. A hole is filled.
4. There is a hole which is filled.
5. There is a hole which is not filled.
6. All holes are not filled.
7. Some holes are not filled.
8. A hole is not filled.
9. It is not the case that all holes are filled.
10. It is not the case that some holes are filled.
11. It is not the case that all holes are not filled.
12. It is not the case that some holes are not filled.
13. Not all holes are filled.
14. Not all holes are not filled.

Chapter 16

Implication and Deduction

We studied two ways in which a set Σ of of wffs in sentential logic could imply a sentence α. First we considered logical implication or entailment $\Sigma \models \alpha$. This type of implication means that every truth assignment which make every sentence in Σ true also makes α true. In symbols, for any truth assignment s, if $s \models \Sigma$, then $s \models \alpha$. We then considered deductive implication $\Sigma \vdash \alpha$. This type of implication means that a proof of α can be constructed from (finitely many) sentences in Σ. We learned in the Completeness and Soundness Theorems that these two notions of implication coincide.

We now proceed with a similar line of reasoning in predicate logic. We will follow the example of sentential logic with the adjustments that we need a replacement for truth assignments and that our rules of inference for proof must account for quantifiers.

This process with Predicate logic will give three possible types of implication. First, we define a deductive calculus which allows us to prove some wffs from others. Then we consider what it means for a collection Σ of predicate wffs to imply a predicate wff α when all of the wffs are treated as objects in sentential logic (in either of the two equivalent manners $\models$ or $\vdash$). Finally, in Chapter 19 we replace the notion of truth assignment from sentential logic with the idea of a model. If every model of a collection Σ of wffs satisfies α, then we say that Σ logically implies α.

Our deductive mechanisms will be such that all three of these notions of implication will coincide.

16.1 Deduction and Proof

We first define a deduction of α from Σ in a manner almost identical to sentential logic.

Proof Suppose that Σ is a set of wffs and that α is a wff. A *proof* or *deduction* of α from Σ is a list $\gamma_1, \gamma_2, \ldots, \gamma_n$ of wffs so that $\gamma_n = \alpha$ and each wff γ_i in the list

- is either an axiom or
- is in Σ or
- follows from two earlier wffs by Modus Ponens.

If there is a proof of α from Σ, then we say that α is *provable* or *deducible* from Σ. In symbols, we express this as $\Sigma \vdash \alpha$.

This is the definition from sentential logic with one change. Here, some of the γ_is may be *axioms*, where in sentential logic, they were *tautologies*. Recall that predicate versions of the sentential tautologies are included in our axioms. Thus the mechanisms for deduction in predicate logic should be just as strong as those in sentential logic. The difference here is that we have variables and quantifiers within our wffs.

We will mimic the notation from sentential logic for our proofs in predicate logic. Notice in this example how the logical axioms allow us to manipulate the quantifiers.

Example 16.1: Write a proof to show that

$$\{\forall x(P(x) \to Q(x)), P(y)\} \vdash Q(y)$$

1.	$\forall x(P(x) \to Q(x))$	premise
2.	$P(y)$	premise
3.	$[\forall x(P(x) \to Q(x))] \to [P(y) \to Q(y)]$	axiom
4.	$P(y) \to Q(y)$	Modus Ponens 1, 3
5.	$Q(y)$	Modus Ponens 4,2

Proofs will be quite tedious if all we have to work with are Modus Ponens and axioms. We will shortly derive other rules of inference and proof techniques which will make it easier to write proofs.

16.2 Sentential Implication

Suppose that L is a predicate language. A *prime formula* over L is any atomic formula or a formula of the form $\forall x\alpha$ or $\exists x\alpha$. That is, a formula is prime if it is either atomic, or if the entire formula is in the scope of a single quantifier.

Example 16.2: These formulas are prime

$$v_1 = v_2,\ \forall x(P(x) \wedge \exists y(Q(x,y) \rightarrow R(x))),\ \forall x(P(x) \vee Q(x,y)).$$

These formulas are not prime

$$(v_1 = v_2) \vee (v_1 = v_3),\ \forall x P(x) \wedge \exists y Q(y).$$

Any formula over L can be constructed by combining prime formulas with our logical connectives. This observation leads to another possible interpretation of logical implication. Let L' be the sentential language whose sentence symbols are the prime formulas from L. Then every first order formula over L is a sentential formula over L'. If Σ is any collection of wffs over L and if α is a wff in L, then we will say that Σ *sententially implies* α if Σ logically implies α in sentential logic over L'. We will express this in symbols as $\Sigma \models_S \alpha$.

Recall that Δ is the collection of all the logical axioms we declared for predicate logic in Chapter 9. We can use induction along with the Completeness and Soundness Theorems from sentential logic to prove:

Theorem 16.3: Sentential Implication Theorem: Suppose that Σ is a collection of wffs in a predicate language and that α is a wff in the same language. Then $\Sigma \vdash \alpha$ if and only if $\Sigma, \Delta \models_S \alpha$.

Proof: Let S be the set of all sentences α for which $\Sigma \vdash \alpha$. Suppose that s is a truth assignment that makes every sentence in $\Sigma \cup \Delta$ true. We use an induction argument to show that $s(\alpha) = T$ for every α in S. Suppose that $\alpha \in S$. If $\alpha \in \Sigma$ or $\alpha \in \Delta$, then $s(\alpha) = T$ by assumption. Suppose then that $\gamma \rightarrow \alpha$ and γ are in S and that $s(\gamma \rightarrow \alpha) = s(\gamma) = T$. Since $\{\gamma \rightarrow \alpha, \gamma\} \models_S \alpha$, it follows that $s(\alpha) = T$. By induction $s(\alpha) = T$ for all $\alpha \in S$. Consequently, if $\Sigma \vdash \alpha$ (that is, if $\alpha \in S$), then $\Sigma, \Delta \models_S \alpha$.

Now suppose that $\Sigma, \Delta \models_S \alpha$. By the compactness theorem for sentential logic, there is a finite subset $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ of $\Sigma \cup \Delta$ so that $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} \models_S \alpha$. It follows that

$$\gamma_1 \rightarrow (\gamma_2 \rightarrow (\cdots \rightarrow (\gamma_n \rightarrow \alpha) \cdots))$$

is a tautology and, hence, an axiom and is in Δ (see exercise 9 in 5.4). Then repeated applications of Modus ponens to $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ along with this axiom will give α. Thus, if $\Sigma, \Delta \models_S \alpha$ then $\Sigma \vdash \alpha$. □

16.3 Derived rules of inference

As with sentential logic, proofs in predicate logic with just modus ponens get somewhat laborious. Since we have at our disposal the tautologies from sentential logic within our axioms we could prove all of the derived rules of

inference from Chapter 8 for predicate logic. In fact, their proofs would be identical (with the word "tautology" replaced by "axiom." We will, then, assume we have these derived rules at our disposal. Of course, these derived rules do not even acknowledge the quantifiers we are using. We will also derive rules that allow us to work with the quantifiers in a way that is compatible with their meaning. The easiest way of doing so uses proofs with temporary assumptions. We can derive the proof techniques from sentential logic which use temporary assumptions if we have an extension of the Deduction Theorem to predicate logic. Here is such an extension.

Theorem 16.4: Deduction Theorem for Predicate Logic: Suppose that Σ is a set of wffs in a predicate language and that α and β are wffs in the same language. Then $\Sigma, \alpha \vdash \beta$ if and only if $\Sigma \vdash \alpha \rightarrow \beta$.

Proof: Our proof uses the Sentential Implication Theorem of the previous section along with the Deduction Theorem for Sentential Logic.

$\Sigma, \alpha \vdash \beta$		
if and only if	$\Sigma, \Delta, \alpha \models_S \beta$	Sentential Implication Theorem
if and only if	$\Sigma, \Delta \models_S \alpha \rightarrow \beta$	Deduction Theorem, Sentential Logic
if and only if	$\Sigma \vdash \alpha \rightarrow \beta$	Sentential Implication Theorem

□

Since we have the Deduction Theorem, we could now derive all of the proof techniques of sentential logic which employ temporary assumptions. Their derivations would be virtually identical for predicate logic.

We now have at our disposal in predicate logic all derived rules of inference and all temporary assumption techniques which we used in sentential logic. We use these now to derive some techniques involving quantifiers. Our first new rule of inference is based on the fourth quantifier axiom of predicate logic (you should look at that). This inference basically says that if we want to prove $\forall x \alpha(x)$, we simply need to prove $\alpha(a)$ for some arbitrary symbol a.

Universal Generalization: If $\Sigma \vdash \alpha(x)$ and x does not occur free in Σ, then $\Sigma \vdash \forall x \alpha(x)$.

To prove $\forall x \alpha(x)$, it suffices to prove $\alpha(a)$ for some unused (arbitrary) symbol a.

We include a proof here by induction only to illustrate the use of the axioms of predicate logic.

Proof: Suppose that x does not occur in Σ. We wish to prove that Σ proves $\forall x \alpha$ for any α provable from Σ. To do so, we use induction based on

the recursive definition of formulas provable from Σ. Let S be the set of all α provable from Σ for which $\Sigma \vdash \forall x\alpha$. We prove this is all formulas provable from Σ.

If α is a logical axiom, then $\forall x\alpha$ is also a logical axiom, so $\Sigma \vdash \alpha$.

If α is in Σ, then x does not occur in α. Then $\alpha \to \forall x\alpha$ is an axiom. It follows then that $\Sigma \vdash \forall x\alpha$ by Modus Ponens.

Suppose now that γ and $\gamma \to \alpha$ are in S. Then $\Sigma \vdash \forall x\gamma$ and $\Sigma \vdash \forall x(\gamma \to \alpha)$. Now, $\forall x(\gamma \to \alpha) \to (\forall x\gamma \to \forall x\alpha)$ is an axiom. By Modus Ponens, we get $\Sigma \vdash (\forall x\gamma \to \forall x\alpha)$. Another application of Modus Ponens now gives $\Sigma \vdash \forall x\alpha$.

We now have by induction that S is the set of all formulas provable from Σ. □

Our next rule of inference says that if we know that $\alpha(v)$ is true for every possible value of v, then it is true for a specific value of v. This is based on the second quantifier axiom.

Universal Instantiation: If x is a symbol and v is a variable, from $\forall v\alpha(v)$ infer $\alpha(x)$.

1.	$\forall v\alpha(v)$	premise
2.	$\forall v\alpha(v) \to \alpha(x)$	axiom
3.	$\alpha(x)$	Modus Ponens 2,1

Our next rule of inference allows us to prove an existential. To prove $\exists v\alpha(v)$, we simply need to establish $\alpha(x)$ for (almost) any x. The proof exploits the definition of $\exists$ in terms of $\forall$.

Existential Generalization: If v is a variable which does not occur free in $\alpha(x)$, then from $\alpha(x)$ infer $\exists v\alpha(v)$.

To prove $\exists v\alpha(v)$, prove $\alpha(x)$ for some symbol x.

1.	$\alpha(x)$	premise
2.	$\neg\exists v\alpha(v)$	Temp. Assumption BWOC
3.	$\forall v\neg\alpha(v)$	Equivalent 2
4.	$\neg\alpha(x)$	Universal Instantiation
5.	$\alpha(x) \wedge \neg\alpha(x)$	Conjunction 1,4
6.	$\exists v\alpha(v)$	Contradiction 2-5

Our last new rule of inference uses the definition of $\exists$ along with universal generalization. This rule simply allows us to give names to objects which we know to exist. If $\exists v\alpha(v)$ is true, we can name the object that makes α true. The restriction is that we should use a symbol that has not been used before.

Existential Instantiation: If x does not occur in $\alpha(v)$, β, or in Σ, and if $\Sigma, \alpha(x) \vdash \beta$, then $\Sigma, \exists v \alpha(v) \vdash \beta$. To prove that $\Sigma, \exists v \alpha(v) \vdash \beta$, suppose $\alpha(x)$ and prove β.

Proof: By the deduction theorem, $\Sigma \vdash \alpha(x) \to \beta$. Using the contrapositive, $\Sigma \vdash \neg\beta \to \neg\alpha(x)$. The deduction theorem now gives $\Sigma, \neg\beta \vdash \neg\alpha(x)$. By universal generalization, $\Sigma, \neg\beta \vdash \forall v \neg\alpha(v)$. Now, $\forall v \neg\alpha(v)$ is equivalent to $\neg\exists v \alpha(v)$, so $\Sigma, \neg\beta \vdash \neg\exists v \alpha(v)$. The deduction theorem and the contrapositive now give $\Sigma, \exists v \alpha(v) \vdash \beta$. □

16.4 Examples

Here are several examples of how to combine our derived rules of inference, our rules with temporary assumptions, and our rules involving quantifiers to write proofs in predicate logic.

Example 16.5: Write a proof to show that

$$\{\forall x(P(x) \to Q(x)), \exists x P(x)\} \vdash \exists x Q(x)$$

1.	$\forall x(P(x) \to Q(x))$	premise
2.	$\exists x P(x)$	premise
3.	$P(y)$	Existential Instantiation 2
4.	$P(y) \to Q(y)$	Universal Instantiation 1
5.	$Q(y)$	Modus Ponens 4,3
6.	$\exists x Q(x)$	Existential Generalization 5

Note the strategy in this example. First we applied instantiation rules to manufacture statements without quantifiers. We first do existential instantiation to "name" elements that the premises declare must exist. Then we use sentential rules of inference (derived rules if necessary). Then we generalize with one of the generalization rules.

Example 16.6: Write a proof to show that

$$\{\forall x(P(x) \vee Q(x)), \forall x(R(x) \wedge \neg Q(x))\} \vdash \forall x P(x)$$

1.	$\forall x(P(x) \vee Q(x))$	premise
2.	$\forall x(R(x) \wedge \neg Q(x))$	premise
3.	$P(a) \vee Q(a)$	Universal Instantiation
4.	$R(a) \wedge \neg Q(a)$	Universal Instantiation
5.	$\neg Q(a)$	Simplification 4
6.	$P(a)$	Disjunctive Syllogism 3, 5
7.	$\forall x P(x)$	Universal Generalization 6

The a in lines 3 and 4 represents some $a$rbitrary element. In a paragraph proof, these lines would be preceeded by a comment like "Let a be arbitrary." This is to set up universal generalization in line 7. Universal generalization can only be applied with a generic or arbitrary element such as this.

Example 16.7: Write a proof to show that

$$\{\forall x(T(x) \to P(x)), \exists x[\neg(Q(x) \wedge S(x)) \to (T(x) \wedge \forall z U(z))]\}$$

$$\vdash \exists x(P(x) \vee Q(x))$$

1.	$\forall x(T(x) \to P(x))$	premise
2.	$\exists x[\neg(Q(x) \wedge S(x)) \to (T(x) \wedge \forall z U(z))]$	premise
4.	$\neg(Q(b) \wedge S(b)) \to (T(b) \wedge \forall z U(z))$	Existential Inst. 2
5.	$T(b) \to P(b)$	Universal Inst. 1
6.	$\neg P(b)$	Temp. Assumption
7.	$\neg T(b)$	Modus Tollens 5,6
8.	$\neg T(b) \vee \neg \forall z U(z)$	Addition 7
9.	$\neg(T(b) \wedge \forall z U(z))$	Equivalent 8
10.	$Q(b) \wedge S(b)$	Modus Tollens 4,9
11.	$Q(b)$	Simplification 10
12.	$P(b) \vee Q(b)$	Disjunctive Proof 6-11
13	$\exists x(P(x) \vee Q(x))$	Existential Gen. 12

16.5 Exercises

Write a proof for the stated deduction.

1. $\{\forall x[(P(x) \wedge Q(x)) \to S(x)], \exists x[P(x) \wedge Q(x)]\} \vdash \exists x[S(x) \vee T(x)]$
2. $\{\exists x P(x), \exists x Q(x)\} \vdash \exists u \exists v[P(u) \wedge Q(v)]$
3. $\{\forall x[P(x) \vee Q(x)], \forall x[Q(x) \to \neg S(x)], \forall x S(x)\} \vdash \forall x P(x)$
4. $\{\exists x[P(x) \wedge \neg P(x)]\} \vdash \forall y Q(y)$
5. $\{\forall x[P(x) \vee Q(x)], \forall x[P(x) \vee R(x)]\} \vdash \forall x[(\neg R(x) \vee \neg Q(x)) \to P(x)]$
6. Make up an example of a sentential deduction whose proof requires direct proof. Provide the proof.
7. Make up an example of a sentential deduction whose proof requires disjunctive syllogism and modus tollens. Provide the proof.
8. Make up an example of a sentential deduction whose proof requires disjunctive proof, Modus Ponens, and addition. Provide the proof.

9. Make up an example of a sentential deduction whose proof requires cases and simplification. Provide the proof.
10. Convert each of your examples to examples involving quantifiers.

Chapter 17

Sets

In the last chapter, we defined the notation and made assumptions to lay the base of predicate logic. We would like now to follow the steps we took with sentential logic and address truth, logical implication, and deductions. We immediately hit a wall when we try to address truth. If we consider a sentence such as

For all x, x weighs less than a pound.

and ask if the sentence is true or false, then our answer depends on what the variable x represents. If x can only be selected from the collection of all living canaries, then the statement is true. If x can be any living human, then the statement is false. When considering the sentences in a predicate language, we will usually have a "universe" in mind that provides objects to which the quantifiers refer. This universe will be accompanied by interpretations of the predicate symbols of the language. This universe and the meanings of the predicates form what we will call a model of the predicate language. Before we can address models (and then truth, and then implication) we need to know some basics about set theory.

17.1 Sets

In any spoken language, at any point in time, there are only finitely many words. This has surprising consequences when you try to define words. Suppose we try to define the word "little." We may write an expression which describes what this word means. Our definition relies on the meanings of all of the words in our expression. We could then write expressions to define the words used in the definition. Then we could try to define these words, and so on. Since there are only finitely many words in the English language, one of two things must happen – either we reuse words (and end up with a circular definition) or we find words that are not defined.

An extreme example of this can be found if you look up "little." Many dictionaries will have this definition: "small in size." The same dictionaries will define "small" as "little in size." If we do not know what "little" or "small" means, these dictionaries are useless.

To avoid circular definitions, mathematicians begin with *primitives* – undefined terms. The most fundamental primitive in all of mathematics is *set*. We will not define what a set is. Presumably, *collection* is a synonym. Sets contain (another primitive) things which we call *elements*. To indicate that an element x is in a set A, we write $x \in A$. This notation can be read as "x is an element of A," or as "x is in A," or if necessary, "x in A." To express that x is not in A, we would write $x \notin A$.

If we can list the elements of a set, we will do so between braces. For example, the set containing the symbols a, b, 1, and 2 is $\{a, b, 1, 2\}$. We can even use braces to list infinite sets if there is a clear pattern. For example, the even integers are $\{\ldots, -4, -2, 0, 2, 4, 6, \ldots\}$. Order and repetition within braces do not matter. For example, the sets $\{a, b, c\}$ and $\{c, b, a, a, b, c, c\}$ are the same sets.

Two important sets which we will use frequently are the *integers*

$$\mathbb{Z} = \{0, -1, 1, -2, 2, -3, 3, \ldots\}$$

and the *natural numbers*[a]

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}.$$

[a]Historically, the definition of the natural numbers did not include 0. Many modern mathematicians include 0 in $\mathbb{N}$. Doing so pains me greatly. Most societies took centuries to realize the need for 0 as a number, so this number does not seem remotely natural. However, including 0 here will make the lives of my students easier, and it provides for elegant parallels between arithmetic in $\mathbb{N}$ and set theory. So, after decades of resistance, I am finally conceding and allowing 0 to be a natural number – at least for this class.

To describe a set which is too big or complicated to simply list, we can sometimes use **set builder** notation. This notation looks like:

$$\{x : P(x)\} \qquad \text{or} \qquad \{x \in S : P(x)\}.$$

The colon inside the braces is read as "such that." The notation on the left is defined to mean the set of all x such that $P(x)$ is true. This means that an element x is in the set if and only if the predicate $P(x)$ is true. The notation on the right is similar; however, this notation lets us restrict our attention to things which are in the set S. This is the set of all x in the set S for which the statement $P(x)$ is true.

Some examples of the use of set builder notation are:

Example 17.1: If $H(x)$ is "x is a horse" and $B(x)$ is "x is brown" then

$$\{x : H(x) \wedge B(x)\}$$

is the set of all brown horses.

Example 17.2: The set of days of the week in set builder notation is

$$\{x : x \text{ is a day of the week}\}.$$

Of course, we could also simply list the elements of this set:

{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}.

Example 17.3: If L is a sentential language, and s is a truth assignment on L, then the set of all wffs which s makes true is

$$\{\alpha : s(\alpha) = T\}.$$

We could be slightly more specific

$$\{\alpha : \alpha \text{ is a wff over } L \text{ and } s(\alpha) = T.\}$$

Example 17.4: Suppose that L is a sentential language. Let W be the set of all wffs over L, and let Σ be a collection of wffs over L. Then

$$\{\alpha \in W : \Sigma \vdash \alpha\}$$

is the set of all sentences provable from Σ.

Example 17.5: The set of even integers can be expressed this way:

$$\{x : \exists n[(n \in \mathbb{Z}) \wedge (x = 2n)]\}$$

or we can cheat a bit. This is a common way to describe the set of even integers:

$$\{2n : n \in \mathbb{Z}\}$$

Example 17.6: The set of positive integers can be expressed as

$$\{x : (x \in \mathbb{Z}) \wedge (x > 0)\}$$

or

$$\{x \in \mathbb{Z} : x > 0\}.$$

17.2 Subsets

A set A is a *subset* of another set B if every element of A is also an element of B. We denote this relationship by $A \subseteq B$. This notation is read as "A is a subset of B." For example, $\{1, 2, 3\}$ is a subset of $\{1, 2, 3, 4\}$, so $\{1, 2, 3\} \subseteq \{1, 2, 3, 4\}$.

Caution: It is very important not to confuse the symbols $\in$ and $\subseteq$. The notation $X \in Y$ implies that Y is a set and X is a single element of that set. The notation $X \subseteq Y$ implies that X and Y are both sets and X is a subset of Y. However, there are times when X and Y might both be sets and we still have $X \in Y$. For example, this happens if $X = \{1, 2\}$ and $Y = \{\{1, 2\}, 3\}$. Note here that Y is a set with two elements. They are $\{1, 2\}$ and 3.

The *empty set* (denoted $\emptyset$) is the set $\{\}$ which contains no elements. The empty set is a subset of every set (including itself) since the implication "If $x \in \emptyset$ then $x \in A$" is always true because $x \in \emptyset$ is always false.

17.3 Exercises

Let $E(x)$ be "x is even." Let $O(x)$ be "x is odd." Let $P(x)$ be "x is prime." Recall that $\mathbb{Z}$ is the set of integers, so to say "x is an integer" you can write $x \in \mathbb{Z}$. Write the following sets in set-builder notation.

1. The set of integer multiples of 3.
2. The set of odd integers less than 10.
3. The set of all integers which are either even or greater than 10.
4. The set of all integers which are prime.
5. The set of all even prime integers.

List the elements of these sets

6. $\{x \in \mathbb{N} : x^2 < 10\}$
7. $\{x \in \mathbb{N} : (x < 20) \wedge \exists y[(y \in \mathbb{N}) \wedge ((x = 3y) \vee (x = 5y))]\}$
8. $\{n : (n \in \mathbb{N}) \wedge \exists m[(m \in \mathbb{N}) \wedge (n = m^2)] \wedge (n < 20)\}$
9. $\{x \in \mathbb{N} : \exists a \exists b[(a \in \mathbb{N}) \wedge (b \in \mathbb{N}) \wedge (a > 1) \wedge (b > 1) \wedge (a < 5) \wedge (b < 5) \wedge (x = ab)]\}$

These exercises explore the number of subsets of a finite set.

10. List all of the subsets of $\emptyset$
11. List all of the subsets of $\{1\}$
12. List all of the subsets of $\{1, 2\}$
13. List all of the subsets of $\{1, 2, 3\}$
14. How many subsets does $\{1, 2, 3, 4\}$ have?

17.4 Set Operations

In sentential logic, we had logical connectives or logical operators which we used to create new statements from existing statements. We also have set operations by which we can create new sets from existing sets.

Set Operations: Suppose that A and B are sets.
Intersection: The **intersection** of A and B is the set

$$A \cap B = \{x : (x \in A) \wedge (x \in B)\}.$$

Union: The **union** of A and B is the set

$$A \cup B = \{x : (x \in A) \vee (x \in B)\}.$$

Difference: The difference of A and B is the set

$$A - B = \{x : (x \in A) \wedge (x \notin B)\}.$$

Note that the symbol for intersection resembles that for "and." This is no accident. The word "and" is important in the definition of intersection. Similarly, note that the symbol for union resembles that for "or."

17.5 Exercises

Let $A = \{a, b, c, d, e\}$, $B = \{c, d, f\}$, and $C = \{a, f\}$. Find the indicated sets.

1. $A \cup B$
2. $A \cap B$
3. $A - B$
4. $(A \cap B) \cup C$
5. $A - (B \cup C)$
6. $(A - B) \cup B$
7. $(A - B) \cup (B - A)$
8. $(A \cup B) - (A \cap B)$

17.6 Identities

The definitions of our set operations are intimately related to our logical connectives. A result of this fact is that logical equivalences can be used to

demonstrate certain equalities or identities among sets. We illustrate this here and list some identities which directly mimic our logical equivalences.

Let A, B, and C be sets and consider $A \cap (B \cup C)$. By applying the definitions above and the distributive law of $\wedge$ over $\vee$, we see that

$$\begin{array}{lll} & x \in A \cap (B \cup C) & \\ \text{iff} & (x \in A) \wedge (x \in B \vee x \in C) & \text{definition} \\ \text{iff} & (x \in A \wedge x \in B) \vee (x \in A \wedge x \in C) & \text{distributive law} \\ \text{iff} & (x \in (A \cap B)) \vee (x \in (A \cap C)) & \text{definition.} \\ \text{iff} & x \in (A \cap B) \cup (A \cap C) & \text{definition.} \end{array}$$

Thus, $\cap$ distributes over $\cup$. We can apply other logical equivalences to derive these identities for set operations.

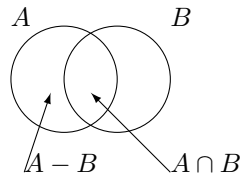
Identities for Set Operations

The following are true for any sets A, B, and C.

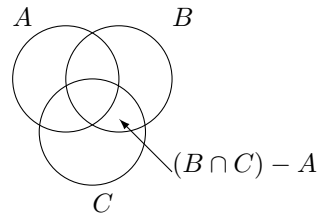
$A \cap B = B \cap A$	commutative law
$A \cup B = B \cup A$	commutative law
$A \cap (B \cap C) = (A \cap B) \cap C$	associative law
$A \cup (B \cup C) = (A \cup B) \cup C$	associative law
$A \cap A = A$	idempotent law
$A \cup A = A$	idempotent law
$A \cap (A \cup B) = A$	absorption law
$A \cup (A \cap B) = A$	absorption law
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	distributive law
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	distributive law
$A - (B \cap C) = (A - B) \cup (A - C)$	DeMorgan's Law
$A - (B \cup C) = (A - B) \cap (A - C)$	DeMorgan's Law

17.7 Venn Diagrams

Diagrams can be drawn to help visualize set operations. Each set is represented by an oval or a circle. The intersection, union, and difference of the sets can then be seen geometrically:



The union of A and B would be all of the area inside either circle. These diagrams of circles are called **Venn Diagrams**. Venn Diagrams can also be used to visualize more complicated set operations:



17.8 Exercises

Draw three circles as above and shade each of the following sets.

1. $(A \cap B) \cup (A \cap C)$
2. $A - (A \cap B \cap C)$
3. $(C - (A \cap B)) - (A \cap C)$
4. Draw Venn Diagrams to convince yourself of the absorption laws.

Chapter 18

Relations

Predicates express properties of and relationships between the variables involved. For example

x is taller than y.

expresses a relationship between x and y. The sentence

x robbed the bank.

expresses the property that x may have of having robbed the bank. Mathematicians use the notion of a relation to describe abstract properties and relationships.

P is 1-ary (or unary) relation on a set A if P is a subset of A. When thinking of a subset P as a relation, we will write $P(x)$ for $x \in P$. If we let P be the set of all people who robbed the bank. Then $P(x)$ and $x \in P$ both mean the same thing as the predicate "x robbed the bank."

P is a 2-ary (or binary) relation on a set A if P is a set of ordered pairs of A (such as (x, y)). When thinking of P as a relation, we will write $P(x, y)$ for $(x, y) \in P$. If we let P be the set of all pairs of people (x, y) so that x is the father of y, then $P(x, y)$ and $(x, y) \in P$ both mean the same thing as the predicate "x is the father of y."

P is a 3-ary (or ternary) relation on a set A if P is a set of ordered triples of A (such as (x, y, z)). When thinking of P as a relation, we will write $P(x, y, z)$ for $(x, y, z) \in P$.

We can continue this process indefinitely. If n is a positive integer, then P is an n-ary relation on a set A if P is a set of n-tuples of A (such as $(x_1, x_2, \ldots, x_n)$). When thinking of P as a relation, we will write

$$P(x_1, x_2, \ldots, x_n) \text{ for } (x_1, x_2, \ldots, x_n) \in P.$$

For emphasis, we will also say

"$P(x_1, x_2, \ldots, x_n)$ is true" to mean that "$(x_1, x_2, \ldots, _n) \in P$ is true."

The notation we are using for relations is intended to match exactly the notation we have used for predicates and open formulas. The three concepts are intimately related. An open formula $\alpha(x, y)$ can be used to define a set – the set P of all pairs (x, y) for which $\alpha(x, y)$ is true (whatever that means). In set builder notation, $P = \{(x, y) : \alpha(x, y)\}$. This set of ordered pairs can be treated as a relation. In this case, $P(x, y)$ means that $(x, y) \in P$, which means that $\alpha(x, y)$ is true. Both $P(x, y)$ and $\alpha(x, y)$ can be interpreted as the predicate "The ordered pair (x, y) is in P."

Example 18.1: Suppose that $A = \{1, 2, 3, 4\}$. Define this relation on A:

$$P = \{(x, y, z) : y \text{ is strictly between } x \text{ and } z\}.$$

Then we would say that $P(1, 2, 3)$ is true but $P(2, 1, 3)$ is not. We equate $P(x, y, z)$ with the defining predicate "y is strictly between x and z."

Example 18.2: Let $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let

$$P = \{x \in A : x \text{ is prime}\}.$$

Then we equate the expression $P(x)$ with the predicate "x is prime." Thus $P(3)$ is true, but $P(4)$ is not. We can list the elements of P as $\{2, 3, 5, 7\}$.

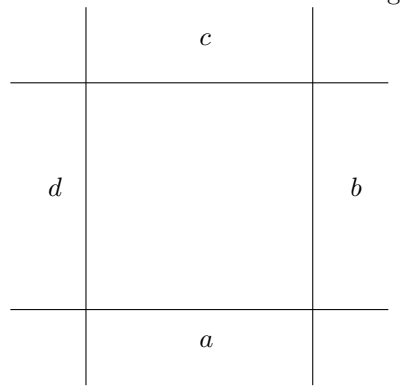
We will often use special notation for binary relations. For some binary relations, we will write xRy for $R(x, y)$. Equality is a binary relation, but we do not write $= (x, y)$. Instead, we write $x = y$. Another example is $\leq$. We write $x \leq y$ rather than $\leq (x, y)$.

Example 18.3: Let A be the set $\{2, 3, 4, 5, 6\}$. Define a binary relation D on A so that for all $x, y \in A$, xDy exactly when y is a multiple of x. Then $2D4$ and $3D6$, but it is not the case that $2D3$. We could write these statements also as $D(2, 4)$ and $D(3, 6)$ but not $D(2, 3)$. As a set of ordered pairs, D is

$$D = \{(2, 2), (2, 4), (2, 6), (3, 3), (3, 6), (4, 4), (5, 5), (6, 6)\}.$$

As a predicate, xDy means "y is a multiple of x."

Example 18.4: Let A be the set of lines in this figure:



So $A = \{a, b, c, d\}$. Define a relation I on A so that xIy exactly when line x intersects line y. Then (for example) aId and aIb but it is not the case that aIc.

18.1 Exercises

1. List two more pairs which are not in the relation I of Example 18.4 and two more pairs that are.
2. Define a relation S on $\mathbb{N}$ so that $S(x, y)$ means that $x + 1 = y$. List a few pairs in the relation S. We will see later that most of the properties of the natural numbers which we studied in arithmetic can be derived from this relation.
3. Let $A = \{1, 2, 3, 4, 5, 6\}$. Define R on A so that xRy means that $x - y \in A$. List the pairs in R.
4. Let E and F be two unary relations on $\mathbb{N}$ so that $E(x)$ means "x is even" and $F(x)$ means "x is a multiple of 4."
   (a) Find an x which makes $E(x) \wedge F(x)$ true.
   (b) Find an x which makes $E(x) \wedge \neg F(x)$ true.
   (c) Which of these is true?
      i. $\forall x(E(x) \to F(x))$
      ii. $\forall x(F(x) \to E(x))$
      iii. $\exists x(E(x) \to F(x))$
      iv. $\exists x(F(x) \to E(x))$

Chapter 19

Models

The truth value of a sentential wff was defined in terms of truth values assigned to the basic sentence symbols. In predicate logic, each basic predicate symbol may be sometimes true and sometimes false based on what objects are substituted for the variables involved. Thus truth values will also be affected by what objects are allowed to be substituted for variables.

Suppose that L is a set of predicate symbols. To discuss truth values of sentences over L, we will first restrict our attention to a set A which we will call our universe that contains elements which are "legal" to substitute for variables. Each predicate symbol in L is associated to an actual relation on A. The set A with the associated relations will be called an L-structure. Truth or falseness of sentences and formulas can then be considered within this structure or toy world. Elements of A can be substituted for variables in formulas. Truth values then become a question of containment in relations.

19.1 Structures

Suppose that L is a set of predicate symbols. An *L-structure* is a set A (called the *universe* of the structure) along with for any n-ary predicate symbol P in L an n-ary relation P^A on A.

Suppose that L contains a single binary predicate symbol R. Then an L-structure is simply a set A along with a binary relation R^A on A. The superscripted A just reflects that R^A is a specific relation on the set A. If we are working in an environment where there is not much chance of confusion, we will drop the superscript. We will usually express an L-structure this way:

$$\mathbf{A} = \langle A, R \rangle.$$

We use plaintext for the set which is the universe and boldfaced text for the name of the structure.

Example 19.1: Suppose L contains a single binary predicate symbol R. Let $B = \{1, 2, 3, 4, 5\}$. Define R on B so that xRy means $x \leq y$. Then $\mathbf{B} = \langle B, R\rangle$ is an L-structure.

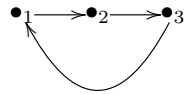
Example 19.2: Let $A = \{1, 2, 3\}$ and define R on A so that $1R2$ and $2R3$ and $3R1$. Let $\mathbf{A} = \langle A, R\rangle$. Then $\mathbf{A}$ is also an L-structure where L is as in the previous example.

Example 19.3: Suppose that M contains a ternary predicate P and a binary predicate symbol Q. Let $C = \{0, 1, 2, 3, 4, 5\}$. Define P on C so that $P(x, y, z)$ means that $x + y = z$. Define Q on C so that xCy means $x + y = 5$. Then $\mathbf{C} = \langle C, P, Q\rangle$ is an M-structure.

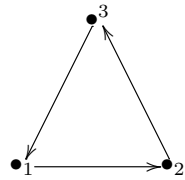
Example 19.4: Let D be the set of all living people in the United States. Define F on D so that $F(x)$ means x is female. Define M on D so that $M(x)$ means x is male. Define $R(x, y)$ to mean x is married to y. Then $\mathbf{D} = \langle D, F, M, R\rangle$ is an L-structure where $L = \{F, M, R\}$.

We can draw pictures of structures of languages which have only one binary relation. The picture consists of one point for each element of the universe of the structure. For each relation xRy, there is an arrow in the picture from x to y. Such a pictorial representation is called a *directed graph* or a *digraph*.

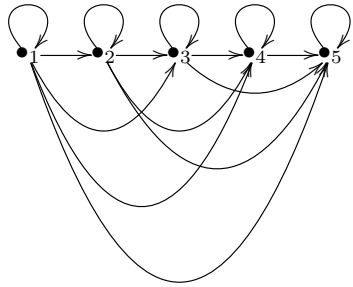
Example 19.5: The directed graph for the structure in Example 19.2 can be draw this way:



What matters in directed graphs is how the points are connected by arrows, not the physical arrangement of the points in the picture. Thus, this is another depiction of the graph from Example 19.2:



Example 19.6: The directed graph for the structure in Example 19.1 can be draw this way:



19.2 Satisfaction

We give here a recursive characterization of what it means for a sentence to be true in a structure. However, while the recursive definition is good for induction proofs and for making further definitions about satisfaction, it may do little for reasoning. When deciding the truth value of a sentence in a model, interpret $\forall$ and $\exists$ in the obvious way, substitute elements of the universe for variables, and use the relations of the model and what we know about the logical connectives to determine truth values. (This means you might want to skip to the examples and come back to read the definition later.)

Suppose that γ is a wff in a predicate language L and that the free variables in γ are among $v_1, \ldots, v_n$. Suppose also that $\mathbf{A}$ is an L-structure and that $a_1, a_2, \ldots a_n \in \mathbf{A}$. We define recursively what it means for $\gamma(a_1, a_2, \ldots, a_n)$ to be true. Our definition is based on the recursive definition of wffs in Chapter 9. It mimics in part the recursive definition of how to extend a truth assignment given in Chapter 2.

1. If $\gamma(v_1, \ldots, v_n)$ is $v_i = v_j$, then for any $a_1, \ldots, a_n \in \mathbf{A}$ $\gamma(a_1, \ldots, a_n)$ is true if and only if $a_i = a_j$.

2. If $\gamma(v_1, \ldots, v_n)$ is $P(v_1, \ldots, v_n)$ for some predicate symbol P, then for any $a_1, \ldots, a_n \in \mathbf{A}$ $\gamma(a_1, \ldots, a_n)$ is true if and only if $P^A(a_1, \ldots, a_n)$.

3. Suppose that $\alpha(v_1, \ldots, v_n)$ and $\beta(v_1, \ldots, v_n)$ have been given truth values for every assignment of variables in $\mathbf{A}$. The truth values of $\neg\alpha$, $\alpha \vee \beta$, $\alpha \wedge \beta$, $\alpha \rightarrow \beta$, and $\alpha \leftrightarrow \beta$ are defined according the their respective sentential logical connectives.

4. Suppose that $\alpha(v_1, \ldots, v_n, v)$ is a wff whose truth values have been assigned for all substitutions of its free variables. Let $a_1, \ldots, a_n \in \mathbf{A}$.

   (a) If γ is $\forall v\alpha$. Then $\gamma(a_1, \ldots, a_n)$ is true if and only if $\alpha(a_1, \ldots, a_n, b)$ is true for every b in $\mathbf{A}$.

(b) If γ is $\exists v\alpha$. Then $\gamma(a_1, \ldots, a_n)$ is true if and only if $\alpha(a_1, \ldots, a_n, b)$ is true for at least one b in $\mathbf{A}$.

Notice that if α is a sentence (i.e. α has no free variables) then in any model, this definition gives a truth value for α which does not depend on any substitution of variables. If α is true in a structure $\mathbf{A}$, we will write $\mathbf{A} \models \alpha$. In this case, we say that $\mathbf{A}$ *models* or *satisfies* α. If Σ is a set of sentences in L, then $\mathbf{A} \models \Sigma$ means that $\mathbf{A} \models \alpha$ for every sentence in $\mathbf{\Sigma}$.

If a wff α has a free variable v, then we say that a α is true in a structure $\mathbf{A}$ (or that $\mathbf{A}$ satisfies α, or $\mathbf{A} \models \alpha$) if $\forall v\alpha$ is true in $\mathbf{A}$.

As we said before, the recursive definition is good for induction proofs and for making further definitions about satisfaction, but it may do little for reasoning. When deciding the truth value of a sentence in a model, interpret $\forall$ and $\exists$ in the obvious way, substitute elements of the universe for variables, and use the relations of the model and what we know about the logical connectives to determine truth values.

Example 19.7: The structure $\mathbf{B}$ of Example 19.1 satisfies these sentences:

1. $\exists x \forall y(xRy)$
2. $\forall x \forall y(xRy \vee yRx)$

The structure does not satisfy these sentences:

3. $\exists x \exists y[(x \neq y) \wedge xRy \wedge yRx]$
4. $\exists x \neg(xRx)$

Sentence (1) says that there is an element which is related to every element. In this case, that means that there is an element which is less than or equal to every element. Sentence (2) says that any two elements are related one way or the other.

Sentence (3) says that there are nonequal elements which are related "both ways." Here this means that there are two distinct numbers which are each less than the other. Sentence (4) says that there is a number not related to itself. Here, this means that there is a number which is not less than or equal to itself.

Example 19.8: The structure $\mathbf{C}$ of Example 19.3 satisfies these sentences:

1. $\forall x \forall y \forall z[P(x, y, z) \leftrightarrow P(y, x, z)]$
2. $\forall x \exists y(xCy)$

The structure does not satisfy these sentences:

3. $\forall x P(x, x, x)$
4. $\exists x \forall y(xCy)$

Sentence (1) says that $x + y = z$ if and only if $y + x = z$. Sentence (2) declares that for each x in M the number $5 - x$ is in M.

Sentence (3) says that $x + x = x$ is always true. Sentence (4) says that there is a single x which you can add to any y to get 5. This would imply that $5 - y$ is the same number for every value of y.

Example 19.9: The structure **D** of Example 19.4 satisfies these sentences:

1. $\exists x \forall y \neg R(x, y)$
2. $\forall x \forall y (R(x, y) \rightarrow R(y, x)$
3. $\neg \exists x \exists y \exists z [(x \neq y) \wedge R(x, z) \wedge R(y, z)]$
4. $\forall x [(F(x) \vee M(x)) \wedge \neg (F(x) \wedge M(x))]$

Until recently, **D** satisfied

5. $\forall x, y [R(x, y) \rightarrow ([F(x) \wedge M(y)] \vee [M(x) \wedge F(y)])]$

The structure does not satisfy these sentences:

6. $\exists x R(x, x)$
7. $\exists x \exists y \exists z [(x \neq y) \wedge R(x, z) \wedge R(y, z)]$

Sentence (1) says that there is someone who is not married to anyone. Sentence (2) says that if x is married to y, then y is married to x. Sentence (3) declares that marriage is monogomous. Sentence (4) declares that every persons is male or female and not both. Sentence (5) declares that every married couple includes a male and a female.

Sentence (6) declares that some person is married to himself or herself. Sentence (7) declares that someone is married to two distinct people.

19.3 Models

Suppose that L is a set of predicate symbols and that Σ is a set of sentences over L. A *model* of Σ is an L-structure **A** for which $\mathbf{A} \models \Sigma$ (that is, every sentence in Σ is true in **A**).

Suppose that R is a binary relation on a set A. Let $\mathbf{A} = \langle A, R \rangle$. R is *reflexive* if **A** satisfies

$$\forall x (xRx).$$

R is *symmetric* if **A** satisfies

$$\forall x \forall y (xRy \rightarrow yRx).$$

R is *antisymmetric* if **A** satisfies

$$\forall x \forall y [(x \neq y) \rightarrow \neg (xRy \wedge yRx)].$$

R is *transitive* if $\mathbf{A}$ satisfies

$$\forall x \forall y \forall z[(xRy \wedge yRz) \rightarrow xRz].$$

These four possible properties of a binary relation are inspired by properties of $=$ and $\leq$. Any reflexive, symmetric, and transitive relation is called an *equivalence relation*. Any reflexive, antisymmetric, and transitive relation is called an *order relation*. If R is an order relation on a set A, then $\langle A, R\rangle$ is called an *ordered set*.

Some examples of ordered sets are:

Example 19.10: The usual order $\leq$ on the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ is an order relation, so $\langle \mathbb{N}, \leq\rangle$ is an ordered set.

Example 19.11: If A is any set the the set of subsets of A with the relation $\subseteq$ is an ordered set. If $A = \{1, 2, 3\}$, then the ordered set of subsets of A contains eight elements: $\emptyset$, $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{1, 2, 3\}$.

Example 19.12: Define a relation $|$ on $\mathbb{N}$ so that $x|y$ means that y is a multiple of x (or x is a factor of y). Then $|$ is an order relation.

Let G be the language with two unary predicate symbols P and L and a binary predicate symbol ON. We will call any model of these sentences:

$$\forall x(P(x) \vee L(x))$$

$$\neg\exists x(P(x) \wedge L(x))$$

$$\forall x \forall y(xONy \rightarrow ((P(x) \wedge L(y)) \vee (P(y) \wedge L(x))))$$

$$\forall x \forall y(xONy \rightarrow yONx)$$

a *geometry*. The formula $L(x)$ is intended to mean "x is a line." $P(x)$ is intended to mean "x is a point." The relation $xONy$ is intended to mean "x is on y." The first two sentences here guarantee that each "thing" in our model is either a point or a line. The second insists that points and lines are different. The third sentence allows points to be on lines and lines to be on points but not points on points or lines on lines. The last sentence insists that saying a point is on a line is the same as saying the line is on the point.

Example 19.13: Let L be the set of all lines on the plane. Let P be the set of all points on the plain. Let $A = L \cup P$. Treat the subsets L and P as unary relations. Let ON be the set of all pairs (p, l) where p is a point and l is a line with p on l. Then $\langle A, P, L, ON\rangle$ is a geometry.

We can draw pictures to depict structures which are geometries. Those x for which $P(x)$ is true (the points) will be represented with •s. Those x for which $L(x)$ is true (the lines) will be represented with line segments or curves. The relationship ON should be almost obvious.

Example 19.14: Here are two depictions of geometries.
This geometry has three points and three lines.

This geometry has four points and six lines. Notice that lines only intersect at the corners of the square.

19.4 Exercises

1. Let $A = \{0, 1, 2, 3, 4, 5\}$ and $B = \{0, 1\}$. Consider $\langle A, \leq\rangle$, $\langle B, \leq\rangle$, $\langle \mathbb{N}, \leq\rangle$, and $\langle \mathbb{Z}, \leq\rangle$ with the usual orders. Which of these structures are models of these sentences?
   (a) $\forall x \forall y \forall z[(x = y) \vee (x = z) \vee (y = z)]$
   (b) $\exists x \forall y(x \leq y)$
   (c) $\exists x \exists z \forall y[(x \leq y) \wedge (y \leq z)]$
   (d) $\forall x \exists y(x \leq y)$
   (e) $\exists x \exists y \exists z[(x \neq y) \wedge (x \neq z) \wedge (y \neq z)]$

2. Draw the digraph of a model of each of these sentences. Your language has one binary relation symbol $\Rightarrow$.
   (a) $\forall x(x \Rightarrow x)$
   (b) $\exists x \forall y(x \Rightarrow y)$
   (c) $\forall x \forall y(x \Rightarrow y)$
   (d) $\exists x \exists y \exists z[(x \Rightarrow y) \wedge (y \Rightarrow z)]$
   (e) $\forall x \forall y \neg[(x \Rightarrow y) \wedge (y \Rightarrow x)]$
   (f) $\forall x \forall y([(x \Rightarrow y) \vee (y \Rightarrow x)] \wedge \neg[(x \Rightarrow y) \wedge (y \Rightarrow x)])$

3. Let sentences $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ be define this way.

$$\begin{array}{lll} \alpha_1 & \text{is} & \forall x \forall y[(P(x) \wedge P(y)) \rightarrow \exists z(xONz \wedge yONz)] \\ \alpha_2 & \text{is} & \forall x \forall y[(L(x) \wedge L(y)) \rightarrow \exists z(zONx \wedge zONy)] \\ \alpha_3 & \text{is} & \exists x[P(x) \wedge \forall y(L(y) \rightarrow xONy)] \\ \alpha_4 & \text{is} & \exists x[L(x) \wedge \forall y(P(y) \rightarrow yONx)] \end{array}$$

Draw geometries which satisfy these statements or explain why this is not possible.

(a) α_1, α_2, α_3 but not α_4

(b) α_1, α_2, α_4 but not α_3

(c) α_1, α_3, α_4 but not α_2

(d) α_2, α_3, α_4 but not α_1

19.5 Definable Subsets and Relations

If L is a predicate language and $\mathbf{A}$ is an L-structure, then the predicate symbols in L can be used to define new relations or subsets or even elements of $\mathbf{A}$.

Example 19.15: In $\langle \mathbb{N}, \leq \rangle$, let $P(x)$ be the predicate $\forall y(x \leq y)$. The only element of $\mathbb{N}$ for which $P(x)$ is true is $x = 0$. That is, $P(0)$ is true, but $P(x)$ is false for $x \neq 0$. We say that $P(x)$ uniquely defines the element 0.

Here, P defines the least element of $\mathbb{N}$.

Example 19.16: Suppose that $\mathbf{A} = \langle A, \leq \rangle$ is an ordered set. We can define a binary relation $<$ on $\mathbf{A}$ so that $x < y$ means $(x \leq y) \wedge (x \neq y)$. The relationship between $\leq$ and $<$ is the usual relationship between these symbols on sets of numbers.

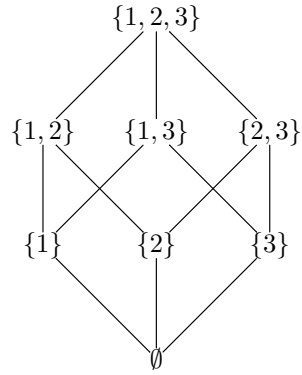
Example 19.17: Suppose that $\mathbf{A} = \langle A, \leq \rangle$ is an ordered set. Define a relation $\prec$ on A so that $x \prec y$ means $(x \leq y) \wedge \forall z[((x \leq z) \wedge (z \leq y)) \rightarrow ((x = z) \vee (y = z))]$. Then $x \prec y$ means that $x \leq y$ and that there are no elements of our model between x and y. The relation $\prec$ is called the *covering relation* of $\mathbf{A}$.

The covering relation of an ordered set can allow us to draw pictures called *Hasse diagrams* of finite ordered sets. Points labeled by the elements of the ordered set are arranged on the page so that if $x \leq y$, then the point for x is lower on the page than that for y. Then lines or curves are drawn from x up to y if $x \prec y$.

Example 19.18: The Hasse diagram of $\{1, 2, 3\}$ under the usual order $\leq$ is

In this model, each element is uniquely definable. 1 is the only element which is less than or equal to every element. 2 is the only element which is both strictly less than and strictly greater than some elements. 3 is the only greater than or equal to every element.

Example 19.19: The Hasse diagram of the set of subsets of $\{1, 2, 3\}$ looks like



19.6 Exercises

Write sentences describing these subsets of an ordered set.

1. The least element.
2. The set of minimal (bottom) elements of the ordered set.
3. The set of elements which are not minimal or maximal.
4. The set of elements which are comparable to every element in the ordered set (x and y are comparable if either $x \leq y$ or $y \leq x$).

Within the natural numbers with the usual order, write sentences which define the following elements and sets.

1. The number 1
2. The number 2

3. The set $\{0, 1, 2\}$
4. The set $\{3, 4, 5, \ldots\}$

1. Is 0 definable in $\langle \mathbb{Z}, \leq \rangle$?
2. Is 0 definable in $\mathbb{Z}$ if you can use $+$?
3. Is 0 definable in $\mathbb{Z}$ if you can use $\cdot$?
4. What operations in $\mathbb{Z}$ would you need to define 1?
5. What operations in $\mathbb{Z}$ would you need to define the natural numbers?

Draw a Hasse diagram of an ordered set satisfying each of these sentences.

1. $\exists x \forall y (x \leq y)$
2. $\exists x \exists z \forall y [(x \leq y) \wedge (y \leq z)]$
3. $\exists x \exists y [(x \neq y) \wedge \forall z ([x \leq z] \vee [y \leq z])]$
4. $\exists x \exists y \neg [(x \leq y) \vee (y \leq q)]$
5. $\exists x \exists y \exists z \exists w [(x \leq y) \wedge (z \leq y) \wedge (z \leq w)]$

19.7 Logical Implication

Suppose that Σ is a set of wffs of a first order language L and that α is any wff over L. We say that Σ *logically implies* α if every model of Σ satisfies α. In symbols we express this as $\Sigma \models \alpha$ (read as "Σ entails α" or "Σ logically implies α").

This definition of logical implication is quite similar to logical implication for sentential logic. In fact, any entailment from sentential logic will carry over to predicate logic. The reason for this is how we recursively defined satisfaction to coincide with truth assignments. The question is whether the tie to models in predicate logic forces some entailments to hold beyond those carried over from sentential logic. That is, do predicates, quantifiers, and the possible complexity of models make entailment more complicated?

As with sentential logic, the most important example of entailment is this one.

Example 19.20: Modus Ponens: $\{\alpha \to \beta, \alpha\} \models \beta$
If a structure satisfies $\alpha \to \beta$ and α, then by the recursive definition of satisfaction it must also satisfy β.

19.8 Consistency, Completeness, Compactness

We have defined three manners of implication in first order logic:

> **Logical Implication:** $\Sigma \models \alpha$ if every model of Σ must satisfy α.
>
> **Deduction:** $\Sigma \vdash \alpha$ if there is a proof of α from Σ.
>
> **Sentential Implication:** $\Sigma \models_S \alpha$ if Σ logically implies α when these are viewed as sentential wffs over a language whose sentence symbols are the prime formulas in the predicate language.

The Sentential Implication Theorem 16.3 provides a connection between $\vdash$ and $\models_S$. This, combined with the fact that $\vdash$ and $\models$ are equivalent for sentential logic via the Soundness and Compactness Theorems, should hint that there is a strong connection between these notions for predicate logic.

We can establish Soundness for first order predicate logic in the following manner. Suppose that $\Sigma \vdash \alpha$. We can (or could) prove by induction that $\Sigma \models \alpha$. If $\alpha \in \Sigma$, then clearly $\Sigma \models \alpha$. If α is provable because $\Sigma \vdash \gamma \to \alpha$ and $\Sigma \vdash \gamma$, then $\Sigma \models \alpha$ will follow from the fact that Modus Ponens preserves truth. The hard part of the induction is if α is a logical axiom (a member of Δ). In this case, we must prove (by induction) that $\models \alpha$ – that is, every model of our predicate language must satisfy α. This proof would require a deeper discussion of substitution than we have had (or will have). We must be content with the idea that our axioms are reasonable enough to hold in any model. In summary, if our axioms are valid, since Modus Ponens preserves truth, $\vdash$ should imply $\models$.

Our definition of consistency for predicate logic is identical to that in sentential logic: Σ is *consistent* if Σ cannot prove any contradiction. Σ is *inconsistent* if Σ can be used to prove a contradiction. Using the same tricks from sentential logic, we can prove that an inconsistent set of wffs can be used to prove anything.

Satisfiability for first order predicate logic appears more complicated than sentential logic because truth assignments have been replaced (in some sense) with models. A collection Σ of first order formulas is *satisfiable* if there is a structure $\mathbf{A}$ which models Σ (ie if there is a structure which satisfies the formulas in Σ). We may use the words "can be modelled" for "is satisfiable."

Suppose that Σ has a model. If Σ were inconsistent and could be used to prove a contradiction, then that contradiction would have to be true in the model. Since this is not possible, then Σ must be consistent. This combined with the discussion above about Modus Ponens preserving truth (would) give us Soundness Theorems.

Theorem 19.21: Soundness Theorems – First Order Logic

- If $\Sigma \vdash \alpha$, then $\Sigma \models \alpha$.
- If Σ is satisfiable, then Σ is consistent.

Here is a flawed attempt at extending Completeness to first order logic: Suppose that $\Sigma \models \alpha$. *Then any truth assignment to prime formulas which makes the formulas in* Σ *true must also make* α *true.* That is, $\Sigma \models_S \alpha$. It follows that $\Sigma, \Delta \models_S \alpha$ since any truth assignment that makes $\Sigma \cup \Delta$ true must make Σ true, and hence, α also. But by the Sentential Implication Theorem 16.3, this implies that $\Sigma \vdash \alpha$.

The problem with this argument is in the italicized sentence. $\Sigma \models \alpha$ means that whenever the formulas in Σ are true *in a model*, then α is true in that model. Suppose that there is a truth assignment to prime formulas that cannot be realized in a model. $\Sigma \models \alpha$ can say nothing about this truth assignment. This assignment may make the formulas in Σ true, but since there is no model to connect this truth to α, we would have no idea about the truth value of α. To extend Completeness to first order logic, we need a version of the Completeness Theorem claiming consistency implies satisfiability. This is well beyond the tools we have here, so we will simply note the Completeness and Compactness Theorems for Predicate Logic.

Theorem 19.22: Completeness Theorems - First Order Logic

- If $\Sigma \models \alpha$, then $\Sigma \vdash \alpha$.
- If Σ is consistent, then Σ is satisfiable.

Theorem 19.23: Compactness Theorem – First Order Logic

- If $\Sigma \models \alpha$ then there is some finite set Σ_0 of wffs in Σ so that $\Sigma_0 \models \alpha$.
- If every finite subset of Σ is satisfiable, then Σ is satisfiable.

Chapter 20

Some Geometry

In Chapter 19 we introduced geometries as structures in a language with two unary predicates P and L and a binary predicate ON that satisfy these axioms:

G1 $\forall x(P(x) \vee L(x))$

G2 $\neg\exists x(P(x) \wedge L(x))$

G3 $\forall x\forall y(xONy \rightarrow ((P(x) \wedge L(y)) \vee (P(y) \wedge L(x)))$

G4 $\forall x\forall y(xONy \rightarrow yONx)$

The predicates P and L are used to refer to points and lines, respectively. The first two axioms insist that every element is either a point or a line but not both. The third axiom insists that points may be on lines or lines on points, but not points on points or lines on lines. The fourth axiom declares that saying a point is on a line is the same as saying that the line is on the point. In this chapter we investigate these geometries to get practice with some concepts in first order predicate logic. First, we use P, L, and ON to define new terms.

- Two lines m and n are said to *intersect* at a point x if x is on both m and n.
- Two lines m and n are *parallel* if either $m = n$ (so they are not really *two*) or if m and n do not intersect. This is expressed as $m \parallel n$.
- Three points x, y, and z are *collinear* if there is a line n which is on all three points.

20.1 Exercise

Express each of the definitions above in predicate logic.

20.2 Properties of Geometries

Using our predicates and defined terms, we can make statements which a geometry may or may not satisfy:

1. Any two points are on exactly one line.
2. Each line is on at least two points.
3. Any two lines intersect in at most one point.
4. No line is on every point.
5. There is a line.
6. There is a point.
7. Every point is on a line.
8. Given any point, there exist two other points so that the three are not collinear.
9. No point is on every line.
10. If p is a point which is not on a line l, then there is a line through p parallel to l.
11. There is a line which is on exactly two points.
12. Any two lines intersect.
13. No two lines intersect.

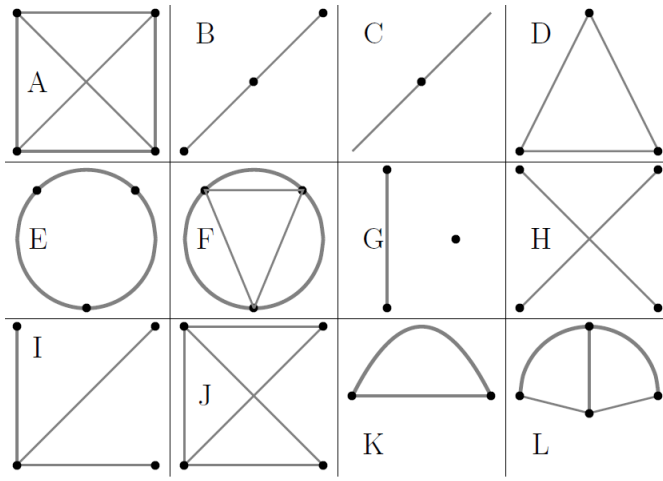


Figure 20.1: Several finite geometries.

20.3 Exercises

1. Convert each of the statements above to predicate logic.
2. For each property in the list above, list the geometries from Figure 20.1 which satisfy that property.

20.4 Linear Spaces

Any geometry which satisfies these three axioms is a **linear space**[1].

L1 Any two distinct points are on at least one line.

L2 No two distinct points are on two distinct lines.

L3 Each line is on at least two points.

20.5 Exercises

1. Convert L1, L2, and L3 to predicate logic.
2. Draw all linear spaces with exactly two points.
3. Draw all linear spaces with exactly three points.
4. Draw all linear spaces with exactly four points.
5. Are there any linear spaces with fewer than two points?
6. Prove this: **Theorem:** Any two lines intersect in at most one point.
7. Prove this: **Theorem:** If l_1 and l_2 are lines and if every point on l_1 is also on l_2, then $l_1 = l_2$.

20.6 Linear-Plus

We will call a geometry **linear-plus** if it satisfies the following axioms. Note that these are the axioms of a linear space along with two additional axioms.

LP1 Any two distinct points are on at least one line.

LP2 No two distinct points are on two distinct lines.

LP3 Each line is on at least two points.

LP4 No line is on every point.

LP5 There is a line.

[1] Don't let the word "space" here bother you. Mathematicians often use the word space to describe some organized system. We could just as well have called these "linear geometries."

20.7 Exercises

1. Convert LP4 and LP5 to predicate logic.
2. Draw all linear-plus geometries with exactly four points.
3. Prove this: **Theorem:** There is a point.
4. Prove this: **Theorem:** There are at least two points.
5. Prove this: **Theorem:** There are at least three points.
6. Must a linear-plus geometry have at least four points? If so, prove it. If not, give a smaller example.
7. Prove this: **Theorem:** Every point is on at least one line.
8. Prove this: **Theorem:** Every point is on at least two lines.
9. Prove this: **Theorem:** There are at least three lines.
10. Prove this: **Theorem:** Given any point, there exist two other points so that the three are not collinear.
11. Prove this: **Theorem:** No point is on every line.

Chapter 21

Basic Proof Techniques

21.1 The Axiomatic Method

Mathematics begins with ideas called **primitives**. These are concepts which are undefined, but which everyone is assumed to understand. For example, the idea of a set is a primitive. From primitives, mathematicians make formal **definitions**. These definitions are usually made to make common concepts rigorous enough to be useful. Mathematicians must usually assume some knowledge about the primitives and definitions with which they work. Statements which are assumed to be true without proof are called **axioms**, **postulates**, or **premises**. From the definitions and axioms, mathematicians make conjectures which they attempt to prove to be true. These facts which they prove take on titles such as theorems, propositions, lemmas, and corollaries. In mathematical writing, **theorem** or **proposition** is usually the generic title given to a proven fact. A **lemma** is generally a fact which is proven as a stepping stone to prove a theorem. A **corollary** is a fact which usually follows quickly from a previous lemma or theorem.

Most mathematicians rarely write formal proofs like those we wrote in Chapter 8. Instead, they write arguments which are convincing to other mathematicians. The steps in these arguments are usually combinations of rules of inference and applications of facts known already to be true. From here on, when we say "proof," we will mean these convincing arguments.

There are three basic reasons that mathematicians write proofs.

1. **To be certain:** Sometimes, a concept which appears to be true on the surface actually is not. We are not always aware of this until we attempt to rigorously show it is true and discover the error.

2. **To know why:** At times, there are ideas whose truth we are aware of, but which we do not truly understand. The process of discovering a proof can lead us to a deeper understanding of why these are true.

3. **To communicate:** Mathematics relies on communication between mathematicians to develop. Ideas propogate most quickly when they are pondered by different minds with varying experiences and intuitions.

The proof techniques in Chapter 10 provide several common basic proof strategies. We give examples of these strategies here in proofs using words rather than in formal proofs using rules of inference. The strategies given here are general and will be fundamental to the rest of the text.

21.2 Working Environment

For the proofs in this section, we will be working in the **natural numbers**. This is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$[1]. The natural numbers have two operations, addition and multiplication which we assume satisfy the properties below. We will have an opportunity later to discuss how $\mathbb{N}$ can be defined from a small set of axioms from which these properties can be derived.

For all $x, y, z, w \in \mathbb{N}$, the following are true

Properties of Addition:

$x + (y + z) = (x + y) + z$	associative law
$x + y = y + x$	commutative law
$y + x = z + x$ if and only if $y = z$	cancellation law
$x + 0 = 0 + x = x$	additive identity
$x \cdot 0 = 0 \cdot x = 0$	absorption law

Properties of Multiplication:

$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	associative law
$x \cdot y = y \cdot x$	commutative law
$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	distributive law
$x \cdot 1 = 1 \cdot x = x$	multiplicative identity
if $x \neq 0$ then $y \cdot x = z \cdot x$ iff $y = z$	cancellation law

These are the "usual" properties of arithmetic which allow us to manipulate equations and expressions involving natural numbers.

[1]Historically, the definition of the natural numbers did not include 0. Many, perhaps most, modern mathematicians include 0 in $\mathbb{N}$. Doing so pains me greatly. Most societies took centuries to realize the need for 0 as a number, so this number does not seem remotely natural. However, including 0 here will make the lives of my students easier, and it provides for elegant parallels between arithmetic in $\mathbb{N}$ and set theory. So, after decades of resistance, I am finally conceding and allowing 0 to be a natural number.

To make notation simpler, we will usually use juxtaposition to symbolize multiplication. Also, we will use an order of operations which requires us to perform multiplication prior to addition. These are typical conditions with which we should be familiar.

It is very important at this point in time to remember that we cannot subtract or divide. Subtraction may result in negative numbers (which are not in $\mathbb{N}$), and division may result in fractions (which may not be in $\mathbb{N}$). Our replacement for subtraction and division are the cancellation rules.

Order: We will say that a natural number n is **less than or equal** to a natural number m if $n + k = m$ for some natural number k. This is denoted by $n \leq m$. We will say that n is **less than** m if $n \leq m$ but $n \neq m$. This is denoted by $n < m$.

If $n \leq m$, we may also say that m is **greater than or equal** to n and write $m \geq n$. Similarly, we can use $m > n$ to mean that m is **greater than** n.

Divisibility: We will say that a natural number n **divides** a natural number m if there is a natural number k so that $nk = m$. This relationship will be denoted by $n|m$. If $n|m$, then we may also say that m is a **multiple** of n or that n is a **factor** of m.

Note the similarity between the definitions of less than and divides. The operations addition and multiplication have the associative law, commutative law, and cancellation law in common. They are quite similar. Thus it stands to reason that these two relations might share some common properties.

21.3 Direct Proof

Most theorems in mathematics are statements of the form $P \to Q$. P is called the **hypothesis** of the theorem, and Q is the **conclusion**. One method of proving theorems such as these is to assume that P is true and use this assumption to show that Q must be true.

Theorem 21.1: If l, m, and n are natural numbers so that $l|m$ and $m|n$, then $l|n$.

Discussion: It is a good idea to do scratch work before attempting a proof. We are going to assume that l, m, and n are natural numbers, that $l|m$, and that $m|n$. We need to show that $l|n$. The first step is to consider the definition of divides and see what we know and what we need. The definition of divides tells us that there are natural numbers a and b so that $la = m$ and $mb = n$. What we need is a natural number k so that $lk = n$. By substituting la for m in $mb = n$, we see that $lab = n$, so $k = ab$ is a logical choice.

Comments: When writing a proof, there are a lot of things to consider. First, the proof should be written in complete sentences with proper English grammar. Second, one of the main purposes of the proof is to communicate,

so be sure to explain to the reader what is happening. Use transitions to indicate progress through the proof. Do not assume the reader knows what you are thinking. Make sure that you define all of the appropriate variables for the reader.

Proof: Suppose that l, m, and n are natural numbers so that $l|m$ and $m|n$. We will show that $l|n$. By the definition of divides, there are natural numbers a and b so that $la = m$ and $mb = n$. Let $k = ab$. It follows that $lk = lab = mb = n$. By the definition of divides, we see that $l|n$ □

In most mathematical writing, the end of a proof is usually marked by a symbol such as the empty box above. This allows the reader to easily see where proofs end.

Not all implications are stated as clearly as that in the previous example. For example:

Theorem 21.2: For any natural numbers l, m, and n, if $m|n$ then $ml|nl$.

This theorem could be restated as, "If l, m, and n are natural numbers so that $m|n$ then $ml|nl$." Sometimes, the implication in a theorem is completely hidden. For example:

Theorem 21.3: Every natural number is divisible by 1.

This theorem could be written, "If n is a natural number, then $1|n$. (This is trivially true since $1 \cdot n = n$ by definition.)

21.4 Exercises

Prove each of the following using direct proof:

1. **Theorem:** If l, m, and n are natural numbers so that $l \leq m$ and $m \leq n$, then $l \leq n$.
2. **Theorem:** For any natural numbers l, m, and n, if $m|n$ then $ml|nl$.
3. **Theorem:** For any natural numbers l, m, and n, if $m \leq n$, then $m + l \leq n + l$.
4. **Theorem:** For any natural numbers l, m, and n, if $m \leq n$, then $ml \leq nl$.
5. **Theorem:** If a, b, c, and d are natural numbers, $a \leq b$, and $c \leq d$, then $a + c \leq b + d$.
6. **Theorem:** If a, b, c, and d are natural numbers, $a \leq b$, and $c \leq d$, then $ac \leq bd$.

21.5 Even and Odd Numbers

A natural number n is **even** if $2|n$. A natural number n is **odd**[2] if there is a natural number k with $n = 2k + 1$.

We will employ the following theorem (which we will not prove here) repeatedly in our examples.

Theorem 21.4: (Even-Odd Dichotomy) Every natural number is either odd or even but not both.

Theorem 21.5: The sum of two even natural numbers is even.

Observation: Again, this is not written as an implication. We can change the theorem to an implication by writing it as "If n and m are even natural numbers, then $n + m$ is also even."

Discussion: We will assume that n and m are even integers. We must look at what we know and what we need. We know that $2|n$ and $2|m$. This means that there are natural numbers a and b so that $2a = n$ and $2b = m$. We need to show that $2|(n + m)$, so we take a look at this sum. What we know tells us that $n + m = 2a + 2b$. The distributive law tells us that $n + m = 2(a + b)$. Since $(a + b)$ is a natural number, this means $2|(n + m)$. We are ready for a proof.

Proof: Let n and m be even natural numbers. We will show that $n + m$ is even. By the definition of even, $2|n$ and $2|m$. By the definition of divides, there are natural numbers a and b so that $2a = n$ and $2b = m$. Note now that $2(a + b) = n + m$. Hence $2|(n + m)$. By the definition of even, $n + m$ is even. □

Note: In this proof, we could have let $k = a + b$ so that the final equality looked more like the definition of divisibility ($2k = n + m$). This is not necessary, but it may have helped make a complicated proof more readable.

21.6 Exercises

Prove the following.

1. **Theorem:** If n and m are odd natural numbers, then $n + m$ is even.

[2]This is one place where including 0 in $\mathbb{N}$ is convenient. If we did not include 0, we would have to define a natural number n to be odd if either $n = 1$ or there is a natural number k with $n = 2k + 1$. Including 0 simplifies almost all theorems and proofs which refer to odd natural numbers.

2. **Theorem:** If n is an even natural number, then $n \cdot n$ is an even natural number. (Hint: write $n = 2k$ and square)

3. **Theorem:** Suppose that n and m are natural numbers. If n is even, then nm is even.

21.7 If-and-only-if

Many theorems contain the words "if and only if." It is common to abbreviate "if and only if" as "iff." Recall that the statement "P if and only if Q" (called a bi-implication) is the same as "If P then Q, and if Q then P." Thus in order to prove a bi-implication, we can simply prove the two implications.

Theorem 21.6: A natural number n is even if and only if $n + 1$ is odd.

Discussion: There are two things to prove here. First, if n is even, then $n + 1$ is odd. Second, we need to prove that if $n + 1$ is odd, then n is even.

Proof: Suppose that n is a natural number. We will prove that n is even if and only if $n + 1$ is odd. First suppose that n is even. Then there exists a natural number k so that $n = 2k$. Then $n + 1 = 2k + 1$. By the definition of odd, $n + 1$ is odd. Hence, if n is even, then $n + 1$ is odd.

Now suppose that $n+1$ is odd. By the definition of odd, there is a natural number k so that $n+1 = 2k+1$. Cancelation now gives $n = 2k$, so $2|n$. Thus n is even. Hence, if $n + 1$ is odd, then n is even.

We have proven that a natural number n is even if and only if $n + 1$ is odd. □

21.8 Exercises

Prove the following theorems (note the "iff").

1. **Theorem:** Let m, n, and l be natural numbers. Then $m \leq n$ if and only if $m + l \leq n + l$.

2. **Theorem:** Let m and n be natural numbers. Then $m < n$ if and only if there is a natural number $k \neq 0$ so that $m + k = n$.

3. **Theorem:** A natural number n is odd if and only if $n + 1$ is even.

21.9 Proofs With Cases

Recall that the statement $(P \vee Q) \to R$ is equivalent to the statement $(P \to R) \wedge (Q \to R)$. Thus, to prove $(P \vee Q) \to R$, we could prove $P \to R$ and $Q \to R$. When we do so, we are using **cases**. Consider the following theorem

Theorem 21.7: If n is a natural number then $n^2 + 3n$ is even.

Discussion: There are two cases to consider here – either n is even or n is odd. In both cases, we will invoke the definition to express n either as $2k$ or as $2k + 1$ for some k. We will then substitute and follow our noses.

Proof: Suppose that n is a natural number. We will prove that $n^2 + 3n$ is even. There are two cases – either n is even or n is odd. Suppose first that n is even. Then there is a natural number k so that $n = 2k$. It follows that

$$\begin{aligned} n^2 + 3n &= (2k)^2 + 3(2k) \\ &= 2(2k^2 + 3k). \end{aligned}$$

If we let $l = 2k^2 + 3k$, then $n^2 + 3n = 2l$ is even. Thus if n is even, then $n^2 + 3n$ is also even.

Next suppose that n is odd. Then there is a natural number k so that $n = 2k + 1$. It follows that

$$\begin{aligned} n^2 + 3n &= (2k + 1)^2 + 3(2k + 1) \\ &= 4k^2 + 4k + 1 + 6k + 3 \\ &= 4k^2 + 10k + 4 \\ &= 2(2k^2 + 5k + 2) \end{aligned}$$

If we let $m = 2k^2 + 5k + 2$, then $n^2 + 3n = 2m$ is even. Thus if n is odd then $n^2 + 3n$ is even.

We have proven that if n is either even or odd, then $n^2 + 3n$ is even. Since every natural number is either even or odd, $n^2 + 3n$ is even for all natural numbers n. □

21.10 Exercises

Define the following notions for a natural number n:

- n is **red** if there is a natural number k with $n = 3k$.
- n is **white** if there is a natural number k with $n = 3k + 1$.
- n is **blue** if there is a natural number k with $n = 3k + 2$.

Assume the following theorem.

Theorem 21.8: Any natural number is exactly one of red, white, or blue.

Prove the following theorems.

1. **Theorem:** The square of any natural number is either red or white.
2. **Theorem:** If n is any natural number, then $n^2 + n$ is even.

21.11 Contrapositive

We know that a statement of the form $P \to Q$ is equivalent to its contrapositive $\neg Q \to \neg P$. Often it is easier to prove the contrapositive of a theorem rather than directly proving the theorem.

Theorem 21.9: Let n be a natural number. If n^2 is even, then n is even.

Discussion: If we were to assume that n^2 is even, this would give us very little information about n. However, if we use the contrapositive, the theorem is quite easy to prove.

Proof: Let n be a natural number. We will use the contrapositive to prove that if n^2 is even then n is even. The contrapositive is "If n is not even, then n^2 is not even." Suppose then that n is not even. Since every natural number is either even or odd, n must be odd. This means that that there is a natural number k so that $n = 2k+1$. It follows that $n^2 = 4k^2+4k+1 = 2(2k^2+2k)+1$ is odd. By the theorem in 21.4, since n^2 is odd, it is not even. We have proven that if n is not even, then n^2 is not even. This is the contrapositive of the theorem. □

21.12 Exercises

Prove the following theorems using the contrapositive.

1. **Theorem:** For any natural number n if n^2 is odd, then n is odd.
2. **Theorem:** For any natural numbers a and b, if $2|ab$, then either $2|a$ or $2|b$.
3. **Theorem:** For any natural number n, if $3n + 1$ is odd, then n is odd.

21.13 Contradiction

Suppose we want to prove a statement P is true. Suppose also that if we assume P to be false, then we can prove that a contradiction C would have to be true (recall that a contradiction is always false). This means that we can prove $\neg P \to C$ where C is false. It must follow that P is true. The statement $\neg P \to C$ is equivalent to its contrapositive $\neg C \to P$. Since C is false, $\neg C$ is true, so the implication $\neg C \to P$ would force P to be true by Modus Ponens.

This is the basis of proof by contradiction. To prove P by way of contradiction, assume $\neg P$ and try to prove a contradiction.

Theorem 21.10: If n is an natural number and n^2 is even, then n is even.

Discussion: We proved this above using the contrapositive. That is a better option, but we prove the same theorem by contradiction as an example. Many times, contradiction proofs can be rewritten as contrapositive proofs.

Proof: Suppose that n is a natural number and that n^2 is even. We will use contradiction to prove that n is even. Suppose by way of contradiction that n is not even. Then n is odd and there is a natural number k so that $n = 2k + 1$. It follows that

$$n^2 = (2k + 1)^2 = 2(2k^2 + 2k) + 1$$

so n^2 is odd. But then n^2 is both odd and even. This contradicts Theorem 21.4, so the assumption that n is not even must be false. It has to be the case that n is even. □

21.14 Exercise

Use contradiction to prove these theorems

1. **Theorem:** If n is a natural number so that n^2 is red, then n is red. (Use the definitions and theorem from 21.10
2. **Theorem:** If n is a natural number so that $5n + 1$ is even, then n is odd.

Chapter 22

The Natural Numbers

22.1 In Search of Truth

In the seventeenth century, Leibniz imagined a machine which could mechanically distinguish between statements which are true and statements which are false. The crux of this machine would be a formal language in which statements could be manipulated algebraically and truth could be calculated. We have now developed enough of sentential and predicate logic that it is almost time to return to this quest. The path we would like to pursue is to list all statements in a way that they can be naturally (or mechanically) associated with natural numbers: $\alpha_0, \alpha_1, \alpha_2, \ldots$. Then consider the set of those natural numbers n for which α_n is a true statement. This is a set of natural numbers. In the previous chapter, we considered sets of natural numbers which are definable using predicate logic and some operations and inequalities. Might it be that the set of natural numbers which correspond to true statements is definable? If so, then the first order description of this set would provide the mechanical means of testing a statement for truth that Leibniz dreamed of. Truth would be definable in a rigid mathematical sense. Of course, considering all statements in any spoken language seems to be a bit much to ask, so we will actually consider statements in the language of the natural numbers and about the natural numbers. We begin with a discussion of the natural numbers and a pursuit of a set of axioms for the natural numbers.

22.2 Symbols for Operations and Constants

So far, first order logic for us has involved only predicates. Predicates can be used to describe operations and constants. For example, in the natural numbers, we could consider the relation

$$SUM = \{(x, y, z) : x + y = z\}.$$

Then, $SUM(x, y, z)$ would mean $x + y = z$, so the ternary relation SUM defines the operation of addition. Also, we can write a sentence declaring that a ternary relation $P(x, y, z)$ defines a binary operation – this would declare that for every x and y there is exactly one z so that $P(x, y, z)$. Of course, using relation notation for operations seems cumbersome to those of us who were trained to use infix notation for operations – it is easier for us to think in terms of $x + y = z$ rather than $SUM(x, y, z)$. For this reason, we will assume from now on that our first order languages may have symbols for operations along with relations.

Another tool we are used to using from elementary math is the notion of of a constant – a fixed number or element which we may refer to by name. In the natural numbers, we can use the relation $Z = \{0\}$ to identify the number zero. For any x, $Z(x)$ means that $x = 0$. Again, it is simpler just to assume that we have names for constants when we need them rather than identifying them with relations – it is easier to refer to 0 than to refer to some x for which $Z(x)$ holds. Therefore, we will also assume from now on that our first order languages may have constant symbols.

22.3 Peano Arithmetic

There is a natural operation on the natural numbers called the *successor* operation. It is defined by $s(x) = x+1$ (the successor of x is one more than x). It was Hermann Günther Grassmann (1809-1877) who first called attention to the fact that addition and multiplication and many of their properties on the natural numbers could be derived from this successor operation. When mathematicians were formalizing logic and mathematics in the late nineteenth century, Peano took these observations about the successor operation and created an axiom system for the natural numbers. The Peano Axioms for the natural numbers include the logical axioms of equality along with those axioms listed below.

The first two axioms give us "a place to begin" and a way of constructing natural numbers. These two axioms are not really statements in predicate logic. They merely give us a symbol 0 and a unary operation s in the language of the natural numbers.

P1. There is a natural number called 0[1].

P2. For each natural number n, there is a unique natural number $s(n)$ call the *successor* of n.

The next axiom insures that the natural numbers "begin" at 0.

P3. The natural number 0 is not a successor. For any natural number n, $s(n) \neq 0$.

[1]Peano's system actually began with 1. Many modern derivations begin with 0.

So far, the axioms imply that the natural numbers contain as a subset

$$\{0, s(0), s(s(0)), \ldots\}.$$

We wish for these numbers to be all distinct. This is guaranteed by the next axiom.

P4. Different natural numbers have different successors. If $n \neq m$, then $s(n) \neq s(m)$.

This axiom insures that the natural numbers do not "wrap around" on themselves. At this point, the natural numbers contain at least an infinite chain such as

$$0 \quad \to s(0) \quad \to \quad s(s(0)) \quad \to \quad s(s(s(0))) \quad \to \quad \cdots$$

The axioms so far do not force the natural numbers not to contain in addition to this chain other similar infinite chains or even some other "doubly infinite" chains such as

$$\cdots \quad \to \quad a \quad \to s(a) \quad \to \quad s(s(a)) \quad \to \quad \cdots$$

The fifth Peano axiom prevents these additional chains from existing.

P5. For any set K of natural numbers if

- K contains 0 and
- For any natural number n, if $n \in K$ then $s(n) \in K$

then K is the entire set of natural numbers.

This last axiom insures that any natural number can be constructed by applying the successor operation repeatedly to 0.

There is one issue with the final Peano axiom. This axiom *quantifies over sets* of natural numbers. It is a statement in *second order logic*.

22.4 Why Successors?

Why would we want axioms for the natural numbers expressed in terms of the successor operation instead of the more familiar arithmetic operations such as addition or multiplication? The successor operation can be imitated in many environments. These environments then can be considered logical expansions of the natural numbers. Some theorems proven for the natural numbers will then extend naturally to these other environments.

As an example of an environment where we can find a successor operation we turn to set theory. In set theory, we will interpret 0 to be $\emptyset$ and we will define the successor operation to be

$$s(A) = A \cup \{A\}.$$

Thus, for example, $S(\{a,b,c\}) = \{a,b,c,\{a,b,c\}\}$. (Notice that we have the odd situation where both $A \in S(A)$ and $A \subseteq S(A)$.). This operation s and constant 0 satisfy the first four Peano Axioms. Within set theory, then, we can build a copy of the natural numbers:

$$0 \text{ is } \emptyset$$

$$1 \text{ is } 0 \cup 0 = \emptyset \cup \{\emptyset\} = \{\emptyset\}$$

$$2 \text{ is } 1 \cup \{1\} = \{\emptyset\} \cup \{\{\emptyset\}\} = \{\emptyset, \{\emptyset\}\}$$

$$3 \text{ is } 2 \cup \{2\} = \{\emptyset, \{\emptyset\}\} \cup \{\{\emptyset, \{\emptyset\}\}\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$$

These are more succinctly written in this way

$$0 = \emptyset$$

$$1 = \{0\}$$

$$2 = \{0,1\}$$

$$3 = \{0,1,2\}$$

$$\vdots$$

$$n = \{0,1,\ldots,(n-1)\}$$

Notice that with these identifications, $n \leq m$ means the same thing as $n \subseteq m$.

Having an identifiable copy of the natural numbers in set theory along with symbols for the language of the natural numbers (so far, 0 and s) will allow us later to extend some results about the natural numbers to all of set theory.

22.5 Arithmetic Operations

Grassmann noticed that the usual arithmetic operations on the natural numbers could be defined in terms of the successor operation. The beginning observation is that if we know how to add (or multiply) 0, and if we know how addition interacts with the successor operation, then we know how to add 1, and then 2, and so on – we then know how to add any natural number.

First, if m is any natural number, $m+0=0$. Suppose that we know how to calculate $m+n$ for some natural number n. How do we calculate $m+s(n)$? Here we resort to what we were told in grade school. We are associating $s(n)$ with $n+1$. We know that $m+(n+1)$ *should be* $(m+n)+1$ which is $s(m+n)$. Thus we can *define* $m+s(n)=s(m+n)$. Thus

$$m+0=m$$

$$m+1=m+s(0)=s(m+0)=s(m)$$

$$m+2=m+s(1)=s(m+1)=s(s(m))$$

and so on. We can add to our axioms two axioms which uniquely determine addition in the natural numbers:

A1. $\forall x(x + 0 = x)$

A2. $\forall x \forall y(x + s(y) = s(x + y))$

We can construct similar axioms for multiplication and exponentiation. In both cases, the first axiom says how to operate with 0. The second axiom says how to operate with the successor of a natural number.

M1. $\forall x(x \cdot 0 = 0)$

M2. $\forall x \forall y(x \cdot s(y)) = (x \cdot y) + x$

E1. $\forall x(x^0 = s(0))$

E2. $\forall x \forall y(x^{s(y)} = x^y \cdot x)$

Finally, we can also define the relations $\leq$ and $<$ on the natural numbers in terms of the successor operation. Since $+$ has been defined with the successor operation, it is more convenient to use $+$. For any natural numbers x and y, $x < z$ if and only if z is some positive number larger than x. If $x = z$, then $x + 0 = z$. Thus we have

L1. $\forall x \forall z[(x \leq z) \leftrightarrow (\exists y(x + y = z))]$

Then we also have

L2. $\forall x \forall y[(x < y) \leftrightarrow (x \leq y \wedge x \neq y)]$

L3. $\forall x \forall y[(x < y) \vee (x = y) \vee (y < x)]$

L4. $\forall x \forall y[(x < s(y)) \leftrightarrow (x \leq y))]$

We now have our language and axioms for number theory. The language consists of a constant symbol 0; operation symbols s, $+$, $\cdot$, and exponentiation; and two relation symbols $<$ and $\leq$. Our axioms are P1-5, A1-2, M1-2, E1-2, and L1-4. It can be proven that any model of these axioms in this language is exactly the natural numbers. From these axioms we can derive the usual properties of arithmetic – associativity, commutativity, distributivity, etc. However, these proofs would make heavy use of P5 *which is not a first order statement.*

Here is an example of how to use the definitions to do arithmetic:

Example 22.1: Calculate $1 \cdot 2 = 2$ using the recursive definitions of multiplication and addition.

$$\begin{aligned} s(0) \cdot s(s(0)) &= [s(0) \cdot s(0)] + s(0) \\ &= [[s(0) \cdot 0] + s(0)] + s(0) \\ &= s(0) + s(0) \\ &= s(s(0) + 0) \\ &= s(s(0)) \end{aligned}$$

22.6 Exercises

Use the definitions of addition, multiplication, and exponentiation to calculate

1. $s(0) + s(s(0))$
2. $s(0) \cdot s(0)$
3. $s(0) \cdot (s(0) + s(0))$
4. $s(s(0))^{s(0)}$

22.7 First Order Axioms for $\mathbb{N}$

This question remains to be addressed: Can we construct first order axioms for the natural numbers which are complete in the sense that all of the first order sentences which are true in the natural numbers can be proven from these axioms?

Any first order model of P1-P4, A1-2, M1-2, E1-2, and L1-4 will contain a copy of the natural numbers – the smallest subset containing 0 and closed under s. This copy will satisfy all of the "nice" properties of the natural numbers. The rest of the model will satisfy some of these properties.

For the rest of these notes, let Π be the set of sentences P1-P4, A1-2, M1-2, E1-2, and L1-4.

22.8 Exercises

1. Use the language of the natural numbers to write a predicate that declares "x is multiple of y."
2. Use the language of the natural numbers to write a predicate $P(x)$ so that $P(x)$ is true if and only if x is even.
3. Use the language of the natural numbers and a predicate $P(x)$ to write a pair of wffs to recursively define $P(x)$ to mean that x is even.
4. Use the language of the natural numbers to write a predicate $P(x)$ which is true if x is prime.
5. Use the successor operation in set theory to calculate $s(\{a, b\})$.

22.9 Axiom P5

The oddest of the Peano Axioms is probably the fifth. This axiom is a tool for showing that certain facts are true for all natural numbers. It is the basis of mathematical induction. We give an example here of a pure application of this axiom, and then describe the general notion of induction later.

Theorem 22.2: For any natural numbers x, y, and z, the equality $(x + y) + z = x + (y + z)$ holds.

Proof: Let $x, y \in \mathbb{N}$ and let S be the set of all $z \in \mathbb{N}$ for which $x+(y+z) = (x + y) + z$. We will use Axiom 5 to show that $S = \mathbb{N}$. First, we must show that $0 \in S$. Notice that by the definition of $+$ we have[2]

$$\begin{aligned} x + (y + 0) &= x + y \\ &= (x + y) + 0 \end{aligned}$$

so $0 \in S$. Next, assume that $z \in S$. This means that $x + (y + z) = (x + y) + z$. We will show that $s(z) \in S$. That is, we must show that $x + (y + s(z)) = (x + y) + s(z)$. Again, we need only apply the definition of $+$ several times.

$$\begin{aligned} x + (y + s(z)) &= x + s(y + z) \\ &= s(x + (y + z)) \\ &= s((x + y) + z) \\ &= (x + y) + s(z) \end{aligned}$$

Thus if $z \in S$, then also $s(z) \in S$. By Axiom 5, we can conclude that $S = \mathbb{N}$. It follows that for all x, y, and z in $\mathbb{N}$, $(x + y) + z = x + (y + z)$. □

Here is another pure application of Axiom 5.

Theorem 22.3: Every natural number either equals 0 or is the successor of a natural number.

Proof: Let S be the set of all natural numbers which are either successors or which are equal to 0. Note that 0 is in S by definition. Thus S satisfies the first requirement of Axiom 5. Next, we must show that if $k \in S$, then $s(k) \in S$. Suppose that $k \in S$. The number $s(k)$ is the successor of k. By the definition of S, $s(k) \in S$ (since $s(k)$ is a successor). Hence, if $k \in S$, then $s(k) \in S$. We have satisfied the second condition of Axiom 5. By the fifth Peano Axiom, $S = \mathbb{N}$. It follows that every natural number is either a successor or is equal to 0. □

22.10 Principle of Mathematical Induction

Axiom 5 is the basis for mathematical induction:

[2]This is an instance where including 0 in $\mathbb{N}$ makes life easier. The arithmetic in the first step of proofs using Axiom 5 to establish properties of addition and multiplication are often simpler with 0 than with 1.

Suppose $P(n)$ is a predicate about some natural number n. Let m be a natural number. If these two statements are true

- $P(m)$ is true and
- For any $k \geq m$ in $\mathbb{N}$, if $P(k)$ is true, then $P(k+1)$ is true

then $P(n)$ is true for all n greater than or equal to m.

Theorem 22.4: For any natural number n, this equality holds

$$4 \cdot (0^3 + 1^3 + \cdots + n^3) = n^2(n+1)^2$$

Proof: Let $P(n)$ be the predicate "$4 \cdot (0^3 + 1^3 + \cdots + n^3) = n^2(n+1)^2$." We will use induction to prove that $P(n)$ is true for all natural numbers n. First, note that $4 \cdot (0^3) = 0$ and $0^2 \cdot (0+1)^2 = 0$, so that $P(0)$ is true. Next, suppose that k is a natural number and that $P(k)$ is true. That is, we are assuming that

$$4 \cdot (0^3 + 1^3 + \cdots + k^3) = k^2(k+1)^2$$

Observe that

$$\begin{aligned} 4 \cdot (0^3 + 1^3 + \cdots + k^3 + (k+1)^3) &= 4 \cdot (0^3 + 1^3 + \cdots + k^3) + 4(k+1)^3 \\ &= k^2(k+1)^2 + 4(k+1)^3 \\ &= (k+1)^2(k^2 + 4(k+1)) \\ &= (k+1)^2(k^2 + 4k + 4) \\ &= (k+1)^2(k+2)^2 \\ &= (k+1)^2([k+1]+1)^2 \end{aligned}$$

Hence, $P(k+1)$ is true. We have established that $P(0)$ is true and that $P(k)$ implies $P(k+1)$. By mathematical induction, we can conclude that $P(n)$ is true for all natural numbers n. □

Here is another example of proof by induction.

Theorem 22.5: For any natural number n, $4^{2n+1} + 1$ is divisible by 5.

Proof: For any natural number n, let $P(n)$ be the open statement "$4^{2n+1} + 1$ is divisible by 5." We will use mathematical induction to show that $P(n)$ is true for all natural numbers n. First, note that $4^{2 \cdot 0 + 1} + 1 = 5$ is divisible by 5, so $P(0)$ is true. Next, suppose $P(k)$ is true for some natural number k. That is, we are assuming $4^{2k+1} + 1$ is divisible by 5. This means

there is a natural number l so that $4^{2k+1} + 1 = 5l$. Observe

$$\begin{aligned} 4^{2(k+1)+1} + 1 &= 4^{2k+3} + 1 \\ &= 4^{2k+1}4^2 + 1 \\ &= 4^{2k+1} \cdot 16 + 1 \\ &= 4^{2k+1}(3 \cdot 5 + 1) + 1 \\ &= 4^{2k+1} \cdot 3 \cdot 5 + 4^{2k+1} + 1 \\ &= 4^{2k+1} \cdot 3 \cdot 5 + 5l \\ &= 5(4^{2k+1} \cdot 3 + l) \end{aligned}$$

From the definition of divisibility, we see that 5 divides $4^{2(k+1)+1} + 1$, so $P(k+1)$ is true. Thus, if $P(k)$ is true, so is $P(k+1)$. We have established that $P(0)$ is true and that $P(k)$ implies $P(k+1)$. By mathematical induction, we can conclude that $P(n)$ is true for all natural numbers n. □

22.11 Exercises

Use induction to prove the following.

1. For all natural numbers n, $2(0 + 1 + \cdots + n) = n(n + 1)$.
2. For all natural numbers n, $6(0^2 + 1^2 + \cdots + n^2) = n(n + 1)(2n + 1)$.
3. $3^{2n+1} + 1$ is divisible by 4 for all natural numbers n.
4. For all natural numbers $n \geq 1$,

$$1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n} = 2 - \frac{1}{2^n}.$$

5. Guess a formula for $1 + 3 + \cdots + (2n + 1)$ and use induction to prove that your guess is correct.

Chapter 23

Incompleteness

In this chapter, we address the question of whether or not there is a nice set of axioms which can prove all of the sentences which are true in $\mathbb{N}$.

23.1 Theories

A *theory* in a first order language L is a set of sentences Σ in L which is closed under deduction in the sense that if $\Sigma \vdash \alpha$, then $\alpha \in \Sigma$.

We will have two main sources for theories. If Σ is a set of first order sentences, then the set of *consequences* of Σ is the set

$$\mathrm{Cn}(\Sigma) = \{\alpha : \Sigma \vdash \alpha\}.$$

This is the set of all sentences provable from Σ. This set is (almost) by definition closed under deduction. If $\mathbf{A}$ is any first order structure, then the **theory of A** is the set

$$\mathrm{Th}(\mathbf{A}) = \{\alpha : A \models \alpha\}.$$

This is the set of all sentences which $\mathbf{A}$ satisfies.

Suppose that Σ is a subset of a theory T. Σ *axiomatizes* T if $\mathrm{Cn}(\Sigma) = T$. Our goal is to address this question:

Can $\mathrm{Th}(\mathbb{N})$ be axiomatized by a "nice" set of sentences.

Here, "nice" could mean finite, definable, recursively enumerable, or recursive (computable, decidable). To keep things simple, our discussion in this chapter will focus almost completely on definable.

A theory Σ is *complete* if for any sentence α in the language of Σ, either α or $\neg\alpha$ is in Σ. A set Σ of wffs is complete if $\mathrm{Th}(\Sigma)$ is complete.

Caution: Do not confuse this notion of complete with that discussed in the first two parts of these notes. The completeness theorems in the earlier chapters were about our *deductive system*. These systems were complete in the sense that any logical implication can be proven. Completeness here is a measure of the extent or size of a theory.

Not all theories are complete. Let Σ be the sentences which define an ordered set and let $T = \mathrm{Cn}(\Sigma)$. Let α be the sentence that declares that a model has exactly two elements. Since there is a two element model of T and a three element model of T, it cannot be that T contains α or $\neg\alpha$. Just because $\mathbf{A}$ is a model of a set Σ of sentences, we cannot conclude that the only sentences true in $\mathbf{A}$ follow logically from Σ.

Recall that Π is the set of axioms P1-4, A1-2, M1-2, E1-2, and L1-2. We can now ask these questions.

- Is $\mathrm{Cn}(\Pi)$ complete?
- Is it true that $\mathrm{Cn}(\Pi) = \mathrm{Th}(\mathbb{N})$?
- Can $\mathrm{Th}(\mathbb{N})$ be axiomatized by a "nice" set of sentences?
- Is $\mathrm{Th}(\mathbb{N})$ definable?

We will see in this chapter that each of these has a negative answer.

23.2 Examples

Our spoken language has the *expressive power* to contain sentences which make statements about the truth and provability of other sentences, collections of sentences, or even about themselves. The Liar's Paradox of Eubulides is an illustration of some of the confusion that this expressiveness can lead to. It is also an example of how logicians have been aware of the complications of self-reference since the birth of logic. Self-reference was an essential part of the diagonal arguments of the previous chapter.

The languages of formal logic are supposed to be idealized approximations of our spoken language in which thought, deduction, and communication can be made precise. It is reasonable then to consider whether or not we can create models in which statements can be made about sentences, truth, and provability.

Here is an example of a simple way to construct a language with the *expressive power* to make statements about sentences and provability. You should note that the language is simple (only a single predicate). It is the particular model that allows expressibility.

Example 23.1: Let L be the first order language with one unary predicate symbol P. Let F be the set of all wffs over L. We can turn F into an L-structure by defining a unary relation P on F.

Let $P = \mathrm{Cn}(\Delta)$ be the set of all logical consequences of Δ (the axioms of predicate logic). Then in F, $P(x)$ means precisely "x is provable from Δ."

On the other hand, if $P = F - \mathrm{Cn}(\Delta)$, then in F, $P(x)$ means "x is not probable from Δ."

Here is a slightly more complicated example in which we have a formula that essentially says "I am not provable from Δ."

Example 23.2: Let A be the set of all wffs over a first order language with one unary predicate symbol P and a constant symbol 0. Let $0^A = P(0)$. Let $S = \mathrm{Cn}(\Delta)$ be the set of wffs which are provable from Δ, and let $P^A = A - S$.

In the model A, $P(x)$ says "x is not provable from Δ." Then $P^A(0^A)$ would be "0^A is not provable from Δ." But since $0^A = P(0)$, $P^A(0^A)$ is "$P(0)$ is not provable from Δ." – within A, this is "I am not provable from Δ."

In this last example, is $P^A(0^A)$ true. Ponder this for a moment before going on. . ..

Theorem 23.3: Suppose that L is a first order language and that $\mathbf{A}$ is an L-structure. Suppose that Σ is a collection of sentences which are true in $\mathbf{A}$. If there is an L-sentence α which in $\mathbf{A}$ expresses that it is not the case that $\Sigma \vdash \alpha$, then α is true in $\mathbf{A}$ and is not provable from Σ.

The sentence α expresses "α is not provable from Σ." This has to be true in $\mathbf{A}$. If not, then it would not be the case that α is not provable from Σ. This means that α is provable from Σ. Since $\mathbf{A} \models \Sigma$, then also $\mathbf{A} \models \alpha$. But then $\mathbf{A} \models \neg\alpha \wedge \alpha$ which is impossible. By contradiction, $\mathbf{A} \models \alpha$.

23.3 Gödel's Theorems

In 1931 Kurt Gödel proved that the scenario described in Theorem 23.3 can (almost) be constructed in the natural numbers. He describe a manner of associating every wff in the language of the natural numbers with a natural number in a mechanically recognizable way. This encoding should be not

too surprising to the modern reader. Such encodings are implemented daily within computers. Gödel's numbering system was such that concepts such as provability could be recognized within the system. For any "nice" (think definable) subset Σ of wffs, Gödel could construct a sentence σ in the language of the natural numbers so that σ is true if and only if σ is not provable from Σ. As in Theorem 23.3, σ must be true and must not be provable from Σ.

A special case of one of Gödel's Theorems is listed here. The notation $G(\sigma)$ is a natural number associated with the wff σ, and $G(\Sigma)$ is the set of numbers associated with the wffs in the set Σ. (Gödel's proof is outlined in a little more detail in Section 23.5)

Theorem 23.4: Suppose that Σ is a set of sentences true in $\mathbb{N}$ so that $G(\Sigma)$ is definable. There is a sentence σ which is true in $\mathbb{N}$ but which cannot be proven from Σ.

Suppose in the theorem that $\Sigma = \text{Th}(\mathbb{N})$ is the set of all wffs which are true in the natural numbers. Then any σ true in $\mathbb{N}$ must be in Σ already, so σ would be provable from Σ. If $G(\Sigma)$ were definable, then this would contradict the theorem. Therefore:

Theorem 23.5: Tarski Undefinability Theorem Truth is not definable in $\mathbb{N}$. That is, there is no formula τ in the language of $\mathbb{N}$ so that for any sentence σ

$$\mathbb{N} \models [\sigma \leftrightarrow \tau(G(\sigma)].$$

Theorem 23.4 declares that any definable set Σ of sentences true in $\mathbb{N}$ must be incomplete (There is a true sentence which is not provable from Σ). Just like Theorem 23.3 the proof of Theorem 23.4 relies on the fact that the sentences modeled by $\mathbb{N}$ must be consistent. The proof can be adjusted to show:

Theorem 23.6: Any definable set of sentences in the language of $\mathbb{N}$ is either incomplete or inconsistent.

In particular, there is no definable set of sentences Σ which are true in $\mathbb{N}$ (and hence consistent) from which every true sentence in $\mathbb{N}$ can be proven (Σ cannot be complete).

Theorem 23.7: $\text{Th}(\mathbb{N})$ cannot be axiomatized by any definable set of sentences.

It follows that

Theorem 23.8: Cn(Π) is not complete and Cn(Π) ≠ Th(ℕ).

As remarked earlier, we can realize the language of number theory within set theory via the successor operation. By basing the axioms used in the proofs of Gödel's Theorems on the successor operation, we can extend these theorems to *reasonable mathematical systems which are at the same level of complexity as the natural numbers.* The Incompleteness Theorems for number theory can be extended to set theory to establish:

Theorem 23.9: Incompleteness Theorem for Set Theory: Set theory is either incomplete or inconsistent.

23.4 Exercises

1. What do Gödel's theorems say about statements such as, "Everything which is true can be proven?"
2. Why is it that Gödel's theorems do not contradict the Completeness Theorem for predicate logic?
3. A layman's translation of Göodel's theorems is, "Any sufficiently complex system is either incomplete or inconsistent." This statement has been used to criticize Christians by saying that the Bible is sufficiently complex, so it is either incomplete or inconsistent. Discuss this attack.

23.5 Gödel's Proof

In this section, we describe an encoding due to Gödel of formulas in the language of ℕ as natural numbers.

First, we can encode any finite ordered list of natural numbers as a natural number. Let $p_1, p_2, p_3, \ldots$ be the list of prime natural numbers – so $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, and so on. If $(n_1, n_2, \ldots, n_k)$ is any finite ordered list of natural numbers, let

$$\#(n_1, n_2, \ldots, n_k) = p_1^{n_1+1} p_2^{n_2+1} \cdots p_k^{n_k+1}.$$

For example

$$\#(5, 1, 2) = 2^6 3^2 5^3 = 64 \cdot 9 \cdot 125 = 72000.$$

Note: The "+1" in the exponent is present to accomodate any 0s that may be in the list. Since $n^0 = 1$, if we only had $p_k^{n_k}$, any 0s would not add a factor to the number encoding of the sequence.

Next, we can convert any expression in the language of ℕ to a finite ordered list of natural numbers. To do so, we first need to assign a number to each symbol in our language. An example assignment could be:

Symbol	Number	Symbol	Number	Symbol	Number
0	0	$=$	7	$\leftrightarrow$	14
s	1	$\forall$	8	)	15
$+$	2	$\exists$	9	(	16
$\cdot$	3	$\neg$	10	v_1	17
exponent	4	$\wedge$	11	v_2	18
$<$	5	$\vee$	12	v_3	19
$\leq$	6	$\rightarrow$	13		
					$\vdots$

Now any expression in the language of $\mathbb{N}$ can be written as a list of natural numbers by replacing each symbol in the expression by its corresponding number above. The list associated with an expression α will be denoted as $List(\alpha)$. For example

$$List((\exists v_1(v_1 = s(v_1)))) = (16, 9, 17, 16, 17, 7, 1, 16, 17, 15, 15, 15)$$

Any expression in the language of $\mathbb{N}$ can be encoded as a natural number. If α is any expression in the language of $\mathbb{N}$, then the Gödel number of α is $\#(List(\alpha))$. We will denote this as $G(\alpha)$. For example

$$\begin{aligned} G((\exists v_1(v_1 = s(v_1)))) &= \#(List((\exists v_1(v_1 = s(v_1))))) \\ &= \#(16, 9, 17, 16, 17, 7, 1, 16, 17, 15, 15, 15) \\ &= 2^{17}3^{10}5^{18}7^{17}11^{18}13^{8}17^{2}19^{17}23^{18}29^{16}31^{16}37^{16} \end{aligned}$$

This is a BIG number.

If Σ is any set of expressions, we will denote the set of Gödel numbers of elements of Σ as $G(\Sigma)$. And if $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ is an ordered list of expressions, then we define

$$G(\alpha_1, \alpha_2, \ldots, \alpha_k) = \#(G(\alpha_1), G(\alpha_2), \ldots, G(\alpha_k)).$$

Of course, each $G(\alpha_i)$ is BIG, so this number is HUGE!.

Note that each natural number can be expressed in the language of number theory. The language contains a symbol for 0. Then 1 is $s(0)$. 2 is $s(s(0))$ and so on. Thus, in being able to refer to all expressions, Gödel numbers can in particular reflect expressions about each natural.

We now have a way of encoding every expression and every finite ordered list of expressions from the language of $\mathbb{N}$ as a natural number. The power of the encoding is that it is simple to pass from wffs to Gödel numbers and back in a mechanical (computable, derivable) way. The importance of the encoding is that now each wff can be referenced as a number – certain natural numbers serve as "names" for wffs. Moreover, each of the following possible properties of a natural number n is definable in $\mathbb{N}$:

- The number n is the Gödel number of wff.
- The number n is the Gödel number of a wff with no free variables (or one, or two, etc.).
- The number n is the Gödel number of a wff α with one free variable with the number q substituted for the free variable.
- The number n is the Gödel number of an ordered list of wffs.
- The number n is the Gödel number of an ordered list of wffs which is a valid deduction.
- The number n is the Gödel number of a valid deduction of α – where α is a wff.
- The number n is the Gödel number of a valid deduction of α from Σ – where α is a wff and Σ is a set wffs so that $G(\Sigma)$ is definable.

23.6 Exercises

1. Convert this list of numbers to a natural number using the method prescribed in this section: (1, 2, 3, 4)
2. Convert this list of numbers to a natural number using the method prescribed in this section: (0, 1, 0, 1, 0, 1)
3. Convert this number to a list of numbers using the method of this section: 180
4. Convert this number to a list of numbers using the method of this section: 381150
5. Consider this recursive definition of a toy language using the symbols x, y, z, N, and R:

   - x, y, and z are wffs.
   - If α and β are wffs, then $\alpha\beta R$ and αN are wffs.
   - These are the only ways in which an expression with these symbols can be a wff.

   We outline a way to convert any wff in this language to a natural number mimicing the method of this chapter.

   - First, associate each symbol from our language with a number in the following manner:

N	R	x	y	z
1	2	3	4	5

- If α is a wff, let $List(\alpha)$ be the list of natural numbers obtained by replacing each symbol in α with natural numbers as in the table above. For example

$$List(xNyR) = (3, 1, 4, 2).$$

- If α is a wff, let $G(\alpha) = \#List(\alpha)$.

(a) List several wffs in this language.

(b) Convert the following wffs to natural numbers:

i. xN
ii. xyR
iii. $xyzRR$

(c) Which of the following numbers are $G(\alpha)$ for some wff α?

i. 288
ii. 864
iii. 583200
iv. 2916000

23.7 A Proof of an Incompleteness Theorem

We are now ready to attack Theorem 23.4.

Theorem 23.10: Suppose that Σ is a set of sentences true in $\mathbb{N}$ so that $G(\Sigma)$ is definable. There is a sentence σ which is true in $\mathbb{N}$ but which cannot be proven from Σ.

Proof: Define a relation R on $\mathbb{N}$ so that $R(a, b, c)$ if and only if

a is the number of a wff α and c is the number of a deduction from Σ of $\alpha(b)$.

That is, if

- α is a wff with a free variable and
- $b \in \mathbb{N}$ and
- Γ is an ordered list of wffs

then $R(G(\alpha), b, G(\Gamma))$ means that

$$\Gamma \text{ is a proof of } \alpha(b) \text{ from } \Sigma.$$

By our assumptions, R is definable since Σ is definable, so there is a wff ρ so that

$$R(a, b, c) \leftrightarrow \rho(a, b, c).$$

Let q be the Gödel number of $\forall v_3 \neg \rho(v_1, v_1, v_3)$. Let σ be $\forall v_3 \neg \rho(q, q, v_3)$. Then σ declares that no v_3 is the Gödel number of a deduction from Σ of $\alpha(q)$ where α is the wff whose Gödel number is q. But the wff whose Gödel number is q is $\forall v_3 \neg \rho(v_1, v_1, v_3)$, so σ declares there is no proof from Σ of $\forall v_3 \neg \rho(q, q, v_3)$. This last wff is σ, so σ says that there is no proof of σ from Σ.

$$\sigma \leftrightarrow (\sigma \text{ is not provable from } \Sigma.)$$

The sentence σ must be true in $\mathbb{N}$ as in Theorem 23.3. □

Chapter 24

Cardinality

We now address the notion of the cardinality or size of a set. This concept involves some surprising consequences which have been motivational in the development of set theory and logic. The techniques involved in addressing cardinality involve techniques which are useful in answering questions about computability and the existence of Liebniz's Truth Machine.

24.1 Cardinality

Suppose that A and B are sets. A *one-to-one correspondence* between A and B is a pairing of elements from A with elements from B so that every element of A is paired with exactly one element of B and every element of B is paired with exactly one element of A.[1]

Example 24.1: Here is a one-to-one correspondence between $\{a, b, c\}$ and $\{0, 1, 2\}$.

$$\begin{array}{ccc} a & b & c \\ \updownarrow & \updownarrow & \updownarrow \\ 0 & 1 & 2 \end{array}$$

For any set A, we have a "number" which is associated with the "size" of A. We call this the *cardinality* of A and denote it as $|A|$. Since A may be infinite, $|A|$ cannot be a "number" in the usual sense, so we introduce the notion of cardinal numbers. We make the following assumptions about cardinal numbers and cardinalities.

[1]This is truly an atrocious definition. What in the world is a "pairing?"

Cardinal Numbers

1. To every set A is associated a unique cardinal number $|A|$ called the cardinality of A.
2. For every cardinal number κ, there is a set A with $|A| = \kappa$.
3. For any $n \in \mathbb{N}$, $|\{0, 1, 2, \ldots, (n-1)\}| = n$.
4. For any set A, $|A| = 0$ if and only if $A = \emptyset$.
5. For any sets A and B, $|A| = |B|$ if and only if there is a one-to-one correspondence between A and B.
6. For any sets A and B, $|A| \leq |B|$ if and only if there is a one-to-one correspondence between A and a *subset* of B.
7. For any sets A and B, $|A| < |B|$ if and only if $|A| \leq |B|$ and $|A| \neq |B|$.

A set A is *finite* if $|A|$ is a natural number; otherwise, A is *infinite*. A set A is *countably infinite* if $|A| = |\mathbb{N}|$. A is *countable* if A is finite or countably infinite. In this case, there is a one-to-one correspondence between A and a subset of $\mathbb{N}$. If A is not countable, then A is *uncountably infinite* or simply *uncountable*.

Example 24.2: The set of even natural numbers is countably infinite as is witnessed by this one-to-one correspondence:

$$\begin{array}{ccccccc} 0 & 2 & 4 & 6 & 8 & \ldots \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ 0 & 1 & 2 & 3 & 4 & \ldots \end{array}$$

Example 24.3: The set of all integers is countably infinite as is witnessed by this one-to-one correspondence:

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & \ldots \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ 0 & -1 & 1 & -2 & 2 & \ldots \end{array}$$

The one-to-one correspondence establishing that a set is countable gives an ordered list of the elements of the set. Thus, we may use the word *listable* to say that a set is countable. Providing a list of the elements of a set will establish that the set is countable.

We have to try a bit harder to find uncountable sets (but they are everywhere).

A *sequence*[2] of 0s and 1s is an infinite ordered list of 0s and 1s such as 0011001100110011 .... If we name a sequence f, then the n^{th} number in the

[2]This is yet again an example of an inexcusably atrocious definition.

sequence will usually be called $f(n)$ or f_n. We begin numbering at 0, so that the first number in the sequence is $f(0)$, the second $f(1)$, and so on.

The proof of the following theorem makes use of a *diagonalization* technique due to Cantor which is a *self-reference* technique.

Theorem 24.4: There are uncountably many sequences of 0s and 1s.

Proof: We prove that we cannot list all sequences of 0s and 1s. Suppose that we have a list $f_0, f_1, f_2, \ldots$ of sequences of 0s and 1s (a sequence of sequences!). We will construct a sequence which is not in the list. For each $n = 0, 1, 2 \ldots$, let $g(n) = 1 - f_n(n)$ (so $g(n)$ is the opposite of the n^{th} number in the n^{th} sequence). Then g is a sequence of 0s and 1s, but for each n $g(n) \neq f_n(n)$. This means that $g \neq f_n$ for each n. Thus g is not in our list of sequences. Thus, any attempt to list all sequences of 0s and 1s must omit some sequence. These sequences are not listable or countable. □

We can associate any sequence of 0s and 1s with a subset of $\mathbb{N}$ in this manner: If f is a sequence, then define a set A of natural numbers so that $n \in A$ if and only if $f(n) = 1$. Thus, there are as many subsets of N as there are sequences of 0s and 1s.

Theorem 24.5: $\mathbb{N}$ has uncountably many subsets.

We can also associate every sequence of 0s and 1s with a real decimal number. We demonstrate this with an example. We associate the sequence $010101010101\ldots$ with the real number $0.010101010101\ldots$. Thus, there are as many real numbers as there are sequences of 0s and 1s. Actually, each of these numbers is between 0 and 1!

Theorem 24.6: There are uncountably many real numbers between 0 and 1.

24.2 Cantor's Theorem and Consequences

Theorem 24.5 declares that there are more subsets of $\mathbb{N}$ than there are elements of $\mathbb{N}$. This is a characteristic which is shared by all sets. The *powerset* of a set A is the set of all subsets of A. Denote the powerset of A as PA.

Theorem 24.7: Cantor's Theorem: For any set A, $|A| < |PA|$.

Proof: Suppose that A is a set and that P is its powerset. If A is the empty set, then A has 0 elements and P has 1 element, so $|A| < |PA|$.

Suppose that A is not empty and suppose by way of contradiction that there is a one-to-one correspondence between A and P. Every element of A is paired with an element $f(a)$ in P. Note that $f(a)$ is a *subset* of A, so we can ask whether or not $a \in f(a)$. Define B to be this subset of A:

$$B = \{a \in A : a \notin f(a)\}.$$

Since f is a one-to-one correspondence, there is some $x \in A$ with $f(x) = B$. Now we ask if $x \in B$. If $x \in B$, then by the definition of B, $x \notin f(x) = B$. This is a problem, so suppose that $x \notin B$. Then (again, by the definition of B) $x \in f(x) = B$. Thus

$$x \in B \text{ if and only if } x \notin B.$$

This contradiction shows that $|A| \neq |P|$.

Since P contains all of the singelton subsets of A, P must be at least as large as A, so $|A| < |P|$. □

Cantor's Theorem gives us:

An Infinitude of Infinities: By Cantor's Theorem, we must have

$$|\mathbb{N}| < |P\mathbb{N}| < |PP\mathbb{N}| < |PPP\mathbb{N}| < \cdots$$

so there are infinitely many different infinite cardinal numbers. We usually define

$$\aleph_0 = |\mathbb{N}|, \aleph_1 = |P\mathbb{N}|, \aleph_2 = |PP\mathbb{N}|, \aleph_3 = |PPP\mathbb{N}|, \cdots$$

so that

$$\aleph_0 < \aleph_1 < \aleph_2 < \aleph_3 < \cdots$$

A natural question to ask is if there are any cardinalities between these. The assumption is not.

Continuum Hypothesis: There is no set A with

$$\aleph_0 < |A| < \aleph_1.$$

Remarkably, this hypothesis can be assumed or rejected without affecting the consistency of set theory.

The reasoning involved in the proof of Cantor's Theorem arises also when we consider the collection of all sets. The set theory laid out by Frege allowed for this collection to be a set. Bertrand Russell pointed out that this creates an inconsistency or paradox. Suppose that U is the set of all sets. If U is

itself a set, then we have the odd situation where $U \in U$. This observation motivates us to define

$$B = \{x \in U : x \notin x\}.$$

We now ask the question whether or not B has the same odd property as U. If $B \in B$, then from the definition of B, $B \notin B$. If $B \notin B$, then from the definition of B, it has to be that $B \in B$. Thus

$$B \in B \text{ if and only if } B \notin B.$$

Most mathematicians avoid this contradiction today by declaring that U is not itself a set and by requiring definitions of sets by first order predicates to restrict possible elements to lie within a known set. For example, $\{x : P(x)\}$ is meaningless unless we know where x may "live." $\{x \in A : P(x)\}$ is meaningful as long as A is a set.

We can also argue by way of cardinalities that the collection U of all sets should not be a set. If U is a set, then it should have a cardinality. Since U contains the singleton subsets of every possible set, this cardinality should be the largest among the cardinalities of all sets. However, from Cantor's Theorem, we know that $|U| < |PU|$. If U were a set, we would again have a contradiction. U is *too big* to be a set.

24.3 Exercises

1. Let $A = \{10, 11, 12, 13, 14, \ldots\}$. Find a one-to-one correspondence to show that $|A| = |\mathbb{N}|$.
2. Find a one-to-one correspondence between $\mathbb{Z}$ and $\mathbb{N}$ to show that $|Z| = |N|$.
3. You are the clerk at a hotel with a countably infinite number of rooms. One night, every room is full. A new guest arrives. How can you make room for him?
4. List all of the subsets of $\{a\}$.
5. List all of the subsets of $\{a, b\}$.
6. List all of the subsets of $\{a, b, c\}$.
7. Draw Hasse diagrams of the sets of subsets in the previous two exercises.

Chapter 25

Machines

In Chapter 2 we noted that at the end of the 1600s Leibniz envisioned a machine which could recognize true statements. Before we can discuss the existence (or nonexistence) of such a machine, we need to talk about what a machine is. In this chapter we introduce three different types of theoretical machines or computers that have been used in the study of computability, logic, and language.

25.1 Turing Machines

Turing machines were introduced by Alan Turing in the 1930s. The "machinery" that makes up a Turing machine begins with a *tape* that has cells in which symbols may be written:

···	b	b	b	0	1	1	0	b	b	b	b	b	···

This tape serves as the memory of the machine. The tape extends indefinitely in both directions so that the machine has unlimited memory. The machine also includes a read/write *head* that points to a particular location on the tape:

···	b	b	b	0	1	1	0	b	b	b	b	b	···
						⇑							

The machine is said to be reading whatever symbol the head is pointing at. Programs for a Turing machine may read the symbol to which the head is pointing and (based on the symbol being read) write a symbol on the tape (at the location of the head) and move the head either to the right or to the left. A Turing machine also has an internal *state* which is used to determine which commands should be executed. There can only be finitely many states. We will use names of states such as s_0, s_1, s_2 .... We will record the state of the machine along with the location of the head like so:

⋯	b	b	b	0	1	1	0	b	b	b	b	b	⋯
						⇑							
						s_0							

We will view a command for a turing Machine is a row in a table such as this:

State	Symbol	New Symbol	New State	Direction
s_2	0	1	s_4	Right

This command means:

If the machine is in state s_2 and the head is reading "0" then

- Write a "1"
- Change to state s_4
- Move one step to the Right.

If the machine reaches a state/symbol combination for which there is no command, then the machine "halts" (simply stops running). We will assume that our machines always begin in state s_0 with the head pointing at the left-most nonblank cell of the tape. We will use a "b" to indicate a blank cell.

Example 25.1: Execute these Turing instructions:

State	Symbol	New Symbol	New State	Direction
s_0	0	1	s_0	Right
s_0	1	0	s_0	Right

on a Turing machine with this initial configuration:

⋯	b	b	b	0	1	1	0	b	b	b	b	b	⋯
				⇑									
				s_0									

We begin with this configuration:

⋯	b	b	b	0	1	1	0	b	b	b	b	b	⋯
				⇑									
				s_0									

Since we are in state s_0 reading a 0, we execute the first command. We write a 1, stay in state s_0, and move one step right:

⋯	b	b	b	1	1	1	0	b	b	b	b	b	⋯
					⇑								
					s_0								

Now we are in state s_0 reading a 1, so we execute the second command. We write a 0, state in state s_0, and move right:

···	b	b	b	1	0	1	0	b	b	b	b	b	···
						⇑							
						s_0							

We are in state s_0 reading a 1. Again, according to the second command, we write a 0, state in state s_0, and move right:

···	b	b	b	1	0	0	0	b	b	b	b	b	···
							⇑						
							s_0						

We are in state s_0 reading a 0. According to the first command, we write a 1, stay in state s_0, and move right:

···	b	b	b	1	0	0	1	b	b	b	b	b	···
								⇑					
								s_0					

Finally, we are in state s_0 reading a blank cell. Since there is no command for this, the machine halts. What have we accomplished? This Turing machine has passed along the tape from left to write changing every 0 to a 1 and changing every 1 to a 0.

Example 25.2: Execute these Turing instructions:

State	Symbol	New Symbol	New State	Direction
s_0	0	1	s_0	Right
s_0	1	1	s_0	Right
s_0	b	1	s_1	Left
s_1	0	1	s_1	Left
s_1	1	1	s_1	Left
s_1	b	1	s_0	Right

on a Turing machine with this initial configuration:

···	b	b	b	0	b	b	b	b	b	b	b	b	···
				⇑									
				s_0									

We begin with:

···	b	b	b	0	b	b	b	b	b	b	b	b	···
				⇑									
				s_0									

According to the first instruction, we write a 1, stay in state s_0, and move right:

···	b	b	b	1	b	b	b	b	b	b	b	b	···
					⇑								
					s_0								

Now, according to the third command, we write a 1, change to state s_1 and move left:

$\cdots$	b	b	b	1	1	b	b	b	b	b	b	b	$\cdots$
				$\Uparrow$									
				s_1									

Write a 1, stay in state s_1, and move left:

$\cdots$	b	b	b	1	1	b	b	b	b	b	b	b	$\cdots$
			$\Uparrow$										
			s_1										

Write a 1, change to state s_0, and move right:

$\cdots$	b	b	1	1	1	b	b	b	b	b	b	b	$\cdots$
				$\Uparrow$									
				s_0									

If we continue this process, as long as we are in state s_0, we will write 1s and move right until we reach a blank cell. There, we will write a 1, change to state s_1 and start moving left. In state s_1, we move left, writing 1s, until we reach a blank cell. This process adds a 1 to the right of the tape and then to the left and repeats indefinitely. The machine never halts. A natural question to ask is

> How can one determine whether or not a Turing machine will halt?

This turns out to be a difficult question which we will address in Chapter 26.

Turing machines may seem like toys with no real application. However, with correct encoding of symbols, Turing machines are capable of any computation that a modern computer can perform.

25.2 Exercises

1. Run these Turing commands:

State	Symbol	New Symbol	New State	Direction
s_0	0	0	s_0	Right
s_0	1	1	s_0	Right
s_0	b	1	s_1	Left
s_1	1	1	s_1	Left
s_1	0	0	s_1	Left
s_1	b	b	s_2	Right

on a Turing machine with this initial configuration:

$\cdots$	b	b	b	0	1	1	0	b	b	b	b	b	$\cdots$
				$\Uparrow$									
				s_0									

2. Run these Turing commands:

State	Symbol	New Symbol	New State	Direction
s_0	0	0	s_1	Right
s_0	1	1	s_1	Right
s_0	b	b	s_1	Right
s_1	0	0	s_0	Right
s_1	1	1	s_0	Right
s_1	b	b	s_0	Right

   on a Turing machine with this initial configuration:

$\cdots$	b	b	b	0	1	1	0	b	b	b	b	b	$\cdots$
				$\Uparrow$									
				s_0									

3. Design a Turing machine with these specifications:

   - The machine uses symbols 0, 1, and b for blank.
   - The machine starts with the head at the left-most non-blank cell.
   - The machine should move the head all the way to the far right of the non-empty cells and write a 1 in the first blank cell to the right.

4. Design a Turing machine with these specifications:

   - The machine uses symbols 0, 1, and b for blank.
   - The machine starts with the head at the left-most non-blank cell.
   - Assume that the tape begins with non-blank cells looking something like: $11\cdots11011\cdots111$, a bunch of 1s, followed by a 0, followed by a bunch of 1s.
   - The machine should remove one 1 from the left and replace the middle 0 with a 1.

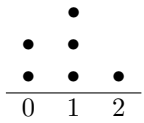
   If we represent the number n on the tape as n 1s in order, then our first tape has two numbers m and n separated by a 0. This machine ends with a sequence of $m + n$ 1s, so this is an adding machine.

5. Design a Turing machine with these specifications:

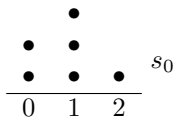
   - The machine uses symbols 0, 1, and b for blank.
   - The machine starts with the head at the left-most non-blank cell.
   - The machine should back up the head up one place to the left and then write a copy of the first non-blank cell on the tape (the cell it was initially reading).

25.3 Minsky Machines

A Minsky machine or register machine is another type of machine whose instructions are based on finitely many states. Instead of a tape, the memory of a Minsky machine is a set of bowls (also called urns or registers) in which are placed pebbles. This is how we will picture a machine with three bowls. There are two pebbles in the first bowl, three in the second, and one in the third:



We will write the interal state of a Minsky machine to the right of the bowls:



Commands for a Minsky machine will either place a pebble in a particular bowl or remove a pebble from a bowl and then change states. Since bowls may be empty, the action of removing a pebble may fail. The state to which the machine changes after trying to remove a pebble will depend on whether or not the removal was successful. For us, commands for Minsky machines will look like rows in tables such as this:

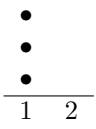
State	Action	Bowl	Pass	Fail
s_0	+	1	s_1	
s_1	-	2	s_2	s_3

The first command says that if the machine is in state s_0, then add (hence the +) a pebble to bowl 1 and go to state s_1. "Pass" means that the action was successful, and "Fail" means the action was not successful. Since adding a pebble will always be possible (We have arbitrarily many pebbles, and our bowls can hold as many pebbles as we like.) we do not need a Fail state. The second command says to remove (hence the -) a pebble from bowl 2. If removal is successful, go to state s_2. If removal is not successful, go to state s_3. If a Minsky machine ever encounters a state with no command, the machine halts. Such a state is a halting state.

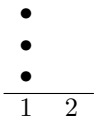
Example 25.3: Execute these Minsky machine commands:

State	Action	Bowl	Pass	Fail
s_0	-	1	s_1	s_2
s_1	+	2	s_0	

on a machine with this configuration:



We begin like so:



Since we are in state s_0, we remove one pebble from bowl 1 and go to state s_1:

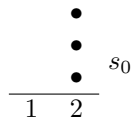
In state s_1, we add a pebble to bowl 2 and go to state s_0:

Remove another pebble from bowl 1 and go to s_1:

Add another pebble to bowl 2 and go to s_0:

Remove another pebble from bowl 1 and go to s_1:

Add a pebble to bowl 2 and go to s_0:



Now we try to remove a pebble from bowl 1, but bowl 1 is empty. Since the action fails, we go to s_2. Since there is no command for s_2, the machine now halts. Here is a summary: In s_0, we are removing pebbles from bowl 1. In s_1, we are placing the pebbles back in bowl 2. Finally, s_2 is the halting state. What have we accomplished here? We have moved the contents of bowl 1 to bowl 2.

Minsky machines can perform a computation equivalent to any computation that a Turing machine can perform. In particular, Minsky machines can perform basic arithmetic.

Example 25.4: Design a Minsky machine which adds. The machine should have three bowls. Assume that the third bowl begins empty. After the machine runs, the third bowl should contain the sum of the numbers of pebbles from the first two bowls.

This task sounds more difficult than it is. First, we mimic the previous example to move the contents of bowl 1 to bowl 3:

State	Action	Bowl	Pass	Fail
s_0	-	1	s_1	s_2
s_1	+	3	s_0	

Then, once we reach s_2, we mimic the previous example again to move the contents of bowl 2 to bowl 3:

State	Action	Bowl	Pass	Fail
s_0	-	1	s_1	s_2
s_1	+	3	s_0	
s_2	-	2	s_3	s_4
s_3	+	3	s_2	

Example 25.5: It is easy to construct a Minsky machine which does not halt:

State	Action	Bowl	Pass	Fail
s_0	+	1	s_0	

So we again have the obvious question:

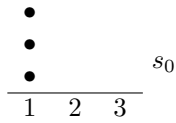
How can one determine whether or not a Minsky machine will halt?

25.4 Exercises

1. Execute these Minsky machine commands:

State	Action	Bowl	Pass	Fail
s_0	-	1	s_1	s_3
s_1	+	2	s_2	
s_2	+	3	s_0	

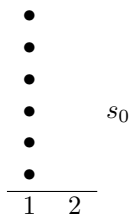
   on a machine with this configuration:



2. Execute these Minsky machine commands:

State	Action	Bowl	Pass	Fail
s_0	-	1	s_1	s_4
s_1	-	1	s_3	s_4
s_3	+	2	s_0	

   on a machine with this configuration:



3. Design a Minsky machine with 3 bowls. The machine should copy bowl 1 to bowl 2 and to bowl 3. So, if the machine begins with 4 pebbles in bowl 1, it should end with 0 in bowl 1, 4 in bowl 2, and 4 in bowl 3.

4. Design a Minsky machine with 1 bowl. If the machine begins with an odd number of pebbles in bowl 1, then the machine should end with 1 pebble in bowl 1. If the machine begins with an even number of pebbles in bowl 1, then it should end with no pebbles in bowl 1.

5. Desing a Minsky machine with two bowls which will subtract bowl 1 minus bowl 2. You should have two bowls. For each pebble in bowl 2, remove a pebble from bowl 1 until one or the other bowl is empty.

25.5 Language Recognition

Leibniz wanted to construct a machine which could identify true and false statements. As a step in this direction, we will define here what it means for a Turing machine to recognize a language. Suppose that α is an expression written using the symbols used by a Turing machine T. We assume that T has a state s_f identified as its *final state* (or happy state). We say that T recognizes α if T halts in state s_f when run on a tape with α written on it[1].

Example 25.6: Suppose that a Turing machine T is governed by these commands:

State	Symbol	New Symbol	New State	Direction
s_0	0	0	s_2	Right
s_0	1	1	s_1	Right
s_1	1	1	s_0	Right

and suppose that s_0 is T's happy state. What expressions (sequences of 0s and 1s) are recognized by T?

First note that if the machine is in state s_0 and reads a 0, then the machine goes to s_2 and halts because there are no commands for state s_2. Also, note that there are no commands for state s_1 while reading a 0. Therefore, if the machine reads a 0 in either state, the machine will halt either in s_1 or s_2, and the expression on the tape will not be recognized (because the happy state is s_0). Thus any expression recognized by T can contain only 1s. Now consider what happens when the machine reads 1s. When the first 1 is read, the machine goes to s_1 (unhappy). When the second 1 is read, the machine returns to s_0 and is happy. The next 1 sends T to s_1 (unhappy), and the next to s_0 (happy). The machine alternates s_0, s_1, s_0, s_1, and so forth. It is in state s_0 every other 1 beginning with the second 1. Thus, T will recognize an expression if and only if that expression is exactly a sequence of an even number of 1s.

A natural question to ask is which sets or languages can be recognized by Turing machines. It can be shown that a language can be recognized by a Turing machine if and only if it is generated by a phrase structure grammar.

Example 25.7: Make up a phrase structure grammar which will generate the language recognized by the Turing machine in the previous example.

We will use A as the start symbol for our grammar, and we will have one other (terminal) symbol 1. We have two production rules:

$$A \to 11A \text{ and } A \to 11.$$

[1] An alternative is to say that α is recognized if T halts in any state and to assume that T will not halt if α is not recognized. Our notion is equivalent and parallels what we will do later with Finite State Automata.

The first rule allows us to grow our number of 1s (two at a time to keep that number even). The second rule allows us to remove the start symbol from the derivation.

We can also use Minsky machines to recognize sets; however, what we can recognize are numbers. As with Turing machines, we need to make sure our Minsky machine has a happy state s_f specified. Then the machine recognizes a number n if the machine halts in state s_f when the machine is run beginning with n pebbles in the first bowl and all other bowls empty.

Example 25.8: Suppose a Minsky machine M has one bowl and these commands

State	Action	Bowl	Pass	Fail
s_0	-	1	s_1	s_3
s_1	-	1	s_2	s_4
s_2	-	1	s_0	s_4

and suppose that the happy state of M is s_3. What numbers does M recognize?

If M successfully takes one pebble from the bowl, then it go to s_1. If it successfully takes a second pebble from the bowl, it goes to s_2. On the third pebble, it goes to s_0. Then the process repeats. The states follow a pattern s_0, s_1, s_2, s_0, s_1, s_2, s_0, s_1, s_2, and so on. M lands in s_0 on the 0^{th}, 3^{rd}, 6^{th}, 9^{th}, and so on pebbles, so M is in state s_0 immediately after the number of pebbles that have been drawn is a multiple of 3. If the bowl is empty at this point in time, the next attempt to remove a pebble will land the machine in its happy state s_3. If the machine is empty after the number of marbles drawn is not a multiple of 3, then the next removal will land the machine in an unhappy state s_4. It seems that a number is recognized by this machine if and only if the number is a multiple of 3.

25.6 Exercises

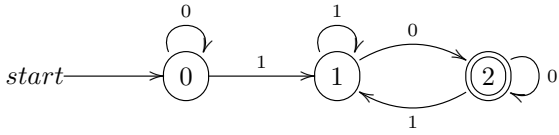
1. Design a Turing machine which will recognize expressions which contain only 1s.
2. Design a Turing machine which will recognize sequences of 0s and 1s that look like 1010101. The 1s and 0s alternate, and the sequences begin and end with a 1.
3. Write a phrase structure grammar which generates the language from the previous exercise.
4. Design a Turing machine which will recognize expressions of 0s and 1s that contain exactly 2 1s.
5. Design a Turing machine which will recognize expressions of 0s and 1s that contain more than 2 1s.

6. Desing a Turing machine which recognizes expressions of the form 01, 0011, 000111,.... These expressions contain some number of 0s followed by the same number of 1s.

25.7 Finite State Automata

Our third type of machine is a finite state automaton (plural, automata). Finite state automata (as their name indicates) are also based on finitely many states. These machines accept finite sequences of symbols as input. They read the symbols one at a time, and commands specify what state to move to if the machine is in a specific state reading a particular symbol[2]. One or more states of the machine are declared to be *halting states* (or, as I like to say, happy states). If the machine is in one of these states when it runs out of input symbols, the input is said to be recognized. Otherwise, the input is not recognized.

We could use rows in tables to list the commands for finite state automtata as we did with Turing machines and Minsky machines; however, there is a convenient way to represent all the information regarding a finite state automaton in a graphical way. Each state of the machine is represented as an oval containing the name of the state. The ovals for the halting states are usually embellished with bold print or double lines. Each instruction for the machine is an arrow. An arrow from state A to state B labeled by a symbol s means that if the machine is in state A reading an s, then go to state B. The initial state of the machine is usually indicated with a start arrow. Here is an example of a diagram of a finite state automaton:



Example 25.9: Run the finite state automaton above on the input 0110100.

As indicated by the start arrow, we begin in state 0.

- The first symbol we read is a 0: 0110100. Since we are in state 0 reading a 0, we follow the arrow above state 0...which leads back to state 0.
- The next symbol is a 1: 0110100. Since we are in state 0 reading a 1, we follow the arrow from state 0 to the right to state 1.
- Next comes another 1: 0110100. Since we are in state 1 reading a 1, we follow the arrow above state 1 back to state 1.

[2]We are studying finite state automata without output. Finite state automata with output print an output symbol at each step of the computation

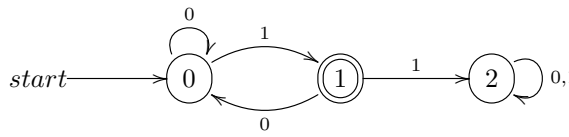
- Next is a 0: 011$\underline{0}$100. We are in state 1 reading a 0, so we follow the arrow to the right which leads to state 2.
- We are momentarily happy, but we are in state 2 reading a 1: 0110$\underline{1}$00 so we follow the arrow to the left of state 2 back to state 1.
- In state 1, readind a 0: 01101$\underline{0}$0, we follow the arrow back to state 2.
- One symbol left, a 0: 0110100$\underline{0}$. Since we are in state 2 reading a 0, we stay in state 2.
- We are out of input symbols, so the computation is done. Since we are in the halting state, state 2, the machine recognizes the input.

Example 25.10: Run the finite state automaton above on the input 101.

Again, we begin in state 0.

- In state 0, reading $\underline{1}$01, go to state 1.
- In state 1, reading 1$\underline{0}$1, go to state 2.
- In state 2, reading 10$\underline{1}$, go to state 1.
- We are out of input symbols, so the computation is done. Since we are in state 1 and not the halting state, state 2, the machine does note recognize the input.

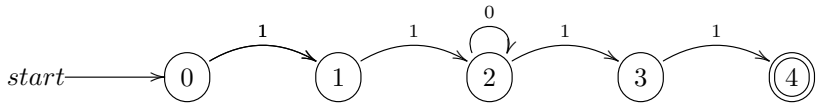
Example 25.11: What sequences of 0s and 1s will this finite state automaton recognize?



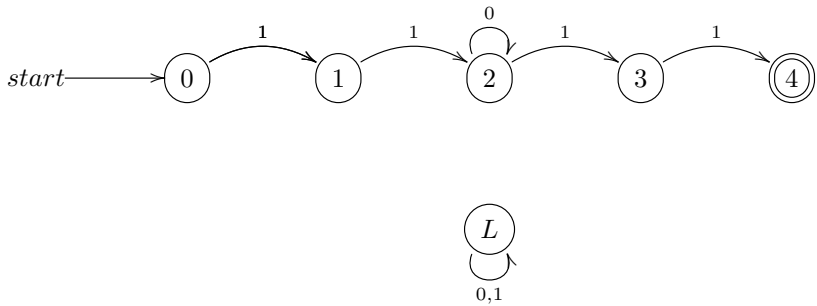
Before we begin, notice the loop on state 2 is labeled by both a 0 and a 1. If the machine ever reaches state 2, it never leaves state 2. In particular, it cannot get back to the halting state. If the machine starts in state 0 and reads any number of 0s, it remains in state 0. The first 1 it encounters moves the machine to state 1. Another 1 will move the machine to state 2. While in state 1, a 0 will send the machine back to state 0, where it will remain until another 1 is read. If the input does not contain two consecutive 1s, then the machine will not reach state 2. If the input ends in a 0, the machine will end in state 0. To end in state 1 (the happy state) the input cannot have two consecutive 1s and must end in a 1.

Example 25.12: Design a machine which will recognize any input of 0s and 1s which begins with two 1s, then has any number of 0s, and then ends with two 1s.

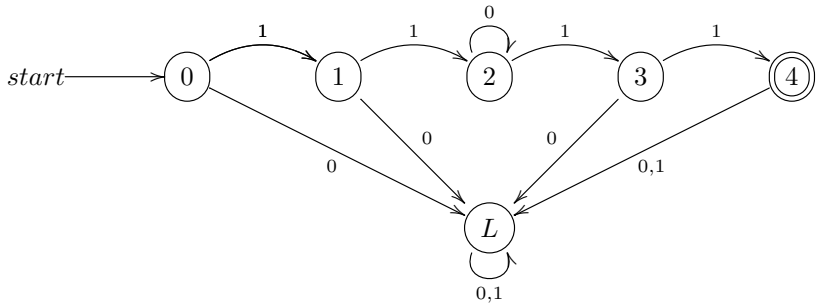
The inputs we want to recognize look like 11000..00011. First, we set up states that will take any input that looks like this to a happy state.



Next, we use a trick to prevent the machine from reaching the happy state if any "bad" input is given to the machine. We do this by introducing a new state "L" that will act as a trap. To make it a trap, we we add a loop mapping L to itself on any input symbol:



Finally, note that the arrows in the top half of our diagram are exactly the symbols we want to read to get to the happy state. We add arrows for any other symbol to send the machine straight to L if a bad symbol is read:



A natural question to ask is what sets of symbols or what languages can be recognized by a finite state automaton. These are those languages generated by phrase structure grammars where every production rule is of the form $A \to a$ or $A \to aB$ where A and B are nonterminal symbols and a is a terminal symbol. Such languages are called *regular languages*. An example of a set which cannot be recognized by a finite state automaton is the set

01, 0011, 000111, 00001111, 0000011111, 000000111111...

The sequences in this set consist of any number of 0s followed by the same number of 1s.

25.8 Exercises

1. Design a finite state automaton which will recognize expressions which contain only 1s.
2. Design a finite state automaton which will recognize sequences of 0s and 1s that look like 1010101. The 1s and 0s alternate, and the sequences begin and end with a 1.
3. Design a finite state automaton which will recognize expressions of 0s and 1s that contain exactly 2 1s.
4. Design a finite state automaton which will recognize expressions of 0s and 1s that contain more than 2 1s.

Chapter 26

Computability

A 0-1 sequence f is *computable* if there is an algorithm or program which with input n will output $f(n)$. Note that any program is just a finite list of symbols. There are countably many such lists (in any reasonable, fixed language), so we will assume that we have a list of all programs $P_0, P_1, \ldots$. Since there are countably many programs, there are countably many computable sequences. We can list all computable sequences as $C_0, C_1, C_2, \ldots$. Since there are uncountably many sequences, *some sequences are not computable.* This is good. Otherwise, the definition of computable sequence would be pointless.

Theorem 26.1: There is a 0-1 sequence f for which there is no program which inputs n and outputs $f(n)$.

We address the question as to whether or not there is a program which will decide if a given program will stop or run forever. This is known as the Halting Problem.

Halting Problem: Is there an algorithm which accepts as input a description of a program P and an input to that program and decides if P halts on that input?

The answer to this question was given independently by Church and Turing. We give a very brief outline of the idea of Turing's proof.

Theorem 26.2: Church-Turing There is no program which will determine if any given program will halt on any given input.

Proof: Suppose that U is a program which accepts two natural numbers as input and outputs either "halts" or "does not halt" on every input. We will

construct a program and input so that U does not draw the correct conclusion about halting for that program. Thus U cannot determine whether or not programs halt. Since U is arbitrary, no program can make this determination.

Now we construct a new program H which takes a single input n and follows these rules:

1. If $U(n,n) =$"does not halt" then H halts and outputs "halt."
2. If $U(n,n) =$"halts" then H loops forever (does not halt).

Thus $H(n)$ is in some sense the opposite of $U(n,n)$.

Now, since H is a program, there is a number k so that $H = P_k$. We now consider $U(k,k)$. If the result of $U(k,k)$ is "does not halt" then H, by definition, must halt on input k. This means that P_k must halt on input k. If the result of $U(k,k)$ is "halts" then H does not halt on input k. This means that P_k does not halt on input k. Thus

$$U(k,k) = \text{"halts" if and only if } P_k \text{ does not halt on input } k.$$

The program U cannot draw the correct conclusion about what P_k does on input k. □

In the previous argument, associate "halts" with 1 and "does not halt" with 0. The program U outputs either 1 or 0. For any n, $H(n)$ is *almost* defined to be $1 - U(n,n)$. Instead of outputting $1 - U(n,n)$, H instead *performs* $1 - U(n,n)$.

Our next theorem basically says that the list of computable sequences is not computable. It declares that there is no "universal machine" that can perform the task of every program.

Theorem 26.3: There is no program U which accepts two inputs m and n and whose output $U(m,n)$ is $C_m(n)$.

Proof: Suppose that there is a program U which when given a pair of natural numbers m, n as inputs will output $C_m(n)$. We could express this as $U(m,n) = C_m(n)$. Let $g(n) = 1 - C_n(n)$. Now, g cannot be in the list $C_0, C_1, \ldots$ since for any n, $g(n) \neq C_n(n)$ so $g \neq C_n$. However, we can compute g in the following way:

- Input n.
- Calculate $U(n,n)$.
- Output $1 - U(n,n)$.

Thus we have a contradiction. The sequence g is computable, but it is not in our list of all computable sequences. □

26.1 Truth Machines

Over the past century, there have been many (equivalent) suggestions about what an algorithm or computer program should be. In any case, there are finitely many commands or assumptions in some language that are applied to the input x. The machine may use arbitrarily much memory and time, but any output must come after only finitely much memory and time have been used.

It is not much of a step to consider a program to be written in a language that corresponds to some first order language. The commands of the program correspond to sentences in the language. Calculations correspond to deductions. This implies that the decidability of a set corresponds to the ability to *prove* that elements either are or are not in the set based on finitely many axioms or assumptions.

It seems reasonable, then, that computability or decidability should be equivalent to deducibility from finitely many axioms. This assumption is known as *Church's Thesis*.

A consequence of Theorem 23.7 is that there is no finite set of sentences which axiomatize $\mathrm{Th}(\mathbb{N})$. Since Church's Thesis identifies computability with deduction from a finite set of axioms, this result can be expanded to:

Theorem 26.4: Church-Turing: There is no algorithm which will determine if an arbitrary sentence in the language of $\mathbb{N}$ is true in $\mathbb{N}$.

The situation is even more bleak than it sounds for Leibniz's Truth Machine. One could argue that asking to determine the truth value of *any* statement about the natural numbers is unreasonable. We can isolate our attention to what sounds like a much simpler question: Is there an algorithm which, given a polynomial with natural number coefficients, will determine if there are natural numbers that make that polynomial 0? This is the type of exercise that students are asked to perform in an introductory algebra class.

In 1970 Yuri Matiyasevich, extending work of Julia Robinson, Martin Davis, and Hilary Putnam, proved that the answer to this question is negative. For every Minsky machine $\mathbf{M}$, there is a polynomial $P(x_1, \ldots, x_9)$ with nine variables so that there are natural numbers $n_1, n_2, \ldots, n_9$ with $P(n_1, n_2, \ldots, n_9) = 0$ if and only if the machine $\mathbf{M}$ halts. Therefore, if a machine could decide if any polynomial can be made zero with natural number inputs, then that machine could be used to determine if any Minsky machine halts and would give a positive solution to the Halting Problem. Since no machine can solve the Halting Problem, no machine can solve the polynomial problem either.

26.2 Diagonalization

Many of the theorems in these notes involve a common theme which is usually described as a "diagonalization argument." In each example below, note how a negation is applied to a formula (logical or arithmetical) in which two variables are identified.

- In Theorem 23.4, let $\alpha(x, y)$ be $\exists v_3 \rho(x, y, v_3)$. Then the formula σ is precisely $\neg\alpha(q, q)$.
- In Theorem 24.4, let $h(x, y) = f_x(y)$. Then $g(n)$ is defined to be $1 - h(n, n)$.
- In Cantor's Theorem 24.7, let $\alpha(x, y)$ be "$x \in f(y)$." Then the set B is the set of all $x \in A$ for which $\neg\alpha(x, x)$ holds.
- In the discussion of Russell's Paradox, let $\alpha(x, y)$ be "$x \in y$." The set B in question here is the set of all $x \in U$ for which $\neg\alpha(x, x)$ holds.
- In Theorem 26.2, the machine H performs the opposite of $U(n, n)$. If we interpret U logically, then $H(n)$ is equivalent to $\neg U(n, n)$. If we interpret U arithmetically, then $H(n)$ is (almost) $1 - U(n, n)$.
- In Theorem 26.3, we encounter the expression $1 - U(n, n)$.