Phil 150/250 - Mathematical Logic
Course Notes

Prof. Thomas Icard
Philosophy Department

Draft of November 20, 2023



Contents

1 Introduction **3**

1.1 A Brief Potted History 3

2 Review: Sets, Relations, Functions **8**

2.1 Sets 8

2.2 Relations 11

2.3 Functions 13

3 Boolean Logic **15**

3.1 The Boolean Connectives 15

3.2 Truth Tables and Semantics 17

3.3 Truth Functions 20

3.4 Metalogical Notions 21

3.5 The Material Conditional 24

3.6 More Connectives 26

3.7 Normal Forms: DNF and CNF 29

3.8 A Formal Proof System 31

3.9 Metalogical Concerns 43

4 Applications of Boolean Logic **46**

4.1 Natural Language: Logical Connectives in English 46

4.1.1 Successes 46

4.1.2 Puzzles 48

4.1.3 Gricean Pragmatics 49

4.2 Computation I: Circuits 52

4.3 Computation II: Complexity 54

4.3.1 A Curious Connection to Physics 56

4.3.2 Example: The Clique Problem 57

4.3.3 Reducing SAT problems to Clique Problems 58

4.3.4 Reducing Clique Problems to SAT Problems 59

4.4 Psychology: Boolean Concepts 60

4.5 Mathematics: Boolean Algebra 63

5 Modal Logic **69**
5.1 General Modal Logic . . . 69
5.1.1 The Language of Modal Logic . . . 69
5.1.2 Modal Models . . . 70
5.1.3 Frames & Definability . . . 73
5.2 Epistemic Logic . . . 76
5.3 Temporal Logic . . . 81
5.3.1 Branching Time Temporal Logic . . . 85
5.3.2 *Since*, *Until*, and Program Specifications . . . 87

6 First Order Logic **89**
6.1 Reasoning about Relations . . . 89
6.2 First Order Languages . . . 94
6.3 Models of First Order Logic . . . 97
6.4 Metalogical Notions . . . 105
6.5 Undecidability of First Order Logic . . . 109
6.5.1 A Primer on Computability and Turing Machines . . . 109
6.5.2 Encoding Turing Computations into First Order Logic . . . 112
6.5.3 Establishing Undecidability . . . 115
6.6 A First Order Proof System . . . 116
6.6.1 The Universal Quantifier . . . 117
6.6.2 The Existential Quantifier . . . 122
6.6.3 Some Remarks on Metatheory . . . 124
6.7 First Order Arithmetic . . . 125

Chapter 1

Introduction

Logic stands at the foundation of a number of disciplines, often appearing where you least expect it. This course is intended as a broad introduction to key mathematical and conceptual ideas in the study of logic. In addition to providing the essentials of Boolean (propositional) logic, modal logic, and First Order (predicate) logic, we will also survey some applications of these ideas in semantics and philosophy of language, computer science, psychology, epistemology, and mathematics, with some brief pointers in addition to topics in microeconomics and even physics.

Those who finish and master this course should be well prepared for more advanced courses in mathematical logic (such as Phil 151), and should understand the application of logic in the study of language, information, and computation. Pointers to further study will be given throughout.

1.1 A Brief Potted History

For the first two-thousand years or so, the study of logic was primarily concerned with the nature of **argumentation** and distinguishing **good** from **bad reasoning**. The first systematic work in logic in the west was by the philosopher Aristotle, pictured here (arguing with Plato).



He introduced a basic form of reasoning known as the **syllogism**, which came to dominate logic in Europe and much of the Middle East for centuries.[1] The syllogism is a simple two-premise, one-conclusion argument pattern such as this:

[1] There were also very rich logical traditions in India and China going back to the same period, even slightly earlier.

$$\frac{\begin{array}{c}\text{All zookeepers are pacifists}\\ \text{All pacifists are troglodytes}\end{array}}{\text{All zookeepers are troglodytes}}$$

The interesting feature of this argument is that the conclusion is *guaranteed* to be true, provided the premises are true. It may or may not be that all zookeepers are pacifists, or that all pacifists are troglodytes. The important point is that, *if* both of these are true, then it *must* be true that all zookeepers are troglodytes. This is a **valid** argument of the following form:[2]

$$\frac{\begin{array}{c}\text{All } A \text{ are } B\\ \text{All } B \text{ are } C\end{array}}{\text{All } A \text{ are } C}$$

We have just pointed out that it does not matter here what A, B, and C are. The conclusion is guaranteed to be true whenever the premises are. As Aristotle put it in the *Prior Analytics*,

> "In correct logic, certain things are supposed, and something else results of necessity."

On this traditional conception of **logical argumentation**, the logician is not concerned with the way the world in fact is (truth and falsity), but merely with the abstract form of good argumentation—that is, with the rules governing truth and falsity.

Lest one suppose from the example above that syllogisms are all trivial, try to determine whether the following is a good logical argument:

$$\frac{\begin{array}{c}\text{No students are astronauts}\\ \text{Some astronauts are quadragenarians}\end{array}}{\text{Some quadragenarians are not students}}$$

Syllogisms were the main focus of the western study of logic through the Middle Ages, and were extended to capture basic reasoning about **time**, **possibility** and **necessity**, etc. Much of this was done outside Europe, by such notable thinkers as the Islamic scholar Ibn Sīnā:



The syllogism was so central for so long that in the 18th century philosopher Immanuel Kant declared,

> "Logic has not gained much since Aristotle's time, indeed it cannot. Logic is in a permanent state from which it cannot undergo any change."

Following this long period, the 19th century saw rather dramatic progress in mathematics—especially the more abstract parts of mathematics—and logic was no exception. The first major advance was by a self-taught English thinker George Boole, who conceived of logic as the study of **laws of thought**, as he put it.

[2] We will define validity more precisely later.



Essentially, Boole used very basic ideas from elementary algebra to reduce logical reasoning—in particular logical reasoning involving **or**, **and**, **not**, etc., now called **Boolean connectives**—to simple arithmetic (like addition and multiplication) with the numbers 0 and 1. So called **Boolean reasoning** (Boolean algebra, logic, etc.) still provides the foundation for basically all of the digital and symbolic sciences, including circuits, probability theory, and many other fundamental areas.[3]

Along with the aforementioned increase in abstraction within mathematics came a need for even more expressive logical formalisms than Boolean logic. Mathematicians were starting to reason explicitly and precisely with complicated **functions**, **relations**, and **quantificational** statements, such as the famous ϵ/δ definition of limit, which is of the form:

For every $\epsilon > 0$, there is some $\delta > 0$ such that ...

To capture this more powerful reasoning, logicians concerned with **foundations of mathematics**—most significantly the German logician Gottlob Frege—



developed a system called **First Order Logic** (a.k.a. **Predicate Logic**) that could capture much of it. In this system (in modern notation) the above statement would be rendered:[4]

$$\forall\epsilon\,(\epsilon > 0 \rightarrow \exists\delta\,(\delta > 0 \wedge \dots))$$

This work on foundational issues in mathematics eventually had some unexpected consequences, when in the early 20th century work by mathematical logicians such as Alan Turing and Kurt Gödel





provided the ideas that made the first modern, all-purpose computers possible. They also proved some amazing results, whose significance for philosophy, mathematics, physics, computing, and cognition are still being debated today.[5]

[3] We will spend the first five weeks mastering this calculus.

[4] You should not yet be able to read this. We will spend the last four weeks studying First Order Logic. The successor to this course, Phil 151, is dedicated to a deeper understanding of First Order Logic, including the grand metalogical theorems characterizing its scope and limits.

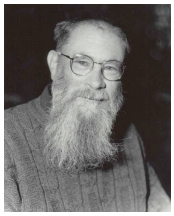
[5] Gödel's Incompleteness Theorems are the main topic in the successor to the successor to this course, Phil 152.

Following this mathematical period, some of the most significant advances in logic came out of philosophy, as well as early developments in cognitive science and artificial intelligence. In the latter, scholars such as Noam Chomsky, Barbara Partee, and John McCarthy (here at Stanford!) developed new logical tools to understand the **structure of language**, and **commonsense reasoning**:



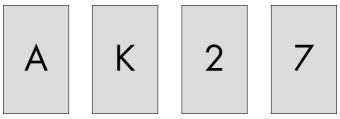


Philosophers such as Ruth Barcan Marcus and David Lewis developed new tools in **modal logic** to aid in understanding notions like **knowledge**, **causality**, **time**, **obligation**, **possibility**, and other philosophically important concepts:[6]

In fact, a persistent theme of the past 50 years has been the rich exchange of ideas and methods between those working in philosophical logic and AI practitioners, who are concerned with many common themes: rationality, planning, knowledge and belief, causality, etc.

Also in the 1960s logic became a favorite object of study for **psychologists** interested in the understanding human reasoning. Countless studies were done on ordinary subjects' performance on syllogistic reasoning tasks (like the patterns mentioned above). Psychologists were also interested in reasoning about **conditional statements**, as in the famous "Wason Card Selection Task." Suppose I show you the following cards, and tell you that each card has *a number on one side* and *a letter on the other side*:



Now consider the following hypothesis about these cards:

> **Hypothesis**: *If a card has an* ***even*** *number on one side, it has a* ***vowel*** *on the other.*

The question for you is now:

> **Question**: What is the minimum number of cards you will need to turn over to verify whether or not the hypothesis is true? In particular, which cards are necessary for this verification?

[6] We will study Modal Logic—including logics of knowledge and of time—in the two weeks between Boolean and First Order Logic.

In the original study by Wason—which has been replicated many times since and in many contexts and variations—the vast majority of subjects get this question wrong. (Do you know the right answer?) Explaining how ordinary people approach logical problem solving tasks like this, the syllogism, and others, has been one of the primary aims for work in psychology of reasoning. It has even spurred interest in developing so called **natural logics** that might diverge from "correct" mathematical reasoning, but which more closely mirror how people reason in ordinary contexts.

Since the 1960s, both fundamental understanding of, and applications, of logic have flourished in every possible direction. Logic has been studied using tools from algebra, topology, physics, statistics, and game theory, just to name a few. In the other direction, logic has found application in natural language understanding, electronics, cognitive psychology, programming language design and theory, metaphysics, and even Nobel Prize winning work in economics.

Attempting to summarize all of the current work and directions in logic is a hopeless task. Instead, the aim of this course is to provide you with the basic tools necessary to explore this landscape in more depth—in whatever direction interests you—and to give you some sense for why logic is such a powerful tool in all of these otherwise quite diverse contexts.

Chapter 2

Review: Sets, Relations, Functions

Elementary understanding of sets, relations, and functions will be assumed throughout the course, and will often come in handy for defining important concepts. Here we review some of the basics. This section can be skipped if you are already very familiar with these concepts.

2.1 Sets

We understand a **set** to be any collection of objects.[1] For instance the collection of all the books in Tanner Library forms a set. So does the collection of all the stars in the universe. Some sets are more abstract, such as the **natural numbers**:

$$\mathbb{N} = \{0, 1, 2, \dots\}$$

That is, $\mathbb{N}$ is the collection of numbers including 0, 1, 2, 3, and all the rest. Another familiar set is given by the **integers**, both positive and negative numbers:

$$\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$$

We could also come up with new sets such as the odd natural numbers:

$$Odd = \{1, 3, 5, 7, \dots\}$$

And of course there are many abstract sets that do not consist of numbers, such as the set of all isosceles triangles, or the set of arpeggios that appear in Beethoven's ninth symphony.

As a matter of notation, it is often useful to describe a set of things that satisfies some condition, using a variable x, e.g.:

$$\{x \mid x \text{ is a natural number}\}$$

is simply another way to refer to $\mathbb{N}$. This is read, "the set of x, such that x is a natural number."

[1] There is a mathematically precise *theory of sets*. However, the standard theory is founded on First Order Logic, which we will use sets to characterize! We may return to this puzzle at the end of the course.

If an object a is an element of a set A, we write:

$$a \in A$$

pronounced, "a is in A." So, for example, $17 \in \mathbb{N}$, but $-17 \notin \mathbb{N}$, while $-17 \in \mathbb{Z}$.

It is often useful to define new sets from old sets using special operations:

Definition 2.1.1. The **union** of two sets A and B, denoted $A \cup B$, is the set of everything that is an element of A **or** is an element of B:

$$A \cup B \quad \triangleq \quad \{x \mid x \in A \text{ or } x \in B\}$$

(The $\triangleq$ symbol means this equality holds by definition.)

Definition 2.1.2. The **intersection** of A and B, denoted $A \cap B$, is the set of everything that is an element of A **and** and an element of B:

$$A \cap B \quad \triangleq \quad \{x \mid x \in A \text{ and } x \in B\}$$

Definition 2.1.3. The **complement** of B in A, denoted $A - B$, is the set of everything that is an element of A but is **not** an element of B:

$$A - B \quad \triangleq \quad \{x \mid x \in A \text{ and } x \notin B\}$$

When we write $A = B$, we mean that the two sets A and B have the same elements. Thus, for instance, $\mathbb{Z} \cap \mathbb{N} = \mathbb{N}$, to take a boring example.

A very special set is the **empty set**, denoted $\varnothing$, which has no elements. That is, for any object a whatsoever we have $a \notin \varnothing$. Notice also, e.g., that $Odd \cap \{x : x \text{ is a planet } \} = \varnothing$, and $\varnothing \cup \varnothing = \varnothing$.

In addition to set equality, it is useful to have a notion of one set being contained in another:

Definition 2.1.4. A is contained in B, written $A \subseteq B$, if for every object $a \in A$, we have also $a \in B$. We will sometimes write $A \subsetneq B$ to mean that $A \subseteq B$ but $A \neq B$; that is, everything in A is in B, but there is something in B that is not in A. Likewise, we will write $A \supseteq B$ to mean that $B \subseteq A$.

Note as a consequence of Definition 2.1.4 that $A = B$ just in case both $A \subseteq B$ and $B \subseteq A$. We are now in position to prove (informally) a number of elementary facts about sets, the first of which we prove in excruciating detail:

Fact 2.1.1. For any sets A, B, and C, we have $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

Proof. To show two sets are equal, we must show every element of one is an element of the other. In particular we must show both $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$ and $A \cup (B \cap C) \supseteq (A \cup B) \cap (A \cup C)$.

For the "$\subseteq$" direction, suppose we take an arbitrary $a \in A \cup (B \cap C)$. Then either (i) $a \in A$, or (ii) $a \in B$ and $a \in C$ (since $a \in B \cap C$). In case (i), if $a \in A$ then certainly $a \in A \cup B$ and also $a \in A \cup C$, simply by virtue of the fact that $a \in A$. So $a \in (A \cup B) \cap (A \cup C)$. In case (ii), since

$a \in B$ we have $a \in A \cup B$, and since $a \in C$ we have $a \in A \cup C$; hence again $a \in (A \cup B) \cap (A \cup C)$. Since we have shown that an arbitrary a in $A \cup (B \cap C)$ must also be in $(A \cup B) \cap (A \cup C)$, this proves that $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$.

The "$\supseteq$" direction is similar. Suppose we take an arbitrary $a \in (A \cup B) \cap (A \cup C)$. Then we know both $a \in A \cup B$ and $a \in A \cup C$. We want to show $a \in A \cup (B \cap C)$. For this let us make a case distinction: either (i) $a \in A$, or (ii) $a \notin A$. In case (i), if $a \in A$ then certainly $a \in A \cup (B \cap C)$, so we are done. In case (ii), if $a \notin A$ then because $a \in A \cup B$, we can conclude that $a \in B$. And because $a \in A \cup C$, we can conclude $a \in C$. Hence $a \in B \cap C$, from which it follows that $A \cup (B \cap C)$. $\square$

Exercise 2.1.1. Show, by a similar argument, that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

On the other hand, some plausible looking equalities turn out not to hold. For instance, it is not the case that $A - (B \cup C) = (A - B) \cup (A - C)$. Can you come up with a counterexample?

Here is one: let A be $\mathbb{N}$, let B be Odd, and let $C = Even = \mathbb{N} - Odd$. Then of course $\mathbb{N} = B \cup C$, so $A - (B \cup C) = \mathbb{N} - (Odd \cup Even) = \varnothing$. However, $A - B = \mathbb{N} - Odd = Even$ and $A - C = \mathbb{N} - Even = Odd$; hence $(A - B) \cup (A - C) = Even \cup Odd = \mathbb{N}$. But $\varnothing \neq \mathbb{N}$, showing that the equality does not hold in general.

Exercise 2.1.2. Which of the following are true? If it holds, show why. If not, give a counterexample, i.e., specific sets for which it fails.

1. $A - (B \cup C) = (A - B) \cap (A - C)$
2. $A \cap (B - C) \subseteq (A - B) \cup (A - C)$
3. $A \cap (B - C) \supseteq (A - B) \cup (A - C)$

Another useful construction is the powerset operation:

Definition 2.1.5. The **powerset** of a set A, denoted $\wp(A)$, is the set of all subsets of A:

$$\wp(A) = \{B \mid B \subseteq A\}$$

For instance, the powerset of $\mathbb{N}$ is the set of all sets of natural numbers. Thus, $Odd \in \wp(\mathbb{N})$ and $Even \in \wp(\mathbb{N})$ as well. But also $\{2, 3, 4, 5\} \in \wp(\mathbb{N})$, for example.

Exercise 2.1.3. Which of the following hold? Again, give a counterexample or prove it:

1. $A \subseteq \wp(A)$
2. $A \in \wp(A)$
3. $\wp(A \cap B) = \wp(A) \cap \wp(B)$
4. $\wp(A \cup B) = \wp(A) \cup \wp(B)$

A final set construction we will need concerns taking **ordered pairs** of elements from sets A and B. An ordered pair, written $\langle a, b\rangle$, is just that: an object that has as its *first* element a and its *second* element b.[2] Given sets A and B, we can form the set of all the ordered pairs taking their first element from A and second element from B:

Definition 2.1.6. The **Cartesian product** of A and B, written $A \times B$, is the set of ordered pairs:

$$A \times B \quad \triangleq \quad \{\langle a, b\rangle \mid a \in A, b \in B\}$$

Thus, for example, $\mathbb{N} \times \mathbb{N} = \{\langle n, m\rangle \mid n, m \in \mathbb{N}\}$, the set of pairs of natural numbers. Or, $Odd \times Even = \{\langle n, m\rangle \mid n \text{ is odd and } m \text{ is even}\}$. And so on.

Exercise 2.1.4. Which of the following hold? Give a counterexample or prove it:

1. $A \times (B \cup C) = (A \times B) \cup (A \times C)$
2. $(A \times B) \cup (C \times D) = (A \cup C) \times (B \cup D)$
3. $(A \times B) \cap (C \times D) = (A \cap C) \times (B \cap D)$

When we take a product of a set with itself, $A \times A$, it will be convenient to write this as A^2. In fact, this notation can be generalized, so that A^n is the n-fold product of A with itself: $A \times \cdots \times A$, i.e., the set of n-tuples of elements of A: $A^n = \{\langle a_1, \ldots, a_n\rangle \mid a_1, \ldots, a_n \in A\}$. An n-tuple is just like a pair, except that it has n ordered elements instead of 2.

2.2 Relations

Definition 2.2.1. A **binary relation** R on a set A is any subset $R \subseteq A \times A$.

For example, there is a relation $Borders$ defined on the set of American states, and for instance $\langle California, Nevada\rangle$ in this set; that is, California stands in the $Borders$ relation with Nevada. A very familiar relation on the set of natural numbers is the "less than" relation $< \subseteq \mathbb{N} \times \mathbb{N}$:

$$< = \{\langle n, m\rangle \mid n, m \in \mathbb{N} \text{ and } n \text{ is less than } m\}$$

More standardly, in place of $\langle n, m\rangle \in <$, we simply write $n < m$; e.g., instead of $\langle 13, 15\rangle \in <$, we write $13 < 15$. For a relation $R \subseteq A \times A$ in general, we will typically write Rab instead of $\langle a, b\rangle \in R$. Some relations have interesting properties. We list a few of these:

Definition 2.2.2. A relation $R \subseteq A \times A$ is **reflexive** if Raa for all $a \in A$.

Definition 2.2.3. A relation $R \subseteq A \times A$ is **symmetric** if whenever Rab, also Rba for any $a, b \in A$.

Definition 2.2.4. A relation $R \subseteq A \times A$ is **transitive** if whenever Rab and Rbc, also Rac.

[2] There are ways of encoding ordered pairs as themselves *sets*, but we have no need to do that in this class. For us, they are special kinds of objects distinct from sets (which are just unordered collections).

Neither of our example relations above is reflexive. (But can you think of one that is?) $Borders$ is symmetric but not transitive: if a borders b, then certainly b borders a; while a might border b and b border c, it does not follow that a borders c. Meanwhile $<$ is transitive but not symmetric.

A useful way of depicting binary relations is by drawing pictures in which the (directed) edges represent instances of the relation. For instance, the following represents a relation between two objects i and j as follows:

$$i \longrightarrow j$$

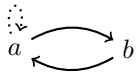
We can also depict somewhat more abstract notions, for instance, the idea of symmetry. We write this as follows:



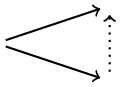
This is to be read as: any time you have an arrow from one point to another (the unbroken arrow), there will also be an arrow from the second point back to the first (the dotted arrow). In the same way we can also depict transitivity:



Informally, this means that if you can take two steps along the relation R from some point a to b and then from b to c, you can also take a single step from a to c. It is important to note that the diagrams do not have names for the particular points. This is because the statement is supposed to hold for *any* points you pick. For instance, if we have some point a related by R to b and b related to a again, if R is transitive we will also have a related to a. That would be a special case of the depiction above, where the first and third points are in fact the same. More concretely:



Exercise 2.2.1. A relation R is called **Euclidean** if whenever Rab and Rac, we have Rbc:



Show that if R is symmetric and transitive, then it is also Euclidean. You may use pictures (as above) if you think it helps. But you should give the argument in words.

Exercise 2.2.2. Suppose that R is reflexive and Euclidean. Show that R must also be symmetric.

We can also think of relations $R \subseteq A \times B$ relating two different sets A and B. For instance, the relation $CapitalOf$ relates the set of cities with the set of countries: e.g., $\langle Tokyo, Japan\rangle \in CapitalOf$. In fact, we will define functions to be special kinds of relations of this sort.

Beyond binary relations we can also speak of n-ary relations for arbitrary n. In particular any subset $R \subseteq A^n$ is an n-ary relation. For instance, an example of a ternary relation might be $Between$. We could say that $\langle a, b, c\rangle \in Between$ if a is between b and c. A mathematical example is being equivalent mod n. Thus, $EqMod(i, j, n)$ holds of i, j, and n just in case n divides $i - j$.

2.3 Functions

Functions are a familiar part of secondary math education. Intuitively they are "maps" from one set to another. For instance, a trigonometric function like cos maps one real number to another real number. Or, for a much simpler example, the "successor" function S maps a number n to $n+1$; it takes n as *input*, and returns $n+1$ as *output*. There are also functions that map from one **domain** A to a separate **range** B. For instance, the function height maps people to their heights (e.g., in inches). Extending the familiar mathematical notation, we could write $\mathsf{height}(\text{Lebron James}) = 80$.

In this course we will think of functions as special kinds of *relations*. We say a relation $R \subseteq A \times B$ is **functional** if, for every $a \in A$, there is **at most one** $b \in B$, such that Rab. This captures the idea that a function is supposed to map each element of its domain to a **unique** element of its range. We simply *identify* functions with their corresponding functional relations. Thus, e.g., the successor function $S \subseteq \mathbb{N} \times \mathbb{N}$ is given by $\{\langle m, n\rangle \mid n = m + 1\}$. But here, instead of Snm, we write the more familiar $S(n) = m$. More generally, when $F \subseteq A \times B$, with "domain" A and "range" B, we will write $F(a) = b$ instead of Fab (or even more properly, $\langle a, b\rangle \in F$). This notation is justified by the fact that specifying a picks out the corresponding b uniquely. We also follow ordinary notation and write $F : A \to B$ instead of $F \subseteq A \times B$.

The definition of a *functional relation* $F : A \to B$ above allows that there may be $a \in A$ for which there is no $b \in B$ such that $F(a) = b$. If we require that every $a \in A$ is mapped to some $b \in B$, then we say that F is a **total function**. Otherwise, it is a **partial function**.

Some other special properties that functions may enjoy include:

Definition 2.3.1. A function $F : A \to B$ is **one-to-one**, or **injective**, if whenever $F(a) = F(b)$ we have $a = b$. That is, F maps each distinct element of A to a distinct element of B.

Definition 2.3.2. A function $F : A \to B$ is **onto**, or **surjective**, if for every $b \in B$ there is some $a \in A$ such that $F(a) = b$. I.e., everything in the range is mapped to by something in the domain.

Clearly, the successor function S defined on $\mathbb{N}$ is one-to-one but not onto. There is no natural number n such that $S(n) = 0$, for example. The function height is not one-to-one, since there are many people that have the same height. It is probably not onto either.

Exercise 2.3.1. Given an example of a function that is onto but not one-to-one.
(This should be very easy, once you understand the concepts. You get to choose the domain/range!)

Exercise 2.3.2. Suppose we have a function $F : \wp(A) \to A$ s.t. $F(B) \in B$ for any nonempty $B \subseteq A$.

1. Is F necessarily one-to-one?
2. Is F necessarily onto?

Show why or why not, again by informal proof or counterexample, for both questions.

We will also be considering more general n-ary functions that take multiple arguments, of the form $F : A_1 \times \cdots \times A_n \to B$. These are also to be understood as special kinds of functional relations $F \subseteq A_1 \times \cdots \times A_n \times B$, where for each n-tuple $(a_1, \ldots, a_n)$, with $a_1 \in A_1, \ldots, a_n \in A_n$, there is at

most one $b \in B$ for which the $n+1$-tuple $(a_1, \ldots, a_n, b) \in F$. This is again what justifies the notation $F(a_1, \ldots, a_n) = b$. Examples of n-ary functions with $n > 1$ are plentiful. A very simple example is addition. That is, $+$ takes two arguments and returns their sum. But again, instead of writing $+(n, m)$, we will write this in a more customary way, $n + m$.

Chapter 3

Boolean Logic

Imagine you are searching for some destination and you come to a fork in the road. You can either go left or right, and you know your destination is one way or the other. Then you come to learn that the left road will definitely *not* take you there. What can you conclude? The answer is patently obvious, and even some famous dogs have evidently made the reasonable inference:[1]

$$\frac{\text{It is to the left or to the right.} \qquad \text{It is not to the left.}}{\text{It is to the right.}}$$

It is also apparent that what makes this simple inference work is simply the presence of some key words, in particular, *or* and *not*. That is, the following pattern is always a good pattern, in the sense that, whenever the premises are true, the conclusion is guaranteed to be true as well:

$$\frac{A \text{ or } B \qquad \text{not } B}{A}$$

Words like *or* and *not* are the natural language counterparts to so called **Boolean connectives**, which will be our object of study in this part of the course.[2] Though patterns like that above may appear trivial, Boolean logic becomes surprisingly complex very quickly. In fact, relatively simple questions that can be posed in this system turn out to be equivalent to fundamental outstanding open problems in mathematics, of significant practical importance.[3] We will therefore proceed with due diligence and assiduity.

3.1 The Boolean Connectives

In order to learn logic we are going to have to learn some new languages. It is a bit like learning a foreign language, except that the rules of the game are completely and unambiguously specified. In particular we must learn the rules for how we can build meaningful expressions. Then we must understand how these expressions are to be interpreted. In this section we focus on the first task.

[1] See the ancient story of Chrysippus's Dog.
[2] We will return to the relation between Boolean logic and these natural language expressions in §4.1.
[3] Including a million dollar prize for a solution. See §4.3 for more.

The **language of Boolean logic** is given first by an infinite list of **propositional atoms**:

$$A_1, A_2, A_3, \ldots$$

We will sometimes also use other uppercase roman letters like B, C, ..., P, Q, ..., etc. Think of these symbols as standing for **atomic propositions**, of the sort that can either be true or false. For instance:

"Rome is the capital of Italy."	"UMP synthase is an enzyme."
"A walrus is a type of insect."	"Grover Cleveland founded Cleveland."
"The switch is on."	"The switch is off."

And so on. The interest in Boolean logic comes when we combine these propositional atoms together with the Boolean connectives. Our first Boolean connective is $\wedge$, which loosely corresponds with natural language 'and'. Intuitively, if A stands for "Rome is the capital of Italy" and B stands for "Grover Cleveland founded Cleveland," then $A \wedge B$ stands for "Rome is the capital of Italy, **and** Grover Cleveland founded Cleveland."

We will also have a connective $\neg$ that **negates** a proposition, corresponding roughly to English 'not'. So, e.g., $\neg B$ will roughly mean "It is **not** the case that Grover Cleveland founded Cleveland." (This happens to be true.)

Finally, we introduce a third connective $\vee$, corresponding roughly to 'or', so that $A \vee B$ can be read as "Either Rome is the capital of Italy **or** Grover Cleveland founded Cleveland."

Once we have built up complex formulas using these connectives, we can continue, building even more complex formulas. For instance, we can write $(A \wedge B) \vee (\neg A \wedge \neg B)$, meaning roughly, "Either Rome is the capital of Italy and Grover Cleveland founded Cleveland, or neither is true."

Putting all of this together we can define an official language $\mathcal{L}^{bool}$ of Boolean logic.

Definition 3.1.1. Given an infinite set Prop of propositional atoms, $\mathcal{L}^{bool}$ is defined to be the following set of **well-formed formulas** (often called *wff*s):

1. Every propositional atom $A \in \mathsf{Prop}$ is a well-formed formula of $\mathcal{L}^{bool}$.
2. If φ and ψ are well-formed formulas of $\mathcal{L}^{bool}$, then so are:
   (a) $(\varphi \wedge \psi)$
   (b) $\neg\varphi$
   (c) $(\varphi \vee \psi)$
3. Nothing else is a well-formed formula of $\mathcal{L}^{bool}$.

This allows us to write any number of complex wffs. These are all examples off Boolean wffs:

$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$ $\neg\neg\neg\neg A_{22}$

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

And so on. Some of these are more difficult to give an intuitive gloss on, but they are all well-formed according to our simple little grammar in Definition 3.1.1. For instance, to see why the first formula is a wff, we can reason as follows:

- A_1, A_2, and A_{16} are all wffs by clause 1 of Definition 3.1.1, all being in Prop.
- $\neg A_2$ and $\neg A_{16}$ are wffs by clause 2(b), given that A_2 and A_{16} are both wffs.
- $(\neg A_2 \vee A_1)$ is a wff by clause 2(c), given that $\neg A_2$ and A_1 both are.
- $(\neg A_{16} \wedge (\neg A_2 \vee A_1)))$ is a wff by clause 2(a), again given the above.
- Finally, $(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$ if a wff, again by clause 2(c).

We can conclude the others are wffs by similar reasoning. As far as syntax goes, it is just this simple!

Aside on color scheme: A logical language like $\mathcal{L}^{bool}$ is a special kind of **symbolic system**, that has its own rules and interpretation (which we will give shortly), and which we want to distinguish clearly from our ordinary natural language symbols and marks that make up the bulk of these notes. Therefore, every logical symbol or formula in any of our logical languages will be coded in blue, to make it patently clear that we are talking about this symbolic system that is our object of study.

If you have ever worked with programming languages, many editors will also highlight your text in different colors, e.g., by making commented text green. Think of the color scheme this way: as a cognitive aid to make it totally unambiguous, for instance, that we are talking about a logical symbol A (in this case, a propositional atom), and not the uppercase English letter.[4]

Aside on parentheses: Though strictly speaking our logical language requires parentheses around every (sub)formula of the form $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$, we will sometimes leave these off if no confusion arises. In particular, it will be convenient to leave off the outermost parentheses of a formula, e.g., $A \vee (B \wedge C)$. However, when in doubt, err on the side of caution and include parentheses! This is important. For example, the following are two different wffs: $(A \wedge B) \vee C$ and $A \wedge (B \vee C)$, so you definitely do not want to write simply $A \wedge B \vee C$. We will see why in the next sections.

3.2 Truth Tables and Semantics

In the previous section we gave some intuitive glosses on the connectives $\wedge$, $\vee$, and $\neg$, which we will henceforth refer to as the **conjunction**, **disjunction**, and **negation** symbols. We said they correspond roughly to English 'and', 'or', and 'not'. In this section we will make their interpretation mathematically precise. Consider a formula like $\neg A$. The propositional atom A could be either true

[4] You may notice that nothing in §2 was in blue. This is because our ordinary reasoning about sets, relations, and functions was not done in a *formal language*. This was in a semi-formal "mathematical" extension of ordinary English.

or false. If it is true, then we will want $\neg A$ to be false; and if it is false, then we will want $\neg A$ to be true. That is, in general $\neg\varphi$ should *reverse* the truth-value of φ.

We can capture this basic idea with a **truth table**:

φ	$\neg\varphi$
T	F
F	T

This says that an arbitary wff φ might take on one of two truth-values, T or F. And whichever it is, $\neg\varphi$ will take on the opposite truth-value.

And how about the truth table for the conjunction symbol $\wedge$? Since $\wedge$ connects two wffs, we will need a slightly larger table. A moment's reflection suggests the following:

φ	ψ	$(\varphi \wedge \psi)$
T	T	T
T	F	F
F	T	F
F	F	F

A formula like $(\varphi \wedge \psi)$ will *only* be true when *both* φ and ψ are true. In all other cases it is false.

Finally, what about the disjunction symbol? Its truth table looks as follows:

φ	ψ	$(\varphi \vee \psi)$
T	T	T
T	F	T
F	T	T
F	F	F

It is only false when both φ and ψ are false, true as long as at least one of them is true.

Truth tables allow us to determine, in a systematic manner, what the truth-value of some *complex* wff is, given truth-values of all the propositional atoms that occur in it. Let us walk through an example. Suppose we have determined that A_1 and A_2 are both true, while A_{16} is false. What is the truth-value of the formula we built above, $(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$? The truth tables defining $\wedge$, $\vee$, and $\neg$ above allow us to determine this:

- Because A_2 has value T, it follows that $\neg A_2$ has value F.
- Because A_{16} has value F, $\neg A_{16}$ has value T.
- Because $\neg A_2$ has value F and A_1 has value T, $(\neg A_2 \vee A_1)$ has value T.
- Because $\neg A_{16}$ has value T and $(\neg A_2 \vee A_1)$ has value T, $(\neg A_{16} \wedge (\neg A_2 \vee A_1)))$ has value T.
- Finally, since both disjuncts have value T, $(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$ itself has value T.

Indeed, we can do this methodically by following how we constructed the formula (recall the previous section). First, simply copy over all the truth values from the left-hand-side to the right-hand-side:

A_1	A_2	A_{16}	$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$
T	T	F	T F T T

Then apply the truth table rules for negation:

A_1	A_2	A_{16}	$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$
T	T	F	T **T** F **F** T T

Then for the innermost disjunction:

A_1	A_2	A_{16}	$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$
T	T	F	T T F F T **T** T

Then the conjunction:

A_1	A_2	A_{16}	$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$
T	T	F	T T F **T** F T T T

And finally for the outermost disjunction:

A_1	A_2	A_{16}	$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$
T	T	F	T **T** T F T F T T T

This shows that the overall formula in this case receives value T.

We just determined the truth-value of this formula for one particular choice of truth-values for the propositional atoms that appear in it. But we could just as well do this for *all possible* combinations of truth-values for the atoms. In general, if we have n distinct atoms appearing in a Boolean formula, there will be 2^n possible combinations of truth-values. So in this case we have to consider $2^3 = 8$:

A_1	A_2	A_{16}	$(A_1 \vee (\neg A_{16} \wedge (\neg A_2 \vee A_1)))$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	T

Going through the same method as above for line 2 of this table, we fill in all the rest of the rows, specifying what truth-value the overall formula obtains, given every possible assignment of truth-values to the propositional atoms. You should go through one or two of these rows yourself and make sure you understand why the table looks as it does.

Exercise 3.2.1. Write out truth-tables for the following formulas, listing only the final truth value of the whole formula, as in the example above.

1. $(B \wedge (B \wedge (C \wedge A)))$
2. $\neg(((A \vee \neg A) \vee (B \vee \neg C)) \wedge ((A \wedge \neg A) \vee (\neg B \wedge C)))$

3.3 Truth Functions

Intuitively we can think of a row of the truth table as specifying one particular way a situation might be: different **atomic facts** may be true or false. Then the truth-table method allows us to determine exactly the truth-values of complex propositions on the basis of the atomic assignments, together with the "meanings" of the **truth-functional** connectives, $\wedge$, $\vee$, and $\neg$.

It will be useful to make this idea a bit more precise by using the notion of a **valuation**.

Definition 3.3.1. A **valuation** is a function $v : \mathsf{Prop} \to \{\mathsf{T}, \mathsf{F}\}$, assigning T or F to every propositional atom in Prop. We can then say that a valuation v makes an arbitrary formula $\varphi \in \mathcal{L}^{bool}$ true, written $v \vDash \varphi$, where this is defined as follows (we write $v \nvDash \varphi$ if $v \vDash \varphi$ fails to hold, in which case we say v makes φ false):

1. $v \vDash A$ if and only if $v(A) = \mathsf{T}$, for $A \in \mathsf{Prop}$;
2. $v \vDash \neg\varphi$ if and only if $v \nvDash \varphi$;
3. $v \vDash \varphi \wedge \psi$ if and only if both $v \vDash \varphi$ and $v \vDash \psi$;
4. $v \vDash \varphi \vee \psi$ if and only if either $v \vDash \varphi$ or $v \vDash \psi$.

That is, a valuation maps each atomic proposition to a truth-value, and is canonically extended to determine truth/falsity of *all* formulas in a way that matches our truth-table definitions of the connectives. The basic idea is that we can build up a formula piece-by-piece, as in the earlier definition of our language $\mathcal{L}^{bool}$. The **semantics** of the language follows this construction all along the way. At the basic step we assign truth-values to atomic propositions; then these truth-value assignments extend to truth-values for more complex formulas in a standardized ("canonical") way.

For example, suppose $\mathsf{Prop} = \{A_1, A_2, A_3, \dots\}$ and we consider a valuation $v : \mathsf{Prop} \to \{\mathsf{T}, \mathsf{F}\}$ that sends every atomic proposition to F, i.e., $v(A_i) = \mathsf{F}$ for all $i > 0$. Then we can determine the truth-assignments for more complex formulas using the definition above (analogous to our use of truth tables). Then consider $(A_1 \wedge A_2) \vee (\neg A_1 \wedge \neg A_2)$.

- Because $v(A_1) = v(A_2) = \mathsf{F}$, we also have $v \nvDash A_1$ and $v \nvDash A_2$ (by Definition 3.3.1, part 1).
- Then by Definition 3.3.1, part 3, $v \nvDash A_1 \wedge A_2$.
- By Definition 3.3.1, part 2, we have both $v \vDash \neg A_1$ and $v \vDash \neg A_2$.
- Hence by Definition 3.3.1, part 3 again, we have $v \vDash \neg A_1 \wedge \neg A_2$.
- Finally, by 3.3.1, part 4, we have $v \vDash (A_1 \wedge A_2) \vee (\neg A_1 \wedge \neg A_2)$.

The reasoning here is essentially the same as what we did above with truth tables. But the ability to refer succinctly and abstractly to truth-functional **valuations** will prove useful.

3.4 Metalogical Notions

Equipped with the technology we have developed so far, we can already formulate some of the most important logical notions. Many of these arose in the context of theorizing about **logical argumentation** and **deductive reasoning**, and we will correspondingly start with this perspective in mind. But later we will see that these technical notions are useful for many other purposes.

Some logical formulas $\varphi \in \mathcal{L}^{bool}$ are guaranteed to be true, no matter what truth-value we assign to the propositional atoms. A classic example is $A \vee \neg A$, sometimes referred to grandly as the **law of excluded middle**. A simple truth table shows that this formula cannot be made false by any assignment of truth-value to A:

A	$A \vee \neg A$
T	T
F	T

Such formulas are called **logically valid**, or just **valid**:

Definition 3.4.1. A formula φ is **valid** if, for every valuation $v : \mathsf{Prop} \to \{\mathsf{T}, \mathsf{F}\}$, we have $v \vDash \varphi$; or equivalently, if every row of the truth table for φ receives value T.

Thus, in order to show that a formula φ is valid, it suffices to write out the truth table for φ and demonstrate that every row is assigned T.

Exercise 3.4.1. Which of the following formulas are valid? Show why or why not.

1. $(A \wedge \neg B) \vee (B \wedge \neg A)$
2. $(A \wedge \neg B) \vee \Big((B \wedge \neg A) \vee \big((A \wedge B) \vee (\neg A \wedge \neg B)\big)\Big)$
3. $\neg(A \wedge \neg A)$
4. $\neg A \vee \neg\neg A$
5. $\big(A \wedge (B \wedge C)\big) \vee \big(\neg A \vee (\neg B \vee \neg C)\big)$
6. $(A \vee B) \vee (\neg B \vee \neg A)$

In some sense the opposite of a valid formula—a formula that is always true—is a **contradictory** formula—a formula that is never true. A simple example is $A \wedge \neg A$.

Definition 3.4.2. A **contradiction** $\varphi \in \mathcal{L}^{bool}$ is a formula for which $v \nvDash \varphi$ for every valuation v; alternatively, for which every row of the truth table for φ is F. These formulas are also sometimes called **unsatisfiable**, because there is no truth assignment that "satisfies" them.

Exercise 3.4.2.

1. Show that φ is a contradiction just in case $\neg\varphi$ is valid. That is, show that if φ is a contradiction, then $\neg\varphi$ must be valid; and vice versa, that if $\neg\varphi$ is valid, then φ must be a contradiction.
2. Show that if φ is valid (/ a contradiction), then $\neg\neg\varphi$ is also valid (/ a contradiction).

Valid and contradictory formulas are special, in that their truth-values do not depend at all on the truth-values of the atomic propositions they contain. In the context of an argument, it is clear why these categories might be of interest. If I claim some simple fact about the world is true—e.g., that Grover Cleveland founded Cleveland (which is in fact false)—you may question this. However, if the claim I make is **valid**—e.g., that either Grover Cleveland founded Cleveland or he didn't—then this is a lot harder to challenge. On the opposite end, if in the course of an argument you force me to admit some **contradiction** is true, that looks like bad news for me. We will come back to a formal understanding of these sorts of argumentative strategies in §3.8.

Many formulas are of course neither valid nor contradictory. One important final category captures those formulas that are not contradictory, or not unsatisfiable—i.e., **satisfiable**—meaning they are true under some valuations:

Definition 3.4.3. A formula φ is **satisfiable** if there is at least one valuation v such that $v \vDash \varphi$; that is, at least one row of the truth table returns T.

Exercise 3.4.3. Which of the formulas from Ex. 3.4.1 are satisfiable?

Exercise 3.4.4. Suppose $\neg\varphi$ is satisfiable. Is it possible that φ is valid? What about $\neg\varphi$?

These notions so far concern categories of single formulas. We can also lift these to *sets* of formulas. This is particularly useful for satisfiability

Definition 3.4.4. A set $\Gamma \subseteq \mathcal{L}^{bool}$ of formulas is **satisfiable** if there is at least one valuation v such that $v \vDash \varphi$ for every formula $\varphi \in \Gamma$. (Note it must be the same valuation for every $\varphi \in \Gamma$.)

Otherwise the set Γ is **unsatisfiable**.

For a trivial example of a satisfiable set, consider $\Gamma = \{A_1, A_1 \wedge A_2, A_1 \wedge A_2 \wedge A_3, \dots\}$. The valuation sending each A_i to T will satisfy Γ. For an example of an **un**satisfiable set, consider $\Gamma = \{A_1 \vee A_2 \vee A_3, \neg A_1, \neg A_2, \neg A_3\}$.

Now that we are talking about sets of formulas, one of the most important logical notions is that of **entailment**, which captures the idea of one formula "following logically" from some others:

Definition 3.4.5. Suppose $\Gamma \subseteq \mathcal{L}^{bool}$ is a set of formulas and $\varphi \in \mathcal{L}^{bool}$ is a formula. We say that Γ **logically entails** (or simply **entails**) φ, if for every valuation v such that $v \vDash \gamma$ for every $\gamma \in \Gamma$, we also have $v \vDash \varphi$. When this holds we write $\Gamma \Vdash \varphi$ (notice the difference with $\vDash$).

Informally, we say that Γ entails φ, i.e., that $\Gamma \Vdash \varphi$, if any way of making all the formulas in Γ true will also ensure that φ will be true.

For instance, returning to the example we began this chapter with, suppose $\Gamma = \{A \vee B, \neg B\}$. Then $\Gamma \Vdash A$. This is because any valuation that makes both $A \vee B$ and $\neg B$ true will necessarily make A true as well. Indeed, we can see this using a slight extension of our truth table method:

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

Putting the variables that occur in the relevant formulas before the first doube-line (||), then the premises (elements of Γ) in the middle, and the conclusion (φ) on the right, to determine whether $\Gamma \Vdash \varphi$, we merely need to look at those lines of the truth table on which all the premises get value T. In this case there is only one, namely the second line. Because on this line A also receives T, this shows the entailment indeed holds:

A	B	$A \vee B$	$\neg B$	A
T	T	T	F	T
T	F	**T**	**T**	**T**
F	T	T	F	F
F	F	F	T	F

Exercise 3.4.5. Which of the following entailments hold? Show using a truth table.

1. $\{A, \neg(A \wedge B)\} \Vdash \neg B$
2. $\{A \vee B \vee C, A \vee \neg B, B \vee \neg C, C \vee \neg A\} \Vdash A$
3. $\{A \wedge \neg B, \neg C \vee \neg A\} \Vdash B$
4. $\{A, B, \neg C\} \Vdash \neg(A \vee C)$
5. $\{\neg A, A\} \Vdash B$

A special case of entailment is when the set Γ is just a single formula $\Gamma = \{\gamma\}$. Then instead of writing $\{\gamma\} \Vdash \varphi$, we will drop the brackets and simply write $\gamma \Vdash \varphi$. Another special case is when $\Gamma = \varnothing$. Instead of writing $\varnothing \Vdash \varphi$, we will simply write $\Vdash \varphi$. Notice that $\Vdash \varphi$ is therefore another shorthand for saying that φ is *valid*.

A special situation is when we have entailments in both direction: $\gamma \Vdash \varphi$ and $\varphi \Vdash \gamma$. We then say that the formulas γ and φ are **logically equivalent**.

Definition 3.4.6. Formulas γ and φ are **logically equivalent** if both $\gamma \Vdash \varphi$ and $\varphi \Vdash \gamma$; that is, the truth tables for γ and φ are identical. We write this $\gamma \dashv\Vdash \varphi$.

Exercise 3.4.6. Which of the following equivalences hold? Show using truth tables.

1. $(A \vee B) \vee C \dashv\Vdash A \vee (B \vee C)$
2. $(A \wedge B) \vee C \dashv\Vdash A \wedge (B \vee C)$
3. $(A \wedge B) \wedge C \dashv\Vdash A \wedge (B \wedge C)$
4. $\neg(A \wedge B) \dashv\Vdash \neg A \wedge \neg B$
5. $\neg(A \wedge B) \dashv\Vdash \neg A \vee \neg B$
6. $\neg(A \vee B) \dashv\Vdash \neg A \wedge \neg B$
7. $A \wedge (B \vee C) \dashv\Vdash (A \wedge B) \vee (A \wedge C)$
8. $A \vee (B \wedge C) \dashv\Vdash (A \vee B) \wedge (A \vee C)$

Intuitively, if two logical formulas are logically equivalent, then they essentially "mean the same thing." They are true in all of the same situations (that is, on all the same assignments of truth and falsity to the atomic propositions).

Exercise 3.4.7. Given a wff φ, how many other formulas are logically equivalent to φ?

3.5 The Material Conditional

So far we have been working with a logical language that uses only three Boolean connectives: $\wedge$, $\vee$, and $\neg$. In this section we will introduce a fourth: the **(material) conditional** $\rightarrow$. Its truth table is as follows:

φ	ψ	$(\varphi \rightarrow \psi)$
T	T	T
T	F	F
F	T	T
F	F	T

That is, $(\varphi \rightarrow \psi)$ is false only when φ (the *antecedent* of the conditional) is true, while ψ (the *consequent* of the conditional) is false. Otherwise it is true. A way of thinking about this is that $(\varphi \rightarrow \psi)$ "claims" that ψ is true *on the condition* that φ is true; in all other cases, there is no claim (and thus the formula overall is simply true).

Extending Definition 3.3.1 of a **valuation**, we would include a clause 5 for the extended language as follows:

5. $v \vDash \varphi \rightarrow \psi$ if and only if either $v \nvDash \varphi$ or $v \vDash \psi$.

corresponding to the truth table above.

Warning: Often people will informally read $(\varphi \rightarrow \psi)$ as "If φ, then ψ." While this can sometimes be helpful for intuition, it can be misleading. As an exercise, try to think of some conditional statements in English, and determine whether the truth table above captures the right intuitions about when the sentence is true. We will return to this issue in §4.1. See especially Ex. 4.1.1.

The material conditional is useful in many places. Indeed, we will be using it extensively in the rest of this course. It also captures nicely various types of mathematical reasoning. To take just a simple example, recall Ex. 3.4.2 above. There we asked you to prove two *conditional statements* of the form: if one statement holds, then so does another. The way to show this is to *suppose* the first statement holds, and then show using that fact that the second statement must also hold. We do not need to show anything about the case when the first statement does not hold. Again, we will come back to these *deductive reasoning principles* in Section 3.8.

Exercise 3.5.1. Which of the following are valid? Show why or why not.

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow B) \rightarrow A$
3. $A \rightarrow (A \rightarrow B)$
4. $A \rightarrow (\neg A \rightarrow B)$
5. $A \rightarrow (A \vee C)$
6. $A \vee (A \rightarrow B)$
7. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
8. $(A \rightarrow (B \vee C)) \rightarrow ((A \rightarrow B) \vee (A \rightarrow C))$

Exercise 3.5.2. Which of the following hold? Show why or why not.

1. $(A \to B) \dashv\Vdash (\neg B \to \neg A)$
2. $(A \wedge B) \to A \Vdash (B \to A)$
3. $(A \vee B) \to A \Vdash (B \to A)$
4. $A \to B, A \Vdash B$
5. $A \to B, B \Vdash A$
6. $A \to B, B \to C \Vdash A \to C$
7. $(\neg A \vee B) \dashv\Vdash A \to B$
8. $A \to (B \wedge \neg B) \dashv\Vdash \neg A$
9. $(B \wedge \neg B) \to A, B \Vdash A$
10. $(B \vee \neg B) \to A \dashv\Vdash A$

Another reason the material conditional is important is that, in a certain precise sense, it *internalizes* the notion of logical entailment. Indeed, we have the following fact, which is sometimes given the honorific title, **The Deduction Theorem**:

Fact 3.5.1. $\{\varphi_1, \ldots, \varphi_n, \varphi_{n+1}\} \Vdash \psi$, if and only if $\{\varphi_1, \ldots, \varphi_n\} \Vdash \varphi_{n+1} \to \psi$.

Or in the simplest case (when $n = 0$), we have $\varphi \Vdash \psi$ if and only if $\Vdash \varphi \to \psi$.

Proof Sketch. We prove just the simplest case of $n = 0$ and leave the general case as an exercise. Suppose $\varphi \Vdash \psi$. We want to show $\Vdash \varphi \to \psi$. By the assumption we know that for any valuation v such that $v \Vdash \varphi$, we also have $v \Vdash \psi$. But that is just to say, any row of the truth table for φ and ψ that assigns to φ value T (if such there be) will also assign ψ value T. That is, in the truth table for these formulas, there is no row sending φ to T but ψ to F. For instance, it might look schematically something like this (with all atomic propositions appearing in φ in ψ to the left of the double bar):

$\ldots$	φ	ψ
	$\vdots$	$\vdots$
$\ldots$	F	T
	$\vdots$	$\vdots$
$\ldots$	T	T
	$\vdots$	$\vdots$
$\ldots$	F	F
	$\vdots$	$\vdots$
$\ldots$	T	T
	$\vdots$	$\vdots$

But that is enough to guarantee that $\varphi \to \psi$ is valid. The truth table for $\to$ guarantees that this formula receives value T on all such rows.

...	φ	ψ	$\varphi \to \psi$
	⋮	⋮	⋮
...	F	T	T
	⋮	⋮	⋮
...	T	T	T
	⋮	⋮	⋮
...	F	F	T
	⋮	⋮	⋮
...	T	T	T
	⋮	⋮	⋮

The other direction is just the reverse of this reasoning. If $\varphi \to \psi$ is valid, then that means any valuation sending φ to T will also send ψ to T. That is just to say $\varphi \Vdash \psi$. □

Note also that as a consequence of Fact 3.5.1, we have $\{\varphi_1, \ldots, \varphi_n, \varphi_{n+1}\} \Vdash \psi$, if and only if $\Vdash \varphi_1 \to (\varphi_2 \to \ldots (\varphi_n \to (\varphi_{n+1} \to \psi)) \ldots)$, by simply appealing to the fact $n + 1$ times.

3.6 More Connectives

In the previous section we introduced a new connective $\to$ on top of the three connectives we already had. In a sense this was not necessary. As you in fact showed in Ex. 3.5.2, part 7, each time we want to be able to express something of the form $(\varphi \to \psi)$, we could simply write instead $(\neg\varphi \vee \psi)$. The two expressions have the same truth tables—they are logically equivalent. In this sense, $\to$ did not really add any expressive capability that we did not have before.

What about the three connectives we defined at the start? Could we do with even fewer? In a certain sense, we could. Suppose we had begun with only $\vee$ and $\neg$, without $\wedge$. Then whenever we wanted to write $\varphi \wedge \psi$ we could instead write $\neg(\neg\varphi \vee \neg\psi)$; for these are logically equivalent:

φ	ψ	$\varphi \wedge \psi$	$\neg(\neg\varphi \vee \neg\psi)$
T	T	T	T
T	F	F	F
F	T	F	F
F	F	F	F

no matter what φ and ψ are.[5] Likewise, if we wanted to start out with only $\wedge$ and $\neg$, and leave out $\vee$, we could have done this, because every formula of the form $\varphi \vee \psi$ is equivalent to $\neg(\neg\varphi \wedge \neg\psi)$, as you are invited to verify.

Another useful connective that is in a sense superfluous, but which will be useful at various points, is the **biconditional** symbol $\leftrightarrow$. Its truth table looks as follows:

[5]The careful reader will note that, here and elsewhere in these notes, depending on what φ and ψ are, some of these rows of the truth table may be impossible. For instance, if φ is a tautology, then there will be no row on which it receives F. This is indeed a slight abuse of notation, but relays the point we want to make. Talking about this in terms of truth functions is more precise, but truth tables are easier to picture.

φ	ψ	$(\varphi \leftrightarrow \psi)$
T	T	T
T	F	F
F	T	F
F	F	T

So that $(\varphi \leftrightarrow \psi)$ essentially says that φ and ψ are truth-functionally *equivalent*: this formula is true just in case φ and ψ have the *same* truth value.

Exercise 3.6.1. Analogous to the examples above, what is a formula using only $\vee$, $\wedge$, $\neg$, and $\rightarrow$ that is equivalent to $(\varphi \leftrightarrow \psi)$?

Better yet, what is a formula using only $\vee$ and $\neg$ that is equivalent to $(\varphi \leftrightarrow \psi)$

Another useful connective—though it is a bit degenerate as a connective—is $\bot$, sometimes called "absurdity" or "contradiction" or "falsum." The connective $\bot$ actually takes no arguments, it is a wff by itself. And its truth table is very simple: it always receives value F! Thus, if we were going to extend Definition 3.3.1 of a valuation to include $\bot$ in the language, we would have a clause:

6. $v \not\models \bot$

What is a formula in the language without $\bot$ that would be equivalent to it? There are many of them. In fact, any contradiction will do. For instance, choose an arbitrary propositional atom A and consider $A \wedge \neg A$. These two formulas have the same truth tables:

A	$(A \wedge \neg A)$	$\bot$
T	F	F
F	F	F

Thus, while it will at various points be convenient to have $\bot$ as part of our language—and we will use it sometimes—adding it does not increase our expressivity.

We could go on defining new connectives, and asking whether these new connectives could already be defined using the old connectives we already have. However, it might be useful to know that, at some point, we have *all the connectives we could want*. That is, any further connectives we add are just for convenience (or perhaps for succinctness); they do not make our language more expressive.

In this direction, let us introduce a notion of *arity* for a connective. We say that $\neg$ is **unary**, because it attaches to only a single formula. The other connectives we have defined so far are **binary**, in that they connect *two* formulas. Falsum $\bot$ is a 0-ary connective. Moreover, we could just as well add, e.g., ternary connectives. For example, we could introduce a (not obviously useful) connective $*$ such that $*(\varphi, \psi, \chi)$ has the following truth table:

φ	ψ	χ	$*(\varphi, \psi, \chi)$
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	T

Exercise 3.6.2. Write a formula using only $\wedge$, $\vee$, and $\neg$ that is logically equivalent to $*(\varphi, \psi, \chi)$.

In general, how can we ever know there will not be some connective of arity, say, 613 that we cannot define using one of our primitive connectives? Toward that end we define:

Definition 3.6.1. A set $\mathcal{C}$ of connectives is **truth-functionally complete** if every other n-ary connective (for any n) can be defined out of connectives in $\mathcal{C}$.

Fact 3.6.1. The set $\{\wedge, \vee, \neg\}$ is truth-functionally complete.

Proof. Suppose we have some n-ary connective $\$$. We want to demonstrate that we can define $\$(\varphi_1, \ldots, \varphi_n)$. We do this simply as follows. Write out the truth table for $\$(\varphi_1, \ldots, \varphi_n)$, and suppose that $A_1, \ldots, A_m$ are the propositional atoms that occur in $\varphi_1, \ldots, \varphi_n$. Our formula, using only $\wedge$,$\vee$, and $\neg$, will be a large disjunction of conjunctions. In particular, for each row of the truth table on which $\$(\varphi_1, \ldots, \varphi_n)$ receives T we include a conjunction

$$l_1 \wedge \cdots \wedge l_m$$

where for $i \leqslant m$: $l_i = A_i$ if A_i receives value T on that row, and $l_i = \neg A_i$ if A_i receives value F on that row. Disjoining all of these conjunctions together gives us our equivalent formula: it is true under exactly the same valuations as the original formula $\$(\varphi_1, \ldots, \varphi_n)$. □

As an illustration, returning to the example above of the ternary connective $*$, suppose we take in particular $*(A, B, C)$ for some atoms A, B, and C. Then again, our truth table looks like this:

A	B	C	$*(A, B, C)$
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	T

Our equivalent formula in this case will include a conjunct for each row that receives value T:

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

You should note from your answer to Ex. 3.6.2 that this method will clearly not always give you the *shortest* equivalent formula.

Having already mentioned above that we could also eliminate either $\wedge$ or $\vee$, we obtain:

Corollary 3.6.1. The sets $\{\wedge, \neg\}$ and $\{\vee, \neg\}$ are both truth-functionally complete.

Exercise 3.6.3. Show that $\{\rightarrow, \neg\}$ and $\{\rightarrow, \bot\}$ are both truth-functionally complete.

[Hint: Try the second one first.]

In this setting we can also ask whether there are other interesting truth-functionally complete connectives. Here is a curious one. Define a binary operator $\dagger$ to have the following truth table:

φ	ψ	$(\varphi \dagger \psi)$
T	T	F
T	F	F
F	T	F
F	F	T

Exercise 3.6.4. Show that $\{\dagger\}$ is truth-functionally complete.

Exercise 3.6.5. Define another binary connective that alone is truth-functionally complete.

Exercise 3.6.6. (Bonus) Is the set $\{\wedge, \vee\}$ truth-functionally complete? Why or why not?

3.7 Normal Forms: DNF and CNF

The formula in the previous section used to define the $*$ operator

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

has a particular form: it is a disjunction of conjunctions of **literals**. A literal is a propositional atom, or the negation of a propositional atom. Such a formula is said to be in **disjunctive normal form**, or DNF for short:

Definition 3.7.1. A formula is in **disjunctive normal form** (DNF) if it is a disjunction of conjunctions of literals, i.e., of the form $(l_{11} \wedge \cdots \wedge l_{1k}) \vee \cdots \vee (l_{n1} \wedge \cdots \wedge l_{nm})$.

Fact 3.7.1. Every formula $\varphi \in \mathcal{L}^{bool}$ is equivalent to a formula φ' in DNF.

Proof. One way to see this is to use a similar trick to the proof of Fact 3.6.1. Since φ has a truth table, we can simply read off the conjunctions from the rows of the truth table that make φ true.

Another way to show it is to use the logical equivalences in Figure 3.1, most of which we have noted: After first reducing our formula to one using only these three connectives, we then iteratively apply these equivalences to obtain a formula in DNF. $\square$

$$\varphi \wedge (\psi \vee \chi) \dashv\Vdash (\varphi \wedge \psi) \vee (\varphi \wedge \chi) \qquad \neg\neg\varphi \dashv\Vdash \varphi$$
$$\neg(\varphi \wedge \psi) \dashv\Vdash (\neg\varphi \vee \neg\psi) \qquad \neg(\varphi \vee \psi) \dashv\Vdash (\neg\varphi \wedge \neg\psi)$$
$$(\varphi \wedge \psi) \dashv\Vdash (\psi \wedge \varphi) \qquad (\varphi \vee \psi) \dashv\Vdash (\psi \vee \varphi)$$

Figure 3.1: Equivalences for converting to DNF.

For a rather complicated and unpleasant example, how would we find a formula in DNF equivalent to the following?

$$\neg(A \wedge (B \to C)) \wedge (B \vee (D \to \neg C))$$

We can proceed carefully as follows:

1. First eliminate all occurrences of $\to$, using the equivalence noted earlier, so that $(B \to C)$ gets replaced with $(\neg B \vee C)$ and $(D \to \neg C)$ gets replaced with $(\neg D \vee \neg C)$. The resulting formula looks like:
$$\neg(A \wedge (\neg B \vee C)) \wedge (B \vee (\neg D \vee \neg C))$$

2. Next, push the negation on the first conjunct inward, using the equivalence $\neg(\varphi \wedge \psi) \dashv\Vdash (\neg\varphi \vee \neg\psi)$, so that instead of $\neg(A \wedge (\neg B \vee C))$ we have $(\neg A \vee \neg(\neg B \vee C))$:
$$(\neg A \vee \neg(\neg B \vee C)) \wedge (B \vee (\neg D \vee \neg C))$$

3. Do this once more for the formula inside, so that $\neg(\neg B \vee C)$ becomes $(\neg\neg B \wedge \neg C)$:
$$(\neg A \vee (\neg\neg B \wedge \neg C)) \wedge (B \vee (\neg D \vee \neg C))$$

4. Using the "double negation" equivalence $\neg\neg\varphi \dashv\Vdash \varphi$, rewrite $\neg\neg B$ as simply B:
$$(\neg A \vee (B \wedge \neg C)) \wedge (B \vee (\neg D \vee \neg C))$$

5. Using the "distribution law" $\varphi \wedge (\psi \vee \chi) \dashv\Vdash (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$, with $\varphi = (\neg A \vee (B \wedge \neg C))$, $\psi = B$, and $\chi = (\neg D \vee \neg C)$, we can rewrite the whole thing as:
$$\big((\neg A \vee (B \wedge \neg C)) \wedge B\big) \vee \big((\neg A \vee (B \wedge \neg C)) \wedge (\neg D \vee \neg C)\big)$$

6. On the left disjunct, taking advantage of the symmetry of $\wedge$, namely $(\varphi \wedge \psi) \dashv\Vdash (\psi \wedge \varphi)$, we have the equivalent:
$$\big(B \wedge (\neg A \vee (B \wedge \neg C))\big) \vee \big((\neg A \vee (B \wedge \neg C)) \wedge (\neg D \vee \neg C)\big)$$

7. Using the distribution law again, we have that $\big(B \wedge (\neg A \vee (B \wedge \neg C))\big)$ is equivalent to

$\big((B \wedge \neg A) \vee (B \wedge (B \wedge \neg C))\big)$, indeed to $\big((B \wedge \neg A) \vee (B \wedge B \wedge \neg C)\big)$, giving us:

$$(B \wedge \neg A) \vee (B \wedge B \wedge \neg C) \vee \big((\neg A \vee (B \wedge \neg C)) \wedge (\neg D \vee \neg C)\big)$$

8. Next, the distribution law gives us that $(\neg A \vee (B \wedge \neg C)) \wedge (\neg D \vee \neg C)$ is equivalent to $\big((\neg A \vee (B \wedge \neg C)) \wedge \neg D\big) \vee \big((\neg A \vee (B \wedge \neg C)) \wedge \neg C\big)$, which in turn by symmetry is equivalent to $\big(\neg D \wedge (\neg A \vee (B \wedge \neg C))\big) \vee \big(\neg C \wedge (\neg A \vee (B \wedge \neg C))\big)$, leaving us with

$$(B \wedge \neg A) \vee (B \wedge B \wedge \neg C) \vee \big(\neg D \wedge (\neg A \vee (B \wedge \neg C))\big) \vee \big(\neg C \wedge (\neg A \vee (B \wedge \neg C))\big)$$

9. Applying the distribution law once more to the third and fourth disjuncts gives us:

$$(B \wedge \neg A) \vee (B \wedge B \wedge \neg C) \vee (\neg D \wedge \neg A) \vee (\neg D \wedge B \wedge \neg C) \vee (\neg C \wedge \neg A) \vee (\neg C \wedge B \wedge \neg C)$$

Here are some easier exercises:

Exercise 3.7.1. Put the following into DNF:

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

Although we will not use it in this class, it is worth noting that every formula is also equivalent to one in **conjunctive normal form** (CNF): a conjunction of disjunctions of literals. The proof of this uses an analogous distribution law. Instead of $\varphi \wedge (\psi \vee \chi) \dashv\vdash (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$, we use the fact that $\varphi \vee (\psi \wedge \chi) \dashv\vdash (\varphi \vee \psi) \wedge (\varphi \vee \chi)$.

3.8 A Formal Proof System

We have hinted at various points about using a logical system for **deductive reasoning**: deriving conclusions that follow logically from the premises. In this section we define such a reasoning system explicitly, with formal derivation rules. There are many types of **proof systems** for classical Boolean logic. The one we are presenting here is a "Fitch-style" system, named after the American philosopher Frederic Fitch. It is designed to mirror deductive reasoning patterns as naturally as possible. We will mention another system ("Hilbert style" system, named after the mathematician David Hilbert), which is in some ways simpler, but much more difficult to use, in §3.9.

We will be concerned with deriving a **conclusion** formula φ from some number of **premise** formulas $\psi_1, \ldots, \psi_n$. Every such proof will have the following form, with $\psi_1, \ldots, \psi_n$ listed in turn above the horizontal line, and φ appearing on the last row of the proof (line $n + k$).

1	ψ_1
$\vdots$	$\vdots$
n	ψ_n
$\vdots$	$\vdots$
$n+k$	φ

For each of the connectives we are going to give rules that allow deriving new formulas from old formulas. Let us start with conjunction.

Our first rule schema allows us to *eliminate* a conjunction in a proof. It is therefore called **conjunction elimination**, and notated $\wedge$E. The general schema has two forms, one to infer the left conjunct and another to infer the right conjunct:

$\vdots$	$\vdots$	
n	$\varphi \wedge \psi$	
$\vdots$	$\vdots$	
$n+i$	φ	$\wedge$E, n

$\vdots$	$\vdots$	
n	$\varphi \wedge \psi$	
$\vdots$	$\vdots$	
$n+i$	ψ	$\wedge$E, n

That is, if on row n we have derived $\varphi \wedge \psi$, then on any subsequent row $n+i$ we can derive φ or ψ. On the right hand side is the **justification** for this step, which includes $\wedge$E, together with the number of the row on which the conjunction occurs.

This was the rule for eliminating a conjunction. What about to *introduce* a conjunction? A moment's thought suggests the following rule schemata:

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	ψ	
$\vdots$	$\vdots$	
k	$\varphi \wedge \psi$	$\wedge$I, i, j

$\vdots$	$\vdots$	
i	ψ	
$\vdots$	$\vdots$	
j	φ	
$\vdots$	$\vdots$	
k	$\varphi \wedge \psi$	$\wedge$I, i, j

The rule of **conjunction introduction**, notated $\wedge$I, allows us to conclude a conjunction holds whenever we have derived both conjuncts.

Even though we only have two rule schemata so far, we already have enough now to produce our first simple proof. Recall that conjunction is symmetric, i.e., we have $(\varphi \wedge \psi) \dashv\vdash (\psi \wedge \varphi)$ for any formulas φ and ψ. How might we derive, for instance $B \wedge A$ from the premise $A \wedge B$? Here is a

simple proof. We list $A \wedge B$ as the only premise, apply $\wedge$E twice, then apply $\wedge$I to derive $B \wedge A$:

1	$A \wedge B$	
2	A	$\wedge$E, 1
3	B	$\wedge$E, 1
4	$B \wedge A$	$\wedge$I, 2, 3

This is our first example of a well formed proof.

How about disjunction? The introduction rule for disjunction is straightforward. We have one for adding a disjunct to each side:

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	$\varphi \vee \psi$	$\vee$I, i

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	$\psi \vee \varphi$	$\vee$I, i

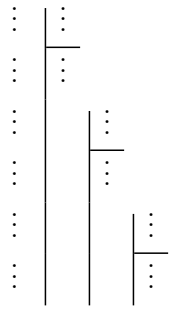
However, the rule for *eliminating* a disjunction is more complex. It requires a slight expansion of our notion of proof, to include the ability to make a **subproof**. The $\vee$E schema is as follows:

$\vdots$	$\vdots$	
i	$\varphi \vee \psi$	
$\vdots$	$\vdots$	
j	φ	
$\vdots$	$\vdots$	
k	χ	
$\vdots$	$\vdots$	
l	ψ	
$\vdots$	$\vdots$	
m	χ	
$\vdots$	$\vdots$	
n	χ	$\vee$E, i, j–k, l–m

Intuitively, a disjunction $\varphi \vee \psi$ says that either φ holds or ψ holds, but we may not know which. In order to conclude something from this, we could show that some other conclusion χ holds no matter which of φ or ψ is true. The $\vee$E rule captures this idea. We first *suppose*, in a subproof, that φ holds, and try, in that subproof, to derive χ. If we can do the same thing by supposing ψ holds, deriving χ in another subproof, then this rule allows us to derive χ *on the main column*, outside the

subproof. At that point we cite not only the row number of the original disjunction (i), but also the sequences of numbers for the subproofs ($j - k$, $l - m$).

We can in general introduce a subproof at any point in a Fitch proof, including subproofs within subproofs, and so on (see further on for many examples):



For all of the rule schemata given above, if we introduce a new formula by citing one of these rules, *all the rows cited must be in a subproof that is still open.* In particular we are not allowed to cite rows that appear in subproofs that have already been closed (such that the vertical line has ended). This will become clearer.

Here is another, more challenging example of a concrete proof, of another entailment that we verified using truth tables: let us derive $(A \wedge B) \vee (A \wedge C)$ from premise $A \wedge (B \vee C)$.

1	$A \wedge (B \vee C)$	
2	A	$\wedge$E, 1
3	$B \vee C$	$\wedge$E, 1
4	$\quad B$	
5	$\quad A \wedge B$	$\wedge$I, 2, 4
6	$\quad (A \wedge B) \vee (A \wedge C)$	$\vee$I, 5
7	$\quad C$	
8	$\quad A \wedge C$	$\wedge$I, 2, 7
9	$\quad (A \wedge B) \vee (A \wedge C)$	$\vee$I, 8
10	$(A \wedge B) \vee (A \wedge C)$	$\vee$E, 3, 4–6, 7–9

Note our strategy here. Because we started with a conjunction, we first eliminated this so that we could use the two conjuncts individually. We then realized that one of these conjuncts was a disjunction. So to use it we would need to consider a case distinction: show that our conclusion follows no matter which of the two disjuncts holds. We did this in two subproofs, by using the fact that $\vee$I always allows us to add on a "trivial" disjunct.

Exercise 3.8.1. In the other direction, derive $A \wedge (B \vee C)$ from the premise $(A \wedge B) \vee (A \wedge C)$.

While we are at it, we can also derive the other distribution law; that is, we can interderive $\varphi \vee (\psi \wedge \chi)$ and $(\varphi \vee \psi) \wedge (\varphi \vee \chi)$. Let us show that an instance of the first follows from the second, leaving the other direction as an exercise. This proof is a bit trickier, and requires subproofs within subproofs.

1	$(A \vee B) \wedge (A \vee C)$	
2	$A \vee B$	$\wedge$E, 1
3	$A \vee C$	$\wedge$E, 1
4	$\quad A$	
5	$\quad A \vee (B \wedge C)$	$\vee$I, 4
6	$\quad B$	
7	$\quad\quad A$	
8	$\quad\quad A \vee (B \wedge C)$	$\vee$I, 7
9	$\quad\quad C$	
10	$\quad\quad B \wedge C$	$\wedge$I, 6, 9
11	$\quad\quad A \vee (B \wedge C)$	$\vee$I, 10
12	$\quad A \vee (B \wedge C)$	$\vee$E, 3, 7–8, 9–11
13	$A \vee (B \wedge C)$	$\vee$E, 2, 4–5, 6–12

The overall strategy is to perform disjunction elimination on $A \vee B$. So we have to do a case distinction: in case of A we can get our result very quickly and easily by a simple disjunction introduction. However, in the case of B, in order to get our result we seem to need that other conjunct C as well. In this subproof for B (starting at row 6), we must also use the other disjunction, $A \vee C$. In either of these cases, together with the assumption of B, we can get our result. We must then close out of both of these subproofs, to pop back out to the main column to reach our conclusion.

Exercise 3.8.2. In the other direction, derive $(A \vee B) \wedge (A \vee C)$ from $A \vee (B \wedge C)$.

Before giving the rules for negation, let us suppose that we have in our language the 0-ary "absurdity" connective $\bot$, and let us give introduction and elimination rules for it first. While this is not strictly necessary, it will make the rules for negation slightly simpler. The rule of $\bot$E is simple. If we have derived $\bot$, that means something has gone wrong. The $\bot$E rule allows us to infer anything whatsoever from falsum. This is sometimes given the title of the **explosion rule**, or the rule of *ex falso quodlibet* ("from falsum anything follows"):

$\vdots$	$\vdots$	
i	$\bot$	
$\vdots$	$\vdots$	
j	φ	$\bot$E, i

The introduction rule for $\bot$ is also reasonably straightforward:

$$\begin{array}{l|ll} \vdots & \vdots & \\ i & \varphi & \\ \vdots & \vdots & \\ j & \neg\varphi & \\ \vdots & \vdots & \\ k & \bot & \bot\text{I}, i, j \end{array} \qquad \begin{array}{l|ll} \vdots & \vdots & \\ i & \neg\varphi & \\ \vdots & \vdots & \\ j & \varphi & \\ \vdots & \vdots & \\ k & \bot & \bot\text{I}, i, j \end{array}$$

That is, we can introduce $\bot$ at any point after we have derived a formula φ and its negation $\neg\varphi$. As usual, in this schema both φ and $\neg\varphi$ must have occurred along the same line as where we want to derive $\bot$, or along a line further to the left.

Finally, let us give the rules for negation. Eliminating negation is straightforward:

$$\begin{array}{l|ll} \vdots & \vdots & \\ i & \neg\neg\varphi & \\ \vdots & \vdots & \\ j & \varphi & \neg\text{E}, i \end{array}$$

If we have two negations, we can get rid of both of them. Introducing a negation now requires the use of subproofs:

$$\begin{array}{l|ll} \vdots & \vdots & \\ i & \quad\varphi & \\ \vdots & \quad\vdots & \\ j & \quad\bot & \\ \vdots & \vdots & \\ k & \neg\varphi & \neg\text{I}, i\text{–}j \end{array}$$

That is, if by conditionally supposing φ we can reach the absurd proposition, this means that we can go ahead and negate φ. This is a kind of "proof by contradiction." Assuming something holds and showing that it leads to absurdity is a way of proving something cannot possibly be true.

Suppose you think φ is false. If you can show that assuming φ leads to a contradiction, you have effectively shown that φ must be false, i.e., that $\neg\varphi$ is true.

Let us now see these rules in action. For instance, how might we derive $\neg(A \vee B)$ from

$(\neg A \wedge \neg B)$? Here is a proof using $\neg$I: assume $A \vee B$ holds and show this leads to absurdity.

1	$(\neg A \wedge \neg B)$	
2	$\neg A$	$\wedge$E, 1
3	$\neg B$	$\wedge$E, 1
4	$(A \vee B)$	
5	A	
6	$\bot$	$\bot$I, 2, 5
7	B	
8	$\bot$	$\bot$I, 3, 7
9	$\bot$	$\vee$E, 4, 5–6, 7–8
10	$\neg(A \vee B)$	$\neg$I, 4–9

Exercise 3.8.3. Give a derivation of $(\neg A \wedge \neg B)$ from $\neg(A \vee B)$.

Here is a more difficult derivation. Let us show that $(\neg A \vee \neg B)$ can be derived from $\neg(A \wedge B)$. This is another proof by contradiction, slyly using $\neg$I and several other rules in concert:

1	$\neg(A \wedge B)$	
2	$\neg(\neg A \vee \neg B)$	
3	$\neg A$	
4	$\neg A \vee \neg B$	$\vee$I, 3
5	$\bot$	$\bot$I, 2, 4
6	$\neg\neg A$	$\neg$I, 3–5
7	A	$\neg$E, 6
8	$\neg B$	
9	$\neg A \vee \neg B$	$\vee$I, 8
10	$\bot$	$\bot$I, 2, 9
11	$\neg\neg B$	$\neg$I, 8–10
12	B	$\neg$E, 11
13	$A \wedge B$	$\wedge$I, 7, 12
14	$\bot$	$\bot$I, 1, 13
15	$\neg\neg(\neg A \vee \neg B)$	$\neg$I, 2–14
16	$(\neg A \vee \neg B)$	$\neg$E, 15

Exercise 3.8.4. Derive $\neg(A \wedge B)$ from $(\neg A \vee \neg B)$.

All of our examples so far have been derivations from a single formula. But we could make formal proofs with any number of premises. For instance, *valid* formulas we know should follow from no premises at all. Indeed, here is a derivation of $A \vee \neg A$ from no premises:

1	$\neg(A \vee \neg A)$	
2	$\quad A$	
3	$\quad A \vee \neg A$	$\vee$I, 2
4	$\quad \bot$	$\bot$I, 1, 3
5	$\neg A$	$\neg$I, 2–4
6	$A \vee \neg A$	$\vee$I, 5
7	$\bot$	$\bot$I, 1, 7
8	$\neg\neg(A \vee \neg A)$	$\neg$I, 1–7
9	$A \vee \neg A$	$\neg$E, 8

We can also derive conclusions from multiple premises. To take a very simple example, consider again our example from the very beginning of the chapter. Let us derive A from $A \vee B$ and $\neg B$:

1	$A \vee B$	
2	$\neg B$	
3	$\quad A$	
4	$\quad A$	R
5	$\quad B$	
6	$\quad \bot$	$\bot$I, 2, 5
7	$\quad A$	$\bot$E, 6
8	A	$\vee$E, 1, 3–8, 5–7

Note on row 4 we wrote R for "repeat." This is just so that every line that is not a premise (or a premise of a subproof) has some justification.

Before closing this section, supposing we want to include $\rightarrow$ in our official language as well, what would be the introduction and elimination rules for this connective? The elimination rule $\rightarrow$E is rather obvious. It is sometimes called the rule of *modus ponens*. Again, we include both versions

(conditional first, then the antecedent; and vice versa).

$$\begin{array}{l|ll}
\vdots & \vdots & \\
i & \varphi \to \psi & \\
\vdots & \vdots & \\
j & \varphi & \\
\vdots & \vdots & \\
k & \psi & \to\text{E}, i, j
\end{array}
\qquad\qquad
\begin{array}{l|ll}
\vdots & \vdots & \\
i & \varphi & \\
\vdots & \vdots & \\
j & \varphi \to \psi & \\
\vdots & \vdots & \\
k & \psi & \to\text{E}, i, j
\end{array}$$

However, the introduction rule requires a subproof. What would it mean to be able to to conclude a material conditional? Intuitively, if by assuming φ holds we can conclude that ψ, then this means that $\varphi \to \psi$ must hold. The rule $\to$I captures this idea:

$$\begin{array}{l|ll}
\vdots & \vdots & \\
i & \quad \varphi & \\
\vdots & \quad \vdots & \\
j & \quad \psi & \\
\vdots & \vdots & \\
k & \varphi \to \psi & \to\text{I}, i\text{–}j
\end{array}$$

Here are some examples. The following is an easy derivation of $A \to (\neg A \to B)$:

$$\begin{array}{l|ll}
 & \quad & \\
1 & \quad A & \\
2 & \quad\quad \neg A & \\
3 & \quad\quad \bot & \bot\text{I}, 2 \\
4 & \quad\quad B & \bot\text{E}, 3 \\
5 & \quad \neg A \to B & \to\text{I}, 2\text{–}4 \\
6 & A \to (\neg A \to B) & \to\text{I}, 1\text{–}5
\end{array}$$

And here is a derivation of $(A \to B) \to (\neg B \to \neg A)$:

1	$A \to B$	
2	$\quad \neg B$	
3	$\quad\quad A$	
4	$\quad\quad B$	$\to$E, 1, 3
5	$\quad\quad \bot$	$\bot$I, 2, 4
6	$\quad \neg A$	$\neg$I, 3–5
7	$\neg B \to \neg A$	$\to$I, 2–6
8	$(A \to B) \to (\neg B \to \neg A)$	$\to$I, 1–7

Exercise 3.8.5. Give derivations of the following:

1. Show that $A \to (B \to A)$ can be derived from no premises.
2. Show that $A \vee (A \to B)$ can be derived from no premises.
3. Show that $A \to C$ can be derived from $A \to B$ and $B \to C$.
4. Show that $B \to A$ can be derived from $(B \vee C) \to A$.
5. Derive $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ from no premises.
6. Derive $(\neg A \to \neg B) \to (B \to A)$ from no premises.

Exercise 3.8.6. What would reasonable introduction and elimination rules be for the biconditional $\leftrightarrow$? And how about for $\dagger$?

Summarizing the presentation of the Fitch proof system for $\mathcal{L}^{bool}$ from this section, our rules for a valid proof are as follows:

- The n premises all begin above the top horizontal line, along the main column, on rows $1 - n$; the conclusion is the last row of the proof, also along the main column.
- Each row of a proof must either be the beginning of a subproof, indented one column to the right, or the conclusion of one of our 10 rules, citing the earlier lines that appear *either along the main column or in a subproof that is still open.*
- It is always possible to use the rule R to repeat a formula that appears earlier. But the same caveat about occurring in a (sub)proof that is still open applies.

All of our proof rules are summarized here (we leave out the duplicate rules for alternative orders):

Conjunction $\wedge$:

$\vdots$	$\vdots$	
n	$\varphi \wedge \psi$	
$\vdots$	$\vdots$	
$n+i$	φ	$\wedge$E, n

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	ψ	
$\vdots$	$\vdots$	
k	$\varphi \wedge \psi$	$\wedge$I, i, j

Disjunction $\vee$:

$\vdots$	$\vdots$	
i	$\varphi \vee \psi$	
$\vdots$	$\vdots$	
j	φ	
$\vdots$	$\vdots$	
k	χ	
$\vdots$	$\vdots$	
l	ψ	
$\vdots$	$\vdots$	
m	χ	
$\vdots$	$\vdots$	
n	χ	$\vee$E, i, j–k, l–m

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	$\varphi \vee \psi$	$\vee$I, i

Falsum $\bot$:

$\vdots$	$\vdots$	
i	$\bot$	
$\vdots$	$\vdots$	
j	φ	$\bot$E, i

$\vdots$	$\vdots$	
i	φ	
$\vdots$	$\vdots$	
j	$\neg\varphi$	
$\vdots$	$\vdots$	
k	$\bot$	$\bot$I, i, j

Negation $\neg$:

$$\begin{array}{l|l|l} \vdots & \vdots & \\ i & \neg\neg\varphi & \\ \vdots & \vdots & \\ j & \varphi & \neg\text{E}, i \end{array} \qquad \begin{array}{l|l|l} \vdots & \vdots & \\ i & \begin{array}{|l} \varphi \\ \hline \vdots \\ \bot \end{array} & \\ \vdots & \vdots & \\ k & \neg\varphi & \neg\text{I}, i\text{–}j \end{array}$$

Material conditional $\rightarrow$:

$$\begin{array}{l|l|l} \vdots & \vdots & \\ i & \varphi \rightarrow \psi & \\ \vdots & \vdots & \\ j & \varphi & \\ \vdots & \vdots & \\ k & \psi & \rightarrow\text{E}, i, j \end{array} \qquad \begin{array}{l|l|l} \vdots & \vdots & \\ i & \begin{array}{|l} \varphi \\ \hline \vdots \\ \psi \end{array} & \\ \vdots & \vdots & \\ k & \varphi \rightarrow \psi & \rightarrow\text{I}, i\text{–}j \end{array}$$

When we have a valid proof of φ from $\psi_1, \ldots, \psi_n$, we write $\{\psi_1, \ldots, \psi_n\} \vdash \varphi$. This is to be read, "there is a well-formed proof of φ from premises $\psi_1, \ldots, \psi_n$." And if we have just a single premise ψ we will often write $\psi \vdash \varphi$.

Exercise 3.8.7. Show the following:

1. $(\neg A \vee B) \vdash (A \rightarrow B)$
2. $(A \rightarrow B) \vdash (\neg A \vee B)$
3. $\vdash (A \rightarrow B) \vee (B \rightarrow A)$
4. $\vdash A \rightarrow \neg\neg A$
5. $\{(A \rightarrow \bot), (B \rightarrow A), (C \vee B)\} \vdash C$
6. $\vdash ((A \rightarrow B) \rightarrow A) \rightarrow A$
7. $\{(A \vee B), \neg C, (B \rightarrow C), (A \rightarrow B)\} \vdash D$
8. $\vdash \neg\neg A \vee \neg A$

3.9 Metalogical Concerns

When looking at truth tables and truth functions, we defined a notion of **logical entailment**, where $\varphi_1, \ldots, \varphi_n \models \psi$ means that every truth function satisfying all of $\varphi_1, \ldots, \varphi_n$ also satisfies ψ; alternatively, that every line of the joint truth table for all these formulas that sends $\varphi_1, \ldots, \varphi_n$ to T also sends ψ to T. Meanwhile, we also just defined a notion $\varphi_1, \ldots, \varphi_n \vdash \psi$, meaning that ψ can be given a **formal proof** using $\varphi_1, \ldots, \varphi_n$ as premises.

It is natural to ask about the relation between these two notions, $\models$ and $\vdash$. Do they coincide? Or might one be more inclusive than the other? A fundamental result shows that they in fact coincide:

Theorem 3.9.1 (Soundness and Completeness Theorem for Boolean Logic)**.** For any set Γ and formula φ, we have $\Gamma \models \varphi$, if and only if $\Gamma \vdash \varphi$.

Let us first give a sketch for the proof of the **soundness** direction: that $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$. How would one show something like this? The main idea is to show that all of the proof rules we have defined preserve truth: assuming all of the statements in the proof up to this point are true, and supposing we apply one of our rules to obtain a new statement, that statement is also guaranteed to be true.

Proof Sketch of Soundness. Given a row n with formula φ_n in some proof, let Φ_n be the set of all formulas that appear *as premises* along the main column or in subproofs that are still currently active, in rows *above* n. We want to show the following property $(*)$ holds of any well formed proof in our system:

$(*)$ For any formula φ_n at row n: $\Phi_n \models \varphi_n$.

As long as $(*)$ holds, this will prove soundness. For suppose $\{\psi_1, \ldots, \psi_m\} \vdash \varphi$; that is, there is a proof with $\psi_1, \ldots, \psi_m$ as premises (along the main column) and φ on the last row, say, k (again, on the main column). Then as a special case of $(*)$ we have that $\Phi_k \models \varphi$.

So how do we show $(*)$? Suppose for a contradiction that it is false of some purported proof, that there is a row n in some proof where φ_n is legally derived, but we do not have $\Phi_n \models \varphi_n$. If this happens there must be some *first* place where it happens: some row n such that all earlier formulas (on lines $k < n$) did follow from their respective sets Φ_k, but φ_n does not follow from Φ_n. In order to derive a contradiction we must go through all 10 (!) proof rules to show that none of these rules could have led to such an outcome. We illustrate a few cases here.

Suppose, for example, that at the problematic row we used an instance of $\rightarrow$E. Then our proof looks something like this:

$\vdots$	$\vdots$	
i	$\psi \rightarrow \varphi$	
$\vdots$	$\vdots$	
j	ψ	
$\vdots$	$\vdots$	
n	φ	$\rightarrow$E, i, j

where n is the first problematic line. By assumption $\psi \to \varphi$ is entailed by Φ_i, i.e., all the premises above it along the main axis, or in subproofs that are still open; and ψ is entailed by Φ_j. However, evidently $\Phi_n \nVdash \varphi$. That is, there is some truth assignment v making everything in Φ_n true, but making φ false. Note that $\Phi_i \subseteq \Phi_j \subseteq \Phi_n$. Hence, $v \vDash \chi$ for every $\chi \in \Phi_i \cup \Phi_j$. But then, by assumption, $v \vDash \psi \to \varphi$ and $v \vDash \psi$. Yet, by the definition of v applied to formulas of the form $\alpha \to \beta$, we must have $v \vDash \varphi$, which contradicts our earlier assumption. Hence, we can conclude that $\to$E could not have led to a violation of $(*)$.

What about, say, $\neg$I? Suppose we have a proof of the form:

$$
\begin{array}{c|l}
\vdots & \vdots \\
i & \quad \begin{array}{|l} \varphi \\ \hline \vdots \\ \bot \end{array} \\
\vdots & \\
j & \\
\vdots & \vdots \\
n & \neg\varphi \qquad\qquad \neg\text{I}, i\text{–}j
\end{array}
$$

and that n is the first row where, in this case, $\neg\varphi$ is not entailed by Φ_n. Then consider the earlier row j. Here we derived $\bot$, and let us suppose $\Phi_j = \{\chi_1, \ldots, \chi_k, \varphi\}$. Then evidently,

$$\{\chi_1, \ldots, \chi_k, \varphi\} \Vdash \bot$$

since n is supposed to be the first row where this fails. That is to say, any truth assignment v for which $v \vDash \chi_1$, ..., $v \vDash \chi_k$, $v \vDash \varphi$ will also have $v \vDash \bot$. Since there are no such valuations (recall $v \nvDash \bot$ for *any* valuation v), we conclude that any valuation v for which $v \vDash \chi_1$, ..., $v \vDash \chi_k$ will also give $v \nvDash \varphi$, i.e. $v \vDash \neg\varphi$. In other words, any valuation making everything in $\Phi_n = \{\chi_1, \ldots, \chi_k\}$ true will also make $\neg\varphi$ true, i.e., $\Phi_n \Vdash \neg\varphi$, a contradiction.

Going through the other 8 cases shows that none of these rules could lead to a row where the formula is not entailed by all the premises in play up to that step. □

Exercise 3.9.1. Fill in the following cases in the proof of Theorem 3.9.1: $\wedge$I, $\neg$E, $\bot$E, $\vee$E, $\to$I.

The proof of the **completeness** direction is somewhat more difficult. There are many ways to show it, and if you take Phil 151 you will prove this result in a very simple way as a homework exercise (which also generalizes easily to the case of infinitely many premises). But here we give a different, more direct proof sketch, which does not involve any advanced techniques.

Proof Sketch of Completeness. First we argue that $\Vdash \varphi$ implies $\vdash \varphi$, ignoring premises. The main idea is to show that we can *derive* φ in our proof system from a formula φ^* that is equivalent to φ but is in conjunctive normal form (a conjunction of disjunctions of literals). This involves showing that each of the equivalences in Fig. 3.1 (with the alternative distribution rule appropriate for CNF instead of DNF) is actually derivable. We have actually done much of this work already.

Next, φ being valid, and thus φ^* being valid, means that each conjunct of φ^* must contain some atom A_i and its negation $\neg A_i$ among its disjuncts. Then it is relatively straightforward to show that each of these conjuncts will be derivable: we can derive any formula of the form $(\alpha \vee A_i \vee \beta \vee \neg A_i \vee \gamma)$. First show that $(A_i \vee \neg A_i)$ is derivable; then it follows immediately that $(A_i \vee \neg A_i) \vee (\alpha \vee \beta \vee \gamma)$ is by $\vee$I; finally, $(\alpha \vee A_i \vee \beta \vee \neg A_i \vee \gamma)$ in derivable because in general $\varphi \vee (\psi \vee \chi) \dashv\vdash (\varphi \vee \psi) \vee \chi$ and $\varphi \vee \psi \dashv\vdash \psi \vee \varphi$. Because each of the conjuncts of φ^* is derivable, the formula is itself derivable by a number of applications of $\wedge$I. We then know that φ is also derivable, because we can apply the CNF-equivalences in reverse to obtain exactly φ.

If one wants to show that $\{\alpha_1, \ldots, \alpha_n\} \Vdash \varphi$ implies $\{\alpha_1, \ldots, \alpha_n\} \vdash \varphi$, first apply the Deduction Theorem (recall Fact 3.5.1) multiple times to show $\Vdash (\alpha_1 \to (\alpha_2 \to \ldots (\alpha_n \to \varphi) \ldots))$. Deriving this from an equivalent CNF as above then gives us $\vdash (\alpha_1 \to (\alpha_2 \to \ldots (\alpha_n \to \varphi) \ldots))$. The final result is by an application of a kind of Deduction Theorem for derivability: that $\Gamma \vdash \alpha \to \beta$ implies $\Gamma \cup \{\alpha\} \vdash \beta$. But this is straightforward by a simple application of $\to$E. □

Optional aside: It is also worth remarking again that this particular proof system we have built is only one among many complete proof systems. The Fitch style system for the language with connectives $\{\wedge, \vee, \neg, \bot, \to\}$ has no **axioms**, only **rules**, in particular introduction and elimination rules for all the connectives. There is an alternative complete system that has only three axioms and one rule, for the language with only $\{\to, \neg\}$ (which we saw in Ex. 3.6.3 is expressively complete). The axioms are the following (which we have already verified are valid):

1. $\varphi \to (\psi \to \varphi)$
2. $(\neg\psi \to \neg\varphi) \to \big((\neg\psi \to \varphi) \to \psi\big)$
3. $\big(\varphi \to (\psi \to \chi)\big) \to \big((\varphi \to \psi) \to (\varphi \to \chi)\big)$

And the sole rule is *modus ponens*: from $\varphi \to \psi$ and φ, infer ψ. A *proof* of φ in this system is just a sequence of formulas, each of which is either a premise, an instance of one of the formulas above, or a result of applying *modus ponens* to two formulas that occurred earlier in the sequence, and such that φ is the last element of the sequence. This system is also complete.

One way (not necessarily the best way) to show soundness and completeness of this system would be to show it is actually equivalent to our proof system: anything provable in one system is also provable in the other. One direction of this is easy, for it is a simple matter to show that we can derive each of the schemata above in our proof system. For instance, 1 is essentially Ex. 3.8.5 part 1, while 3 is essentially Ex. 3.8.5 part 5. We would then need to verify that derivability in our system is closed under *modus ponens*. But that is obvious too because of our rule $\to$E.

The other direction is more challenging. We need to show that all of our rules can be mimicked by derivations in this small system. The first step is to rewrite all formulas using $\{\wedge, \vee, \neg, \bot, \to\}$ with only $\{\to, \neg\}$. Then, for instance, consider $\vee$I. We need to show that we can derive $\alpha \vee \beta$, i.e., $\neg\alpha \to \beta$, from the premise α. This takes a surprising amount of work (just try it!). Further care would be needed to show how we could mimic proofs that involve subproofs in this kind of system. In general we would need to show this by induction on proofs in our system.

Exercise 3.9.2. Show how to derive $A \to C$ from premises $A \to B$ and $B \to C$ in this system.

Chapter 4

Applications of Boolean Logic

The structure of Boolean Logic is genuinely ubiquitous. Here we illustrate just a few example areas where we find Boolean Logic at work.

4.1 Natural Language: Logical Connectives in English

Boolean Logic is a formal system, given by a precisely specified language of wffs, and a mathematically precise semantics in terms of truth functions. But one cannot help but notice the apparent similarity between our truth functional connectives like $\wedge$, $\vee$, $\neg$, and $\rightarrow$, and natural language connectives like 'and', 'or', 'not', and 'if…then'. Translating natural languages into formal languages can have a number of important uses: careful analysis of philosophical arguments couched in natural language; cross-linguistic comparisons of what *meanings* are expressed across different languages; cognitive modeling of what happens when people understand and produce language; and transforming text into a format that is easier for computers manipulate and reason with.

Suppose we decide to render the English expressions above—'and', 'or', 'not', and 'if…then'—as their formal counterparts—$\wedge$, $\vee$, $\neg$, and $\rightarrow$. How faithful of a translation would this be? This question has been debated for many decades, and it is still a matter of controversy in linguistics and philosophy. In this section we give some sense of these issues.

4.1.1 Successes

The examples we introduced at the beginning of Chapter 3 suggest that $\wedge$ may actually not be a terrible rendering of 'and' in English. Again, if we take A to mean,

(1) The NASDAQ is down 1%.

and B to mean,

(2) The S&P 500 is up 0.5%.

then how should we formalize (3)?

(3) The NASDAQ is down 1% and the S&P 500 is up 0.5%.

What claim is being expressed by (3)? Under what conditions could the sentence be said *truly*? Intuitively, it needs to be the case that both A and B are true; otherwise it is false. But that is of course exactly what $\wedge$ does. Evidently $A \wedge B$ means just what we want it to mean in this case.

What about $\vee$ and 'or'? If I say,

(4) A Democrat will win a seat in the Senate, or a Republican will win a seat in the senate.

this prediction will be true as long as at least one of these events occurs, including the possible outcome where both events happen. Indeed, the sentence seems to have a meaning very close to the formal logical counterpart $D \vee R$. People even seem to use 'or' in a way that corresponds naturally our earlier proof pattern: from $A \vee B$ and $\neg A$, infer B. For example, someone who says,

(5) Give me this apple for $2, or I am moving on to the next vendor.

is inviting the addressee to make the inference about what will happen if $2 is not accepted.

English 'not' also seems quite close to how $\neg$ behaves. If I say,

(6) It is not true that a Democrat will win.

this seems to express something very much like $\neg D$, where again D stands for, "A Democrat will win." In fact, even combining 'not' with 'and' seems to correspond closely to the compound logical meaning, witness (7):

(7) It is not the case that both a Democrat will win and a Republican will win.

means something like $\neg(D \wedge R)$: true if at least one of D and R is false, also if they both are.

The conditional construction 'if ... then' in English has at least some features in common with $\rightarrow$. Consider, for instance,

(8) If a Democrat wins, then taxes will go up.

(8) makes a certain claim about what will happen if D turns out to be true. It makes no claim about the case where a Democrat does not win.[1] Speakers even seem to use the truth table for $\rightarrow$ in clever ways, for instance to negate a claim indirectly, as in (9):

(9) Sure, if you were in the Olympics, then I'm the Queen of England.

Since everyone knows the speaker is not the Queen of England $\neg Q$, the only way this conditional claim $O \rightarrow Q$ can be true is if the antecedent O is also false.

Finally, other constructions in English (and other languages) that do not themselves correspond directly to one of these connectives can nonetheless be rendered using the connectives. For instance, consider what 'unless' means in English:

(10) France is not signing the treaty unless Germany is signing the treaty.

[1] You might ask yourself whether you would consider a conditional like (8) true in *any* case where the antecedent is false. Consider, for instance, "If there is a unicorn taking Phil 150, then it has the highest grade in the class." Is this sentence *true*? Is it *false*? (Does it have a truth value at all?)

This might be rendered as something like $\neg G \rightarrow \neg F$. Similarly, 'only if' is often taken to correspond to the material conditional, so that

(11) France is signing the treaty only if Germany is signing the treaty.

will be rendered as $F \rightarrow G$ (which, by the way, would mean that (10) and (11) are assumed to be synonymous, since $F \rightarrow G$ and $\neg G \rightarrow \neg F$ are logically equivalent). You will often see—especially in mathematical contexts, including in this text—the expression 'if and only if'. The convention is that this should be read as the biconditional $\leftrightarrow$, and is sometimes written 'iff'. For, e.g.,

(12) France is signing the treaty if and only if Germany is signing the treaty.

(12) should be understood as $F \leftrightarrow G$, which recall is equivalent to $(F \rightarrow G) \wedge (G \rightarrow F)$.[2]

4.1.2 Puzzles

Despite the evident naturalness of these renderings of English into $\mathcal{L}^{bool}$, there are a number of puzzles where the meanings seem to diverge. We go through a few examples.

Consider the following use of 'and':

(13) Silvia read *To Kill a Mockingbird* and decided to become a lawyer.

Suppose we render (13) as $R \wedge L$, with R standing for, "Silvia read *TKM*," and L standing for, "Silvia decided to become a lawyer." As we observed in Chapter 3, $R \wedge L$ is equivalent to $L \wedge R$: the operator $\wedge$ is symmetric. Specifically, if (13) is rendered as $R \wedge L$, then it will be predicted true even if Silvia read *TKM* only *after* having decided to become a lawyer. One might conclude from this that the use of 'and' in (13) does more than merely assert the joint truth of R and L: it also says something about their temporal order, perhaps even something about the *causal order* between R and L: that her reading played some role in her decision. Maybe, therefore, the truth functional meaning of $\wedge$ is not rich enough to capture some of the subtle things we do with 'and'.

There are also puzzles about 'or' that make it seem as though there is a layer of meaning beyond what is given in the truth table for $\vee$. Take (14):

(14) The meeting will be in either building 90 or in building 100.

If I say (14) under normal circumstance—e.g., if we are on our way to the meeting—you will take me to be saying not only that the meeting is in one of these two places, but that *I do not know which it is*. The connective $\vee$ does not capture this *epistemic* aspect of 'or' at all.

Even 'not' seems to diverge from the logical meaning of $\neg$, as in (15):

(15) Well, the movie is not *not* scary ...

This sentence is not necessarily equivalent to the statement that the movie is scary, even though $\neg\neg S \dashv\vdash S$. Rather, (15) seems to be suggesting that there may be parts of the movie that were somewhat scary, even if the movie as a whole could not be described as scary, which suggests someone might even reasonably make a claim that would appear outright contradictory:

[2]The use of 'if and only if' is perhaps less an example of logic providing a good formalization of ordinary English, as certain uses of English taking on a precise logical meaning in specific (e.g., mathematical) contexts.

(16) The movie is not scary, but it is not not scary.

Puzzles of this sort abound. Linguists and philosophers have come up with dozens and dozens of classes of examples where English and the standard Boolean connectives come apart. The following is an amusing example concerning the material conditional:[3]

Exercise 4.1.1. There was once an advertisement for a tennis magazine that included (17).

(17) If I am not playing tennis, I am watching tennis. And if I am not watching tennis, I am reading about tennis.

Suppose we write P for "playing tennis," W for "watching tennis," and R for "reading about tennis." Assume in the background that the speaker cannot do be doing more than one of these at once, so that $\neg\big((P \wedge W) \vee (P \wedge R) \vee (W \wedge R)\big)$ is true. Suppose also that we render the statements in (17) in terms of the material conditional, so that (17) asserts both $\neg P \rightarrow W$ and $\neg W \rightarrow R$.

We claim that it follows from the assumptions above that the speaker must always be engaged in exactly one of the three activities; that is, exactly one of P, W, and R must be true. Identify which it is, and show using either a formal derivation or a truth table that the atomic sentence in question is entailed by assumptions $\neg\big((P \wedge W) \vee (P \wedge R) \vee (W \wedge R)\big)$, $\neg P \rightarrow W$, and $\neg W \rightarrow R$.

4.1.3 Gricean Pragmatics

Within philosophy, through the middle of the 20th century, there was a split between two views on the relation between logic and language, sometimes called *formalism* and *informalism*. Formalists thought that natural language was too hopelessly ambiguous, vague, imprecise, and generally messy to be used for any kind of systematic philosophical or scientific theorizing, and that such theorizing should therefore be done in an unambiguous, precise formal logical language like $\mathcal{L}^{bool}$. The informalists, by contrast, thought that natural language has its own logic and possesses a subtlety of usage that cannot be fully captured by excessively mathematicized formal systems like logic, and that philosophy would become shallow and too divorced from human experience if done formally.

In the mid 1960s philosopher H.P. Grice took issue with a common presupposition shared by both formalists and informalists: the assumption that examples like those in the previous section actually show there is a deep gulf between the natural language connectives and Boolean connectives. Grice began a program of trying to *explain away* as many of the puzzles above as possible, by invoking *general principles of conversation between rational agents*. The research program has been hugely influential in philosophy and linguistics—far beyond just vindicating the Boolean connectives, offering a very general framework for principles of communication and linguistic pragmatics—and it has even been taken up by computer scientists and psychologists interested in conversation.[4]

Grice's basic idea was that, by and large, interlocutors share a common goal of more or less efficient and effective communication between speaker and hearer. He formulated the following principle as a general characterization of what makes rational cooperative conversation:

[3] This example and exercise is due to Herbert Enderton, *A Mathematical Introduction to Logic*.

[4] This is especially true at Stanford. See, e.g., Frank & Goodman (2012), "Predicting Pragmatic Reasoning in Language Games," *Science*.

Cooperative Principle: Make your conversational contribution such as is required at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged.

Grice supposed that this principle is one that people should—and largely *do*—observe, and that this fact can explain much of the apparent divergence between logic and language. Specifically, he proposed four **maxims** of conversation that are supposed to govern how people talk to one another:

Quantity:

(a) Make your contribution as informative as is required.

(b) Do not make your contribution more informative than is required.

Quality:

(a) Do not say what you believe to be false.

(b) Do not say that for which you lack adequate evidence.

Relation:

Be relevant.

Manner:

(a) Avoid obscurity of expression.

(b) Avoid ambiguity.

(c) Be brief.

(d) Be orderly.

These maxims are supposed to be more specific versions of the Cooperativity Principle, and are not meant to be a closed and final list. Already, however, Grice claimed that we could use these to explain some of the puzzles above.

Consider, for instance, the problem in (14) with 'or'. It seemed like the disjunctive claim is saying not only that one of the disjuncts is true (that the meeting is in one of the two buildings), but that the speaker does not know which. For Grice, this is indeed a reasonable inference to make. But it is not because this is dictated by the meaning of 'or'. Rather, *given* the standard meaning of 'or' as $\vee$, this inference is a result of the listener making reasonable assumptions about the speaker being a cooperative conversational partner, and making an appropriate inference.

In particular, drawing upon the maxims of *Quantity* and *Quality*, our listener might reason:

1. The speaker S said "The meeting is either in building 90 or 100."

2. S knows that we are both trying to make it to the meeting, and therefore that it is relevant which building the meeting is in.

3. If S were observing the maxim of Quantity, and if S knew which building the meeting is in, then S would have specified which building it is.

4. S is observing the maxim of Quantity.

5. Therefore—given 1, 3, and 4—the only way S could have been more specific would have been to say something for which there was insufficient evidence.

6. S is observing the maxim of Quality.

7. Therefore, S does not know (or have sufficient evidence about) where the meeting is.

This informal argument reconstructs reasoning that a person *could* undergo to reach the conclusion that the speaker is uncertain about where the meeting was. Grice never made the *psychological* claim that people do in fact go through reasoning of this sort, but only that the rational argument could be reconstructed, and that it does not require revising the assumption that 'or' means $\vee$.

How about the example of 'and' and temporal order from (13)? The explanation here is even simpler. A speaker could go through the following reasoning:

1. S said, "Silvia read *TKM* and decided to become a lawyer."

2. If S were observing the fourth maxim of Manner—Be orderly—then S would mention the events in the order in which they occurred.

3. S is observing the maxim of Manner.

4. Therefore, S is telling me that Silvia read *TKM and then* decided to become a lawyer.

One could even try to derive the additional inference of a causal relation between the events, using the maxim of *Relation*:

5. If S were observing the maxim of Relation, then the fact that these two events were mentioned together in the same sentence suggests that the second must be somehow relevant to the first.

6. S is observing the maxim of Relation.

7. Silvia's reading of the novel must be somehow relevant to her deciding to become a lawyer.

8. Given the inferred temporal order (4 above), general world knowledge (including perhaps knowledge about the novel) would suggest that Silvia decided to become a lawyer as a result of her reading *To Kill a Mockingbird*.

To repeat, the details of these arguments are not critical (and are in fact a matter of much psychological, philosophical, and linguistic conversy). The important point is that the reasoning seems to draw only on very general knowledge about the world, the speaker, and the assumption that the speaker is being cooperative, viz. following the maxims.[5] It does not require giving up the assumption that the logical connectives provide an appropriate analysis of the connectives in English.

[5]The careful reader may notice that, in order for this to work, we may also need assumptions about what the interlocutors know about what each other knows about what each other knows, and so on. This also is a matter of some controversy. But we will in fact come back to this type of *higher-order knowledge* in a different setting in §5.2.

To distinguish with what sentences like (13) and (14) **entail**—which we might presume is just what is entailed by their standard logical renderings using $\wedge$ and $\vee$—Grice reserved the term **conversational implicature** for the other reasonable inferences that seem to follow from these sentences. As we say, (13) **implicates**, without **entailing**, that the events happened in a particular order; likewise (14) merely **implicates** that the speaker does know more.

Exercise 4.1.2. Another example of an apparent implicature is from 'only if' to the stronger 'if and only if.' For instance, consider (18):

(18) You can turn your homework in late only if you use a late day.

If we were to render the meaning of 'only if' in terms of $\rightarrow$, this would become something like $T \rightarrow L$, where T is, "You can turn in your homework late," and L is, "You are using a late day." However, one might reasonably understand (18) to be asserting the stronger claim $T \leftrightarrow L$. Surely one can infer from (18) that if you use a late date then you can turn in your homework late!

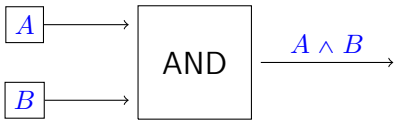
Try to construct an informal argument—similar to those above, using the Gricean maxims—that would lead to this evidently reasonable conclusion.

Literally tomes have been written about Gricean pragmatics, and this is only the tip of the iceberg. The important take-away is that at least some of the apparent mismatches between logic and language can potentially be explained in this way. Some puzzles are more difficult to diagnose using this strategy—consider the negation and conditional examples above, which may genuinely require alternative non-Boolean analyses—and the exact relation between the connectives in English and the Boolean connectives remains an ongoing research question across several disciplines.

4.2 Computation I: Circuits

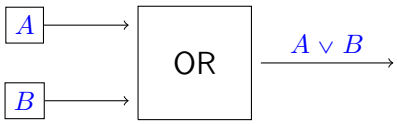
In this section we explore a quite different application of Boolean Logic: not to natural language, but to the fundamentals of computing. Electrical circuits form the underpinnings of essentially every modern digital computing device, and their activity can be described using the same tools we have been developing for Boolean Logic.

Imagine an electrical device that takes two inputs, say, A and B, each of which may take on one of two values—T or F—and which outputs another value, again, either T or F. For instance, this device might output T just in case both of its inputs have value T:

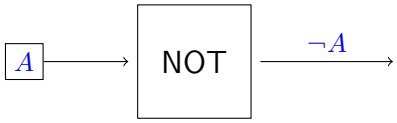


In other words, this device computes $A \wedge B$: it outputs T if they are both T, and F otherwise. We call such a device an AND gate.

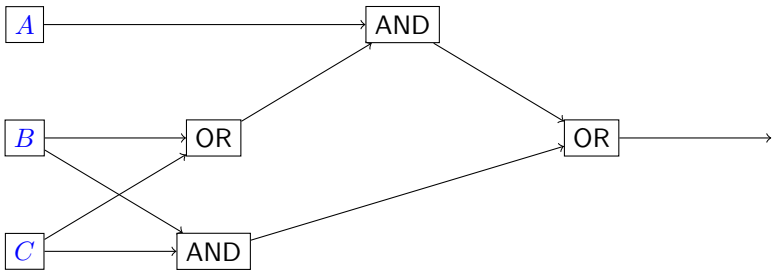
We can also imagine a device that computes $A \vee B$: it outputs T if either of A or B (or both) has value T, and otherwise F. This device is called an OR gate.



There may also be logic gates that only take a single input and reverse the value of that input: if the input has value T, then the device outputs value F; if the input has value F, then the device outputs T. This is called a NOT gate.



Circuits become most interesting when we *compose them together* to compute new logical operations. For instance, consider the following circuit:



This particular circuit can be seen to give an output that corresponds to whatever the *majority* of input values are. In other words, it computes the function:

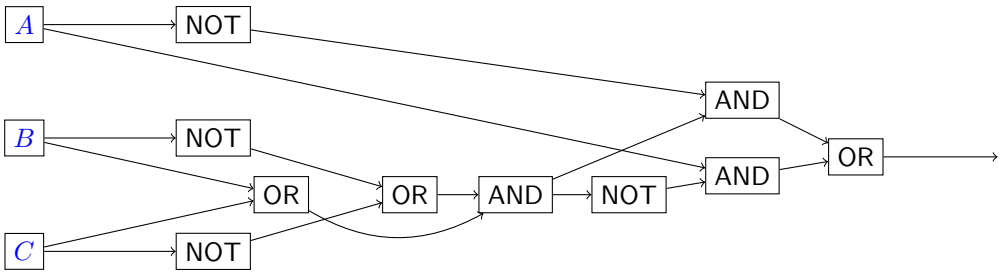
$$(A \wedge (B \vee C)) \vee (B \wedge C) \,.$$

Note how we can actually "reuse" the same gates several times in the circuit. In this example, the formula uses (not including parentheses) nine symbols, whereas the circuit only includes seven gates.

An even more extreme instance of this kind of "reuse" can be seen in the following example. What would a propositional formula look like that says "an odd number of A, B, and C is true"? Here is one way of writing such a formula:

$$(A \wedge \neg((B \vee C) \wedge (\neg B \vee \neg C))) \vee (\neg A \wedge (B \vee C) \wedge (\neg B \vee \neg C)) \,.$$

This formula uses 25 symbols. However, we can construct an equivalent circuit using only 13 gates:



You should verify for yourself that this machine indeed computes the function above. Note also that an entire subformula—$(B \vee C) \wedge (\neg B \vee \neg C)$—only needs to be created once.

In general, for any formula of boolean logic—that is, any formula in our language $\mathcal{L}^{bool}$—it is possible to construct such a machine that computes the same truth function. That is, for any given "input"—assignment of T or F to the atomic sentences—that row of the truth table for this formula will give the same result as the corresponding logic circuit's output. It is also obvious that for any circuit of this form, there is a boolean formula that is equivalent to it in this sense.

We can of course imagine logic gates that perform tasks other than AND, OR, and NOT. For instance, recall the logical connective $\dagger$, for which the formula $A \dagger B$ is true just in case A and B are both false. In the study of circuits, such a logic gate is called a NOR gate.



We saw above in Ex 3.6.4 that $\dagger$ by itself is truth-functionally complete. This means that, in principle, we could build any possible circuit, perform any logical operation whatsoever, using only this single kind of logic gate.

Exercise 4.2.1. Using only NOR gates, construct a logic circuit that computes $A \rightarrow (B \vee C)$.

Circuits form the basis of essentially all modern digital computing devices. At the most fundamental level, all of our computers are essentially just performing boolean operations; that is, given the above, from a certain perspective, they are just computing truth values of complex formulas given the truth values of simpler formulas, according to the standard truth tables! The study and theory of circuits is closely bound up with questions about propositional logic. It is also closely tied up with the study of computational complexity, to which we now turn.

4.3 Computation II: Complexity

Consider the problem of determining whether a given row in the truth table for φ is T or F. Intuitively, this task is relatively easy. If we are already given the row, we just have to start with the "innermost" subformulas and work our way out, until we have computed the truth value of the main ("outermost")

connective. In fact, if we count the number of intuitive "steps" we need to go through, it is not much higher than the *length* of the formula, for instance, the number of atomic sentences that appear in the formula. This is an instance of a computational problem that is said to be *solvable in polynomial time*. It is in the *complexity class* P.

The problem of determining whether a given formula φ (specifically one in conjunctive normal form) is *satisfiable*—that is, whether there is a line in the truth table for φ that comes out T—is intuitively much harder. That is, it takes a lot longer. We have to construct the entire truth table ourselves, and check each one, going through the above computation many times. As we know, if there are k atomic sentences in the formula φ, then the truth table for φ has 2^k rows. Thus, we have to go through the steps described in the previous paragraph up to 2^k times. This number grows extremely fast. For instance, with just 10 propositional atoms the truth table already has over a thousand rows. With 20, the number is already over a million.

If we could somehow "guess" the right row and merely verify that this row indeed satisfies the formula, then the task would be easy, just as easy as computing a single row in the truth table. However, no one knows how to make smart guesses like this in general. Every algorithm for determining satisfiability—this is traditionally called the SAT problem—takes time *exponential* in the number of atomic sentences. Even if we could get this down from 2^k to something like, say, 1.1^k, this would still be intractable. For formulas with 145 propositional atoms, which is not that many for real industrial applications, we would still require over a million steps. With 500 the number is well above 10^{20}.

The SAT problem is the best studied example of a problem which is said to be *solvable in non-deterministic polynomial time*, or in *complexity class* NP. What this means exactly is not important for our purposes. The important point is that problems like this would be "easy" (in class P) if we knew how to guess the right solution, and only needed to verify that it was a solution. But all known algorithms for problems like this take exponentially long. The biggest open question in complexity theory, and arguably one of the biggest questions in all of mathematical science, is:

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

That is, for problems like SAT, is it possible to solve these in a tractable (specifically polynomially bounded) amount of time?

The SAT problem is an instance of what is called an NP-complete problem: it is a member of the class NP, and if it is also in class P, then so is every other NP-complete problem! Many, many problems in computer science—and other fields like chemistry, sociology, etc.—are known to be in this class. Thus, if you could determine an efficient method for solving the SAT problem, you would thereby automatically be solving thousands of other problems, none of which currently has a perfect (reliable and efficient) solution. By contrast, proving instead that there is no efficient method for determining satisfiability would show that none of these problems has an efficient solution either.

4.3.1 A Curious Connection to Physics

[This section is fully optional, only for those who are interested.]

In the mid 1990s researchers concerned with the SAT problem noticed there are some intriguing connections between the Boolean satisfiability problem and ideas from modern physics.

If φ is in conjunctive normal form:

$$\varphi \quad = \quad (l_{11} \vee \cdots \vee l_{1n}) \wedge \cdots \wedge (l_{k1} \vee \cdots \vee l_{km})$$

then we call each of these conjuncts (which is itself a disjunction) a **clause**. A special case of the Boolean satisfiability problem is when each of these clauses contains exactly *three* literals. (So, in the expression above, e.g., $n = \cdots = m = 3$.) This version of the problem is called the 3-satisfiability problem, or **3SAT** for short. It turns out 3SAT is just as hard as the general satisfiability problem, indeed it is also NP-complete. If we could solve 3-SAT efficiently, we would be able to solve the general SAT problem efficiently (and hence every other problem in NP).

Suppose φ has c clauses, and contains overall a propositional atoms. Then we say the **density** of φ is given by the ratio c/a of clauses to atoms. A robust finding in the SAT-solving literature is that the difficulty of the satisfiability problem is closely tied up with the density of formulas. Formulas of very low density tend to be satisfiable, while formulas of high density tend to be unsatisfiable. (Intuitively, why might this be?) Figure 4.3.1 shows results of an empirical study of SAT instances.

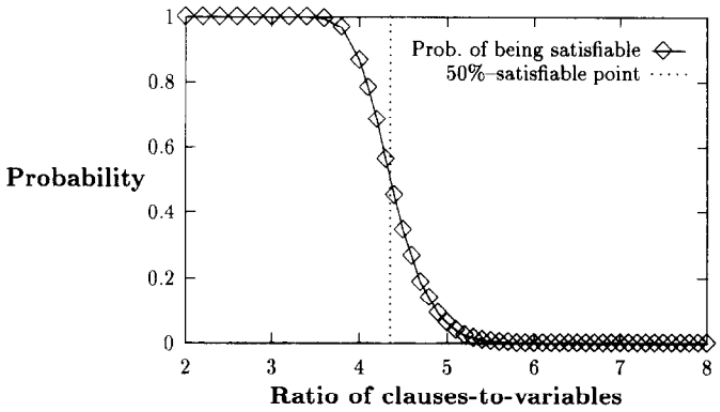


Figure 4.1: Empirical SAT data from Mitchell et al. (1992), *Artificial Intelligence*.

To say that formulas of a certain density have **asymptotic probability** 0 (or 1) of being satisfiable means: if we look at larger and larger formulas with that density, the proportion of those that are satisfiable will converge to 0 (1). Indeed, there are now mathematical results showing such convergence definitely holds for certain densities:

Theorem 4.3.1. For 3SAT instances with density < 3.52, the asymptotic probability of being satisfiable is 1. For 3SAT instances with density > 4.4898, the asymptotic probability is 0.[6]

[6]The first result is from Kaporis et al. (2006), "The Probabilistic Analysis of a Greedy Satisfiability Algorithm,"

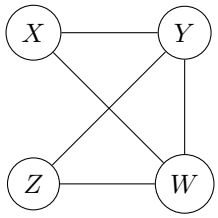
As can be seen from the graph, there seems to be a rather dramatic transition from (asymptotic) unsatisfiability to satisfiability. Intuitively, one might try to locate much of the difficulty in SAT solving with this transition. While for lower or higher densities, formulas all tend to be (un)satisfiable, here formulas are roughly equally likely to be unsatisfiable as satisfiable. The proposal in the literature is that this represents a genuine **phase transition**, just as, for instance, there is a sharp phase transition from liquid to solid states of matter as described by thermodynamics. An open mathematical question is where this transition occurs. What exactly is the density such that asymptotically 50% of instances are satisfiable? The current estimate, based on empirical studies of randomly generated instances, is around 4.26.

4.3.2 Example: The Clique Problem

To give a sense for how the Boolean satisfiability problem is, from a complexity point of view, equivalent to many other hard problems, we will consider one specific example of how we can translate back and forth between logic problems and, in this case, graph problems.

Given a graph (V, E) with vertices V and edges E, a *clique* is a subset S of the vertices, such that all elements of S are connected (via E) to all other elements of S. For instance, if V are all the people on a social network, and the edge relation E holds between two people if they are friends, a clique is a set of people all of whom are friends with each other.

A typical problem in the analysis of graphs—including social networks, and many other kinds of graphs—is to determine whether the graph has a clique of size at least k. For instance, the graph below has a clique of size 3, but none of size 4.



For instance, the set $\{X, Y, W\}$ forms a clique, as does $\{Y, Z, W\}$, however the one four-element set $\{X, Y, Z, W\}$ does not form a clique, since there is no edge between X and Z.

In general, we can consider the following problem:

> **The Clique Problem**: Given a graph (V, E) and a number k, does there exist a clique in (V, E) of size at least k?

This problem is also known to be NP-complete, and in fact we can see in what sense every instance of the Clique Problem can be *transformed* into a SAT problem, and *vice versa*.

Random Structures and Algorithms. The second is from Diaz et al. (2008), "A New Upper Bound for 3-SAT," *Foundations of Software Technology and Theoretical Computer Science*.

4.3.3 Reducing SAT problems to Clique Problems

First, let us see how to create a clique problem out of a SAT problem. Suppose we have a formula φ in conjunctive normal form, with k conjuncts.[7] Then we will create a graph as follows:

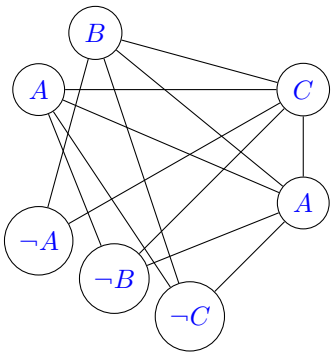
1. For each occurrence of an atomic proposition or a literal, create a vertex.
2. Draw an edge between two vertices whose atomic propositions appear in *different* clauses (conjuncts), if neither is the negation of the other.

Then you can convince yourself, the formula φ is satisfiable, if and only if this graph has a clique of size k (the number of conjuncts). Intuitively, such a clique shows how to make each of the conjuncts in the CNF true, by selecting a disjunct to make true (consistent with all the other clauses).

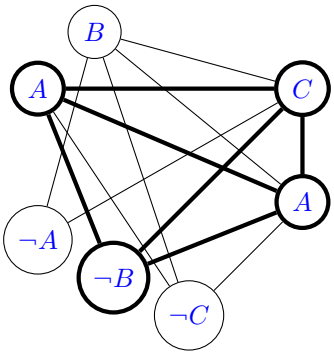
Consider the formula in CNF:

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

The recipe above tells us to construct a graph as follows. We include a vertex for each atomic proposition occurrence, and make the edges as described above:



Is there a clique of size 4 in this case? In fact, there is:



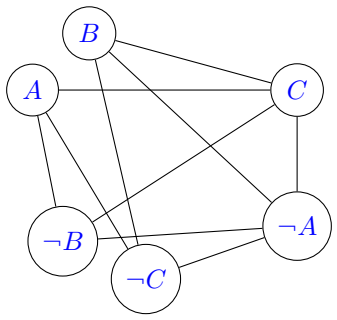
[7] Recall, a formula in CNF is a conjunction of disjunctions of literals (atomic propositions and negations thereof).

This clique points the way toward finding a row in the truth table that would make the formula above true. In this case, the row that makes A and C both true but B false will make the whole formula $(A \vee B) \wedge C \wedge (\neg A \vee \neg B \vee \neg C) \wedge A$ true. Thus, it is satisfiable.

By contrast, consider the following formula

$$(A \vee B) \wedge C \wedge (\neg B \vee \neg C) \wedge \neg A \, .$$

If we construct the corresponding graph it will look like this:



If you go through all the cases, you will be able to see that there is no clique of size 4. Hence the formula is not satisfiable (it is a contradiction). As you can also see from this example, checking to determine whether there is a clique of the size 4 takes a while! This reflects the sense in which this problem, like the satisfiability problem, is computationally difficult in general.

Exercise 4.3.1. Consider the following Boolean formula in conjunctive normal form:

$$(\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee \neg C) \wedge (C \vee D) \wedge A$$

Draw the graph for the corresponding Clique Problem. Are there any cliques of size 4? What can you conclude about whether this formula is satisfiable?

4.3.4 Reducing Clique Problems to SAT Problems

[You may ignore this section if you would like. I am just including it for completeness.]

In the other direction, suppose we have a graph (V, E) and a given number k. Assume the set V has n elements, so that we can think of $V = \{1, 2, 3, \ldots, n\}$ as consisting of the first n numbers. Let us then build a propositional logical language $\mathcal{L}$ with atomic sentences as follows:

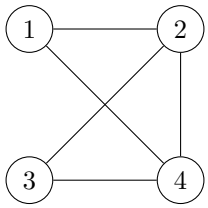
$$\begin{array}{ccccc} A_{1,2} & A_{1,3} & A_{1,4} & \cdots & A_{1,n} \\ & A_{2,3} & A_{2,4} & \cdots & A_{2,n} \\ & & A_{3,4} & \cdots & A_{3,n} \\ & & & \ddots & \vdots \\ & & & & A_{n-1,n} \end{array}$$

That is, we have a variable $A_{i,j}$ for each pair i, j with $i < j$. The intuitive interpretation is that $A_{i,j}$ should be true if there is an edge between vertices i and j. We can write down a very long propositional logic formula (in $\mathcal{L}^{bool}$) as follows:

1. Let ψ be the conjunction of all the $A_{i,j}$'s, where the atomic sentence $A_{i,j}$ is negated if there is no edge between i and j. Thus ψ describes all the connections that do (and do not) hold between vertices.

2. Let χ be the *disjunction* of all conjunctions that say some set V' of vertices form a clique of size k, i.e., that say $A_{i,j}$ for each pair i, j in the set V'.

Finally let φ be the formula $\psi \wedge \chi$. This formula essentially says that there is some clique of appropriate size. Here is an example. Return to the following instance, and suppose $k = 3$.



Then our formula, constructed as above, will first have a conjunct:

$$\psi \quad = \quad (A_{1,2} \wedge \neg A_{1,3} \wedge A_{1,4} \wedge A_{2,3} \wedge A_{2,4} \wedge A_{3,4})$$

It will also include the following disjunction:

$$\chi \quad = \quad (A_{1,2} \wedge A_{1,3} \wedge A_{2,3}) \vee (A_{1,3} \wedge A_{1,4} \wedge A_{3,4}) \vee (A_{2,3} \wedge A_{2,4} \wedge A_{3,4})$$

The final formula $\varphi = \psi \wedge \chi$ is satisfiable if and only if this graph has a clique of size 3. Indeed, this formula is satisfiable, which means that the graph does have a size 3 clique.

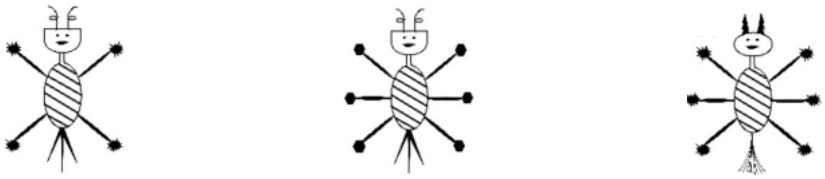
This equivalence holds in general: it is always possible to transform a clique problem into a SAT problem in this way. For many NP-hard problems, this is in fact how they are solved in practice. Because there has been so much work on optimizing SAT solving (starting in the 1970s and continuing through to today), many problems in graph theory and other areas are first translated into a logic problem,[8] and then these fast (but still imperfect, because of the aforementioned difficulties) algorithms are applied and used to answer the original question, e.g., about whether a clique of a certain size exists.

4.4 Psychology: Boolean Concepts

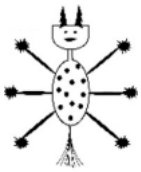
Imagine you just arrived at Stanford, and you are learning about the interesting insects we have on the San Francisco peninsula. Your friend tells you the following are all 'blipets', a commonly known

[8]For those who may wonder, it is always possible to do this in an efficient way, so that the number of atomic sentences in the logical expression is not appreciably larger than the size of the graph, for instance.

insect that especially likes to eat trash:



The following, by contrast, is not a blipet:



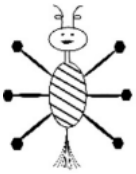
On the basis of these few observations, what might you conclude blipets are? Perhaps a reasonable guess would be: any insect with a striped body. After all, the three striped insects you have seen are blipets, while the one non-striped insect is not a blipet.

But suppose you now see two more insects, which your friend tells you are also blipets:



It looks like the simple hypothesis that blipets are simply the striped ones was wrong. But now you might slightly complicate your hypothesis: perhaps the blipets are the ones that are *either* striped, *or* with wiry antennae instead of horns on top of the head.

You next see another insect that looks like this:



Having formulated your hypothesis, you proudly assert to your friend that this must also be a blipet—it is both striped and has wiry antennae!—only to be told, no, this is actually not a blipet. From here you realize that what it takes to be a blipet is more complicated, and you end up with the following conclusion: something is a blipet if *either* (i) it is not striped *and* has wiry antennae, *or* (ii) it is striped *and either* has horns *or* a spiky tail. This hypothesis turns out to work time and time again, and you conclude that you have successfully learned what a blipet is.

Suppose we take an atom S to correspond with the property of being striped, so that $\neg S$ would correspond with not being striped (in this case, being dotted). And we similarly let W correspond

to having wiry antennae on the head, so that $\neg W$ corresponds with have horns. And finally, we let T correspond with having a spiky (three-pronged) tail, so $\neg T$ corresponds with having a bushy tail.

Using this scheme we could characterize your successive hypotheses as follows:

1. After the first four observations you hypothesized S.
2. Following the next two you hypothesized $S \vee W$.
3. Upon the seventh observation you concluded $(\neg S \wedge W) \vee \big(S \wedge (\neg W \vee T)\big)$.

In other words, we can characterize the different possible concepts in this simple repertoire using Boolean formulas. A formula φ in this little language picks out a concept in the sense that some objects "fall under" (satisfy) the formula, while others do not.

This way of thinking about concepts and categories is quite general for cases where falling under the concept amounts to the presence or absence of various (perhaps combinations of) properties or features. For example, in baseball what distinguishes the concept of a *strike*? It is *either* a swing *and* a miss, *or* a pitch down the middle *without* a swing, *or* a foul ball with fewer than two strikes on count. Once you start looking, examples abound. With sufficient training, and sometimes explicit direction, people are able to learn many concepts of this sort.

The psychological literature on (Boolean and other) concepts is enormous. We report here on just one notable finding that relates to what we have already studied. It concerns how empirically *difficult* it is for people to learn different concepts.

Given only a finite number of (on/off) features, represented by a finite list of propositional atoms (e.g., in the example above, this was S, W, T), we can characterize any concept by a certain formula in disjunctive normal form: simply write out explicitly each possible combination of features that a member of the category could possibly have. For instance, in our example from above, our final concept could be characterized by:

$$(\neg S \wedge W \wedge T) \vee (\neg S \wedge W \wedge \neg T) \vee (S \wedge \neg W \wedge \neg T) \vee (S \wedge W \wedge T) \vee (S \wedge \neg W \wedge T)$$

You should verify that all of the positive instances of blipet "satisfy" this formula, while none of the negative instances do. Next consider the *shortest* (in terms of number of symbols) Boolean formula that is logically equivalent to that DNF formula; call it φ^*. In this case that happens to be the formula we wrote above:

$$\varphi^* \quad = \quad (\neg S \wedge W) \vee \big(S \wedge (\neg W \vee T)\big)$$

In a provocative paper that appeared in *Nature* in 2000,[9] psychologist Jacob Feldman showed that much of the difficulty people encounter in learning concepts—in data collected over many decades of research—could be explained simply in terms of the length of φ^*: this simplest Boolean formula logically equivalent to the DNF for a given concept. While the precise significance of this finding has been a matter of controversy, it is at least suggestive of the possibility that rather sophisticated aspects of Boolean Logic are at work deep within human psychology!

[9] See Feldman (2000), "Minimization of Boolean complexity in human concept learning," *Nature*.

4.5 Mathematics: Boolean Algebra

If we think about Boolean Logic semantically, as a system for reasoning about truth-functions, then it becomes natural to identify any two Boolean formulas $\varphi, \psi \in \mathcal{L}^{bool}$ that are logically equivalent. For instance, $A \vee \neg B$ and $\neg(\neg A \wedge B)$, being equivalent, encode the same truth-functions. Let us define a special relation $\leqslant$ on $\mathcal{L}^{bool}$ so that $\varphi \leqslant \psi$ just in case $\varphi \Vdash \psi$. Then say that $\varphi \equiv \psi$ if both $\varphi \leqslant \psi$ and $\psi \leqslant \varphi$. Evidently $\equiv$ is an **equivalence relation** (reflexive, symmetric, and transitive). Let us then consider the set $\mathcal{L}^{bool}_{\equiv}$ of **equivalence classes** of formulas. That is:

$$\mathcal{L}^{bool}_{\equiv} \quad \triangleq \quad \{[\varphi] \mid \varphi \in \mathcal{L}^{bool}\}$$

where

$$[\varphi] \quad \triangleq \quad \{\psi \mid \varphi \equiv \psi\}$$

is the equivalence class of φ under $\equiv$. We can then lift the relation $\leqslant$ to equivalence classes to say that $[\varphi] \leqslant [\psi]$ just in case $\varphi \Vdash \psi$.

Exercise 4.5.1. Show that $\leqslant$ is well-defined on $\mathcal{L}^{bool}_{\equiv}$. That is, show that if $\varphi_1 \equiv \varphi_2$ and $\psi_1 \equiv \psi_2$, then $\varphi_1 \Vdash \psi_1$ iff $\varphi_2 \Vdash \psi_2$.

How many equivalence classes would there be if we only have a single propositional atom A in our language $\mathcal{L}^{bool}$? If you think about it you will see that there are exactly four, corresponding to the following diagram, in which the arrow $\rightarrow$ signifies $\leqslant$:[10]

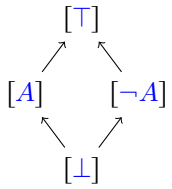


Figure 4.2: The algebra over a single atom A

There are thus only four non-equivalent formulas in this language. Any formula is equivalent to one of these. For instance, $(A \wedge \neg A) \equiv \bot$, and so on.

What if we had two atoms: $\mathsf{Prop} = \{A, B\}$? How many non-equivalent formulas would there be in this slightly extended language? Remembering back to our discussion of truth-functional completeness, we know that with two atoms we can express all (and only) the binary truth-functions $f : \{\mathsf{T}, \mathsf{F}\} \times \{\mathsf{T}, \mathsf{F}\} \rightarrow \{\mathsf{T}, \mathsf{F}\}$. Because there are four pairs of truth values, there are $2^4 = 16$ functions from pairs of truth values to truth values. A further question is, what does the structure of this set of (equivalence classes of) formulas with the relation $\leqslant$ look like? Here is a picture:

[10] In the following, let us take $\top$ to be an abbreviation of $\neg\bot$. I.e., it is a formula that is always true.

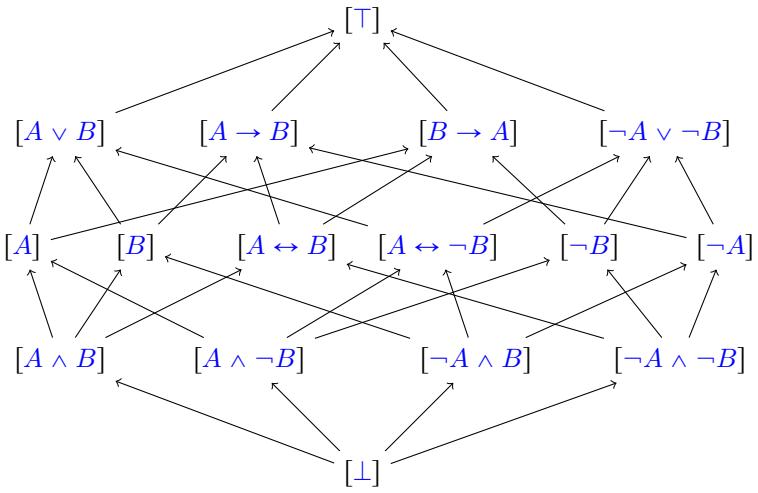


Figure 4.3: The algebra over atoms A, B

It looks similar to the case with one atom, but is slightly larger. The representatives we chose for the equivalence classes are of course somewhat arbitrary. We could have, e.g., chosen $\neg A \vee B$ instead of $A \to B$, and so on. But the structure itself is quite canonical.

We can think of this structure in Fig. 4.3 as an ordered set $(\mathcal{L}^{bool}_{\equiv}; \leqslant)$, but we could also think of it as an algebraic structure $(\mathcal{L}^{bool}_{\equiv}; \wedge, \vee, \neg)$, with special operations $\wedge, \vee, \neg$ taking elements of $\mathcal{L}^{bool}_{\equiv}$ to new elements of $\mathcal{L}^{bool}_{\equiv}$:

$$\begin{aligned} [\varphi] \wedge [\psi] &= [\varphi \wedge \psi] \\ [\varphi] \vee [\psi] &= [\varphi \vee \psi] \\ \neg[\varphi] &= [\neg\varphi] \end{aligned}$$

Then, for instance, $[B \to A] \wedge [\neg A \vee \neg B] = [\neg B]$. Indeed, this works even when we have infinitely many propositional atoms: the structure $(\mathcal{L}^{bool}_{\equiv}; \wedge, \vee, \neg)$ is well defined for any set $\mathcal{L}^{bool}$. In any case, the resulting structure can be shown to satisfy all of the following laws:

(B1)	$x \wedge y = y \wedge x$	$x \vee y = y \vee x$
(B2)	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	$x \vee (y \vee z) = (x \vee y) \vee z$
(B3)	$x \wedge (x \vee y) = x$	$x \vee (x \wedge y) = x$
(B4)	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
(B5)	$x \wedge (y \vee \neg y) = x$	$x \vee (y \wedge \neg y) = x$

where x, y, z stand for arbitrary equivalence classes of formulas. Generalizing from the specific context of Boolean Logic, we can say abstractly that any structure of the form $(X; \wedge, \vee, \neg)$, with X any non-empty set and $\wedge, \vee, \neg$ operations on this set satisfying the axioms (B1)-(B5) above, is to be called a **Boolean algebra**. We can recover the kind of directed diagram as in Fig. 4.3 by defining $x \leqslant y$ just in case $x \vee y = y$ (or equivalently, just in case $x \wedge y = x$).

Exercise 4.5.2. Explain why this definition makes sense in the context of Boolean Logic. In other words, show that in fact, $[\varphi] \leqslant [\psi]$ just in case $[\varphi] \vee [\psi] = [\psi]$.

What are some other examples of Boolean algebras? It turns out they are everywhere! One very simple example arises from the structure $\mathbf{2} = (\{0,1\}; \wedge, \vee, \neg)$, where we define $x \wedge y = x \cdot y$ for "conjunction," $x \vee y = \max(x, y)$ for "disjunction," and $\neg x = 1 - x$ for "negation." Defining $\leqslant$ as above, this structure is quite small indeed:



Figure 4.4: The two-element algebra **2**

A quite different example comes from our topic in Chapter 2, namely ordinary reasoning about sets. Given a set S, we are interested in an algebraic structure over $\wp(S)$, the *powerset* of S. This can be obtained by defining the operations $\wedge$, $\vee$, and $\neg$ directly on subsets of S:

$$\begin{aligned} X \wedge Y &= X \cap Y \\ X \vee Y &= X \cup Y \\ \neg X &= S - X \end{aligned}$$

It turns out this defines a Boolean algebra for any set S whatsoever!

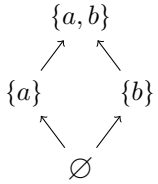
Exercise 4.5.3. Show that (B1)-(B5) all hold. [Recall Fact 2.1.1 and Ex. 2.1.1!]

What do these structures look like? First, note that $\leqslant$ in this case amounts to $\subseteq$. After all we have $X \cup Y = Y$ exactly when $X \subseteq Y$ (and when $X \cap Y = X$).

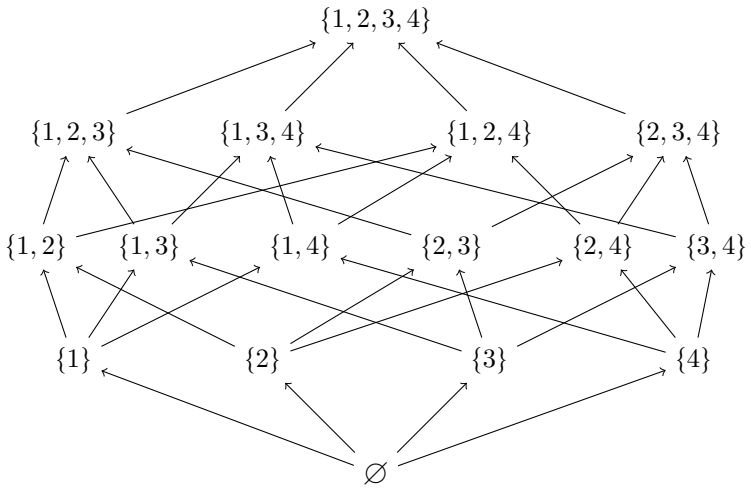
Now consider first a singleton set $S = \{s\}$. This looks just like the structure **2**:



What if we have two elements, $S = \{a, b\}$? Then we obtain a structure which looks quite like the single-atom case arising from Boolean Logic:



For an even larger example, consider a four-element set, say, $S = \{1, 2, 3, 4\}$:



This again should look familiar from the case we already considered of $\mathcal{L}^{bool}$-equivalence classes with two propositional atoms (Fig 4.3). In fact, the two structures are indistinguishable (in technical jargon, they are *isomorphic*). This is no accident. There is a deep mathematical result showing that *any* Boolean algebra whatsoever—not just those arising from Boolean Logic—is structurally identical to a Boolean algebra of subsets of some set S under $\cap$, $\cup$, and $-$:

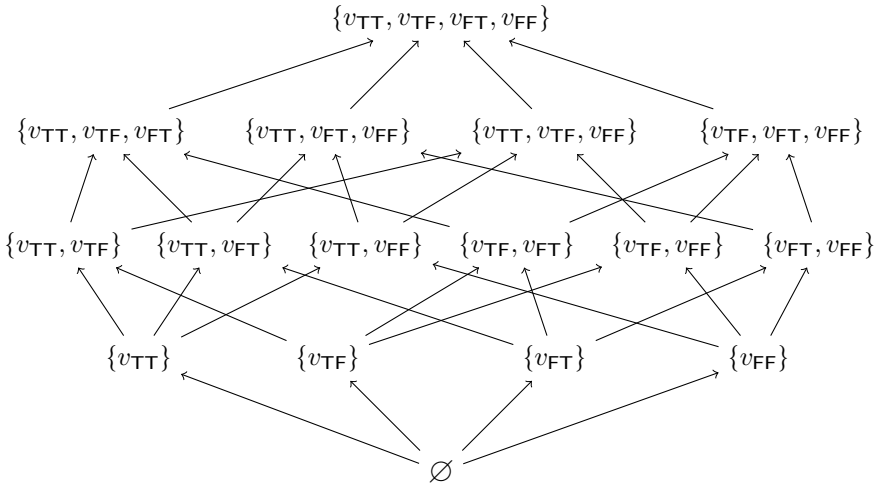
Theorem 4.5.1 (Stone Representation)**.** Every Boolean algebra is isomorphic to a field of sets.

The proof uses techniques and concepts we have not yet covered. But we can get some sense of the proof by considering the Boolean Logic cases we have already seen. If you had a Boolean algebra defined by equivalence classes of formulas, what would be an obvious set for which the subsets correspond one-to-one with equivalence classes of formulas?

First, suppose again we just have a single atom A. We know we need to find a set that corresponds with the four element Boolean algebra above. Suppose we take this set to be all the valuations defined on this language, which in this case includes just two: $v_{\mathsf{T}}(A) = \mathsf{T}$ and $v_{\mathsf{F}}(A) = \mathsf{F}$. Then, indeed, our two structures correspond quite directly in this case (as we have essentially already seen):

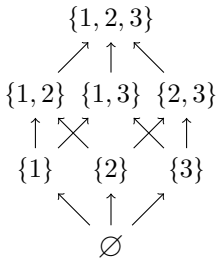


The correspondence is of course that each formula φ is associated with exactly the set of valuations that make φ true. This easily extends to cases with more propositional atoms as well. For instance, suppose we write v_{ab}, with $a, b \in \{\mathsf{T}, \mathsf{F}\}$, to mean that $v_{ab}(A) = a$ and $v_{ab}(B) = b$. Then our two element set whose powerset algebra will correspond with the Boolean logical algebra arising from two atoms $\{A, B\}$ looks as follows:

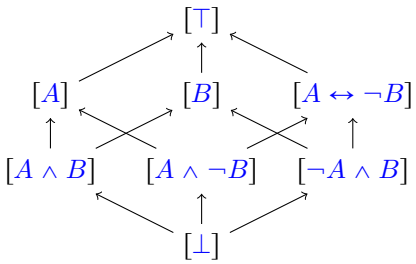


Indeed, this correspondence even works when Prop is infinite (though it becomes harder to depict).

You might wonder what happened to the powerset algebra from a set with three elements, which we seemed to pass in going from one to two propositional atoms. This looks like the following:



Evidently, we cannot obtain this structure straight from a Boolean language $\mathcal{L}^{bool}$ with some number of propositional atoms. However, we can obtain this structure if we help ourselves to a notion of equivalence *modulo a theory*. Recall that before we defined equivalence $\varphi \equiv \psi$ by $\varphi \dashv\vdash \psi$, i.e., $\vdash \varphi \leftrightarrow \psi$ (by Fact 3.5.1). Given a set Γ of formulas, let us define **equivalence modulo** Γ, written $\varphi \equiv_\Gamma \psi$, so that this holds when $\Gamma \vdash \varphi \leftrightarrow \psi$. Everything else is as before. But now we see that this can give us a smaller algebra. Suppose we again take the language with two atoms. And let $\Gamma = \{A \vee B\}$. Then, modulo Γ, new equivalences hold. For instance, $A \to B \equiv_\Gamma B$:



In this way we can obtain any finite Boolean algebra. Moreover, starting with an infinite set of propositional atoms gives us an infinite Boolean algebra.

Exercise 4.5.4. Suppose instead of the formula $A \vee B$, the theory Γ consisted of the formula $A \leftrightarrow B$ (but again in the language with two atoms A and B). Draw the corresponding diagram in this case.

Exercise 4.5.5. Suppose now that we have three propositional atoms, A, B, and C. Provide a theory Γ (a set of formulas, perhaps just one formula) such that the set of $\equiv_\Gamma$-equivalence-classes has exactly 32 elements, i.e., such that the algebraic structure corresponds directly to (is isomorphic to) the powerset algebra over a 5 element set. [You do not need to draw the structure! But do explain why your theory does the trick.]

It turns out that, even in plain Boolean logic, there are many subtle algebraic notions lurking at every turn. Consider, for example, the operation κ of taking an n-ary truth-function $f(A_1, \ldots, A_n)$ and negating each of the atoms, so that

$$\kappa\big(f(A_1, \ldots, A_n)\big) \quad = \quad f(\neg A_1, \ldots, \neg A_n)$$

For instance, $\kappa\big([A \wedge \neg B]\big) = [\neg A \wedge B]$. Let us further define ι to be the identity function, and define two more functionals (a *functional* is a function that takes functions as arguments) on truth-functions/equivalence-classes as follows:

$$\begin{aligned} \theta\big(f(A_1, \ldots, A_n)\big) &= \neg f(A_1, \ldots, A_n) \\ \delta\big(f(A_1, \ldots, A_n)\big) &= \neg f(\neg A_1, \ldots, \neg A_n) \end{aligned}$$

So, for example, $\theta\big([A \wedge \neg B]\big) = [A \to B]$, while $\delta\big([A \wedge \neg B]\big) = [B \to A]$. One can then show that these four functions, κ, ι, θ, and δ form a **group** under function composition. And not just any group, but the Klein four-group, the unique (up to isomorphism) non-cyclic group of order four!

$\circ$	ι	κ	θ	δ
ι	ι	κ	θ	δ
κ	κ	ι	δ	θ
θ	θ	δ	ι	κ
δ	δ	θ	κ	ι

Further algebraic structure comes with even richer languages. Algebraic structures in modal logic—modal logic is our next topic—give rise to a rich area of study, which is currently an active area of research. Particularly in that setting there are also deep connections with other areas of mathematics, such as general topology.

Chapter 5

Modal Logic

Recall in §4.1.2 the observation that the simple Boolean (truth-functional) connectives do not allow us to say anything about time or knowledge. The Boolean connectives just combine truth values together and return some other truth value. In this part of the course we will develop the basics of modal logic, including interpretations of this logic that allow us to talk about time and knowledge. In addition to reasoning about what is *actually* true or false, modal logic allows us to reason about what *might be* true or false in some given (possibly non-actual) states of affairs. From a mathematical point of view, it will allow us to reason about more complex structures than simple valuations.

5.1 General Modal Logic

5.1.1 The Language of Modal Logic

As in §3, let us assume we have some infinite set $\mathsf{Prop} = \{A_1, A_2, \dots\}$ of propositional atoms. Recall a valuation $v : \mathsf{Prop} \to \{\mathsf{T}, \mathsf{F}\}$ is a function sending each of the atoms to T or F. We might think of a valuation as specifying a given state of affairs. Each of the atomic "facts" A_1, A_2, ..., can be either true or false; and a valuation v tells us the truth value of each one. Modal logic will allow us to reason not just about a single valuation ("state of affairs"), but also about alternative valuations (alternative "states of affairs"): about what could be the case, about what will be the case, and so on. To do this we extend our modal language $\mathcal{L}^{bool}$ to a richer language with two new **modal operators** $\Box$ and $\Diamond$, as follows:

Definition 5.1.1. The language $\mathcal{L}^{\Box\Diamond}$ of basic modal logic is given by the following set of wffs:

1. Every propositional atom $A \in \mathsf{Prop}$ is a wff in $\mathcal{L}^{\Box\Diamond}$.
2. If φ and ψ are wffs in $\mathcal{L}^{\Box\Diamond}$, then so are:
   (a) $(\varphi \wedge \psi)$
   (b) $\neg\varphi$
   (c) $(\varphi \vee \psi)$
   (d) $(\varphi \to \psi)$
   (e) $\Box\varphi$
   (f) $\Diamond\varphi$

3. Nothing else is a wff in $\mathcal{L}^{\Box\Diamond}$.

Thus, the language $\mathcal{L}^{\Box\Diamond}$ includes all of the plain Boolean formulas that we had before, in addition to formulas like the following:

$$\Diamond\neg A \qquad \big(\Box(\Box A \to A) \to \Box A\big) \qquad (A \vee \Diamond B) \qquad (A \to \Diamond A) \qquad \Box\Diamond\Box B$$

$$\big((\Diamond A \wedge \Diamond B) \to (\Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A) \vee \Diamond(A \wedge B))\big) \qquad \big(\Box(A \to \neg B) \to (\Box A \to \Box\neg B)\big)$$

And so on. Intuitively, you might think of $\Diamond A$ as meaning, "A might be the case," or "A is true in *some* alternative state of affairs," and $\Box A$ as meaning, "A must be the case," or "A is the case in *all* alternative states of affairs." We now make these intuitions more precise.

5.1.2 Modal Models

To determine whether an arbitrary Boolean formula φ is true, all we need is a valuation saying which of the propositional atoms is true. The connectives then determine automatically whether φ is true. In modal logic we need more than just a single valuation. We define:

Definition 5.1.2. A **modal model** is a triple (S, R, V), where:

- S is some set of points, intuitively possible "states of affairs."
- $R \subseteq S \times S$ is a relation on S, usually called the **accessibility relation**.
- $V : \mathsf{Prop} \to \wp(S)$ is a **modal valuation**, specifying which atoms are true at which points.

What exactly S is—that is what the "states of affairs" are supposed to be—and what R is—what the relation of accessibility is supposed to be—will vary from application to application. For now, we will simply work with modal logic as a formal symbolic system.

Suppose for the moment that we have just three propositional atoms $\mathsf{Prop} = \{A, B, C\}$, instead of the usual infinite set. An example of a modal model is the following. This is a picture of a modal

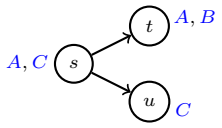


Figure 5.1: A simple model $\mathcal{M}_0 = (S_0, R_0, V_0)$

model we can write more formally—being of the form (S_0, R_0, V_0)—where:

- $S_0 = \{s, t, u\}$ are the states.
- R_0st and R_0su; i.e., $R_0 = \{\langle s, t\rangle, \langle s, u\rangle\}$.
- $V_0(A) = \{s, t\}$, $V_0(B) = \{t\}$, and $V_0(C) = \{s, u\}$.

We have three states, s, t, and u; t and u are both accessible from s; and V_0 specifies that A is true at states s and t, B is true only at t, and C is true at s and u.

Instead of determining truth of a formula relative to a Boolean valuation, we now define truth of a formula **in a model, at a point**:

Definition 5.1.3. The **truth** of $\varphi \in \mathcal{L}^{\Box\Diamond}$ in a model $\mathcal{M} = (S, R, V)$, at a point $s \in S$—written $\mathcal{M}, s \models \varphi$—is defined as follows:

1. $\mathcal{M}, s \models A$, if and only if $s \in V(A)$ $\big($whenever $A \in \mathsf{Prop}\big)$.
2. $\mathcal{M}, s \models \varphi \wedge \psi$, if and only if both $\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models \psi$.
3. $\mathcal{M}, s \models \neg\varphi$, if and only if it does not hold that $\mathcal{M}, s \models \varphi$.
4. $\mathcal{M}, s \models \varphi \vee \psi$, if and only if either $\mathcal{M}, s \models \varphi$ or $\mathcal{M}, s \models \psi$.
5. $\mathcal{M}, s \models \varphi \to \psi$, if and only if either $\mathcal{M}, s \models \varphi$ does *not* hold, or $\mathcal{M}, s \models \psi$ does hold.
6. $\mathcal{M}, s \models \Box\varphi$, if and only if $\mathcal{M}, t \models \varphi$ holds for *every* $t \in S$ such that Rst.
7. $\mathcal{M}, s \models \Diamond\varphi$, if and only if $\mathcal{M}, t \models \varphi$ holds for *some* $t \in S$ such that Rst.

These clauses allow us to determine the truth/falsity of any modal formula, at any point in any modal model. For instance, let us return to the small 3-point model in Fig. 5.1.2.

Note first that by clause 1 of Def. 5.1.3, we have, for example, $\mathcal{M}_0, s \models A$, while $\mathcal{M}_0, s \models \neg B$ by clauses 1 and 3. Likewise, $\mathcal{M}_0, t \models A \wedge B$, while $\mathcal{M}_0, u \models \neg A \wedge \neg B \wedge C$, for instance.

What about more interesting modal formulas like $\Diamond B$? We do have that $\mathcal{M}_0, s \models \Diamond B$. This is because there is a point, namely t, such that R_0st, and $\mathcal{M}_0, t \models B$. We also have $\mathcal{M}_0, s \models \Diamond C$, because there is a point, namely u, such that R_0su, and $\mathcal{M}_0, t \models C$. However, we do not have, e.g., $\mathcal{M}_0, s \models \Diamond(A \wedge C)$, because there is no accessible point where both A and C hold. In particular, even though $A \wedge C$ is true at point s, we do not have R_0ss in this model. So we actually have $\mathcal{M}_0, s \models (A \wedge C) \wedge \neg\Diamond(A \wedge C)$.

What about at point u, for example? Which modal formulas are true there? Do we have, e.g., $\mathcal{M}_0, u \models \Diamond A$? No. For this to hold we would need some point accessible from u where A holds. But there are no points accessible from u at all. In fact, this means we do not even have $\mathcal{M}_0, u \models \Diamond(A \vee \neg A)$. However, strange though it may seem, we do have $\mathcal{M}_0, u \models \Box A$. In fact we also have $\mathcal{M}_0, u \models \Box\neg A$. These both hold vacuously: for $\Box\varphi$ to be true, it must be that any accessible point (if there are any) satisfies φ. Since there are none we have $\mathcal{M}_0, u \models \Box\varphi$ for any φ whatsoever! Thus, in particular we have $\mathcal{M}_0, u \models \Box\bot$. Here are some more facts that hold on this model:

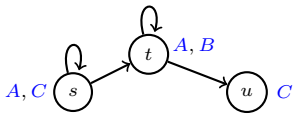
$\mathcal{M}_0, s \models \Box(B \vee C)$ $\qquad$ $\mathcal{M}_0, u \models \Diamond C \to \bot$

$\mathcal{M}_0, s \models \Diamond\Box\bot$ $\qquad$ $\mathcal{M}_0, t \models A \wedge \Box\neg A$

$\mathcal{M}_0, s \models \Box\Box\bot$ $\qquad$ $\mathcal{M}_0, s \models \Box\big(\Box\bot \wedge (A \vee C)\big)$

Do you see why all of these hold? Here is an exercise with a slightly different model to ensure you understand the basic definitions:

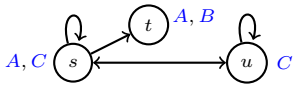
Exercise 5.1.1. Consider the following modal model $\mathcal{M}_1 = (S_1, R_1, V_1)$:



That is, $S_1 = S_0 = \{s, t, u\}$ again, and $R_1 = \{\langle s, s\rangle, \langle s, t\rangle, \langle t, t\rangle, \langle t, u\rangle\}$, while the valuation $V_1 = V_0$ is the same as in $\mathcal{M}_0$. Which of the following hold on this model? Briefly explain why or why not.

1. $\mathcal{M}_1, s \models \Diamond\Diamond C$
2. $\mathcal{M}_1, s \models \Box\Box C$
3. $\mathcal{M}_1, s \models \Diamond\Diamond B$
4. $\mathcal{M}_1, s \models \Diamond\Box(B \vee C)$
5. $\mathcal{M}_1, t \models \Diamond\Box\bot$
6. $\mathcal{M}_1, t \models \Box\Box(B \vee C)$
7. $\mathcal{M}_1, s \models \Diamond\Diamond\Diamond\Diamond(A \wedge C)$
8. $\mathcal{M}_1, t \models \Diamond\Box\Diamond(A \vee C)$
9. $\mathcal{M}_1, s \models \Box(B \to \Diamond C)$
10. $\mathcal{M}_1, s \models \Box(C \to \Diamond\top)$
11. $\mathcal{M}_1, t \models \Box(C \to \Diamond\top)$
12. $\mathcal{M}_1, s \models \Box(\Box C \to \bot)$

Exercise 5.1.2. Answer the same 12 questions as above, but where the model is given as follows:



Like in the case of Boolean Logic, we can define a notion of **validity** for modal formulas:

Definition 5.1.4. A formula $\varphi \in \mathcal{L}^{\Box\Diamond}$ is **valid** if $\mathcal{M}, s \models \varphi$ for *every* model $\mathcal{M}$ and state s.

Fact 5.1.1. The following formulas are all valid:

1. $\Diamond\varphi \leftrightarrow \neg\Box\neg\varphi$
2. $\Box\varphi \leftrightarrow \neg\Diamond\neg\varphi$
3. $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
4. If φ is valid, then so is $\Box\varphi$.

Proof. To see why 1 holds, suppose $\mathcal{M}, s \models \Diamond\varphi$ for some model $\mathcal{M}$ and point s. Then there is some R-accessible point t from s such that $\mathcal{M}, t \models \varphi$. But that just means it is *not* the case that all accessible points from s *fail* to make φ true. Hence $\mathcal{M}, s \models \neg\Box\neg\varphi$. We can argue for the other direction, and for the two directions of 2, similarly.

For 3, suppose at some point s in a model $\mathcal{M}$ we have $\mathcal{M}, s \models \Box(\varphi \to \psi)$. That is to say,

($*$) For any point t such that Rst, if $\mathcal{M}, t \models \varphi$ then $\mathcal{M}, t \models \psi$.

Now suppose $\mathcal{M}, s \models \Box\varphi$. We must show that $\mathcal{M}, s \models \Box\psi$ as well. But this is easy. Take any t such that Rst. Because $\mathcal{M}, s \models \Box\varphi$, we know in particular that $\mathcal{M}, t \models \varphi$. But by $(*)$ above, we know that therefore $\mathcal{M}, t \models \psi$. Since t was an *arbitrary* accessible point from s, we have shown that ψ is true at any accessible point whatsoever, i.e., that $\mathcal{M}, s \models \Box\psi$. Hence $\mathcal{M}, s \models \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$. As s and $\mathcal{M}$ themselves were arbitrary (i.e., we did not assume anything in particular about them), we have shown that $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$ is valid.

For 4, suppose φ is valid: true at all points in all models. We need to show $\Box\varphi$ is also valid. Well, choose any model $\mathcal{M}$ and point s in $\mathcal{M}$. Now take any point t accessible from s in $\mathcal{M}$. Evidently $\mathcal{M}, t \models \varphi$. Hence $\mathcal{M}, s \models \Box\varphi$. □

Remark on Valuations and Boolean/Modal Validity: We said at the beginning of this chapter that we might think of each state as harboring its own valuation. In what sense can we say this? Consider a point s in a model $\mathcal{M} = (S, R, V)$. We could define a valuation v_s on Prop as follows:

$$v_s(A) = \begin{cases} \mathsf{T} & \text{if } s \in V(A) \\ \mathsf{F} & \text{if } s \notin V(A) \end{cases}$$

Then the truth/falsity of a formula without $\Diamond$ or $\Box$ is determined just as in Boolean logic: in particular you can convince yourself that for $\varphi \in \mathcal{L}^{bool}$ we will always have

$$\mathcal{M}, s \models \varphi, \text{ if and only if } v_s \models \varphi$$

Thus, clearly any Boolean tautology will also be modally valid. Moreover, there are formulas also with $\Diamond$ and $\Box$ that behave like tautologies. Consider, e.g., $\Diamond A \vee \neg\Diamond A$. This is also valid, since at any point either $\Diamond A$ will hold, or $\Diamond A$ will fail to hold, in which case $\neg\Diamond A$ will hold. Indeed, it is possible to show that if you take a formula φ and *uniformly replace* each subformula of the form $\Diamond\psi$ or $\Box\psi$ with a new, "fresh" propositional atom,[1] then if the resulting formula is a Boolean tautology, the original formula φ was valid. For instance, if we apply this procedure to the formula

$$\big(\Diamond A \to (\Box\Box B \to \Diamond A)\big)$$

we will obtain a new formula $\big(C \to (D \to C)\big)$, which as we know is valid in Boolean Logic.

5.1.3 Frames & Definability

From a mathematical point of view, a modal model $\mathcal{M} = (S, R, V)$ is just a **colored graph**: the points S are the nodes, R is an edge relation, and V is a coloring, in the sense that if you think of each atom $A \in \mathsf{Prop}$ as a color, then some points/nodes are colored with A (make A true), while other do not. If we forget about the valuation, then we are left with simply a graph. In modal logic these are called **frames**, and they are one of the most important types of objects in modal logic:

Definition 5.1.5. A **modal frame** is a pair $\mathcal{F} = (S, R)$, i.e., just a model without a valuation.

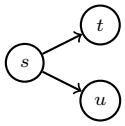
[1] By uniform replacement we mean, e.g., if $\Diamond A$ occurs several times, we replace it with the same atom each time.

It turns out that simple modal formulas can **define** classes of frames, that is, interesting classes of graphs. Analogous to validity on modal models, we can also define notions of validity for frames:

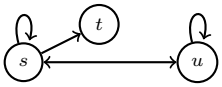
Definition 5.1.6. A formula φ is **valid on a frame** $\mathcal{F} = (S, R)$ if it is true at every point $s \in S$ in every model $\mathcal{M} = (S, R, V)$ built on $\mathcal{F}$. We write this $\mathcal{F} \models \varphi$.

In other words, we have $(S, R) \models \varphi$, just in case for *every* valuation $V : \mathsf{Prop} \to \wp(S)$ and *every* point $s \in S$, we have $(S, R, V), s \models \varphi$.

For example, consider the frame $\mathcal{F}_0 = (S_0, R_0)$ underlying the model above in Fig. 5.1.2:

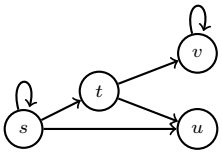


What is an example of a formula that is valid in this frame, but perhaps not valid in all frames (or valid in general)? Here is one: $\Box\bot \vee \Box\Box\bot$. Certainly this formula is not even valid on the underlying frame from the other models we considered:



What is an example of a formula valid on this frame, but not on the other? How about $\Diamond A \to \Diamond\Diamond\Diamond A$? Another example is $\Box\bot \vee \Diamond\Diamond\Box\bot$.

Exercise 5.1.3. Consider the following frame:



Identify at least one formula that is valid on this frame, but not valid on either of the frames above.

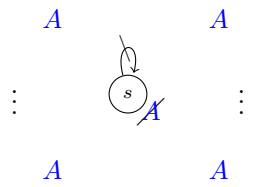
Obviously every formula that is modally valid is also valid on any frame. But if we restrict to interesting *classes* of frames, we obtain interesting new validities. Here is a neat example of a simple formula **characterizing** a natural class of graphs:

Fact 5.1.2. $(S, R) \models \Box A \to A$ if and only if R is reflexive.

Proof. Suppose first that R is reflexive, and let us show that $\Box A \to A$ is valid on the frame. Take any state $s \in S$ and valuation $V : \mathsf{Prop} \to \wp(S)$, and assume $(S, R, V), s \models \Box A$. This means for every t with Rst we have $(S, R, V), t \models A$. But because R is reflexive we know in particular that Rss! So $(S, R, V), s \models A$, and hence $(S, R, V), s \models \Box A \to A$. Because s and V were arbitrary, we have $(S, R) \models \Box A \to A$.

We will prove the other direction by contraposition: suppose R is *not* reflexive, and show that $\Box A \to A$ is not valid on the frame. If R is not reflexive, this means there is at least one point, call it s, such that Rss does *not* hold. To ensure $\Box A \to A$ we need to find a point and a valuation to

falsify it. The point will be s. For the valuation, we will define it as follows. In general we know nothing else about the frame (S, R), except that there is this non-reflexive point s. But to define a fully specified valuation, we need to say exactly which atoms are true at which points. Here is one way to do that. However the rest of (S, R) looks, I want to define V so that $V(A) = S - \{s\}$ and $V(B) = \varnothing$ for all other atoms $B \in \mathsf{Prop}$. That is, we are making A true everywhere except for s, and all other atoms false everywhere. In a schematic picture:



First note that, however (S, R) looks, we are guaranteed that $(S, R, V), s \models \Box A$. This is because every other point in S satisfies A, and we know in particular that Rss does not hold. But also $(S, R, V), s \models \neg A$, since A is false at s. Hence we do not have $(S, R, V), s \models \Box A \to A$; we have falsified the formula, showing that $\Box A \to A$ is not valid on (S, R). □

Fact 5.1.3. $(S, R) \models \Box A \to \Box\Box A$, iff R is transitive.

Proof. We go through this proof a bit more quickly. First suppose (S, R) is transitive, and assume $(S, R, V), s \models \Box A$. We need to show $(S, R, V), s \models \Box\Box A$. Well take any t such that Rst. If we can show that $(S, R, V), t \models \Box A$, we will be done. So take any u such that Rtu. Because R is transitive we know Rsu as well. Hence, because $(S, R, V), s \models \Box A$, we have $(S, R, V), u \models A$. Therefore $(S, R, V), t \models \Box A$, and thus $(S, R, V), s \models \Box\Box A$.

In the other direction, suppose R is not transitive, so that there are points s, t, and u such that Rst and Rtu but not Rsu. Let us consider a valuation V such that $V(A) = S - \{u\}$ and $V(B) = \varnothing$ for all other B. Then note that $(S, R, V), s \models \Box A$. This is guaranteed because every point except for u makes A true, and u is not accessible from s. However, we have $(S, R, V), t \models \neg\Box A$, because u is accessible from t. Therefore, it is not true that $(S, R, V), s \models \Box\Box A$. Hence we have shown that $\Box A \to \Box\Box A$ is false at s in (S, R, V), and is thus invalid on (S, R). □

Exercise 5.1.4. Show that $(S, R) \models A \to \Box\Diamond A$, iff R is symmetric.

Exercise 5.1.5. Show that $(S, R) \models \Diamond A \to \Box\Diamond A$, iff R is euclidean.

Exercise 5.1.6. A relation is *dense* if for each two points s and t such that Rst there is a third point u such that Rsu and Rut. Show that R is dense if and only if $(S, R) \models \Box\Box A \to \Box A$.

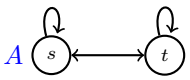
Exercise 5.1.7. A relation is called *serial* if for each s there is some t such that Rst. Show that R is serial, if and only if $(S, R) \models \Box A \to \Diamond A$.

Not all properties of relations are definable by formulas in this basic modal language. For instance, it is relatively easy to show that *ir*reflexivity of a relation cannot be defined. There is a whole theory of frame definability, which we do not develop here. Students who are interested might want to consider the course on modal logic, Phil 154. In the next two sections we consider two concrete applications.

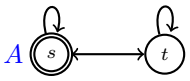
5.2 Epistemic Logic

According to the epistemic interpretation of modal logic, the $\Box$ operator is understood as referring to what a given agent *knows*. When we write $\Box A$, on a first pass at least, this should be read as, "The agent knows that A." To make it clear that this is our intended interpretation let us write K instead of $\Box$. Epistemic logic dates back in philosophy to the work of the Finnish philosopher Jaakko Hintikka in the mid 1960s. However, it has been rediscovered several times since in other fields. Its rediscovery and application in economics were part of what earned Robert Aumann a Nobel Prize in that field. It has also been used profitably in computer science to analyze and reason about what distributed software agents (do not) know about each other.

Given this intended reading of K, how should we think of modal models in this context? The idea is simple, but surprisingly powerful. In a model (S, R, V), let us think of the states S as being possible states of world, *as far as our agent knows*. And let us think of R as being a relation of *epistemic indistinguishability*: Rst holds if in state s the agent cannot tell that we are *not* in state t. Here is a very simple example. Let A stand for the propositional atom, "It is raining in Topeka." I happen not to know whether it is raining in Topeka, so my epistemic situation can be depicted:



We have two possible states, s and t, with A true at s and false at t. Suppose we are in state s, so that it is raining in Topeka. For all I know we could be in a world where it is not raining in Topeka, hence the arrow from s to t. I also do not know anything that would rule out the actual situation, s, so there should also be an arrow from s back to s. In fact, the situation is completely analogous if the true state is t: I cannot determine whether we are in s or t, so there are arrows from t both to s and back to t. If we want to suppose that there is some *ground fact* about a situation, we can draw the actual state with a double circle, meaning s is the actual state, but as far as the agent knows, we could be in any of these states:



If we call this model $\mathcal{M}$, then we have $\mathcal{M}, s \models \neg\mathsf{K}A$, but also $\mathcal{M}, s \models \neg\mathsf{K}\neg A$. Our agent neither knows A nor $\neg A$. (Though of course, as usual, $\mathcal{M}, s \models \mathsf{K}(A \vee \neg A)$.)

Notice that the relation in $\mathcal{M}$ has a number of properties that we have discussed. It is reflexive, symmetric, and transitive. Are these reasonable properties for an epistemic indistinguishability relation to have in general? The question is controversial, but here is a quick argument that each is indeed reasonable, at least to a first approximation.

Reflexivity: If we are actually in state s, how could I *know* that we are not in state s? That is, what would allow me (veridically) to rule out s as being the actual state?

Symmetry: Suppose we are in s, but for all I can tell we are in t. Then I am under the impression that, were we in t, I would not be able to tell we are not in s.

Transitivity: If we are in s, and I believe we could be in t instead, and if we were in t I would not know we were not in u, then already in s I cannot tell we are not in u. If I could I would know we were not in t, since I would not be sure about u in t.

Whether or not one finds these quick arguments convincing, we know from our work in the previous section what it means when we have a relation that enjoys these properties. It means the following formulas are guaranteed to be valid (listed here with their traditional names):

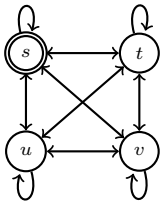
(**T**): $\mathsf{K}A \to A$

(**4**): $\mathsf{K}A \to \mathsf{K}\mathsf{K}A$

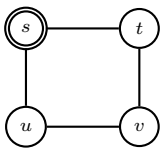
(**5**): $\neg\mathsf{K}A \to \mathsf{K}\neg\mathsf{K}A$

The **T** principle says that anything I know must be true. The **4** principle says, if I know something, then I know that I know. And the **5** principle says that if I do not know something, then I know that I do not know it. All of these principles have been questioned in epistemology (the philosophical analysis of knowledge). That does not prevent this system from being a very useful tool nonetheless.

If we are assuming all of these principles hold, then drawing epistemic models can be even simpler. We never need to draw reflexive loops, transitive arrows can be left out, and arrows do not need to be written with a direction. So, for example, instead of the following frame:

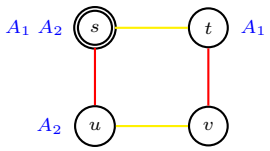


we could simply draw



Epistemic logic becomes most interesting when we represent the knowledge of *multiple agents simultaneously*. After all, there is no reason we could not consider a whole list of modal operators $\mathsf{K}_1, \ldots, \mathsf{K}_n$ for a set $\mathsf{Ag} = \{1, \ldots, n\}$ of agent indices. Then each K_i will correspond to a separate relation R_i, and our models will be tuples, $(S, \{R_i\}_{i \leqslant n}, V)$. Here is a very simple example.

Suppose two people, 1 and 2, are playing a game of cards, and at issue is whether each of them has some red card. It could be they both do, or that neither do, or that just one of them does. Let us suppose that, as a matter of fact, they both do, and both know they do, but neither knows whether the other does. To model this epistemic situation we can introduce two propositional atoms, A_1 and A_2, the former meaning that 1 has a red card, the latter meaning that 2 has a red card. Our model of the situation then might look this this:



where the yellow line corresponds to 1's relation R_1, and the red line corresponds to 2's relation R_2.

In the actual situation, both A_1 and A_2 are true. But neither 1 nor 2 knows both of these are true. Person 1, whose uncertainty in the model is represented by R_1 (the yellow line), is unsure about whether A_2 is true, i.e., about whether person 2 has a red card. This can be seen from the fact that, at the actual state s, state t is accessible from s via an R_1 (yellow) transition, meaning that 1 cannot tell we are *not* in state t, where A_2 is actually false.

On the other hand, 2 cannot tell that we are not in state u. Intuitively this is how it should be, since the only difference between s and u is the truth of A_1. 1 knows we are not in situation u, as can be seen from the fact that there is no R_1 transition to u from s. There is, however, an R_2 (red) transition to u, signifying the fact that 2 is uncertain about whether we are in s or u. The situation is more or less symmetric between 1 and 2: they both know something about themselves (A_1 and A_2, respectively), but do not know about the other.

What about state v? Does either person consider this state possible? In fact, no. Neither 1 nor 2 thinks we could be in v. On the other hand, neither of them knows this about the other! Both think that the other might think that they could be in situation v. Consider this from 1's perspective. For all 1 knows, we could be in state t. Suppose we were. Then A_2 would be false. Because 2 knows whether 2 has a red card, and 1 knows this, 1 would know that 2 was uncertain only about whether 1 has a red card (i.e., about whether A_1 is true). The situation is symmetric for 2.

All these claims about what the two agents know about what the other knows ... etc., can be made precise with our modal language. In particular, the following statements all hold at point s:

$$\mathsf{K}_1 A_1 \qquad \mathsf{K}_2 A_2 \qquad \neg(\mathsf{K}_1 A_2 \vee \mathsf{K}_1 \neg A_2) \qquad \neg(\mathsf{K}_2 A_1 \vee \mathsf{K}_2 \neg A_1)$$

$$\mathsf{K}_1\big(\neg(\mathsf{K}_2 A_1 \vee \mathsf{K}_2 \neg A_1)\big) \qquad \mathsf{K}_2\big(\neg(\mathsf{K}_1 A_2 \vee \mathsf{K}_1 \neg A_2)\big)$$

$$\mathsf{K}_1\big(\mathsf{K}_2 A_2 \vee \mathsf{K}_2 \neg A_2\big) \qquad \mathsf{K}_2\big(\mathsf{K}_1 A_1 \vee \mathsf{K}_1 \neg A_1\big)$$

In words: 1 knows A_1 and 2 knows A_2, while neither knows whether the other has a red card. On the next line, 1 knows that 2 does not know about 1's card, while 2 knows the same thing about 1. And on the last line, 1 nonetheless knows that 2 does know about 2's own card, and 2 knows that 1 knows about 1's card, one way or the other. You should make sure you see why all of these statements indeed hold at s in this model, according to Definition 5.1.3.

We round out this discussion of epistemic logic with a puzzle. Imagine the following situation. Contrary to their father's wishes, three children—call them 1, 2, and 3—have been playing in the mud. Two of them—let us suppose it is children 2 and 3—have mud on their forehead. The father now confronts them. He wants each child to admit their guilt, but he understands that if they can get away with it, they will deny having played in the mud. He says to all three of them together:

Father: At least one of you has mud on your forehead!

Following this he proceeds to ask each child individually if they know whether *they themselves* have mud on their foreheads. Each can see the others and whether they are muddy, but no one can see themselves, so no one knows whether they themselves are muddy. They answer honestly as follows:

Father to Child 1: Do you know whether you have mud on your forehead?

Child 1: No.

Father to Child 2: Do you know whether you have mud on your forehead?

Child 2: No.

Father to Child 3: And do you know whether you have mud on your forehead?

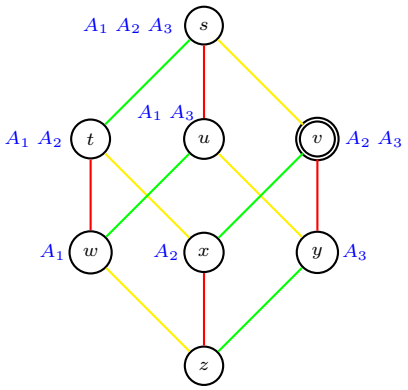
Child 3: Yes, I do know! I am muddy!

And here is the puzzle: *How does Child 3 know this?* Evidently, the father told them something they all knew—that at least one of them was muddy (everyone could see at least one of the others was muddy)—and then the first two children in turn revealed their ignorance. Somehow this led to Child 3 learning her own status.

A sketch of Child 3's reasoning might look as follows (from her own perspective): Suppose I were *not* muddy. I can see that Child 2 is muddy and Child 1 is not. So if I were Child 2, I would have known that I was muddy. For Child 2 also knew at least one of us was muddy. Having seen children 1 and 3 are not muddy, the only possibility would have been that Child 2 was. But Child 2 did not know, so I must be muddy after all.

This is somewhat complicated reasoning, and as you might imagine, things become even more complex as the number of children increases. The order of the announcements from the children also matters for the resolution of the puzzle.

However, we can gain a systematic understanding of the situation using the modal logical framework we have developed up to now. Specifically, we can represent this initial situation with a very large modal model, representing the initial state of ignorance for the three children. Again, each child knows whether the other two children are muddy, but none know their own status. We capture this as follows (where A_1, A_2, and A_3 stand for the statement that child 1, 2, and 3 is muddy, respectively; and the yellow line is R_1, i.e., child 1's indistinguishability relation, the red line is R_2, and the green line is R_3):



Note that in the actual state, v, where children 2 and 3 are muddy, the following all hold:

$$\neg(\mathsf{K}_1 A_1 \vee \mathsf{K}_1 \neg A_1) \qquad \neg(\mathsf{K}_2 A_2 \vee \mathsf{K}_2 \neg A_2) \qquad \neg(\mathsf{K}_3 A_3 \vee \mathsf{K}_3 \neg A_3)$$

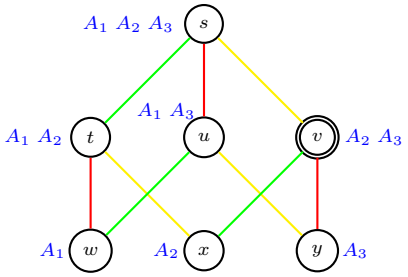
$$\mathsf{K}_2\big(\neg(\mathsf{K}_1 A_1 \vee \mathsf{K}_1 \neg A_1)\big) \qquad \mathsf{K}_1\big(\neg(\mathsf{K}_2 A_2 \vee \mathsf{K}_2 \neg A_2)\big) \qquad \mathsf{K}_1\big(\neg(\mathsf{K}_3 A_3 \vee \mathsf{K}_3 \neg A_3)\big)$$

$$\mathsf{K}_1(\mathsf{K}_2 A_1 \vee \mathsf{K}_2 \neg A_1) \qquad \mathsf{K}_1(\mathsf{K}_3 A_1 \vee \mathsf{K}_3 \neg A_1) \qquad \mathsf{K}_2(\mathsf{K}_1 A_2 \vee \mathsf{K}_1 \neg A_2)$$

That is, no one knows whether they are muddy, and they all know this about each other, but they do know that the others know whether they themselves are muddy.

Note that, while no one considers, e.g., state w possible, we must represent it explicitly in the model. Everyone knows that the world is not such that only 1 is muddy. It is even the case that everyone knows that everyone knows this. But it is *not* the case that everyone knows that everyone knows that everyone knows that this is not true. For instance, 3 considers it possible that we are in x. If we are in x, then 2 would consider it possible that we are in z. If we are in z, 1 would consider it possible that we are in w. That is, 3 considers it possible that 2 considers it possible that 1 considers it possible that we are in w!

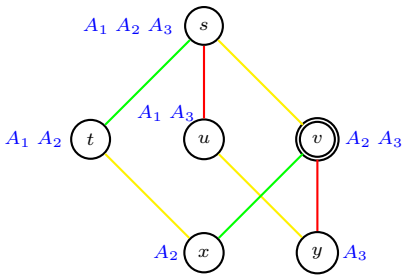
In addition to modeling the initial situation, we can also try to model what happens at various points in the story. For instance, consider the father's first statement, that at least one of them is muddy. This guarantees that z is impossible, and known by everyone to be so:



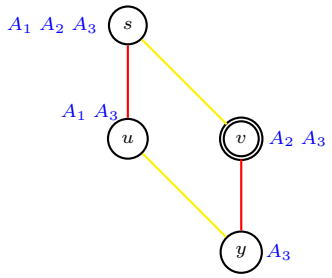
Notice already that the children have acquired at least some knowledge. For instance, 3 knows that if she is not muddy, then 2 knows whether she is not muddy. That is, the following is true at v:

$$K_3\big(\neg A_3 \to (\mathsf{K}_2 A_2 \vee \mathsf{K}_2 \neg A_2)\big)$$

What happens after Child 1's announcement that she does not know whether she is muddy? Intuitively, this eliminates any state where Child 1 *does* know her status. Are there any? In fact, at w Child 1 would know she was muddy, so we eliminate it, since no one could consider it possible that anyone would consider it possible that $(\dots)$ we are in w:



Does anyone know yet of themselves whether they are muddy? Evidently, each still considers some other state possible. But next a crucial step occurs. Child 2 announces that she does not know her status, so we can eliminate any state where she would know (x and t):



In the resulting model, child 3 all of a sudden knows she is muddy! She does not consider any other state possible from v. Thus we can read right off the model the solution to the puzzle.

Exercise 5.2.1. At this point in the story, after 3's announcement that she knows, does either 1 or 2 know their status (whether they are muddy or not)? Why or why not?

Exercise 5.2.2. Suppose the father continues the line of questioning, and asks 1 again whether 1 knows. Suppose then 1 answers truly (whatever this is, according to your answer to the previous question). At that point does 2 know her status?

Exercise 5.2.3. Suppose the father continues asking the remaining children if they know whether they are muddy. Would that ever lead to 1 or 2 coming to know their status? Why or why not?

Exercise 5.2.4. Suppose the protocol had been different. The father first asked 2 whether 2 knew, then 3, and then 1. Would anyone know their status at this point, after each child answered truly? Continuing in this way with more rounds of questioning, would anyone ever figure out their status? If so, what is the first point where this would happen?

Exercise 5.2.5. Imagine that the facts of the situation were different, and that all three children were muddy. How would your answers to the previous questions be different (if at all)?

5.3 Temporal Logic

On the epistemic interpretation of modal logic we consider the points as "epistemically possible states of affairs" and the relation as "epistemic indistinguishability." In this section we consider a rather

different interpretation. The points will be thought of as *states of the world at a particular time* and the relation will be understood as a *precedence relation*: we will have Rtu if time point t precedes time point u. Corresponding to this interpretation, our modal operator $\Diamond$ can be interpreted as, "at some time point in the future," and $\Box$ as, "at every time point in the future."

However, there is something special about time. We of course want to be able to speak about what will be true in the future. But we also want to be able to speak about what *was* true in the past. That is, at a time point t, we might want to speak about some earlier point u such that Rut.

In this direction, instead of the usual two modal operators $\Diamond$ and $\Box$, let us introduce *four* modal operators, F, G, P, and H, with intuitive interpretations as follows:

F: "At some point in the future it will be the case that ..."

G: "It is always going to be the case that ..."

P: "At some point in the past it was the case that ..."

H: "It has always been the case that ..."

You might think of the F and P operators as the $\Diamond$ operator looking forward and backward, respectively, while the G and H are the forward-looking and backward-looking $\Box$ operators.

Our formal language is now only slightly more complicated than the basic modal language:

Definition 5.3.1. The language $\mathcal{L}^{\mathsf{F},\mathsf{G},\mathsf{P},\mathsf{H}}$ of temporal logic is given by the following set of wffs:

1. Every propositional atom $A \in \mathsf{Prop}$ is a wff in $\mathcal{L}^{\mathsf{F},\mathsf{G},\mathsf{P},\mathsf{H}}$.
2. If φ and ψ are wffs in $\mathcal{L}^{\mathsf{F},\mathsf{G},\mathsf{P},\mathsf{H}}$, then so are:
   (a) $(\varphi \wedge \psi)$
   (b) $\neg\varphi$
   (c) $(\varphi \vee \psi)$
   (d) $(\varphi \to \psi)$
   (e) $\mathsf{F}\varphi$
   (f) $\mathsf{G}\varphi$
   (g) $\mathsf{P}\varphi$
   (h) $\mathsf{H}\varphi$
3. Nothing else is a wff in $\mathcal{L}^{\mathsf{F},\mathsf{G},\mathsf{P},\mathsf{H}}$.

Notice this language allows approximately expressing various tenses we invoke in English and other natural languages:

$\mathsf{P}\varphi \approx$ "It was the case that φ."

$\mathsf{F}\varphi \approx$ "It will be the case that φ."

$\mathsf{PP}\varphi \approx$ "It had been the case that φ."

$\mathsf{FP}\varphi \approx$ "It will have been the case that φ."

$\mathsf{PF}\varphi \approx$ "It was going to be the case that φ."

With these intended interpretations in mind, we will write our usual modal modals (S, R, V) with the more suggestive notation $(T, <, V)$ instead, so that T reminds us we are speaking of time points, and $<$ is suggestive of the precedence relation. We can then give our truth clauses for *truth of a formula at a time point* t as with general modal logic:

Definition 5.3.2. The **truth** of $\varphi \in \mathcal{L}^{\mathsf{F},\mathsf{G},\mathsf{P},\mathsf{H}}$ in a model $\mathcal{T} = (T, <, V)$, at a point $t \in T$—written $\mathcal{T}, t \models \varphi$—is defined as follows:

1. $\mathcal{T}, t \models A$, if and only if $t \in V(A)$ $\big($whenever $A \in \mathsf{Prop}\big)$.
2. $\mathcal{T}, t \models \varphi \wedge \psi$, if and only if both $\mathcal{T}, t \models \varphi$ and $\mathcal{T}, t \models \psi$.
3. $\mathcal{T}, t \models \neg\varphi$, if and only if it does not hold that $\mathcal{T}, t \models \varphi$.
4. $\mathcal{T}, t \models \varphi \vee \psi$, if and only if either $\mathcal{T}, t \models \varphi$ or $\mathcal{T}, t \models \psi$.
5. $\mathcal{T}, t \models \varphi \to \psi$, if and only if either $\mathcal{T}, t \models \varphi$ does *not* hold, or $\mathcal{T}, t \models \psi$ does hold.
6. $\mathcal{T}, t \models \mathsf{F}\varphi$, if and only if $\mathcal{T}, u \models \varphi$ holds for *some* $u \in T$ such that $t < u$.
7. $\mathcal{T}, t \models \mathsf{G}\varphi$, if and only if $\mathcal{T}, u \models \varphi$ holds for *every* $u \in T$ such that $t < u$.
8. $\mathcal{T}, t \models \mathsf{P}\varphi$, if and only if $\mathcal{T}, u \models \varphi$ holds for *some* $u \in T$ such that $u < t$.
9. $\mathcal{T}, t \models \mathsf{H}\varphi$, if and only if $\mathcal{T}, u \models \varphi$ holds for *every* $u \in T$ such that $u < t$.

We note again in this context (without proof), the following are *valid* in any temporal model, and therefore also on any temporal frame:

Fact 5.3.1. The following formulas are all valid in every temporal model:

$\mathsf{F}\varphi \leftrightarrow \neg\mathsf{G}\neg\varphi$ $\qquad$ $\mathsf{H}\varphi \leftrightarrow \neg\mathsf{P}\neg\varphi$

$\mathsf{G}\varphi \leftrightarrow \neg\mathsf{F}\neg\varphi$ $\qquad$ $\mathsf{G}(\varphi \to \psi) \to (\mathsf{G}\varphi \to \mathsf{G}\psi)$

$\mathsf{P}\varphi \leftrightarrow \neg\mathsf{H}\neg\varphi$ $\qquad$ $\mathsf{H}(\varphi \to \psi) \to (\mathsf{H}\varphi \to \mathsf{H}\psi)$

However, just as in the case of epistemic logic, one might think there are more interesting frame validities characterizing temporal domains. For instance, here is a principle that is distinctive of the temporal interpretation, and comes out valid just by the fact that we interpret the future modalities and the past modalities using the *same relation*:

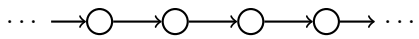
Fact 5.3.2. The following formulas are valid on every temporal frame:

$\varphi \to \mathsf{HF}\varphi$ $\qquad$ $\varphi \to \mathsf{GP}\varphi$

In English: if φ is true, then it has always been the case that it will at some point be true; and it will always be the case that it was at some point true.

Further potential validities will correspond to additional assumptions about the precedence relation $<$, just as before. For instance, if we want $<$ to be transitive, this will mean that both $\mathsf{G}\varphi \to \mathsf{GG}\varphi$ and $\mathsf{H}\varphi \to \mathsf{HH}\varphi$ will be valid.

One important idea is that of a *time line*: the picture of time as a linear relation among points going infinitely into the past and into the future; something like the following:



Just as in Ex. 5.1.7, we have formulas that say time has no beginning and no end:

Fact 5.3.3. The following both hold:

1. $(T, <) \models \mathsf{G}A \to \mathsf{F}A$, iff for each $t \in T$ there is a $u \in T$ such that $t < u$.
2. $(T, <) \models \mathsf{H}A \to \mathsf{P}A$, iff for each $t \in T$ there is a $u \in T$ such that $u < t$.

Interestingly, there is also a formula expressing the *linearity* of $<$, i.e., the absence of *branches*.

Definition 5.3.3. We say a relation $<$ is *linear* if two conditions hold:

1. Whenever $t < u$ and $t < v$, then either (i) $u = v$, (ii) $u < v$, or (iii) $v < u$.
2. Whenever $u < t$ and $v < t$, then either (i) $u = v$, (ii) $u < v$, or (iii) $v < u$.

Here is a more difficult problem:

Exercise 5.3.1. Show that $<$ is linear, if and only if $(T, <) \models (\mathsf{FP}A \vee \mathsf{PF}A) \to (A \vee \mathsf{P}A \vee \mathsf{F}A)$.

[Note that $(T, <) \models (\mathsf{FP}A \vee \mathsf{PF}A) \to (A \vee \mathsf{P}A \vee \mathsf{F}A)$ holds, iff both $(T, <) \models \mathsf{FP}A \to (A \vee \mathsf{P}A \vee \mathsf{F}A)$ and $(T, <) \models \mathsf{PF}A \to (A \vee \mathsf{P}A \vee \mathsf{F}A)$ hold. Given the definition of linearity, it may be easier to break this problem down into two parts.]

Transitivity, forward and backward seriality, and linearity would essentially give us temporal models that look like the one above. But there are still choices about what this time line should look like. For instance, if we think of the time line in terms of a *number line*, should it have the structure of the natural numbers, with discrete time points and nothing in between them, or the structure of a *dense* set like the rational numbers or real numbers, where between each two points there is another point? We already saw a modal principle in Ex. 5.1.6 that corresponds to density: we had that $(T, <) \models \mathsf{GG}A \to \mathsf{G}A$, iff $(T, <)$ is dense. The following question should be easy, but it requires you understand how everything fits together up to this point:

Exercise 5.3.2. Explain why we have $(T, <) \models \mathsf{HH}A \to \mathsf{H}A$, iff $(T, <)$ is dense.

[Do not make the mistake of assuming $(\mathsf{GG}A \to \mathsf{G}A) \leftrightarrow (\mathsf{HH}A \to \mathsf{H}A)$ is valid over temporal frames!]

If we want to assume our time line has the discrete structure of the natural numbers, this will mean certain other formulas will become valid. Here is one of the most interesting, foreshadowing the *induction axiom* for natural numbers (cf. §6.7):

Fact 5.3.4. $(\mathbb{N}, <) \models \big(\mathsf{F}A \wedge \mathsf{G}(A \to \mathsf{F}A)\big) \to \mathsf{GF}A$.

Proof Sketch. Here we are supposing that $T = \mathbb{N}$ and $<$ is the usual order of "less than" on the natural numbers. Suppose for some number n and valuation V, we have $(\mathbb{N}, <, V), n \models \big(\mathsf{F}A \wedge \mathsf{G}(A \to \mathsf{F}A)\big)$, i.e., both $(\mathbb{N}, <, V), n \models \mathsf{F}A$ and $(\mathbb{N}, <, V), n \models \mathsf{G}(A \to \mathsf{F}A)$. We must show that $(\mathbb{N}, <, V), m \models \mathsf{F}A$ for every m with $n < m$, i.e., that there is some p with $m < p$ such that $(\mathbb{N}, <, V), p \models A$.

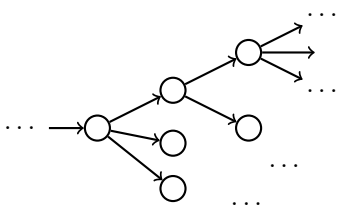
Pick any m with $n < m$. Because $(\mathbb{N}, <, V), n \models \mathsf{F}A$, we know there is a point q, with $n < q$, such that $(\mathbb{N}, <, V), q \models A$. If $m < q$, then we are done. Otherwise, we use the earlier fact that $(\mathbb{N}, <, V), n \models \mathsf{G}(A \to \mathsf{F}A)$, and hence $(\mathbb{N}, <, V), q \models \mathsf{F}A$, which gives us some other point $q' > q$ such that $(\mathbb{N}, <, V), q' \models A$. Again, if $m < q'$, then we are done. Otherwise, we repeat this until we finally get to a point satisfying A that is greater than m. Because there are only finitely many numbers between n and m, we know we will eventually reach such a point. □

Optional aside on continuity: As we know, the rational numbers $(\mathbb{Q}, <)$ and the real numbers $(\mathbb{R}, <)$ are not isomorphic. They are both dense linear orders without endpoints, but $\mathbb{R}$ has properties that $\mathbb{Q}$ lacks, like *Dedekind completeness*: every non-empty set of points $X \subseteq \mathbb{R}$ that has an upper bound (a point r such that $t \leqslant r$ for all $t \in X$) has a *least* upper bound (an upper bound r such that $r < r'$ for any other upper bound r' of X). Perhaps surprisingly, there is a temporal logical axiom that expresses Dedekind completeness. Here is a somewhat difficult exercise:

Exercise 5.3.3. Suppose $(T, <)$ is a linear order without endpoints. Show $(T, <)$ is Dedekind complete, iff $(T, <) \models \big((\mathsf{H}A \to \mathsf{FH}A) \wedge \mathsf{H}(\mathsf{H}A \to \mathsf{FH}A) \wedge \mathsf{G}(\mathsf{H}A \to \mathsf{FH}A)\big) \to (\mathsf{H}A \to \mathsf{G}A)$.

5.3.1 Branching Time Temporal Logic

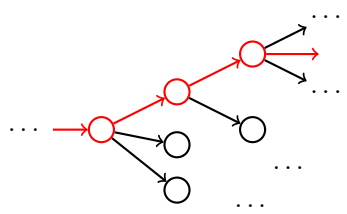
Many, going back to Aristotle, have questioned whether statements about the future could be simply true or false at all. While the picture above is natural from some perspectives, one might worry that the version of temporal logic developed so far assumes some kind of *determinism*, and perhaps precludes any possibility of *free choice* or other sorts of indeterminacy. What if we wanted to allow a *branching future*, with many possible continuations of any given point? That is, something that looks more like the following?



Supposing we can branch only into the future, and not into the past, we can suppose our temporal frame $(T, <)$ has the structure of a *tree*:

Definition 5.3.4. $(T, <)$ is a *tree* if for each $t \in T$, the set of predecessors of t, $\{t' : t' < t\}$, is linear.

Given a tree $(T, <)$ we can then define a *history* h as a maximal set of points in T linearly ordered by $<$. Intuitively a history h defines a *path* through the tree. For instance, one history on the example graph above might look like the red line below (continuing on in both directions):

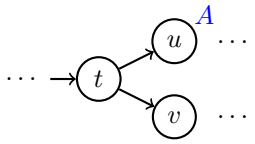


Roughly following a proposal by the medieval thinker William of Ockham, we can interpret modal temporal formulas in this setting, relative to not just a model $(T, <, V)$ and a point t, but also a history h that includes t. Let us introduce one more modal operator—call it $\Box$—which will be interpreted as "historical necessity." Let $\mathcal{H}(t) = \{h : t \in h\}$ be the set of histories containing point t, i.e., all the different ways of continuing from t into the future. Then $\Box\varphi$ will say of a point t that φ is true on all histories that include t:

Definition 5.3.5. The **truth** of $\varphi \in \mathcal{L}^{\mathsf{G},\mathsf{H},\Box}$ in a model $\mathcal{T} = (T, <, V)$, at a history h, at point $t \in h$, is defined, so that:[2]

1. $\mathcal{T}, h, t \models A$, if and only if $t \in V(A)$ $\big($whenever $A \in \mathsf{Prop}\big)$.
2. $\mathcal{T}, h, t \models \varphi \wedge \psi$, if and only if both $\mathcal{T}, h, t \models \varphi$ and $\mathcal{T}, h, t \models \psi$.
3. $\mathcal{T}, h, t \models \neg\varphi$, if and only if it does not hold that $\mathcal{T}, h, t \models \varphi$.
4. $\mathcal{T}, h, t \models \varphi \vee \psi$, if and only if either $\mathcal{T}, h, t \models \varphi$ or $\mathcal{T}, h, t \models \psi$.
5. $\mathcal{T}, h, t \models \varphi \rightarrow \psi$, if and only if either $\mathcal{T}, h, t \models \varphi$ does *not* hold, or $\mathcal{T}, h, t \models \psi$ does hold.
6. $\mathcal{T}, h, t \models \mathsf{G}\varphi$, if and only if $\mathcal{T}, h, u \models \varphi$ holds for *every* $u \in h$ such that $t < u$.
7. $\mathcal{T}, h, t \models \mathsf{H}\varphi$, if and only if $\mathcal{T}, h, u \models \varphi$ holds for *every* $u \in h$ such that $u < t$.
8. $\mathcal{T}, h, t \models \Box\varphi$, if and only if $\mathcal{T}, h', t \models \varphi$ holds for *every* $h' \in \mathcal{H}(t)$.

This allows us to describe and reason about situations where multiple futures are possible. For instance, in the simplest kind of case, we can have a point where at the next instance either A or $\neg A$ could turn out to be true. It might depend, e.g., on what action a given agent takes:



Let h_1 be a history that passes through u, and h_2 one that passes through v (thus, both pass through t), and suppose that in this model all points later than u make A true, while none of the points later than v do. Then note in particular that $\mathcal{T}, h_1, u \models A$, while $\mathcal{T}, h_2, v \models \neg A$. Already at t, we have $\mathcal{T}, h_1, t \models \mathsf{F}A$ and in fact $\mathcal{T}, h_1, t \models \mathsf{G}A$, while $\mathcal{T}, h_2, t \models \mathsf{G}\neg A$. That is, depending on which

[2]We leave out F and P, because we know we can express them with G and H, just as $\Diamond$ can be expressed with $\Box$.

possible future we are focusing on, we could have A true or false forever after. We can express this right in the language, by noticing that at t (on either history $h = h_1$ or $h = h_2$) we have

$$\mathcal{T}, h, t \models \Diamond \mathsf{G} A \wedge \Diamond \mathsf{G} \neg A.$$

Note that this setting is strictly more general than the setting we had before. In particular, we can restrict attention to the non-branching frames with a single formula.

Exercise 5.3.4. Find a formula $\varphi \in \mathcal{L}^{\mathsf{G},\mathsf{H},\Box}$ for which $(T, <) \models \varphi$—where this now means φ is true under all valuations, at all points, and at all histories—iff $(T, <)$ is non-branching.
[Note: the formula from Ex. 5.3.1 will not work. That formula is valid on *all* frames in this setting.]

5.3.2 *Since*, *Until*, and Program Specifications

[This section is optional.]

Two very powerful temporal operators, which can express everything we have done so far and more, were introduced in the 1960s by the linguist and philosopher Hans Kamp. They correspond roughly to English 'since' and 'until'. Their definitions make sense whether we are working with non-branching models or branching models, though here we focus on the non-branching case.

We now let our modal language have only two *binary* modal operators, S and U, so that if φ and ψ are wffs, then so are

$$\begin{aligned} \varphi \mathsf{S} \psi &\approx \text{“}\varphi \text{ has been true ever } \textit{since} \text{ some point when } \psi \text{ was true.”} \\ \varphi \mathsf{U} \psi &\approx \text{“}\varphi \text{ will be true up } \textit{until} \text{ some point when } \psi \text{ will be true.”} \end{aligned}$$

These intuitive readings should only be thought of as approximate. More formally,

$$\begin{aligned} \mathcal{T}, t \models \varphi \mathsf{S} \psi \quad &\text{iff} \quad \mathcal{T}, u \models \psi \text{ holds for some } u \in S \text{ such that } u < t, \\ &\qquad \text{and for every point } v, \text{ with } u < v < t, \text{ we have } \mathcal{T}, v \models \varphi. \\ \mathcal{T}, t \models \varphi \mathsf{U} \psi \quad &\text{iff} \quad \mathcal{T}, u \models \psi \text{ holds for some } u \in S \text{ such that } t < u, \\ &\qquad \text{and for every point } v, \text{ with } t < v < u, \text{ we have } \mathcal{T}, v \models \varphi. \end{aligned}$$

Note right away that we can define all the operators we had before with these two. For instance, $F\varphi$ is equivalent to $\top \mathsf{U} \varphi$, while $P\varphi$ is equivalent to $\top \mathsf{S} \varphi$. But we can also express a lot more that we could not (at least obviously) express before. For instance, we could express something roughly like the English sentence,

(∗) Ever since Stanford University was founded, students have been dying the fountains red up until the Big Game.

If we have an atom F meaning, “Stanford is founded,” R meaning, “Fountains are died red,” and B meaning, “The big game is happening,” then (∗) can be rendered as something like,

$$(R \mathsf{U} B) \mathsf{S} F.$$

Temporal logics with S and U have been influential in linguistics and philosophy, but they have actually had the most widespread influence in computer science. One important way temporal logic is used in computer science is to specify some desirable property that a program or some other system is supposed to satisfy. Many of the variants of temporal logic used in this setting only refer to *future* states. A few simple examples give the flavor of this type of application:

$$\mathsf{G}(R \to \mathsf{F}A) \qquad\qquad \mathsf{G}\big(R \to (\neg M \,\mathsf{U}\, A)\big)$$

The first says, "It will always be the case that, if a request is received, it will eventually be acknowledged." The second says, "It will always be the case that, if a request is received, it will not be marked as acknowledged until it is in fact acknowledged."

In some applications people must deal with transition systems possessing around 10^4 or 10^5 states. The main challenge in this setting is to find fast algorithms for determining whether such a transition system—which in our notation is just a temporal model $\mathcal{T}$ with an initial time t_0—makes some formula true, i.e, whether some specification holds. In other words, the goal is to find a fast way of determining whether $\mathcal{T}, t_0 \models \varphi$. Breakthrough work on this topic has won some researchers Turing Awards (the "Nobel Prize of Computer Science").

Chapter 6

First Order Logic

For many, First Order Logic holds a certain amount of prestige, distinguished among logical systems. Artificial Intelligence pioneer John McCarthy took First Order Logic to be the uniquely best system in which to formalize human common sense reasoning. Before that, the seminal philosopher W.V.O. Quine declared that we ought to take 'logic' to be synonymous with 'First Order Logic'!

Building on top of Boolean Logic, First Order Logic allows us to speak in detail about individuals, functions, predicates, and relations; and most crucially, it allows us to *quantify* over individuals. Intuitively quantification is what we are doing when we use ordinary language *quantifiers* like *all*, *some*, *no*, *at least seven*, and so on. Because of all this rich structure, First Order Logic has been the logic of choice also for formalization of mathematics, and for much of computer science, both for theory and in more practical domains such as reasoning about databases. Its role in linguistics (specifically formal semantics) and in central areas of contemporary philosophy—metaphysics, epistemology, philosophy of language, philosophy of science—has also been central.

Because of the additional expressive power that First Order Logic possesses, the system is also quite a bit more complex than Boolean or Modal Logic. We will therefore proceed slowly, building on what we have already covered. By the end we will be able to formalize First Order Arithmetic—one particular application of First Order Logic—which already captures a significant amount of mathematics, including essentially all of standard number theory. This will also put us in a position to understand the statement of Kurt Gödel's celebrated Incompleteness Theorem.

6.1 Reasoning about Relations

In our study of modal logic, modal models (S, R, V) were based on *frames* (S, R), which are essentially just relational structures, or graphs. In that setting we looked at several "interpretations" of the graph: the epistemic interpretation, where the nodes are possible states of the word, and the relation captures "epistemic indistinguishability"; and the temporal interpretation, on which the nodes are points in time and the relation is temporal precedence. But graphs of course appear all over the place, for many different purposes.

The modal language allowed us to say certain things about graphs. But we could just as well

imagine other kinds of logical languages for talking about such structures. For instance, suppose we wanted to talk about which American states border which other states (and let us restrict attention to the "lower 48," ignoring Hawaii and Alaska):



Let us also suppose—this is going to be one of the novel features of the first order setting—that we actually have a *relation symbol* B in our formal language, corresponding to the "borders" relation in the US, as well as some *constant symbols* for some of the states: c, n, o, t, a, and l, corresponding to California, Nevada, Oregon, Texas, Arkansas, and Louisiana. We could use this little language to make statements about the "borders" relation among these states that we can refer to with constants. So, intuitively, a formula like $B(c, n)$ ought to be true here, since California does indeed border Nevada. On the other hand $B(c, t)$ is false, since California does not border Texas. We might therefore say that $\neg B(c, t)$ should be true. Likewise, we could consider any Boolean combination of these formulas. For example, the formula below is evidently true in this case:

$$B(n, o) \wedge B(c, o) \wedge \neg B(o, t) \wedge \neg B(c, l)$$

as is the following formula:

$$B(o, a) \rightarrow B(l, a)$$

and so on. All we have done at this point is to introduce some notation into our formal language for explicitly talking about whether a given relation holds between some pairs of individuals that we can name. We can embed this into the full Boolean language that we had before, so that instead of "structureless" propositional atoms A_1, A_2, ..., our "atomic formulas" can now have some structure. This is essentially still Boolean Logic, since each of these "atomic formulas" $B(c, a)$, $B(o, t)$, and so on, can either be true or false, and the Boolean connectives again combine them together to produce some complex expression that will itself be either true or false.

We can go beyond Boolean logic by introducing basic examples of *quantificational statements*. For instance, though we do not have a constant in this example language for Washington state, we might want to observe that it is true that *there exists* a state that borders Oregon, but not Nevada. We can say this with the help of the *existential quantifier* $\exists$, so that the formula in question would look like this:

$$\exists x\big(B(x,o) \wedge \neg B(x,n)\big)$$

The way to read this formula is:

"There exists some x, such that x borders Oregon and x does not border Nevada."

or perhaps more colloquially:

"Some x borders Oregon but not Nevada."

The quantifier $\exists$ takes a *variable* x, and asserts there is something—call it x—such that the formula inside the parentheses is true of this x. Another example of a true existential quantification is:

$$\exists x\big(B(x,t) \wedge B(x,a) \wedge \neg B(x,l)\big)$$

This says that there is some x that borders Texas and Arkansas, but not Louisiana. This is true, because there is indeed such a state, namely Oklahoma. Other existential statements might be false. For example, there is no state that borders both Texas and California, so

$$\exists x\big(B(x,t) \wedge B(x,c)\big)$$

is false, i.e., $\neg\exists x\big(B(x,t) \wedge B(x,c)\big)$ is true.

The other important quantifier is the *universal quantifier* $\forall$, which allows us to assert that *all* things in the domain satisfy a given formula. Here is an example:

$$\forall x\big(B(x,c) \rightarrow \neg B(x,t)\big)$$

This formula says, again truly, that anything which borders California does *not* border Texas. Or slightly more precisely:

"For any x you pick, if x borders California then it is not the case that x borders Texas."

The universal quantifier also allows us to assert very general facts about this relation that do not even mention any particular state, such as that the relation is *symmetric*. This requires using two universal quantifiers in a row.

$$\forall x\forall y\big(B(x,y) \rightarrow B(y,x)\big)$$

Again, the way to read this is: for any state x you pick, and any state y you pick, if x borders y, then y also borders x. But, for example, "borders" is not transitive. We can express transitivity also very easily. This would say:

$$\forall x\forall y\forall z\Big(\big(B(x,y) \wedge B(y,z)\big) \rightarrow B(x,z)\Big)$$

That is: for all states x, y, and z, if x borders y and y borders z, then x borders z. This is of course not true, so we instead have:

$$\neg\forall x\forall y\forall z\Big(\big(B(x,y)\wedge B(y,z)\big)\rightarrow B(x,z)\Big)$$

Because our "domain of discourse" in this example includes only the lower 48, the following formula, involving both quantifiers, is true:

$$\forall x\exists y B(x,y)$$

This says: for any state x there exists some state y such that x borders y. Note that this formula is importantly different from the next formula, which says, there exists a particular state y such that, for any state x, x borders y. This formula is false in this case:

$$\exists y\forall x B(x,y)$$

After all, there is no state that every other state borders.

We might also allow ourselves an *equality symbol* in the language, $=$, which we can use to say that two things in the domain are equal. A trivial example is:

$$\exists x\, x=t$$

which simply says that there exists a state that is equal to Texas. But we can also express more interesting claims, such as the following:

$$\forall x\exists y\exists z\big(B(x,y)\wedge B(x,z)\wedge\neg y=z\big)$$

This says, that for every state x, there are at least two *distinct* (non-equal) states y and z such that x borders both of them. Is it true? Almost. The one counterexample is Maine, which borders only one state. This formula is therefore false. But there are other interesting formulas using the equality that are true. For example, here is a very long formula (where we abbreviate $\neg x=y$ as $x\neq y$):

$$\exists x\exists y\exists z\exists w\exists u\big(B(x,y)\wedge B(x,z)\wedge B(x,w)\wedge B(x,u)\wedge y\neq z\wedge y\neq w\wedge y\neq u\wedge z\neq w\wedge z\neq u\wedge w\neq u\big)$$

This says that there is some state x such that it borders (at least) four distinct states. The last six conjuncts ensure that no two of the "witnesses" of the existential statements are equal. We can also say that there is a state that borders at most one other state, which is also true (recall Maine):

$$\exists x\forall y\forall z\Big(\big(B(x,y)\wedge B(x,z)\rightarrow y=z\Big)$$

The way to read this is: there is some state x, such that if you find any two states that x borders, then those states must be the same! (In the particular case at hand, if Maine borders any state y, it must be equal to New Hampshire.)

This particular example of US states is just an illustration. We can come up with languages like this to reason about most any kind of structure. To take an example that will be central in this chapter, suppose we want to talk about the standard natural numbers $\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$. And suppose we have one binary relation in our language $<$. We can already say a lot about these numbers. For instance, we can make the obvious statement that no number is less than itself:

$$\forall x(\neg x < x)$$

We can also state that "less than" is transitive:

$$\forall x \forall y \forall z \big((x < y \wedge y < z) \rightarrow x < z\big)$$

And given that these two hold we can also state that there are infinitely many of them:

$$\forall x \exists y (x < y)$$

We can also define various other notions in terms of $<$ using the expressivity of first order logic, e.g., "less than or equal to" can easily be defined:

$$x \leqslant y \qquad \triangleq \qquad (x < y \vee x = y)$$

We can also say things like, "y is at least three greater than x":

$$\exists u \exists v (x < u \wedge u < v \wedge v < y)$$

And so on. Arithmetic will become most interesting when we can also talk about *arithmetical functions* like plus and times. We will introduce function symbols in the next section.

For one final concrete example, we do not need to restrict ourselves to talking about *binary* relations. Suppose we want to talk about government officials of different US states and whether they are republican or democrat. We might have a *ternary relation symbol* $P(x, y, z)$, meaning that person x holds position y in state z, and one-place relations $R(x)$ and $D(x)$, meaning respectively that x is a republican and democrat. Suppose we also have the same state constant symbols as in the first example—c, n, o, t, a, and l—as well as constants for some positions—g is the position of governor, and s is the position of senator. Then we can say things like the following:

$$\forall x \big(P(x, s, t) \rightarrow R(x)\big)$$

that anyone who is a Texas senator is a republican (which happens to be true at the time of writing). We could also make a stronger (probably false) statement, that anyone who holds any position in Texas is a republican:

$$\forall x \forall y \big(P(x, y, t) \rightarrow R(x)\big)$$

Note that this is equivalent to the following, a point to which we will return:

$$\forall x\big(\exists y\big(P(x,y,t)\big) \to R(x)\big)$$

We can also say, e.g, that if a state has a democratic governor, then it has at least one democratic senator (one would have to check whether this is true):

$$\forall x\forall y\Big(\big(P(y,g,x) \wedge D(y)\big) \to \exists z\big(P(z,s,x) \wedge D(z)\big)\Big)$$

We can also say (truly in this, because of California at the moment) that there is a state such that its governor and senators are all democrats:

$$\exists x\forall y\forall z\Big(\big(P(y,z,x) \wedge (z = g \vee z = s)\big) \to D(y)\Big)$$

Exercise 6.1.1. Suppose we have in a logical language a binary relation symbol R. We have already seen that we can "say" in first-order logic that R is transitive:

$$\forall x\forall y\forall z\Big(\big(R(x,y) \wedge R(y,z)\big) \to R(x,z)\Big)$$

We did the same thing for symmetry:

$$\forall x\forall y\big(R(x,y) \to R(y,x)\big)$$

Write down a formula that says R is *euclidean*.

Exercise 6.1.2. Write down a formula that says R is *dense*. (Recall Ex. 5.1.6 for the definition.)

Exercise 6.1.3. Write down a formula that says R is *linear*. (Recall Def. 5.3.3 for the definition.)

6.2 First Order Languages

In the previous section we introduced *individual variables* $x, y, z, \ldots$, *relation symbols*, $R, P, \ldots$, *quantifiers* $\forall$, $\exists$, the *equality* symbol $=$, and *individual constants* $c, d, \ldots$, on top of our usual Boolean connectives, with their intuitive interpretations. In this section we make the language more precise, and in the next section we make their interpretations more precise. But first, it is useful to introduce one further component of typical first order languages: *function symbols*.

We can add function symbols $f, g, \ldots$ of any arities. For this we will generally use lower case roman letters. For example, suppose we wanted to have a logical language for talking about genealogical information, say, including a two-place relation R for *being related*, and two one-place functions m, for *mother of*, and f for *father of*. Then we could use a first order formula to say that if two people have the same mother or father, then they are related:

$$\forall x\forall y\Big(\big(f(x) = f(y) \vee m(x) = m(y)\big) \to R(x,y)\Big)$$

We could also write the statement that everyone has a father and a mother:

$$\forall x \exists y \exists z \big(f(x) = y \wedge m(x) = z \big)$$

We can also, e.g., define what it is for x to be a grandfather of y.

$$G(x,y) \quad \triangleq \quad \exists z \big(f(z) = x \wedge (f(y) = z \vee m(y) = z) \big)$$

We could define the same notion more simply by iteratively applying a function:

$$G(x,y) \quad \triangleq \quad \big(f(m(y)) = x \vee f(f(y)) = x \big)$$

That is, $f(f(y))$ is "the father of the father of y," while $f(m(y))$ is "the father of the mother of y."

If we move back to arithmetic, we find lots of useful functions. For the moment let us restrict attention to the two binary function symbols $+$ and $*$, meaning *plus* and *times*, respectively. Let us also suppose we have a constant symbols 0 standing for zero, and 1 standing for one. Then in our earlier examples using the relation symbol $<$, we could have actually done without this, defining it in terms of $+$:

$$x < y \quad \triangleq \quad \exists z (z \neq 0 \wedge x + z = y)$$

We can also say lots of other interesting things. For instance, we can essentially refer to any number n by simply adding 1 n times:

$$3 \quad \triangleq \quad (1+1)+1 \qquad\qquad 5 \quad \triangleq \quad (((1+1)+1)+1)+1$$

And in general:

$$n \quad \triangleq \quad (\dots(1+1)\dots)+1 \quad (n \text{ times})$$

We can also express interesting notions like *primeness*:

$$P(x) \quad \triangleq \quad \neg x = 1 \wedge \forall y \forall z \big(y * z = x \to (y = 1 \vee z = 1) \big)$$

Then we can state interesting facts, such as that there are infinitely many primes:

$$\forall x \exists y \big(x < y \wedge P(y) \big)$$

Or, unraveling the definitions of both $<$ and P, we would have the full formula:

$$\forall x \exists y \Big(\exists z (z \neq 0 \wedge x + z = y) \wedge \forall z \forall w \big(z * w = y \to (z = 1 \vee w = 1) \big) \Big)$$

This uses only our two constant symbols 0 and 1, and our two function symbols $+$ and $*$. This is a true statement about $\mathbb{N}$. Whether the following holds is a major open question in mathematics:

$$\forall x \exists y \exists z \big(x < y \wedge P(y) \wedge P(z) \wedge (y + 2 = z) \big)$$

Or again, not using the defined 2, $<$, or P (but taking up two lines!):

$$\forall x \exists y \exists z \Big(\exists z (x + z = y) \wedge \forall v \forall w \big(v * w = y \to (v = 1 \vee w = 1) \big) \wedge$$

$$\forall v \forall w \big(v * w = z \to (v = 1 \vee w = 1) \big) \wedge (y + 1) + 1 = z \Big)$$

This is known as the famous Twin Primes Conjecture. As of the time of writing these notes, the following is known:

$$\forall x \exists y \exists z \big(x < y \wedge P(y) \wedge P(z) \wedge (y + 246 = z) \big)$$

We will return to first order arithmetic in §6.7.

A first order language is determined by the following *non-logical parameters*, essentially choices that depend on the intended domain of application:

- A set $\mathsf{Con} = \{c_1, c_2, \dots, d, e, \dots\}$ of constant symbols.
- A set $\mathsf{Rel} = \{R_1, R_2, \dots, Q, P, \dots\}$ of relation symbols, of specified arities.
- A set $\mathsf{Fun} = \{f_1, f_2, \dots, g, h, \dots\}$ of function symbols, with specified arities.

Given these non-logical parameters, we first define the set of terms for a first order language:

Definition 6.2.1. Given non-logical parameters Con and Rel, and an infinite set of *individual variables*, $\mathsf{Var} = \{x_1, x_2, \dots, y, z, \dots\}$, the set $\mathcal{T}$ of all terms is given as follows:

1. Each variable $x \in \mathsf{Var}$ is a term.
2. Each constant symbol $c \in \mathsf{Con}$ is a term.
3. If $t_1, \dots, t_n$ are terms and $f \in \mathsf{Fun}$ is an n-ary function symbol, then $f(t_1, \dots, t_n)$ is a term.
4. Nothing else is a term.

Definition 6.2.1 allows us to generate terms freely by applying function symbols to any constants or variables, and to apply function symbols to the results of this procedure, and so on, up to any depth. For instance, in the arithmetical case, an example of a complex term would be:

$$(((x + 0) * ((y + y) * 1)) * (1 + 1)) * x$$

And so on. In the mother/father example, we could also build arbitrarily complex terms, e.g., $f(f(m(f(m(m(x))))))$, and so on. Given the set $\mathcal{T}$ of terms, we can then define the full first order language $\mathcal{L}^{\forall\exists}$:

Definition 6.2.2. Suppose we are given non-logical parameters Con, Rel, and Fun, and that $\mathcal{T}$ is the set of terms built up from Con and Rel according to Def. 6.2.1. Then the language $\mathcal{L}^{\forall\exists}$ is determined by the following set of well formed formulas (wffs):

1. If $t_1, t_2 \in \mathcal{T}$ are terms, then $t_1 = t_2$ is a wff.

2. If $t_1, \ldots, t_n \in \mathcal{T}$ are terms, and $R \in \mathsf{Rel}$ is an n-place relation, then $R(t_1, \ldots, t_n)$ is a wff.

3. If $x \in \mathsf{Var}$, and φ and ψ are wffs, the following are also wffs:

$$\neg\varphi \qquad (\varphi \vee \psi) \qquad \forall x\varphi$$
$$(\varphi \wedge \psi) \qquad (\varphi \to \psi) \qquad \exists x\varphi$$

4. Nothing else is a wff in $\mathcal{L}^{\forall\exists}$.

This definition of the language allows us to take any well formed formula whatsoever and append a universal or existential quantifier to the front. That is, we can form wffs like $\forall x P(y)$, or $\exists x R(c, d)$, even when x does not occur in the inside subformula. It also allows us to write all the formulas used as examples in the preceding section, where in each case we have a different set of non-logical parameters. Once we set these non-logical parameters the first order language $\mathcal{L}^{\forall\exists}$ is completely determined, following Definition 6.2.2.

6.3 Models of First Order Logic

In §6.1 the "models" in which we were interpreting formulas were taken as intuitive: the set of US states with its "borders" relation, the natural numbers structure $\mathbb{N}$ with its "less than" relation, and so on. In this section we make the notion of *model* more precise, so that we can say precisely when a formula is true in a model, much like we did in modal logic.

The main component of a first order model is the *domain*, a non-empty set D of entities. These entities could be anything: possible worlds, time points, states, numbers, people, subatomic particles, journal articles, or anything else we might like to reason about. The intended domain, like the non-logical parameters, will generally depend on the application.

We can also think of D as having some implicit structure. The only structure we will care about is what we can talk about with our formal logical language. So, e.g., if we have a constant symbol c, we think of this as "denoting" or "referring to" some entity in the domain. Similarly, if we have, say, a binary relation symbol R, then we think of this as picking out some relation on D, i.e., some set of pairs of elements of D. This is indeed what we were doing in the state case: we thought of the borders relation B as holding of exactly those pairs of states that in fact border each other. Similar with function symbols: e.g., we thought of the "father of" function f as denoting a genuine function on the domain of people, taking a person as argument and returning that person's father as value.

The second component of a model makes the intuitions precise. In addition to a domain D, we also have an *interpretation function* $\mathcal{I}$ sending each non-logical parameter to a mathematical object of the appropriate sort, using the set D. For instance, where c is a constant, $\mathcal{I}(c) \in D$. That is, a constant is simply interpreted as an element of D. A relation, however, is a slightly more complex object. For instance, a binary relation symbol R should be interpreted as a genuine binary relation: $\mathcal{I}(R) \subseteq D \times D$. And in general, an n-ary relation symbol P should be interpreted as an n-ary relation on D: we should have $\mathcal{I}(P) \subseteq D \times \cdots \times D$ (n times D).

Returning once more to the states example, if we were to formalize the model in this case, our domain would be the set of all states (in the lower 48):

$$D = \{\text{alabama, arizona, } \ldots, \text{ wyoming}\}$$

The interpretation of the relation B—$\mathcal{I}(B)$—would be the set of all those pairs that border:

$$\mathcal{I}(B) = \{\langle\text{alabama, florida}\rangle, \langle\text{florida, alabama}\rangle, \langle\text{california, oregon}\rangle, \ldots, \langle\text{wyoming, montana}\rangle\}$$

And finally, the interpretations of the constants are totally straightforward:

$$\mathcal{I}(c) = \text{california}, \quad \mathcal{I}(n) = \text{nevada}, \quad \mathcal{I}(o) = \text{oregon},$$

$$\mathcal{I}(t) = \text{texas}, \quad \mathcal{I}(a) = \text{arkansas}, \quad \mathcal{I}(l) = \text{louisiana}$$

Function symbols similarly get mapped by $\mathcal{I}$ to *genuine functions* on the domain D. If f is a unary function symbol, then $\mathcal{I}(f) : D \to D$ is a function from D to D. More generally, if g is an n-ary function symbol, then $\mathcal{I}(g) : D \times \cdots \times D \to D$ is an n-ary function, taking n-tuples of elements from D and returning another element from D.

For example, in the natural number case, our domain is simply the set $\mathbb{N}$. How do we interpret $+$? This is actually obvious: it is the function that takes two numbers and returns their sum! So $\mathcal{I}(+) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is the function such that $\mathcal{I}(+)(n, m) = n + m$, for any two numbers $n, m \in \mathbb{N}$. This means, e.g., $\mathcal{I}(+)(2, 3) = 5$.

Instead of writing $\mathcal{I}(c)$, $\mathcal{I}(R)$, or $\mathcal{I}(f)$, we will instead write the $\mathcal{I}$ as a superscript: $c^{\mathcal{I}}$, $R^{\mathcal{I}}$, and $f^{\mathcal{I}}$. This will facilitate writing these functions in a more familiar way, so, e.g., we can write $2 +^{\mathcal{I}} 3$ instead of $\mathcal{I}(+)(2, 3)$, or $B^{\mathcal{I}}(\text{california, oregon})$ instead of $\langle\text{california, oregon}\rangle \in \mathcal{I}(B)$.

It is important to understand the difference between what we are saying in the formal language and what we are saying outside the formal language (in the "metalanguage"). Statements like $0 +^{\mathcal{I}} 1 = 1$ are statements of ordinary arithmetic, in the metalanguage, since $+^{\mathcal{I}}$ is the *interpretation* of the $+$ symbol, and it is interpreted as ordinary addition. So this is just the same as saying $0+1 = 1$. By contrast, a statement like $0 + 1 = 1$ is in the formal language $\mathcal{L}^{\forall\exists}$. Of course, we will want these to be closely related, so that $0 + 1 = 1$ basically *means* that $0 +^{\mathcal{I}} 1 = 1$, i.e., $0 + 1 = 1$.

The final ingredient we need, in order to say when a formula is true in a model, is the notion of a *variable assignment*, a function $g : \mathsf{Var} \to D$ sending individual variables to elements of the domain. This is important if we want to say when a formula like $P(x)$ is true or false. If we do not know what x refers to, then we cannot say whether it is true or false. Variable assignments give us this information. This is similar to the assignment of constants to elements by $\mathcal{I}$, but it is going to be especially critical that these variable assignments can *vary* when we use quantifiers.

If $g : \mathsf{Var} \to D$ is a particular variable assignment, then let g^x_a be another variable assignment,

which is exactly like g, except that g_a^x sends variable x to element a of the domain. That is:

$$g_a^x(y) = \begin{cases} g(y) & \text{if } y \neq x \\ a & \text{if } y = x \end{cases}$$

Note that g may already have sent x to a. In that case these two assignment functions are the same.

To see an example of this assignment variation, let us return to the state example, with domain given by the set of states, and so on. If our variables are $\mathsf{Var} = \{x_1, x_2, x_3, \dots\}$, an example of a variable assignment might be a function $g : \mathsf{Var} \to D$ such that $g(x_1) =$ alabama, while $g(x_2) =$ vermont, and for all other $i \geqslant 3$, we have $g(x_i) =$ utah. That is, g is the function that maps variables to elements of the domain as follows:

$$\begin{aligned} x_1 &\mapsto \text{alabama} \\ x_2 &\mapsto \text{vermont} \\ x_3 &\mapsto \text{utah} \\ x_4 &\mapsto \text{utah} \\ x_5 &\mapsto \text{utah} \\ &\vdots \end{aligned}$$

We can then talk about a new function, $g^{x_4}_{\text{florida}}$, which is exactly like g, except that it sends variable x_4 to florida instead of utah. So this function $g^{x_4}_{\text{florida}}$ looks like:

$$\begin{aligned} x_1 &\mapsto \text{alabama} \\ x_2 &\mapsto \text{vermont} \\ x_3 &\mapsto \text{utah} \\ x_4 &\mapsto \text{florida} \\ x_5 &\mapsto \text{utah} \\ &\vdots \end{aligned}$$

We could further shift this function, to talk about yet another function, $(g^{x_4}_{\text{florida}})^{x_1}_{\text{delaware}}$, which is exactly like $g^{x_4}_{\text{florida}}$, except that it sends x_1 to delaware:

$$\begin{aligned} x_1 &\mapsto \text{delaware} \\ x_2 &\mapsto \text{vermont} \\ x_3 &\mapsto \text{utah} \\ x_4 &\mapsto \text{florida} \\ x_5 &\mapsto \text{utah} \\ &\vdots \end{aligned}$$

And we could continue on like this, further shifting the function away from the one we started with. These assignment shifts will prove very useful when defining what quantifiers do.

Definition 6.3.1. Suppose we are given a model $\mathfrak{M} = (D, \mathcal{I})$ and a variable assignment $g : \mathsf{Var} \to D$, and that $\mathcal{T}$ is the set of terms of $\mathcal{L}^{\forall\exists}$ (for some choice of non-logical parameters). Then we define the **interpretation** of a term t in the model $\mathfrak{M}$, under assignment g—written $[\![t]\!]^{\mathfrak{M}}_g$—as follows:

1. If $t = c \in \mathsf{Con}$ is a constant, then $[\![c]\!]^{\mathfrak{M}}_g = c^{\mathcal{I}}$.
2. If $t = x \in \mathsf{Var}$ is a variable, then $[\![x]\!]^{\mathfrak{M}}_g = g(x)$.
3. If $t = f(t_1, \ldots, t_n)$ is a complex term, resulting from application of a function symbol f to terms $t_1, \ldots, t_n$, then $[\![f(t_1, \ldots, t_n)]\!]^{\mathfrak{M}}_g = f^{\mathcal{I}}([\![t_1]\!]^{\mathfrak{M}}_g, \ldots, [\![t_n]\!]^{\mathfrak{M}}_g)$.

The interpretation of a term t in the model should be understood as a specification of its *formal meaning*. To illustrate, let us return to the example of the mother/father language with two unary functions f and m, and suppose we also have some individual constants: t and j, denoting Talia Shire and Jason Schwartzmann, respectively. Let our domain D be the set of all living people. What we mean when we say that we are interpreting the constants in this way is that $t^{\mathcal{I}}$ = Talia Shire, and $j^{\mathcal{I}}$ = Jason Schwartzman. Then by the above definition, where $\mathfrak{M} = (D, \mathcal{I})$, we also have $[\![t]\!]^{\mathfrak{M}}_g$ = Talia Shire, while $[\![j]\!]^{\mathfrak{M}}_g$ = Jason Schwartzman. What about $[\![m(j)]\!]^{\mathfrak{M}}_g$? According to Def. 6.3.1, this should be $m^{\mathcal{I}}([\![j]\!]^{\mathfrak{M}}_g)$. We of course know what $[\![j]\!]^{\mathfrak{M}}_g$ is: this is just Jason Schwartzman himself. What about $m^{\mathcal{I}}$? This is interpreted as the mother-of function, taking any person and returning that person's mother. So in this case, because the mother of Jason Schwartzman is Talia Shire, we have $[\![m(j)]\!]^{\mathfrak{M}}_g$ = Talia Shire. Thus, in fact, $[\![m(j)]\!]^{\mathfrak{M}}_g = [\![t]\!]^{\mathfrak{M}}_g$.

Definition 6.3.2. Suppose $\mathfrak{M} = (D, \mathcal{I})$ is a model, a pair of a domain and an interpretation function appropriate to the non-logical parameters. And suppose $g : \mathsf{Var} \to D$ is a variable assignment. We define $\mathfrak{M} \models \varphi\ [g]$—that φ is true in $\mathfrak{M}$ under assignment g—as follows:

$$
\begin{array}{rll}
\mathfrak{M} \models t_1 = t_2\ [g] & \text{iff} & [\![t_1]\!]^{\mathfrak{M}}_g = [\![t_2]\!]^{\mathfrak{M}}_g. \\
\mathfrak{M} \models P(t_1, \ldots, t_n)\ [g] & \text{iff} & P^{\mathcal{I}}([\![t_1]\!]^{\mathfrak{M}}_g, \ldots, [\![t_n]\!]^{\mathfrak{M}}_g). \\
\mathfrak{M} \models \neg\varphi\ [g] & \text{iff} & \text{it is not the case that } \mathfrak{M} \models \varphi\ [g]. \\
\mathfrak{M} \models \varphi \wedge \psi\ [g] & \text{iff} & \text{both } \mathfrak{M} \models \varphi\ [g] \text{ and } \mathfrak{M} \models \psi\ [g]. \\
\mathfrak{M} \models \varphi \vee \psi\ [g] & \text{iff} & \text{either } \mathfrak{M} \models \varphi\ [g] \text{ or } \mathfrak{M} \models \psi\ [g]. \\
\mathfrak{M} \models \varphi \to \psi\ [g] & \text{iff} & \text{either not } \mathfrak{M} \models \varphi\ [g] \text{ or } \mathfrak{M} \models \psi\ [g]. \\
\mathfrak{M} \models \forall x\varphi\ [g] & \text{iff} & \text{for every } d \in D : \mathfrak{M} \models \varphi\ [g^x_d] \\
\mathfrak{M} \models \exists x\varphi\ [g] & \text{iff} & \text{for some } d \in D : \mathfrak{M} \models \varphi\ [g^x_d]
\end{array}
$$

The crucial clauses here of course are for atomic formulas—equality statements or predications—and for quantificational statements. For equality statements we have just seen an example in the previous paragraph. In the language with two constants, t and j, and two function symbols, f and

m, and in the model where D is the set of all living people and $\mathcal{I}$ is defined as above, we have that $\mathfrak{M} \models t = m(j)\ [g]$.[1] Why? Because for this to hold, we need that $[\![m(j)]\!]^{\mathfrak{M}}_g = [\![t]\!]^{\mathfrak{M}}_g$, which is true.

To see an example of the second clause, return to the state border scenario. Suppose our model $\mathfrak{M} = (D, \mathcal{I})$ is as given above: D is the set of states and $\mathcal{I}$ interprets B as the real border relation, and the constants as their corresponding states. Let g be the variable assignment on $\mathsf{Var} = \{x_1, x_2, \dots\}$ above that sends x_1 to alabama, x_2 to vermont, and all the rest to utah. Then consider the formula $B(x_4, n)$. Is this formula true in $\mathfrak{M}$ under g? That is, does $\mathfrak{M} \models B(x_4, n)\ [g]$ hold? According to Def. 6.3.2, we have:

$$\begin{aligned}
\mathfrak{M} \models B(x_4, n)\ [g] \quad &\text{iff} \quad B^{\mathcal{I}}([\![x_4]\!]^{\mathfrak{M}}_g, [\![n]\!]^{\mathfrak{M}}_g) \\
&\text{iff} \quad B^{\mathcal{I}}(g(x_4), n^{\mathcal{I}}) \\
&\text{iff} \quad B^{\mathcal{I}}(\text{utah}, \text{nevada}) \\
&\text{iff} \quad \text{Utah borders Nevada}
\end{aligned}$$

Because in the model Utah does in fact border Nevada, this does hold.

Finally, let us consider some examples of sentences involving the quantifiers. Continuing with the same model, the same non-logical parameters, and the same assignment function g, recall we said earlier that the formula $\forall x \exists y B(x, y)$ is true. This is because we have:

$$\begin{aligned}
\mathfrak{M} \models \forall x \exists y B(x, y)\ [g] \quad &\text{iff} \quad \text{for every state } d \in D : \mathfrak{M} \models \exists y B(x, y)\ [g^x_d] \\
&\text{iff} \quad \text{for every state } d \in D, \text{ there is some state } e \in D : \mathfrak{M} \models B(x, y)\ [(g^x_d)^y_e] \\
&\text{iff} \quad \text{for every } d \in D, \text{ there is some } e \in D : B^{\mathcal{I}}([\![x]\!]^{\mathfrak{M}}_{(g^x_d)^y_e}, [\![y]\!]^{\mathfrak{M}}_{(g^x_d)^y_e}) \\
&\text{iff} \quad \text{for every } d \in D, \text{ there is some } e \in D : B^{\mathcal{I}}\big((g^x_d)^y_e(x), (g^x_d)^y_e(y)\big) \\
&\text{iff} \quad \text{for every } d \in D, \text{ there is some } e \in D : B^{\mathcal{I}}(d, e) \\
&\text{iff} \quad \text{for every state } d \in D, \text{ there is some } e \in D : d \text{ borders } e.
\end{aligned}$$

Of course, this is exactly what we thought it meant before. But now we have a way of making this formally precise and systematic. Consider the other formula, which we said was not true: $\forall x \exists y \exists z \big(B(x, y) \wedge B(x, z) \wedge \neg y = z\big)$. We can now see in a formal sense why this fails. (In the following let h be the assignment function that we would write as $((g^x_d)^y_e)^z_b$. It will save significant space to rename it in this way.)

[1] Note that this holds for any variable assignment whatsoever.

$\mathfrak{M} \models \forall x \exists y \exists z \big(B(x,y) \wedge B(x,z) \wedge \neg y = z\big)\ [g]$ iff for every $d \in D$:

$\mathfrak{M} \models \exists y \exists z \big(B(x,y) \wedge B(x,z) \wedge \neg y = z\big)\ [g^x_d]$

iff for every $d \in D$, there is $e \in D$:

$\mathfrak{M} \models \exists z \big(B(x,y) \wedge B(x,z) \wedge \neg y = z\big)\ [(g^x_d)^y_e]$

iff for every $d \in D$, there are $e, b \in D$:

$\mathfrak{M} \models B(x,y) \wedge B(x,z) \wedge \neg y = z\ [h]$

iff for every $d \in D$, there are $e, b \in D$: $B^{\mathcal{I}}([\![x]\!]^{\mathfrak{M}}_h, [\![y]\!]^{\mathfrak{M}}_h)$ and $B^{\mathcal{I}}([\![x]\!]^{\mathfrak{M}}_h, [\![z]\!]^{\mathfrak{M}}_h)$ and $[\![y]\!]^{\mathfrak{M}}_h \neq [\![z]\!]^{\mathfrak{M}}_h$

iff for every $d \in D$, there are $e, b \in D$: $B^{\mathcal{I}}\big(h(x), h(y)\big)$ and $B^{\mathcal{I}}\big(h(x), h(z)\big)$ and $h(y) \neq h(z)$

iff for every $d \in D$, there are $e, b \in D$: $B^{\mathcal{I}}(d, e)$ and $B^{\mathcal{I}}(d, b)$ and $e \neq b$

iff for every state d, there are *distinct* e and b (i.e., $e \neq b$) such that d borders e and d borders b

Recall that this is not true, even of the lower 48, because Maine borders only New Hampshire.

Note that in many of these examples the initial assignment function that we start with makes no difference. The only way it is used is when the quantifiers $\forall$ and $\exists$ require shifting around the assignment function, to "point it" at different elements of the domain. Intuitively, the universal quantifier requires us to point it at every single object d in the domain and check that the formula is still true when we are thinking of x as referring to that object d. The existential quantifier requires that the formula will be true for at least one way of "pointing" x at an object in the domain.

We can define the *free variables* of any formula, in a straightforward way.

Definition 6.3.3. The set of free variables of a term t, written $FV(t)$, is defined:

1. $FV(x) = \{x\}$ for any variable $x \in \mathsf{Var}$.
2. $FV(c) = \varnothing$ for any constant $c \in \mathsf{Con}$.
3. $FV\big(f(t_1, \ldots, t_n)\big) = FV(t_1) \cup \cdots \cup FV(t_n)$.

Then we can use this definition to define the set of free variables of a formula φ, written $FV(\varphi)$:

1. $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$.
2. $FV\big(P(t_1, \ldots, t_n)\big) = FV(t_1) \cup \cdots \cup FV(t_n)$.
3. $FV(\neg\varphi) = FV(\varphi)$.
4. $FV(\varphi \wedge \psi) = FV(\varphi \vee \psi) = FV(\varphi \to \psi) = FV(\varphi) \cup FV(\psi)$.

5. $FV(\forall x\varphi) = FV(\exists x\varphi) = FV(\varphi) - \{x\}$.

In other words, the free variables of a formula are just the variables that do not get *bound* by a quantifier. For instance, x is free in $R(x,c)$, but not in $\exists x R(x,c)$. It is also free in, e.g., $\forall y \exists x P(y) \to R(y,x)$, but not in $\forall y \exists x \big(P(y) \to R(y,x)\big)$. If x occurs both free and bound in φ, then it is still among the free variables of φ. So, e.g., $FV\big(\exists x Px \vee \exists y R(x,y)\big) = \{x\}$.

Exercise 6.3.1. In each of the formulas below, identify all the free variables:

1. $Q(x,y,z)$
2. $\exists y P(x)$
3. $\exists x \forall x \big(R(x,x) \vee P(x)\big)$
4. $\exists x \forall y \big(Q(x,y,z) \to \exists z P(z)\big)$
5. $\forall x \big(\exists y P(x,y) \vee \exists w Q(x,y,w)\big)$
6. $\forall x \big(\exists x R(x,x) \to P(x)\big)$
7. $\forall x \exists y \forall z Q(x,y,z) \to \exists w S(x,y,w)$
8. $\forall x \exists y \forall z \big(Q(x,y,z) \to S(x,y,w)\big)$

It is possible to show that if $FV(\varphi) = \varnothing$, then the assignment function makes no difference: for any model $\mathfrak{M}$, if $\mathfrak{M} \models \varphi\ [g]$ for some assignment g, then we also have $\mathfrak{M} \models \varphi\ [g]$ for any other g'. This should be intuitive. For example, when determining whether $\mathfrak{M} \models \forall x P(x)\ [g]$, we have to check whether $\mathfrak{M} \models P(x)\ [g^x_d]$ holds for any choice of d in the domain. Because $g^x_d(x) = d$, it clearly does not matter what g sent x to, because the universal quantifier requires us to vary it anyway, to make sure that $P(x)$ holds for *anything* we want to point x to. A similar point holds for existential quantification, and indeed, any formula with no free variables (either because there are no variables, or because the variables are all bound by quantifiers) will be blind to the initial assignment function.

For the remainder of this section we go over some more abstract example models, to practice the concepts and definitions introduced so far.

Consider a domain $D = \{a,b,c,d\}$, and a language with non-logical parameters as follows:

- Constants c, t.
- A unary function symbol f.
- A unary predicate symbol P.
- A ternary predicate symbol Q.

And finally us define a particular interpretation function $\mathcal{I}$ for the non-logical parameters:

- $c^{\mathcal{I}} = c$ and $t^{\mathcal{I}} = a$.
- $P^{\mathcal{I}} = \{a,b,c\}$
- $f^{\mathcal{I}}(a) = b$, $f^{\mathcal{I}}(b) = b$, $f^{\mathcal{I}}(c) = a$, $f^{\mathcal{I}}(d) = a$
- $Q^{\mathcal{I}} = \{\langle a,a,b\rangle, \langle b,a,d\rangle, \langle d,d,d\rangle, \langle c,a,d\rangle, \langle a,d,d\rangle\}$

That is, we have a domain with four objects, two of them have names in the formal language, there is a one place predicate that holds of three of the objects, a function taking two of the elements to a and two to b, and a three-place predicate that holds five different triples.

Once we specify an assignment function we will be able to talk about truth or falsity of any formula in the entire language. Let $g : \mathsf{Var} \to D$ be the assignment function sending all variables to c. That is, $g(v) = c$ for all $v \in \mathsf{Var}$. The following are all examples of formulas that are true in this model under this assignment function:

1. $\mathfrak{M} \models \exists x \exists y \exists z \big(\neg x = y \wedge \neg x = z \wedge \neg y = z \wedge P(x) \wedge P(y) \wedge P(z) \big)$ $[g]$
2. $\mathfrak{M} \models f(c) = t$ $[g]$
3. $\mathfrak{M} \models \forall x \forall y \big(Q(z,x,y) \rightarrow \neg P(y) \big)$ $[g]$
4. $\mathfrak{M} \models \neg \forall x \forall y \forall z \big(Q(z,x,y) \rightarrow \neg P(y) \big)$ $[g]$
5. $\mathfrak{M} \models \forall x \big(Q(t,t,x) \rightarrow f(x) = x \big)$ $[g]$
6. $\mathfrak{M} \models \forall x \big(\forall y \exists z Q(z,y,x) \rightarrow \neg P(y) \big)$ $[g]$
7. $\mathfrak{M} \models \forall x \big(\neg f(x) = t \rightarrow P(x) \big)$ $[g]$
8. $\mathfrak{M} \models \exists y \forall x \big(\neg f(x) = y \rightarrow P(x) \big)$ $[g]$
9. $\mathfrak{M} \models \forall x \exists y \big(\neg f(x) = y \rightarrow P(x) \big)$ $[g]$
10. $\mathfrak{M} \models \forall x \forall y \big(\exists z (f(x) = z \wedge f(z) = y \wedge \neg y = z) \rightarrow f(x) = t \big)$ $[g]$

Let us see why some of these are true. Consider, e.g., 3. This formula looks counterintuitive, but it actually comes out true because of the fact that $g(z) = c$, and hence also $(g_i^x)_j^y(z) = c$. In the following let us abbreviate $(g_i^x)_j^y$ by h. We then have:

$$\begin{aligned}
\mathfrak{M} \models \forall x \forall y \big(Q(z,x,y) \rightarrow \neg P(y) \big) \; [g] \quad \text{iff} \quad & \text{for every } i,j \in D : \mathfrak{M} \models Q(z,x,y) \rightarrow \neg P(y) \; [h] \\
\text{iff} \quad & \text{for every } i,j \in D : \text{ if } Q^{\mathcal{I}}([\![z]\!]_h^{\mathfrak{M}}, [\![x]\!]_h^{\mathfrak{M}}, [\![y]\!]_h^{\mathfrak{M}}) \text{ then not } P^{\mathcal{I}}([\![y]\!]_h^{\mathfrak{M}}) \\
\text{iff} \quad & \text{for every } i,j \in D : \text{ if } Q^{\mathcal{I}}\big(h(z), h(x), h(y)\big) \text{ then not } P^{\mathcal{I}}\big(h(y)\big) \\
\text{iff} \quad & \text{for every } i,j \in D : \text{ if } Q^{\mathcal{I}}(c,i,j) \text{ then not } P^{\mathcal{I}}(j)
\end{aligned}$$

But this is of course true, because c only stands in the relation $Q^{\mathcal{I}}$ with elements a and d: we only have $\langle c, a, d \rangle \in Q^{\mathcal{I}}$. That is, the only i and j with which c stands in this relation are a and d, and of course d does not satisfy the predicate $P^{\mathcal{I}}$. So the formula is true. Next, consider 5:

$$\begin{aligned}
\mathfrak{M} \models \forall x \big(Q(t,t,x) \rightarrow f(x) = x \big) \; [g] \quad \text{iff} \quad & \text{for every } i \in D : \\
& \text{if } Q^{\mathcal{I}}([\![t]\!]_{g_i^x}^{\mathfrak{M}}, [\![t]\!]_{g_i^x}^{\mathfrak{M}}, [\![x]\!]_{g_i^x}^{\mathfrak{M}}) \text{ then } f^{\mathcal{I}}([\![x]\!]_g^{\mathfrak{M}}) = [\![x]\!]_{g_i^x}^{\mathfrak{M}} \\
\text{iff} \quad & \text{for every } i \in D : \text{if } Q^{\mathcal{I}}\big(t^{\mathcal{I}}, t^{\mathcal{I}}, g_i^x(x)\big) \text{ then } f^{\mathcal{I}}\big(g_i^x(x)\big) = g_i^x(x) \\
\text{iff} \quad & \text{for every } i \in D : \text{if } Q^{\mathcal{I}}(a,a,i) \text{ then } f^{\mathcal{I}}(i) = i
\end{aligned}$$

There is only one i such that $\langle a, a, i \rangle \in Q^{\mathcal{I}}$, namely b. And because $f^{\mathcal{I}}(b) = b$, this come out true.

Exercise 6.3.2. In the same manner as the examples above, show why 2, 4, and 7 all hold.

Exercise 6.3.3. Which of the following hold in this model $\mathfrak{M}$ under g? You do not have to show why, only whether it holds or not. (That said, you may get partial credit if you get it wrong but made only a small mistake in showing why it holds.)

1. $\forall x\big(P(x) \vee P(f(x))\big)$
2. $\exists x Q(f(x), x, x)$
3. $\exists x\big(f(x) = x \wedge \exists y \exists z Q(y, x, z)\big)$
4. $\forall x \forall y \forall z\big(Q(x, y, z) \to (y = t \vee f(y) = t)\big)$
5. $x = c$
6. $\forall x\ y = c$
7. $\exists x f(x) = y$
8. $\exists x f(f(f(x))) = x$
9. $\exists x \neg Q(x, x, x) \to f(x) = t$
10. $\exists x\big(Q(x, x, x) \wedge Q(c, t, x)\big)$

Exercise 6.3.4. Suppose we slightly change the model to a different interpretation function $\mathcal{I}'$, so

- $c^{\mathcal{I}'} = b$ and $t^{\mathcal{I}'} = d$.
- $f^{\mathcal{I}'}(a) = f^{\mathcal{I}'}(b) = f^{\mathcal{I}'}(c) = d$, $f^{\mathcal{I}'}(d) = a$
- $P^{\mathcal{I}'} = \{a, c, d\}$
- $Q^{\mathcal{I}'} = \{\langle a, a, b\rangle, \langle b, a, d\rangle, \langle d, d, d\rangle, \langle a, d, d\rangle\}$

And we change the assignment g to g', where $g'(x) = b$, $g'(y) = d$, $g'(z) = b$, and $g'(v) = a$ for all other $v \in \mathsf{Var}$. Which of 1-10 now hold from Ex. 6.3.3 in $\mathfrak{M}' = (D, \mathcal{I}')$ under g'?

6.4 Metalogical Notions

When we were working on Boolean Logic, we first defined truth of a formula under a valuation. Then we lifted this notion to talk about validity, satisfiability, etc. We did something similar in the modal setting, and we are going to do the same thing here.

Definition 6.4.1. Fix a first order language $\mathcal{L}^{\forall\exists}$ given by some set of non-logical parameters. We say $\varphi \in \mathcal{L}^{\forall\exists}$ is (first order) *valid* if $\mathfrak{M} \models \varphi\ [g]$ for all models $\mathfrak{M}$ and assignment functions g.

What are some examples of valid formulas? As in modal logic any Boolean tautology in this setting will certainly be first order valid. This includes not only tautologies like $P(c) \vee \neg P(c)$, but also formulas like

$$\big(\forall x \exists y R(x, y) \to Q(z)\big) \to \big(\neg Q(z) \to \neg \forall x \exists y R(x, y)\big)$$

This formula is an instance of the Boolean tautology $(A \to B) \to (\neg B \to \neg A)$. In fact, the similarities with Modal Logic go even further. If we think of $\forall$ as analogous to $\Box$ (which, recall, asserted something held "for all" successors) and $\exists$ as analogous to $\Diamond$, then the following should not be surprising (cf. Fact 5.1.1):

Fact 6.4.1. The following first order formulas are all valid (for any formula φ):

1. $\exists x \varphi \leftrightarrow \neg \forall x \neg \varphi$
2. $\forall x \varphi \leftrightarrow \neg \exists x \neg \varphi$
3. $\forall x(\varphi \to \psi) \leftrightarrow (\forall x \varphi \to \forall x \psi)$
4. If φ is valid, then so is $\forall x \varphi$.

But we also now have other interesting validities that do not correspond to any modal validities. One example is the following, for any term t:

$$\forall x P(x) \rightarrow P(t)$$

If $P^{\mathcal{I}}$ holds of everything, then certainly it holds of whatever the term t denotes (i.e., $t^{\mathcal{I}}$)! Using the fact above, part 1, and what we know about Boolean Logic, we can also see that this is valid:

$$P(t) \rightarrow \exists x P(x)$$

If $P^{\mathcal{I}}$ holds of $t^{\mathcal{I}}$, then it holds of something! There are also some less obvious validities, like this:

$$\forall x\big(\forall y R(y,y) \rightarrow R(x,x)\big)$$

It says, of any x, if $R^{\mathcal{I}}$ holds between any element and itself, then in particular it holds between (whatever is denoted by) x and itself. Because we only allow non-empty domains, we also have, e.g.,

$$\forall x P(x) \rightarrow \exists x P(x)$$

A most curious example of a valid formula is the following:

$$\exists x\big(P(x) \vee \forall y \neg P(y)\big)$$

This says, there exists some particular element of the domain, such that either it satisfies $P^{\mathcal{I}}$ or nothing does! To see why this is valid, consider two (exhaustive) cases: either something satisfies $P^{\mathcal{I}}$ or nothing does. If something does, then there is your witness to the existential claim, making the left disjunct true. If nothing does, then pick anything, and the right disjunct will still be true, since *ex hypothesi* everything fails to satisfy it.

If a formula is not valid, we need to be able to find a countermodel and an assignment function to prove it. Since validity asserts truth in all models at all assignment functions, to show invalidity we only need to find one counterexample. For example, the following formula is not valid:

$$\forall x\big(\exists y P(y,y) \rightarrow \exists y P(x,y)\big)$$

Here is a very small model to show it. Let the domain $D = \{a, b\}$ consist of just two elements, and define $\mathcal{I}$ so that $P^{\mathcal{I}} = \{\langle a, a\rangle\}$. And let g be any assignment function (note the formula has no free

variables). Then we have

$$\begin{array}{lll}(D,\mathcal{I}) \models \forall x\big(\exists y P(y,y) \to \exists y P(x,y)\big)\ [g] & \text{iff} & \text{for all } d \in D: \\ & & \text{if there is an } e \in D : P^{\mathcal{I}}([\![y]\!]^{(D,\mathcal{I})}_{(g^x_d)^y_e}, [\![y]\!]^{(D,\mathcal{I})}_{(g^x_d)^y_e}) \\ & & \text{then there is a } f \in D : P^{\mathcal{I}}([\![x]\!]^{(D,\mathcal{I})}_{(g^x_d)^y_f}, [\![y]\!]^{(D,\mathcal{I})}_{(g^x_d)^y_f}) \\ & \text{iff} & \text{for all } d \in D: \\ & & \text{if there is an } e \in D : P^{\mathcal{I}}(e,e) \\ & & \text{then there is a } f \in D : P^{\mathcal{I}}(d,f)\end{array}$$

If we pick d to be b, then while there is an e such that $P^{\mathcal{I}}(e,e)$—namely a—there is no f such that $P^{\mathcal{I}}(b,f)$. That is, b gives us our counterexample to the universal claim.

Or consider another invalid formula:

$$P(c) \to P(x)$$

Again, a two element domain will do. Let $D = \{a,b\}$ and define $\mathcal{I}$ so that $c^{\mathcal{I}} = a$ and $P^{\mathcal{I}} = \{\langle a\rangle\}$. Let g be any assignment function that sends x to b. Then:

$$\begin{array}{lll}(D,\mathcal{I}) \models P(c) \to P(x)\ [g] & \text{iff} & \text{if } P^{\mathcal{I}}([\![c]\!]^{(D,\mathcal{I})}_g) \text{ then } P^{\mathcal{I}}([\![x]\!]^{(D,\mathcal{I})}_g) \\ & \text{iff} & \text{if } P^{\mathcal{I}}(c^{\mathcal{I}}) \text{ then } P^{\mathcal{I}}(g(x)) \\ & \text{iff} & \text{if } P^{\mathcal{I}}(a) \text{ then } P^{\mathcal{I}}(b)\end{array}$$

which of course is false, showing that $(D,\mathcal{I}) \models P(c) \to P(x)\ [g]$ fails, and hence $P(c) \to P(x)$ is not a valid formula.

Exercise 6.4.1. Which of the following are valid first order formulas? If it is valid, provide a short argument. If it is not, provide a countermodel (as in the examples above).

1. $\forall x \forall y\big(R(x,y) \vee R(y,x)\big)$
2. $\exists x\big(P(x) \wedge Q(x)\big) \to \big(\exists x P(x) \wedge \exists x Q(x)\big)$
3. $\big(\exists x P(x) \wedge \exists x Q(x)\big) \to \exists x\big(P(x) \wedge Q(x)\big)$
4. $\big(\forall x P(x) \wedge \forall x Q(x)\big) \to \forall x\big(P(x) \wedge Q(x)\big)$
5. $\forall x\big(P(x) \vee Q(x)\big) \leftrightarrow \big(\forall x P(x) \vee \forall x Q(x)\big)$
6. $\exists x\big(P(x) \vee Q(x)\big) \leftrightarrow \big(\exists x P(x) \vee \exists x Q(x)\big)$
7. $P(x) \to \exists x P(x)$
8. $P(x) \to \forall x P(x)$
9. $\forall x P(x) \to P(z)$
10. $\forall x P(x) \vee \exists x \neg P(x)$

As with previous logical systems, we can think of validity as a special case of *entailment*, namely entailment from the empty set of premises:

Definition 6.4.2. Suppose Γ is a set of first order formulas. We write $\Gamma \Vdash \varphi$ iff: for any model $\mathfrak{M}$ and assignment function g for which $\mathfrak{M} \models \gamma\ [g]$ for each $\gamma \in \Gamma$, we also have $\mathfrak{M} \models \varphi\ [g]$.

For example, the following entailment holds:

$$\{\forall x\big(P(x) \to Q(x)\big), \neg Q(c), \big(P(c) \vee P(d)\big)\} \Vdash P(d)$$

We can see this by the following argument. Suppose we have a model $\mathfrak{M} = (D, \mathcal{I})$ and assignment function g on which all the premises are true. This means:

$$\begin{array}{rcl} \mathfrak{M} \models \forall x\big(P(x) \to Q(x)\big)\ [g] & \text{implies} & \text{for all } d \in D:\ \text{if } [\![x]\!]^{\mathfrak{M}}_{g^x_d} \in P^{\mathcal{I}} \text{ then } [\![x]\!]^{\mathfrak{M}}_{g^x_d} \in Q^{\mathcal{I}} \\ & \text{which implies} & \text{for all } d \in D:\ \text{if } g^x_d(x) \in P^{\mathcal{I}} \text{ then } g^x_d(x) \in Q^{\mathcal{I}} \\ & \text{and thus} & \text{for all } d \in D:\ \text{if } d \in P^{\mathcal{I}} \text{ then } d \in Q^{\mathcal{I}} \\ & \text{i.e.,} & P^{\mathcal{I}} \subseteq Q^{\mathcal{I}}. \end{array}$$

And because $\mathfrak{M} \models \neg Q(c)\ [g]$, working out what this means, we also have $c^{\mathcal{I}} \notin Q^{\mathcal{I}}$, and hence, because $P^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$, also $c^{\mathcal{I}} \notin P^{\mathcal{I}}$. While $\mathfrak{M} \models P(c) \vee P(d)\ [g]$ means that either $c^{\mathcal{I}} \in P^{\mathcal{I}}$ or $d^{\mathcal{I}} \in P^{\mathcal{I}}$. These latter two facts mean that $d^{\mathcal{I}} \in P^{\mathcal{I}}$, and thus that $\mathfrak{M} \models P(d)\ [g]$. Note that once we unpack what it means for a given entailment to hold, the rest of the reasoning is just "ordinary" naïve set theoretic reasoning of the sort we did in §2.

What does it take to show that some set Γ does *not* entail a formula φ? Consider the following non-entailment as an example:

$$\{Px \to \forall x P(x)\} \nVdash P(c)$$

If we believe this entailment does not hold, then we need to find a model $\mathfrak{M}$ and an assignment function g such that $\mathfrak{M} \models Px \to \forall x P(x)\ [g]$, but not $\mathfrak{M} \models P(c)\ [g]$. But this is in fact easy. We can do this with a single one-point model: let $D = \{a\}$, let $\mathcal{I}(P) = \varnothing$, $\mathcal{I}(c) = a$, and let g be the only possible function from Var to D, namely that for which $g(v) = a$ for all $v \in \mathsf{Var}$. Then $(D, \mathcal{I}) \models Px \to \forall x P(x)\ [g]$ just because $g(x) = a \notin P^{\mathcal{I}}$ and hence $\mathfrak{M} \models P(x)\ [g]$ fails. But of course we do not have $\mathfrak{M} \models P(c)\ [g]$, because $c^{\mathcal{I}} = a \notin P^{\mathcal{I}}$.

For another, more complex example, consider the following purported non-entailment:

$$\{P(c_1), Q(c_2), \exists x(P(x) \to \forall y Q(y))\} \nVdash Q(c_1)$$

Consider a model with two elements $D = \{a, b\}$ and let $\mathcal{I}$ be defined so that $\mathcal{I}(P) = \{a\}$, $\mathcal{I}(Q) = \{b\}$, $\mathcal{I}(c_1) = a$, and $\mathcal{I}(c_2) = b$. Then clearly $(D, \mathcal{I}) \models P(c_1)\ [g]$ and $(D, \mathcal{I}) \models Q(c_2)\ [g]$ (for any choice of g). Also $(D, \mathcal{I}) \models \exists x(P(x) \to \forall y Q(y))$. The witness for this is of course b. We have $\mathfrak{M} \models P(x) \to \forall y Q(y)\ [g^x_b]$ simply because $g^x_b(x) = b \notin P^{\mathcal{I}}$, so the conditional holds vacuously. Moreover, ${c_1}^{\mathcal{I}} \notin Q^{\mathcal{I}}$, which shows that the entailment indeed does not go through.

Exercise 6.4.2. Suppose we fix a first order language with a one-place predicate symbol P and a two-place predicate symbol R, as well as a one-place function symbol f, and infinitely many constant symbols $\{c_1, c_2, c_3, \dots\}$. Which of the following entailments hold? Again, justify your answer with an informal argument.

1. $\{\exists y R(c_2, y), \exists y \forall x\Big(\big(R(x, y) \to P(x)\big) \wedge \forall z\big(R(c_2, z) \to z = y\big)\Big)\} \Vdash P(c_2)$

2. $\{P(c_1), P(c_2), P(c_3), \dots\} \Vdash \forall x P(x)$

3. $\{\exists x \forall y \big(R(x,y) \to P(y)\big)\} \Vdash \forall y \exists x \big(R(x,y) \to P(y)\big)$

4. $\{\forall x \exists y f(y) = x\} \Vdash \forall x \forall y \big(f(x) = f(y) \to x = y\big)$

5. $\{\forall x R(x,x), \forall x \forall y \forall z \big((R(x,y) \wedge R(y,z)) \to R(x,z)\big)\} \Vdash \forall x \forall y \big(R(x,y) \to R(y,x)\big)$

6.5 Undecidability of First Order Logic

Recall in Boolean Logic that we had an algorithm to determine whether a given Boolean formula was valid: simply check the truth table and determine if all the rows are T. This may be rather complex, but at least it is possible in principle. That is, if we had arbitrarily much space and time for computation we would be able to determine for any Boolean formula whether it is valid. It turns out that all the modal logical systems we studied are also decidable, though some, such as the temporal logics, have complexity that is known to be worse than NP. (That modal logics are decidable should not necessarily be obvious to you at this point.)

How does First Order Logic fare on this point? It turns out—as Alonzo Church and Alan Turing independently showed in the early 1930s—the question of whether a given first order formula φ is valid is *undecidable*! That is to say, it is impossible to write a computer program that would take a first order formula as input, and (truly and effectively) output 'yes' or 'no' depending on whether the formula is valid or not. This is so even if we allow the computer program arbitrarily much time and space to compute. The aim of this section is to explain why this is so. In short, what we will do is show how to encode the operations of an arbitrary Turing Machine into first order logic, in such a way that we can reduce the problem of deciding whether a given Turing Machine will halt on a given input to the problem of deciding validity of first order formulas. Because we know the "Halting Problem" is undecidable, it will follow that determining validity (and hence also satisfiability, entailment, and so on) is also an undecidable problem.

6.5.1 A Primer on Computability and Turing Machines

This section includes everything you will need to know about Turing Machines. Of course, if you have not had experience with them before, they will remain somewhat mysterious. The important feature of this class of machines is that they are *universal*, in the sense that they can simulate any computation whatsoever that any other known computing machine can perform. In particular, given enough time and space, they can carry out any computation that your laptop computer can achieve. Because of this, if we can show that there are certain tasks that a Turing Machine cannot even in principle perform, this is good reason to believe that no machine could perform the task.

A Turing Machine consists of an infinite tape (infinite in both directions) with 0's and 1's in the cells of the tape, and with a *read/write head* pointing to a particular cell of the tape:

...	0	0	**1**	1	1	0	...

Apart from this, a machine is defined by a finite set $\mathcal{Q}$ of *states*, and a finite set of rules telling us what to do if we are in a given state and reading a particular digit (0 or 1). For instance, the following is an example of a set of rules for a machine with three states (0, 1, and 2 — state 0 is not listed because it will be a distinguished "halt state"):

	0	1
1	1L2	1R1
2	0R0	1L2

On the leftmost column are the states 1 and 2, and we always begin in state 1. Imagine we are looking at a 1 on the tape (that is, the read/write head is pointing to a 1). Then we use these instructions as follows (with the relevant instruction bolded):

	0	**1**
1	1L2	**1R1**
2	0R0	1L2

This says that, if we are in state 1 (first row), reading a 1 (right column), then we should rewrite that 1 as a 1 again, then move to the right one cell, and remain in state 1.

Likewise, if we look at, say, the top-left rule:

	0	1
1	**1L2**	1R1
2	0R0	1L2

this tells us that, if we are in state 1, reading a 0, then we should rewrite that 0 as a 1, move one step to the left, and enter state 2.

We always begin in state 1, with some input string z (a string of 0s and 1s) written on the tape, with all other cells of the tape containing 0, and with our read/write head positioned at the first (leftmost) bit of z. The computation is said to *halt* if we ever reach the distinguished halt state 0.

To see an example, imagine we run the machine described above on input 111. Then we start with three consecutive 1s on the tape, looking at the first.

...	0	0	**1**	1	1	0	...

Our first rule tells us to rewrite it as a 1, go right, and stay in state 1, so we end up with:

...	0	0	1	**1**	1	0	...

...	0	0	1	1	**1**	0	...

...	0	0	1	1	1	**0**	...

When we reach a 0, we look at the other instruction, which tells us to write the 0 as a 1, go left, and enter state 2. Continuing on in this way, we have the following series of configurations:

...	0	0	1	1	**1**	1	...

...	0	0	1	**1**	1	1	...

...	0	0	**1**	1	1	1	...

...	0	**0**	1	1	1	1	...

until we eventually reach state 0 with the following configuration:

...	0	0	**1**	1	1	1	...

At this point, because we have reached state 0, the machine is said to halt, and the output is what is given on the tape, which in this case is 1111. If you inspect this machine and try out a few other possible inputs, you will see that this machine computes the "successor" function $x \mapsto x + 1$, with inputs and outputs written in unary notation (i.e., where the number k is represented by k 1s).

Because this simple machine computes the successor function, we say that the task of determining the successor of a number is *computable*. We say, in general, that a numerical function that can be computed by a Turing Machine is computable. However, there are numerical tasks and functions that are not computable, i.e., for which there is no Turing Machine that can perform them. The most famous of these, due to Turing himself, is the so called *Halting Problem*.

This machine we described above happens to halt (i.e., reach state 0) on its input 111. But there are also machines we can define that never halt on some inputs. For instance, consider the following machine on input 0:

	0	1
1	0R1	1R0

Beginning on the leftmost bit of the input, which is just 0, we have:

...	0	0	**0**	0	0	0	...

Our table of rules tells us, if we are in state 1 looking at a 0, we rewrite it as a 0, move right, and remain in state 1:

...	0	0	0	**0**	0	0	...

But now we are back in the same situation, and it becomes clear that we will continue reading the same rule, moving right and writing a 0, over and over *ad infinitum*, never reaching state 0.

Turing was able to show that this task could actually be understood as a numerical task, but that it is not a computable task. An important step in this argument (which we skip here) is to show that all the Turing Machines M could be encoded as binary strings $\ulcorner M \urcorner$, so that we could list all of the (codes of) Turing Machines in a row:

$$\ulcorner M_1 \urcorner, \ulcorner M_2 \urcorner, \ulcorner M_3 \urcorner, \ldots$$

This allows us to state general problems about machines themselves as problems about numbers!

Given a yes/no question, let us say that the question is *undecidable* if the task is not computable. Then it can be shown that the problem of determining in general whether an arbitrary machine M_i would halt on an input z is an undecidable question.

Theorem 6.5.1 (Turing)**.** The Halting Problem is undecidable.

Proof Sketch. Suppose for a contradiction that the problem is decidable, so that there is a machine—call it M_h—that takes as input the code $\ulcorner M_i \urcorner$ of any machine M_i, halts and returns 1 if M_i halts on its own code $\ulcorner M_i \urcorner$ and 0 otherwise. Given M_h, it would be easy to define a second machine $M_!$, which does exactly the same thing as M_h, but instead goes into an infinite loop when M_h would halt with output 1. In other words, $M_!$ (which is easily seen to exist, provided that M_h does) takes as input a code $\ulcorner M_i \urcorner$ of a machine, loops of M_i halts on $\ulcorner M_i \urcorner$, and halts if M_i loops on $\ulcorner M_i \urcorner$.

Now, assuming that $M_!$ exists, it also has a code $\ulcorner M_! \urcorner$. Let us consider the question of whether $M_!$ halts on $\ulcorner M_! \urcorner$. Suppose it does. Then by the way $M_!$ was defined, it should loop when given itself as input. But if it loops on its own code as input, then it should halt on that same input! In other words, $M_!$ halts on $\ulcorner M_! \urcorner$ if and only if it does not halt on $\ulcorner M_! \urcorner$, which is a contradiction. Thus $M_!$ could not possibly exist, which means M_h could not have existed. In other words, there is no machine that decides the Halting Problem, and thus the problem is undecidable. □

Given the undecidability of the Halting Problem, a convenient way to show other problems are undecidable is to show that, if we could solve these problem we would be able to solve the Halting Problem. In other words, we *reduce* these problems to the Halting Problem, showing they are just as hard. This will be our strategy for showing that the validity problem for First Order Logic is undecidable: if we could decide whether an arbitrary formula of First Order Logic was valid, we would be able to decide the Halting Problem.

6.5.2 Encoding Turing Computations into First Order Logic

Suppose we are given a machine M with states $\mathcal{Q}$ and an input string z. From this we are going to build a single formula $\varphi_{M,z}$, which will be a valid first order formula if and only if M halts on z. We use a language tailor-made for the purpose of describing Turing Machine computations. In particular we have the following non-logical parameters:

- Two constant symbols: $0, 1$
- Two unary function symbols: $\overline{0}, \overline{1}$
- A binary relation symbol R_q for each state $q \in \mathcal{Q}$

We can of course interpret this language in any model whatsoever that provides an interpretation of the constants, function symbols, and relation symbols. But there is one kind of model in particular that has particular interest. These are the "intended models" that capture the idea that we are talking about Turing Machine computations. They are of the form $\mathfrak{M}_{M,z} = (D, \mathcal{I})$, and are defined relative to a machine M and an input z:

- D is the set of all finite strings over the alphabet $\{0, 1\}$, i.e., all sequences $n_1 n_2 \ldots n_k$ of all lengths $k > 0$, where each $n_i \in \{0, 1\}$.
- $\mathcal{I}$ is defined so that:

- $\mathcal{I}(0) = 0$ and $\mathcal{I}(1) = 1$
- $\mathcal{I}(\overline{0})$ is that function which takes a string $w \in D$ and appends 0 to the beginning, yielding a new string $0w$. $\mathcal{I}(\overline{1})$ is defined analogously, with $\mathcal{I}(\overline{1})(w) = 1w$.
- For each $q \in \mathcal{Q}$, we define $\mathcal{I}(R_q)$ to consist of those pairs $\langle x, y\rangle$ such that it is possible for our machine to begin with input z on the tape and reach state q with y on the tape and the read/write head on the first symbol of the string y, and with x on the tape *in reserve order* to the left. [This is the only one that depends on M and z.]

Clearly we always have $\mathfrak{M} \models R_1(0, z)$; that is, we can reach state 1 while reading the first symbol of the input z, and with a 0 to the left (because we always begin this way). We now build up a number of further formulas that describe the rules of our machine M.

For each rule of the form in state q, when reading n, write m, go **right**, and enter state q', we include a formula that essentially says this:

$$\forall x \forall y \Big(R_q\big(x, \overline{n}(y)\big) \to R_{q'}\big(\overline{m}(x), y\big)\Big)$$

That is, if we can reach state q looking at $n \in \{0, 1\}$, with x (in reverse) to the left and y to the right, then we can also reach state q' looking at the first symbol of y, and with (what was n but is now) m to the left and the rest of x (again, in reverse) to the left of that.

We likewise have formulas for rules in which the injunction is to go left instead of right, that is, of the form, in state q, when reading n, write m, go **left**, and enter state q':

$$\forall x \forall y \Big(R_q\big(\overline{p}(x), \overline{n}(y)\big) \to R_{q'}\big(x, \overline{p}\big(\overline{m}(y)\big)\big)\Big)$$

Finally, we need one more set of formulas saying, for each rule q and each $n \in \{0, 1\}$, if we get to the end of a sequence reading n, we can always extend it by adding a 0 to the beginning or end (since everything outside of the bits we are considering at any given time is a 0):

$$\forall x \big(R_q(n, x) \to R_q(\overline{n}(0), x)\big) \wedge \forall x \big(R_q(x, n) \to R_q(x, \overline{n}(0))\big)$$

Let σ be the conjunction of all of these formulas for all the instructions. Then our formula $\varphi_{M,z}$ is:

$$\varphi_{M,z} \quad \triangleq \quad \big(\sigma \wedge R_1(0, z)\big) \to \exists x \exists y R_0(x, y)$$

That is, if we have a machine that satisfies all of these rules, and we start with the assumption that we can reach state 1 while looking at the first bit of the input z, then we can eventually reach the halting state 0, i.e., there are strings x and y such that we can reach state 0 with these on the tape.

Proposition 6.5.1. For any machine M and input z, the formula $\varphi_{M,z}$ is valid iff M halts on z.

Proving this rigorously is somewhat involved. One direction is reasonably straightforward. If $\varphi_{M,z}$ is valid, then in particular it is true in the "standard model" described above, which tells us that the machine does indeed reach state 0. But in the other direction, we need to show that if M

halts on z, then $\varphi_{M,z}$ is valid. This is not terribly difficult either, but it takes a bit more work, which we skip here. Instead, we give a sense for why it is true by example.

Let us first return again to the example from the previous discussion:

	0	1
1	1L2	1R1
2	0R0	1L2

In this example we have three relation symbols R_0, R_1, and R_2. Now consider the input 11. We want to show the following formula is valid:

$$\begin{aligned}
&\Big(\forall x\forall y\Big(R_2\big(\overline{0}(x),\overline{1}(y)\big) \to R_2\big(x,\overline{0}\big(\overline{1}(y)\big)\big)\Big)\wedge\\
&\quad\forall x\forall y\Big(R_2\big(\overline{1}(x),\overline{1}(y)\big) \to R_2\big(x,\overline{1}\big(\overline{1}(y)\big)\big)\Big)\wedge\\
&\quad\forall x\forall y\Big(R_1\big(\overline{0}(x),\overline{0}(y)\big) \to R_2\big(x,\overline{0}\big(\overline{1}(y)\big)\big)\Big)\wedge\\
&\quad\forall x\forall y\Big(R_1\big(\overline{1}(x),\overline{0}(y)\big) \to R_2\big(x,\overline{1}\big(\overline{1}(y)\big)\big)\Big)\wedge\\
&\quad\forall x\forall y\Big(R_1\big(x,\overline{1}(y)\big) \to R_1\big(\overline{1}(x),y\big)\Big)\wedge\\
&\quad\forall x\forall y\Big(R_2\big(x,\overline{0}(y)\big) \to R_0\big(\overline{0}(x),y\big)\Big)\wedge\\
&\forall x\big(R_1(0,x)\to R_1(\overline{0}(0),x)\big)\wedge\forall x\big(R_1(x,0)\to R_1(x,\overline{0}(0))\big)\wedge\\
&\forall x\big(R_1(1,x)\to R_1(\overline{1}(0),x)\big)\wedge\forall x\big(R_1(x,1)\to R_1(x,\overline{1}(0))\big)\wedge\\
&\forall x\big(R_2(0,x)\to R_2(\overline{0}(0),x)\big)\wedge\forall x\big(R_2(x,0)\to R_2(x,\overline{0}(0))\big)\wedge\\
&\forall x\big(R_2(1,x)\to R_2(\overline{1}(0),x)\big)\wedge\forall x\big(R_2(x,1)\to R_2(x,\overline{1}(0))\big)\wedge\\
&\qquad R_1\big(0,\overline{1}(1)\big)\Big) \quad\to\quad \exists x\exists y R_0(x,y)
\end{aligned}$$

Fact 6.5.1. This formula is valid.

Proof. Suppose we have a model $\mathfrak{M}$ that makes the antedent true. In particular, $\mathfrak{M}\models R_1\big(0,\overline{1}(1)\big)$, and because $\mathfrak{M}\models\forall x\forall y\Big(R_1\big(x,\overline{1}(y)\big)\to R_1\big(\overline{1}(x),y\big)\Big)$, we have $\mathfrak{M}\models R_1\big(\overline{1}(0),1\big)$. Moreover, because $\mathfrak{M}\models\forall x\big(R_1(x,1)\to R_1(x,\overline{1}(0))\big)$ we also have $\mathfrak{M}\models R_1\big(\overline{1}(0),\overline{1}(0)\big)$, and thus again $\mathfrak{M}\models R_1\big(\overline{1}(\overline{1}(0)),0\big)$. Because $\mathfrak{M}\models\forall x\big(R_1(x,0)\to R_1(x,\overline{0}(0))\big)$ we have $\mathfrak{M}\models R_1\big(\overline{1}(\overline{1}(0)),\overline{0}(0)\big)$, in which case we can use the fact that $\mathfrak{M}\models\forall x\forall y\Big(R_1\big(\overline{1}(x),\overline{0}(y)\big)\to R_2\big(x,\overline{1}\big(\overline{1}(y)\big)\big)\Big)$ and obtain $\mathfrak{M}\models R_2\big(\overline{1}(0),\overline{1}(\overline{1}(0))\big)$. Because $\mathfrak{M}\models\forall x\forall y\Big(R_2\big(\overline{1}(x),\overline{1}(y)\big)\to R_2\big(x,\overline{1}\big(\overline{1}(y)\big)\big)\Big)$, we know that $\mathfrak{M}\models R_2\big(0,\overline{1}(\overline{1}(\overline{1}(0)))\big)$. Since $\mathfrak{M}\models\forall x\big(R_2(0,x)\to R_2(\overline{0}(0),x)\big)$ we obtain that $\mathfrak{M}\models R_2\big(\overline{0}(0),\overline{1}(\overline{1}(\overline{1}(0)))\big)$. And since $\mathfrak{M}\models\forall x\forall y\Big(R_2\big(\overline{0}(x),\overline{1}(y)\big)\to R_2\big(x,\overline{0}\big(\overline{1}(y)\big)\big)\Big)$ we see that $\mathfrak{M}\models R_2\big(0,\overline{0}(\overline{1}(\overline{1}(\overline{1}(0))))\big)$. Finally, by the fact that $\mathfrak{M}\models\forall x\forall y\Big(R_2\big(x,\overline{0}(y)\big)\to R_0\big(\overline{0}(x),y\big)\Big)$, we obtain our desired result that $\mathfrak{M}\models R_0\big(\overline{0}(0),\overline{1}(\overline{1}(\overline{1}(0)))\big)$, which of course means that $\mathfrak{M}\models\exists x\exists y R_0(x,y)$. □

By contrast, if we have a machine that does not halt on some input, it is easy to find a countermodel showing the formula is not valid. Let us return to the machine M on input 0:

	0	1
1	0R1	1R0

In this case our crucial formula $\varphi_{M,0}$ looks as follows:

$$\begin{aligned}
&\Big(\forall x\forall y\Big(R_1\big(x,\overline{0}(y)\big) \rightarrow R_1\big(\overline{0}(x),y\big)\Big)\wedge\\
&\forall x\forall y\Big(R_1\big(x,\overline{1}(y)\big) \rightarrow R_0\big(\overline{1}(x),y\big)\Big)\wedge\\
&\forall x\big(R_1(0,x) \rightarrow R_1(\overline{0}(0),x)\big) \wedge \forall x\big(R_1(x,0) \rightarrow R_1(x,\overline{0}(0))\big)\wedge\\
&\forall x\big(R_1(1,x) \rightarrow R_1(\overline{1}(0),x)\big) \wedge \forall x\big(R_1(x,1) \rightarrow R_1(x,\overline{1}(0))\big)\wedge\\
&R_1(0,0)\Big) \quad\rightarrow\quad \exists x\exists y R_0(x,y)
\end{aligned}$$

Fact 6.5.2. This formula $\varphi_{M,0}$ is not valid.

Proof. The intended model $\mathfrak{M}_{M,0}$ described above would be a countermodel, but there are also even simpler countermodels. For instance, consider a model $\mathfrak{M} = (\{a,b\},\mathcal{I})$ with two elements, a and b, and for which $\mathcal{I}$ is defined:

- $\mathcal{I}(0) = a$, $\mathcal{I}(1) = b$.
- $\mathcal{I}(\overline{0})$ and $\mathcal{I}(\overline{1})$ are both the identity function.
- $\mathcal{I}(R_1) = \{\langle a,a\rangle\}$, while $\mathcal{I}(R_0) = \varnothing$.

It is then easy to show that $\mathfrak{M}$ makes all conjuncts in the antecedent true, but $\exists x\exists y R_0(x,y)$ false. □

Exercise 6.5.1. Note that this machine does halt on input 1. Explain why $\varphi_{M,1}$ is valid.

6.5.3 Establishing Undecidability

What we have shown is that, for any machine M and binary string z, we can obtain our formula $\varphi_{M,z}$ in a straightforward, mechanical manner. Together with the claim that this formula is valid iff M halts on z (Proposition 6.5.1), this established the following result:

Theorem 6.5.2 (Church/Turing)**.** The validity problem for First Order Logic is undecidable.

To repeat, the reason is that if we could solve the validity problem we would be able to solve the Halting Problem: give me any instances of the halting problem—a machine M and an input z—and I will present for you a formula $\varphi_{M,z}$ that is valid iff M does indeed halt on z. Because the Halting Problem cannot be algorithmically solved—it is an undecidable problem—the validity problem must too be undecidable. This result easily gives us the following corollary, since validity is simply entailment from an empty set of premises:

Corollary 6.5.1. The entailment problem for First Order Logic are both also undecidable.

Exercise 6.5.2. Explain why the satisfiability problem for First Order Logic is undecidable.

6.6 A First Order Proof System

To show an entailment $\Gamma \nvdash \varphi$ does not hold is relatively easy: all we have to do is find a single countermodel, a model that makes the premises in Γ true but the conclusion φ false. However, to show entailment does hold is harder: we somehow have to argue that *any* model of the premises will be a model of the conclusion. Ideally we would have some way of deriving conclusions that follow from premises in a more direct way. This is exactly what a proof system is for. In Boolean logic we used our proof system to derive Boolean formulas that *semantically followed from* other formulas. That is, instead of having to show every model of the premises was a model of the conclusion—every truth assignment making all the premises true made the conclusion true—we could just find a proof. We will do the same thing now in the first order case.

Our first order proof system builds on the proof system we had for Boolean Logic. Indeed, we will include the introduction and elimination rules for all Boolean connectives that we had before. But now we also need to include three new sets of introduction/elimination rules: for the equality symbol $=$, the universal quantifier $\forall$, and the existential quantifier $\exists$. Before presenting these, a note on notation: In the following where we write $\varphi(x)$, this should be understood as some formula with x occurring (0 or more times) freely in φ. If we then write $\varphi(c)$, this stands for the same formula with the constant c replacing some specific free occurrence of x in φ. So, for example, if $\varphi(x)$ were the formula $\exists y\big(R(x,y) \to P(x)\big)$, then by $\varphi(c)$ we would mean $\exists y\big(R(c,y) \to P(c)\big)$.

The introduction rule for $=$ is extremely simple:

$\vdots$	$\vdots$	
n	$t = t$	$=$I

That is, we can simply assert that any term t is equal to itself, with no reference to any earlier line. The elimination rule is only slightly more complicated:

$\vdots$	$\vdots$	
i	$t_1 = t_2$	
$\vdots$	$\vdots$	
j	$\varphi(t_1)$	
$\vdots$	$\vdots$	
k	$\varphi(t_2)$	$=$E, i, j

$\vdots$	$\vdots$	
i	$t_1 = t_2$	
$\vdots$	$\vdots$	
j	$\varphi(t_2)$	
$\vdots$	$\vdots$	
k	$\varphi(t_1)$	$=$E, i, j

This rule allows us to use an equality statement by replacing one (occurrence of a) term with another term we know to be equal. It is sometimes called the rule of "replacement of equivalents." With these two simple rules we can already prove all the usual facts about equality. For instance, we should be able to show that it is symmetric and transitive. Here is a very easy proof that (for any

terms t and s) we have $t = s \vdash s = t$ (where again $\vdash$ means derivability).

1	$t = s$	
2	$t = t$	=I
3	$s = t$	=E, 1, 2

Our assumption is $t = s$, and in one line we can derive the statement $t = t$ by =-introduction. Now where our formula $\varphi(t)$ is $t = t$, by one application of =-elimination we can derive $\varphi(s)$, which in this case is $s = t$: we have replaced the first occurrence of t with s. We can also derive transitivity:

1	$t = s$	
2	$s = r$	
3	$t = r$	=E, 1, 2

This is just by replacing s in $\varphi(s) = s = r$ with the term t, to obtain $\varphi(t) = t = r$.

6.6.1 The Universal Quantifier

Let us now look at the rules for the quantifiers. The elimination rule for $\forall$ is relatively straightforward. It essentially codifies in a rule something that should be obvious: if $\forall x \varphi(x)$ holds, then $\varphi(t)$ ought to hold for any term t, as long as t does not contain any variables that would become bound in φ. (In particular this means that we are safe as long as t is built up only from function symbols and constants). We are allowed to replace uniformly every free occurrence of x with the term t:

$\vdots$	$\vdots$	
i	$\forall x \varphi(x)$	
$\vdots$	$\vdots$	
j	$\varphi(t)$	$\forall$E, i

Exercise 6.6.1. Show that if we allowed replacing x with a variable that would be bound in φ, we would be able to derive from no premises a formula that is not valid.

The introduction rule is a bit more complicated. It involves a more intricate kind of *conditional reasoning* of the sort we used in the rule for $\rightarrow$-introduction. Suppose we take some new constant symbol c that does not appear anywhere in the premises or conclusion (or elsewhere in the proof). That is, c is a totally fresh constant symbol. Suppose moreover that we can show $\varphi(c)$ holds: by assuming nothing whatsoever about c, we are nonetheless able to show that φ follows from earlier premises, when variable x is replaced with c. Then, because c was arbitrary, we seem to have shown that φ holds of anything, which should allow us to conclude $\forall x \varphi(x)$.

To take a very simple example, suppose we take $\varphi(x) = P(x) \rightarrow P(x)$. We ought to be able to derive $\forall x \varphi(x) = \forall x\big(P(x) \rightarrow P(x)\big)$. Our method for doing this is to take some fresh constant c, and

show using all our other rules that we can derive $P(c) \to P(c)$. Having done that we will be able to conclude $\forall x\big(P(x) \to P(x)\big)$. Thus, in a rule our method is codified as follows:

⋮	⋮	
i	$\boxed{c}$	
⋮	⋮	
j	$\varphi(c)$	
k	$\forall x \varphi(x)$	$\forall$I, i–j

That is, we introduce a new subproof, but instead of putting a new premise on the first line, we introduce a new, fresh constant symbol c (in a box, to make clear that it is just a fresh constant symbol). Then we show that we can derive φ with c in place of x. If we can do that, we are allowed to conclude $\forall x \varphi$. Here is how this would work on the simple example from the previous paragraph:

1	$\boxed{c}$	
2	$P(c)$	
3	$P(c)$	R
4	$P(c) \to P(c)$	$\to$I, 2–3
5	$\forall x\big(P(x) \to P(x)\big)$	$\forall$I, 1–4

How about deriving the counterintuitive validity $\forall x\big(\forall y R(y, y) \to R(x, x)\big)$ we saw earlier? We ought to be able to derive this from no premises. To do this we will need to use both the introduction and elimination rules for $\forall$, but otherwise it has almost exactly the same structure as the previous proof:

1	$\boxed{c}$	
2	$\forall y R(y, y)$	
3	$R(c, c)$	$\forall$E, 2
4	$\forall y R(y, y) \to R(c, c)$	$\to$I, 2–3
5	$\forall x\big(\forall y R(y, y) \to R(x, x)\big)$	$\forall$I, 1–4

For another example of a derivation using the elimination rule for $\forall$, consider the following entailment we observed earlier:

$$\{\forall x\big(P(x) \to Q(x)\big), \neg Q(c), \big(P(c) \vee P(d)\big)\} \nvdash P(d)$$

We would like to be able to show that there is also a derivation in this case:

$$\{\forall x\big(P(x) \to Q(x)\big), \neg Q(c), \big(P(c) \vee P(d)\big)\} \vdash P(d)$$

In fact, this is relatively easy to show, again using our earlier work on Boolean logic.

1	$\forall x\big(P(x) \to Q(x)\big)$	
2	$\neg Q(c)$	
3	$P(c) \vee P(d)$	
4	$P(c) \to Q(c)$	$\forall$E, 1
5	$\quad P(c)$	
6	$\quad Q(c)$	$\to$E, 4, 5
7	$\quad \bot$	$\bot$I, 2, 6
8	$\neg P(c)$	$\neg$I, 5–7
9	$\quad P(c)$	
10	$\quad \bot$	$\bot$I, 8, 9
11	$\quad P(d)$	$\bot$E, 10
12	$\quad P(d)$	
13	$\quad P(d)$	R
14	$P(d)$	$\vee$E, 3, 9–11, 12–13

After line 4, this whole proof is exactly like our earlier proofs in Boolean Logic!

How about the following entailment?

$$\forall x\big(P(x) \to Q(x)\big) \nvdash \forall x P(x) \to \forall x Q(x)$$

Can we show that the second formula is actually derivable from the first?

1	$\forall x\big(P(x) \to Q(x)\big)$	
2	$\quad \forall x P x$	
3	$\quad\quad \boxed{c}$	
4	$\quad\quad P(c) \to Q(c)$	$\forall$E, 1
5	$\quad\quad P(c)$	$\forall$E, 2
6	$\quad\quad Q(c)$	$\to$E, 4, 5
7	$\quad \forall x Q(x)$	$\forall$I, 3–6
8	$\forall x P(x) \to \forall x Q(x)$	$\to$I, 2–7

In the next example we go back to prove something that you in fact proved in Ex. 2.2.1, that a relation which is symmetric and transitive is also Euclidean! A forewarning is that this proof is quite involved. However, it should correspond roughly to the sequence of steps you go through in

giving an informal proof of the same fact.

1	$\forall x \forall y \big(R(x,y) \to R(y,x)\big)$	
2	$\forall x \forall y \forall z \Big(\big(R(x,y) \wedge R(y,z)\big) \to R(x,z)\Big)$	
3	$\boxed{c}$	
4	$\boxed{d}$	
5	$\boxed{e}$	
6	$R(c,d) \wedge R(c,e)$	
7	$R(c,e)$	$\wedge$E, 6
8	$R(c,d)$	$\wedge$E, 6
9	$\forall y \big(R(c,y) \to R(y,c)\big)$	$\forall$E, 1
10	$R(c,d) \to R(d,c)$	$\forall$E, 9
11	$R(d,c)$	$\to$E, 10, 8
12	$\forall y \forall z \Big(\big(R(d,y) \wedge R(y,z)\big) \to R(d,z)\Big)$	$\forall$E, 2
13	$\forall z \Big(\big(R(d,c) \wedge R(c,z)\big) \to R(d,z)\Big)$	$\forall$E, 12
14	$\big(R(d,c) \wedge R(c,e)\big) \to R(d,e)$	$\forall$E, 13
15	$R(d,c) \wedge R(c,e)$	$\wedge$I, 11, 7
16	$R(d,e)$	$\to$E, 14, 15
17	$\big(R(c,d) \wedge R(c,e)\big) \to R(d,e)$	$\to$I, 6–16
18	$\forall z \Big(\big(R(c,d) \wedge R(c,z)\big) \to R(d,z)\Big)$	$\forall$I, 5–17
19	$\forall y \forall z \Big(\big(R(c,y) \wedge R(c,z)\big) \to R(y,z)\Big)$	$\forall$I, 4–18
20	$\forall x \forall y \forall z \Big(\big(R(x,y) \wedge R(x,z)\big) \to R(y,z)\Big)$	$\forall$I, 3–19

The general strategy is to introduce new constants for *arbitrary* elements of the domain, c, d, and e, and argue that if $R(c,d)$ and $R(c,e)$ both hold, then from the fact that R is transitive and symmetric, we are guaranteed that $R(d,e)$ will also hold, which shows that R is indeed Euclidean.

How about a proof involving equality? Here is a simple example:

1	$\forall x \big(R(x,c) \to x = d\big)$	
2	$R(e,c)$	
3	$P(e)$	
4	$R(e,c) \to e = d$	$\forall$E, 1
5	$e = d$	$\to$E, 4, 2
6	$P(d)$	$=$E, 5, 3

For a slightly more complex example, we ought to be able to show the following:

$$\{\forall x Q(x, f(x), f(f(x))), \forall y\big(P(y) \to f(y) = y\big), \forall x P(x)\} \vdash \forall x Q(x, x, x)$$

The first premise says that Q holds of anything together with f applied to that thing, and f applied to the result of applying f. The second premise says that anything satisfying P is equal to f applied to it. The third says that everything satisfied P. It therefore follows that Q holds of anything with itself also in the second and third arguments.

1	$\forall x Q(x, f(x), f(f(x)))$	
2	$\forall y\big(P(y) \to f(y) = y\big)$	
3	$\forall x P(x)$	
4	$\boxed{c}$	
5	$P(c)$	$\forall$E, 3
6	$P(c) \to f(c) = c$	$\forall$E, 2
7	$f(c) = c$	$\to$E, 6, 5
8	$Q(c, f(c), f(f(c)))$	$\forall$E, 1
9	$Q(c, c, f(c))$	$=$E, 7, 8
10	$Q(c, c, c)$	$=$E, 7, 9
11	$\forall x Q(x, x, x)$	$\forall$I, 4–10

Note that from step 8 to 9, using elimination of $=$ we replace each occurrence of $f(c)$ with its provable equivalent c. But this still leaves us with a term $f(c)$ in the third argument (because we started out with $f(f(c))$ in that argument). So we have to apply $=$E once more to obtain c itself in the third argument.

Exercise 6.6.2. Give derivations showing each of the following:[2]

1. $\{\forall x\big(P(x) \to Q(x)\big), \forall x\big(Q(x) \to R(x)\big)\} \vdash \forall x\big(P(x) \to R(x)\big)$
2. $\vdash \big(\forall x P(x) \wedge \forall x Q(x)\big) \to \forall x\big(P(x) \wedge Q(x)\big)$
3. $\{\forall x\big(P(x) \to \neg Q(x)\big), \forall x\big(R(x) \to Q(x)\big)\} \vdash \forall x\big(P(x) \to \neg R(x)\big)$
4. $\{\forall x \forall y\big(R(x, y) \to R(y, x)\big), \forall x R(x, f(x)), \forall x\big(R(f(x), x) \to Q(x)\big)\} \vdash \forall x Q(x)$
5. $\vdash \forall x \forall y\big(x = y \to f(x) = f(y)\big)$

Exercise 6.6.3. Give a first order proof of the claim that any relation which is Euclidean and reflexive is also symmetric.

[2]Note that this first derivation is of perhaps the most famous inference pattern in logic. It is essentially a formalization of the classic syllogism of Barbara, which featured prominently in Aristotle's logic.

6.6.2 The Existential Quantifier

Since we know $\exists x\varphi$ is logically equivalent to $\neg\forall x\neg\varphi$, we could in principle do without rules for the existential quantifier. We do not strictly need $\exists$ for expressing new logical statements, since we can do everything with $\forall$. However, as in the case of additional Boolean connectives, it can be enlightening to see what the rules would look like anyway. As one might expect, the rules are "dual" to those of the universal quantifier. The existential introduction rule is quite simple. If t is a term with no variables (made up from constant and function symbols only), then we can replace each occurrence of t with x and existentially quantify the formula:

$$\begin{array}{l|ll} \vdots & \vdots & \\ i & \varphi(t) & \\ \vdots & \vdots & \\ j & \exists x\varphi(x) & \exists\text{I}, i \end{array}$$

It simply allows us to conclude from $\varphi(t)$ holding of some t that there is something satisfying φ. The elimination rule is more interesting. In order to use an existential claim, we might proceed as follows. Suppose we know $\exists xP(x)$, say. If we can show something follows from this by assuming that P holds of some fresh constant, and moreover our conclusion does not involve c in any way, that will show that this conclusion follows *merely from the assumption that something satisfies* P. The rule captures this idea more generally:

$$\begin{array}{l|ll} \vdots & \vdots & \\ i & \exists x\varphi(x) & \\ \vdots & \vdots & \\ j & \quad \boxed{c}\ \varphi(c) & \\ \vdots & \quad \vdots & \\ k & \quad \psi & \\ l & \psi & \exists\text{E}, i, j\text{–}k \end{array}$$

On line j we introduce the fresh constant c while simultaneously adding the assumption that φ holds with c substituted in for x. Then showing that any ψ follows from this assumption is enough to give us ψ from the assumption of $\exists x\varphi(x)$ itself. Note that the restriction that ψ not contain c is critical, otherwise we would be able to derive things that did not follow.

Exercise 6.6.4. Give an example of an invalid formula that would be derivable (from no premises) if we did not require that c not occur anywhere in ψ.

Now that we have introduction and elimination rules for $\exists$ as well, we can actually derive the

equivalences above. For instance, we can show

$$\vdash \forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$$

as well as:

$$\vdash \exists x P(x) \leftrightarrow \neg\forall x \neg P(x)$$

For the first:

1	$\forall x P(x)$	
2	$\exists x \neg P(x)$	
3	$\boxed{c}\ \neg P(c)$	
4	$P(c)$	$\forall$E, 1
5	$\bot$	$\bot$I, 3, 4
6	$\bot$	$\exists$E, 3–5
7	$\neg\exists x \neg P(x)$	$\neg$I, 2–6
8	$\neg\exists x \neg P(x)$	
9	$\boxed{c}$	
10	$\neg P(c)$	
11	$\exists x \neg P(x)$	$\exists$I, 10
12	$\bot$	$\bot$I, 11, 8
13	$\neg\neg P(c)$	$\neg$I, 10–12
14	$P(c)$	$\neg$E, 13
15	$\forall x P(x)$	$\forall$I, 9–14
16	$\forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$	$\leftrightarrow$I, 1–7, 8–15

We leave the other equivalence as an exercise (see below). For another mixed quantifier example, consider the following formula.

$$\{\forall x\big(\forall y R(x,y) \rightarrow Q(x,c)\big)\} \vdash \forall x \exists y\big(\neg R(x,y) \vee Q(x,c)\big)$$

This one is rather difficult, and you should first convince yourself that it is indeed entailed (that is, if we replace $\vdash$ by $\models$ this does hold). The proof requires a subtle combination of quantificational and Boolean reasoning. One proof of this (the shortest I could find) looks as follows:

1	$\forall x\big(\forall y R(x,y) \to Q(x,c)\big)$	
2	$\boxed{d}$	
3	$\neg\exists y\big(\neg R(d,y) \vee Q(d,c)\big)$	
4	$\boxed{e}$	
5	$\neg R(d,e)$	
6	$\neg R(d,e) \vee Q(d,c)$	$\vee$I, 5
7	$\exists y\big(\neg R(d,y) \vee Q(d,c)\big)$	
8	$\bot$	$\bot$I, 3, 7
9	$\neg\neg R(d,e)$	$\neg$I, 5–8
10	$R(d,e)$	$\neg$E, 9
11	$\forall y R(d,y)$	$\forall$I, 4–10
12	$\forall y R(d,y) \to Q(d,c)$	$\forall$E, 1
13	$Q(d,c)$	$\to$E, 11, 12
14	$\neg R(d,c) \vee Q(d,c)$	$\vee$I, 13
15	$\exists y\big(\neg R(d,y) \vee Q(d,c)\big)$	$\exists$I, 14
16	$\bot$	$\bot$I, 3, 15
17	$\neg\neg\exists y\big(\neg R(d,y) \vee Q(d,c)\big)$	$\neg$I, 3–16
18	$\exists y\big(\neg R(d,y) \vee Q(d,c)\big)$	$\neg$E, 17
19	$\forall x\exists y\big(\neg R(x,y) \vee Q(x,c)\big)$	$\forall$I, 2–18

Note that we have had to introduce a different constant d in step 2, and e in step 4, because c already appears in one of the formulas. This is what we mean when we say the constant must be "fresh."

Exercise 6.6.5. Show the following are all derivable.

1. $\vdash \exists x P(x) \leftrightarrow \neg\forall x\neg P(x)$
2. $\{\exists y\forall x R(x,y)\} \vdash \forall x\exists y R(x,y)$
3. $\{\neg\forall x\big(P(x) \to Q(x)\big)\} \vdash \exists x\big(P(x) \wedge \neg Q(x)\big)$
4. $\vdash \forall x\exists y f(x) = y$
5. (Bonus) $\{\exists y\forall x\big(f(x) = y \leftrightarrow x = c\big)\} \vdash \forall x\big(f(x) = f(c) \to x = c\big)$

6.6.3 Some Remarks on Metatheory

As in Boolean Logic, after working with our proof system for a while, we can ask the question: is it sound and complete? That is, do the two notions $\vdash$ and $\Vdash$ coincide in this first order setting, given

the extra rules we have provided for $=$, $\forall$ and $\exists$? The answer is positive. We state without proof:

Theorem 6.6.1 (Completeness of First Order Logic)**.** Fix any first-order language $\mathcal{L}^{\forall\exists}$. Then for any set of $\mathcal{L}^{\forall\exists}$ formulas Γ and any other $\mathcal{L}^{\forall\exists}$ formula φ, we have:

$$\Gamma \Vdash \varphi \quad \text{iff} \quad \Gamma \vdash \varphi$$

This result is far from trivial. It was first proven by Kurt Gödel in his doctoral dissertation, and its proof is one of the main topics of the successor to this course, Phil 151. It is also important because it tells us something about how "tame" First Order Logic is.

We saw in §6.5 that the problem of determining whether $\Gamma \Vdash \varphi$ (or given Theorem 6.6.1, whether $\Gamma \vdash \varphi$) is in general undecidable: there is no algorithmic way of solving this problem (Theorem 6.5.2). However, Theorem 6.6.1 tells us that the set of valid formulas,

$$\{\varphi \in \mathcal{L}^{\forall\exists} \mid \Vdash \varphi\}$$

is nonetheless *semi*-decidable, meaning that it *is* possible to write a computer program that will print out the valid formulas one by one, in such a way that every validity will eventually be printed. How do we do this? Assuming we have only finitely many non-logical parameters,[3] we can simply print out all the formulas with proofs of length 1 (there are only finitely many!), then all the formulas of length 2, then of length 3, and so on. Because any formula that is valid is guaranteed to have a proof, by Theorem 6.6.1, and that proof will have some finite length, we will eventually get to it in our enumeration of formulas with proofs.[4] Not all sets of formulas are even semi-decidable, and we will see one example in the next section.

The undecidability of First Order Logic shows in a very precise sense it is more complex than Boolean or Modal Logic. What we gain in incredible expressive power we pay for in complexity of the basic logical questions we might want to be able to ask and answer. This tradeoff between expressiveness and complexity is a pervasive theme in mathematical logic.

6.7 First Order Arithmetic

First Order Logic has found fundamental applications across a number of fields. In addition to various central areas of philosophy—including metaphysics, epistemology, philosophy of language, and others—it has also played a critical role in AI and in theoretical computer science, in linguistics, and in cognitive psychology. However, perhaps its most impressive application is in mathematics itself. Indeed, formalization of mathematical ideas and results—requiring the ability to talk about relations and functions of arbitrary complexity—was the primary impetus that led to what we now call First Order Logic.

We focus here on a particularly perspicuous—and also historically quite important—application, namely to number theory, often called the "Queen of Mathematics." We already saw in the previous

[3] It is also possible if there are (countably) infinitely many.

[4] The similarity with the halting problem should be evident: it is possible to *enumerate* the codes of machines that halt on their own input, even though it is not possible to find a *decision procedure* for this problem.

section the language of number theory—let us call it $\mathcal{L}^A$. In $\mathcal{L}^A$ we have two constant symbols 0 and 1, and two binary function symbols, $+$ and $*$. As we saw in §6.2, we can already express some interesting mathematical statements with this simple language, e.g., the statement expressing the open mathematical problem of whether there are infinitely many "twin" pairs of prime numbers.

Given our understanding of first order models, we can say exactly what the *standard model* of the natural numbers is. It is a structure $\mathfrak{N} = (\mathbb{N}, \mathcal{I})$, where $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ is the usual set of non-negative integers, and $\mathcal{I}$ interprets our non-logical symbols in the expected way:

$0^{\mathcal{I}} = 0$

$1^{\mathcal{I}} = 1$

$+^{\mathcal{I}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, such that $+^{\mathcal{I}}(n, m) = n + m$, for any $n, m \in \mathbb{N}$.

$*^{\mathcal{I}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, such that $*^{\mathcal{I}}(n, m) = n \times m$, for any $n, m \in \mathbb{N}$.

So, among formulas of $\mathcal{L}^A$, we are really interested in those that are true in the standard model, that is, those $\varphi \in \mathcal{L}^{\mathcal{A}}$ such that:

$$\mathfrak{N} \models \varphi$$

Let us call this set $\mathcal{T}(\mathfrak{N})$—sometimes called the (first order) *theory* of the natural numbers. For instance, the statement that there are infinitely many prime numbers (recall §6.2) is in this set, but we are not sure whether the statement that there are infinitely many twin pairs of primes is in the set. Indeed, to determine whether $\varphi \in \mathcal{T}(\mathfrak{N})$ one will typically have to do some number theory!

One of the goals of formalization is to achieve the possibility of *automation*, and in particular the possibility of automating the search for *proofs*. It follows from Theorem 6.6.1 that the search for first order validities can be partly automated: we could simply search for proofs of a given formula φ, e.g., by going through all possible proofs of increasing lengths, and if φ is a first order validity we know we will eventually find a proof (though this may take a very long time). One might hope to do something similar to this for the specific set of formulas $\mathcal{T}(\mathfrak{N})$, the set of true *arithmetical formulas*.

In pursuing this goal we are led to the idea of an *axiomatic (first order) theory*. The idea is simple: in addition to the usual introduction and elimination rules for first order logic, we augment our proof system with some *arithmetical axioms*—formulas that we know are true and that we can use anywhere in a proof—and a special kind of proof rule that we know is sound with respect to the natural numbers—that is, which always leads from truths to truths—namely, a rule encoding the principle of *mathematical induction*.

As axioms, let us take the following:

1. $\forall x(\neg x + 1 = 0)$
2. $\forall x \forall y(x + 1 = y + 1 \to x = y)$
3. $\forall x(x + 0 = x)$
4. $\forall x \forall y\big(x + (y + 1) = (x + y) + 1\big)$
5. $\forall x(x * 0 = 0)$

6. $\forall x \forall y \big(x * (y + 1) = (x * y) + x \big)$

All of these should be self-evident facts about the natural numbers. The first just says that nothing added to 1 gives 0; the second says that if adding 1 to n and m gives you the same number, then $n = m$; the third and fourth inductively characterize addition; while the fifth and sixth inductively characterize multiplication. What it means to say that these are *axioms* is that, in a proof, we are free to use any of these formulas without additional justification. Using these axioms by themselves we cannot prove very much, but if we add to them not only the rules of First Order Logic, but also the *rule of induction*, we will obtain a very powerful system. The rule of induction is as follows:

$$
\begin{array}{l|l|l}
\vdots & \vdots & \\
i & \varphi(0) & \\
\vdots & \vdots & \\
j & \quad \boxed{n} \;\; \varphi(n) & \\
\vdots & \quad \vdots & \\
k & \quad \varphi(n+1) & \\
l & \forall x \varphi(x) & \text{IND, } i, \, j\text{–}k
\end{array}
$$

The rule of induction has much the same form as the universal introduction rule—indeed, it is also used to introduce a universal statement. However, it is very specific to the natural numbers. Suppose we have show that some formula φ holds when we replace all the free instances of x by the constant 0. Suppose moreover that, by introducing a new constant symbol n and assuming that φ of n, we can show that it also holds of $n + 1$. Then the induction rule allows us to conclude that φ holds of anything whatsoever. More informally, the induction rule says:

> "If some statement is true of 0, and it being true of (an arbitrary) n implies it is also true of $n + 1$, then that statement is true of all natural numbers."

The resulting system, consisting of these six axioms and the rule IND, in addition all the rules for First Order Logic, is usually called *Peano Arithmetic* (PA), named the 19th century mathematician Giuseppe Peano. If φ is derivable in PA, we will write

$$\vdash_{\mathsf{PA}} \varphi,$$

to distinguish it from ordinary derivability ($\vdash$). What do proofs in PA look like? Here is a simple example. We have the statement that $\forall x (x + 0 = x)$ as an axiom. But what about the closely

related statement $\forall x(0 + x = x)$? Here is a proof of this in PA:

1	$\forall x(x + 0 = x)$	Ax
2	$0 + 0 = 0$	$\forall$E, 1
3	$\boxed{n}$ $0 + n = n$	
4	$\forall x \forall y\big(x + (y + 1) = (x + y) + 1\big)$	Ax
5	$\forall y\big(0 + (y + 1) = (0 + y) + 1\big)$	$\forall$E, 4
6	$0 + (n + 1) = (0 + n) + 1$	$\forall$E, 5
7	$0 + (n + 1) = n + 1$	=E, 6, 3
8	$\forall x(0 + x = x)$	IND, 2, 3–7

On line 2 we showed that the statement holds of 0. Then on line 3 we assumed that it held of an arbitrary n showed that it follows from this assumption that it holds for $n + 1$. This allowed us to conclude that it holds of all x on line 6.

Exercise 6.7.1. Show that $\vdash_{\mathsf{PA}} \forall x(0 * x = 0)$.

Having proven this very simple fact, we will now be able to prove something ever slightly more interesting, namely the commutativity of addition!

$$\vdash_{\mathsf{PA}} \forall x \forall y(x + y = y + x)$$

To prove officially we would have to reprove the above fact about adding 0 to a number. Instead of repeating all of those steps we will now allow something that we did not allow before: using facts proven elsewhere with a simple justification that we already proved them! The proof will still be quite long, but it would be much longer if we did not allow this. When we use a fact that we have already proven, we will mark the justification for this as ✓. Here then is the proof:

1	$\boxed{c}$	
2	$\forall x(x + 0 = x)$	Ax
3	$\forall x(0 + x = x)$	✓
4	$0 + c = c$	$\forall$E, 3
5	$c + 0 = c$	$\forall$E, 2
6	$0 + c = c + 0$	$=$E, 4, 5
7	$\forall y(0 + y = y + 0)$	$\forall$I, 1–6
8	$\boxed{n}$ $\forall y(n + y = y + n)$	
9	$0 + (n + 1) = (n + 1) + 0$	$\forall$E, 7
10	$(n + 1) + 0 = (n + 1) + 0$	$=$I
11	$(n + 1) + 0 = 0 + (n + 1)$	$=$E, 10, 9
12	$\boxed{m}$ $(n + 1) + m = m + (n + 1)$	
13	$\forall x \forall y\big(x + (y + 1) = (x + y) + 1\big)$	Ax
14	$\forall y\big((n + 1) + (y + 1)\big) = \big((n + 1) + y\big) + 1$	$\forall$E, 13
15	$\big((n + 1) + (m + 1)\big) = \big((n + 1) + m\big) + 1$	$\forall$E, 14
16	$\big((n + 1) + (m + 1)\big) = \big(m + (n + 1)\big) + 1$	$=$E, 15, 12
17	$\forall y\big(m + (y + 1) = (m + y) + 1\big)$	$\forall$E, 13
18	$m + (n + 1) = (m + n) + 1$	$\forall$E, 17
19	$\big((n + 1) + (m + 1)\big) = \big((m + n) + 1\big) + 1$	$=$E, 16, 18
20	$n + m = m + n$	$\forall$E, 8
21	$\big((n + 1) + (m + 1)\big) = \big((n + m) + 1\big) + 1$	$=$E, 19, 20
22	$\forall y\big(n + (y + 1) = (n + y) + 1\big)$	$\forall$E, 13
23	$n + (m + 1) = (n + m) + 1$	$\forall$E, 22
24	$\big((n + 1) + (m + 1)\big) = \big(n + (m + 1)\big) + 1$	$=$E, 21, 23
25	$n + (m + 1) = (m + 1) + n$	$\forall$E, 8
26	$\big((n + 1) + (m + 1)\big) = \big((m + 1) + n\big) + 1$	$=$E, 24, 25
27	$\forall y\big((m + 1) + (y + 1)\big) = \big((m + 1) + y\big) + 1$	$\forall$E, 13
28	$\big((m + 1) + (n + 1)\big) = \big((m + 1) + n\big) + 1$	$\forall$E, 27
29	$\big((n + 1) + (m + 1)\big) = \big((m + 1) + (n + 1)\big)$	$=$E, 26, 28
30	$\forall y\big((n + 1) + y = y + (n + 1)\big)$	IND, 11, 12–29
31	$\forall x \forall y\big(x + y = y + x\big)$	IND, 7, 8–30

Exercise 6.7.2. Show that $\vdash_{\mathsf{PA}} \forall x \forall y (x * y = y * x)$. You may use facts already proven (in the text or in your own exercises), with justification ✓ (as in line 3 in the proof above).

We could proceed in proving more and more basic arithmetical facts in this system, amassing an increasingly large arsenal of statements to be used in derivations of more and more interesting truths of arithmetic. In fact, to this day no one has found an example of a "natural" statement about natural numbers that is true but *cannot* be proven in PA. It is a surprisingly robust system.

We might, however, wonder whether PA really does capture everything true about the natural numbers. That is, do we have the following kind of strong *completeness* of PA?

$$\vdash_{\mathsf{PA}} \varphi \qquad \overset{?}{\Leftrightarrow} \qquad \mathfrak{N} \models \varphi$$

This would be of much interest: it would show that there is a single system in which all the truths about numbers could be derived. In one of the most important events in the history of logic, Kurt Gödel showed that PA does not possess this kind of completeness, and more importantly, he proved something stronger:

Theorem 6.7.1 (Gödel's First Incompleteness Theorem)**.** There is no axiomatic system extending PA that is both sound (from which you can prove only arithmetical truths) and complete.

This shows, not only does PA fail to be complete, but there is no way of extending it—e.g., by adding new axioms—to get a complete axiomatization, unless you are able to prove things that are not in fact true! Gödel's Theorem has been strengthened and generalized in a number of ways, but the main takeaway has been of central importance in the foundations of mathematics: it shows that there can be no complete first order axiomatization of number theory. Any first order system of axioms and rules will necessarily miss some truths about the numbers!

Exercise 6.7.3. In §6.6.3 we cited Gödel's earlier result, Theorem 6.6.1, which said that the rules of First Order Logic were complete. Meanwhile, we just said that Gödel also proved an *In*completeness theorem. Explain in your own words what each of these theorems say and why there can be both a Completeness Theorem and an Incompleteness Theorem in First Order Logic.