Formal Foundations of Linguistic Theory

Carl Pollard and Scott Martin

October 9, 2015

Contents

Introduction

Even though linguistics departments are, by tradition, usually located in colleges of humanities, linguistics itself aspires to be—and at its best, manages to be—a science. That is, linguists aim to do much the same thing that scientists in general (such as physicists, geologists, biologists, and chemists) do: to make observations of certain kinds of natural phenomena, and then state **empirical hypotheses** about them. The only difference is that the phenomena linguists study have to do not with swinging pendulums, tectonic plates, zebra mussels, or hydrocarbons, but with human language: how it sounds, what it means, how it varies across space and time, how it is learned, used, and understood.

Roughly speaking, an empirical hypothesis is just a well-informed and careful guess about what certain kinds of events will be like, based on past observations of events of that kind. To put it a bit more precisely, an empirical hypothesis is a general statement about a class of phenomena that has the following properties:

1. It is *clear and unambiguous*, that is, there is no question what it asserts (how things would have to be in order for it to be true).

2. It is *general*, in the sense that even though it is based only on a finite number of observations, it makes predictions about how other phenomena of the same kind will unfold.

3. There is a way to tell whether or not a given observation of the kind of phenomenon in question is consistent with it, so that if the hypothesis is wrong, there is some hope of finding out that it is wrong.

This third property of empirical hypotheses is called **falsifiability**. Especially valued are empirical hypotheses with the additional property of being *illuminating*, in the sense of being sufficiently simple and comprehensible to help us grasp some of the hidden orderliness or systematicity in seemingly random or chaotic phenomena.

Scientific theories usually do not directly describe the natural phenomena under investigation, but rather a mathematical idealization of them that abstracts away from various complicating factors. For example, a theory about how the earth, the sun, and the moon move under mutual gravitation might ignore such complications as the sizes of the three bodies, friction arising from the presence of interstellar dust, the gravitational force exerted by other planets and stars, or relativistic effects that become significant only as the velocities of the bodies in question approach the speed of light. In the mathematical idealization, the time might be represented by a real number; the mass of each of the three bodies by a positive real number; its location in space (or more precisely the location of its center of gravity) at a particular time by three real numbers (the x, y, and z coordinates relative to a coordinate system); its velocity at a particular time by three more real numbers; the state of the three-body system at a given time (real number) t by the 18 real numbers that specify the locations and velocities of the three bodies at time t; and the evolution of the system over time by 18 functions that give the value of each of these 18 parameters at each time t. And the theory itself is a mathematical specification of which evolutions ('paths' through 18-dimensional Euclidean space) are possible. Armed with such a theory, we can predict, given the state of the system at a given time t_0, what state it will be in at any future time t_1.

A **linguistic theory** is just a set of empirical hypotheses about a class of natural language phenomena. Linguists often refer to the process of formulating empirical hypotheses about human languages as 'capturing linguistic generalizations.' This is just a fancy name for linguistic theorizing, and the purpose of this book is to introduce some techniques for doing just that. Linguistic theories make predictions not about celestial bodies, but rather about natural languages, for example:

- How their words can sound
- How their words can be combined into phrases
- What meanings they can express
- Which natural language arguments are judged valid
- How the meanings of sentences can be related to the meanings of the words they contain

The mathematical idealizations used in formulating linguistic theories are often called **representations** or **models** of the phenomena in question. In

this book the first of these terms will be preferred, to avoid confusion with a different, technical, use of the term *model* (in the sense of an interpretation of a logical theory) to be introduced in later chapters. For example, phrases (roughly speaking, multi-word expressions, including sentences) are often represented as (mathematical) *trees*; phonemes (roughly, minimal units of linguistic sound) as (mathematical) *graphs* of a certain kind; the sequences of sounds that make up (the phonology of) words as (mathematical) *strings* of (representations of) phonemes; and linguistic meanings as (mathematical) *functions* of various kinds. (Note that it is typical for technical mathematical terms, such as *tree*, *string*, and *function* to have other, nonmathematical meanings!)

The techniques introduced in this book are drawn from areas of mathematics (such as set theory, logic, algebra, and formal language theory) that are usually described as **discrete**, as opposed to **continuous** (such as calculus, differential equations, Fourier analysis, or probability). The natural numbers are discrete; the real numbers are continuous. The subdisciplines of linguistics that most readily lend themselves to analysis by discrete methods include (but are not limited to) the following:

morphology (how words are built up from their meaningful subparts)

syntax (how words combine into successively larger phrases, including sentences)

semantics (how linguistic expressions manage to refer to things in the world and express propositions about them, and how it is that some propositions follow from, or are **entailed** by, other propositions).

There are also parts of **phonology** (how human languages structure spoken sounds) and **computational linguistics** (the analysis and manipulation of human language using computational concepts or computer programs) that yield to such methods. But other linguistic disciplines, such as phonetics, psycholinguistics, sociolinguistics, and historical linguistics in general call for continuous methods. Interestingly, many of the discrete mathematical techniques that come into play in the analysis of human language are the same ones used in analyzing the artificial languages employed in logic and computer science.

This is an applied mathematics book, not a linguistics book, and so the emphasis is primarily on the mathematical concepts and techniques themselves, not on the phenomena to which they are applied. In fact, most of these are of inherent interest independent of the linguistic applications,

and it is entirely possible to master them without knowing or caring about linguistics at all! But the book is written primarily with the needs of linguistics students in mind, and many techniques discussed are accompanied by illustrative linguistic applications.

Part I

Fundamentals

Chapter 1

Sets

In order to have a clear understanding of the different kinds of mathematical entities that are useful for structuring linguistic representations, we will start out with an overview of **set theory**. Sets are basic mathematical entities whose existence is taken for granted by most mathematicians, and set theory begins with certain assumptions about them. Set theory is the workspace that most mathematicians work in, but more importantly for us, it is where the idealized representation of natural phenomena by linguists and other scientists is carried out. That is, sets are used to construct the representations of natural language phenomena that linguistic theories talk about.

As discussed in the introduction, the kinds of mathematical entities that have proven to be useful for representing such things (words, phrases, sentences, their meanings, valid arguments, etc.) are not real numbers or real-valued functions, but rather discrete (in the sense of being non-continuous) things such as natural numbers, strings, trees, algebras, formal languages, and proof systems. It turns out that all these kinds of linguistic representations are themselves sets.

1.1 Sets and Membership

We assume that there exist things which we call **sets**, and that there is a relationship, called **membership**, which, for any two sets, either does or does not hold of them. That is, if A is a set and B is a set, then either A **is a member of** B (written $A \in B$) or A is not a member of B (written $A \notin B$). There are many ways to say this. The members of a set are also called its **elements**, and instead of saying A is a member of B, we often say

it **belongs to** B, or is **in** B, or is **contained in** B. Intuitively, sets can be thought of as something like collections, where the members are the things collected, or as invisible baskets, with the members being the things in the baskets. But set theory will never tell us what sets are; they are basic and cannot be reduced to, or explained in terms of, more basic things that are not sets. That is, they are the **unanalyzed primitives** of set theory.

We will make certain assumptions about how membership works based on these intuitions, and then try to ascertain what follows from them. These assumptions themselves, together with the facts that follow from them, constitute set theory. To be slightly more precise, they are *a* set theory, since some assumptions about how sets should work are controversial. In this chapter, we will make some of the most generally accepted of these assumptions explicit and consider some of their consequences. (In Chapter 8, we will also consider some of the more controversial assumptions about how sets work.)

For the time being, we will state our assumptions about sets in English, and conduct our reasoning about what follows from these assumptions using intuitively valid English arguments called **informal proofs**. In Part II, we will see that it is possible to **formalize** the assumptions of set theory with the help of specialized symbolic systems (formal logics, such as predicate logic). In that case the formalized counterparts of the assumptions are called **axioms**; the additional formulas that follow from them are called **theorems**; and the formalized counterparts of the English arguments we make to justify these theorems are called **formal proofs**.

In fact, informal (but precise) natural language reasoning is the norm among mathematicians and natural scientists. Usually they don't bother to formalize proofs unless they are studying proofs as mathematical objects in their own right. We will have occasion to do just that in Part **??**, for the (perhaps surprising) reason that linguistic expressions and their meanings can themselves be thought of as proofs in certain kinds of logical systems.

In ascertaining what follows from the assumptions we will make about sets and membership, the reasoning we use will be pretty much the same kind of reasoning we use when we draw conclusions from assumptions about ordinary things, e.g. kitchen appliances, furniture, people, etc. (There are, however, some ways of arguing and ways of expressing arguments that are typical of mathematical discourse, which we will look at more closely in Chapter 2.) In practice, mathematics consists of more or less ordinary reasoning about not-so-ordinary things. The upshot, seemingly paradoxical, is that so-called formal linguistics is mostly done within *informal* set theory. The resolution

of the apparent paradox is that even informal set theory is more precise and explicit than linguistic analysis that uses no set theory at all.

Now we're ready to start introducing our basic assumptions about sets, and considering some of their consequences.

1.2 Basic Assumptions about Sets

We have already assumed that there *are* sets, and that if A and B are sets, then either $A \in B$ or $A \notin B$. But to be able to *do* anything with sets, we need to make some assumptions about how they work. The assumptions we make in this chapter are the ones that are generally considered the most basic, intuitively plausible, and uncontroversial. Later we will add a few more (but not many more), including some that not all mathematicians are entirely comfortable with (see Chapter 8). We give each assumption a name, to make it easy to refer to.

Assumption 1 (Extensionality)**.** If A and B have the same members, then they are the same set (written $A = B$).

Note that in stating this assumption, we did not bother to mention that A and B are sets. That's because we have already established that we are now doing (informal) set theory, and in set theory, the *only* things being talked about are sets. Note also that we do not have to explicitly assume (though it is true) that if A and B do *not* have the same members, then they are *not* the same set (written $A \neq B$). That's because, if they were the same set, then *everything* about them, including what members they have, would be the same. This reasoning is no different than the kind of reasoning we would use to conclude (given that A and B are people), that if A and B do not have the same blood type, then they cannot be the same person: if they were the same person, everything about them—including their blood types—would be the same.

If every member of A is a member of B, we say that A is a **subset** of B, (or, alternatively, that A is **included in** B), written $A \subseteq B$. Note that if $A \subseteq B$, B might have members that are not in A. On the other hand, if both $A \subseteq B$ and $B \subseteq A$, then it follows from Extensionality that $A = B$. If $A \subseteq B$ but $A \neq B$ then we say A is a **proper** subset of B, written $A \subsetneq B$.[1]

Assumption 2 (Empty set)**.** There is a set with no members.

[1] We avoid the symbol $\subset$ since some authors use it in place of our $\subseteq$, and others in place of our $\subsetneq$.

Note that from this assumption together with Extensionality we can conclude that the there is *only* one set with no members. We call this set the **empty set**. The empty set is usually denoted by the symbol '$\emptyset$.' Starting in Chapter 6, we'll sometimes write it as '0' (the symbol for the number zero), because according to the most usual way of doing arithmetic within set theory, the number zero and the empty set are the same thing (in spite of what you may have been taught in other math classes!).

Now so far, we have no basis for concluding that there are any sets other than the empty set, not even sets with only one member. For example, we are not even able to make a valid argument that there is a set with $\emptyset$ as its only member. We remedy this situation by adding a few more assumptions, beginning with the following:

Assumption 3 (Pairing)**.** For any sets A and B, there is a set whose only members are A and B.

Note that, because of Extensionality again, there is *only* one set whose only members are A and B, which we write as $\{A, B\}$. Of course we could just as well have called this set $\{B, A\}$. More generally, we will notate any nonempty finite set by listing its members, separated by commas, between curly brackets, in any order. (In Chapter 6, we'll get clear about what we mean when we say a set is 'finite,' but for now we'll just rely on intuition.) Notice that nothing rules out the possibility that A and B are the same set, so it follows from pairing that for any set A there is a set whose only member is A, namely $\{A, A\}$. Of course, once we realize this, then we might as well just call it $\{A\}$ rather than $\{A, A\}$: repetitions inside the curly brackets don't make any difference because for any given set, either A is a member of it or it isn't—it doesn't make any sense to talk about *how many times* one set is a member of another.

A set with only one member is called a **singleton**. A special case of singleton sets is the set $\{0\}$ whose only member is 0. This set is also called 1, because according to the usual way of doing arithmetic within set theory, it is the same as the number one. Going one step further, we can use Pairing again to form the set $\{0, 1\}$, also known as 2. There is a general pattern here, which we will explain in Chapter 6.

We can now form a new set by pairing two sets together. But what if we want to form a set by combining the *members* of two sets? The following assumption enables us to do so:

Assumption 4 (Union)**.** For any set A, there is a set whose members are those sets which are members of (at least) one of the members of A.

Once again, Extensionality ensures the uniqueness of such a set, which is called the **union** of A, written $\bigcup A$. As a special case, if $A = \{B, C\}$, then $\bigcup A$ is the set each of whose members is in either B or C (or both). This set is usually written $B \cup C$. Note that this is not the same thing as $\{B, C\}$, at least not for all possible values of B and C. Are there any values of B and C for which $B \cup C = \{B, C\}$? It turns out that the answer depends on what other assumptions we make about sets. The specific set of assumptions we will adopt will not enable us to answer this question one way or the other.

Exercise 1.1. Prove that for any sets a, b, and c, there is a set whose only members are a, b, and c. (*Note*: this way of wording the problem is *not* intended to imply that a, b, and c are necessarily distinct from each other.)

For any set A, the **successor** of A, written $s(A)$, is the set $A \cup \{A\}$. That is, $s(A)$ is the set with the same members as A, except that A itself is also a member of $s(A)$.[2] For example, 1 is the successor of 0, and 2 is the successor of 1.

Exercise 1.2. Prove that 1 is the successor of 0 and that 2 is the successor of 1.

Exercise 1.3. Prove that 0, 1, and 2 are distinct.

To enable the formation of the set of all subsets of some set, we add the following:

Assumption 5 (Powerset)**.** For any set A, there is a set whose members are the subsets of A.

Yet again, Extensionality guarantees the uniqueness of such a set. We call it the **powerset** of A, written $\wp(A)$. It's important to realize that $\wp(A)$ is never the same set as A (a fact which we will prove in Chapter 6). That's because the subsets of a set are not the same as the members of the set. For example, 0 is a subset of 0 (in fact, every set is a subset of itself), but obviously 0 is not a member of 0 (since 0 is the empty set).

Exercise 1.4. What is the powerset of 4?

[2] Nothing we have said rules out the possibility that $A \in A$, in which case $A = s(A)$. However, the most widely used set theory (called *Zermelo-Fraenkel* set theory) includes an assumption (called *Foundation*) which does rule out this possibility. We will not assume Foundation in this book. For more discussion of this point, see Chapter 8.

1.3 Russell's Paradox and Separation

Why do we need the powerset assumption? Why don't we just *define* $\wp(A)$ to be the set of all subsets of A? The answer is that the other assumptions we have made so far do not seem to enable us to conclude that there actually *is* such a set. More generally, whenever one says "the set of all sets such that . . . ," there is no guarantee that the assumptions one has made about sets enable one to conclude that there actually is a set meeting that description. That may seem counterintuitive, but, perhaps surprisingly, there is a knockdown argument that there is no such guarantee, which was discovered by the philosopher and mathematician Bertrand Russell.[3]

The argument runs as follows. Consider the description *the set of all sets which are not members of themselves*. Suppose for a moment there were such a set, called R. Then is R a member of R? Well, either it is or it isn't. In the first case, we see right away that R cannot be a member of R. And in the second case, we see right away that R must be a member of R. Either way, we arrive at a contradiction, and so our temporary assumption that there is a set whose members are the sets which are not members of themselves must have been false. This argument is called **Russell's Paradox**.

Russell's Paradox shows that, in general, we cannot assume that, for any set description, we can take for granted the existence of a set meeting that description. However, there is a more cautious assumption that proves to be extremely useful and which so far has not been shown to result in paradox.

Assumption 6 (Separation)**.** If A is a set and $P[x]$ is a condition on x (where x is a variable that ranges over sets), then there is a set, written $\{x \in A \mid P[x]\}$, whose members are all the x in A that satisfy $P[x]$.

Separation is so called because, intuitively, we are separating out from A some members that are special in some way, and collecting them together into a set. We call Separation an assumption, but to be more precise it is an assumption **schema**: for each condition $P[x]$, we get a different separation assumption. For the moment we remain deliberately vague about what we mean by a *condition on* x. (We'll clear this up in Part II when we formalize set theory using predicate logic.) For the moment, the easiest way to get an idea of what we mean by a condition on x is to look at some examples.

Suppose we have two sets A and B. Then by taking $P[x]$ to be the condition $x \in B$, Separation guarantees the existence of the set consisting of

[3] Russell made this argument in a famous letter written in 1902 to Gottlob Frege, another philosopher and mathematician, whose accomplishments include the invention of predicate calculus and of modern linguistic semantics.

those members of A which are also in B. This set is called the **intersection** of A and B, written $A \cap B$. A and B are said to **intersect** if $A \cap B$ is non-empty; otherwise they are said to be **disjoint**. A set is called **pairwise disjoint** if no two distinct members of it intersect.

Alternatively, by taking $P[x]$ to be the condition $x \notin B$, Separation guarantees the existence of the set consisting of those members of A which are *not* in B. This set, called the **complement of B relative to A**, or the **set difference of A and B**, is written $A \setminus B$.

A rather different application of Separation shows that there can be no set of all sets. For suppose there were; then applying Separation to it using the condition $x \notin x$, we would have the set of all sets which are not members of themselves. But as we already saw (Russell's Paradox), there can be no such set.

Exercise 1.5. Why did we assume Union instead of defining it as an application of Separation?

1.4 Ordered Pairs and Cartesian (Co-)Products

Sets do not embody any notion of order: $\{A, B\} = \{B, A\}$. But for linguistic applications, clearly we cannot escape from dealing with order! For example, we cannot describe the phonology of a word without specifying the order of the phonemes in it, nor can we fully describe a sentence without specifying the order of its words. One way we might imagine responding to this need is simply to *assume* that for any A and B, there is an **ordered pair** with A first and B second, written $\langle A, B\rangle$. But what properties should we assume that ordered pairs have? Perhaps surprisingly, it turns out that once we have gotten clear about how ordered pairs should work, the assumptions we have already made about sets enable us to conclude that sets with the desired properties already exist. So we do not need to make any further assumptions in order to have ordered pairs.

In fact, the crucial property of ordered pairs, from which their usefulness derives, is that they are *uniquely determined by their components*, in the sense that $\langle A, B\rangle = \langle C, D\rangle$ if and only if $A = C$ and $B = D$. Any way of defining the notion of ordered pair that results in their demonstrably having this property will suffice. The approach we will adopt here is the standard one, due originally to Kuratowski, which is to define the ordered pair of A and B as the set $\{\{A\}, \{A, B\}\}$. (Note that we can form this set by successive invocations of the Pairing assumption.) A and B are called, respectively, the **first** and **second component** of $\langle A, B\rangle$. Notice that an ordered pair has

either one or two members. In the first case, which arises when $A = B$, the ordered pair is just $\{\{A\}\}$, and both components are A. In the second case, the ordered pair has two members, one with one member and one with two members. In that case, the first component of the pair is the one that belongs to the set with one member, and the second component is the member of the two-member set which is not the member of the one-member set.

Exercise 1.6. Letting $3 = s(2)$, how many members does $\bigcup \langle 2, 3 \rangle$ have? What are they? (*Note*: you will have to use the definition of ordered pair. You are not required to prove anything here.)

Exercise 1.7. Prove that for any sets a, b, c, and d, if $\langle a, b \rangle = \langle c, d \rangle$, then $a = c$ and $b = d$. (*Hint*: notice that either $a = b$ or not, so you can split the proof into two cases.)

Given two sets A and B, it is also useful to have the notion of the **cartesian product** of A and B, written $A \times B$, which is the set of all ordered pairs $\langle a, b \rangle$ such that $a \in A$ and $b \in B$. As it turns out, we do not have to *assume* that cartesian products exist, because their existence follows from Separation. A and B are called the **factors** of $A \times B$.

Exercise 1.8. Prove the existence of cartesian products. In other words, prove that for any sets A and B, there is a unique set whose members are those ordered pairs $\langle c, d \rangle$ such that $c \in A$ and $d \in B$. (*Hint*: use Separation. The hard part is deciding which set to start with from which the ordered pairs in question are to be separated out.)

Exercise 1.9. Prove that for any set A, $A \times \emptyset = \emptyset$.

Having defined ordered pairs, we can now proceed to define an **ordered triple** to be an ordered pair whose first component is an ordered pair:

$$\langle a, b, c \rangle =_{\text{def}} \langle \langle a, b \rangle , c \rangle$$

and correspondingly the threefold cartesian product:

$$A \times B \times C =_{\text{def}} (A \times B) \times C$$

These definitions can be extended to quadruples, quintuples, etc. in the obvious way. Special cases of cartesian products, called cartesian **powers**, are ones where the factors are all the same set A. These are notated with

parenthesized 'exponents' (superscripts), e.g.

$$\begin{aligned} A^{(2)} &= A \times A \\ A^{(3)} &= A \times A \times A \\ &\vdots \end{aligned}$$

Additionally, we define $A^{(1)}$ to be A, and we define $A^{(0)}$ to be 1. This last definition is less mysterious than it appears to be, but we will be in a better position to explain the motivation for it a little later. (It is actually closely related to the reason that $n^0 = 1$ in arithmetic, but for some readers, that may seem equally mysterious. We will revisit this point in Chapter 6.)

Less well known than cartesian product, but also important in some of our applications, is the notion of the **cartesian coproduct**, also called the **disjoint union**) of A and B, written $A + B$. This is defined as

$$(\{0\} \times A) \cup (\{1\} \times B)\,,$$

the set of all ordered pairs $\langle C, D\rangle$ such that either $C = 0$ and $D \in A$ or $C = 1$ and $D \in B$. A and B are called the **cofactors** of $A + B$.

Intuitively, $A+B$ is the union of two sets, 'copies' of A and B respectively, and these copies are disjoint, even if A and B are not. As with cartesian products, there is a straightforward extension to more than two cofactors. For the case of identical cofactors (called cartesian **copowers**), there does not seem to be a standard notation; here we write $A_{(n)}$, which, intuitively, is the union of n pairwise disjoint copies of A. So it should not come as much of a surprise that $A_{(1)}$ is defined to be A and $A_{(0)}$ is defined to be 0.

Chapter 2

Mathese

2.1 Introduction

Mathematicians (well, English-speaking ones, anyway) talk and write about things logical and mathematical (including set theory and anything they construct inside it) in a mixture of ordinary colloquial English and a special purpose dialect of English, which we will refer to as **Mathese**. Mathese is intended to avoid the ambiguity, vagueness, and imprecision of much ordinary colloquial English. It is a good idea to get into the habit of judiciously using Mathese when writing about formally rigorous linguistic theory for an audience with a reasonable degree of mathematical sophistication—e.g. when writing answers to the exercises in this book. (*Alert*: it is every bit as important *not* to write this way for a general linguistic audience!) Of course, unless you have an unusually strong mathematical background, it takes some time to get the hang of Mathese, so mastery may not be immediate. It's also okay to use ordinary English as long as the meaning is completely clear.

In its most basic form, all Mathese has is a few 'logicky' expressions and some basic predicates for talking about set membership and equality. Fortunately, it's permissible to add new predicates and names to the language as needed, as long as you take care to *define* them in terms of expressions that are already in the language, as will be explained below. (Without such abbreviations, Mathese quickly becomes opaque to the point of sheer incomprehensibility.) There are also symbols for abbreviating expressions, which are mostly used in displayed calculations and inside of set descriptions; the abbreviations (especially the logicky ones) are usually *not* used in writing

Mathese prose (which is what you would use to write proofs for this book's exercises).[1]

2.2 'Logicky' expressions

2.2.1 Variables

These are upper- or lower-case Roman letters (usually italicized in typing), with or without numerical subscripts, used roughly as pronouns or as names of arbitrary sets, e.g. x, y, x_0, x_1, X, Y, etc.

2.2.2 Sentence-level expressions

And Mathese 'and' is abbreviated using the **conjunction** symbol $\wedge$. It is used mainly for combining sentences, as in:

S_1 and S_2. (*Abbreviated form*: $\mathrm{S}_1 \wedge \mathrm{S}_2$)

A sentence formed this way is called a **conjunctive** sentence. Here S_1 is called the **first conjunct** and S_2 is called the **second conjunct**. A conjunctive sentence is considered to be true if both conjuncts are true; otherwise it is false.

Or Mathese 'or' is abbreviated using the **disjunction** symbol $\vee$. Like 'and', it is used mainly for combining sentences, as in:

S_1 or S_2. (*Abbreviated form*: $\mathrm{S}_1 \vee \mathrm{S}_2$)

A sentence formed this way is called a **disjunctive** sentence. Here S_1 is called the **first disjunct** and S_2 is called the **second disjunct**. Mathese *or* is **inclusive** disjunction, so that a disjunctive sentence is true if *either or both* of the disjuncts are true, and it is false otherwise.

Implies Mathese 'implies' is abbreviated using one of the two **implication** symbols $\rightarrow$ or $\supset$. A synonym for 'implies' is 'if ... then ...'. It too is used for combining sentences, as in:

S_1 implies S_2. (*Abbreviated forms*: $\mathrm{S}_1 \rightarrow \mathrm{S}_2$ or $\mathrm{S}_1 \supset \mathrm{S}_2$)

[1] In Part II, we'll introduce some formal languages, called *first-order languages*, which consist *entirely* of such symbols. By then, you'll have a good intuitive feeling for what such symbols mean. If you've taken a basic course in predicate logic, you'll already be familiar with these.

A sentence formed this way is called a **conditional** or **implicative** sentence. Here S_1 is called the **antecedent** and S_2 is called the **consequent**. *Caution*: this does not mean quite exactly the same thing as *if* S_1 *then* S_2 in ordinary English. One difference is that a conditional Mathese sentence is considered to be true if the consequent is true, no matter whether the antecedent is true or false and even if the antecedent and the consequent seem to have nothing to do with each other, e.g. the statement

If there does not exist a set with no members, then $0 = 0$

is true. Another difference is that a conditional Mathese sentence is considered to be true if the antecedent is false, no matter whether the consequent is true or false, e.g.

If $0 \neq 0$ then $1 \neq 1$

is true!

If and only if Mathese 'if and only if,' usually written simply as 'iff,' is abbreviated using the **biimplication** symbol $\leftrightarrow$. It is used to combine sentences as in:

S_1 iff S_2. (*Abbreviated form*: $S_1 \leftrightarrow S_2$)

A sentence of this form is called a **biconditional**. S_1 iff S_2 can be thought of as shorthand for:

S_1 implies S_2, and S_2 implies S_1.

Consequently, a sentence of this form is considered to be true if either

1. Both S_1 and S_2 are true, or
2. Both S_1 and S_2 are false.

Otherwise, it is false.

It is not the case that Mathese 'it is not the case that' is abbreviated using one of the two **negation** symbols $\neg$ or $\sim$. It is placed before a sentence in order to negate it, as in:

It is not the case that S. (*Abbreviated forms*: $\neg$S or $\sim$ S)

A sentence of this form is called a **negative** sentence. Here S is called the **scope** of the negation. Unsurprisingly, a negative sentence is considered to be true if the scope is false, and false if the scope is true. For any sentence S, the sentence *it is not the case that* S is called the **negation** of S, or, equivalently, the **denial** of S.

Note that often, the effect of negation with *it is not the case that* can be achieved by ordinary English **verb negation**, which (simplifying slightly) involves replacing the finite verb (the one that agrees with the subject) V with 'does not V' if V is not an auxiliary verb (such as *has* or *is*), or negating V with a following *not* or *-n't* if it *is* an auxiliary. Thus, for example, these pairs of sentences are equivalent (express the same thing):

> It is not the case that 2 belongs to 1.
> 2 does not belong to 1.
>
> It is not the case that 1 is empty.
> 1 isn't empty.

But negation by *it is not the case that* and verb negation cannot be counted on to produce equivalent effects if the verb is in the scope of a quantifier (see the following section). For example, these are not equivalent:

> It is not the case that for every x, x belongs to x.
> For every x, x doesn't belong to x.

The first is clearly true (for example, 0 doesn't belong to 0), but the truth of the second cannot be determined on the basis of the assumptions in Chapter 1, and in fact different ways of adding further set-theoretic assumptions resolve the issue in different ways (see Chapter 8).

Note that for predicates with an abbreviatory symbol, such as *equals* ($=$) and *belongs to* ($\in$), the effect of verb negation is accomplished by a diagonal slash, e.g. $\neq$ 'is not equal to,' $\notin$ 'is not a member of.'

2.2.3 Quantificational expressions

For all Mathese 'for all,' abbreviated by the **universal quantifier** symbol $\forall$, forms a sentence by combining first with a variable and then with a sentence, as in:

For all x, S. (*Abbreviated form*: $\forall x$S)

Synonyms of 'for all' include 'for each,' 'for every,' and 'for any.' A sentence formed in this way is said to be **universally quantified**, or simply **universal**.

There exists ... such that Mathese 'there exists ... such that', abbreviated by the **existential quantifier** symbol $\exists$, forms a sentence by combining first with a variable and then with a sentence, as in:

There exists x such that S. (*Abbreviated form*: $\exists x$S)

Synonyms of 'there exists ... such that' include 'for some' and 'there is a(n) ... such that.' A sentence formed in this way is said to be **existentially quantified**, or simply **existential**.

There exists unique ... such that In Mathese, 'there exists unique ... such that,' abbreviated $\exists!$, combines first with a variable, then with a sentence, as in:

There exists unique x such that S.
(*Abbreviated form*: $\exists! x$S)

This is understood to be shorthand for

$$\exists x(\mathrm{S}[x] \wedge \forall y(\mathrm{S}[y] \to (y = x))) .$$

In a quantificational expression of the form $\forall xS$ or $\exists xS$, the variable x is said to be **bound** by the quantifier, and the sentence S is called the **scope** of the quantifier. Usually the bound variable also occurs in the scope; if it doesn't, then the quantification is said to be **vacuous**. If a sentence contains variables which are not bound by any quantifier, those variables are called **free**. A sentence is called **closed** if it has no free variables, and **open** otherwise. A sentence whose free variables are $x_0, \ldots, x_n$ is often called a **condition** on $x_0, \ldots, x_n$. The number of free variables in a condition is called its **arity**. Thus conditions might be **nullary** (no free variables, i.e. a closed sentence), **unary** (one free variable), **binary** (two free variables), **ternary** (three free variables), etc.

As long as we are using Mathese only to talk about set theory, we can assume that the variable used to form a quantified sentence ranges over all sets, that is, 'for all x' is implicitly understood as 'for all sets x.' Similarly, 'there exists x' is implicitly understood as 'there exists a set x.'

However, often we want to quantify not over *every* set, but just over the sets that satisfy some condition on x, $\mathrm{S}_1[x]$. Then we say, for example:

For every x with $\mathrm{S}_1[x]$, $\mathrm{S}_2[x]$.

This is understood to be shorthand for

For every x, $\mathrm{S}_1[x]$ implies $\mathrm{S}_2[x]$.
(*Abbreviated form*: $\forall x(\mathrm{S}_1[x] \to \mathrm{S}_2[x])$)

If such a sentence is true, then we say that $\mathrm{S}_1[x]$ is a **sufficient condition** for $\mathrm{S}_2[x]$, or, equivalently, that $\mathrm{S}_2[x]$ is a **necessary condition** for $\mathrm{S}_1[x]$. A special case of this is that a sentence of this format is true if, no matter what x is, $\mathrm{S}_1[x]$ is false. Such a sentence is said to be **vacuously true**. For example, the sentence

For every x with $x \neq x$, $x = 2$

is (vacuously) true. If a universal sentence of the form

For every x, $\mathrm{S}_1[x]$ iff $\mathrm{S}_2[x]$

(i.e. whose scope is a biconditional) is true, then we say $\mathrm{S}_1[x]$ is a **necessary and sufficient condition** for $\mathrm{S}_2[x]$.

Existential sentences are abbreviated in a similar way:

There exists x with $\mathrm{S}_1[x]$, such that $\mathrm{S}_2[x]$.

is understood to be shorthand for the following:

There exists x such that $\mathrm{S}_1[x]$ and $\mathrm{S}_2[x]$
(*Abbreviated form*: $\exists x(\mathrm{S}_1[x] \wedge \mathrm{S}_2[x])$)

Note here the use of parentheses for disambiguation. Without the parentheses, it would be hard to be sure whether the scope of the quantification is the conjunctive sentence or just its first conjunct. This is a common device in Mathese. Both round and square parentheses can be used, and multiple sets of parentheses can be used in the same sentence.

Exercise 2.1. Translate the following Mathese sentences into Mathese symbols. Don't worry about whether the sentences are true or false, or provable or unprovable; in math it's necessary to be able to express things that are false, and things that are true but unprovable.

1. There exists a set x such that [(the empty set is in x) and (for every set y, if y is in x then the successor of y is in x)].

2. There does not exist a set x such that [(x is a member of x) and (for every set y, if y is in x then y is equal to x)].

3. For every set n, n belongs to the successor of n.

4. For every set n, either n is zero or there exists a set m such that n is the successor of m.

Exercise 2.2. Translate the mathese sentences in Exercise 2.1 into sentences of ordinary English. (If you are starting to have trouble distinguishing Mathese from ordinary English, imagine you have to translate these sentences for your parents, dentist, vet, bartender, etc.)

2.3 Defining Predicates

At the outset, the only predicates in Mathese are *equals* (abbreviated $=$) or synonyms such as *is the same as* or *is identical to*, and *is a member of* (abbreviated $\in$) or synonyms such as *belongs to* or *is an element of*. But we can *define* new predicates in terms of these and other predicates which have already been defined. The *arity* of a defined predicate is the arity of the condition that is used to define it. For example, we define x *is empty* to mean $\forall y(y \notin x)$, and x *is a singleton* to mean $\exists! y(y \in x)$; these are unary predicates. In x *is a subset of* y (*abbreviation*: $x \subseteq y$), $\subseteq$ is a binary predicate defined by the condition $\forall z(z \in x \rightarrow z \in y)$.

2.4 Defining Names

If we can prove (i.e. provide a persuasive valid argument based only on our assumptions about set theory and other things that have already been proved) that there exists a unique set x such that $S[x]$, where $S[x]$ is some condition on x, then we permit ourselves to bestow a name on that set. For example, it is easy to show that there is a unique set x such that x is empty. (The existence part of the proof is by the Empty set assumption, and the uniqueness part of the proof is an application of Extensionality.) In this case, as we saw in Chapter 1, the set in question is named $\emptyset$ (read 'the empty set').

2.4.1 Functional Names

Often we can show that for any set y, there exists a unique set x satisfying some condition $\mathrm{S}[x, y]$. In such cases, we permit ourselves to introduce a **functional name**, which is basically a scheme which, for each y, provides a name for the unique set x such that $\mathrm{S}[x, y]$. To make an analogy with real life: obviously everybody has a mother, so we can use the functional name

y's mom to refer to the unique individual x such that x is a mother of y, no matter who y is. Returning to sets, it is easy to prove that for any set y, there is a unique set x such that y is the only member of x. This justifies introducing the functional name $\mathsf{singleton}(y)$, abbreviated $\{y\}$. Likewise, we introduce the functional name $\mathsf{successor}(y)$, abbreviated $s(y)$, which, for each set y, names the unique set x that satisfies the binary condition $x = y \cup \{y\}$.

This naming convention extends to names that depend on more than one variable. Again, to take a real-life example, we might introduce the functional name **x's seniority over y**: for any two individuals x and y this is defined to be the number of days (rounded off) from x's birthdate to y's birthdate (this is a negative integer if y's birthdate precedes x's). The general principle is that if, for some positive natural number n and some $(n+1)$-ary condition $\mathrm{S}[x_0, \ldots, x_n]$ we can prove

$$\forall x_1 \cdots \forall x_n \exists! x_0 \mathrm{S}[x_0, \ldots, x_n] \,,$$

then we are allowed to make up a functional name $\mathsf{name}(x_1, \ldots, x_n)$ which for each choice of values for the n variables $x_1, \ldots, x_n$ provides a name for the unique set which satisfies the condition for that choice of values.

Exercise 2.3. Translate the following sequences of symbols into clear, unambiguous English, either standard English or Mathese or a mixture of the two, whichever you prefer. Again, don't worry about whether the sentences are true, or whether they are provable. (*Note*: In Mathese, there is a standard way of avoiding repeating sequences of quantifiers of the same kind (i.e. all universal or all existential), e.g.:

> Instead of *For every x, for every y ...*, say *For all x and y ...* or *For any two sets x and y, ...*
>
> Instead of *There exists x such that there exists y such that there exists z ...*, say *There exist three sets x, y, and z such that ...*)

1. $\forall x \exists y \forall z (z \in y \leftrightarrow \exists u \exists v [z \in u \wedge u \in v \wedge v \in x])$
2. $\exists! x \forall y (y \in x \leftrightarrow y \notin y)$
3. $\forall x \forall y [x \neq y \rightarrow (s(x) \neq s(y))]$
4. $\forall x ([0 \in x \wedge \forall y (y \in x \rightarrow s(y) \in x)] \rightarrow z \subseteq x)$

Chapter 3

Relations

3.1 Introduction

Intuitively, a **relation** is the sort of thing that either does or does not hold between certain things, e.g. the love relation holds between Kim and Sandy just in case Kim loves Sandy, and the less-than relation holds between two natural numbers A and B just in case $A < B$. How should we represent relations mathematically if sets are all we have to work with? A simple-minded first pass might be to represent the love relation as the set of all pairs $\{A, B\}$ such that A and B are two people and A loves B. (Actually, A and B would not be *people* at all, but rather certain sets that we have chosen as theoretical stand-ins for (representations of) people: remember that the only things in our mathematical workspace are sets!) Unfortunately, this is too simple, since, for example, we are left with no way to represent unrequited love: what if Kim loves Sandy but Sandy does not love Kim?

A more promising approach is to represent love as the set of *ordered pairs* $\langle A, B\rangle$ such that A loves B. Of course nobody is under the illusion that a set of ordered pairs is the answer Cole Porter had in mind when he wrote *What is this Thing Called Love?* It is what a formal semanticist would call the **extension** of the love relation. (The appropriate way to mathematically represent the actual love relation, as opposed to its extension, is a question we will turn to in Part **??** when we consider how to represent linguistic meaning.) To take a less vexing example, we can consider the relation $\subseteq_U$ of

set inclusion restricted to the subsets of a given set U to be the following set of ordered pairs:[1]

$$\subseteq_U =_{\text{def}} \{\langle A, B\rangle \in \wp(U) \times \wp(U) \mid A \subseteq B\}$$

More generally, we now *define* the notion of relation as follows: a **relation** between A and B is a subset of $A \times B$. Equivalently, it is

- A set of ordered pairs whose first and second components are in A and B respectively.
- A member of $\wp(A \times B)$.

In the special case where $A = B$, we speak of a relation **on** A. For example, $\subseteq_A$ is a relation on $\wp(A)$. But note: according to the way we have defined the notion of a relation, there is no $\subseteq$ relation!

Exercise 3.1. Why is there no $\subseteq$ relation?

As a matter of notation, we usually write $a\ R\ b$ to mean that the ordered pair $\langle a, b\rangle$ is in the relation R; that is, $a\ R\ b$ is just another way to say $\langle a, b\rangle \in R$.

An important special case arises when $A = B$ and the relation is

$$\mathsf{id}_A =_{\text{def}} \{\langle x, y\rangle \in A \times A \mid x = y\}\ .$$

This relation is called the **identity** relation on A. For any set A, the **subset inclusion** relation

$$\subseteq_A =_{\text{def}} \{\langle x, y\rangle \in \wp(A) \times \wp(A) \mid x \subseteq y\}$$

and the **proper subset inclusion** relation

$$\subsetneq_A =_{\text{def}} \{\langle x, y\rangle \in \wp(A) \times \wp(A) \mid x \subsetneq y\}$$

are both relations on $\wp(A)$. The **less than** relation

$$< =_{\text{def}} \{\langle m, n\rangle \in \omega \times \omega \mid m \subsetneq n\}$$

is a relation on the set ω of natural numbers. (We will get clear about what a natural number is and show that there is a set of them in Chapter 6.)

[1] On the right-hand side of the following definition, we are making use of a commonplace notational convention whereby $\{\langle x, y\rangle \in A \times B \mid \phi\}$ abbreviates $\{z \in A \times B \mid \exists x \exists y(\phi \wedge z = \langle x, y\rangle)\}$.

3.2 Inverting and Composing Relations

If R is a relation from A to B, the **inverse** of R is the relation from B to A defined as follows:

$$R^{-1} =_{\text{def}} \{\langle x, y\rangle \in B \times A \mid y\ R\ x\}$$

For example, suppose $\leq$ is the standard order on the natural numbers (to be defined precisely in Chapter 6); its inverse is the relation $\geq$. Similarly, $<^{-1} = >$ and ${\subseteq_A}^{-1} = \supseteq_A$, where $\supseteq_A$ holds of any two subsets a and b of A just in case $b \subseteq a$. And the inverse of the (extension) of the love relation is the is-loved-by relation. It is easy to see that for any set A,

$$\mathsf{id}_A^{-1} = \mathsf{id}_A\ ,$$

and that for any relation R,

$$(R^{-1})^{-1} = R\ .$$

As we have seen, a relation is defined as a subset of a cartesian product $A \times B$. More precisely, this should have been called a **binary** relation. Likewise, we can define a **ternary relation** among the sets A, B, and C to be a subset of the threefold cartesian product $A \times B \times C$; thus a ternary relation is a set of ordered triples. For $n > 3$, n-fold cartesian products and n-ary relations are defined in the obvious way.

Recall that a **cartesian power** is a cartesian product all of whose factors are the same, e.g.

$$\begin{aligned} A^{(0)} &= 1 \\ A^{(1)} &= A \\ A^{(2)} &= A \times A \\ A^{(3)} &= A \times A \times A \end{aligned}$$

Correspondingly, a **unary relation** on A is just a subset of A, and a **nullary relation** on A is a subset of 1, i.e. either 1 or 0.

Suppose R is a relation from A to B and S is a relation from B to C. Then the **composition** of S and R is the relation from A to C defined[2] by

$$S \circ R =_{\text{def}} \{\langle x, z\rangle \in A \times C \mid \exists y \in B(x\ R\ y \wedge y\ S\ z)\}\ .$$

[2]This composition is sometimes written with the composed relations in the other order, as $R\,;S$.

It is easy to see that if R is a relation from A to B, then

$$\mathsf{id}_B \circ R = R = R \circ \mathsf{id}_A \,.$$

Suppose R is a relation from A to B. Then the **domain** and **range** of R are defined as

$$\mathsf{dom}(R) =_{\text{def}} \{x \in A \mid \exists y \in B(x \; R \; y)\}$$

and

$$\mathsf{ran}(R) =_{\text{def}} \{y \in B \mid \exists x \in A(x \; R \; y)\}$$

respectively.

3.3 Special Properties of Relations

Here we collect some useful definitions for future reference. Throughout, we assume R is a binary relation on A.

- Distinct $a, b \in A$ are (R-)**comparable** if either $a \; R \; b$ or $b \; R \; a$; otherwise, they are **incomparable**. R is **connex** iff a and b are comparable for all distinct $a, b \in A$.
- R is **reflexive** if $a \; R \; a$ for all $a \in A$ (i.e. $\mathsf{id}_A \subseteq R$). R is **irreflexive** if $a \not R \; a$ for all $a \in A$ (i.e. $\mathsf{id}_A \cap R = \emptyset$).
- R is **symmetric** if $a \; R \; b$ implies $b \; R \; a$ for all $a, b \in A$ (i.e. $R = R^{-1}$). R is **asymmetric** if $a \; R \; b$ implies $b \not R \; a$ for all $a, b \in A$ (i.e. $R \cap R^{-1} = \emptyset$).
- R is **antisymmetric** if $a \; R \; b$ and $b \; R \; a$ imply $a = b$ for all $a, b \in A$ (i.e. $R \cap R^{-1} \subseteq \mathsf{id}_A$). Thus asymmetry is a special case of antisymmetry; more specifically, a relation is asymmetric iff it is both antisymmetric and irreflexive.
- A relation R is **transitive** if $a \; R \; b$ and $b \; R \; c$ imply $a \; R \; c$ for all $a, b, c \in A$ (i.e. $R \circ R \subseteq R$). R is **intransitive** if $a \; R \; b$ and $b \; R \; c$ imply $a \not R \; c$ for all $a, b, c \in A$ (i.e. $(R \circ R) \cap R = \emptyset$).

It is important to note, in particular, that being asymmetric is not defined as simply not symmetric, and similarly, being intransitive is not equivalent to not being transitive. For example, given a relation R, it is possible for $a \; R \; b$ and $b \; R \; c$ to imply $a \; R \; c$ only for *some* (but not *all*) $a, b, c \in A$; in that case R is neither transitive nor intransitive.

Exercise 3.2. Let A be a fixed set. In this question "relation" means "binary relation on A." Prove that:

1. The intersection of two transitive relations is a transitive relation.
2. The intersection of two symmetric relations is a symmetric relation.
3. The intersection of two reflexive relations is a reflexive relation.

Suppose R is a binary relation on A. Then the **reflexive closure** of R is the relation $R \cup \mathsf{id}_A$, and the **irreflexive interior** of R is the relation $R \setminus \mathsf{id}_A$.

Exercise 3.3. Prove the following:

1. A relation is reflexive iff it is equal to its reflexive closure, and irreflexive iff it is equal to its irreflexive interior.
2. The reflexive closure of a relation R is the intersection of the set of reflexive relations on A which have R as a subset.
3. The irreflexive interior of a relation R is the union of the set of irreflexive relations which are subsets of R.

Exercise 3.4. Let A be any set. What are the reflexive closure and reflexive interior of the following relations?

1. id_A
2. $\subseteq_A$
3. $<$

Exercise 3.5. Which relations discussed so far are symmetric? Which are asymmetric? Which are antisymmetric?

Exercise 3.6. Prove that a relation is asymmetric iff it is both antisymmetric and irreflexive.

Exercise 3.7. Which relations discussed so far are transitive? Which are intransitive?

Chapter 4

Preorders and Equivalences

In this chapter, we examine some special kinds of binary relations: orders, preorders, and equivalence classes.

4.1 Orders and Preorders

A **preorder** is a reflexive transitive relation; and an **order** is an antisymmetric preorder. Preorders are often notated with the symbol $\sqsubseteq$ (read 'less than or equivalent to') or one of its variants. One of the most useful orders overall is the subset relation $\subseteq_A$ on $\wp(A)$. Another example is the relation $\leq$, which is an order on the natural numbers (as we will see in Chapter 6). In linguistic applications, as we will see later on, one of the most widely used orders is the *dominance* order on the nodes of a *tree* (Chapter 10), used in logic to formalize the notion of a *proof* (Chapter 11), and in many syntactic theories to represent the (putative) constituent structure of a linguistic expression. (But not in all syntactic theories; for example, in the family of syntactic theories known as *categorial grammar*, the notion of constituent plays little or no role.)

In many approaches to formal semantics of natural languages, the mathematical objects that model declarative sentence meanings (usually called *propositions*) are preordered by a relation called *entailment*. Without getting technical at this point, if p and p' are the propositions expressed by two natural-language sentence utterances S and S', p entails p' just in case, no matter what the world is like, if S is true with the world that way, then so is S'. In order to have a formal theory of this, we will have to have a way of set-theoretically representing sentence utterances, propositions, and possible ways the world might be. Considerable care is needed here, since one

and the same sentence can express different propositions depending on the context of utterance, and utterances of different sentences can express the same proposition. A controversial issue here is whether or not the entailment relation is antisymmetric. In other words: if two sentences always agree in truth value no matter what the world is like, then must they express the same proposition? We will take up these and related issues in due course.

Let $\sqsubseteq$ be a preorder on A, $S \subseteq A$, and $a \in A$. Then a is a **upper** (**lower**) **bound** of S iff, for every $b \in S$, $b \sqsubseteq a$ ($a \sqsubseteq b$). Suppose moreover that $a \in S$. Then a is **greatest** (**least**) in S iff it is an upper (lower) bound of S. a is a **top** (**bottom**) iff it is greatest (least) in A; if there is a unique top (bottom), it is often written $\top_{\sqsubseteq}$ ($\bot_{\sqsubseteq}$). And a is **maximal** (**minimal**) in S iff $a \sqsubseteq b$ implies $b \sqsubseteq a$ ($b \sqsubseteq a$ implies $a \sqsubseteq b$) for every $b \in S$. It is not hard to see that if S has any greatest (least) elements, then they and only they are maximal (minimal) elements of S.

Now suppose the preorder $\sqsubseteq$ is also antisymmetric (i.e. it is an order). Then S can have at most one greatest (or least) member; in particular, there can be at most one top (or bottom). If a is greatest (or least) in S, then it is the unique maximal (or minimal) element of a.

But it is possible (even if $\sqsubseteq$ is antisymmetric) for a to be the unique maximal (or minimal) element of S without being the greatest (or least) element in S. For that matter, S can have more than one maximal (or minimal) element without any of them being greatest (or least).

Exercise 4.1. Letting S be a set, construct suitable examples of the following:

1. A preorder with a unique maximal (or minimal) element a that is not greatest (or least) in S,

2. A preorder with more than one maximal (or minimal) element that is not greatest (or least), and

3. An antisymmetric preorder with a unique greatest (or least) member.

In a connex preorder, for a to be maximal (minimal) in S is the same thing as for a to be greatest (least) in S. A connex (pre)order is called a **(pre)chain**. A chain is also called a **total order**, or a **linear order**. A chain is called a **well-ordering** provided every non-empty subset of A has a least element. The most familiar example of a well-ordering is the standard ($\leq$) order on the natural numbers.

For linguists, the most familiar chains are the *linear precedence (LP)* orders that arise in the representation of the constituent structure (within

linguistic theories that countenance such things) of a linguistic expression by an ordered tree, namely

1. The LP order on the daughters (immediate constituents) of a nonterminal node, and

2. The LP order on the preterminals.

We will take a close look at the use of tree representations in syntax in Chapters 9 and 10.

4.2 Equivalence Relations

An **equivalence** relation is a symmetric preorder. The symbol $\equiv$ is often used to notate equivalence relations.

Exercise 4.2. Show that the intersection of two equivalence relations on a set A is itself an equivalence relation.

If $\equiv$ is an equivalence relation on a set A, then for each $a \in A$, the ($\equiv$-)**equivalence class** of a is

$$[a]_{\equiv} =_{\text{def}} \{b \in A \mid a \equiv b\} .$$

Usually the subscript is dropped when it is clear from context which equivalence relation is in question. The members of an equivalence class are called its **representatives**. The set of equivalence classes, written $A / \equiv$, is called the **quotient** of A by $\equiv$.

Exercise 4.3. Prove that if $\equiv$ is an equivalence relation on A, then $A / \equiv$ is a **partition** of A, i.e. it is pairwise disjoint and its union is A.

Exercise 4.4. Which relations discussed in Chapter 3 are equivalences? What are their equivalence classes?

For any binary relation R on a set A, the **symmetric interior** of R, written $\mathsf{Sym}(R)$, is defined to be the relation $R \cap R^{-1}$. For example, if R is the relation that holds between a pair of people when the first respects the other, then $\mathsf{Sym}(R)$ is the relation of mutual respect.

Exercise 4.5. Prove that the symmetric interior of a preorder is an equivalence relation.

If $\sqsubseteq$ is a preorder, then $\mathsf{Sym}(\sqsubseteq)$ is called the equivalence relation **induced by** $\sqsubseteq$ and written $\equiv_{\sqsubseteq}$, or just $\equiv$ if it's clear from the context which preorder is under discussion. If $a \equiv b$, then we say a and b are **tied** with respect to the preorder $\sqsubseteq$.

Also, for any relation R, there is a corresponding asymmetric relation called the **asymmetric interior** of R, written $\mathsf{Asym}(R)$ and defined to be $R \setminus R^{-1}$. For example, the asymmetric interior of the love relation on people is the unrequited love relation.

In a context where there is a fixed preorder $\sqsubseteq$, $a \sqsubseteq b$ is usually read 'a is less than or equivalent to b'; if in addition $\sqsubseteq$ is antisymmetric (i.e. an order), then it is read 'a is less than or equal to b' because the only thing tied with a is a itself.

In a context where there is a fixed preorder $\sqsubseteq$, $\mathsf{Asym}(\sqsubseteq)$ is usually read 'strictly less than,' and abbreviated $\sqsubset$. Careful: if $a\ \mathsf{Asym}(\sqsubseteq)\ b$, then not only are a and b not equal, but also they are not equivalent.

If $\sqsubseteq$ is a preorder, then we say c is **strictly between** a and b to mean that a is strictly less than c and c is strictly less than b.

Given a preorder $\sqsubseteq$ on a set A and $a, b \in A$, we say a is **covered by** b if a is strictly less than b and there is nothing strictly between them. The relation consisting of all such pairs $\langle a, b\rangle$ is called the **covering relation induced by** $\sqsubseteq$ and written $\prec_{\sqsubseteq}$, or just $\prec$ when no confusion can arise.

Exercise 4.6. Prove that $\prec$ is an intransitive relation.

Exercise 4.7. Let $\leq$ be the usual order on ω. What is the induced covering relation? (*Hint*: we encountered it earlier, under another name.)

Exercise 4.8. Let $\leq$ be the usual order on the real numbers. What is the induced covering relation?

Exercise 4.9. Remember from grade school (or maybe middle school?) that any positive rational number less than 1 can be represented in a unique way by a fraction $m\,/\,n$ where both m and n are nonzero natural numbers, $m < n$, and m and n have no common factor (other than 1), i.e. the fraction is 'reduced to lowest terms.' In this context, let $\leq$ represent the usual order on the set of such numbers (just take it on faith that there *is* such a set). What is the induced covering relation on $\leq$?

Exercise 4.10. Let U be a set, $\subseteq_U$ the subset inclusion relation on $\wp(U)$, and $\prec$ the corresponding covering relation. In simple English, how do you tell by looking at two subsets A and B of U whether $A \prec B$?

Exercise 4.11. *Background.* For this exercise, let $\mathsf{Refl}(R)$ be the reflexive closure of R, defined in §3.3. Clearly if R is transitive then $\mathsf{Refl}(R)$ is a preorder.

Now suppose P is the set of all people who have ever lived (i.e. a set that we are using to *represent* the collection of people who have ever lived) and let D be a transitive asymmetric relation on P used to represent the relation that holds between a pair of people if the first is a descendant of the second. Let $\sqsubseteq =_{\text{def}} \mathsf{Refl}(D)$, and $\prec$ the corresponding covering relation. To keep things simple, assume (counterfactually, of course) that

A. Every person has exactly two parents, and

B. Any two people with a parent in common have both of their parents in common.

Then do the following:

1. In plain English, why did we require that D be transitive an asymmetric? (That is, what facts of life are modeled by imposing these conditions on D?)

2. Write a formula (a sentence made up of Mathese symbols) expressing the condition A. (*Hint*: it is much easier to express this in terms of $\prec$ than in terms of D!)

3. Write a formula expressing the condition B. (Same hint as immediately above.)

4. Suppose a and b are two people. Write a formula that means that a and b are cousins. (To eliminate any variation in or unclarity about the meaning of English kinship terms, assume that a person's cousins are the children of his or her parents' siblings, not counting ones with whom he or she has a parent in common.)

Translate the following into plain English, using familiar kinship terms.

5. $a \prec b$

6. $b \prec^{-1} a$

7. $a \prec \circ \prec b$

8. $a \, (\prec \circ \prec^{-1}) \setminus \mathsf{id}_P \, b$

9. $a \, (\prec^{-1} \circ \prec) \setminus \mathsf{id}_P \, b$

4.3 Least Upper Bounds and Greatest Lower Bounds

Throughout this section, $\sqsubseteq$ is a preorder on a set A, and $\equiv$ $(= \equiv_{\sqsubseteq})$ is the equivalence relation induced by the preorder.

The set of upper (lower) bounds of S is denoted by $\mathrm{UB}(S)$ ($\mathrm{LB}(S)$). In case S is a singleton $\{a\}$, $\mathrm{UB}(S)$ ($\mathrm{LB}(S)$) is written $\uparrow a$ ($\downarrow a$), read **up** of a (**down** of a).

A least member of $\mathrm{UB}(S)$ is called a **least upper bound (lub)** of S, and a greatest member of $\mathrm{LB}(S)$ is called a **greatest lower bound (glb)** of S. In case $S = A$, the notions of lub and top (glb and bottom) coincide.

Exercise 4.12. Find an example where a lub of S does not belong to S.

Exercise 4.13. Show that if there are any bottoms, they are the least upper bounds of $\emptyset$, and if there are any tops, they are the greatest lower bounds of $\emptyset$.

Exercise 4.14. Prove that if $\sqsubseteq$ is an order on A and $S \subseteq A$, then S can have at most one glb (lub).

For the case $S = A$, this last exercise implies that there can be at most one top and at most one bottom. In the special case $S = \{a, b\}$, if S has a glb (lub), it is usually written $a \sqcap b$ ($a \sqcup b$).

Chapter 5

Functions

5.1 Basic Properties

A relation F between A and B is called a **(total) function** from A to B provided for every $x \in A$, there exists a unique $y \in B$ such that $x\ F\ y$. In that case we write $F : A \to B$. This is often expressed by saying that F **takes** members of A as **arguments** and **returns** members of B as **values** (or, alternatively, **takes its values** in B). Obviously,

$$\mathsf{dom}(F) = A\,.$$

For each $a \in \mathsf{dom}(F)$, the unique b such that $a\ F\ b$ is called the **value** of F **at** a, written $F(a)$. Equivalently, we say F **maps** a to b, written $F : a \mapsto b$.

In formal semantics, linguistic meanings are often represented as functions of certain kinds. For example, it is fairly standard (but not unproblematic) to represent declarative sentence meanings as functions from a set W of *possible worlds* (which themselves are taken to be representations of different possible ways the world might be) to the set 2 (i.e. $\{0, 1\}$). Here 1 and 0 are identified, respectively, with the intuitive notions of truth and falsity, and the set 2 is often called the set of **truth values**. Not quite so straightforward is the use of function terminology by syntacticians, for example referring to the subjects and complements of a verb as its *grammatical arguments*. If a verb were really a function, then what would its domain and codomain be? In due course we'll look into the motivation for talking about verbs and other linguistic expressions as if they were functions.

Note that for any set A, the identity relation id_A is the function from A to A such that

$$\mathsf{id}_A(a) = a$$

for every $a \in A$. In some linguistic theories, identity functions serve as the meanings of *referentially dependent* expressions such as pronouns and gaps.

It is not hard to see (after some reflection) that a relation R from A to B is a function from A to B iff

$$R \circ R^{-1} \subseteq \mathsf{id}_B$$

and

$$\mathsf{id}_A \subseteq R^{-1} \circ R\,.$$

We note here a confusing though standard bit of terminology. Given a function $F : A \to B$, we often call B the **codomain** of F. What is confusing is that if B is a proper subset of some other set B', then clearly also $F : A \to B'$; but then B' must be the codomain of F! Evidently the notion of codomain of a function is not well-defined. Technically, we can clear up this confusion by defining a **(set theoretic) arrow** from A to B to be an ordered triple $f = \langle A, B, F\rangle$, where $F : A \to B$. Now we can unambiguously refer to A and B as the domain and codomain of f, respectively; F is called the **graph** of f. The point is that two distinct arrows can have the same domain and the same graph but different codomains. Thus when we speak (loosely) of a function $F : A \to B$ having B as its codomain, we are really talking about the arrow $\langle A, B, F\rangle$. Having called attention to this abuse of language, we will persist in it without further comment.

For any sets A and B, the **exponential** from A to B is the set of arrows from A to B. This is written B^A, read 'B to the A.' An alternative notation is $A \Rightarrow B$, read 'A into B.' Note for any set A there is a unique function $\Diamond_A : \emptyset \to A$ and a unique function $\Box_A : A \to 1$.

Exercise 5.1. Suppose A is a set. What are the unique functions from $\emptyset$ to A and from A to 1?

Some other important functions include the **successor** function **suc**, the unary operation on the set of natural numbers that maps each natural number to its successor, and the **arithmetic** functions, such as **addition** $(+)$, **multiplication** $(\cdot)$, and **exponentiation** $(\star)$, all binary operations on the set of natural numbers. (In Chapter 6 we will show how to define these functions recursively.) Other examples are the members of A^n, which, for any set A, are called the A-**strings** of length n. These functions, as we will see starting in Chapter 9, are indispensible for formalizing theories of phonology and syntax.

A relation F between A and B is called a **partial function** from A to B provided there is a subset $A' \subseteq A$ such that F is a (total) function from A' to B. If F is a partial function from A to B, we write $F : A \rightharpoonup B$ to distinguish it from the total case.

For $n \geq 0$, an n-ary **(total) operation** on a set A is a function from $A^{(n)}$ to A. So a unary operation on A is just a function from A to itself, and a nullary operation on A is a function from 1 (i.e. $\{0\}$) to A. It is easy to see that there is a one-to-one correspondence between A and A^1, with each $a \in A$ corresponding to the function from 1 to A that maps 0 to a.

Suppose $F : A \to B$. Then F is called:

- **injective**, or **one-to-one**, or an **injection**, if it maps distinct members of A to distinct members of B;
- **surjective**, or **onto**, or a **surjection**, if $\mathsf{ran}(F) = B$; and
- **bijective**, or **one-to-one and onto**, or a **bijection**, or a **one-to-one correspondence**, if it is both injective and surjective.

An important special case of injective functions are defined as follows: if $A \subseteq B$, then the function $\mu_{A,B} : A \to B$ that maps each member of A to itself is called the **embedding** of A into B. (Note that $\mu_{A,B}$ has the same graph as id_A, but possibly a larger codomain.) Also injective are the functions ι_1 and ι_2, called **canonical injections**, from the cofactors A and B of a coproduct $A + B$ into the coproduct, defined by $\iota_1(a) = \langle 0, a\rangle$ and $\iota_2(b) = \langle 1, b\rangle$ for all $a \in A$ and $b \in B$.

Standard examples of surjections are the **projections** π_1 and π_2 of a product $A \times B$ onto its factors A and B respectively, defined by $\pi_1(\langle a, b\rangle) = a$ and $\pi_2(\langle a, b\rangle) = b$ for all $a \in A$ and $b \in B$. For a set A with an equivalence relation $\equiv$, another example of a surjection is the function from A to $A / \equiv$ that maps each member of A to its equivalence class.

Exercise 5.2. Prove that, for any function $f : A \to B$, there is an equivalence relation $\equiv_f$, with two members of A being equivalent just in case f maps them to the same member of B.

Any identity function is a bijection, and we can prove that **suc** is a bijection from ω (the set of natural numbers) to the set $\omega \setminus \{0\}$ of positive natural numbers (see Chapter 6). Other examples of bijections include the function from 2 to 2 that maps 0 and 1 to each other, and the complement operation on the powerset of a set. And for any set A, there is a bijection from A to A^1 that maps each $a \in A$ to the nullary operation that maps 0 to

a. More generally, for any natural number n, there is a bijection from $A^{(n)}$ to A^n that maps each A-string of length n to an n-tuple of elements of A.

For each function $f : A \to 2$, the **kernel** of F is the subset

$$\mathsf{ker}(f) =_{\text{def}} \{x \in A \mid f(x) = 1\} \ .$$

Operations on 2 are called **truth functions**, and, as we will see in Chapter **??**, are used to define the meanings of the first-order logical connectives, such as $\neg$, $\wedge$, $\vee$, and $\rightarrow$.

If B is a set and A one of its subsets, then there is a function $\chi_{A,B} : B \to 2$ such that, for each $b \in B$, $\chi_{A,B}(b) = 1$ iff $b \in A$. This function is called the **characteristic** function of A relative to B, or simply the characteristic function of A if B can be understood from the context (in which case the subscript B is usually omitted).

Exercise 5.3. Prove that for any set B, the function $f_B : \wp(B) \to 2^B$ that maps each subset of B to its characteristic function is a bijection.

Exercise 5.4. Prove that if $f : A \to B$ is a bijection, then its inverse relation f^{-1} is itself a function, and in fact bijective.

Exercise 5.5. Describe the inverse of the function $f_B : \wp(B) \to 2^B$.

Exercise 5.6. Suppose B is a set, and $g_B : \wp(B) \to \wp(B)$ the function that maps each subset B of A to $B \setminus A$. Show that g_B is a bijection.

Exercise 5.7. What is the inverse of g_B?

Exercise 5.8. For any set U, let $\approx_U$ be the binary relation on $\wp(U)$ such that $A \approx_U B$ iff there is a bijection from A to B. Prove that $\approx_U$ is an equivalence relation.

5.2 Composing Functions

Since functions are relations, the definition of composition for relations makes sense when the two relations being composed are functions. Thus if $F : A \to B$ and $G : B \to C$, then $G \circ F : A \to C$, and for every $x \in A$,

$$G \circ F(x) = G(F(x)) \ .$$

It is not hard to see that[1]

$$G \circ F = \{\langle x, z\rangle \in A \times C \mid \exists y \in B(y = F(x) \wedge z = G(y))\} \ .$$

[1]Note that in the set description on the right-hand side of the following equation, we make use of a commonplace notational convention whereby $\exists y \in B\phi$ abbreviates $\exists y(\phi \wedge y \in B)$.

For example, taking it one faith for the moment that there is a set ω whose members are precisely the natural numbers, and that the familiar (binary) arithmetic operations (addition, multiplication, and exponentiation) have been given satisfactory set-theoretic definitions (we will make this precise in due course), let F and G be the functions from ω to ω such that

$$\begin{aligned} F(x) &= x^2 \\ G(x) &= x + 2 \end{aligned}$$

for all $x \in \omega$. Then $G \circ F$ is given by

$$G \circ F(x) = x^2 + 2 \, .$$

Suppose once again that $F : A \to B$ and $G : B \to C$, and suppose moreover that $H : C \to D$. Then it is not hard to see that

$$H \circ (G \circ F) = (H \circ G) \circ F$$

Since functions are relations, the following hold for any function $F : A \to B$:

$$\begin{aligned} &\mathsf{id}_B \circ F = F = F \circ \mathsf{id}_A \\ &F \circ F^{-1} \subseteq \mathsf{id}_B \\ &\mathsf{id}_A \subseteq F^{-1} \circ F \end{aligned}$$

Additionally, it is easy to see that F is surjective iff

$$F \circ F^{-1} = \mathsf{id}_B \, ,$$

and F is injective iff

$$\mathsf{id}_A = F^{-1} \circ F \, .$$

As a special case, if F is a unary operation on A, then

$$\mathsf{id}_A \circ F = F \circ \mathsf{id}_A = F \, .$$

If in addition F is bijective, then the relation F^{-1} is also a unary operation on A, and

$$F \circ F^{-1} = F^{-1} \circ F = \mathsf{id}_A \, .$$

Exercise 5.9. Prove that the composition of two bijections is a bijection.

5.3 Restrictions and Images

Suppose $F : A \to B$, $A' \subseteq A$, and $B' \subseteq B$. Then the **restriction** of F to A' is the function from A' to B given by

$$F \restriction A' = \left\{ \langle u, v \rangle \in F \mid u \in A' \right\} .$$

Note that this is the same function as $F \circ \mu_{A',A}$. The **image** of A' by F is the set

$$F[A'] =_{\text{def}} \left\{ y \in B \mid \exists x \in A' (y = F(x)) \right\} .$$

The **preimage** (or **inverse image**) of B' by F is the set

$$F^{-1}[B'] =_{\text{def}} \left\{ x \in A \mid \exists y \in B' (y = F(x)) \right\} .$$

This is more simply described as

$$\left\{ x \in A \mid F(x) \in B' \right\} .$$

5.4 Monotonicity and Antitonicity

Now suppose we have two sets A and B (pre)ordered by $\sqsubseteq$ and $\leq$ respectively. A function $f : A \to B$ is called **monotonic** or **order-preserving** with respect to the given (pre)orders provided, for all $a, a' \in A$, if $a \sqsubseteq a'$, then $f(a) \leq f(a')$; and f is called **antitonic** or **order-reversing** with respect to the given (pre)orders provided for all $a, a' \in A$, if $a \sqsubseteq a'$ then $f(a') \leq f(a)$. A monotonic (respectively, antitonic) bijection is called a **(pre)order isomorphism** (respectively, **(pre)order anti-isomorphism**) provided its inverse is also monotonic (respectively, antitonic). Two (pre)ordered sets are said to be **(pre)order-isomorphic** provided there is a (pre)order isomorphism from one to the other. Intuitively speaking, (pre)order-isomorphic (pre)orders are 'copies of each other,' differing only in which members they contain.

It is possible to consider to different preorders on the same set. For example, besides the usual order $\leq$ on the nonzero natural numbers, we could also the consider the order $\sqsubseteq$ that holds between a pair of nonzero natural numbers if the first is a factor of (i.e. divides evenly into) the second.

If $\sqsubseteq$ and $\leq$ are two preorders on the same set A, we can ask whether the identity function on A is monotonic from the first to the second. Interestingly, many (all?) languages have a special grammatical construction, called the

correlative comparative construction, to describe situations of this kind. A typical English example is a sentence such as *The more expensive an SUV is, the more cupholders it has*, which asserts that the identity function on the set of SUVs is monotonic from $\sqsubseteq$ to $\leq$, where, for two SUVs a and b, $a \sqsubseteq b$ means b costs at least as much as a, and $a \leq b$ means that b has at least as many cupholders as a.

Chapter 6

Induction and Recursive Definition

6.1 The Natural Numbers

In Chapter 1, we introduced 0 (also known as $\emptyset$), its successor $1 = s(0) = 0 \cup \{0\} = \{0\}$, 1's successor $2 = s(1) = 1 \cup \{1\} = \{0, 1\}$, and 2's successor $3 = s(2) = 2 \cup \{2\} = \{0, 1, 2\}$. This is how the first four natural numbers are usually modeled within set theory; it's intuitively obvious that we could go on in the same way to model as many of the natural numbers as time would permit. Note that $0 \in 1 \in 2 \in 3 \in \cdots$ and $0 \subseteq 1 \subseteq 2 \subseteq 3 \cdots$ Is there a set consisting of *all* the natural numbers? The assumptions we made in Chapter 1 do not seem to enable us to draw this conclusion. It would be most useful to have such a set, but we are not yet quite in a position to add the assumption that there is a set whose members are precisely the natural numbers, since so far we haven't said what a natural number is! But we are about to.

A set is called **inductive** iff it has 0 as a member and has the successor of each of its members as a member. We then define a **natural number** to be a set which belongs to every inductive set. It is not hard to show that 0, 1, 2, and 3 are all natural numbers. But at this stage, for all we know, *every* set might be a natural number. After all, even though we defined what it means for a set to be inductive, at this point we don't know that there *are* any inductive sets! What if there weren't any? In that case, it's easy to see that indeed every set would be a natural number. And then, since (as we already know) there is no set of all sets, there could not be a set of all the

natural numbers. So if we want there to be a set of all natural numbers, there better be at least one inductive set.

We now add to our assumptions about sets the following:

Assumption 7 (Natural Numbers)**.** There is a set whose members are the natural numbers.

By Extensionality, there can only be one such set. We call it ω. With the help of this assumption, it is now easy to prove the following two theorems:[1]

Theorem 6.1. *ω is inductive.*

Exercise 6.1. Prove Theorem 6.1.

Theorem 6.2. *ω is a subset of every inductive set.*

Exercise 6.2. Prove Theorem 6.2.

The relation $<$ (read **less than**) on ω is defined by $n < m$ iff $n \in m$, and the relation $\leq$ (read **less than or equal to**) by $n \leq m$ iff $n < m$ or $n = m$. (So $\leq$ is the reflexive closure of $<$.)[2] The terminology 'less than or equal to' is justified, since in fact $\leq$ is an order, as we will show. In fact we will show more, namely that $\leq$ is a well-ordering (and in particular a linear order).

6.2 Induction and Recursive Definition

The following theorem is a corollary of the preceding one (6.2):

Theorem 6.3 (Principle of Mathematical Induction)**.** *The only inductive subset of ω is ω.*

Exercise 6.3. Prove the Principle of Mathematical Induction.

[1] A **theorem** is just something important that we can prove. More generally, something that we can prove is usually called a **proposition**. (*Note*: this is a different use of the term *proposition* than in linguistic semantics, where it refers to the interpretation of a declarative sentence utterance.) So a theorem is an important proposition. A **lemma** is a proposition which is not so important in and of itself, but which is used in order to prove a theorem. And a **corollary** of a proposition is another proposition which is easily proved from it.

[2] Later we will be able to prove that, for any two natural numbers n and m, $n < m$ iff $n \subsetneq m$, and $n \leq m$ iff $n \subseteq m$.

The Principle of Mathematical Induction (PMI) is one of the mathematician's most important resources for proving theorems. It is applicable any time we want to prove that a condition $\phi[n]$ is true for every natural number n. The trick is to consider the set $\{n \in \omega \mid \phi[n]\}$ and show that it is inductive. To put it another way, we first prove $\phi[0]$ (this is called the **base case** of the proof) and then prove that, if we assume $\phi[k]$ for an arbitrary natural number k (the so-called **inductive hypothesis**), then $\phi[s(k)]$ follows (the so-called **inductive step**). By way of illustration, we prove the following:

Proposition 6.4. *Let* $\mathbf{suc} : \omega \to \omega$ *be the function that maps each natural number to its successor. Then* $\mathsf{ran}(\mathbf{suc}) = \omega \setminus \{0\}$.

Proof. Obviously $0 \notin \mathsf{ran}(\mathbf{suc})$. Let T be the set of all natural numbers that are either 0 or else the successor of some natural number. We must show that T is inductive, that is that

1. $0 \in T$, and
2. for each $n \in T$, $\mathbf{suc}(n) \in T$.

But both of these are immediate consequences of the definition of T. □

Why do we persist in saying "$\mathbf{suc}(n)$" instead of "1+n"? Answer: because the operation of addition for natural numbers has not been defined yet. Yet it seems clear how addition works: for any natural number m, $m + 0$ should be m; and if k is nonzero (so that it is the successor of some other natural number n), then $m + k$ should be the successor of $m + n$. That is, for each $m \in \omega$ we would like to *define* addition by the equations

$$m + 0 = m$$

and

$$m + \mathbf{suc}(n) = \mathbf{suc}(m + n) .$$

Definitions of this kind are called **recursive**. But how do we know recursive definitions make sense? The answer is provided by the **Recursion Theorem**, henceforth abbreviated **RT**:

Theorem 6.5 (Recursion Theorem)**.** *Let* X *be a set,* $x \in X$*, and* $F : X \to X$*. Then there exists a unique function* $h : \omega \to X$ *such that*

1. $h(0) = x$; *and*
2. *for every* $n \in \omega$, $h(\mathbf{suc}(n)) = F(h(n))$.

RT is not hard to prove, but the proof is a bit long. So we relegate it to an appendix (A), and turn straightaway to some applications.

6.3 Arithmetic

6.3.1 Addition

As our first application of RT, let's show that the informal recursive definition of addition given above actually makes sense.

To get started, suppose $m \in \omega$. We'll use RT to show there is a function A_m such that $A_m(0) = m$ and $A_m(\mathbf{suc}(n)) = \mathbf{suc}(A_m(n))$. The trick, as always when applying RT, is to find the right instantiations of X, x, and F. In the present case the happy choices are $X = \omega$, $x = m$, and $F = \mathbf{suc}$; with these choices, the function h whose unique existence is guaranteed by RT has just the properties we want for A_m. We then define the **addition** operation $+ : \omega^{(2)} \to \omega$ such that, for all $m, n \in \omega$, $m + n =_{\text{def}} A_m(n)$. It follows from this definition that $m + 0 = m$ for all $m \in \omega$ and $m + \mathbf{suc}(n) = \mathbf{suc}(m + n)$ for all $m, n \in \omega$, as desired.

Theorem 6.6. *For every natural number n, $1 + n = \mathbf{suc}(n)$.*

Exercise 6.4. Prove Theorem 6.6.

6.3.2 Multiplication

Turning next to multiplication, we first use RT to define multiplication by a fixed natural number m. We want a function M_m such that

1. $M_m(0) = 0$; and
2. for every $n \in \omega$, $M_m(\mathbf{suc}(n)) = m + M_m(n)$.

To this end, we apply RT again, this time with $X = \omega$, $x = 0$, and $F = A_m$. We then define the **multiplication** operation $\cdot : \omega^{(2)} \to \omega$ such that $m \cdot n =_{\text{def}} M_m(n)$. So $m \cdot 0 = 0$ and $m \cdot (1 + n) = m + m \cdot n$, which is as it should be.

Theorem 6.7. *For every $n \in \omega$, $1 \cdot n = n$.*

Exercise 6.5. Prove Theorem 6.7.

With more time and ambition, one can also prove the familiar Five Laws of Arithmetic (hereafter we omit the '$\cdot$' for multiplication):

Commutativity of Addition $m + n = n + m$

Associativity of Addition $m + (n + p) = (m + n) + p$

Commutativity of Multiplication $mn = nm$

Associativity of Multiplication $m(np) = (mn)p$

Distributivity of Multiplication Over Addition $m(n+p) = mn+mp$

Exercise 6.6. Prove the commutativity of addition.

Exercise 6.7. Use RT to define the **exponentiation** operation $m \star n$ customarily written m^n. (*Hint*: define $m \star n$ to be $E_m(n)$ where $E_m(0) = 1$ and $E_m(\mathbf{suc}(n)) = m \cdot E_m(n)$. That is, as with $+$ and $\cdot$, start by holding m fixed. The heart of the problem is to correctly identify the appropriate values of X, x, and F to use in applying RT.) This definition should establish the first two of the following three general properties of exponentiation:

1. $m^0 = 1$
2. $m^{1+n} = m(m^n)$
3. $m^{n+p} = (m^n)(m^p)$

Note that the second is a special case of the third, which is called the **Law of Exponents**.

6.3.3 The Infinitude of the Natural Numbers

Everyone knows that there is an infinite number of natural numbers, but what exactly does that mean? A set is called **finite** if it is in one-to-one correspondence with a natural number, and **infinite** otherwise. A set is called **Dedekind infinite** if it is in one-to-one correspondence with a proper subset of itself. On the basis of the assumptions we've made so far about sets, it's possible to prove (see Chapter 7) that any Dedekind-infinite set is infinite.[3] Since we already know that $\mathsf{ran}(\mathbf{suc}) = \omega \setminus \{0\}$, we could then show ω is infinite if we could show that $\mathbf{suc} : \omega \to \omega$ is injective. This is of course the case; a sketch of a proof follows.

First, we define a set A to be **transitive** iff every member of a member of A is itself a member of A. It is easy to see that all three of the following conditions on a set A are equivalent to transitivity:

1. $(\bigcup A) \subseteq A$;

[3] To prove the converse, however, we need an additional assumption, viz. the Choice assumption (AC, Chapter 7). AC also enables us to prove that ω is a 'smallest' infinite set, in the sense of being in one-to-one correspondence with a subset of any other infinite set.

2. every member of A is a subset of A; and
3. $A \subseteq \wp(A)$.

The proof that **suc** is injective requires a couple of preliminary results:

Lemma 6.8. *If A is transitive, then $\bigcup s(A) = A$.*

Proof. We use the (easily proved) general fact about union that

$$\bigcup(x \cup y) = \left(\bigcup x\right) \cup \left(\bigcup y\right)$$

and reason as follows:

$$\begin{aligned}\bigcup s(A) &= \bigcup(A \cup \{A\}) \\ &= (\bigcup A) \cup (\bigcup\{A\}) \\ &= (\bigcup A) \cup A \\ &= A\end{aligned}$$

The last step follows from the fact that $\bigcup A \subseteq A$ for transitive A. □

Lemma 6.9. *Every natural number is transitive.*

Exercise 6.8. Prove Lemma 6.9.

Theorem 6.10. suc *is injective.*

Proof. Suppose $\mathbf{suc}(m) = \mathbf{suc}(n)$. Then $\bigcup\mathbf{suc}(m) = \bigcup\mathbf{suc}(n)$. But m and n are transitive (by Lemma 6.9), so (by Lemma 6.8) $\bigcup\mathbf{suc}(m) = m$ and $\bigcup\mathbf{suc}(n) = n$. Therefore $m = n$. □

As noted above, the infinitude of ω is a corollary of this.

6.3.4 The Well-Ordering of ω

We now have the resources to establish that the relation $\leq$ on ω is an order, indeed a well-ordering (i.e. a chain such that every nonempty subset of ω has a least member). Given how obvious this seems, the argumentation required is surprisingly intricate and too long to reproduce in full detail here, so we content ourselves with an outline, including key lemmata and proof sketches.

Recall that by definition $m < n$ iff $m \in n$, and $m \leq n$ iff $m < n$ or $m = n$.

Theorem 6.11. *For all $n \in \omega$, $n = \{m \in \omega \mid m < n\}$.*

Proof. To show inclusion, suppose $m \in n$. Since ω is transitive, $m \in \omega$. Then $m < n$. To show the reverse inclusion, suppose $m < n$. Then by definition, $m \in n$. □

Lemma 6.12. *For all $m, n \in \omega$, $m < \mathbf{suc}(n)$ iff $m \leq n$.*

Proof. We have

$$\begin{aligned} & m < \mathbf{suc}(n) \\ \text{iff } & m \in \mathbf{suc}(n) \\ \text{iff } & m \in n \cup \{n\} \\ \text{iff } & m \in n \text{ or } m \in \{n\} \\ \text{iff } & m \in n \text{ or } m = n \\ \text{iff } & m \leq n \,. \end{aligned}$$

□

Lemma 6.13. *For all $m, n \in \omega$, $m \in n$ iff $\mathbf{suc}(m) \in \mathbf{suc}(n)$.*

Proof. For the *only-if* direction, assume $\mathbf{suc}(m) \in \mathbf{suc}(n)$. Then $\mathbf{suc}(m) < \mathbf{suc}(n)$, so by Lemma 6.12, $\mathbf{suc}(m) \leq n$, i.e. either $\mathbf{suc}(m) \in n$ or $\mathbf{suc}(m) = n$. If $\mathbf{suc}(m) \in n$, then $m \in \mathbf{suc}(m) \in n$, so $m \in n$ by transitivity. Otherwise $\mathbf{suc}(m) = n$; but $\mathbf{suc}(m) = m \cup \{m\}$, from which it follows easily that $m \in n$.

For the if direction, we use PMI. Let

$$T = \{n \in \omega \mid \forall m \in n(\mathbf{suc}(m) \in \mathbf{suc}(n))\} \,.$$

It is sufficient to show that T is inductive. □

Exercise 6.9. Complete the proof of Lemma 6.13 by showing that T is inductive.

Lemma 6.14. *For all $n \in \omega$, $n \notin n$.*

Proof. This is another inductive proof. Let $T = \{n \in \omega \mid n \notin n\}$. It suffices to show that T is inductive. The base case is trivial, and the inductive step is an easy consequence of Lemma 6.13. □

Theorem 6.15. *$<$ is transitive, irreflexive, and connex.*

Proof. Transitivity follows readily from Lemma 6.9 and irreflexivity from Lemma 6.14. Connexity is proved inductively, by showing that the set

$$T = \{n \in \omega \mid \forall m \in \omega(m \neq n \to (n \in m \vee m \in n))\}$$

is inductive. □

Exercise 6.10. Complete the proof of Theorem 6.15 by proving that T is inductive. (*Hint*: for the inductive step, use Lemmas 6.12 and 6.13.)

As two easy consequences of this theorem, we have the following:

Corollary 6.16. *For all $m, n \in \omega$, $m \in n$ iff $m \subsetneq n$.*

Corollary 6.17. *$\leq$ is a chain.*

And finally:

Theorem 6.18. *$\leq$ is a well-ordering.*

Proof. Suppose $A \subseteq \omega$ has no least element. It suffices to show $A = \emptyset$. To this end, let B be the set of all natural numbers n such that no natural number less than n belongs to A. All that is required is to show B is inductive. □

Exercise 6.11. Complete the proof of Theorem 6.18 by showing that B is inductive. (*Hint*: use Lemma 6.12 in the inductive step.)

6.4 Transitive Closure and Reflexive Transitive Closure

Let R be a binary relation on A. Then informally, the **transitive closure** of R, written R^+, transitive closure is usually 'defined' as follows: For all $n \in \omega$, recursively define $h(n)$ by $h(0) = \mathsf{id}_A$ and $h(n+1) = h(n) \circ R$. Then $R^+ =_{\text{def}} \bigcup_{n>0} h(n)$.[4] We leave as an exercise the formal justification of this definition using RT. Similarly, the **reflexive transitive closure** of R, written R^*, is $\bigcup_{n\in\omega} h(n)$. Note that $R^* = R^+ \cup \mathsf{id}_A$.

Exercise 6.12. Use RT to give a correct definition of the function h used in the definition of transitive closure.

[4]Note that if $f : A \to B$, we write $\bigcup_{x\in A} f(x)$ for $\bigcup \mathsf{ran}(f)$. Also, $\bigcup_{n>0}$ abbreviates $\bigcup_{n\in\omega\setminus\{0\}}$ and $\bigcup_{i<n}$ abbreviates $\bigcup_{i\in n}$.

Lemma 6.19. *For any binary relation R on a set A, R^+ is transitive.*

Exercise 6.13. Prove Lemma 6.19.

Theorem 6.20. *The transitive closure of R is the intersection of all the transitive relations of which R is a subset, i.e.*

$$R^+ = \bigcap \left\{ S \subseteq A^{(2)} \mid R \subseteq S \text{ and } S \text{ is transitive} \right\} .$$

6.5 Replacement and Strong Recursion

It turns out that the RT is not as powerful as we might like. For example, for some set U, suppose we wish to define a function with domain ω such that $f(0) = U$ and for each $n \in \omega$, $f(\mathbf{suc}(n)) = \bigcup f(n)$. Precisely such a function is employed in the standard definition of the transitive closure of U:

$$\mathcal{T}(U) =_{\text{def}} \bigcup_{n \in \omega} f(n) .$$

Unfortunately, the RT as presently formulated doesn't enable us to define such an f recursively, since there is evidently no guarantee that there is a set X to which all the $f(n)$ belong. Similarly, our RT doesn't enable us to define a function g with domain ω such that $g(0) = u$ and for each $n \in \omega$, $g(\mathbf{suc}(n)) = \wp(f(n))$. Essentially, the problem is that axiomatic concepts like union and powerset are not functions but in many ways act as if they were.

To solve these and similar problems, we need a stronger version of RT that takes as its point of departure not necessarily a function (the one that is applied at each step of the recursive definition), but rather a weaker notion of a **functional condition**. Suppose $\phi[x, y]$ is a condition with x and y among its free variables. Then $\phi[x, y]$ is called x, y-**functional** provided, for every x, there is a unique y such that $\phi[x, y]$ holds. Significantly, this does not require that there be a function f such that $\phi[x, y]$ holds iff $y = f(x)$. For example, $\phi[x, y]$ could be the condition $y = \bigcup x$ or the condition $y = \wp(x)$. Then we can assume the following:

Assumption 8 (Replacement)**.** Let $\phi[x, y]$ be x, y-functional and A any set. Then there is a set whose members are those y such that $\phi[x, y]$ holds for some $x \in A$.

Note that, as for Separation, the Replacement assumption is actually a schema of assumptions, with one assumption for each choice of the functional

condition $\phi[x, y]$. Informally, Replacement says that if we replace each member of a set by something that the member is related to by a functional condition, the result is also a set.

With this new assumption in place, we can state a generalized version of RT, called the **Strong Recursion Theorem** (**SRT**):

Theorem 6.21 (Strong Recursion Theorem)**.** *Let $\phi[x, y]$ be x, y-functional and A a set. Then there is a function h with domain ω such that*

1. *$h(0) = A$, and*
2. *for every $n \in \omega$, $\phi[h(n), h(\mathbf{suc}(n))]$.*

A proof of Theorem 6.21 is given by Abian (1965).

Theorem 6.22. *For any set U, $\mathcal{T}(U)$ is transitive.*

Exercise 6.14. Prove Theorem 6.22.

Theorem 6.23. *For every transitive set T with $U \subseteq T$, $\mathcal{T}(U) \subseteq T$, i.e., $\mathcal{T}(U)$ is the smallest transitive set containing U.*

Exercise 6.15. Prove Theorem 6.23.

6.6 Hasse Diagrams

A **Hasse diagram** is a kind of textual (paper or blackboard) diagrammatic representation of a preorder $\sqsubseteq$ on a set A, made up of dots and straight line segments directly connecting two dots (here "directly" means there are no dots on the line segment other than the two being connected). The line segments are of two kinds:

1. Nonhorizontal (i.e. either slanting or vertical) single line segments, and
2. Horizontal double line segments.

The interpretation is as follows: the dots represent the members of A; if (the dots representing) b and a are connected by a single nonhorizontal line segment and b is higher (on the page or board) than a, then $a \prec b$; and if a and b are connected by a horizontal double line segment, then a and b are *tied*, i.e. $a \sqsubseteq b$ and $b \sqsubseteq a$. (So if $\sqsubseteq$ is an order, there will be no horizontal double line segments.) Figure 6.1 shows a simple example of a Hasse diagram.

Any finite preorder can be represented by a Hasse diagram, but not every infinite one can. (There can be infinite Hasse diagrams, but there is not

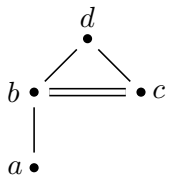


Figure 6.1: Hasse diagram of the reflexive transitive closure of the relation $\{\langle a, b\rangle, \langle b, c\rangle, \langle c, b\rangle, \langle b, d\rangle, \langle c, d\rangle\}$ on the set $\{a, b, c, d\}$.

enough time to draw all of one! Sometimes the gist of an infinite Hasse diagram can be conveyed with judicious use of ellipsis ('and so on') dots, though.) For antisymmetric preorders (i.e. orders) the property of being representable by a Hasse diagram is easy to express precisely in set-theoretic terms: it is the property of being the reflexive transitive closure of its own covering relation. It can be shown (though the details are a bit tedious) that any finite order has this property.

Exercise 6.16. Draw Hasse diagrams to show that it is possible for:

1. An order to have a unique maximal element but no top;
2. An order to have more than one maximal element but no top;
3. A preorder to have more than one top;
4. An order to have no maximal element.

Exercise 6.17. Draw a Hasse diagram for the subset-inclusion order on the set $\wp(3)$.

Chapter 7

Infinities

7.1 Equinumerosity

Two sets A and B are said to be **equinumerous**, written $A \approx B$, iff there is a bijection from A to B. It follows that a set is finite iff it is equinumerous with a natural number and infinite otherwise.

It is easy to show that equinumerosity is an equivalence relation on the powerset of any set. It is not hard to show that for any set A, $\wp(A) \approx 2^A$; the bijection in question maps each subset of A to its characteristic function (with respect to A).

Intuitively speaking, equinumerosity may seem to amount to 'having the same number of members.' As we soon will see, this intuition is essentially on the mark in the case of finite sets. But when the sets involved are infinite, intuition may fail us. For example, all the following sets can be shown to be (pairwise) equinumerous: ω, $\omega \times \omega$, the set $\mathbb{Z}$ of integers, and the set $\mathbb{Q}$ of rational numbers.[1]

Not all infinite sets are equinumerous! To put it imprecisely but suggestively, there are *different sizes of infinity*. For example, as Cantor famously proved, $\omega \not\approx I$, where I is the set of real numbers from 0 to 1. It is beyond the scope of this book to consider how the real numbers are modeled set-theoretically, but for our purposes it will suffice to think of I as the set of 'decimal expansions,' i.e. the set of functions from ω to $10\,(= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\})$ excluding the ones which for some natural

[1] Actually proving all these things would of course require us to model 'the integers' and 'the rationals' as sets. There are standard ways of doing that, but limitations of space and time prevent us from spelling them out here.

number n assign 9 to every natural number greater than or equal to n.[2] The proof is surprisingly simple: suppose f is an injection from ω to I. Then f cannot be a surjection. To see why, let r be the member of I (i.e. the function from ω to 10) which, for each $n \in \omega$, maps n to 6 if $f(n)(n) = 5$ and maps n to 5 otherwise. A moment's thought shows that r cannot be in the range of f!

Theorem 7.1 (Cantor)**.** *For any set A, $A \not\approx \wp(A)$.*

Proof. Let g be a function from A to $\wp(A)$. We will show g cannot be surjective, and therefore cannot be bijective. To this end, let $B = \{x \in A \mid x \notin g(x)\}$. Then obviously $B \in \wp(A)$. But B cannot be in the range of g. For suppose it were. In that case there would exist a $y \in A$ such that $B = g(y)$. But then $y \in B$ iff $y \notin g(y)$. In other words, $y \in B$ iff $y \notin B$, a contradiction. □

7.2 Dedekind Infinity

As mentioned in §6.3.3, a set is said to be *Dedekind infinite* iff it is equinumerous with a proper subset of itself. (Contrast this with the definition that a set is *infinite* if it is not equinumerous with any natural number.)

Theorem 7.2. *No natural number is Dedekind infinite.*

Exercise 7.1. Prove Theorem 7.2. (*Hint*: show that the set whose members are the natural numbers n such that every injective function from n to n is bijective is inductive.)

Corollary 7.3. *No finite set is Dedekind infinite.*

Exercise 7.2. Prove Corollary 7.3.

Corollary 7.4. *Every Dedekind infinite set is infinite.*

Exercise 7.3. Prove Corollary 7.4.

Is the converse of this corollary true? We will return to this question later in this chapter.

Corollary 7.5. *ω is infinite.*

[2] We can omit these because they have alternative decimal expansions, e.g. .7999... represents the same real number as .8000...

Proof. This follows from the preceding corollary (7.4) together with the fact that the successor function is a bijection from ω to $\omega \setminus \{0\}$ (see Proposition 6.4 and Theorem 6.10). □

Corollary 7.6. *No two distinct natural numbers are equinumerous.*

Exercise 7.4. Prove Corollary 7.6. (*Hint*: use the fact that the $\leq$ order on ω is connex (Corollary 6.17 and Theorem 6.18), together with the corollary above (7.5).)

Corollary 7.7. *Every finite set is equinumerous with a unique natural number.*

Exercise 7.5. Prove Corollary 7.7.

The unique natural number equinumerous with a finite set A is called the **cardinality** of A, written $|A|$.

Lemma 7.8. *If $C \subsetneq n \in \omega$, then $C \approx m$ for some $m < n$.*

Exercise 7.6. Prove Lemma 7.8. (*Hint*: show that the set whose members are those natural numbers n such that any proper subset of n is equinumerous to a member of n is inductive.)

Theorem 7.9. *Any subset of a finite set is itself finite.*

Proof. Suppose A is a subset of a finite set B. Let $n = |B|$, so there is a bijection $f : B \to n$. Then $f[A] \subseteq f[B] = n$. So either $f[A] = n$ or $f[A] \subsetneq n$. If $f[A] = n$, then $A \approx B$. If $f[A] \subsetneq n$, then by the previous lemma (7.8) $f[A] \approx m$ for some $m < n$. □

7.3 Domination, Countability, and Choice

We say a set A is **dominated** by a set B, written $A \preceq B$, iff there is an injection from A to B, or, equivalently, iff A is equinumerous with a subset of B. If $A \preceq B$ and $A \not\approx B$, A is said to be **strictly dominated** by B, written $A \precnsim B$ or $A \prec B$.

Exercise 7.7. Show that for any sets A, B, and C,

1. $A \preceq A$;
2. if $A \preceq B$ and $B \preceq C$ then $A \preceq C$; and

3. $A \preceq \wp(A)$.

Theorem 7.10 (Schröder-Bernstein)**.** *For any sets A and B, if $A \preceq B$ and $B \preceq A$, then $A \approx B$.*

We have the resources to prove this, but since the proof is rather involved, we postpone it to the appendix.

Before continuing, we need to add to our list of assumptions about sets again (remember our last new assumption was that there is a set whose members are the natural numbers). To state the new assumption, we first need a couple of definitions. First, if A is a set, then the **nonempty powerset** of A, written $\wp_{\text{ne}}(A)$, is just $\wp(A) \setminus \{\emptyset\}$, i.e. the set of nonempty subsets of A. And second, a **choice function** for A is a function $c : \wp_{\text{ne}}(A) \to A$ such that, for each nonempty subset B of A, $c(B) \in B$. The new assumption is this:

Assumption 9 (Choice)**.** There is a choice function for any set.

It has been proved (by Paul Cohen, in 1963) that Choice is *independent* of the other assumptions we have made, in the sense that, if in fact our other assumptions are consistent, then either one of Choice or its denial (that some set does not have a choice function) can be added without leading to inconsistency. But as a practical matter, most working mathematicians prefer to assume Choice, because there are so many useful theorems that cannot be proved without it. One such theorem is the following:

Theorem 7.11. *If A is infinite, then $\omega \preceq A$.*

The proof of this theorem is also deferred to appendix A.

Theorem 7.12 (Dedekind-Pierce)**.** *A set is infinite iff it is Dedekind infinite.*

Proof. The *only-if* part was proven above in Corollary 7.4. Now suppose A is infinite. Then $\omega \preceq A$, that is, there is an injection $f : \omega \to A$. Now define a bijection $g : A \to A \setminus \{f(0)\}$ as follows: if $a \in A$ is not in the range of f, then $g(a) = a$; and if a is in the range of f, so that $a = f(n)$ for some $n \in \omega$, then $g(a) = f(n+1)$. It is easy to see that g is injective and its range is $A \setminus \{f(0)\}$. □

A set is said to be **countable** if it is dominated by ω. An infinite countable set is called **denumerable**, **denumerably infinite**, or **countably infinite**. A set which is not countable is called **uncountable**, **nondenumerable**, or **nondenumerably infinite**.

Corollary 7.13. *Any countably infinite set is equinumerous with ω.*

Exercise 7.8. Prove Corollary 7.13.

Corollary 7.14. *Any infinite subset of ω is equinumerous with ω.*

Exercise 7.9. Prove Corollary 7.14.

Exercise 7.10. Prove that $\wp(\omega)$ is nondenumerable.

Some standard examples of countably infinite sets are the following: ω, $\omega \times \omega$, the positive natural numbers, the even natural numbers, $\mathbb{Z}$ (the integers), and $\mathbb{Q}$ (the rationals). The following sets can all be shown to be equinumerous with $\wp(\omega)$: $\mathbb{R}$ (the reals), the subset I of $\mathbb{R}$ consisting of the real numbers between 0 and 1 (including 0 and 1), $R \setminus Q$ (the irrationals), $R \times R$ (the plane), and ω^ω (infinite sequences of natural numbers).

Exercise 7.11. For any set A, the set of A-**strings**, written A^*, is defined to be $\bigcup_{n\in\omega} A^n$, and the domain of an A-string is called its **length**. Show that if A is nonempty and finite, then A^* is countably infinite. (*Hint*: this means showing there is a bijection from ω to A^*. In practical terms, that amounts to describing a way of listing all the A-strings without repetitions.)

Now consider the following statement:

Proposition 7.15 (Continuum Hypothesis)**.** *There is no set A such that $\omega \precapprox A \precapprox \wp(\omega)$.*

This hypothesis has the same status as the assumption of Choice: it can be proven to be independent of our other assumptions. The same is true of the following generalized form of the Continuum Hypothesis:

Proposition 7.16 (Generalized Continuum Hypothesis)**.** *For any infinite set B, there is no set A such that $B \precapprox A \precapprox \wp(B)$.*

Chapter 8

Varieties of Set Theory

Could there be a set A with $A \in A$? Nothing we have said rules it out. If you *want* to rule it out, here is one way:

Assumption 10 (Foundation)**.** Any nonempty set A has a member none of whose members is a member of A.

This rules out the existence of a set A such that $A \in A$. To see why, suppose that $B \in B$ and $A = \{B\}$. The only member of A is B, but B, by Assumption 10, has at least one member, namely B itself, which is a member of A. Foundation likewise rules out $C \in D \in C$, because of the following: letting $E = \{C, D\}$, both members of E contain a member of E.

More generally, assuming Foundation, there is no finite sequence $A_0 \cdots A_n$ with $A_{\mathbf{suc}(i)} \in A_i$, for some $i < n$, and $A_0 \in A_n$. For we could simply define A to be

$$\bigcup_{i<n} \{A_i\} = \{A_0, \ldots, A_n\} \ .$$

Likewise, there is no *infinite* sequence $f : \omega \to A$, for some set A, such that $f(\mathbf{suc}(n)) \in f(n)$ for every natural number n.

Exercise 8.1. Let A be any set. Show that there is no $f : \omega \to A$ such that $f(\mathbf{suc}(n)) \in f(n)$ for all $n \in \omega$.

None of the material in this book requires Foundation, and some theories (e.g. *situation semantics*) explicitly reject it. In fact, situation semantics uses a different assumption—called the **Antifoundation Axiom** or **AFA**—which is inconsistent with Foundation. We will discuss Antifoundation in due course (§8.2).

8.1 Zermelo-Fraenkel Set Theory and Variants

The most widely used variety of set theory is *Zermelo-Fraenkel set theory*, or *ZF*, which consists of the following axioms:

Extensionality	(our Assumption 1)
Empty set	(our Assumption 2)
Pairing	(our Assumption 3)
Union	(our Assumption 4)
Powerset	(our Assumption 5)
Separation	(our Assumption 6)
Natural Numbers	(our Assumption 7)
Replacement	(our Assumption 8)
Foundation	(our Assumption 10)

The Natural Numbers assumption is sometimes called the *Axiom of Infinity*; the Separation assumption is also known as the *Subset* assumption. And the Foundation assumption is also referred to as the *Axiom of Regularity*.

Another axiom we will consider is Choice (our Assumption 9). Like Foundation, Choice is independent of ZF (this fact was proved by Paul Cohen in 1963). Some popular combinations include the following.

ZF$^-$ ZF minus Foundation	(Assumptions 1–8)
ZFC ZF plus Choice	(Assumptions 1–10)
ZFC$^-$ ZF$^-$ plus Choice	(Assumptions 1–9)

The variant of ZF used in this book is ZFC$^-$. Situation semantics uses ZFC$^-$ plus Antifoundation, which we now introduce.

8.2 Antifoundation

8.2.1 Preliminaries

Definition 8.1 (Directed Graph)**.** A **directed graph** is an ordered pair $\mathcal{G} = \langle G, \longrightarrow \rangle$ where G is a set (the **nodes** of G) and $\longrightarrow \subseteq G \times G$ (the **edges** of G). So a directed graph is just a set with a binary relation.

Figure 8.1: Pictures of 2.

If $n \longrightarrow n'$, we say that n' is a **child** of n (in $\mathcal{G}$). A **path** from n to n' (in $\mathcal{G}$) is a string of nodes $n_0 \cdots n_m$ (for some $m \in \omega$) such that

1. For all $i < m$, $n_i \longrightarrow n_{\mathbf{suc}(i)}$,
2. $n_0 = n$, and
3. $n_m = n'$.

We say that n' is **accessible** from n (in $\mathcal{G}$) provided there is a path from n to n', or, equivalently, if $n \longrightarrow^+ n'$, recalling from §6.4 that $\longrightarrow^+$ is the transitive closure of $\longrightarrow$.

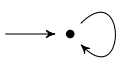
Definition 8.2 (Pointed Graph)**.** A **pointed graph** is an ordered pair $\langle \mathcal{G}, n \rangle$ where $\mathcal{G}$ is a directed graph $\langle G, \longrightarrow \rangle$ and $n \in G$. We call n the **point** of the pointed graph.

A pointed graph is called **accessible**, or an **apg** provided every node is accessible from n.

We can use apgs as pictures of sets. First, the intuition:

is a picture of 0,

is a picture of 1, and both of the pictures in Figure 8.1 are pictures of 2. If there were a set Ω where $\Omega = \{\Omega\}$, this would be a picture of it:



Now the formalities.

(a) Canonical picture of 2.

(b) A noncanonical picture of 2.

Figure 8.2: Decorated pictures of 2.

Definition 8.3 (Decoration)**.** A **decoration** of a graph $\mathcal{G} = \langle G, \longrightarrow \rangle$ is a function $\Delta : G \to A$, for some set A, such that for every node $n \in G$,

$$\Delta(n) = \left\{ x \in A \mid \exists n' (x = \Delta(n') \wedge n \longrightarrow n') \right\} ,$$

i.e. $\Delta(n)$ is the set of all the $\Delta(n')$ for n' a child of n.

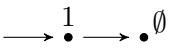
Definition 8.4 (Picture)**.** An apg $\mathcal{G} = \langle G, \longrightarrow, n \rangle$ is a **picture** of a set A iff there is a decoraction Δ of $\langle G, \longrightarrow \rangle$ such that $\Delta(n) = A$.

Examples The apg

is a picture of 0 because there is a decoration

Recalling that $1 = \{\emptyset\}$, the apg

is a picture of 1:



And recalling that $2 = \mathbf{suc}(1) = 1 \cup \{1\}$, the apgs in Figure 8.2 are (decorated) pictures of 2 (compare these with the apgs in Figure 8.1).

Theorem 8.1. *Every set has a picture.*

Proof. Let S be a set and $\mathcal{G} = \langle G, \longrightarrow, n \rangle$ the apg such that $G = \{S\} \cup \mathcal{T}(S)$ (recalling from §6.5 that $\mathcal{T}(S)$ is the transitive closure of S), $\longrightarrow = \{\langle x, y \rangle \in G \times G \mid y \in x\}$, and $n = S$. Let $A = G$ and $\Delta : G \to A$ such that $\Delta = \mathsf{id}_G$. Trivially Δ is a decoration of G with $\Delta(n) = S$, so by definition, $\mathcal{G}$ is a picture of S. □

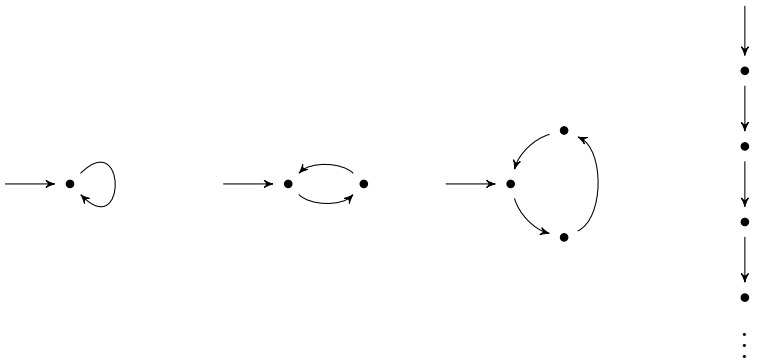


Figure 8.3: Apgs ruled out by Foundation.

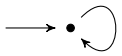
The picture of S constructed in the theorem is called the **canonical** picture of S.

A set can have more than one picture, e.g. the pictures of 2 in Figure 8.2. The picture in Figure 8.2a is the canonical picture of 2; the one in Figure 8.2b is noncanonical. What about the apgs in Figure 8.3? Foundation rules these out as pictures of sets. But suppose we reject Foundation and replace it with the following:

Assumption 11 (Antifoundation (AFA))**.** Every graph has a unique decoration.

Consequences of AFA:

- Every apg is a picture of a unique set.
- There are **nonwellfounded** sets, i.e. sets A such that there are infinite sequences $A_0, A_1, \ldots$ with $A_0 = A$ and $A_{\mathbf{suc}(n)} \in A_n$ for every $n \in \omega$.
- There is a set called Ω whose canonical picture is the following:



Exercise 8.2. What is the canonical picture of 3?

Exercise 8.3. Assuming AFA, give the canonical pictures of all sets $\{x, y\} = x \cup y$.

Exercise 8.4. Assuming AFA, prove that an apg is a picture of Ω iff every node has a child.

In situation semantics, nonwellfounded sets are used to model the meanings of paradoxical sentences like

> This sentence is false.

See Aczel 1988 for more discussion of nonwellfounded sets. One last point: we can never show that ZF^- plus AFA (or ZFC^- plus AFA) is consistent. But it can be shown that they are consistent if ZF is!

Chapter 9

Introduction to Formal Languages

It is a familiar and basic intuition that language somehow involves stringing things together. Examples include stringing phonemes together to form syllables or (phonologies of) morphemes, stringing morphemes together to form words, stringing words together to form phrases (including sentences), and stringing sentences together to form discourses. Indeed, in the early days of syntactic theory (early to mid 1950s), natural languages were modeled as sets of strings, and the notion of a grammar was identified with a mathematical device for listing the members of such sets. But what exactly is a string?

9.1 Strings

As we saw in Chapter 6, for every natural number n,

$$n = \{m \in \omega \mid m < n\} \ .$$

Let us now consider, for some set A and some $n \in \omega$, the set A^n, i.e. the set of arrows (functions with specified codomains) from n to A. The members of this set are called the A-**strings of length** n. In a linguistic application, we would think of the members of A as linguistic entities of some kind (phonemes, morphemes, words, etc.) that we would like to be able to string together, and of a particular A-string of length $n > 0$, f, as one of the possible results of stringing n such entities together, namely the one starting with $f(0)$, then $f(1)$, then $f(2)$, etc. If $f(i) = a_i$ for all $i < n$, then we usually denote f by the string (in the informal sense) of symbols $a_0 \cdots a_{n-1}$. (But in working with strings, it is important to remember that, technically, a string is not

really a bunch of symbols lined up from left to right on a page, but rather a function whose domain is a natural number.) Also, it's important to note that there is exactly one A-string of length 0, denoted by ϵ_A (or just ϵ when no confusion is possible).[1] The set of all A-strings of length greater than 0 is denoted by A^+.

For strings of length 1, a mild notational confusion arises: if $f : 1 \to A$ and $f(0) = a$, then the notation 'a' could refer to either a itself (a member of A), or to the length-one A-string f. It should be clear from context which is intended. Note also that an A-string of length one is the same thing as a nullary operation on A.

The 'infinite counterpart' of an A-string is called an **infinite A-sequence**; technically, an infinite A-sequence is a function from ω to A.

The set of all A-strings, i.e. the union of all the sets A^n, for all $n \in \omega$, is written A^*. Thus $A^* = A^+ \cup \{\epsilon_A\}$. When the identity of the set A is clear from context, we usually speak simply of strings, rather than A-strings. It should be obvious that there is a function from A^* to ω that maps each string to its length, and that the relation on strings of having the same length is an equivalence relation. Of course the sets in the partition induced by that equivalence relation are just the sets A^n. If A is a subset of another set B, then clearly there is an injection $\eta : A^* \to B^*$ that maps each A-string to a B-string just like it except that its codomain is B instead of A.

For each $n \in \omega$, there is an obvious bijection from $A^{(n)}$ to A^n. For $n \geq 2$, the bijection maps each n-tuple $\langle a_0, \ldots, a_{n-1} \rangle$ to the string $a_0 \cdots a_{n-1}$. For $n = 1$, it maps each $a \in A$ to the length-one string that maps 0 to a; and for $n = 0$, it is the (only) function from 1 to $\{\epsilon_A\}$, i.e. the function that maps 0 to ϵ_A.

The binary operation of **concatenation** on A^*, written $\frown$, can be described intuitively as follows: if f and g are strings, then $f \frown g$ is the string that 'starts with f and ends with g.' More precisely, for each pair of natural numbers $\langle m, n \rangle$, if f and g are strings of length m and n respectively, then $f \frown g$ is the string of length $m + n$ such that

1. $(f \frown g)(i) = f(i)$ for all $i < m$; and
2. $(f \frown g)(m + i) = g(i)$ for all $i < n$.

[1] In Chapter 3, we called this $\Diamond_A$, but the name ϵ_A is more usual when we are thinking of it as a string.

It can be proven inductively (though the details are quite tedious) that for any strings f, g, and h, the following equalities hold:[2]

$$(f \frown g) \frown h = f \frown (g \frown h)$$
$$f \frown \epsilon = f = \epsilon \frown f$$

Usually concatenation is expressed without the "$\frown$," by mere juxtaposition; e.g. fg for $f \frown g$. And because concatenation is an associative operation, we can write simply fgh instead of $f(gh)$ or $(fg)h$.

Exercise 9.1. Suppose A is finite and nonempty. Prove that the set of A-languages is nondenumerable. You can use any of the theorems or corollaries stated in Chapter 7, even if their proofs were not given.

9.2 Formal Languages

A **formal** (A-)**language** is defined to be a subset of A^*. But when it is clear that we are talking about formal languages rather than natural languages, we will usually just speak of an *A-language*, or simply a *language* if the identity of A is clear from the context. In the most straightforward application of formal languages to linguistics, we mathematically model a natural language as a set of A-strings, where A is a set each of whose members is (a representation of) one of the words of the natural language in question. Of course this is a very crude model, since it disregards any linguistic structure a sentence has other than the temporal sequence of the words themselves. Additionally, once one speaks of a sentence as a string of words, one is immediately faced with the question of what counts as a word, or, to put it another way, what criterion of identity one is using for words. Is it enough to be homophonous (i.e. to sound the same), so that *meat* and *meet* count as the same word? Or to be homographic (written the same), so that *row* 'linear array' and *row* 'fight' count as the same word? Or must two words have the same sound, meaning, and 'part of speech' (whatever we think that is), so that *murder* counts as two words (one a noun and one a verb)? We will return to these and related questions in later chapters.

[2] As we will see later, the truth of these equations means that A^* together with the nullary operation ϵ and the binary operation $\frown$ is an instance of a kind of algebra called a **monoid**; i.e.

1. $\frown$ is an *associative* operation, and
2. ϵ is a *two-sided identity* for $\frown$.

For the time being, we set such issues aside and assume we know what we mean by a 'word.' Assuming that, we can begin theorizing about questions such as the following: How many sentences (*qua* word strings) does the language have? Is there a way to list all its members? Is there a way to decide whether a given word string is a sentence of the language? Can we construct a plausible model of the process by which people who know the language recognize that a given string is a sentence of the language? Can the processing model somehow be extended to a model of how language users interpret utterances in context?

In order to address such questions, we need some techniques for defining formal languages. Since natural languages uncontroversially have an infinitude of sentences (how do you know?), it will not do to just make a list of A-strings. In due course we'll consider various kinds of **formal grammars**—mathematical systems for specifying formal languages—but we already have a powerful tool for doing just that, namely the Recursion Theorem (RT). One important way RT is used to specify an A-language L is roughly as follows: we start with

1. A set L_0 of A-strings which we know to be in the A-language we wish to define, and

2. A general method for adding more strings to any arbitrary set of strings, i.e. a function F from A-languages to A-languages.

We can think of L_0 as the 'dictionary' of the language we are trying to define and F as its 'rules.' We then define L as the union of the infinite sequence of languages $L_0, \ldots, L_n, \ldots$ where for each $k \in \omega$, L_{k+1} is the result of applying F to L_k.

We now give a simple example of a recursive definition for a language. Intuitively, a *mirror image* string in A is one whose second half is the reverse of its first half. Informally, we define the language $\mathrm{Mir}(A)$ as follows:

1. $\epsilon_A \in \mathrm{Mir}(A)$;

2. if $x \in \mathrm{Mir}(A)$ and $a \in A$, then $axa \in \mathrm{Mir}(A)$;

3. nothing else is in $\mathrm{Mir}(A)$.

Formally, this definition is justified by the RT as follows (here X, x, and F are as in the statement of RT in Chapter 6) we take X to be $\wp(A^*)$, x to be $\{\epsilon\}$, and $F : \wp(A^*) \to \wp(A^*)$ to be the function such that for any A-language S,

$$F(S) = \{y \in A^* \mid \exists a \exists x[a \in A \wedge x \in S \wedge y = axa]\} \ .$$

RT then guarantees the existence of a function $h : \omega \to \wp(A^*)$ such that $h(0) = \{\epsilon\}$ and for every $n \in \omega$, $h(\mathbf{suc}(n)) = F(h(n))$. Finally, we define $\mathrm{Mir}(A)$ to be $\bigcup_{n\in\omega} h(n)$. Intuitively, $h(n)$ is the set of all mirror image strings of length $2n$.

9.3 Operations on Languages

Let A be a set, so that A^* is the set of A-strings, $\wp(A^*)$ is the set of A-languages, and $\wp(\wp(A^*))$ is the set whose members are *sets* of A-languages.

We introduce the following notations for certain particularly simple A-languages:

1. For any $a \in A$, $\underline{a}$ is the singleton A-language whose only member is the string of length one a (remember this is the function from 1 to A that maps 0 to a).

2. $\underline{\epsilon}$ is the singleton A-language whose only member is the **null A-string** (i.e. the unique arrow from 0 to A, a. k. a. ϵ_A). An alternative notation for this language is l_A.

3. $\emptyset$ as always is just the empty set, but for any A we can also think of this as the A-language which contains no strings! An alternative notation for this language is 0_A.

Next, we define some operations on $\wp(A^*)$. In these definitions, L and M range over A-languages.

1. The **fusion** of L and M, written $L \bullet M$, is the set of all strings of the form $u \frown v$ where $u \in L$ and $v \in M$.

2. The **right residual** of L by M, written L/M, is the set of all strings u such that $u \frown v \in L$ for every $v \in M$.

3. The **left residual** of L by M, written $M \setminus L$, is the set of all strings u such that $v \frown u \in L$ for every $v \in M$.

4. The **Kleene closure** of L, written $\mathsf{kl}(L)$, has the following informal recursive definition (formalizing this definition will be the subject of exercise 9.2):

   (a) (*Base clause*) $\epsilon \in \mathsf{kl}(L)$;

   (b) (*Recursion clause*) if $u \in L$ and $v \in \mathsf{kl}(L)$, then $uv \in \mathsf{kl}(L)$; and

(c) nothing else is in $\mathsf{kl}(L)$.

To put it even less formally but more intuitively: the Kleene closure of L is the language whose members are those strings that result from concatenating together zero or more strings drawn from L.

5. The **positive Kleene closure** of L, written $\mathsf{kl}^+(L)$, has the following informal recursive definition:

   (a) (*Base clause*) If $u \in L$, then $u \in \mathsf{kl}^+(L)$;

   (b) (*Recursion clause*) if $u \in L$ and $v \in \mathsf{kl}^+(L)$, then $uv \in \mathsf{kl}^+(L)$; and

   (c) nothing else is in $\mathsf{kl}^+(L)$.

   Intuitively: the positive Kleene closure of L is the language whose members are those strings that result from concatenating together one or more strings drawn from L.

Exercise 9.2. Given an A-language L, use RT to formally define $\mathsf{kl}(L)$. More specifically, read the statement of RT and then tell how X, x, and F must be instantiated so that the function h whose existence is guaranteed by RT has the property that $\mathsf{kl}(L)$ is the union of all the languages $h(n)$, where $h(0)$ is x and for each $k \in \omega$, $h(k+1)$ is $F(h(k))$. (*Hints*: (a) The base clause in the informal definition should tell you what x has to be, and the recursion clause should tell you what F has to be. (b) Another informal, but perhaps more helpful way to define $\mathsf{kl}(L)$ is as the union of all the languages $\mathsf{kl}_n(L)$, where

- $\mathsf{kl}_0(L) = \mathsf{l}_A$; and
- $\mathsf{kl}_{k+1}(L) = L \bullet \mathsf{kl}_k(L)$ for all $k \in \omega$.)

Exercise 9.3. In *propositional logic* (PL), we have

1. A formal language PL whose strings are called *(PL-)formulas*, together with
2. A *semantics*, i.e. a function from PL to some other set whose members, called *propositions*, are thought of as the meanings expressed by the formulas, and
3. A *proof theory*, which is a precise way of formalizing what it means for one formula to *follow from* one or more other formulas.

For now we're concerned only with the language PL itself. We assume given a set Let whose members are usually called *propositional letters*. Informally, the set of formulas is defined as follows:

1. (The length-one string corresponding to) each propositional letter is a formula;
2. (The length-one strings corresponding to) $\top$ and $\bot$ are formulas;
3. If P and Q are formulas, so are $(\sim P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \supset Q)$, and $(P \leftrightarrow Q)$; and
4. Nothing else is a formula.

It's important to be aware that what we are talking about here are formulas (which are strings), and *not* the propositions (whatever *they* are) that the formulas express. For example, in the informal definition, when we write "$(\sim P)$" we mean the string obtained by concatenating together the following symbols in the specified order:

1. The left-paren symbol (;
2. The negation symbol $\sim$;
3. The symbols in the string P; and finally
4. The right paren symbol).

The problem is to define PL formally, using RT. As you should realize by now, this means telling how X, x, and F in the statement of RT should be instantiated. To keep things simple, suppose the only propositional letters are X, Y, and Z. (*Hints*: (a) Clauses 1 and 2 are the bases clauses and clause 3 is the recursion clause. (b) After you think you have the right definition, check to make sure it makes $((X \wedge Y) \wedge Z)$ a formula. If it doesn't, you'll need to revise your definition.)

9.4 Regular Languages

Linguists are often concerned not just with languages, but with *sets* of languages, e.g. the set of finite languages, the set of **decidable** languages (languages for which an algorithm exists that tells for any given string whether it is in the language), the set of **recursively enumerable languages** (languages for which an algorithm exists for listing all its strings while not

listing any strings not in the language), etc. In computational linguistics applications, one of the most important sets of languages is (for a fixed alphabet A) the set $\mathsf{Reg}(A)$ of **regular** A-languages. As with many other important sets of languages, there are several different ways to define this set, all of which give the same result. For our purposes, the simplest way is a recursive definition. The informal version runs as follows:

1. For each $a \in A$, $\underline{a} \in \mathsf{Reg}(A)$;
2. $\mathsf{0}_A \in \mathsf{Reg}(A)$;
3. $\mathsf{I}_A \in \mathsf{Reg}(A)$;
4. For each $L \in \mathsf{Reg}(A)$, $\mathsf{kl}(L) \in \mathsf{Reg}(A)$;
5. For each $L, M \in \mathsf{Reg}(A)$, $L \cup M \in \mathsf{Reg}(A)$;
6. For each $L, M \in \mathsf{Reg}(A)$, $L \bullet M \in \mathsf{Reg}(A)$; and
7. Nothing else is in $\mathsf{Reg}(A)$.

Note that in this definition, the first three clauses are base clauses and the next three are recursion clauses.

Exercise 9.4. Formalize this definition of $\mathsf{Reg}(A)$ using RT. (*Hint*: remember that we are defining not a language, but rather a set of languages, and therefore the choice of X (as in the statement of RT in Chapter 6) is not $\wp(A^*)$, but rather $\wp(\wp(A^*))$).

Exercise 9.5. In *autosegmental-metrical* (AM) theory, the phonological structure of the intonation of an English expression is represented by a string of **tones**.[3] (The question of how such a string is realized as a 'tune' (pitch contour) is a matter of phonetics, which doesn't concern us here.)

In one version of this theory, the set of tones has eleven members:

$$A = \left\{\mathsf{H}^*, \mathsf{L}^*, \mathsf{L}^* + \mathsf{H}, \mathsf{L} + \mathsf{H}^*, \mathsf{H}^-, \mathsf{L}^-, \mathsf{H}\%, \mathsf{L}\%\right\}$$

Note that each of these eleven elements is to be thought of as a single member of the tonal 'alphabet,' no matter how many symbols it is written with. Of

[3] The sense of the term *tone* here is 'minimal unit of intonational phonology.' This is distinct from the notion of *tone* in the sense of pitch levels or contours within words that function as part of the word phonology and distinguish words from each other, e.g. Mandarin *ma* [high level] 'mother' vs. *ma* [high-falling-to-low] 'scold'.

these, the first four are called **pitch accents** (PAs),[4] the next two are called **intermediate-phrase final boundary tones** (IFBTs); and the last two are called **final boundary tones** (FBTs).

Certain sets of tone strings are defined informally as follows:

1. An **intermediate phrase** consists of one or more PAs followed by an IFBT.
2. An **intonation phrase** consists of one or more intermediate phrases followed by an FBT.
3. An **utterance** consists of one or more intonation phrases.

The problem is to *formally* define each of the following as regular languages:

1. The language IP of intermediate phrases;
2. The language IN of intonation phrases; and
3. The language UT of utterances.

You do not need to prove anything, you do not need to use RT, and you do not need to use PMI. All you need to do is express each of the these three languages in terms of

1. The eight tones;
2. The underscore (that maps each tone to the singleton language whose only string is the length-one string of that tone);
3. The empty language $\emptyset$;
4. the singleton language I whose only member is the null string; and
5. The operations union ($\cup$) of languages, fusion ($\bullet$) of languages, Kleene closure (kl), and positive Kleene closure (kl^{+}).

Note: the fact that this is possible means that these languages are all regular. (*Hint*: You will not necessarily need to use all the available operations. Parentheses cannot be used to express optionality; you will have to express it some other way.)

[4]The asterisks in the notation for the pitch accents have nothing to do with the asterisk of formal language theory. Instead, they designate a tone (or part of a tone) that has to be associated with an accented syllable.

9.5 Context-free Languages

Context-free grammars (CFGs) are a particular way of defining languages recursively that is very widely used in syntactic theory; in one form or another, CFGs play a central role in a wide range of syntactic frameworks (here 'framework' means, roughly, a research paradigm or community), including, to name just a few, all forms of transformational grammar (TG); many kinds of categorial grammar (CG); lexical-functional grammar (LFG); generalized phrase structure grammar (GPSG); and head-driven phrase structure grammar (HPSG). In due course it will emerge that CFGs are a rather blunt instrument for modeling natural languages, but they are a good point of departure in the sense that they can be elaborated, refined, and adapted in many ways (some of which we will examine closely) that make them more suitable for this task.

9.5.1 Intuitions

The basic idea behind CFGs is to *simultaneously* recursively define a finite set of different languages, each of which constitutes a set of strings that have the same 'distribution' or 'privileges of occurrence' or 'combinatory potential' in the whole language being defined, which is the union of that set of languages. The languages in that family are called the **syntactic categories** of the whole language.

Getting technical, a CFG consists of four things:

1. A finite set T whose members are called **terminals**;
2. A finite set N whose members are called **nonterminals**;
3. A finite set D of ordered pairs called **lexical entries**, each of which has a nonterminal as its left component and a terminal as its right component,[5] and
4. A finite set P of ordered pairs called **phrase structure rules** (or simply PSRs), each of which has a nonterminal as its left component and a non-null string of nonterminals as its right component.[6]

[5] Formal language theorists usually allow any T-string as the right component of a lexical entry, but we will not need this generality for our applications.

[6] Formal language theorists usually allow any $(N \cup T)$-string containing at least one nonterminal as the right component of a PSR, but again this generality goes beyond the needs of our linguistic applications.

Intuitively, the terminals are the words (or word phonologies, or word orthographies—see above) of the language under investigation. The nonterminals are names of the syntactic categories. The lexical entries make up the dictionary (or lexicon) of the language. And the PSRs provide a mechanism for telling which strings (other than length-one strings of words) are in the language and what syntactic categories they belong to. Once all this is made more precise, the CFG will specify, for each nonterminal A, a T-language C_A, and the language defined by the CFG will be the union over all $A \in N$ of the C_A. We'll make all this precise in two stages, first using an informal recursive definition (the usual kind), and then a more informal or 'official' definition employing the Recursion Theorem (RT).

9.5.2 Informal Definition

First, the informal version. As with all recursive definitions, a CFG has a base part and a recursion part. The base part makes use of the lexicon D and the recursion part uses the set P of PSRs. Starting with the lexicon, remember that formally a lexical entry is an ordered pair $\langle A, t\rangle \in D \subseteq N \times T$; but formal language theorists usually write entries in the form

$$A \to t$$

to express that $\langle A, t\rangle \in D$. In the informal recursive definition, the significance of a lexical entry expressed as follows:

> If $A \to t$, then $t \in C_A$.

That is: for any terminal a which the dictionary pairs with the nonterminal A, the string a of length one will be in the category which that nonterminal names.

Note that it is conventional to abbreviate sets of lexical entries with the same left-hand side using curly brackets on the right-hand side, e.g.

$$A \to \{t_1, t_2\}$$

abbreviates

$$A \to t_1$$
$$A \to t_2\,.$$

As mentioned above, the recursive part of the (informal) recursive definition draws on the set P of PSRs. Technically, a PSR is an ordered pair

$\langle A, A_0, \ldots, A_{n-1} \rangle \in P \subseteq N \times N^+$, but formal language theorists usually write

$$A \to A_0 \cdots A_{n-1}$$

to express that $\langle A, A_0, \ldots, A_{n-1} \rangle \in P$. In the informal recursive definition, the significance of a PSR is expressed this way:

> If $A \to A_0 \cdots A_{n-1}$ and for each $i < n$, $s_i \in C_{A_i}$, then $s_0 \cdots s_{n-1} \in C_A$.

That is: if, for each nonterminal on the right-hand-side of some rule, we have a string belonging to the category named by that nonterminal, then the result of concatenating together all those strings (in the same order in which the corresponding nonterminals appear in the rule) is a member of the category named by the nonterminal on the left-hand side of the rule.

As with lexical entries, sets of rules with the same left-hand side can be abbreviated using curly brackets on the right-hand side.

Before going on to the formal, RT-based formulation of CFGs, we illustrate the informal version with a 'toy' (i.e. ridiculously simplified) linguistic example.

$T = \{\mathsf{Fido}, \mathsf{Felix}, \mathsf{Mary}, \mathsf{barked}, \mathsf{bit}, \mathsf{gave}, \mathsf{believed}, \mathsf{the}, \mathsf{cat}, \mathsf{dog}, \mathsf{yesterday}\}$
$N = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{TV}, \mathrm{DTV}, \mathrm{SV}, \mathrm{Det}, \mathrm{N}\}$

D consists of the following lexical entries:

$$\begin{aligned}
\mathrm{NP} &\to \{\mathsf{Fido}, \mathsf{Felix}, \mathsf{Mary}\} \\
\mathrm{VP} &\to \mathsf{barked} \\
\mathrm{TV} &\to \mathsf{bit} \\
\mathrm{DTV} &\to \mathsf{gave} \\
\mathrm{SV} &\to \mathsf{believed} \\
\mathrm{Det} &\to \mathsf{the} \\
\mathrm{N} &\to \{\mathsf{cat}, \mathsf{dog}\} \\
\mathrm{Adv} &\to \mathsf{yesterday}
\end{aligned}$$

P consists of the following PSRs:

$$\begin{aligned}
\mathrm{S} &\to \mathrm{NP\ VP} \\
\mathrm{VP} &\to \{\mathrm{TV\ NP}, \mathrm{DTV\ NP\ NP}, \mathrm{SV\ S}, \mathrm{VP\ Adv}\} \\
\mathrm{NP} &\to \mathrm{Det\ N}
\end{aligned}$$

In this grammar, the nonterminals are names for the syntactic categories of noun phrases, verb phrases, transitive verbs, sentential-complement verbs, ditransitive verbs, determiners, and common noun phrases.[7] The lexical entries tell us, for example, that Felix (the length-one word string, not the word itself) is a member of the syntactic category C_{NP}, and the PSRs tell us, for example, that the string that results from concatenating two strings, one belonging to the syntactic category C_{NP} (e.g. Felix) and the other belonging to the syntactic category C_{VP} (e.g. barked), in that order (in this case, the length-two string Felix barked), belongs to the syntactic category C_{S}.

9.5.3 Spelling It Out Formally Using Simultaneous Recursion

Finally, we show how to formalize the simultaneous recursive definition of the syntactic categories associated with a CFG, using the RT. As always when applying the RT, the key is making the right choice for the three pieces of data X, x, and F. Since we are defining not a language but rather a function from nonterminals to languages, the right choice for X is not $\wp(T^*)$ but rather $\wp(T^*)^N$; x will be a member of this set, and F will be a function from this set to itself.

So what is x? Intuitively, it should tell us, for each nonterminal A, which strings are in the syntactic category C_A by virtue of the lexicon alone, i.e. without appealing to the recursive part of the definition (the PSRs). That is, x is the function that maps each nonterminal A to the set of strings t (all of which will have length one) such that $A \to t$ is one of the lexical entries.

What about F? What should be the result of applying F to an arbitrary function $L : N \to \wp(T^*)$? Well, for each $A \in N$, we will want $F(L)(A)$ to contain all the strings that were in $L(A)$, together with any strings that can be obtained by applying a rule of the form $A \to A_0 \cdots A_{n-1}$ to strings $s_0 \cdots s_{n-1}$, where, for each $i < n$, s_i belongs to the language that L assigned to A_i. Another way to say this is that F maps each L to the function that maps each nonterminal A to the language which is the union of the following two languages:

1. $L(A)$, and

[7] The category names are a bit confusing, since the categories of noun phrases, verb phrases, and common noun phrases are allowed to contain length-one strings (intuitively, words).

2. The union, over all rules of the form $A \to A_0 \cdots A_{n-1}$, of the languages $L(A_0) \bullet \cdots \bullet L(A_{n-1})$.

Given these values of X, x, and F, the RT guarantees us a unique function h from ω to functions from N to $\wp(T^*)$. Finally, for each nonterminal A, we define the corresponding syntactic category to be

$$C_A =_{\text{def}} \bigcup_{n \in \omega} h(n)(A) \, .$$

Exercise 9.6. Calculate, for as many values of n as you have patience for, and for each nonterminal A, the value of $h(n+1)(A) \setminus h(n)(A)$ (that is, the set of strings that are added to C_A at the nth recursive step).

Exercise 9.7. Let T be a finite set. We say a T-language L is **context-free** if there exists a CFG $\langle T, N, D, P \rangle$ such that L is one of its syntactic categories, i.e. $L = C_A$ for some $A \in N$.

1. Prove that $L = \mathsf{Mir}(2)$ is context-free by presenting a CFG which has L as one of its syntactic categories.

2. Same problem, but with $L = \mathrm{PL}$, the language of propositional logic with three propositional letters in exercise 9.3.

Exercise 9.8. Recall that if $\langle T, N, D, P \rangle$ is a CFG, then, in the informal style of recursive definition, the syntactic categories are defined as follows:

1. (*Base Clause*) If $A \to t \in D$, then $t \in C_A$.

2. (*Recursion Clause*) If $A \to A_0 \cdots A_{n-1} \in P$ and for each $i < n$, $s_i \in C_{A_i}$, then $s_0 \cdots s_{n-1} \in C_A$.

Now using the toy English grammar discussed above, prove that $\mathsf{Mary\ believed\ the\ cat\ bit\ the\ dog} \in C_{\mathrm{S}}$.

Chapter 10

Trees

10.1 Informal Motivation

As we will illustrate presently, given a CFG $\langle T, N, D, P\rangle$, a nonterminal $A \in N$, and a T-string $s \in C_A$, we can use the CFG to guide us in constructing a proof that $s \in C_A$. In fact, as anyone who has taken a course in formal language theory or computational linguistics will already realize, there are general procedures for deciding, given *any* CFG $\langle T, N, D, P\rangle$, *any* T-string s and *any* nonterminal A, whether or not $s \in C_A$. Such a procedure is called a **recognizer** because it tells, in effect, whether the CFG recognizes a given string as a member of a given syntactic category. In order to decide correctly that $s \in C_A$, the recognizer essentially must construct a *proof* that this is the case. What about making the correct decision when $s \notin C_A$? For that to be possible, the recognizer must in some sense 'know' when it has gotten to the point where, had there been a proof that $s \in C_A$, it would have found one; at that point it would render a negative decision. A **parser**, roughly speaking, is just a recognizer which renders not merely a decision but also (symbolic representations of) the proofs (if any) upon which the decision was based.

The construction of recognizers and parsers for CFGs and other kinds of formal grammars, one of the central concerns of both formal language theorists and of computational linguists, is a very highly evolved and subtle discipline, which unfortunately is beyond the scope of this book. However, the fundamental distinction between a parser and a (mere) recognizer has an analog that is relevant even for empirical/theoretical linguists (as opposed to formal language theorists and computational linguists), namely the intuition that a sentence is not just a string of words that belongs to C_{S} but rather a

way that the string in question belongs to C_{S}. To take a very simple example, let's consider a slightly expanded version of the toy English grammar in Chapter 9, as follows:

$$T = \{\mathsf{Fido},\ \mathsf{Felix},\ \mathsf{Mary},\ \mathsf{barked},\ \mathsf{bit},\ \mathsf{gave},\ \mathsf{believed},\ \mathsf{heard},\ \mathsf{the},\ \mathsf{cat},\ \mathsf{dog},\ \mathsf{yesterday}\}$$
$$N = \{\mathrm{S},\ \mathrm{NP},\ \mathrm{VP},\ \mathrm{TV},\ \mathrm{DTV},\ \mathrm{SV},\ \mathrm{Det},\ \mathrm{N},\ \mathrm{Adv}\}$$

D consists of the following lexical entries:

$$\begin{aligned}
\mathrm{NP} &\rightarrow \{\mathsf{Fido},\ \mathsf{Felix},\ \mathsf{Mary}\}\\
\mathrm{VP} &\rightarrow \mathsf{barked}\\
\mathrm{TV} &\rightarrow \mathsf{bit}\\
\mathrm{DTV} &\rightarrow \mathsf{gave}\\
\mathrm{SV} &\rightarrow \{\mathsf{believed},\ \mathsf{heard}\}\\
\mathrm{Det} &\rightarrow \mathsf{the}\\
\mathrm{N} &\rightarrow \{\mathsf{cat},\ \mathsf{dog}\}\\
\mathrm{Adv} &\rightarrow \mathsf{yesterday}
\end{aligned}$$

P consists of the following PSRs:

$$\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{NP\ VP}\\
\mathrm{VP} &\rightarrow \{\mathrm{TV\ NP},\ \mathrm{DTV\ NP\ NP},\ \mathrm{SV\ S},\ \mathrm{VP\ Adv}\}\\
\mathrm{NP} &\rightarrow \mathrm{Det\ N}
\end{aligned}$$

The only additions are

1. The nonterminal Adv (adverb);
2. The terminals heard and yesterday;
3. The lexical entries for yesterday as an adverb and for heard as a sentential-complement verb; and
4. the PSR VP $\rightarrow$ VP Adv.

Now consider the string $s =$ Mary heard Fido bit Felix yesterday. According to our grammar, $s \in C_{\mathrm{S}}$ (the syntactic category of sentences), but few (if any) syntacticians would say that s is an English sentence! Rather, they would say that the word string s corresponds to *two different sentences*,

one roughly paraphrasable as *Mary heard yesterday that Fido bit Felix* and another roughly paraphrasable as *Mary heard that yesterday, Fido bit Felix*. Of course, these two sentences mean different things; but more relevant for our present purposes is that we can also characterize the difference between the two sentences purely in terms of two distinct ways of *proving* that $s \in C_{\mathrm{S}}$.

To understand this point, remember from Chapter 9 that the set of syntactic categories is (informally) defined by simultaneous recursive definition as follows:

1. (*Base Clause*) If $A \to t$, then $t \in C_A$.
2. (*Recursion Clause*) If $A \to A_0 \cdots A_{n-1}$ and for each $i < n$, $s_i \in C_{A_i}$, then $s_0 \cdots s_{n-1} \in C_A$.

Then the two proofs run as follows:

First Proof. From the lexicon and the base clause, we know that Mary, Fido, Felix $\in C_{\mathrm{NP}}$, heard $\in C_{\mathrm{SV}}$, bit $\in C_{\mathrm{TV}}$, and yesterday $\in C_{\mathrm{Adv}}$. Then, by repeated applications of the recursion clause, it follows that:

1. Since bit $\in C_{\mathrm{TV}}$ and Felix $\in C_{\mathrm{NP}}$, bit Felix $\in C_{\mathrm{VP}}$;
2. Since bit Felix $\in C_{\mathrm{VP}}$ and yesterday $\in C_{\mathrm{Adv}}$, bit Felix yesterday $\in C_{\mathrm{VP}}$;
3. Since Fido $\in C_{\mathrm{NP}}$ and bit Felix yesterday $\in C_{\mathrm{VP}}$, Fido bit Felix yesterday $\in C_{\mathrm{S}}$;
4. Since heard $\in C_{\mathrm{SV}}$ and Fido bit Felix yesterday $\in C_{\mathrm{S}}$, heard Fido bit Felix yesterday $\in CP_{\mathrm{VP}}$; and finally,
5. Since Mary $\in C_{\mathrm{NP}}$ and heard Fido bit Felix yesterday $\in C_{\mathrm{VP}}$, Mary heard Fido bit Felix yesterday $\in C_{\mathrm{S}}$.

□

Second Proof. The same as the first proof, up through step 1. From there, we proceed as follows:

2. Since Fido $\in C_{\mathrm{NP}}$ and bit Felix $\in C_{\mathrm{VP}}$, Fido bit Felix $\in C_{\mathrm{S}}$;
3. Since heard $\in C_{\mathrm{SV}}$ and Fido bit Felix $\in C_{\mathrm{S}}$, heard Fido bit Felix $\in C_{\mathrm{VP}}$;
4. Since heard Fido bit Felix $\in C_{\mathrm{VP}}$ and yesterday $\in C_{\mathrm{Adv}}$, heard Fido bit Felix yesterday $\in C_{\mathrm{VP}}$; and finally,

5. Since Mary $\in$ C_{NP} and heard Fido bit Felix yesterday $\in$ C_{VP}, Mary heard Fido bit Felix yesterday $\in$ C_{S}.

□

There is nothing complicated about any of this, but this is not how a syntactician would usually describe the difference between the two homophonous sentences. Instead, s/he would draw two different *tree diagrams* as in Figures 10.1 and 10.2.

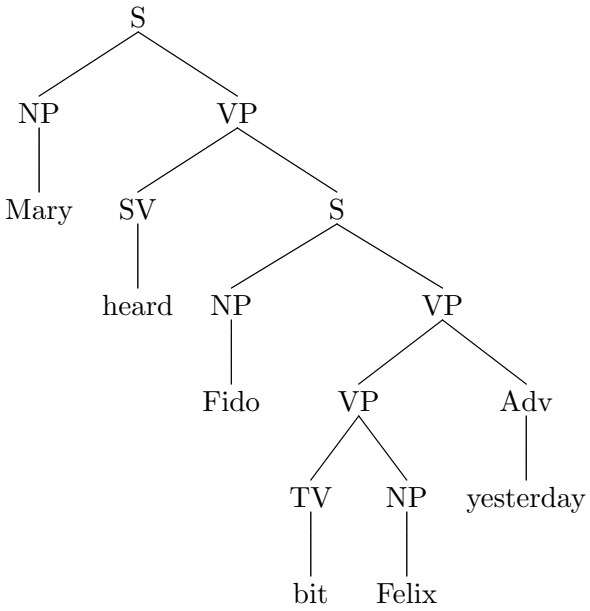


Figure 10.1: Tree diagram with 'low' adverb attachment.

But what exactly *are* tree diagrams, and what is supposed to be their relationship to the linguistic phenomena being theorized about? Well, roughly speaking (we will get more precise in the following two sections), tree diagrams are elaborated Hasse diagrams of mathematical objects called **labeled trees**. And what are labeled trees? Well, **trees** are partially ordered sets of a certain kind, and a labeled tree is a tree together with a function that assigns things called **labels** to the members (called **nodes**) of the partially ordered set. When syntacticians use labeled trees, the labels assigned to the minimal nodes are drawn from the set T of terminals and the labels assigned to the other nodes are drawn from the set N of nonterminals.

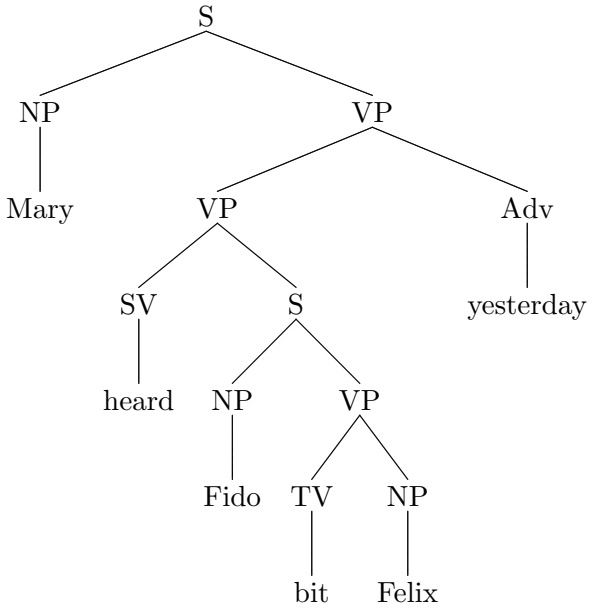


Figure 10.2: Tree diagram with 'high' adverb attachment.

Intuitively, it is pretty clear that these two tree diagrams are closely related to, or in some sense correspond to, the two proofs given earlier (though the precise relationship remains quite obscure at this stage). But when we begin to construct a linguistic theory, which should we use? Should we use labeled trees or elaborations of them as set-theoretic idealizations of sentences, as is done in syntactic frameworks such as HPSG (head-driven phrase structure grammar) and LFG (lexical-functional grammar)? Or is it better to think of a sentence as a proof that a certain string belongs to a certain syntactic category, as is done in CG (categorial grammar)? Or is it perhaps best to use a hybrid approach with both set-theoretic and proof-theoretic aspects, such as most forms of transformational grammar (TG), which include MP (the Minimalist Program) and its predecessor GB (Government-Binding)? We will not be able to answer these questions until we start to formalize logic and look at how formal logic is applied to linguistic theory. But because tree representations are so widely used by syntacticians (not to mention semanticists, computational linguists, and logicians), it is important for us to get clear early on precisely what trees are and how syntacticians use them.

10.2 Trees

10.2.1 Technical Preliminaries

Here we gather together some facts that will simplify the discussion of trees.

Theorem 10.1. *Any nonempty finite order has a minimal (and so, by duality, a maximal) member.*

Proof sketch. Let T be the set of natural numbers n such that every ordered set of cardinality $n+1$ has a minimal member, and show that T is inductive. The main idea of the proof is to show that T is an inductive set. □

Exercise 10.1. Complete the proof of Theorem 10.1 by showing that T is inductive.

Corollary 10.2. *Any nonempty finite chain has a least (and so, by duality, a greatest) member.*

Proof. This follows from the fact (itself a simple consequence of connexity) that in a chain, a member is least (greatest) iff it is minimal (maximal). □

Theorem 10.3. *For any natural number n, any chain of cardinality n is order-isomorphic to the usual order on n (i.e. the restriction to n of the usual $\leq$ order on ω).*

Proof sketch. By induction on n. The case $n = 0$ is trivial. By inductive hypothesis, assume the statement of the theorem holds for the case $n = k$ and let A of cardinality $k+1$ be a chain with order $\sqsubseteq$. By the Corollary (10.2), A has a greatest member a, so there is an order isomorphism f from k to $A \setminus \{a\}$. The rest of the proof consists of showing that $f \cup \{\langle k, a\rangle\}$ is an order isomorphism. □

Exercise 10.2. Complete the proof of Theorem 10.3 by showing that $f \cup \{\langle k, a\rangle\}$ is an order isomorphism.

Theorem 10.4. *Suppose $\sqsubseteq$ is an order on a finite set A. Then $\sqsubseteq = \prec^*$.*

That is: a finite order is the reflexive transitive closure of its own covering relation.

Proof. That $\prec^* \subseteq \sqsubseteq$ follows easily from the definition of reflexive transitive closure and the the transitivity of $\sqsubseteq$. To prove the reverse inclusion, suppose $a \sqsubseteq b$ and let X be the (nonempty, finite) set of all subsets of A which, when

ordered by $\sqsubseteq$, are chains with b as greatest member and a as least member. (X is nonempty since one of its members is $\{a, b\}$.) Then X itself is ordered by $\subseteq_X$, and so by Theorem 10.1 has a maximal member C. Let $n+1$ be $|C|$; by Theorem 10.3, there is an order-isomorphism $f : n+1 \to C$. Clearly $n > 0$, $f(0) = a$, and $f(n) = b$. Also, for each $m < n$, $f(m) \prec f(m+1)$, because otherwise, there would be a c properly between $f(m)$ and $f(m+1)$, contradicting the maximality of C. □

10.2.2 Trees

We now define a *tree* is to be a finite set A with an order $\sqsubseteq$ and a top $\top$, such that the covering relation $\prec$ is a function with domain $A \setminus \{\top\}$. In the linguistic community, the following terminology for trees is standard:

1. The members of A are called the **nodes** of the tree.
2. $\top$ is called the **root**.
3. If $x \sqsubseteq y$, y is said to **dominate** x; and if additionally $x \neq y$, then y is said to **properly dominate** x.
4. If $x \prec y$, then y is said to **immediately dominate** x. In that case $y = \prec(x)$ is called the **mother** of x, and x is said to be a **daughter** of y.
5. Distinct nodes with the same mother are called **sisters**.
6. A minimal node (i.e. one with no daughters) is called a a **terminal** node.
7. A node which is the mother of a terminal node is called a **preterminal** node.

We state here some important facts about trees, sketching some of the proofs and leaving others as exercises.

Theorem 10.5. *No node can dominate one of its sisters.*

Exercise 10.3. Prove Theorem 10.5.

Theorem 10.6. *For any node a, $\uparrow a$ is a chain.*

Proof sketch. Use the RT to define a function $h : \omega \to A$, with $X = A$, $x = a$, and F the function which maps non-root nodes to their mothers and the root to itself. Now let $Y = \mathsf{ran}(h)$; it is easy to see that Y is a chain, and that $Y \subseteq \uparrow a$. It remains to show that $\uparrow a \subseteq Y$. So assume $b \in \uparrow a$; we have to show $b \in Y$.

By definition of $\uparrow a$, $a \sqsubseteq b$, and so by Theorem 10.4 of section 10.2, $a \prec^* b$. From this and the definition (§6.4) of reflexive transitive closure, it follows that there is a natural number n such that $a \prec_n b$, where $\prec_n$ is the n-fold composition of $\prec$ with itself. In other words, there is an A-string $a_0 \cdots a_n$ such that $a_0 = a$, $a_n = b$, and for each $k < n$, $a_k \prec a_{k+1}$. But then $b = h(n)$, so $b \in Y$. □

Corollary 10.7. *Two distinct nodes have a meet iff they are comparable.*

Exercise 10.4. Prove Corollary 10.7.

Theorem 10.8. *Any two nodes have a lub.*

Exercise 10.5. Prove Theorem 10.8.

10.2.3 Ordered Trees

An **ordered tree** is a set A with *two* orders $\sqsubseteq$ and $\leq$, such that the following three conditions are satisfied:

1. A is a tree with respect to $\sqsubseteq$.
2. Two distinct nodes are $\leq$-comparable iff they are not $\sqsubseteq$-comparable.
3. (*No-tangling condition*) If a, b, c, d are nodes such that $a < b$, $c \prec a$, and $d \prec b$, then $c < d$.

In an ordered tree, if $a < b$, then a is said to **linearly precede** b.

Theorem 10.9. *If a is a node in an ordered tree, then the set of daughters of a ordered by $\leq$ is a chain.*

Exercise 10.6. Prove Theorem 10.9.

Theorem 10.10. *In an ordered tree, the set of terminal nodes ordered by $\leq$ is a chain.*

Exercise 10.7. Prove Theorem 10.10.

10.3 Trees in Syntax

In so-called *phrase-structural* approaches to natural language syntax, a CFG $\langle T, N, D, P\rangle$ is used to define a set of **phrase structure trees**. These are just ordered trees equipped with a **labeling** function l from the set of nodes to $T \cup N$ such that, for each node a,

$$\begin{array}{ll} \mathsf{l}(a) \in T & \text{if } a \text{ is a terminal node, and} \\ \mathsf{l}(a) \in N & \text{otherwise.} \end{array}$$

We then speak of the set of phrase structure trees **generated by**, or **licensed by**, or **admitted by** the CFG, defined below.

The Set of Phrase Structure Trees Admitted by a CFG A phrase structure tree is **generated** by the CFG $G = \langle T, N, D, P\rangle$ iff

1. For each preterminal node with label A and (terminal) daughter with label t, $A \to t \in D$; and
2. For each nonterminal nonpreterminal node with label A and linearly ordered (as per Theorem 10.9) daughters with labels $A_0 \cdots A_{n-1}$ respectively, $(n > 0)$, $A \to A_0 \cdots A_{n-1} \in P$.

Additionally, for a phrase structure tree with linearly ordered (as per Theorem 10.10) set of terminal nodes $a_0 \cdots a_{n-1}$ with labels $t_0 \cdots t_{n-1}$ respectively, the string $t_0 \cdots t_{n-1}$ is called the **terminal yield** of the phrase structure tree. The **strong generative capacity** of G is the set of phrase structures that it generates. The **weak generative capacity** of G is the function $\mathsf{wgc} : N \to T^*$ that maps each nonterminal symbol A to the set of T-strings which are terminal yields of phrase structures generated by G with root label A.

Part II

Proof Theory

Chapter 11

Linear Propositional Logic and Natural Deduction

Many different kinds of logic can be directly applied to formalizing theories in syntax, phonology, semantics, pragmatics, and computational linguistics. Some examples include:

- linear logic
- Lambek calculus (intuitionistic bilinear logic)
- intuitionistic propositional logic
- (simply) typed lambda calculus
- dependent type theories
- higher-order logic

This chapter introduces linear logic (Girard, 1987), with Lambek calculus, (both intuitionistic and classical) propositional logic and first-order extensions discussed in Chapters 12 and 13, respectively. Variants of type theory, including typed lambda calculus, dependent type theory, and higher-order logic, are introduced in Part **??**.

To explain linear logic and the logics in subsequent chapters, we first introduce a kind of **proof theory** called **(Gentzen-sequent-style) natural deduction**, or **ND** for short.

11.1 Proof theory

Proof theory is the part of logic concerned with purely *syntactic* methods for determining whether a **formula** is **deducible** from a **collection** of formulas. By 'syntactic', we mean that we are only concerned with the *form* of the formulas, the symbols that comprise them, not their semantic interpretation—the part of logic concerned with how formulas are to be interpreted is called **model theory**. What counts as a formula varies from one proof theory to the next, but usually they are certain strings of symbols. Intuitively, to say that A is deducible from Γ is to say that if the formulas in Γ have been established, then A can also be established.

11.1.1 Finite multisets

What counts as a collection of formulas also varies from one proof theory to the next. In some proof theories, the collections are taken to be sets; in others, strings. We will take them to be **finite multisets**.

Roughly speaking, finite multisets represent a sort of compromise between finite sets and strings. They are like strings in that repetitions matter, but they are like sets in that order does *not* matter. Technically, for any set S, a finite S-multiset is an equivalence class of S-strings, where two strings count as equivalent if they are permutations of each other. But we can alternatively think of a finite S-multiset as a function f from a finite subset S' of S to the positive natural numbers. From this perspective, for every $s \in S'$, $f(s) = n$ if and only if s occurs n times in the multiset. So, writing multisets with square brackets rather than the curly brackets we use for ordinary sets, $[A]$ is a different multiset from $[A, A]$, but $[A, B]$ and $[B, A]$ are the same multiset.

11.1.2 Formulas in linear logic

Another prerequisite for defining a proof theory is to recursively define the set of formulas. The base of the recursion specifies some **basic formulas**, and the recursion clauses tell how to build up additional formulas using **connectives**.

The set of linear logic formulas is defined as follows:[1]

1. Any basic formula is a formula.

2. If A and B are formulas, then so is $A \multimap B$.

[1] Actually, there are many linear logics. The one we describe here, whose only connective is $\multimap$, is implicative intuitionistic linear propositional logic.

3. Nothing else is a formula.

Note that we still need to specify somehow what the basic formulas are. The connective $\multimap$ is called **linear implication**, sometimes also called *lollipop*. We adopt the convention that $A \multimap B \multimap C$ abbreviates $A \multimap (B \multimap C)$; this convention is often called the **right associativity of** $\multimap$. As we'll see, $\multimap$ works a lot like the $\rightarrow$ of familiar propositional logic (see Chapter 13), except that it has fewer options.

11.1.3 A linguistic application: tectogrammar

Linear logic is used in **categorial grammar (CG)** frameworks, such as λ-grammar (Muskens, 2001, 2007), abstract categorial grammar (ACG, de Groote, 2001), linear categorial grammar (LCG, Martin, 2013; Martin and Pollard, 2014) and hybrid grammar, which distinguish between **tectogrammatical structure** (also called **abstract syntax** or **syntactic combinatorics**) and **phenogrammatical structure** (also called **concrete syntax**). Such frameworks are sometimes called **curryesque**, after Haskell Curry, who first made this distinction (1961). Tectogrammatical structure drives the syntactic composition. Phenogrammatical structure (**phenogrammar** or simply **pheno**) is concerned with surface realization, including word order and intonation.

In curryesque frameworks, linear logic formulas, called **tecto types** (or just **tectos**) play a role analogous to that played by *nonterminals* in a context-free grammar (CFG): they can be thought of as names of syntactic categories of linguistic expressions. However, a curryesque grammar has far fewer rules than a CFG, because the combinatory potential of a linguistic expression is encoded in its tecto.

In a simple LCG of English (ignoring details such as case, agreement, and verb inflectional form), we might take the basic tectos, the basic formulas in the underlying linear logic, to be:

Ordinary sentences S

***that*-sentences** $\overline{\text{S}}$

Noun phrases, such as names NP

Dummy pronoun *it* It

Common nouns N

Then some nonbasic tectos, formed based on the definition of linear logic formulas, are the following:

Attributive adjectives $\mathrm{N} \multimap \mathrm{N}$

Complementizer *that* $\mathrm{S} \multimap \overline{\mathrm{S}}$

Intransitive verbs $\mathrm{NP} \multimap \mathrm{S}$

Transitive verbs $\mathrm{NP} \multimap \mathrm{NP} \multimap \mathrm{S}$

Ditransitive verbs $\mathrm{NP} \multimap \mathrm{NP} \multimap \mathrm{NP} \multimap \mathrm{S}$

Sentential-complement verbs $\mathrm{NP} \multimap \overline{\mathrm{S}} \multimap \mathrm{S}$

Quantificational noun phrases $(\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}$

11.1.4 Contexts, Sequents, and Provability

A finite multiset of formulas is called a **context**. Careful: this is a distinct usage from the notion of context as the linguistically relevant features of the situation in which an expression is uttered. We use capital Greek letters (usually Γ or Δ) as metavariables ranging over contexts.

An ordered pair $\langle \Gamma, A \rangle$ of a context and a formula is called a **sequent**. Γ is called the **context** of the sequent and A is called the **statement** of the sequent. The formula occurrences in the context of a sequent are called its **hypotheses** or **assumptions**.

The proof theory recursively defines a set of sequents. That is, it recursively defines a relation between contexts and formulas. The relation defined by the proof theory is called **deducibility**, **derivability**, or **provability**, and is denoted by $\vdash$ (read *deduces*, *derives*, or *proves*). The metalanguage assertion that $\langle \Gamma, A \rangle \in \vdash$ is usually written $\Gamma \vdash A$. Such an assertion is called a **judgment**. The symbol $\vdash$ that occurs between the context and the statement of a judgment is called **turnstile**. If Γ is empty, we usually just write $\vdash A$. If Γ is the singleton multiset with one occurrence of B, we write $B \vdash A$. Commas in contexts represent multiset union, e.g., if $\Gamma = [A, B]$ and $\Delta = [B]$, then $\Gamma, \Delta = [A, B, B]$.[2]

[2]Technically, multiset union is defined as follows. Let S' and S'' be finite subsets of some set S, and $f : S' \to \omega \setminus \{0\}$ and $g : S'' \to \omega \setminus \{0\}$ multisets. Then for every $s \in S' \cup S''$,

$$f, g(s) =_{\mathrm{def}} \begin{cases} f(s) & \text{if } s \in \mathbf{dom}(f) \text{ but } s \notin \mathbf{dom}(g) \\ g(s) & \text{if } s \in \mathbf{dom}(g) \text{ but } s \notin \mathbf{dom}(f) \\ f(s) + g(s) & \text{otherwise} \end{cases} .$$

The base clauses of the proof theory identify certain sequents, called **axioms**, as deducible, The recursion clauses of the proof theory, called **(inference) rules**, are (metalanguage) conditional statements, whose antecedents are conjunctions of judgments and whose consequent is a judgment. The judgments in the antecedent of a rule are called its **premisses**, and the consequent is called its **conclusion**. Rules are notated by a horizontal line with the premisses above and the conclusion below, as follows, where $P_1, \ldots, P_n$ are the premisses and C the conclusion:

$$\frac{P_1 \qquad \cdots \qquad P_n}{C}$$

11.2 (Pure) linear logic

11.2.1 Axioms and rules

The proof theory for (pure) linear logic has one schema of axioms, and two schemas of rules. The axiom schema, called *Refl* (Reflexivity), *Hyp* (Hypotheses), or just *Ax* (Axioms), looks like this:

$$A \vdash A$$

To call this an axiom **schema** is just to say that upon replacing the metavariable A by any (not necessarily basic) formula, we get (a judgment that specifies) an axiom, e.g.:

$$\mathrm{NP} \vdash \mathrm{NP}$$

We often speak of axiom and rule schemas and their instantiations with formulas and contexts interchangeably. In LCG, hypotheses play a role analogous to that of **traces** in frameworks such as the minimalist program (MP) and head-driven phrase structure grammar (HPSG).

The two rule schemas are *Modus Ponens*, also called $\multimap$-Elimination, and *Hypothetical Proof*, also called $\multimap$-Introduction. The Modus Ponens rule schema is defined as follows:

$$\frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B} \multimap\mathrm{E}$$

And the definition of the Hypothetical Proof rule schema is given below.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap\mathrm{I}$$

In both of these rule schemas, A and B are metavariables over formulas and Γ and Δ are metavariables over contexts.

Modus Ponens **eliminates** the connective $\multimap$, i.e. there is an occurrence of $\multimap$ in one of the premisses (called the **major** premiss; the other premiss is called the **minor** premiss) that does not occur in the conclusion. Hypothetical Proof **introduces** $\multimap$, i.e., there is an occurrence of $\multimap$ in the conclusion but not in the premiss. Note that since contexts do not have a notion of order, the premiss $\Gamma, A \vdash B$ of the $\multimap$-Introduction rule only requires A to be among the hypotheses in the context, even though we write it in the rightmost position. Pairs of rules that eliminate and introduce connectives are characteristic of the natural deduction style of proof theory.

11.2.2 Theorems and proof trees

If $\Gamma \vdash A$, then we call the sequent $\langle \Gamma, A \rangle$ a **theorem** (in the present case, of linear logic). It is not hard to see that $\Gamma \vdash A$ if and only if there is a **proof tree** whose root is labeled with the sequent $\langle \Gamma, A \rangle$. By a *proof tree* we mean an ordered tree whose nodes are labeled by sequents, such that

1. the label of each leaf node is an axiom; and
2. the label of each nonleaf node is (the sequent of) the conclusion of a rule such that (the sequents of) the premisses of the rule are the labels of the node's daughters.

In displaying a proof tree, the root appears at the bottom and the leaves at the top (so from a logician's point of view, linguists' trees are upside down). Even though technically the labels are sequents, we conventionally write the corresponding judgments (metalanguage assertions that the sequents are deducible). Instead of edges connecting mothers to daughters as in linguists' trees, we write horizontal lines with the label of the mother below and the labels of the daughters above (just as in inference rules). Sometimes, as a mnemonic, we label the horizontal line with the name of the rule schema that is instantiated.

The simplest possible proof tree in linear logic has just one leaf, which is also the root. In this case the only option is for the node to be labeled by an axiom, e.g., $\mathrm{NP} \vdash \mathrm{NP}$. This just means that any formula is deducible from itself. Although this doesn't sound very exciting, it turns out that an elaborated form of such axioms come into play in hypothetical reasoning in syntax, the categorial grammar analog of *wh*-movement, quantifier raising, focus constructions, etc.

A somewhat more interesting proof tree is the following:

$$\dfrac{\dfrac{\mathrm{NP} \multimap \mathrm{S} \vdash \mathrm{NP} \multimap \mathrm{S} \qquad \mathrm{NP} \vdash \mathrm{NP}}{\mathrm{NP} \multimap \mathrm{S}, \mathrm{NP} \vdash \mathrm{S}}\multimap\mathrm{E}}{\mathrm{NP} \vdash (\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}}\multimap\mathrm{I}$$

This is an instance of the **derived rule** of Type Raising (TR) to be introduced below. The statement in the root sequent is the tecto $(\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}$ of quantificational noun phrases, which we abbreviate QP (for *quantifier phrase*). This enables an ordinary (i.e., nonquantificational) NP to have the so-called higher type of a QP, for example, in coordinate structures such as *Pedro and some donkey*.

Exercise 11.1. Give a natural-deduction proof tree for the following theorem of linear logic, called Antecedent Permutation:

$$\vdash (A \multimap B \multimap C) \multimap B \multimap A \multimap C$$

Exercise 11.2. Give a natural-deduction proof tree for the following theorem of linear logic, a version of Type Raising:

$$\vdash A \multimap (A \multimap B) \multimap B$$

Exercise 11.3. Give a natural-deduction proof tree for the following theorem of linear logic, a version of Geach's Law:

$$\vdash (A \multimap B) \multimap (C \multimap A) \multimap (C \multimap B)$$

11.2.3 Derived rules

In natural deduction, we say that an inference rule is **derivable** if we *could have* proved the conclusion if the premiss(es) had been provable. In other words, we derive an inference rule by presenting a proof tree where

1. the root sequent is the conclusion of the rule, and
2. we allow the premisses of the rule, in addition to the usual axioms, to label the leaves.

For example, the following derived rule is the Converse of Hypothetical Proof (i.e., the premiss and the conclusion are switched):

$$\dfrac{\Gamma \vdash A \multimap B}{\Gamma, A \vdash B}\,\mathrm{CHP}$$

CHP is derivable in our natural deduction formulation of linear logic as follows:

$$\frac{\Gamma \vdash A \multimap B \qquad A \vdash A}{\Gamma, A \vdash B}$$

Notice that the premiss $A \vdash A$ does not appear in the derived rule CHP. That's because a proof that $A \vdash A$ is always available, for any formula A, via the axiom schema. Thus the rule CHP is only contingent on a proof that $\Gamma \vdash A \multimap B$.

Some more useful derived rules are given below.

Hypothetical Syllogism (also called Composition)

$$\frac{\Gamma \vdash B \multimap C \qquad \Delta \vdash A \multimap B}{\Gamma, \Delta \vdash A \multimap C}\ \text{HS}$$

Generalized Contraposition

$$\frac{\Gamma \vdash A \multimap B}{\Gamma \vdash (B \multimap C) \multimap A \multimap C}\ \text{GC}$$

Type Raising

$$\frac{\Gamma \vdash A}{\Gamma \vdash (A \multimap B) \multimap B}\ \text{TR}$$

Once derived, a rule can be used in any proof just as if it were one of the original rules of the proof system.

In linear logic, for any formulas A and B, $A \vdash B$ is a theorem iff the rule schema

$$\frac{\Gamma \vdash A}{\Gamma \vdash B}$$

is derivable. For example, the derived rules GC and TR could just as well be expressed, respectively, as the following theorems:

$$A \multimap B \vdash (B \multimap C) \multimap A \multimap C$$
$$A \vdash (A \multimap B) \multimap B$$

Exercise 11.4. Give a natural-deduction proof tree for the following theorem of linear logic, a version of Generalized Contraposition:

$$\vdash (A \multimap B) \multimap (B \multimap C) \multimap (A \multimap C)$$

Exercise 11.5. Derive the rule of Hypothetical Syllogism.

Chapter 12

The Lambek Calculus and Lambek Grammar

The mathematician Joachim Lambek invented a simple kind of propositional logic in 1958. He called it the **syntactic calculus**, but everyone else calls it the **Lambek calculus**, L for short. L can be thought of as a bidirectional variant of linear logic (LL, Girard, 1987), though historically it predates LL by nearly three decades. Lambek used L to write grammars of a certain kind, now called **Lambek grammars**, where the formulas are used as names of syntactic categories (sets of strings), roughly in the way that nonterminals serve the same purpose in context-free grammar. The mainstream of categorial grammar research, e.g. Morrill's **type-logical grammar (TLG)** and Moortgat's **categorial type logic (CTL)**, extends this approach in various ways.

12.1 Proof Theory of L

The set of L formulas is defined as follows:

1. Any basic formula is a formula. (So, the finite set of basic formulas has to be specified. In the application to natural language grammar, these will be the basic syntactic types, such as NP and S.)

2. If A and B are formulas, then so are $A\backslash B$ and B/A.[1]

[1] The antecedent is always the formula *under* the connective. This differs from the convention in *combinatory categorial grammar (CCG)*, where the antecedent formula is always written to the *right* of the connective.

3. Nothing else is a formula.

The proof theory of L is similar to the proof theory for LL. There is a single axiom, namely

$$A \vdash A\,,$$

which is identical to the axiom in LL. But whereas LL has only a single introduction rule and a single elimination rule for $\multimap$, L has separate rules for introducing and eliminating the two implications. The rules of L are given in Figure 12.1.

$$\frac{\Gamma \vdash A \qquad \Delta \vdash A\backslash B}{\Gamma, \Delta \vdash B}\,\backslash\mathrm{E}$$

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash A\backslash B}\,\backslash\mathrm{I}$$

$$\frac{\Gamma \vdash B/A \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}\,/\mathrm{E}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash B/A}\,/\mathrm{I}$$

Figure 12.1: Proof rules of L.

The differences between LL and L, presented as sequent-style natural-deduction systems, are:

1. LL has *one* implication $\multimap$; whereas L has *two*, $\backslash$ and $/$, as mentioned above.

2. In LL, contexts are *multisets*; whereas in L they are *strings*.

3. In LL, commas in contexts stand for *multiset union*; whereas in L they stand for *concatenation* (of strings; see Chapter 9).

4. In LL, hypotheses are unordered; whereas in the conclusion of /E the hypotheses from the major premiss precede those from the minor premiss, and *vice versa* for $\backslash$E.

5. The rule $\multimap$I can discharge *any one of* the hypotheses; whereas /I discharges the *rightmost* hypothesis, and \I discharges the *leftmost* hypothesis.

12.2 Lambek Grammars Defined

A Lambek grammar G is an ordered triple $\langle \mathrm{W}, \mathrm{B}, \mathrm{D} \rangle$ where

1. W is a nonempty finite set of **words**, e.g., Mary, walks, etc.
2. B is a nonempty finite set of **basic syntactic types**.
3. D is a set of **lexical entries**, ordered pairs $\langle w, A \rangle$ where $w \in \mathrm{W}$ is a word and $A \in \mathrm{B}$ is a (not necessarily basic) syntactic type.

12.2.1 Syntactic Categories of a Lambek Grammar

A Lambek grammar $\mathrm{G} = \langle \mathrm{W}, \mathrm{B}, \mathrm{D} \rangle$ recursively defines a function $\mathbf{C}$ from syntactic types to W-languages as follows:

1. For $\langle w, A \rangle \in \mathrm{D}$, (the length-one string of) $w \in \mathbf{C}_A$.
2. For any $n > 0$ and syntactic types $A_0, \ldots, A_n$, if
   (a) $s_i \in \mathbf{C}_{A_i}$ $(i < n)$, and
   (b) $A_0, \ldots, A_{n-1} \vdash A_n$

   then $s_0 \cdot \cdots \cdot s_{n-1} \in \mathbf{C}_{A_n}$.
3. Nothing else is in any of the $\mathbf{C}_A$.

For each syntactic type A, the language $\mathbf{C}_A$ is called the **syntactic category** corresponding to A.

In a simple Lambek grammar of English (ignoring details such as case, agreement, and verb inflectional form), we might take the basic syntactic types to be:

Ordinary (declarative) sentences S

***that*-sentences** $\overline{\mathrm{S}}$

Main-clause interrogative sentences Q

Embedded interrogative sentences $\overline{\mathrm{Q}}$

Noun phrases (such as names) NP

'Dummy pronoun' *it* It

Common nouns N

Nonpredicative *(to)*-prepositional phrases To

Some more complex types are the following:

Attributive adjectives N/N

Postnominal modifiers (including relative clauses) N\N

Subject relativizers $\mathrm{Rel}_s =_{\mathrm{def}} (\mathrm{N}\backslash\mathrm{N})/(\mathrm{NP}\backslash\mathrm{S})$

Object relativizers $\mathrm{Rel}_o =_{\mathrm{def}} (\mathrm{N}\backslash\mathrm{N})/(\mathrm{S}/\mathrm{NP})$

Declarative complementizer *that* $\overline{\mathrm{S}}/\mathrm{S}$

Determiners of subject QPs $\mathrm{Det}_s =_{\mathrm{def}} (\mathrm{S}/(\mathrm{NP}\backslash\mathrm{S}))/\mathrm{N}$

Determiners of object QPs $\mathrm{Det}_o =_{\mathrm{def}} ((\mathrm{S}/\mathrm{NP})\backslash\mathrm{S})/\mathrm{N}$

Intransitive verbs $\mathrm{VP} =_{\mathrm{def}} \mathrm{NP}\backslash\mathrm{S}$

Transitive verbs $\mathrm{TV} =_{\mathrm{def}} (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}$

Ditransitive verbs $\mathrm{DV} =_{\mathrm{def}} ((\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP})/\mathrm{NP}$

Sentential-complement verbs $\mathrm{SV} =_{\mathrm{def}} (\mathrm{NP}\backslash\mathrm{S})/\overline{\mathrm{S}}$

Interrogative-complement verbs $\mathrm{QV} =_{\mathrm{def}} (\mathrm{NP}\backslash\mathrm{S})/\overline{\mathrm{Q}}$

Nonpredicative preposition *to* To/NP

For example, $\mathrm{a} \cdot \mathrm{kitten} \cdot \mathrm{that} \cdot \mathrm{chris} \cdot \mathrm{saw} \cdot \mathrm{exploded} \in \mathbf{C}_{\mathrm{S}}$ because:

1. $\mathrm{a} \in \mathbf{C}_{\mathrm{Det}_s}$, $\mathrm{kitten} \in \mathbf{C}_{\mathrm{N}}$, $\mathrm{that} \in \mathbf{C}_{\mathrm{Rel}_o}$, $\mathrm{chris} \in \mathbf{C}_{\mathrm{NP}}$, $\mathrm{saw} \in \mathbf{C}_{\mathrm{TV}}$, $\mathrm{exploded} \in \mathbf{C}_{\mathrm{VP}}$; and
2. $\mathrm{Det}_s, \mathrm{N}, \mathrm{Rel}_o, \mathrm{NP}, \mathrm{TV}, \mathrm{VP} \vdash \mathrm{S}$.

Exercise 12.1. Prove the sequent in item 2, immediately above.

12.3 Empirical Predictions of Lambek Grammars

It was conjectured for a long time that the syntactic categories of Lambek grammars were context-free languages (CFLs). This conjecture is usually attributed to Chomsky, and because Chomsky (*inter multa alia*) also asserted that natural languages could not all be CFLs, it also came to be widely believed that Lambek grammar was a hopeless case. Subsequently Stuart Shieber (1985) established definitively that the string set of Swiss German sentences was not a CFL. Then in 1993, Mati Pentus proved that indeed the syntactic categories of a Lambek grammar are CFLs. And so, Lambek grammar as it stands does not provide a satisfactory basis for natural language syntax.

Setting that fact aside, there are some aspects of the analysis of languages like English that Lambek grammars have a hard time with. One is extraction, at least in the general case. To see why, consider how the following $\overline{\text{Q}}$s would be modeled in a Lambek grammar:

(12.1) a. who [t introduced Kim to Sandy]
b. who [Chris introduced Kim to t]
c. who [Chris introduced t to Sandy]

Exercise 12.2. Give syntactic type assignments for *who* that account for the examples in (12.1).

Exercise 12.3. Can you think of a way to specify the entire set of syntactic assignments needed to analyze embedded *who*-questions?

Another point of difficulty for Lambek grammars is the scoping of quantifier phrases (QPs), as in the following:

(12.2) a. Everyone introduced Kim to Sandy.
b. Chris introduced Kim to everyone.
c. Chris introduced everyone to Sandy

Exercise 12.4. Give syntactic type assignments for *everyone* that account for the examples in (12.2). Keep in mind that, since syntactic combination drives semantic composition, a QP must take its scope as a syntactic 'argument'; i.e., there must be a \E or /E step in the proof whose major and minor premisses corresponds to the QP and its scope respectively.

Exercise 12.5. Can you think of a way to specify the entire set of syntactic assignments for *every* needed to account for every possible position where

it can occur and every possible constituent containing that position that it might scope over?

On the other hand, aside from the elegant handling of the syntax-semantics interface (an advantage shared with all forms of categorial grammar), perhaps the biggest selling point for Lambek grammars to linguists is its success in analyzing *coordinate structures*, including coordination of things that are traditionally not analyzed as constituents:

(12.3) Kim tolerates, and Sandy adores, Sicilian miniature donkeys.
(Right Node Raising)

(12.4) Kim gave Chris books and Sandy records.
(Argument Cluster Coordination)

The treatment of coordination is a selling point shared with CCG, which also has the directional implications $\backslash$ and $/$, on which the analysis of coordination hinges. In Lambek grammar (as in CCG), the key idea behind the analysis of coordination is that expressions that can be conjoined have the same syntactic type, a result that is enforced by assigning coordinators (such as *and* and *or*) syntactic types of the form $(A\backslash A)/A$.

In particular, if A is a functor type (i.e., A is $B\backslash C$ or C/B), this analysis guarantees that conjoined functors are looking for their argument in the same direction, which is often (but not invariably) empirically borne out:

(12.5) Kim tolerates, and Sandy adores, Sicilian miniature donkeys.
(Both conjuncts are S/NP)

(12.6) Kim tolerates donkeys, and adores burros.
(Both conjuncts are NP$\backslash$S)

(12.7) * Kim tolerates, and adores burros, Sandy.
(According to the linear categorial grammar discussed in Chapter 11, both conjuncts are NP $\multimap$ S.)

Exercise 12.6. Give Lambek analyses of the sentences in (12.3) and (12.4).

12.4 Extending L with Monoidal Terms

For linguists, it's more practical to present Lambek grammatical analyses in a format that directly associates string-denoting algebraic terms with proof tree nodes. Superficially, these string terms resemble Curry-Howard proof terms (see Chapter **??**), but they aren't really (there is no application or abstraction). In this setup, the grammar assigns string a to syntactic type A

iff the term-labeled sequent $\vdash a : A$ is provable, where the string a is called the **label** for (this instance of) the category A. We will call the system L extended with monoidal terms L^{+}.

The terms here are *algebraic* terms in the language of **monoids** (see Chapter **??**), *not* terms of typed lambda calculus (TLC) or higher-order logic (discussed in Chapters **??** and **??**, respectively). Thus we have:

1. just one type of variable (we use s, t, u),
2. a constant $\mathbf{e}$ for the null string,
3. constants mary, walks, etc., for words,
4. the symbol $\cdot$ for concatenation,
5. but neither of the TLC term constructors (application and lambda abstraction; see Chapter **??**)

There is an equivalence relation defined for monoidal terms similar to the the one for TLC terms, except that of course all the clauses in the definition that mention application or abstraction are missing. In the following definitions, a, a', a'', b, b', and c are all metavariables over monoidal terms.

Definition 12.1 (Equivalences of Equational Reasoning in L^{+})**.** The following describe term-level equivalences.

(ρ) $a \equiv a$

(σ) If $a \equiv a'$, then $a' \equiv a$.

(τ) If $a \equiv a'$ and $a' \equiv a''$, then $a \equiv a''$.

(μ) If $a \equiv a'$ and $b \equiv b'$, then $a \cdot b \equiv a' \cdot b'$.

Definition 12.2 (Monoid Equivalences in L^{+})**.** The equivalences below pertain to the term concatenation operator $\cdot$.

1. $(a \cdot b) \cdot c \equiv a \cdot (b \cdot c)$
2. $\mathbf{e} \cdot a \equiv a$
3. $a \cdot \mathbf{e} \equiv a$

In the language of preorders (see Chapter 4), the first three clauses of Definition 12.1 state that the relation $\equiv$ between terms is an equivalence relation; the last clause states that equivalent terms can be substituted under concatenation. The first clause of Definition 12.2 just says that $\cdot$ is associative, and the last two ensure that **e** is a two-sided **identity** for $\cdot$ (see Chapter **??**).

In L^+, terms are interpreted as strings, syntactic types are interpreted as languages (string sets), and $\backslash$ and $/$ are interpreted as the language residuals. Contexts in sequents are sets (not strings, as in the base system L) because the ordering issues will be taken care of in the term of the statement. The judgment $\Gamma \vdash a : A$ says that under any assignment that maps each variable to a member of the language that interprets the syntactic type it is paired with in Γ, a is interpreted as a member of the language that interprets A.

The axiom of L^+ is just like the axiom for L, except that the metavariable s over term labels is added:

$$s : A \vdash s : A$$

The rules of L^+, given in Figure 12.2, correspond to the rules of L, but they also say what happens to the terms. As you might expect, each rule of L^+ has

$$\frac{\Gamma \vdash a : A \qquad \Delta \vdash b : A\backslash B}{\Gamma, \Delta \vdash a \cdot b : B}\ \backslash\mathrm{E}$$

$$\frac{s : A, \Gamma \vdash s \cdot a : B}{\Gamma \vdash a : A\backslash B}\ \backslash\mathrm{I}$$

$$\frac{\Gamma \vdash a : B/A \qquad \Delta \vdash b : A}{\Gamma, \Delta \vdash a \cdot b : B}\ /\mathrm{E}$$

$$\frac{\Gamma, s : A \vdash a \cdot s : B}{\Gamma \vdash a : B/A}\ /\mathrm{I}$$

Figure 12.2: Proof rules of L^+.

the same structure at the level of syntactic categories as the corresponding rule of L. The elimination rules concatenate the terms in the premisses in a way that respects the location of the argument term (to the left or to the right of the functor). The introduction rules say that if a term contains a

subterm s that originated from a hypothesis, then the functor term in the conclusion is labeled by the result of removing the hypothesized term.

Chapter 13

Propositional and First-order Logics

13.1 Positive Intuitionistic Propositional Logic

The system of *Positive Intuitionistic Propositional Logic (PIPL)* is like linear logic (LL, Chapter 11) except that it has more connectives, more axioms, and more rules. The PIPL connectives are the 0-ary connective T (read *true*), and the three binary connectives $\to$ (intuitionistic implication), $\wedge$ (conjunction), and $\vee$ (disjunction).

With the addition of negation, PIPL can be extended to *intuitionistic* or *classical propositional logic*, depending on what rules are adopted for negation. These in turn can be extended to *first-order logics* with the addition of universal and existential quantifiers and corresponding rules. PIPL also underlies the *type system* of typed lambda calculus (TLC, Chapter **??**) and higher order logic (HOL, Chapter **??**), which are widely used for theorizing about meaning, and in curryesque categorial frameworks for theorizing about phenogrammar.

13.1.1 Axioms and Rules

Like LL, PIPL has the Hypothesis schema

$$A \vdash A\,,$$

but it additionally has the True axiom

$$\vdash \mathsf{T}\,.$$

Intuitively, T is usually thought of corresponding to an arbitrary necessary truth. The True axiom can also be thought of as a nullary introduction rule for T.

The natural deduction rules for PIPL include both introduction and elimination schemas for implication, conjunction, and disjunction. There are also two **structural rules**, called *Weakening* and *Contraction*, which affect only the contexts of sequents. Contexts in PIPL modeled not multisets but rather just sets, and consequently the comma in contexts like Γ, A represents ordinary set union. Sequents, theorems, and proof trees in PIPL are defined in the same way as for linear logic (Chapter 11).

The rules for implication are given in Figure 13.1, and are essentially identical to the LL rules Modus Ponens and Hypothetical Proof except that $\multimap$ is replaced with $\to$. The rules for conjunction, in Figure 13.2 include *two*

$$\frac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B} \to\text{E}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to\text{I}$$

Figure 13.1: PIPL rules for implication.

elimination rules $\wedge$E1 and $\wedge$E2 (for eliminating the left and right conjunct respectively). The rules for disjunction (Figure 13.3) include *two* introduction

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge\text{E1}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge\text{E2}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B} \wedge\text{I}$$

Figure 13.2: PIPL rules for conjunction.

rules $\vee$I1 and $\vee$I2 (for introducing the left and right disjunct respectively). The rules for introducing disjunction just say that a deduced formula can

$$\frac{\Gamma \vdash A \vee B \qquad A, \Delta \vdash C \qquad B, \Delta \vdash C}{\Gamma, \Delta \vdash C} \vee\text{E}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee\text{I1}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee\text{I2}$$

Figure 13.3: PIPL rules for disjunction.

be disjoined with any formula. The elimination rule is a bit different from the elimination rules for the other connectives we have seen so far because nothing from the major premiss $A \vee B$ survives in the conclusion. This rule just says that if some formula C can either be deduced from a set of premisses containing A or one containing B, and if one of A or B must be true (i.e., $A \vee B$ has been proved), then C's deducibility does not depend on a proof of A or a proof of B.

The PIPL structural rules, which are given in Figure 13.4, affect the contexts of a sequent but not the statement. The rule of Weakening (also

$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A} \text{W}$$

$$\frac{\Gamma, B, B \vdash A}{\Gamma, B \vdash A} \text{C}$$

Figure 13.4: PIPL structural rules.

known as W) intuitively says that if we can prove something from certain assumptions, we can also prove it with more assumptions. The intuition behind the Contraction rule (C) is that repeated assumptions can be eliminated. Both of these rules can be seen as consequences of the fact that contexts, in PIPL, are sets: W says that whatever can be proved from some set of premisses Γ can also be proved from any Δ provided that $\Gamma \subseteq \Delta$; C simply says that repetitions do not matter. The structural rules may seem

too obvious to be worth stating, but in fact they *must* be stated, because in some logics (such as LL) they are not available!

Exercise 13.1. Give a natural deduction proof tree for the following theorem of PIPL:

$$\vdash A \to B \to A$$

Exercise 13.2. Give a natural deduction proof tree for the following theorem of PIPL:

$$\vdash (A \to A \to B) \to (A \to B)$$

13.2 Extensions of PIPL

By adding two more connectives—F (false), and $\neg$ (negation)—and corresponding rules and axioms to PIPL we get full *Intuitionistic Propositional Logic (IPL)*. With the addition of one more rule we get *Classical Propositional Logic (CPL)*. And with the addition of rules for (universal and existential) quantification to either IPL or CPL, we get (either intuitionistic or classical) *First-order Logic (FOL)*.

The connective F is usually thought of as corresponding to an arbitrary impossibility (a necessary falsehood), and negation can be defined in terms of implication and F, as follows:

$$\neg A =_{\text{def}} A \to \mathsf{F} \tag{13.1}$$

Exercise 13.3. Using just the rules for PIPL and the definition of negation in (13.1), derive the following theorems:

$$\mathsf{F} \vdash \neg A$$
$$A, \neg A \vdash \neg B$$
$$A \vdash \mathsf{T}$$
$$\neg A, A \vdash \mathsf{F}$$
$$A \vdash \neg(\neg A)$$

Exercise 13.4. Using just the rules for PIPL and the definition of negation in (13.1), derive the following rule:

$$\frac{\Gamma, A \vdash \mathsf{F}}{\Gamma \vdash \neg A}$$

$$\frac{\Gamma \vdash \neg A \qquad \Delta \vdash A}{\Gamma, \Delta \vdash \mathsf{F}}\ \neg\mathrm{E}$$

$$\frac{\Gamma, A \vdash \mathsf{F}}{\Gamma \vdash \neg A}\ \neg\mathrm{I}$$

Figure 13.5: IPL negation rules.

As an alternative to the definition in (13.1), negation can be given as a basic connective, but some rules governing its use need to be stated. Those rules, ¬-Elimination (or *Indirect Proof*) and ¬-Introduction (or *Proof by Contradiction*) are given in Figure 13.5. The elimination rule for negation (¬E) makes deriving a contradiction (proving both A and its denial) equivalent to deriving F; the introduction rule ¬I lets us falsify one of the premisses that produced a proof of F. If ¬ is defined in terms of → and F, as in (13.1), then the rules pertaining to negation do not need to be stated—they are just instantiations of the rule schemas for implication in Figure 13.1 with $B = \mathsf{F}$.

The False axiom

$$\mathsf{F} \vdash A$$

is traditionally called *ex falso quodlibet (EFQ)*. EFQ is easily shown to be equivalent to the following rule of F-Elimination:

$$\frac{\Gamma \vdash \mathsf{F}}{\Gamma \vdash A}\ \mathsf{F}\mathrm{E}$$

This axiom/rule encodes the principle behind *ex falso quodlibet*, which translates roughly as *from falsehood, anything*: any formula A follows from F.

13.2.1 Classical Propositional Logic

Classical Propositional Logic (CPL) is obtained from IPL by the addition of any one of the following, which can be shown to be equivalent:

Reductio ad Absurdum (RAA)

$$\frac{\Gamma, \neg A \vdash \mathsf{F}}{\Gamma \vdash A}\ \mathrm{RAA}$$

Double Negation Elimination (DNE)

$$\frac{\Gamma \vdash \neg(\neg A)}{\Gamma \vdash A}\ \text{DNE}$$

Law of Excluded Middle (LEM)

$$\vdash A \vee \neg A$$

In IPL, each of the preceding three rules/axioms is equivalent to Peirce's Law, which doesn't mention F or $\neg$:

$$\vdash ((A \to B) \to A) \to A$$

Peirce's Law is a theorem of CPL.

Exercise 13.5. Give a CPL natural deduction proof tree for Peirce's Law.

13.2.2 First-order Logics

Rules for quantifiers can be added to either IPL or CPL to obtain either intuitionistic or classical versions of FOL. Much of the terminology having to do with quantification, scope, bound and free variables, etc., was introduced in Chapter 2. As a preliminary to giving the proof rules for quantification, we also add some new terminology pertaining to variable substitution. If replacing every occurrence of x in A with t does not cause any of t's free variables to become bound, then t is *free for x in A*. The formula that results form replacing all free occurrences of x in A by t is written $A[x/t]$.

The rules for quantification, detailing in Figure 13.6, can be thought of as counterparts of those for $\wedge$ and $\vee$ where, instead of just two 'juncts', there is one for each individual in the domain of quantification.

The $\forall$-Elimination rule $\forall$E, also known as *Universal Instantiation*, allows the bound variable in a universally quantified formula to be replaced by any term, with the quantification removed. Its counterpart, the $\forall$-Introduction rule $\forall$I, is also known as *Universal Generalization*—it simply allows the inference that if a formula is true then it is true for all individuals.

The rule of $\exists$-Elimination ($\exists$E) says that if it can be derived from Γ that A holds for some x, and C can be derived from a set of assumptions $\Delta, A[x/y]$, then C is provable from just Γ, Δ, since C too derives from a proof that A holds for some y. By the $\exists$-Introduction rule $\exists$I, also called *Existential Generalization*, it can be concluded that $\exists x A$ based on a proof of A in which the term t has replaced the occurrences of x.

$$\frac{\Gamma \vdash \forall x A}{\Gamma \vdash A[x/t]}\ \forall\mathrm{E} \quad (t \text{ free for } x \text{ in } A)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x A}\ \forall\mathrm{I} \quad (x \text{ not free in any } B \in \Gamma)$$

$$\frac{\Gamma \vdash \exists x A \qquad \Delta, A[x/y] \vdash C}{\Gamma, \Delta \vdash C}\ \exists\mathrm{E} \quad (y \text{ free for } x \text{ in } A,\ y \text{ not free in } A)$$

$$\frac{\Gamma \vdash A[x/t]}{\Gamma \vdash \exists x A}\ \exists\mathrm{I} \quad (t \text{ free for } x \text{ in } A)$$

Figure 13.6: Rules for quantification.

Exercise 13.6. A first-order logic can be extended with *predicates* that produce formulas by relating zero or more individuals. Letting x and y be sets, assume that both $x \in y$ and $x = y$ are formulas corresponding to set membership and equality, respectively, and assume the definition of $\subseteq$ given in §2.3. Then give statements of Assumptions 1–9 in first-order logic extended with $\in$, $=$, and $\subseteq$. As a shorthand, you can use $x \notin y$ to abbreviate $\neg(x \in y)$, and similarly for $=$ and $\subseteq$. You can also abbreviate the formula $(x \to y) \wedge (y \to x)$ by $(x \leftrightarrow y)$.

Appendix A

Deferred Proofs

From Chapter 6

Theorem 6.5 (Recursion Theorem)**.** *Let X be a set, $x \in X$, and $F : X \to X$. Then there exists a unique function $h : \omega \to X$ such that*

(A.1) $h(0) = x$, *and*

(A.2) *For every $n \in \omega$, $h(\mathbf{suc}(n)) = F(h(n))$.*

In the following proof sketch, the key idea is to define

$\mathcal{A} = \{v : \omega \rightharpoonup X \mid$

(a) $v(0) = x \leftrightarrow 0 \in \mathsf{dom}(v)$, and

(b) $\forall n \in \omega(\mathbf{suc}(n) \in \mathsf{dom}(v) \to (n \in \mathsf{dom}(v) \wedge v(\mathbf{suc}(n)) = F(v(n))))$

$\}$

and then define $h = \bigcup \mathcal{A}$.

The proof can be broken into four parts:

1. Show that h is (at least) a partial function.
2. Show that the two clauses in the definition of $\mathcal{A}$ hold when $h = v$.
3. Show that $\mathsf{dom}(h)$ is inductive.
4. Show that h is the only function from ω to X that satisfies the RT conditions.

For a full proof, see Enderton 1977.

Sketch of part 1. Let $S = \{n \in \omega \mid \text{for at most one } y \in X, \langle n, y\rangle \in h\}$. The trick is to show S is inductive and that therefor $S = \omega$ by PMI. □

Sketch of part 2. We already know that $h : \omega \rightharpoonup X$. For (a), suppose $0 \in \mathsf{dom}(h)$. Then by definition of h, there is a $v : \omega \rightharpoonup X$ such that $v(0) = h(0)$. But $v(0) = x$. For (b), suppose $\mathbf{suc}(n) \in \mathsf{dom}(h)$. By definition of h again, there is a $v : \omega \rightharpoonup X$ such that $v(\mathbf{suc}(n)) = h(\mathbf{suc}(n))$. But v satisfies (b), so $n \in \mathsf{dom}(v)$ and $v(\mathbf{suc}(n)) = F(v(n))$. But by definition of h again, $h(n) = v(n)$. So $h(\mathbf{suc}(n)) = v(\mathbf{suc}(n)) = F(v(n)) = F(h(n))$. □

Sketch of part 3. We need to show that $\mathsf{dom}(h)$ is inductive. Then $\mathsf{dom}(h) = \omega$ by PMI. □

Sketch of part 4. Suppose two functions h and h' from ω to X satisfy (A.1) and (A.2). Then we need to show that

$$T = \big\{n \in \omega \mid h(n) = h'(n)\big\}$$

is inductive, and therefore $T = \omega$ by PMI. □

From Chapter 7

Theorem 7.10 (Schröder-Bernstein)**.** *For any sets A and B, if $A \preceq B$ and $B \preceq A$, then $A \approx B$.*

Proof. By definition of $\preceq$, there are injections $f\colon A \to B$ and $g\colon B \to A$. Let C be the unique function from ω to $\wp(A)$ such that $C(0) = A \setminus \mathsf{ran}(g)$ and $C(n+1) = g[f[C(n)]]$ for all $n \in \omega$; henceforth we write C_n for $C(n)$. Now we define $h\colon A \to B$ such that $h(x) = f(x)$ if $x \in \bigcup_{n\in\omega} C_n$ and $h(x) = g^{-1}(x)$ otherwise; this makes sense since $\mathsf{ran}(g) = A \setminus C_0$. We will show h is bijective.

To show h is injective, suppose x and x' are distinct members of A; it suffices to show that $h(x) \neq h(x')$. Since f and g^{-1} are one-to-one, we need only consider the case where $x \in C_m$ and $x' \notin \bigcup_{n\in\omega} C_n$. Now we define $D_n =_{\text{def}} f[C_n]$ for all $n \in \omega$, so that $C_{n+1} = g[D_n]$. Then $h(x) = f(x)$, which is in D_m; but $h(x') = g^{-1}(x')$, which is *not* in D_m (since otherwise we would have $x' \in C_{m+1}$). So $h(x) \neq h(x')$, as desired.

To show h is surjective, let $y \in B$; we will show that $y \in \mathsf{ran}(h)$. Clearly, for each n, $D_n \subseteq \mathsf{ran}(()h)$. So we can assume $y \in B \setminus \bigcup_{n\in\omega} D_n$. Next, we note that, for all n, $g(y) \notin C_n$ (the proof, which is inductive, is left as an exercise). Therefore $g(y) \notin \bigcup_{n\in\omega} C_n$. So $h(g(y)) = g^{-1}(g(y)) = y$. So $y \in \mathsf{ran}(h)$. □

Theorem 7.11. *If A is infinite, then $\omega \preceq A$.*

Proof. Let c be a choice function for A, and let h be the unique function from ω to $\wp(A)$ such that $h(0) = \emptyset$ and $h(n+1) = h(n) \cup \{c(A \setminus h(n))\}$ for all $n \in \omega$. Note for future reference that for any $m, n \in \omega$ with $m < n$, $h(m+1) \subseteq h(n)$. Also define $g\colon \omega \to A$ by $g(n) =_{\text{def}} c(A \setminus h(n))$, so that, for each $n \in \omega$, $h(n+1) = h(n) \cup \{g(n)\}$, and consequently also $g(n) \in h(n+1)$. Clearly, for all $n \in \omega$, $g(n) \notin h(n)$, since $g(n) = c(A \setminus h(n)) \in A \setminus h(n)$.

To complete the proof, we will show g is injective. So let m and n be distinct natural numbers; without loss of generality we can assume that $m < n$. Then $g(m) \in h(m+1)$, and so $g(m) \in h(n)$. But we already showed that $g(n) \notin h(n)$, so $g(m) \neq g(n)$; this shows g is injective as required. □

Bibliography

Abian, Alexander. *The Theory of Sets and Transfinite Arithmetic*. Saunders, London, 1965.

Aczel, Peter. *Non-Well-Founded Sets*, volume 14 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford, California, 1988.

Curry, Haskell. Some logical aspects of grammatical structure. In R. Jakobson, editor, *Structure of Language and its Mathematical Aspects*, number 12 in Proceedings of Symposia in Applied Mathematics. American Mathematical Society, Providence, Rhode Island, 1961. doi:10.1090/psapm/012/9981.

de Groote, Philippe. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, 2001. doi:10.3115/1073012.1073045.

Enderton, Herbert B. *Elements of Set Theory*. Academic Press, London, 1977.

Girard, Jean-Yves. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. doi:10.1016/0304-3975(87)90045-4.

Lambek, Joachim. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958. doi:10.2307/2310058.

Martin, Scott. *The Dynamics of Sense and Implicature*. Ph.D. thesis, Ohio State University, 2013.

Martin, Scott and Carl Pollard. A dynamic categorial grammar. In G. Morrill, R. Muskens, R. Osswald, and F. Richter, editors, *Proceedings of the 19th Conference on Formal Grammar*, number 8612 in Lecture Notes in Computer Science. Springer, 2014. doi:10.1007/978-3-662-44121-3_9.

Muskens, Reinhard. λ-grammars and the syntax-semantics interface. In R. van Rooy and M. Stokhof, editors, *Proceedings of the 13th Amsterdam Colloquium*, 2001.

Muskens, Reinhard. Separating syntax and combinatorics in categorial grammar. *Research on Language and Computation*, 5(3):267–285, 2007. doi:10.1007/s11168-007-9035-1.

Pentus, Mati. Lambek grammars are context free. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 429–433, 1993.

Russell, Bertrand. *Letter to Frege*, pages 124–125. In van Heijenoort 1967, 1902.

Shieber, Stuart M. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343, 1985.

van Heijenoort, J., editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*. Harvard University Press, Cambridge, Massachusetts, 1967.

Index