Metamathematics

Jacob Denson

August 30, 2023

Table Of Contents

Part I

Mathematical Logic

Chapter 1

What's Logic All About?

The mathematical method uses rigorous arguments to discover new facts about assumptions. Metamathematics turns the subject on its head, using the mathematical method to analyze mathematical methods themselves! Traditionally, metamathematics, or logic, was used by philosophers and mathematicians to justify valid forms of reasoning in mathematics. But in the early 20th century, a desire for rigour, combined with the invention of incredibly powerful mathematical tools, culimnated in a plague of foundational mathematical paradoxes in what seemed to be innocuous forms of reasoning. It was in the resulting firestorm where modern metamathematics was forged...

In the late 1800s, a German mathematician named Georg Cantor invented an entirely new way of thinking about mathematics. Instead of discussing particular numbers or shapes or functions, Cantor decided instead to talk about collections of mathematical objects, known as sets, and the ways these sets can be manipulated and described. In a flourish, he solved a problem which had troubled mathematicians for centuries. A number x is called *algebraic* if it solves an equation of the form

$$a_0 + a_1 x + \cdots + a_n x^n = 0,$$

for some rational numbers $a_0, \ldots, a_n$. Since the time of Archimedes, mathematicians had wondered whether all numbers were algebraic, or whether there exists non-algebraic numbers, which we call *transcendental*. Cantor answered this question negatively. Rather than finding an example of a trancendental number, Cantor produced a method of evaluating the size of *infinite* sets; he counted the set of all algebraic numbers, and counted the set of all numbers, and proved that there were far too many numbers for them all to be algebraic. But by creating set theory, a powerful linguistic framework for collecting mathematical concepts

together, Cantor changed the way mathematicians think about mathematics, opening the floodgates for mathematicians to encapsulate the entirety of mathematics, and here the paradoxes began to emerge:

Example (Burali-Forti Paradox (1897))**.** *Consider the set* Ω *of all ordinals. Then* Ω *is a well-ordered set, and hence is order isomorphic to some ordinal, and thus to a proper segment of itself. But no well-ordered set is order isomorphic to a proper subset of itself!*

Example (Cantor's Paradox (1899))**.** *For any set X, we can consider the powerset* 2^X*, which consists of all sets Y which are subsets of X. Cantor proved that the cardinality of* 2^X *is always strictly greater than the cardinality of X. But if X is the set of all mathematical objects, including sets, and sets whose objects are sets, and so on and so forth, then* $2^X \subset X$*, which implies the cardinality of* 2^X *is less than or equal to the cardinality of X, contradicting Cantor's theorem.*

Example (Russell's Paradox (1902))**.** *Consider the set of all sets which do not contain themselves, expressed in set theoretic notation as*

$$X = \{x : x \notin x\}$$

Russell's paradox rests on an innocent question: Is X an element of X? If $X \in X$*, then by definition of the set X, we conclude that* $X \notin X$*. So we are lead to believe that* $X \notin X$*. But then, by construction of X, we conclude X is in X after all! A more colloquial explanation of the paradox considers a town with a single barber, shaving everyone who does not shave themselves. Russell's paradox emerges when we ask who shaves the barber?*

Remark. These paradoxes are not disjoint from each other. Russell's paradox is the simplest set-theoretic paradox, for it relies on no advanced set theory, such as the theory of cardinals and ordinals. However, Cantor's paradox was discovered first, and is essentially equivalent. The proof of Cantor's theorem, that 2^X has larger cardinality than X for any set X, takes an arbitrary surjective map $f : X \to 2^X$, considers $Y = \{x \in X : x \notin f(x)\}$, and then considers y with $f(y) = Y$. We conclude that $y \in Y$ if and only if $y \notin Y$, which is a contradiction. If we replace an arbitrary set X with the set of all sets, then $2^X = X$, so if we take the identify function as f, then $Y = \{x \in X : x \notin x\}$, which is exactly the set involved in Russell's paradox. In fact, it was exactly when going through Cantor's paradox that Russell discovered his paradox.

Henri Poincaré saw the paradoxes as proof that set theory was a "plague on mathematics", which could only be cured when the theory was eradicated as an acceptable mathematical tool. But paradoxes were not limited to set-theoretic concepts. In their work on the Continuum hypothesis, Julius König and Jules Richard found paradoxes attacking our intuitive understanding of how we rigorously specify mathematical objects in the english language.

Example (Richard & König's Paradoxes (1905))**.** *The set of all english expressions is countable, because there are only finitely many expressions of a certain length, and the union of countably many disjoint finite sets is countable. Of particular interest is the set of english expressions describing positive real numbers, which we denote by* S*. The way we interpret these english expressions can be thought of as a function* $f : S \to \mathbb{R}$*, which takes some expressions* $s \in S$ *and tells us which number this expression corresponds to. Since* $\mathbb{R}$ *is uncountable, there are some real numbers which cannot be described in english. We call the numbers in* $f(X)$ *the* definable real numbers*. As the image of a countable set, the set of all definable real numbers can be ordered by some enumeration* $x_1, x_2, x_3, \ldots$*, and so Cantor's diagonal method allows us to take any enumeration of real numbers, and use the enumeration to construct a canonical real number which isn't in the enumeration. Let this undefinable real number be denoted x. We can then construct the expression s, which is*

The number constructed from Cantor's diagonal argument on the enumeration.

We would expect s to define the real number x, i.e. so that $x = f(s)$*. But this is clearly impossible, because x is not an element of* $f(X)$*.*

A similar paradox emerges if we let ω *be some well-ordering of the real numbers. Then we can consider the expression s, which is*

The least number not definable in english with respect to ω*.*

If we try and interpret the expression s using the usual semantics of English phrases, we conclude that the number $x = f(s)$ *is undefinable, which contradicts the fact that x* is *defined by the statement s. This paradox remains true if we limit S to expressions which have at most 200 symbols in them, so that the infinite size of S is not the problem here (this refinement is Berry's Paradox, 1906).*

It seems obvious that a definition is simply a description of the qualities of an object under consideration, but if this were true, there would be no problem

with the arguments above, so we are at an impasse. A precise definition of definability is a key discovery in metamathematics, from which we will obtain the beautiful results of Gödel and Tarski. A related class of paradoxes results from self-reference.

Example (Löb's Pardox (1955))**.** *Consider the Preposition B, which is true when $B \Rightarrow A$ is true. If B is true, then $B \Rightarrow A$ is true, so A is true. But this means we have proven that $B \Rightarrow A$ is true, so B is true, and we therefore conclude by inference that A is true. Since A was arbitrary, we can conclude that every logical statement is true from this form of argument!*

It can be argued that Löb's paradox fails because self referential statements are naturally circular, but Curry showed that this type of self reference emerges in much more subtle ways in naive set theory.

Example (Curry)**.** *For any property P, consider*

$$C = \{x : (x \in x) \Rightarrow P(x)\}$$

Then $C \in C$ holds if and only if $C \in C \Rightarrow P(C)$ holds. We must have $C \in C$, for if $C \notin C$, then $C \in C \Rightarrow P(C)$ holds vacantly, and so we conclude that $C \in C$. But this implies $C \in C \Rightarrow P(C)$, so $P(C)$ is true, irrespective of the content of the statement $P(C)$.

It became clear to the mathematicians of the early 1900s that the current state of logic was not sharp enough to fix and explain away the paradoxes of set theory. There are several approaches mathematicians went to fixing this. One approach, typified by the school of Russell and Whitehead, and exemplified in their book Principia Mathematica (1910-1913) uses *type theory* to restrict languages from self-referential structures. Another school, typified by Hilbert, is to focus on generating precise *axiom systems* for mathematical theories, such as Zermelo-Fraenkel set theory. Alternatively, a more radical approach, due to Brouwer and the intuitionist school, is to reject certain logical laws, like the principle of proof by contradiction, to ensure a consistent theory. The methods that were developed in the process of understanding these paradoxes have generated many useful mathematical perspectives and tools, which are useful to general mathematics.

1.1 Formal Systems

In order to analyze mathematics mathematically, a careful method must be employed to avoid making our reasoning circular. The main framework for con-

structing abstract models simulating the procedure of mathematics, while being clearly separated from mathematical technique, is the idea of a formal system. The main idea of this process is that the *form* of an argument contains all the necessary information required to validify and understand an argument. Formal systems theory takes solely the forms of a mathematical argument, and abstracts them; once an argument has been encoded in a formal system, it is just a sequence of symbols on a page. In this way, mathematical logic is effectively 'mathematical linguistics', studying the languages that a mathematicians reasons with.

To formalize the language of mathematicians, we must first estimate the procedure of of a mathematician at work. First, she accepts some fundamental statements as 'obviously true', known as axioms. Using previously agreed upon logical derivations, she obtains additional statements from the axioms. This describes completely the form of a mathematical argument. It is the basic principle of formal systems that the thought of process of a mathematician, while an important part of the mathematical process, is extraneous to the actual content of the mathematics the mathematician produces, and so statements and derivations contain all effective information to understand the mathematical process. Thus we need only model what a statement is, and what a logical derivation is, and we then have a toolkit for analyzing all of mathematics by mathematical means!

Both statements and derivations will be built from a deceptively simple system of mathematical objects. We take a set Λ, known as an *alphabet*, and consider *strings* over that alphabet, which are finite (possibly empty) sequences of elements in Λ. Strings have turned out to be the right formalization for the majority of mathematical logic (at the point of writing these notes, we mainly describe mathematics in papers, which consist of strings of alphabetical characters), and we shall find they suffice to express all the formal systems we shall consider in these notes.

Example. *If*

$$\Lambda = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \ldots, \mathsf{Z}, \mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots, \mathsf{z}, \textvisiblespace\}.$$

is the set of all 52 lower and upper case letters in the English alphabet, and we view the character $\textvisiblespace$ *as representing a blank space character, then an example of a stringle in this alphabet consists of both complete garble, like* $(\mathsf{A}, \mathsf{F}, \mathsf{c}, \mathsf{B}, \mathsf{V}, \mathsf{v})$*, as well as actual strings that occur in English literature, like*

$$(\mathsf{T}, \mathsf{o}, \textvisiblespace, \mathsf{b}, \mathsf{e}, \textvisiblespace, \mathsf{o}, \mathsf{r}, \textvisiblespace, \mathsf{n}, \mathsf{o}, \mathsf{t}, \textvisiblespace, \mathsf{t}, \mathsf{o}, \textvisiblespace, \mathsf{b}, \mathsf{e}).$$

It is customary to denote a string $(v_1, \ldots, v_n)$ *by "*$v_1 \ldots v_n''$*, or even* $v_1 \ldots v_n$ *if the statement isn't ambiguous, which can often be helped by using a different font to*

represent elements of Λ *then are used elsewhere. We also might write the blank space character as a usual blank space Thus we could write the garbled string as AFcBVv, and the second by* To␣be␣or␣not␣to␣be*, or To be or not to be if it is clear that blank spaces represent a single separator character.*

The concatenation of two strings $s = s_1 \dots s_n$ and $w = w_1 \dots w_m$, denoted by sw, is the string $s_1 \dots s_n w_1 \dots w_m$. If we view concatenation as an associative algebraic operation, then the set of all strings Λ^* is the smallest *monoid* containing Λ: it is essentially the standard construction of the free monoid with generators Λ. This is an interesting viewpoint which is useful in certain specialized areas of mathematical logic, like automata theory, though we will not touch on this viewpoint much here.

Since most of our languages will be constructed from putting basic strings together, we may wish to take complicated strings, such as

It was the best of times, it was the worst of times

and identify more basic components in them, such as worst, or best. A *substring* of a string s is a string t obtained from taking a continguous sequence of characters in s. More precisely, t is a substring of s if there exists two strings w and v such that $s = wtv$. A *prefix* is a substring occuring at the beginning of a string ($s = wt$), and a *suffix* is a substring occuring at the end ($s = tv$). A *language* is a subset of strings over an alphabet. It allows us to separate meaningful strings over the alphabet, like the string

The quick brown fox jumps over the lazy dog

from complete nonsense, like the string

iovuiaesfpauaauupewbpvapbuib

As should be expected, the English language, roughly speaking, is a mathematical language, at least, obtained from taking the subset of strings over the English alphabet which represent coherent English sentences – a language as we defined need not have a precisely defined structure, just an unambiguous way of interpreting whether any string is a member of the language.

In mathematical logic, the languages of study consist of the sets of meaningful formulae in some logical calculus, and various interesting families of sublanguages, such as the provable formulas in some deductive system, the semantically true formulas, or the satisfiable formulas. The construction of the sublanguage of

'provable formulas' is encapsulated in what is called a *formal system*. The most general definition consists of a language L over an alphabet Λ, a set of axioms, which form a subset of L, and a set of inference rules, pairs (Γ, s) where $\Gamma \subset L$ is the premises of the inference, and s is the conclusion. The *theorems* of a formal system are members of the smallest set such that

1. Every axiom is a theorem
2. If (Γ, s) is an inference rule, and all elements of Γ are theorems, then s is a theorem.

If X is a formal system, we shall let $\vdash_X s$ state that s is a theorem of X. Normally though, we will just write $\vdash s$, for the formal system is almost always clear from context. Just as algebra is the study of groups, rings, and fields, metamathematics is the exploration of the different formal systems we can use to model everyday mathematics.

1.2 Why Trust Mathematical Logic?

Before we get to the real work though, we need to settle an important question: How can we ensure the models we apply to analyze mathematics are robust enough to enable us to prove facts about real mathematics? David Hilbert's plan, along with the rest of the formalist school, was to construct a formal system powerful enough to describe all of the present mathematical systems, and one that could discuss itself, and prove itself consistent (without paradox). If such a system could be constructed, we could model all past, present, and future proofs of mathematics in this universal system, and we would be shielded from future paradoxes. Thus the model is proved robust by the fact that all proofs could be modelled in the system. Hilbert would have essentially reduced mathematics to abstract symbol pushing inside the system, but Hilbert did not see this as an issue; this symbol pushing is no different from the symbol pushing inside our minds when we solve a problem, albeit more explicit. However, regardless of whether you believe this approach would be valid, we shall find Hilbert's approach is doomed from the beginning, for no sufficiently advanced consistent formal system can prove itself consistent.

So how can we ensure that our formal systems give correct results about everyday mathematics? If you desire absolute facts, you will be dissapointed. It is unlikely that any of our physical models of the universe are completely accurate.

A physicists's models are ideals, carved from reality in all senses but experimental parameters. No model describes a systems evolution exactly, and it is myopic to suggest a model's perfection. In spite of this, physics still does a bloody good job! In metamathematics, we attempt to form a mathematical model of mathematical principles. Some principles are pinned down for examination, others lost. We hope this model has enough vitality to provide key insights into real-life mathematics. Whether the method is successful can only be determined by the correspondence between the results of metamathematics, and evidence in actual mathematics. So far, the results of metamathematics in provisional, and first order logic have not been proven inaccurate by mathematical innovation.

But even if you don't accept formal systems as a correct model of ordinary mathematics, mathematical logic is still of interest, because we can obtain rigorous theorems connecting the study of strings to the study of mathematical objects. This allows us to relate theorems about interesting axiom systems, like the theory of fields or the theory of vector spaces, to our theory of strings. Such results hold regardless of whether we think that strings model mathematics completely.

A source of confusion in physics is the stylistic treatment of assumptions as absolute facts. A physicist describes

$$\text{“ a planet moving according to the equation } \ddot{x} = -m/x^2 \text{ ”},$$

even if he is actually talking about the dynamical system whose evolution is described by the differential equation $\ddot{x} = -m/x^2$, which *models* the motion of a planet. Such expressions are unavoidable, since they make the study of mechanics much more viceral and appealing to intuition, whereas eschewing the natural language makes the formal equivalent dry to the bone. Keep this principle in mind as we begin to build models of logic. Every theorem we proved is only true in the model of mathematics, and must be judged separately for authenticity outside of the mathematical model we have created. We will avoid doing such evaluation here.

Chapter 2

Prepositional Logic

We shall begin our study of formal systems with propositional logic, the simplest formal system to analyze truth. To understand propositional logic, we construct a mathematical model, known as a formal language, which represents the language in which mathematics is performed. The formal language is then analyzed by common mathematical deduction rules. The standard formal language for logic is an analysis of strings, sequences of abstract symbols from a given alphabet. Strings represent mathematical statements; manipulating these strings models how a mathematician infers some mathematical statement from another. It is best to see the tool in action to understand its utility, so we proceed swiftly into the technicalities involved in the construction of the logic.

2.1 Syntax

Normally, a formal system makes colloquial speech, so each symbol in the alphabet provides a precise representation of some forms of colloquial speech. We begin with propositional logic, which models statements with sentences of mathematics which are composed of basic statements which are true or false, and independent of one another. Some statements are *atomic* in the propositional logic, because they cannot be divided into more base statements by means of the basic logical connectives, to be introduced soon. Socrates is a man is an atomic statement, as is Every woman is human. The statement Socrates is a man and every woman is a human is not atomic, for the statement consists of two separate statements, composed by the connective and. In English, every woman is a human can be broken into statements such as Julie is a human and Laura is a

human, yet propositional logic still considers this statement as atomic; the model we discuss in this chapter does not have the capability to understand such more complex connectives, which are discussed in the realm of predicate logic in the next chapter.

Let Λ be a set disjoint from $\{(,), \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$. These symbols will represent the atomic statements in our representation of propositional logic. The *propositional language with atoms in* Λ, denoted $\mathrm{SL}(\Lambda)$, is the smallest subset of strings over the alphabet $\Lambda \cup \{(,), \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$ such that

1. $\Lambda \subset \mathrm{SL}(\Lambda)$.
2. If $\phi, \psi \in \mathrm{SL}(\Lambda)$, then $(\neg\phi), (\phi \wedge \psi), (\phi \vee \psi), (\phi \Rightarrow \psi), (\phi \Leftrightarrow \psi) \in \mathrm{SL}(\Lambda)$.

An element of $\mathrm{SL}(\Lambda)$ is called a *formula* or *statement*.

Each *connective* of prepositional logic represents a certain linguistical form. Later on, the connection of symbols to meaning will become clear. For now, they are abstract symbols without intrinsic meaning. Viewing a formal system as meaningless symbol shifting is known as the *syntactical view* of a formal system. For now, the table below gives the interpretations we will later give for the logical connectives.

Connective	Name of Connective	Meaning of statement
$\neg\phi$	Negation	ϕ is *not* true
$\phi \wedge \psi$	Conjuction	ϕ *and* ψ both are true
$\phi \vee \psi$	Disjunction	either ϕ *or* ψ is true
$\phi \Rightarrow \psi$	Implication	*If* ϕ is true, *then* ψ is true
$\phi \Leftrightarrow \psi$	Bicondition	ϕ is true *if, and only if* ψ is true

Take care to notice that $\mathrm{SL}(\Lambda)$ is the *smallest* set constructed with the required axioms, in the same way that most 'smallest objects' exist in mathematics, because the intersection of sets satisfying the set of statements defining $\mathrm{SL}(\Lambda)$ also satisfy the statements. This property leads to the most useful proof method in the study of the syntax of a language: *structural induction*. Here is the principle as it applies to the language we have just introduced.

Theorem 2.1 (Structural Induction)**.** *Consider a predicate that can be applied to elements of SL*(Λ)*. Suppose that:*

- *The predicate is true of all elements of* Λ*.*
- *The predicate is true of* ϕ *and* ψ*, then the predicate is also true of* $\neg\phi, \phi \wedge \psi, \phi \vee \psi, \phi \Rightarrow \psi$*, and* $\phi \Leftrightarrow \psi$*.*

Then the predicate is true for all of $SL(\Lambda)$.

Proof. Let P be some predicate to consider. Note the set

$$K = \{\phi \in (\Lambda \cup \{(,), \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\})^* : P(\phi) \text{ is true}\}$$

is a set of strings satisfying the axioms (1) and (2) which define $\mathrm{SL}(\Lambda)$, so that we may conclude $\mathrm{SL}(\Lambda) \subset K$. □

The formulas of prepositional logic are just abstract sequences of symbols. They are not defined to have an intrinsic grammatical structure. To start working with this language however, we must proved that the language necessarily does have a grammatical structure. Since we are working over formulas of arbitrary complexity, structural induction will be the most useful method of proof.

Theorem 2.2. *Any sentence in* $SL(\Lambda)$ *contains as many left as right brackets.*

Proof. Any atom in Λ contains no left brackets, and no right brackets, and thus the same number of each. If ϕ and ψ have as many left brackets as right brackets, then so too does $(\neg\phi)$, and $(\phi \circ \psi)$, where $\circ \in \{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$. By structural induction, we have proved our claim. □

We need a more in depth theorem to correctly parse statements of propositional logic. If statements can be parsed in two different ways, they become ambiguous. For instance, what is the value of $2 + 7 - 5 - 4$? Is it

$$(((2 + 7) - 5) - 4) = 0$$

or

$$(2 + 7) - (5 - 4) = 8$$

In our language, we would hope that parenthesis allow us to remove ambiguity, so we know which order to apply logical operations. The study of syntax allows us to show that statements can be parsed uniquely, and once we have this understanding, we can work with the language much more fluidly, making the understanding of the semantics of the language much more clear. Equations must be understood before we calculate with them.

Theorem 2.3. *If* w *is a non-empty predix of* $\phi \in SL(\Lambda)$*, then* w *has at least as many left brackets as right brackets, and* $w = \phi$ *if and only if* w *has the same number of left and right brackets.*

Proof. We perform a structural induction. If $\phi \in \Lambda$, then $w = \phi$, and the theorem is trivial. Now suppose ϕ and ψ satisfy the theorem. We now prove that $(\neg\phi)$ and $(\phi \circ \psi)$ satisfy the theorem:

- w is a prefix of $(\neg\phi)$: March through all cases. The string w is either equal to $($, or $(\neg$, or $(\neg v$, where v is a prefix of ϕ, or $(\neg v)$. Using the inductive hypothesis, we see that w always has at least the same number of left brackets than right brackets. On the other hand, suppose that w has exactly the same number; we immediately see the only possible case where this is true is when $w = (\neg\phi)$, which proves the result.

- w is a prefix of $(\phi \circ \psi)$, for some $\circ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$: Continue marching through all the possible cases. We can either write w as $($, or $(v$, or $(\phi\circ$, or $(\phi \circ w$, or $(\phi \circ \psi)$. In each case, w always has at least the same number of left brackets than right brackets, and the only case where w has the same number is if $w = (\phi \circ \psi)$. □

Corollary 2.4. *Every string in SL(Λ) can be written uniquely as an atom Λ, or $(\neg\phi)$ and $(\phi \circ \psi)$, where ϕ and ψ are elements of SL(Λ). The unique connective in the representative is known as the* principal connective *of the statement.*

Proof. Such representations trivially exist by the construction of SL(Λ). Suppose we have two representations. If one of the representations is an element of Λ, the other representation must have length one, and is therefore equal to the other representation. If we have two representations $(\neg\phi) = (\neg\psi)$, then by chopping off symbols, we conclude that $\phi = \psi$. It is impossible to have two distinct representations $(\phi \circ \psi) = (\neg\phi)$, for no element of SL$(\Lambda)$ begins with $\neg$. Finally, suppose we have two representations $(\phi \circ \psi) = (\eta \circ \nu)$. Then either ϕ is a prefix of η, or η is a prefix of ϕ, and both have balanced brackets, which implies $\phi = \eta$, and by chopping letters away, we conclude $\psi = \nu$. □

Given an arbitrary language constructed recursively, it is in general very difficult to verify if a language is ambiguous. In the theory of computability, we discover that there is no general algorithm with which we can prove any given recursively constructed language is ambiguous. However, the particular languages we use to represent formal systems tend to have a fairly simple syntax, and therefore we can prove that these languages are ambiguous with relative ease.

Because we have unique parsing, we can define functions on terms of propositional logic recursively, focusing only on the principal connecting in the definition.

For instance, given two sets $\Lambda = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, and $\Gamma = \{\mathsf{P}, \mathsf{Q}, \mathsf{R}\}$, we can obtain a natural bijection from $\mathrm{SL}(\Lambda)$ to $\mathrm{SL}(\Gamma)$ by extending the map

$$\mathsf{X} \mapsto \mathsf{P} \quad \mathsf{Y} \mapsto \mathsf{Q} \quad \mathsf{Z} \mapsto \mathsf{R}$$

by the recursive definition

$$f(\phi \circ \psi) = f(\phi) \circ f(\psi) \quad f(\neg\phi) = \neg f(\phi)$$

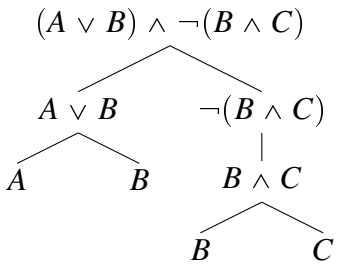
Such a map is well defined on all of $\mathrm{SL}(\Lambda)$ by the corollary, and can be shown by a certain structural induction to be bijective. It is easy to see that this map preserves the semantic properties of propositional logic we will soon define, so that all systems of propositional logic are essentially equivalent.

It is useful to have a visual specification of the unique way to parse a formulae. Given a formula ϕ, define the *parse tree* of ϕ inductively by the following process:

- If ϕ is atomic, then the parse tree of ϕ consists of a single node, ϕ itself.
- If $\phi = \eta \circ \nu$, where $\circ$ is a binary connective, then the parse tree has a parent node ϕ, descending into two subtrees, the first of which being the parse tree of η, and the second of which the parse tree for ν.
- If $\phi = \neg\psi$, then the parse tree has a parent node ϕ, with a single edge descending into the parse tree of ψ.

The *complexity* of a given formula ϕ is the height of its parse tree. A *subformula* of a formula ϕ is a formula associated to one of the nodes in the parse tree of ϕ. It is easy to see that they are the only substrings of ϕ which are valid formulas in the language. An occurence of a formula ψ in a formula ϕ is a node in the parse tree of s whose associated string is ψ.

Example. *The parse tree for* $(A \vee B) \wedge \neg(B \wedge C)$ *is*



Thus the formula has complexity 3*, for the longest branch in the parse tree has length three. The subformulas of the string are simply the nodes in the tree. The tree also tells us that there are two occurences of B in the formula, whereas only one occurence of $(B \wedge C)$.*

Before we finish with the study of the syntax of propositional logic, it is interesting to discuss a less natural, but syntactically simpler method of forming sentences, called *Polish notation*, after its inventor, the Polish logician Jan Łukasiewicz. Rather than writing connectives in *infix notation*, like $(u \wedge v)$ and $(u \Rightarrow v)$, we use *prefix notation*, writing these sentences as $\wedge uv$ and $\Rightarrow uv$. Surprisingly, we do not need brackets to parse statements anymore. As a temporary notation, say two strings s and w are *comparable* if one is the prefix of the other.

Lemma 2.5. *If $\phi_1, \ldots, \phi_n, \psi_1, \ldots, \psi_m$ are formulas in Polish notation, and $\phi_1 \ldots \phi_n$ is comparable to $\psi_1 \ldots \psi_m$, then $n = m$, and $\phi_i = \psi_i$ for each i.*

Proof. We prove by induction on the number of characters in $\phi_1 \ldots \phi_n$. If $\phi_1 = \neg\eta$, then the first character of ψ_1 must be nonempty, and equal to $\neg\nu$ for some formula ν. It then follows that $\eta\phi_2 \ldots \phi_n$ is comparable to $\nu\psi_1 \ldots \psi_m$, so $n = m$, and $\eta = \nu$, $\phi_i = \psi_i$, and it follows that $\phi_1 = \psi_1$ as well. Similarily, if $\phi_1 = \circ\eta_1\eta_2$, then $\psi_2 = \circ\nu_1\nu_2$, and by removing $\circ$, we find $\eta_1 = \nu_1$, $\eta_2 = \nu_2$, and $\phi_i = \psi_i$, and then $\phi_1 = \psi_1$ is easy to see as well. □

Theorem 2.6. *Every character in a formula in polish notation begins a unique subformula in polish notation.*

Proof. We prove by structural induction. If $\phi \in \Lambda$, the theorem is trivial. If $\phi = \neg\psi$, then $\neg$ begins a unique subformula, because if $\neg\eta$ and $\neg\nu$ are subformulas of ϕ, then $\neg\eta$ and $\neg\nu$ are comparable, hence $\eta = \nu$. By induction, every character in ψ begins a unique subformula. If $\phi = \circ\eta_1\eta_2$, then we know that $\circ$ begins a unique subformula (using the last lemma), and there is no way to find a subformula ψ which begins in η_1 and ends in η_2, hence we may apply induction to show every other character begins a unique subformula. □

This theorem says that Polish notation is an especially efficient language with which to discuss compositions of terms in sentential logic – it is the most efficient formulation provided the subterms of sentential logic are properly represented in the calculus. It is easy to check that these properties continue to hold when we add a more complicated syntax in predicate logic, with functions $\mathsf{f}(t_1, \ldots, t_n)$ being denoted as $\mathsf{f}t_1 \ldots t_n$, predicates $\mathsf{P}(t_1, \ldots, t_n)$ as $\mathsf{P}t_1 \ldots t_n$, and quantifiers denoted $\exists x\phi$ and $\forall x\phi$.

2.2 Semantics

We can understand the discussion in the last section without any understanding of what symbols mean. Now we want to interpret the symbols, giving the symbols meaning. A basic semantic method is to define whether a statement is 'true'. Define a *truth assignment* on a set Λ to be a map $f : \Lambda \to \{\top, \bot\}$, where $\top$ and $\bot$ are two arbitrarily chosen representatives of truth and falsity. The notation suggests that we will be taking truth assignments on the set of propositional variables in the propositional language $\mathrm{SL}(\Lambda)$, and this will define the semantics of propositional logic.

Example. *A* Boolean function *is an assignment over the domain* $\Lambda = \{\top, \bot\}^n$. *It is common to define such functions by truth tables. For a given Boolean function, we form a table with* $n + 1$ *columns, and* 2^n *rows. In each row, we fill out a particular element of* $\{\top, \bot\}^n$, *and in the last column, the value of image of the truth assignment under* f. *One may combine multiple n-ary truth functions into the same table for brevity. As an example, we define the Boolean functions* $H_\wedge, H_\vee, H_\Rightarrow$, $H_\Leftrightarrow$, *and* $H_\neg$.

x	y	$H_\wedge(x,y)$	$H_\vee(x,y)$	$H_\Rightarrow(x,y)$	$H_\Leftrightarrow(x,y)$	$H_\neg(x)$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$

These functions express the semantic interpretation of the corresponding operators in propositional logic.

We may extend truth assignments $f : \Lambda \to \{\top, \bot\}$ naturally to an assignment $f : \mathrm{SL}(\Lambda) \to \{\top, \bot\}$. This is analogous to how homomorphisms between two rings R and S naturally extend to homomorphisms between the polynomial rings $R[X]$ and $S[X]$. Here we construct the assignment recursively. Let $f : \Lambda \to \{\top, \bot\}$ be an arbitrary truth assignment. Define

$$f(\neg\phi) = H_\neg(f(\phi)) \qquad f(\phi \circ \psi) = H_\circ(f(\phi), f(\psi))$$

Because of our study of syntax, it is easy to see that f is a well defined function on all terms of $\mathrm{SL}(\Lambda)$. We say an arbitrary term $\phi \in \mathrm{SL}(\Lambda)$ is a *tautology*, if, for any truth assignment f on Λ, $f(\phi) = \top$. ϕ is a *contradiction* if $f(\phi) = \bot$ for any truth assignment f. A statement which is neither a tautology nor a

contradiction is known as a *contingent term*. We summarize the statement "ϕ is a tautology" by $\vDash \phi$. We say a family of term $\phi_1, \ldots, \phi_n$ *semantically implies* ψ, written $\phi_1, \ldots, \phi_n \vDash \psi$, if $f(\psi) = \top$ whenever $f(\phi_1) = \cdots = f(\phi_n) = \top$.

Suppose that we wish to verify whether $\phi \in \mathrm{SL}(\Lambda)$ is a tautology. Let $A_1, \ldots, A_n \in \Lambda$ be all the atomic statements which occur in ϕ. Define a boolean function $g : \{0,1\}^n \to \{0,1\}$, with

$$g(x_1 \ldots, x_n) = f^{(x_1,\ldots,x_n)}(\phi)$$

where $f^{(x_1,\ldots,x_n)}$ is the truth assignment obtained by mapping A_i to x_i. This is well defined, because if h and k are two truth assignments which agree on the A_i, then they agree at ϕ. If f is an arbitrary truth assignment, then

$$g(f(A_1), \ldots, f(A_n)) = f(\phi)$$

which can be seen from an easy structural induction. Thus ϕ is a tautology if and only if $g(x_1, \ldots, x_n) = \top$ for all choices of x_i. Therefore one need only construct the truth table of g to confirm whether s is a tautology or not. To prevent errors, it is best to construct a truth table containing all subformulas of ϕ, so that one can verify that calculations are consistent with other calculations. This may be done side by side, in the same table.

Example. *For any variable $A \in \Lambda$, $A \vee \neg A$ is a tautology, $A \wedge \neg A$ is a contradiction, and $\neg A$ is contingent, which is verified by the truth table*

A	$A \vee \neg A$	$A \wedge \neg A$	$\neg A$
$\top$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$	$\top$

The first formula is an instance of the law of excluded middle.

Example. *Let $\phi \vDash \psi$ be a tautology, and suppose that ϕ and ψ have no variables in common. Then we can conclude that ϕ is a contradiction, or ψ is a tautology, for if there is a truth assignment f with $f(\phi) = \top$, and a truth assignment g on ψ with $g(\psi) = \bot$, then we may combine the truth assignments to create a truth assignment h in which $h(\phi) = \top$ and $h(\psi) = \bot$, and then $h(\phi \Rightarrow \psi) = \bot$.*

Because of the truth table construction, there is an algorithm for determining if any given term $\phi \in \mathrm{SL}(\Lambda)$ is a tautology. However, there is a catch: if a term ϕ contains n variables, the algorithm we have constructed will take $\Omega(2^n)$ steps to

compute whether ϕ is a tautology. Determining whether a term with 150 variables is a tautology will take more steps than atoms in the observable universe! The $\mathbf{P} \neq \mathbf{NP}$ conjecture states that there is no algorithm which takes $O(n^K)$ determining if a statement of $\mathrm{SL}(\Lambda)$ is a tautology for any integer K, so that there is a fundamental limit to the efficiency of determining whether a term is a tautology. There are more efficient methods for determining whether certain subfamilies of terms of propositional logic are tautologies, but it is doubtful whether we can find an efficient algorithm to determine if an arbitrary formula is satisfiable.

Theorem 2.7 (Semantic Modus Ponens)**.** *If $\vDash \phi$ and $\phi \vDash \psi$, then $\vDash \psi$.*

Proof. Let f be a truth assignment. Then $f(\phi) = \top$ and

$$f(\phi \Rightarrow \psi) = H_{\Rightarrow}(f(\phi), f(\psi)) = H_{\Rightarrow}(\top, f(\psi)) = \top$$

This holds only when $f(\psi) = \top$. □

The next theorem relies on a useful string manipulation technique which shall later prove a useful formalism. If $\phi \in \Lambda^*$, $A = (A_1, \ldots, A_n)$ is formed from distinct letters of Λ, and $\psi = (\psi_1, \ldots, \psi_n) \in \Lambda^*$, then we shall let

$$\phi[\psi_1/A_1, \ldots, \psi_n/A_n] = \phi[\psi/A]$$

be the *substitution* of ϕ, denoting the string in Λ^* obtained from swapping all occurences of A_i with ψ_i.

Theorem 2.8. *If $\vDash \phi$, then $\vDash \phi[\psi/A]$*

Proof. Consider a truth assignment f. We shall define another truth assignment $\tilde{f}$ such that $f(\phi[\psi/A]) = \tilde{f}(\phi)$ for all v. Define $\tilde{f}(A_i) = f(\psi_i)$, and if $y \notin x$, define $\tilde{f}(y) = f(y)$. Our base case, where ϕ is a variable, satisfies the claim by construction. Then, by induction, if $\psi = (\eta \circ \nu)$, then

$$\tilde{f}(\psi) = H_{\circ}(\tilde{f}(\eta), \tilde{f}(\nu)) = H_{\circ}(f(\eta[\psi/A]), f(\eta[\psi/A])) = f(\phi[\psi/A])$$

A similar proof answers the case where $\phi = (\neg\eta)$. Now since ϕ is a tautology, we conclude that $f(\phi[\psi/A]) = \tilde{f}(\phi) = \top$, so $\phi[\psi/A]$ is a tautology. □

Corollary 2.9. *If $\eta_1, \ldots, \eta_n \vDash \nu$, then $\eta_1[\psi/A], \ldots, \eta_n[\psi/A] \vDash \nu[\psi/A]$.*

2.3 Truth Functional Completeness

We hope that propositional logic can model all notions of truth, such that all truth functions can be formed from our original set. Here we argue why our logic can model all such notions, provided we have infinitely many propositional variables. Let X be a set of boolean functions. The *clone* of X is the smallest set containing X and all projections $\pi_k : \{0,1\}^n \to \{0,1\}$ onto the kth coordinate, and in addition, if $g : \{0,1\}^n \to \{0,1\}$ and $f_1, \ldots, f_n : \{0,1\}^m \to \{0,1\}$ are in the clone, then so is $g(f_1, \ldots, f_n) : \{0,1\}^m \to \{0,1\}$. Λ is *truth functionally complete* if its clone is the set of all boolean functions.

Example. *$\{H_\neg, H_\wedge, H_\vee\}$ is a truth functionally complete, since every formula can be put in* conjunctive normal form*. Given an arbitrary $f : \{\top, \bot\}^n \to \{\top, \bot\}$, we write*

$$f(x_1, \ldots, x_n) = \bigvee_{\substack{(y_1,\ldots,y_n)\in\{0,1\}^n \\ f(y_1,\ldots,y_n)=\top}} \bigwedge_{i=1}^{n} H_\Leftrightarrow(x_i, y_i)$$

where we define $\bigcirc_{i=1}^{n} f_i = H_\circ(f_n, \bigcirc_{i=1}^{n-1} f_i)$ for any connective $\circ$. Since

$$H_\Leftrightarrow(x,y) = H_\wedge(H_\Rightarrow(x,y), H_\Rightarrow(y,x)) = H_\wedge(H_\vee(H_\neg(x), y), H_\vee(H_\neg(y), x))$$

we find that $\{H_\neg, H_\wedge, H_\vee\}$ is truth functionally complete. These connectives can be further reduced, by noticing that

$$H_\wedge(x,y) = H_\neg(H_\vee(H_\neg(x), H_\neg(y)))$$

a truth functional form of Boole's inequality, implying $\{H_\neg, H_\vee\}$ is truth functionally complete. We can also consider a disjunctive normal form *of any Boolean function, which we leave to the reader to formulate.*

Say a formula ϕ is *satisfiable*, if there is some truth assignment f such that $f(\phi) = \top$. It is a standard computational problem to verify whether such a formula is satisfiable, as a great many problems can be reduced to satisfiability. For instance, say one wishes to verify whether a graph (V, E) is m colorable (that is, there is a function $f : V \to \{1, \ldots, m\}$ such that if $(v, w) \in E$, $f(v) \neq f(w)$). For each vertex $v \in V$ and color $i \in \{1, \ldots, m\}$, let v_i be a variable, which we interpret to be true if v is coloured with color i. A graph is colorable if and only if the statement

$$\bigwedge_{v\in V}\left(\bigvee_{i=1}^{m} v_i\right) \wedge \bigwedge_{(v,w)\in E}\left(\bigwedge_{i=1}^{m}(\neg v_i \vee \neg w_i)\right) \wedge \bigwedge_{v\in V}\left(\bigwedge_{i,j=1}^{m}(v_i \vee \neg v_j)\right)$$

is satisfiable. The first big clause says that some color is assigned to each vertex, the last that the color is unique. The middle clause is the coloring constraint. If a formula is in disjunctive normal form, it is algorithmically easy to verify whether the formula is satisfiable. We just need to check whether one of the disjunctive clauses is consistent, which can be done in a time proportional to the size of the formula. Checking whether a formula is in conjunctive normal form is much more difficult – in fact, in computability theory one discovers that almost all interesting problems can be reduced to a satisfiability problem without decreasing the efficiency of the task, so if it was easy to determine if a CNF is solvable, then we could solve a great many seemingly difficult problems in an easy manner. The **P** = **NP** conjecture implies that there is no polynomial time computable way of turning conjunctive normal forms to equivalent disjunctive normal forms. The standard conversions which exist increase the size of the formula exponentially.

Example. *The mathematician Henry M. Sheffer found a single truth function which is truth functionally complete. Consider the* Sheffer stroke $x|y$*, also known as* NAND*, defined by the truth table*

x	y	$H_{\mid}(x,y)$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	$\top$	$\bot$

Then $H_{\neg}(x) = H_{\mid}(x,x)$*, and* $H_{\vee}(x,y) = H_{\mid}(H_{\neg}(x), H_{\neg}(y))$*, which implies, since this set is truth functionally complete, that the Sheffer stroke is truth functionally complete.*

The previous example is incredibly important to circuit design. Logical statements can be represented by boolean functions. Since all truth functions can be built from the Sheffer stroke, we need only make an atomic circuit for the Sheffer stroke, and then all other circuits are constructed by combining Sheffer strokes together.

2.4 Deduction

When mathematicians want to derive whether a statement is true, they do not construct truth functions and take a truth table for the function. This would be computationally infeasible, and would not aid in understanding *why* the statement

is true. Instead, they provide a proof of the result. Here we shall provide the mechanics for modelling a mathematical argument. We will show that the method of truth tables and arguments are equivalent – a statement is a tautology if and only if it can be proved. This is known as a *completeness result*, for it says that our semantic understanding of a theory is the same as our deductive understanding.

First, we thin out the connectives in our theory. Since $\Rightarrow$ and $\neg$ are truth functionally complete, we can consider a system consisting only of these connectives, and reinterpret other formulas as semantically equivalent formulas in the reduced theory. Next, we define the *theorems* of $\mathrm{SL}(\Lambda)$, which are elements of the smallest set of terms such that

1. Any axiom is a theorem, which are statements of the form

$$
\begin{aligned}
&(A1) && \phi \Rightarrow (\psi \Rightarrow \phi) \\
&(A2) && (\phi \Rightarrow (\psi \Rightarrow \eta)) \Rightarrow ((\phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow \eta)) \\
&(A3) && (\neg\phi \Rightarrow \neg\psi) \Rightarrow ((\neg\phi \Rightarrow \psi) \Rightarrow \phi)
\end{aligned}
$$

2. Modus Ponens holds in our system. If $\phi \Rightarrow \psi$ and ϕ are theorems, then ψ is a theorem. When we apply modus ponens, we may write that the theorem was obtained by (MP).

We shall write $\vdash \phi$ to state that ϕ is a theorem.

For the terms of $\mathrm{SL}(\Lambda)$, the 'smallness' characterization of the terms of $\mathrm{SL}(\Lambda)$ gives us the method of structural induction. For theorems, the smallness criterion gives us a abstract notion of a 'proof'. A statement ϕ is a theorem of $\mathrm{SL}(\Lambda)$ if and only if there is a sequence of formulae $(\psi_1, \ldots, \psi_n)$ such that $\psi_n = \phi$, and each ψ_i is either an axiom, or is obtained from some ϕ_j and ϕ_k by modus ponens, where $j, k < i$. This sequence is known as a *proof*. Often, we list a proof from top to bottom, where we reference how we obtained each element of the sequence alongside the proof. A proof is a certificate guaranteeing that a statement is a theorem of $\mathrm{SL}(\Lambda)$, and finding a proof is the only constructive way to guarantee that a statement is a theorem.

Example. *Let us construct a proof of* $\vdash \phi \Rightarrow \phi$*, for any* $\phi \in SL(\Lambda)$.

$\phi \Rightarrow \phi$	
1. $(\phi \Rightarrow ((\phi \Rightarrow \phi) \Rightarrow \phi))$	*(A1)*
2. $(\phi \Rightarrow ((\phi \Rightarrow \phi) \Rightarrow \phi)) \Rightarrow ((\phi \Rightarrow (\phi \Rightarrow \phi)) \Rightarrow (\phi \Rightarrow \phi))$	*(A2)*
3. $((\phi \Rightarrow (\phi \Rightarrow \phi)) \Rightarrow (\phi \Rightarrow \phi))$	*(1),(2),(MP)*
4. $(\phi \Rightarrow (\phi \Rightarrow \phi))$	*(A1)*
5. $\phi \Rightarrow \phi$	*(3),(4),(MP)*

In future proofs, we shall be able to use $\vdash \phi \Rightarrow \phi$ *implicitly, since we now know the statement can be proved in any of its forms by copying and pasting this proof, combined with a substitution. We will denote its application by* (I).

In mathematics, we often work in logical systems where additional axioms are introduced to the system. In group theory, we assume that operations are associative. In geometry, we assume there is a line between any two points. To perform mathematics, we add additional axioms to logic, and prove results from these axioms. If Γ is a set of sentences in $\mathrm{SL}(\Lambda)$, then we may consider each member of Γ to be an axiom. We write $\Gamma \vdash \phi$ if one may prove ϕ assuming all formulae in Γ have already been proved. That is, we may write a sequence $(\psi_1, \ldots, \psi_n)$, where $\psi_n = \phi$, and each ψ_i is either an axiom, an element of Γ, or is obtained by modus ponens from previous elements of the sequence.

Theorem 2.10 (Deduction Theorem)**.** *If* $\Gamma \cup \{\phi\} \vdash \psi$*, then* $\Gamma \vdash \phi \Rightarrow \psi$.

Proof. We prove the theorem by induction of the size of the proof of ψ. Consider a particular proof $(\eta_1, \ldots, \eta_n)$ of ψ from $\Gamma \cup \{\phi\}$. Suppose that $n = 1$. Then $\eta_1 = \psi$, and η must either be an axiom, an element of Γ, or equal to ψ. In the first and second case, the proof is equally valid in Γ, and so $\Gamma \vdash \phi \Rightarrow \psi$ follows from the axiom $(\psi \Rightarrow (\phi \Rightarrow \psi))$. If $\phi = \psi$, Then we have shown that $\vdash \phi \Rightarrow \psi$ (this is the identity rule we just proved), so obviously $\Gamma \vdash \phi \Rightarrow \psi$. Now we consider the problem proved for $m < n$. $\eta_n = \psi$ is either an axiom, an element of Γ, equal to ϕ, or proved by modus ponens from $\eta_i = (\eta_j \Rightarrow \psi)$, where $j < i$. We have already justified the constructions of all cases but the last. By induction, $\Gamma \vdash \phi \Rightarrow (\eta_j \Rightarrow \psi)$ and $\Gamma \vdash \phi \Rightarrow \eta_j$. But $(\phi \Rightarrow (\eta_j \Rightarrow \psi)) \Rightarrow ((\phi \Rightarrow \eta_j) \Rightarrow (\phi \Rightarrow \psi))$ is an axiom, so $\Gamma \vdash \phi \Rightarrow \psi$. □

The deduction theorem is constructive, in the sense that it gives an algorithm to compute any proof of $\Gamma \cup \{\phi\} \vdash \psi$ from a proof of $\Gamma \vdash \phi \Rightarrow \psi$. What's more, if the

proof has length n, then the new proof has length $O(n)$, so the deduction theorem is a polynomial time computable reduction. For this reason, we often utilize the deduction theorem when describing proofs in the formal system. Similarily, given any proof of a particular statement, we can introduce it as a future rule, since there is a constructive method to introduce that particular statement in any future proof: copy and paste the old proof into the new proof. This saves the descriptions of the formal proofs in this book from getting too unwiedly.

Example. *For any statements ϕ and ψ, $\{\phi \Rightarrow \psi, \psi \Rightarrow \eta\} \vdash \phi \Rightarrow \eta$. This follows from a basic application of (A2). But this implies the two cut rules, that*

$$\vdash (\phi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \eta) \Rightarrow (\phi \Rightarrow \eta))$$
$$\vdash (\psi \Rightarrow \eta) \Rightarrow ((\phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow \eta))$$

these statements are a little bit more tricky to prove without the deduction theorem, though technically the proof of the deduction theorem gives a constructive way to obtain such a proof. We denote an application of these rules as (CUT).

Example. *Let us prove the double negation elimination axiom, $\vdash \neg\neg\phi \Rightarrow \phi$ by the deduction theorem.*

$\neg\neg\phi \Rightarrow \phi$	
1. $\neg\neg\phi$	
2. $(\neg\phi \Rightarrow \neg\neg\phi) \Rightarrow ((\neg\phi \Rightarrow \neg\phi) \Rightarrow \phi)$	*(A3)*
3. $(\neg\neg\phi) \Rightarrow (\neg\phi \Rightarrow \neg\neg\phi)$	*(A1)*
4. $(\neg\phi \Rightarrow \neg\neg\phi)$	*(1),(3),(MP)*
5. $(\neg\phi \Rightarrow \neg\phi) \Rightarrow \phi$	*(2),(4),(MP)*
6. $\neg\phi \Rightarrow \neg\phi$	*(I)*
7. ϕ	*(5),(6),(MP)*
8. $\neg\neg\phi \Rightarrow \phi$	*(1-7),(DT)*

In future proofs, application of the statement will be denoted $(\neg\neg E)$. Now lets

prove negation introduction, $\vdash \phi \Rightarrow \neg\neg\phi$.

$\phi \Rightarrow \neg\neg\phi$	
1. $\neg\neg\neg\phi \Rightarrow \neg\phi$	$(\neg\neg E)$
2. ϕ	
3. $(\neg\neg\neg\phi \Rightarrow \neg\phi) \Rightarrow ((\neg\neg\neg\phi \Rightarrow \phi) \Rightarrow \neg\neg\phi)$	*(A3)*
4. $(\neg\neg\neg\phi \Rightarrow \phi) \Rightarrow \neg\neg\phi$	*(1), (3), (MP)*
5. $\phi \Rightarrow (\neg\neg\neg\phi \Rightarrow \phi)$	*(A1)*
6. $\neg\neg\neg\phi \Rightarrow \phi$	*(2),(5),(MP)*
7. $\neg\neg\phi$	*(4),(6),(MP)*
8. $\phi \Rightarrow \neg\neg\phi$	*(2-7), (DT)*

We shall denote this rule $(\neg\neg I)$.

Example. *Lets prove* $\neg\phi \vdash \phi \Rightarrow \psi$*, by proving* $\neg\phi, \phi \vdash \psi$.

$\neg\phi \Rightarrow (\phi \Rightarrow \psi)$	
1. $\neg\phi$	
2. ϕ	
3. $(\neg\psi \Rightarrow \neg\phi) \Rightarrow ((\neg\psi \Rightarrow \phi) \Rightarrow \psi)$	*(A3)*
4. $\neg\phi \Rightarrow (\neg\psi \Rightarrow \neg\phi)$	*(A1)*
5. $\neg\psi \Rightarrow \neg\phi$	*(1),(4), (MP)*
6. $(\neg\psi \Rightarrow \phi) \Rightarrow \psi$	*(3),(5),(MP)*
7. $\phi \Rightarrow (\neg\psi \Rightarrow \phi)$	*(A1)*
8. $\neg\psi \Rightarrow \phi$	*(2),(7),(MP)*
9. ψ	*(6),(8),(MP)*
10. $\phi \Rightarrow \psi$	*(2-9),(DT)*
11. $\neg\phi \Rightarrow (\phi \Rightarrow \psi)$	*(1-10), (DT)*

This is a proof of the law of contradiction*, denoted* (LC).

Example. *Consider the following proof.*

$(\phi \Rightarrow \psi) \Rightarrow (\neg\psi \Rightarrow \neg\phi)$	
1. $\phi \Rightarrow \psi$	
2. $\neg\psi$	
3. $\neg\psi \Rightarrow (\neg\neg\phi \Rightarrow \neg\psi)$	*(A1)*
4. $\neg\neg\phi \Rightarrow \neg\psi$	*(2),(3),(MP)*
5. $(\neg\neg\phi \Rightarrow \neg\psi) \Rightarrow ((\neg\neg\phi \Rightarrow \psi) \Rightarrow \neg\phi)$	*(A3)*
6. $(\neg\neg\phi \Rightarrow \psi) \Rightarrow \neg\phi$	*(4),(5),(MP)*
7. $\neg\neg\phi$	
8. $\neg\neg\phi \Rightarrow \phi$	$(\neg\neg E)$
9. ϕ	*(7),(8),(MP)*
10. ψ	*(1),(9),(MP)*
11. $\neg\neg\phi \Rightarrow \psi$	*(7-10), (DT)*
12. $\neg\phi$	*(6),(11), (MP)*
13. $\neg\psi \Rightarrow \neg\phi$	*(2),(12),(MP)*
11. $(\phi \Rightarrow \psi) \Rightarrow (\neg\psi \Rightarrow \neg\phi)$	*(1-10), (DT)*

This is the law of contraposition (LCP).

Example. *Lets prove* $\vdash (\phi \Rightarrow \psi) \Rightarrow ((\neg\phi \Rightarrow \psi) \Rightarrow \psi)$.

$(\phi \Rightarrow \psi) \Rightarrow ((\neg\phi \Rightarrow \psi) \Rightarrow \psi)$	
1. $\phi \Rightarrow \psi$	
2. $(\phi \Rightarrow \psi) \Rightarrow (\neg\psi \Rightarrow \neg\phi)$	*(LCP)*
3. $\neg\psi \Rightarrow \neg\phi$	*(1),(2),(MP)*
4. $\neg\phi \Rightarrow \psi$	
5. $(\neg\psi \Rightarrow \neg\neg\phi)$	*(4),(LCP)*
6. $\neg\neg\phi \Rightarrow \phi$	$(\neg\neg E)$
7. $(\neg\psi \Rightarrow \neg\neg\phi) \Rightarrow ((\neg\neg\phi \Rightarrow \phi) \Rightarrow (\neg\psi \Rightarrow \phi))$	*(CUT)*
8. $(\neg\neg\phi \Rightarrow \phi) \Rightarrow (\neg\psi \Rightarrow \phi)$	*(5),(7),(MP)*
9. $\neg\psi \Rightarrow \phi$	*(6),(8),(MP)*
10. $(\neg\psi \Rightarrow \neg\phi) \Rightarrow ((\neg\psi \Rightarrow \phi) \Rightarrow \psi)$	*(A3)*
11. $(\neg\psi \Rightarrow \phi) \Rightarrow \psi$	$(3), (10), (MP)$
12. ψ	*(9), (11), (MP)*
13. $(\neg\phi \Rightarrow \psi) \Rightarrow \psi$	$(2-9), (DT)$
14. $(\phi \Rightarrow \psi) \Rightarrow ((\neg\phi \Rightarrow \psi) \Rightarrow \psi)$	$(1-10), (DT)$

We have essentially argued that $(\phi \vee \neg\phi) \Rightarrow \psi$ *implies* ψ.

Proofs give us a constructive way to verify whether a theorem is true in propositional logic, but it seems much more tricky to show that a theorem *cannot* be proved. Fortunately, semantic truth provides a simple invariant to decide if a theorem isn't provable.

Theorem 2.11 (Soundness of Propositional Logic)**.** *If* $\vdash \phi$*, then* $\models \phi$.

Proof. A trivial structural induction. □

This theorem shows that there are some statements which are not provable in our system, because there are some statements which are not tautologies. In fact, we know that if ϕ is provable, then $\neg\phi$ is not provable, for otherwise we could conclude that $\phi \wedge \neg\phi$, which is certainly not consistent. We call an axiom system like this *absolutely consistent*. We shall show that all tautologies are provable, which shows the system is *complete*. A complete system is effectively one in which all theorems which were meant to be able to be proved, are able to be proved – it is a justification that we have supplied enough axioms and deductive rules to fully satisfy the semantic meaning of a formal system.

Lemma 2.12. *Let $A_1, \ldots, A_n \in \Lambda$ be the family of all atomic propositions which occur in some sentence $\phi \in SL(\Lambda)$. Let f be a truth assignment. Define a function $f_* : SL(\Lambda) \to SL(\Lambda)$, such that $f_*(\psi) = \psi$ if $f(\psi) = \top$, and $f_*(\psi) = \neg\psi$ if $f(\psi) = \bot$. Then $f_*(A_1), \ldots, f_*(A_n) \vdash f_*(\phi)$.*

Proof. We prove the result by structural induction. If $\phi = A_1$, then $f_*(A_1) = f_*(\phi)$, and $f_*(\phi) \vdash f_*(\phi)$ is thus a trivial theorem of sentential logic. If $\phi = \neg\psi$, we consider two cases. If $f_*(\psi) = \psi$, then $f_*(\phi) = \neg\neg\psi$, and by induction, $f_*(A_1), \ldots, f_*(A_n) \vdash \psi$, and by using the theorem $\psi \vdash \neg\neg\psi$, we conclude $f_*(A_1), \ldots, f_*(A_n) \vdash f_*(\phi)$. If $f_*(\psi) = \neg\psi$, then $f_*(\phi) = \phi$, and the theorem is trivial. If $\phi = (\eta \Rightarrow \nu)$, then either $f(\eta) = \top$ and $f(\nu) = \top$, or $f(\eta) = \bot$. In the first case, we have $f_*(A_1), \ldots, f_*(A_n) \vdash \nu$, from which $f_*(A_1), \ldots, f_*(A_n) \vdash (\eta \Rightarrow \nu)$ follows by the theorem $\vdash \nu \Rightarrow (\eta \Rightarrow \nu)$. In the second case, $f_*(A_1), \ldots, f_*(A_n) \vdash \neg\eta$, and we can then use $\vdash \neg\eta \vdash (\eta \Rightarrow \nu)$. □

Corollary 2.13 (Completeness Theorem)**.** *If $\vDash \phi$, then $\vdash \phi$.*

Proof. Let $A_1, \ldots, A_n$ be the variables in some tautology ϕ. By the last lemma, we find that $A_1, \ldots, A_n \vdash \phi$, and also that $A_1, \ldots, \neg A_n \vdash \phi$. By the deduction theorem, we conclude that $A_1, \ldots, A_{n-1} \vdash A_n \Rightarrow \phi$, and $A_1, \ldots, A_{n-1} \vdash (\neg A_n) \Rightarrow \phi$. But then, since $\vdash (A_n \Rightarrow \phi) \Rightarrow ((\neg A_n \Rightarrow \phi) \Rightarrow \phi)$, we conclude that $A_1, \ldots, A_{n-1} \vdash \phi$ and by a similar construction, that $A_1, \ldots, \neg A_{n-1} \vdash \phi$. By induction, we can eliminate all variables on the left hand side to conclude that ϕ is provable. □

Notice that every proof we have given leading to the completeness theorem is constructive – that is, one could effectively write an algorithm which constructs the items in the proof. This means that propositional logic is *decidable*, i.e. there is an effective algorithm that takes a statement ϕ of propositional logic, and either returns a proof of the statement $\vdash \phi$, if such a proof exists, or states such a proof does not exist in the other case. The same algorithm can be converted to determine proofs certifying statements of the form $\Gamma \vdash \phi$ for *finite sets of sentences* Γ. However, if Γ is infinite, this problem is often undecidable; for instance, let $I \subset \mathbb{N}$ be any set of integers such that testing whether an integer is an element of I is undecidable. Then if $\Gamma = \{P_n : n \in I\}$, then it is an undecidable problem to determine whether $\Gamma \vdash P_m$ for a general integer m.

Before we finish our discussion of semantics, we note that there are many other axioms systems which can be used to define a propositional calculus (in the

sense that they prove all tautologies). Most interesting is the axiom system whose only connective is the Sheffer stroke, and whose only axiom schema is

$$(B|(C|D))|((E|(E|E))|((F|C)|((B|F)|(B|F))))$$

and whose rule of inference is to infer D from $B|(C|D)$, and B. Of course, it is incredibly unintuitive for us to attempt proofs in such a system, but it is certainly interesting that truth functional completeness can be obtained from a single connective, and a single axiom schema.

2.5 Verifying Propositional Formulas

Because of the completeness theorem, verifying that a particular formula of propositional logic is provable is reduced to an algorithm. If a formula ϕ contains variables $A_1, \ldots, A_n$, we simply have to consider all truth assignments of the form

$$f : \{A_1, \ldots, A_n\} \to \{\top, \bot\},$$

and then check that $f(\phi) = \top$. The completeness theorem tells us a formula is provable if and only if it is true under all truth assignments, so this suffices to decide whether the formula is provable. For any formula ϕ of length n, there can be at most n variables in ϕ, so the algorithm has time complexity $\Theta(n2^n)$. Such a runtime is undesirable, but without assuming the famous $\mathbf{P} = \mathbf{NP}$ conjecture, it is doubtful whether we can find an algorithm that is any more efficient. Here we introduce certain techniques, which simplify testing whether a formula is a tautology in certain circumstances.

Theorem 2.14 (Craig Interpolation)**.** *Let $\phi \vDash \psi$, and let the set of atomic statements shared by ϕ and ψ be $A_1, \ldots, A_n$. Then there is a statement η, known as the* interpolant*, containing only the variables A_i, such that $\phi \vDash \eta$ and $\eta \vDash \psi$.*

Proof. We proceed by induction on the number of variables in ϕ which do not occur in ψ. If every variable in ϕ occurs in ψ, let $\eta = \phi$. In the general case, fix some variable A in ϕ but not in η, take a variable B which occurs in both ϕ and ψ, and define

$$\eta = \phi[(B \wedge \neg B)/A] \vee \phi[(B \vee \neg B)/A]$$

If $f(\phi) = \top$ and $f(A) = \bot$, then $f(\phi[(B \wedge \neg B)/A]) = \top$, and if $f(A) = \top$, then $f(\phi[(B \vee \neg B)/A]) = \top$, so $\phi \vDash \eta$. In addition, $\eta \vDash \psi$. Let $f(\eta) = \top$. Then

$f(\eta[(B \wedge \neg B)/A]) = \top$, or $f(\eta[(B \vee \neg B)/A]) = \top$. In the first case, we modify the truth assignment f so that $f(A) = \bot$ (without changing the values of ϕ or ψ). Then $f(\phi) = \top$, so $f(\eta) = \top$. The other case follows by letting $f(A) = \top$. By induction, we can find an interpolant for any statement. □

Now suppose we wish to verify a formula of the form $\phi \Rightarrow \psi$ is satisfiable, where the number of variables of ϕ and ψ is few in number. The above theorem provides a constructive way to find a formula η containing only the variables that occur in both ϕ and ψ, such that $\eta \Rightarrow \psi$ holds if and only if $\phi \Rightarrow \psi$ holds, and this vastly simplifies the truth table calculation. However, if ϕ is length n with a variables, ψ is length m with b variables, and ϕ and ψ share c variables in common, then the η constructed above will have length $\Theta(n2^{a-c})$, and the calculation of the truth table of $\eta \Rightarrow \psi$ will take $\Theta((n2^{a-c} + m)2^b)$ time, which is still exponential, but in certain cases can be feasible to calculate. This is especially true if we can simplify η to a simpler form by eliminating redundant parts of the interpolant.

Example. *Consider $\phi = (B_1 \Rightarrow A) \wedge (A \Rightarrow B_2)$ and $\psi = (B_1 \wedge Z) \Rightarrow (B_2 \wedge Z)$. Then $\phi \models \psi$. We find an interpolant by the construction above of the form*

$$\begin{aligned}\eta = &\left[(B_1 \Rightarrow (B_1 \vee \neg B_1)) \wedge ((B_1 \vee \neg B_1) \Rightarrow B_2)\right] \\ &\vee \left[(B_1 \Rightarrow (B_1 \wedge \neg B_1)) \wedge ((B_1 \wedge \neg B_1) \Rightarrow B_2)\right]\end{aligned}$$

This equation can be simplified to $\neg B_1 \vee B_2$.

The second technique to reduce the complexity of verifying an equation is known as *resolution*. We assume the formula we are given is in conjunctive normal form. Let $\phi = \psi_1 \wedge \cdots \wedge \psi_n$ be be such a formula of propositional logic. We will view a conjunction of clauses as a set $\phi = \{\psi_1, \ldots, \psi_n\}$, where ϕ is true in an interpretation only when each ψ_i is true. Without loss of generality, we may assume that no ψ_i contains A and $\neg A$ in disjunctions, for then the conjunct is vacously satisfied. Suppose ψ_i contains an instance of a statement A, and ψ_j contains an instance of a statement $\neg A$. The disjunction obtained from ψ_i and ψ_j by concatenating all disjuncts together, dropping all occurrences of A and $\neg A$, and then removing duplicates, is called the resolution of ψ_i and ψ_j with respect to the variable A, and we will denote the resulting clause by $\mathrm{Res}_A(s_i, s_j)$. Then $\psi_i, \psi_j \models \mathrm{Res}_A(\psi_i, \psi_j)$, because semantically, either A is true or $\neg A$ is true, and in the first case some clause of ψ_j other than $\neg A$ must be true, and in the second some clause of ψ_i other than A must be true. Given ϕ, let the *resolution* $\mathrm{Res}(\phi)$ be the smallest set of clauses containing all clauses in ϕ, and closed under resolution.

It is clear that ϕ is true if and only if $\mathrm{Res}(\phi)$ is true, because we have only added clauses which are logically implied by the other clauses of ϕ.

Example. *If*

$$\phi = \{A \vee \neg B \vee \neg C, \neg A \vee \neg B \vee D, A \vee C \vee D\} = \{\alpha, \beta, \kappa\}$$

then $Res(\phi)$ contains the clauses of ϕ, and in addition the clauses

$$\begin{aligned} Res_A(\alpha,\beta) &= \neg B \vee \neg C \vee D \\ Res_A(\beta,\kappa) &= \neg B \vee C \vee D \\ Res_C(\alpha,\kappa) &= A \vee \neg B \vee D \\ Res_C(Res_A(\alpha,\beta), Res_A(\beta,\kappa)) &= \neg B \vee D \end{aligned}$$

so the resolution has seven clauses.

If $\mathrm{Res}_A(\phi,\psi)$ ever equals the empty clause, for some formulas ϕ and ψ, then $\phi = A$, $\psi = \neg A$, and ϕ and ψ can never be simultaneously satisfied. Thus a formula ϕ is unsatisfiable if $\mathrm{Res}(\phi)$ contains the empty clause. What is more interesting is that the converse is true, though it's a little tricky to prove.

Lemma 2.15. *Let $Res_A(\phi)$ denote the term obtained by concatenating all possible terms by resolution on the variable A, and then removing all clauses which contain an instance of ϕ or $\neg\phi$. If $Res_A(\phi)$ is satisfiable, and contains no empty clauses, then ϕ is satisfiable.*

Proof. Let f be a truth assignment with $f(\mathrm{Res}_A(\phi)) = \top$. We then claim that by modifying f to set $f(A) = \top$ or $f(A) = \bot$, we can make $f(\phi) = \top$. If ψ is a clause of ϕ not containing A or $\neg A$, then $f(\psi) = \top$. Conversely, if no $\psi \in \phi$ contains $\neg A$, then we may set $f(A) = \top$, and then all clauses are satisfied. Similarily, if no ψ contains A, we set $f(A) = \bot$. Thus we are reduced to the case where some clause of ϕ contains A, and some other clause contains $\neg A$. Let

$$g(A) = \begin{cases} \top & A = C \\ f(A) & A \neq C \end{cases} \qquad h(A) = \begin{cases} \bot & A = C \\ f(A) & A \neq C \end{cases}$$

Suppose that $h(\eta) = \bot$, and $g(\psi) = \bot$. Then η must contain an instance of A, and ψ must contain an instance of $\neg A$. We note that $\nu = \mathrm{Res}_A(\psi,\eta)$ contains no instances of A or $\neg A$ and is nonempty, hence $f(\nu) = g(\nu) = h(\nu) = \top$, hence there is some term in η or ψ already satisfied by the interpretation of ϕ.

This implies that the condition $g(\eta) = \bot$ and $h(\psi) = \bot$ is impossible, so either $g(\eta) = \top$ for all clauses, or $h(\psi) = \top$ for all clauses, and this completes the proof. □

Theorem 2.16. *ϕ is a contradiction iff Res(ϕ) contains an empty clause.*

Proof. We prove by induction on the number of variables in the clauses of ϕ. If ϕ contains only a single variable, then either $\phi = \{A\}$, $\phi = \{\neg A\}$, or $\phi = \{A, \neg A\}$. The first two are satisfiable, and their resolution does not contain the empty clause, and the second is a contradiction, and its resolution contains the empty clause. Now if ϕ is a contradiction, then either $\text{Res}_A(\phi)$ contains an empty clause, or $\text{Res}_A(\phi)$ must be a contradiction. In the second case, we apply induction on the number of atoms to conclude that $\text{Res}(\text{Res}_A(\phi))$ contains an empty clause, and $\text{Res}(\text{Res}_A(\phi)) \subset \text{Res}(\phi)$. □

Resolution gives us an algorithm to calculate whether any formula of prepositional logic is true. Given such a statement ϕ, take $\neg\phi$, and convert it to a conjunctive normal form ψ. Then ϕ is a tautology if and only if ψ is a contradiction, so we just determine if $\text{Res}(\psi)$ contains an empty clause, and this tells us if ϕ is a tautology. There are programming languages, like *Prolog*, which utilize these methods to verify inputted logical formuli (involving further computational techniques for formulas involving quantifiers, introduced in the next Chapter).

2.6 Sequent Calculi

The complete formal system we have studied is styled in the sense of a great many formal systems, known as *Hilbert systems*. A Hilbert system just takes axioms, and deductive rules, and then forms proofs as sequences $(\phi_1, \ldots, \phi_n)$. But there are a great many styles of formal systems, and this section I shall detail my personal favourite, natural deduction. Most actual proofs in mathematics do not follow a linear style. We instead form a proof by combining prior deductions in a non-linear way to reach the conclusion, the end of the proof. Thus natural deduction does not model a proof as a sequence $(\phi_1, \ldots, \phi_n)$, but instead as a tree, whose root node is the conclusion we are attempting to form. The nodes of the tree will not consist of formulas, but instead of sequents, which we have almost already seen, which are pairs of sequences of formulas of the form $\phi_1, \ldots, \phi_n \vdash \psi_1, \ldots, \psi_m$, which we interpret as proving $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow (\psi_1 \vee \cdots \vee \psi_m)$. Why the assymmetry? It turns out that this will give us symmetry in *proofs*, which we shall

require later. A manifestation of this symmetry is that if $\phi \Rightarrow \psi$ is true, then both $(\phi \wedge \eta) \Rightarrow \psi$ and $\phi \Rightarrow (\psi \vee \eta)$ is true, so the sequents $\phi, \eta \vdash \psi$ and $\phi \vdash \psi, \eta$ may be derived from the sequent $\phi \vdash \psi$. We have already treated the semantics of propositional logic, so we may just state the axioms and deduction rules. Unlike a Hilbert system, this system has far more deduction rules than axioms, which is used to argue why this system is more *natural* – we more naturally deal with deduction rules. The only axioms are of the form $\phi \vdash \phi$, and the deduction rules are the edges from which we form our tree,

$$\frac{\phi, \Gamma \vdash \Delta}{\phi \wedge \psi, \Gamma \vdash \Delta} \ (\wedge L) \qquad \frac{\phi, \Gamma \vdash \Delta}{\psi \wedge \phi, \Gamma \vdash \Delta} \ (\wedge L) \qquad \frac{\Gamma \vdash \Delta, \phi \quad \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta, \phi \wedge \psi} \ (\wedge R)$$

$$\frac{\Gamma \vdash \Delta, \phi}{\Gamma \vdash \Delta, \phi \vee \psi} \ (\vee R) \qquad \frac{\Gamma \vdash \Delta, \phi}{\Gamma \vdash \Delta, \psi \vee \phi} \ (\vee R) \qquad \frac{\phi, \Gamma \vdash \Delta \quad \psi, \Gamma \vdash \Delta}{\phi \vee \psi, \Gamma \vdash \Delta} \ (\vee L)$$

$$\frac{\Gamma \vdash \Delta, \phi \quad \psi, \Sigma \vdash \Pi}{\phi \Rightarrow \psi, \Gamma, \Sigma \vdash \Delta, \Pi} \ (\Rightarrow L) \qquad \frac{\phi, \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta, \phi \Rightarrow \psi} \ (\Rightarrow R)$$

$$\frac{\Gamma \vdash \Delta, \phi}{\neg\phi, \Gamma \vdash \Delta} \ (\neg L) \qquad \frac{\phi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg\phi} \ (\neg R)$$

We also have *structural rules*

$$\frac{\Gamma \vdash \Delta}{\phi, \Gamma \vdash \Delta} \ (KL) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \phi} \ (KR)$$

$$\frac{\phi, \Gamma \vdash \Delta}{\phi, \phi, \Gamma \vdash \Delta} \ (WL) \qquad \frac{\Gamma \vdash \Delta, \phi}{\Gamma \vdash \Delta, \phi, \phi} \ (WR)$$

$$\frac{\Gamma_1, \phi, \psi, \Gamma_2 \vdash \Delta}{\Gamma_1, \psi, \phi, \Gamma_2 \vdash \Delta} \ (CL) \qquad \frac{\Gamma \vdash \Delta_1, \phi, \psi, \Delta_2}{\Gamma \vdash \Delta_1, \psi, \phi, \Delta_2} \ (CR)$$

Note that the names for these structural rules alude to the Curry-Howard correspondence, where implication formulas correspond to combinators in combinatory logic. The rules KL and KR are called *thinning*, or *weakening*, the rules WL and WR are called *contraction*, and the rules CL and CR are called *permutation*. A proof of a sequent $\Gamma \vdash \Delta$ is a tree, whose root is $\Gamma \vdash \Delta$, whose leaves are axioms of the form $s \vdash s$, and such that each edge is annotated by the appropriate deduction rule, for which the deduction is accurate. It turns out it is fairly simple to form deductions, since we may work backwards in most cases to determine which deduction rules to apply.

Example. *Consider a proof of Pierce's law, $(\phi \Rightarrow \psi) \Rightarrow \phi \vdash \phi$. The only way to obtain an implication on the left hand side of the sequent is to apply the rule $(\Rightarrow L)$, so we must first find a way to prove $\vdash (\phi \Rightarrow \psi), \phi$ and $\phi \vdash \phi$. We will eventually prove sequent calculus is complete, and so we, intuitively, know that proving these two statements is possible. To obtain the first statement, we likely need to use $(\Rightarrow R)$, which means we must prove $\phi \vdash \psi, \phi$, and this is proved from the axiom $\phi \vdash \phi$ from the structural axiom (KR). Thus we obtain the following proof tree.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi \vdash \phi}{\phi \vdash \phi, \psi}\,(KR)}{\vdash \phi, \phi \Rightarrow \psi}\,(\Rightarrow I) \qquad \phi \vdash \phi}{(\phi \Rightarrow \psi) \Rightarrow \phi \vdash \phi, \phi}\,(\Rightarrow L)}{(\phi \Rightarrow \psi) \Rightarrow \phi \vdash \phi}\,(WR)$$

This graph gives a proof of Pierce's law via sequent calculus.

Example. *The sequent $\phi, \phi \Rightarrow \psi \vdash \psi$ is provable using the CUT rule.*

$$\dfrac{\phi \vdash \phi \qquad \psi \vdash \psi}{\phi, \phi \Rightarrow \psi \vdash \psi}\,(\Rightarrow L)$$

The sequent $\Rightarrow L$ is essentially modus ponens.

It turns out the deduction rules have given here are truth-functionally complete. But there are some unintuitive aspects of the rules given. For instance, every deduction step is *additive*, i.e. no rule above ever reduces the complexity of the statements involved. This makes it impossible to form parts of a proof tree of the form

$$\dfrac{\vdash \phi \qquad \phi \vdash \psi}{\vdash \psi}\,(?)$$

where (?) stands for some sequence of deductive statements given above, what we could call a *derived rule*. Nonetheless, what is true about the sequent calculi is that *if* it is possible to prove $\vdash \phi$ and $\phi \vdash \psi$ in a sequent tree, one can *then* prove $\vdash \psi$ in a proof tree; in order to prove the two statements, previous statements must have been proved, which can be used to infer ψ using the deductive rules above. In fact, one can see this as a special case of the *cut rule*

$$\dfrac{\Gamma \vdash \Delta, \phi \qquad \phi, \Sigma \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}\,(\text{CUT})$$

We will see, more generally, that proof trees using the cut rule do not expand the family of statements that can be proved using sequents. We say a formula is *principal* for an application of a rule if it is a lower sequent created by an application of the rule, and a *subaltern* formula for a rule is a formula in an upper sequent which is used primarily in the application of the rule. For an occurence of a formula in a sequent in a proof, it's *parametric ancestors* are precisely occurences of the same formula which occur in a sequent above, and are obtained by applying rules that do not affect the occurence of that formula. The *degree* δ of an application of the (CUT) rule is precisely the degree of the formula removed. The *contraction measure* μ of the application is the number of applications of (WL) and (WR) to the parametric ancestors of the formula removed. The *left rank* ρ_1 is the maximal number of consecutive sequents in the subtree rooted in the left upper sequent, in which the formula removed occurs on the left hand side of the formula. Define the right rank ρ_2 analogously, as the maximal number of consecutive sequents in the subtree rooted in the right upper sequent. Define the *rank* to be $\rho = \rho_1 + \rho_2$.

Theorem 2.17. *(CUT) is admissable.*

Proof. We perform a triple induction in δ, μ, and ρ, i.e. proving that any application of the (CUT) rule with degree δ, contraction measure μ, and rank ρ is admissable. We note that in general, $\delta \geqslant 0$, $\mu \geqslant 0$, and $\rho \geqslant 2$.

Let us first consider this simplest situation, where $\rho = 2$, $\delta = 0$, and $\mu = 0$. The fact that $\delta = 0$ implies that the cut formula ϕ is atomic, and the fact that $\rho = 2$ implies that ϕ never occurs in sequences above the upper left and right sequent in the application of the cut rule. There are only a few possiblities on what the upper sequents look like: each of the formulas is either the axiom $\phi \vdash \phi$, or is obtained by a weakening rule (KL) or (KR), which introduces ϕ. It is clear the (CUT) rule can be removed in the case that the formulas on either side are axioms, because then the lower sequent can be introduced as an axiom, i.e.

$$\text{we can replace} \quad \dfrac{\dfrac{\phi \vdash \phi \qquad \phi \vdash \phi}{\phi \vdash \phi}\,(\text{CUT})}{\vdots} \qquad \text{with} \qquad \dfrac{\phi \vdash \phi}{\vdots}$$

If one formula is an axiom, then the lower sequent is just a copy of the other upper sequent, and the axiom, and resulting (CUT) rule, can be removed, e.g. if the axiom occurs on the left, then

$$\text{we can replace} \quad \dfrac{\phi \vdash \phi \qquad \dfrac{\vdots}{\phi, \Delta \vdash \phi}}{\dfrac{\phi, \Delta \vdash \phi}{\vdots}} \text{ (CUT)} \quad \text{with} \quad \dfrac{\dfrac{\vdots}{\phi, \Delta \vdash \phi}}{\vdots}$$

If both formulas are obtained by thinning, then the lower sequent can be obtained via applications of weakening, i.e. we can replace

$$\dfrac{\dfrac{\dfrac{\vdots}{\Delta_1 \vdash \Psi_1}}{\phi, \Delta_1 \vdash \Psi_1} \qquad \dfrac{\vdots}{\Delta_2 \vdash \Psi_2, \phi}}{\dfrac{\Delta_1, \Delta_2 \vdash \Psi_1, \Psi_2}{\vdots}} \text{ (CUT)} \quad \text{with} \quad \dfrac{\dfrac{\dfrac{\vdots}{\Delta_1 \vdash \Psi_1}}{\Delta_1, \Delta_2 \vdash \Psi_1, \Psi_2}\ (K)}{\vdots}$$

Thus we've proved the base case where $\rho = 2$, $\delta = 0$, and $\mu = 0$. This in fact covers all cases where $\rho = 2$ and $\delta = 0$, because in such a situation it is impossible for μ to be positive, since this would imply that ϕ appeared twice, in order to be contracted, which would imply a rank of at least three or more.

Now we move onto the case where $\rho = 2$, $\delta > 0$, and $\mu = 0$. Now the cut formula is no longer atomic, so may have been introduced by connective or quantifier rules. Because of the number of rules, there are a large number of technical cases, which we leave to the reader to verify completely. Let us consider the case where the cut formula is of the form $\phi \wedge \psi$. Since $\rho = 2$, the situation must look as follows

$$\dfrac{\dfrac{\dfrac{\vdots}{\phi, \Theta \vdash \Lambda}}{\phi \wedge \psi, \Theta \vdash \Lambda} \qquad \dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta, \phi} \quad \dfrac{\vdots}{\Gamma \vdash \Delta, \psi}}{\Gamma \vdash \Delta, \phi \wedge \psi}}{\dfrac{\Gamma, \Theta \vdash \Delta, \Lambda}{\vdots}}$$

We now move the (CUT) rule 'up the tree', i.e. replacing this part of the proof with

$$\dfrac{\dfrac{\vdots}{\phi, \Theta \vdash \Lambda} \qquad \dfrac{\vdots}{\Gamma \vdash \Delta, \phi}}{\Gamma, \Theta \vdash \Delta, \Lambda}\ (CUT)$$

The value of δ has been decreased in the value of the cut, so that we can apply the inductive hypothesis to completely remove the cut. For other cases, i.e. where the cut formula is of the form $\phi \Rightarrow \psi$, one must introduce *two* smaller cuts, but since these two cuts have lower degree this causes no problems. Given that if $\rho = 2$, it is impossible for μ to be positive, we have no proved all cases where $\rho = 2$.

Finally, we consider the case where $\rho > 2$. The complexity of this part of the proof multiplies considerably because it is now possible for the cut formula to be left latent in formulas in formulas above the sequent. We refer to Chapter 2 of Bimbó, Proof Theory: Sequent Calculi and Related Formulisms for more of the technicalities. □

We desire to show this system is complete. Let us define a sequence $s_1, \ldots, s_n \vdash w_1, \ldots, w_m$ to be a *tautology* if, under any truth assignment that makes each formula $s_1, \ldots, s_n$ true, one of the formulas $w_1, \ldots, w_m$ are true. It is simple to prove that any statement proved using the sequent calculus is a tautology: one need only verify that an axiom is a tautology, and that deductive rules prove tautologies. To prove that our deductive system is *complete*, i.e. that every tautology is provable, we can cheat by using some of our previous work; we know that modus ponens is a derived deductive rule in the sequent calculus with the (CUT) rule. If we can show that one can derive that for any statement s of the form $(A1)$, $(A2)$, or $(A3)$ in SL(Λ), the sequent $\vdash s$ is provable in the sequent calculus, then, because modus ponens is the only deductive rule of SL(Λ), one can convert any proof of SL(Λ) into a sequent calculus proof. Since SL(Λ) is complete, it follows that the sequent calculus is complete. Of course, one must prove any statements which fit $(A1)$, $(A2)$, and $(A3)$, but this is not too difficult a matter, and is left to the reader.

Chapter 3

First Order Logic

It is logical to conclude that "Julie is a human" and "Laura is a human" from the general statement that "All women are human"? In Propositional logic, we are unable to model this deduction. First order logic (also known as the lower predicate calculus) is a formal system modelling these derivations.

3.1 Language

The syntax of predicate logic is less homogenous, for our language must contain nouns, like "Julie" and "Laura", which are the things we talk about, and separate words we apply to nouns, obtaining truth values. These are known as *terms* and *predicates* respectively. Terms should model both definite nouns, such as "Julie" and "Laura", as well as variables, like as x and y, which can stand for many definite nouns at once. We should also have relational nouns, such as "The school x went to", a statement describing a noun which varies in interpretation based on the value of x. Definite nouns are known as *constants*, and relational nouns are known as *functions*. Functions will be separated based on their *arity*, the number of arguments they take. For example, "the closest common ancestor of x and y" is a 'binary' function, "the birthday of x" is a 'unary' function. To form the syntax of this language, we start with an at least countable set Λ of variables, a set Δ of constants, and for each n, a set Ψ_n of n-ary functions. The set of *terms* of first order logic is the smallest set $T(\Lambda, \Delta, \Psi)$ such that $\Lambda, \Delta \subset T(\Lambda, \Delta, \Psi)$, and if $t_1, \ldots, t_n \in T(\Lambda, \Delta, \Psi)$, and f is a function in Ψ_n, then $f(t_1, \ldots, t_n) \in T(\Lambda, \Delta, \Psi)$.

After introducing the terms of first order logic, we introduce *predicates*, which model statements about nouns. For instance, "x is a human" is a predicate. Pred-

icates, like functions, are separated based on arity. For each n, let Π_n be a set of n-ary predicates. Given Λ, Δ, and Ψ like before, we shall define an *atomic formula* to be a string of the form $P(t_1, \ldots, t_n)$, where $P \in \Pi_n$, and $t_1, \ldots, t_n \in T(\Lambda, \Delta, \Psi)$. Then the statements of the first order language $\mathrm{FO}(\Lambda, \Delta, \Psi, \Pi)$ is defined to be the smallest language containing all atomic formulae, and also closed under the logical operations $\wedge, \vee, \Rightarrow, \neg, \Leftrightarrow$, as in sentential logic, as well as *quantifiers*, such that if $x \in \Lambda$ is a variable, and ϕ is a statement, then $(\forall x : \phi)$ and $(\exists x : \phi)$ are formulae in the language. We call the set of all statements formed by the process above a *first order language*.

Using much the same methods as in the syntax of propositional logic, we can verify that the syntax of first order logic has the property that all statements in the language have a unique principal connective, and can be parsed uniquely. We leave this result to the reader, since it does not require too many tecniques. Given that we have already discussed the basic syntax of sentential logic, we will also leave it to the reader to fill in the syntactic details when needed when analyzing the interesting portion of first order logic. We shall also rely on informal abbreviations for various statements in the language, such as writing $(\forall x_1 : (\forall x_2 : \ldots (\forall x_n : \phi) \ldots))$ for the statement $(\forall x_1, x_2, \ldots, x_n : \phi)$, and writing certain binary predicates via infix notation, e.g. writing $a = b$ and $a \leqslant b$ as informal notation for the statements $= (a, b)$ and $\leqslant (a, b)$.

Example. *We can construct a first order language to discuss ring theory formally. To model the concepts of ring theory, we need constants* 0 *and* 1*, to represent the additive and multiplicative identities, as well as a unary function* $-$ *to represent the additive inverse, as well as binary functions* $+$ *and* $\cdot$ *to represent the structure of addition and multiplication in a ring. The resulting first order language* L_{ring} *is the* language of rings.

Formally speaking, 'the language of rings' is not unique, because we have not specified a choice of variables. Nonetheless, we will never treat a language in such a way that the exact symbols for the variables are relevent, and so this will not concern us, provided we have an infinite number of variables, so that we can form statements of arbitrarily large complexity.

Example. *The language* L_{ord} *of ordered sets is constructed by considering no constants, and a single binary predicate* $<$*, used to model the ordering relation.*

Example. *The language* $L_{digraph}$ *of directed graphs is constructed with no constants, and a single binary predicate R, modelling the adjacency relation.*

We often use variables in mathematics for the purpose of *substitution*: computations in algebra done with variables should remain true when arbitrary numbers are substituted for the variables. We want to do the same thing with variables in first order logic. But their interaction with *quantifiers* makes this process a little tricky. For instance, consider the following statement:

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

We cannot naively substitute an arbitrary term for the variable x, because the result will be nonsense, i.e. if we substitute "Fido the Dog" for x, we obtain

For any Fido the Dog, if Fido the Dog is a man, then Fido the Dog is mortal.

The resulting sentence is nonsense, because the variable did not stand for something to be naively replaced in the formula above. The important feature of the language is that variables can be *bound*. This is analogous to the use of variables for representation integrals in calculus, i.e. in the equation

$$\int_0^1 x\,dx = 1/2.$$

Substituting numbers for x in the formula above would lead to uninterpretable equations. So we should not substitute *bound variables*.

Another, slightly more subtle problem occurs when substituting *variables* for other variables. Certainly, if we know that for any choice of x and y, $x^2 + y^2 = 3$, then we can substitute y for x to conclude that for any choice of y, $y^2 + y^2 = 3$. But with variable binding there can be problems. If we substitute x for y in the formula

For any x, there is y such that if x is a man, then y is a dog.

The resulting substitution is

For any y, there is y such that if y is a man, then y is dog.

We normally interpret variables as being bound to the closest nested quantifier, which means these two equations are no longer equivalent.

We wish to perform substitutions in our calculations using the language of first order logic while avoiding these two problems. Let ϕ be an arbitrary string in a first order language. An occurrence of a variable x is *bound* in ϕ if it occurs in

a subformula of the form $(\forall x : \psi)$ or $(\exists x : \psi)$. An occurrence is *free* if it is not bound, and a variable y is *free for substitution for x in ϕ* if x does not occur in any subformula of the form $(\forall y : \psi)$ or $(\exists y : \psi)$, where y is a free variable in ϕ. Given formulas $\psi_1, \ldots, \psi_n$, we will only consider substitutions of all free occurences of variables $x_1, \ldots, x_n$ in a formula ϕ, provided that for each j, and each free variable y that occurs in ψ_j, y is free for substitution for x_j in ϕ. The substitution of ψ_j for each occurence of x_j will be denoted $\phi[\psi_1/x_1, \ldots, \psi_n/x_n]$.

This approach avoids the problems with interpretation that occur above. In the first example, x is a bound variable, and so will not be substituted. In the second example, x is a free variable, but y is not free for substitution for x in the formula. Given a formula ϕ, if we list the free variables as $x_1, \ldots, x_n$, it will often be convenient to informally denote ϕ by $\phi(x_1, \ldots, x_n)$ or $\phi(x)$, and to denote the substitution $\phi[t/x]$ as $\phi(t_1, \ldots, t_n)$ or $\phi(t)$.

3.2 First Order Formal Systems

We now introduce a family of axioms and deduction rules which we will see captures the semantics of first order logic. For simplicity, we consider only formulae in the connectives $\forall$, $\neg$, and $\Rightarrow$, since all the other basic symbols are equivalent to formulae formed by these connectives. We shall use an axiom system consisting of five schemata, the original (A1), (A2), and (A3) found in sentential logic, as well as two new first order schema (A4) and (A5). Let ϕ and ψ be formula, and suppose that t a term free for substition for a variable x. Then (A4) is

$$(\forall x : \phi(x)) \Rightarrow \phi(t).$$

(A5) is a schema such that, for any formula ϕ containing no free occurences of x,

$$(\forall x : \phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow (\forall x : \psi)).$$

Our rules of inference are modus ponens (MP), to infer ψ from ϕ and $\phi \Rightarrow \psi$, as well as universal generalization (UG), inferring $(\forall x : \phi)$ from ϕ for any variable x. As in predicate logic, we shall let $\vdash \phi$ denote the fact that ϕ is a theorem of the system of first order logic. If Γ is a set of formulae, we shall let $\Gamma \vdash \phi$ state that ϕ is provable assuming Γ are additional axioms.

Remark. The deduction rule (UG) is somewhat subtle. It is not equivalent to introducing an additional axiom schema of the form $\phi \Rightarrow (\forall \mathsf{x} : \phi)$, for this statement is not sound in all models of predicate logic. Hopping ahead to assume knowledge of

the semantics of first order logic, consider the formula $\mathsf{P}(\mathsf{x}) \Rightarrow (\forall \mathsf{x} : \mathsf{P}(\mathsf{x}))$. Take a model M with universe $\{a, b\}$, and let $\mathsf{P}_M = \{a\}$. If we consider an assignment of variables f such that $f(\mathsf{x}) = a$, then the formula $\mathsf{P}(\mathsf{x})$ is satisfied, but $(\forall \mathsf{x} : \mathsf{P}(\mathsf{x}))$ is not. Thus $\mathsf{P}(\mathsf{x}) \Rightarrow (\forall \mathsf{x} : \mathsf{P}(\mathsf{x}))$ is not semantically valid, and hence not provable in first order logic. On the other hand, the deductive rule (UG) does preserve soundness; if a structure M is given, and $M \vDash \mathsf{P}(\mathsf{x})$, then $M \vDash (\forall x : \mathsf{P}(\mathsf{x}))$. It is true that every model that satisfies Γ satisfies $\mathsf{P}(\mathsf{x})$, then every model that satisfies Γ also satisfies $(\forall \mathsf{x} : \mathsf{P}(\mathsf{x}))$. The problem is again the occurence of free variables, a quirk of the standard syntax of first order logic. This quirk lead Moses Schönfinkel and Haskell Curry to invent combinatory logic, a logical system designed to avoid having quantified variables in the syntax.

To begin working with this formal system, we notice our system includes (A1), (A2), (A3), and (MP), all the rules of propositional calculus, which justifies a very useful property of our new formal system.

Theorem 3.1. *Let ϕ be a formula in first order logic, and consider a statement ψ in propositional logic, obtaining by replacing each outermost occurence of a quantifier $(\forall x : \phi)$ with a distinct propositional variable. Then if ψ is a tautology, ϕ is provable in first order logic.*

Proof. By the completeness theorem for propositional logic, ψ is provable using the axioms (A1), (A2), (A3), and (MP). If one takes the proof, and reverses the substitution procedure, i.e. replacing each propositional variable with it's original subformula, then one obtains a proof of ϕ in first order logic. □

We shall denote an application of the deduction theorem by (TAUT). There are many useful schemas obtained from this. Let ϕ and ψ be arbitrary statements of first order logic. Then the following holds:

- Negation Elimination (NE): $\vdash \neg\neg\phi \Rightarrow \phi$
- Negation Introduction (NI): $\vdash \phi \Rightarrow \neg\neg\phi$
- Conjunction Elimination (CE): $\vdash \phi \wedge \psi \Rightarrow \phi$, $\vdash \phi \wedge \psi \Rightarrow \psi$.
- Conjunction Introduction (CI): $\vdash \phi \Rightarrow (\psi \Rightarrow (\phi \wedge \psi))$.

One can just apply the completeness theorem of sentential logic to obtain these statements as theorems.

Example. *If t is free for x in $\phi(x)$, then $\phi(t) \Rightarrow (\exists x : \phi(x))$ is a theorem of first order logic, a formalization of existential introduction. We recall that $(\exists x : \phi(x))$ can be rewritten as $\neg(\forall x : \neg\phi(x))$ for the purpose of deduction.*

$$
\begin{array}{|l}
\phi(\mathsf{t}) \Rightarrow \neg(\forall \mathsf{x} : \neg\phi(\mathsf{x})) \\
\hline
1.\ (\forall \mathsf{x} : \neg\phi(\mathsf{x})) \Rightarrow \neg\phi(\mathsf{t}) \qquad \textit{(A4)} \\
2.\ ((\forall \mathsf{x} : \neg\phi(\mathsf{x})) \Rightarrow \neg\phi(\mathsf{t})) \Rightarrow (\phi(\mathsf{t}) \Rightarrow \neg(\forall \mathsf{x} : \neg\phi(\mathsf{x}))) \qquad \textit{(TAUT)} \\
3.\ \phi(\mathsf{t}) \Rightarrow \neg(\forall \mathsf{x} : \neg\phi(\mathsf{x})) \qquad \textit{(1),(2),(MP)}
\end{array}
$$

Without the tautology theorem, this deduction would be much longer. It will be useful in further proofs to note that we never applied (UG) to any formulae here. We denote an application of this proof by (EI).

The deduction theorem cannot be carried directly over to first order logic, for assumptions in proofs and premises in logical statements are interpreted differently in certain circumstances. $\phi \vdash (\forall x : \phi)$ is always true for any statement ϕ, yet $\vdash \phi \Rightarrow (\forall x : \phi)$ is not always a theorem, as we saw in the last remark. The problem is, of course, the free variables again causing trouble. In a proof $(\psi_1, \ldots, \psi_n)$ of $\Gamma \vdash \phi$, we say ψ_i *depends on* $\eta \in \Gamma$ if ψ_i is η, or ψ_i is inferred from ψ_j and ψ_j depends on η.

Theorem 3.2. *Suppose that $\Gamma, \phi \vdash \psi$, and there is a proof of ψ which involves no application of universal generalization on a formula which depends on ϕ, by a variable x which is free in ϕ. Then $\Gamma \vdash \phi \Rightarrow \psi$.*

Proof. We perform induction on the length of proofs. If ψ can be proved in one statement, then ψ is either an instance of an axiom, or an element of $\Gamma \cup \{\phi\}$. If $\phi \neq \psi$, then $\Gamma \vdash \psi$, since the proof is equally valid here, and hence $\Gamma \vdash \phi \Rightarrow \psi$. If $\phi = \psi$, then $\phi \Rightarrow \psi$ is a tautology, so $\Gamma \vdash \phi \Rightarrow \psi$ is valid. Now suppose we have a proof $(\eta_1, \ldots, \eta_{n+1})$. By induction, for each η_i, we have $\Gamma \vdash (\phi \Rightarrow \eta_i)$. If η_{n+1} is an axiom, an element of Γ, or equal to ϕ, we have already justified the implication. Suppose η_{n+1} was inferred by modus ponens from η_i and η_j, where η_j has the form $\eta_i \Rightarrow \eta_{n+1}$. Then $\Gamma \vdash (\phi \Rightarrow \eta_i)$, and $\Gamma \vdash (\phi \Rightarrow (\eta_i \Rightarrow \eta_{n+1}))$ by induction. The statement

$$\nu = ((\phi \Rightarrow (\eta_i \Rightarrow \eta_{n+1})) \Rightarrow (\phi \Rightarrow \eta_i)) \Rightarrow (\phi \Rightarrow \eta_{n+1})$$

is a tautology, so $\Gamma \vdash \nu$, and by modus ponens, we find $\Gamma \vdash (\phi \Rightarrow \eta_{n+1})$. Otherwise η_{n+1} is of the form $(\forall x : \eta_i)$, obtained from some η_i by universal

generalization. By induction, $\Gamma \vdash \phi \Rightarrow \eta_i$, so $\Gamma \vdash (\forall x : \phi \Rightarrow \eta_i)$. By assumption, x does not occur as a free variable of ϕ, so we have the axiom $(\forall x : \phi \Rightarrow \eta_i) \Rightarrow (\phi \Rightarrow (\forall x : \eta_i))$, and we conclude $\Gamma \vdash \phi \Rightarrow (\forall x : \eta_i)$. □

Example. *We can now show that for any statement ϕ,*

$$\vdash (\forall x, y : \phi)) \Rightarrow (\forall y, x : \phi).$$

To see this, we apply our newly established deduction theorem.

$(\forall \mathsf{x}, \forall \mathsf{y} : \phi) \Rightarrow (\forall \mathsf{y}, \forall \mathsf{x} : \phi)$	
1. $(\forall \mathsf{x}, \forall \mathsf{y} : \phi)$	
2. $(\forall \mathsf{x}, \forall \mathsf{y} : \phi) \Rightarrow (\forall \mathsf{y} : \phi)$	*(A4)*
3. $(\forall \mathsf{y} : \phi)$	*(1),(2),(MP)*
4. $(\forall \mathsf{y} : \phi) \Rightarrow \phi$	*(A4)*
5. ϕ	*(3),(4),(MP)*
6. $(\forall \mathsf{x} : \phi)$	*(5),(UG)*
7. $(\forall \mathsf{y}, \forall \mathsf{x} : \phi)$	*(6),(UG)*
8. $((\forall \mathsf{x}, \forall \mathsf{y} : \phi) \Rightarrow (\forall \mathsf{y}, \forall \mathsf{x} : \phi)$	$(1-7), (DT)$

Care needs to be taken in order to ensure these steps are accurate, and do not apply universal generalization on a free variable.

Example. *For any ϕ and ψ, $\vdash (\forall x : \phi \Leftrightarrow \psi) \Rightarrow ((\forall x : \phi) \Leftrightarrow (\forall x : \psi))$.*

$(\forall \mathsf{x} : \phi \Leftrightarrow \psi) \Rightarrow ((\forall \mathsf{x} : \phi) \Leftrightarrow (\forall \mathsf{x} : \psi))$	
1. $(\forall \mathsf{x} : \phi \Leftrightarrow \psi)$	
2. $(\phi \Leftrightarrow \psi)$	*(A4)*
3. $(\forall \mathsf{x} : \phi)$	
4. ϕ	*(A4), (3), (MP)*
5. ψ	*(2),(4),(MP)*
6. $(\forall \mathsf{x} : \psi)$	*(UG)*
7. $(\forall \mathsf{x} : \phi) \Rightarrow (\forall \mathsf{x} : \psi)$	*(3-6),(DT)*
8. $(\forall \mathsf{x} : \psi)$	
9. ψ	*(A4), (8), (MP)*
10. ϕ	*(2),(9),(MP)*
11. $(\forall \mathsf{x} : \phi)$	*(UG)*
12. $(\forall \mathsf{x} : \psi) \Rightarrow (\forall \mathsf{x} : \phi)$	*(8-11), (DT)*
13. $(\forall \mathsf{x} : \phi) \Leftrightarrow (\forall \mathsf{x} : \psi)$	*(7),(12),(TAUT),(MP)*
14. $(\forall \mathsf{x} : \phi \Leftrightarrow \psi) \Rightarrow ((\forall \mathsf{x} : \phi) \Leftrightarrow (\forall \mathsf{x} : \psi))$	*(2-14), (DT)*

The next result is an immediate consequence of the deduction theorem, left to the reader.

Theorem 3.3. *If ϕ has no free occurrences of y, then*

$$\vdash ((\forall x : \phi(x)) \Leftrightarrow (\forall y : \phi(y)))$$

The next theorem is more involved, but very useful. We define substitution on formulas in the following way. Given formulae ϕ, ψ, η, we define the formula $\phi[\psi/\eta]$, obtained from swapping η with ψ, by the base case $\eta[\psi/\eta] = \psi$, and the recursive case by delving into subformulas, *provided the formula isn't just η.*

Theorem 3.4. *Let $x_1, \ldots, x_n$ be all free variables of ϕ that occurs as a bound variable in ψ or η. Then*

$$(\forall x_1, \ldots, x_n : \eta \Leftrightarrow \psi) \Rightarrow (\phi \Leftrightarrow \phi[\psi/\eta])$$

The theorem also holds if $\phi[\psi/\eta]$ is replaced with a formula obtained only be swapping some (but perhaps not all) occurences of η.

Proof. We prove, like always, by structural induction. If no occurrences are swapped, we are left with the formula

$$(\forall x_1, \ldots, x_n : \psi \Leftrightarrow \eta) \Rightarrow (\phi \Leftrightarrow \phi)$$

which is a tautology, hence trivial. Thus we may assume that η does occur in ϕ. If ϕ is an atomic formula, then we are left with the case that $\phi = \eta$, and then

$$(\forall x_1, \ldots, x_n : \psi \Leftrightarrow \eta) \Rightarrow (\psi \Leftrightarrow \eta)$$

is an instance of (A4). The remaining cases are relatively simple, and left to the reader. □

It is also useful in mathematics to make arguments of the following form. Suppose we have a theorem of the form $(\exists x : \phi(x))$. We introduce a new constant c, which is not used in any axiom of the formal system, and consider the formula $\phi(c)$. If we end up with a formula $\eta(c)$, we conclude that $(\exists x : \phi(x))$ implies $(\exists x : \eta(x))$. Though our system is not capable of expressing these arguments, it is satisfying to know that such arguments do not increase the amount of theorems one may prove in first order logic. Temporarily, we shall write $\vdash_C \phi$ for a proof of this form. Formally, we write $\vdash_C \phi$ if there is a sequence of formulas $(\eta_1, \ldots, \eta_n)$ such that each η_i is an axiom, is inferred by (MP) or (UG) from a previous formula, or there is a preceding formula $\eta_j = (\exists x : \psi(x))$, where $\eta_i = \psi(c)$, and c is a new constant which does not occur in any prior formulae or explicitly in axioms of the formal system. We require that no application of (UG) is made on a variable which is free in some $(\exists x : \phi)$, in a formula which depends on this formula. Finally, we require that η_n does not contain any of the constants introduced by the final rule. To prove that this method can be applied to our formal system, we require a certain formula, proved now, and integral to the proof that $\vdash_C$ is redundant.

Example. *If x is not free in ψ, $((\exists x : \phi(x)) \Rightarrow \psi) \Leftrightarrow (\forall x : \phi(x) \Rightarrow \psi)$.*

$((\exists \mathsf{x} : \phi(\mathsf{x})) \Rightarrow \psi) \Leftrightarrow (\forall \mathsf{x} : \phi(\mathsf{x}) \Rightarrow \psi)$	
1. $(\exists \mathsf{x} : \phi(\mathsf{x})) \Rightarrow \psi$	
2. $\phi(\mathsf{x})$	
3. $(\exists \mathsf{x} : \phi(\mathsf{x}))$	*(EI)*
4. ψ	*(1),(3),(MP)*
5. $\phi(\mathsf{x}) \Rightarrow \psi$	*(2-4),(DT)*
6. $(\forall \mathsf{x} : \phi(\mathsf{x}) \Rightarrow \psi)$	*(UG)*
6. $((\exists \mathsf{x} : \phi(\mathsf{x})) \Rightarrow \psi) \Rightarrow (\forall \mathsf{x} : \phi(\mathsf{x}) \Rightarrow \psi)$	*(1-6),(DT)*
7. $(\forall \mathsf{x} : \phi(\mathsf{x}) \Rightarrow \psi)$	
8. $\phi(\mathsf{x}) \Rightarrow \psi$	*(A4)*
9. $\neg\psi \Rightarrow \neg\phi(\mathsf{x})$	*(8),(TAUT),(MP)*
10. $(\forall \mathsf{x} : \neg\psi \Rightarrow \neg\phi(\mathsf{x}))$	*(UG)*
11. $\neg\psi \Rightarrow (\forall \mathsf{x} : \neg\phi(\mathsf{x}))$	*(10),(A5),(MP)*
12. $\neg(\forall \mathsf{x} : \neg\phi(\mathsf{x})) \Rightarrow \psi$	*(11),(TAUT),(MP)*
13. $(\forall \mathsf{x} : \phi(\mathsf{x}) \Rightarrow \psi) \Rightarrow ((\exists \mathsf{x} : \phi(\mathsf{x})) \Rightarrow \psi)$	*(7-12),(DT)*

Theorem 3.5. *If $\vdash_C \phi$, then $\vdash \phi$.*

Proof. Let $(\eta_1, \ldots, \eta_n)$ be a proof of ϕ, and suppose that

$$\eta_{i_1} = (\exists x_1 : \nu_1(x_1)) \quad \eta_{i_2} = (\exists x_2 : \nu_2(x_2)) \quad \ldots \quad \eta_{i_m} = (\exists x_m : \nu_m(x_m))$$

are all existence formulae used in the proof, from which new constants $c_1, \ldots, c_m$ are introduced. Certainly $\nu_1(c_1), \ldots, \nu_m(c_m) \vdash \phi$. By the universal generalization condition on $\vdash_C$, we can apply the deduction theorem to conclude that $\nu_1(c_1), \ldots, \nu_{m-1}(c_{m-1}) \vdash \nu_m(c_m) \Rightarrow \phi$. In the proof of this statement, replace all instances of c_m with a new variable y_m. This is then still a valid proof (for variables can be operated on in at least the same capacity as constants), hence $\nu_1(c_1), \ldots, \nu_{m-1}(c_{m-1}) \vdash \nu_m(y_m) \Rightarrow \phi$. This implies $\nu_1(c_1), \ldots, \nu_{m-1}(c_{m-1}) \vdash (\forall y_m : \nu_m(y_m) \Rightarrow \phi)$, and by applying the recently proved example, since we know y_m does not occur in ϕ at all, we conclude $\nu_1(c_1), \ldots, \nu_{m-1}(c_{m-1}) \vdash (\exists y_m : \nu_m(y_m)) \Rightarrow \phi$. By induction, we may assume that $\nu_1(c_1), \ldots, \nu_{m-1}(c_{m-1}) \vdash (\exists y_m : \nu_m(y_m))$, which implies, by a particular use of (MP), that $\nu_1(c_1), \ldots, \nu_{m-1}(c_{m-1}) \vdash \phi$, and we may recursively prove that $\vdash \phi$. □

3.3 Interpretations

Languages are defined in terms of the subject manner we wish to study, but may be interpreted in many different ways. For instance, the axioms which define the logic of group theory may be interpreted relative to whichever group we interpret the axioms as agreeing with. We would hope that a statement is true if and only if it is true in every interpretation of the axioms. To begin discussing this, we must precisely define what we mean by interpretation. We rely on a set-theoretic interpretation formulated by Alfred Tarski in 1933, and refined together with Robert Vaught in 1956.

Let L be a first order language. Then L has a set of atomic predicates and a set of formulas. A *structure* M defined over a language L is a set U_M, sometimes called the *universe of discourse* of the structure M, equipped with an n-ary relation P_M for each n-ary predicate P in the language L, and a function $f_M : M^n \to M$ for each function f in the language L. We also call M an *interpretation* of L. We will often abuse notation by identifying M with U_M.

Example. *The standard interpretations of the constants and functions in the language of rings L_{ring} give rise to several interpretations, namely $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{Z}_2$ are all interpretations of this first order language.*

Example. *The sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ can all be interpreted in a natural way as structures over the language L_{ord}.*

Example. *Any directed graph can be interpreted as a structure over the language $L_{digraph}$. Even graphs that are* infinite *can be interpreted over this language.*

For a given first order language L, the set of all structures forms a category $\mathsf{Struct}(L)$, if we define a morphism between two structures M and N to be a function $F : U_M \to U_N$ such that:

- For each constant c in the language L, $F(c_M) = c_N$.
- For each n-ary function f and $a_1, \ldots, a_n \in U_M$,

$$F(f_M(a_1, \ldots, a_n)) = f_N(F(a_1), \ldots, F(a_n)).$$

- For each n-ary predicate P, and any $a_1, \ldots, a_n \in U_M$,

$$P_M(a_1, \ldots, a_n) \quad \text{implies} \quad P_N(F(a_1), \ldots, F(a_n)).$$

If f is a morphism satisfying the stronger condition that if $a_1, \ldots, a_n \in U_M$, then $P_M(a_1, \ldots, a_n)$ holds *if and only if* $P_N(F(a_1), \ldots, F(a_n))$ holds, we say that F is a *literal embedding*. If, for any sentence $\phi(x_1, \ldots, x_n)$, and any $a_1, \ldots, a_n \in U_M$,

$$\Gamma \vDash \phi(a_1, \ldots, a_n) \quad \text{holds if and only if} \quad \Gamma \vDash \phi(F(a_1), \ldots, F(a_n)) \quad \text{holds,}$$

then we say that F is an *elementary embedding*. Being an elementary embedding is in general a much stronger condition than being a literal embedding, but if F is surjective, than the two notions are equivalent. Note, however, that there are elementary embeddings that are *not surjective* – we will later see that the real numbers $\mathbb{R}$ can be embedded in models of non-standard analysis. An *isomorphism* of structures will be an invertible morphism in $\mathsf{Struct}(L)$, i.e. a morphism whose inverse is also a morphism. An isomorphism is then a bijective literal embedding.

Example. *Consider $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$ as structures over L_{poset}. Then the canonical map $\mathbb{N} \to \mathbb{Q}$ is a literal embedding, but not an elementary embedding, i.e. because if $\phi(x)$ is the sentence $(\forall \mathsf{y} : \mathsf{y} \geqslant \mathsf{x})$, then $\mathbb{N} \vDash \phi(0)$, but $\mathbb{Q} \nvDash \phi(0)$. On the other hand, we will later see via Vaughn's test that the theory of* dense linear orderings *is complete, i.e. any model of L_{poset} whose interpretation of $<$ gives a dense linear ordering has the same theorems. As a result, we will find that the map $\mathbb{Q} \to \mathbb{R}$ is an elementary equivalence.*

Remark. Embeddings in general need not be injective. However, we will later restrict our analysis to *theories with equality*, and to *normal models*, and for such models, a literal embedding will be injective, because literals involving the equality predicate must be preserved.

In sentential logic, when we assign a truth value to a set of variables, we may extend the definition of truth to all formulas. When we assign a meaning to each variable in a first order language, we may define a meaning on all terms, and from these meanings, assign truth to statements in the corresponding language. Consider a particular interpretation of a first order language with variables Λ, and consider an *assignment* $f : \Lambda \to M$, i.e. a map from the variables of a language to particular elements of the universe of discourse. Extend the domain of f to the set of all terms by defining

$$f(c) = c_M \qquad f(g(t_1, \ldots, t_n)) = g_M(f(t_1), \ldots, f(t_n))$$

Using this definition, we may define whether a formula is satisfied in a model of a first order theory. We shall now define what it means for an assignment to

satisfy a formula in an interpretation. Given an assignment f, let $f[a/x]$ denote the assignment

$$f[a/x](y) = \begin{cases} f(y) & y \neq x, \\ a & y = x. \end{cases}$$

Let us now determine what it means for an assignment of variables to satisfy a statement:

1. f satisfies $P(t_1, \ldots, t_n)$ if $P_M(f(t_1), \ldots, f(t_n))$ holds.
2. f satisfies $(\forall x : \phi)$ if ϕ is satisfied by $f[a/x]$, for all $a \in M$.
3. f satisfies $(\exists x : \phi)$ if there is some $a \in M$ such that $f[a/x]$ satisfies ϕ.
4. An assignment f satisfies $\phi \circ \eta$ or $\neg\phi$, where $\circ$ and $\neg$ are logical connectives, if the truth evaluation of ϕ and η under f is consistent with the connectives as in sentential logic.

If M is an interpretation, then a formula ϕ is *valid* for an interpretation, denoted $M \models \phi$, if ϕ is true under every assignment under M. A statement is false if it is true under no interpretation, or alternatively, if the negation of the statement is true under the interpretation. An interpretation is a *model* for a set of formulas Γ if every formula in Γ is true for the interpretation. The deductive theory $\mathrm{Th}(M)$ consisting of all ϕ such that $M \models \phi$ is called the *theory* of M.

Example. *The formula $(\forall x, \exists y : x + y = 0)$ in the language of arithmetic we have seen before, is satisfied by the interpretations $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$, but not $\mathbb{N}$. Thus $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$ are models of this formula.*

Lemma 3.6. *If $M \models \phi$ and $M \models (\phi \Rightarrow \psi)$, then $M \models \psi$.*

Lemma 3.7. *If a formula ϕ contains free variables $x_1, \ldots, x_n$, and two assignments $f : \Lambda \to M$ and $g : \Lambda \to M$ agree on the free variables, then f satisfies ϕ if and only if g satisfies ϕ.*

Proof. We shall first verify that if t is a term containing variables $x_1, \ldots, x_n$, on which f and g agree, then $f(t) = g(t)$. If t is a variable, or a constant, the proof is easy. But then by induction, for an n-ary function u,we have

$$\begin{aligned} f(u(t_1, \ldots, t_n)) &= u_M(f(t_1), \ldots, f(t_n)) \\ &= u_M(g(t_1), \ldots, g(t_n)) \\ &= g(u(t_1, \ldots, t_n)) \end{aligned}$$

Thus the theorem is verified by structural induction. If ϕ is an atomic formula $P(t_1, \ldots, t_n)$, then f satisfies $P(t_1, \ldots, t_n)$ if and only if g does, because $f(t_i) = g(t_i)$. If ϕ is $(\forall x : \psi)$, and f satisfies ϕ, then $f[a/x]$ satisfies ψ for all $a \in M$. But then $g[a/x]$ agrees on all free variables of $f[a/x]$, so $g[a/x]$ satisfies ψ by induction. It follows that g satisfies ϕ as well. The remaining cases are easily shown, and are left as an exercise. □

Theorem 3.8. *If ϕ contains no free variables, then either $M \vDash \phi$ or $M \vDash \neg\phi$.*

Proof. We apply the previous lemma, with no free variables at all. It follows that either ϕ is satisfied by all assignments, or by none of them. In the former case, we have $M \vDash \phi$, and in the latter, $M \vDash \neg\phi$. □

Lemma 3.9. *$M \vDash (\exists x : \phi)$ if and only if $M \vDash \neg(\forall x : \neg\phi)$.*

Proof. If an assignment f satisfies $(\exists x : \phi)$, then ϕ is satisfied by some $f[a/x]$. But then f does not satisfy $(\forall x : \neg\phi)$ for $f[a/x]$ does not satisfy $\neg\phi$. Conversely, if an assignment f does not satisfy $(\exists x : \phi)$, then every $f[a/x]$ satisfies $\neg\phi$. □

Lemma 3.10. *$M \vDash \phi$ if and only if $M \vDash (\forall x : \phi)$.*

Proof. If $M \vDash \phi$, then every assignment f satisfies ϕ, so certainly every $f[a/x]$ satisfies ϕ, and thus f satisfies $(\forall x : \phi)$, hence $M \vDash (\forall x : \phi)$. Conversely, suppose $M \vDash (\forall x : \phi)$. Then, every assignment satisfies $(\forall x : \phi)$, and thus in particular satisfies ϕ. □

Let ϕ contain free variables $x_1, \ldots, x_n$. The *closure* of ϕ is the string

$$(\forall x_1, x_2, \ldots, x_n : \phi).$$

The above proof shows a formula is satisfied if and only if its closure is, so it is sufficient to study the semantics of closed formulas.

Theorem 3.11. *Consider a form of sentential logic, whose variables are all atomic formulas, and we interpret formulas of the form $(\forall x : \phi)$, and $(\exists x : \phi)$ as 'variables' as well. Then if a statement is a tautology, then it is satisfied under all interpretations.*

Proof. The connectives of predicate logic are exactly the connectives of sentential logic once we hide away the existential and universal quantifiers. If the statement is a tautology, then regardless of how we interpret the formula, the statement will be satisfied. □

Lemma 3.12. *If t and u are terms, x is a variable, and f is an assignment, then*

$$f[f(u)/x](t) = f(t[u/x])$$

Proof. If t is a variable unequal to x, or t is a constant, then

$$f[f(u)/x](t) = f(t) = f(t[u/x])$$

If $t = x$, then

$$f[f(u)/x](t) = f(u) = f(t[u/x])$$

For a structural induction, let $t = g(t_1, \ldots, t_n)$. Then

$$\begin{aligned} f[f(u)/x](t) &= g_M(f[u/x](t_1), \ldots, f[u/x](t_n)) \\ &= g_M(f(t_1[u/x]), \ldots, f(t_n[u/x])) = f(t[u/x]) \end{aligned}$$

Thus the theorem holds in general. □

Lemma 3.13. *f satisfies $\phi(u)$ if and only if $f[f(u)/x]$ satisfies $\phi(x)$.*

Proof. If $\phi(x)$ is $P(t_1(x), \ldots, t_n(x))$, then $\phi(u) = P(t_1(u), \ldots, t_n(u))$, and

$$P_M(f(t_1(u)), \ldots, f(t_n(u))) = P_M(f[f(u)/x](t_1), \ldots, f[f(u)/x](t_n))$$

Which shows that f satisfies $\phi(u)$ if and only if $f[f(u)/x]$ satisfies ϕ. If ϕ is formed by standard sentential connectives, the theorem is trivial. If $\phi = (\forall y : \psi)$, where $y \neq x$, then $\phi(u) = (\forall y : \psi[u/x])$, and by definition, f satisfies $\phi(u)$ if and only if $f[a/y]$ satisfies $\psi(u)$ for all a, which by induction implies that $f[f(u)/x][a/y]$ satisfies ψ for all a, so $f[f(u)/x]$ satisfies ϕ. Similar results hold if ϕ's primitive connective is the existential quantifier. □

Theorem 3.14. *For any formula ϕ and term t free for x, $M \models (\forall x : \phi) \Rightarrow \phi(t)$*

Proof. Let f be an assignment satisfying $(\forall x : \phi)$. Then $f[f(t)/x]$ satisfies ϕ, so f satisfies $\phi(t)$. □

Theorem 3.15. *If ϕ does not contain x as a free variable, then*

$$M \models (\forall x : \phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow (\forall x : \psi))$$

Proof. If f satisfies $(\forall x : \phi \Rightarrow \psi)$ and ϕ, then $f[a/x]$ satisfies $\phi \Rightarrow \psi$, and since ϕ doesn't contain x as a free variable, $f[a/x]$ also satisfies ϕ, so f satisfies $(\forall x : \psi)$. □

Given a first order language F, together with a specific interpretation M with universe of discourse U, we can generate a new first order language, $L(M)$, by including all elements of the set U as constants. The interpretation of F in M extends naturally to an interpretation of $L(M)$, by interpreting the new constants in $L(M)$ as expected. This allows us to discuss formulas of the form $M \models \phi(u_1, \ldots, u_n)$, where $u_1, \ldots, u_n \in U$ are formulas containing elements of U. If the free variables of a formula ϕ are $x_1, \ldots, x_n$, then the set $\{u \in U^n : M \models \phi(u_1, \ldots, u_n)\}$ will be called the relation relative to ϕ.

A statement is *logically valid* if it is true under all possible interpretations. A statement is *satisfiable* if it is true under at least one interpretation, and *contradictory* if it is false under every interpretation. A set of statements is satisfiable if they can be simultaneously satisfied by some interpretation. A statement ϕ is a *logical consequence* of a set of statements Γ if every interpretation which satisfies every statement of Γ also satisfies ϕ. It is clear that in the system of first order logic we have produced is *sound* for the semantics we have just provided, i.e. it only produces logically valid statements. In the next section, we will prove the system is also *complete*, i.e. *all* logically valid statements are provable.

3.4 Gödel's Completeness Theorem

The completeness theorem is best understood in the context of *theories*, which are subsets Γ of sentences in a first order language L. We say a theory Γ is a *deductive theory* if it is closed under deduction, in the sense that if $\Gamma \vdash \phi$, then $\phi \in \Gamma$. An *extension* of a theory is just a theory which is a superset of the original theory. If Γ is a theory, we let $\mathrm{Th}(\Gamma)$ denote the *deductive closure* of Γ, i.e. the smallest deductive theory containing Γ. A set Γ is *consistent* if for any sentence ϕ, we do not have both $\Gamma \vdash \phi$ and $\Gamma \vdash \neg\phi$, or equivalently, if $\mathrm{Th}(\Gamma) \neq L$. A *complete* theory is a theory in which $\Gamma \vdash \phi$ or $\Gamma \vdash \neg\phi$ always holds. If a theory is consistent or complete, so is its deductive closure.

At times, we shall consider extensions of first order language L_1, i.e. other languages L_2 obtained by adding additional prepositions, constants, and variable symbols into the language. It is simple to see that if Γ is a theory of L_1, then $\mathrm{Th}_{L_1}(\Gamma) = \mathrm{Th}_{L_2}(\Gamma) \cap L_1$, i.e. adding extra symbols does not increase the deductive power of a first order language.

Lemma 3.16. *If a theory has a model, then it is consistent.*

Proof. If Γ has a model, and $\Gamma \vdash \phi$ and $\Gamma \vdash \neg\phi$, then ϕ and $\neg\phi$ must both be

valid in the model, by soundness of our axiom system. This is impossible, which gives a contradiction. □

Example. *Consider the language of groups, i.e. the first order language L_{group} with a constant e, a binary multiplication function $\cdot$, and a binary predicate $=$. The theory of groups is the theory Γ_{group} over L_{group} given by the four axioms*

$$x(yz) = (xy)z$$

$$ex = x \qquad xe = x$$

$$(\forall x, \exists y : xy = e \wedge yx = e)$$

together with the equality axioms

$$(x = y) \Rightarrow (y = x) \qquad ((x = y) \wedge (y = z)) \Rightarrow (x = z) \qquad (x = y) \Rightarrow (xz = xy)$$

This set of statements is a first order theory. It is also interesting to note that the theory is finitely axiomatizable*, in the sense that the theory Γ_{group} is a set of eight statements. Any model of this theory is just a plain old group, with a slight nuance; we need not interpret the $=$ relation in our theory as true equality in our model, but instead only an equivalence relation. In the case where the $=$ relation is interpreted as true equality (in what we will soon call a* normal model*), we can conclude that the model is actually a group. The completeness theorem will show that every theorem proved in this formal system is valid in any group. This theory is consistent, but certainly not complete, for $\Gamma_{group} \cup \{(\forall x, y : xy = yx)\}$ and $\Gamma_{group} \cup \{\neg(\forall x, y : xy = yx)\}$ are both consistent theories (abelian and non abelian groups exist), so by soundness, it cannot be true that $\Gamma_{group} \vdash (\forall x, y : xy = yx)$ nor can it be true that $\Gamma_{group} \vdash \neg(\forall x, y : xy = yx)$.*

Example. *The theory of equality $\Gamma_{equality}$ is defined over the language of equality $L_{equality}$, which has a single binary predicate $=$. The axioms of $\Gamma_{equality}$ are*

$$(x = y \wedge y = z) \Rightarrow x = z \qquad x = x \qquad x = y \Rightarrow y = x.$$

A model of the theory of equality is exactly a set with an equivalence relation. We can extend this theory to the theory of partial orders, by considering the language L_{poset} extending $L_{equality}$ by the addition of another binary predicate $<$. The theory Γ_{poset} is given by all statements in $\Gamma_{equality}$, together with the three additional statements.

$$x = y \Rightarrow \neg(x < y) \qquad (x < y \wedge y < z) \Rightarrow x < z$$

the theory Γ_{poset} *of partial orders can be extended to the theory of linear orders* Γ_{linear} *by the addition of a single extra statement*

$$x < y \vee x = y \vee y < x.$$

We can consider the theory of linear partial orders with no last element by adding the two statements

$$(\forall x, \exists y : y < x) \quad (\forall x, \exists y : x < y).$$

The theory $\Gamma_{dense\ poset}$ *extends* Γ_{poset} *by the addition of a single extra axiom*

$$(x < y) \Rightarrow (\exists z : x < z \wedge z < y).$$

Models of algebraic systems are interesting as a toy example to test our techniques mathematical logic, but logic will also give us some very interesting insights in algebra, as we will see later.

Theorem 3.17. *If* Γ *is consistent, and if* $\Gamma \nvdash \neg\phi$*, then* $\Gamma \cup \{\phi\}$ *is also consistent.*

Proof. Suppose that $\Gamma \cup \{\phi\} \vdash \psi$ and $\Gamma \cup \{\phi\} \vdash \neg\psi$. Without loss of generality, we may assume that ϕ contains no free variables, for the logical closure of $\Gamma \cup \{\phi\}$ is the same as the logical closure of $\Gamma \cup \{(\forall x_1, \ldots, x_n : \phi)\}$, because of axiom (A4) and (UG). But then, by the deduction theorem,

$$\Gamma \vdash \phi \Rightarrow \psi \qquad \Gamma \vdash \phi \Rightarrow \neg\psi$$

But then $\Gamma \vdash \neg\phi$, because

$$((\phi \Rightarrow \psi) \wedge (\phi \Rightarrow \neg\psi)) \Rightarrow \neg\phi$$

is a tautology, which gives a contradiction. □

Lemma 3.18 (Lindenbaum)**.** *If* Γ *is a consistent theory, then there is a complete consistent extension of* Γ.

Proof. Let $\mathcal{K}$ be the set of all consistent extensions of Γ, ordered by inclusion. If $\mathcal{A}$ is a chain of consistent extensions of Γ, then we claim $\bigcup \mathcal{A}$ is consistent. Suppose that

$$\bigcup \mathcal{A} \vdash \phi \qquad \bigcup \mathcal{A} \vdash \neg\phi$$

Then there is a proof $(\eta_1, \eta_2, \ldots, \eta_n)$ of ϕ from $\bigcup \mathcal{A}$, and a proof $(\nu_1, \nu_2, \ldots, \nu_n)$ of $\neg\phi$. Since each side is finite, each uses only finitely many axioms, which implies that there is $\Gamma \in \mathcal{A}$ containing all axioms. But then $\Gamma \vdash \phi$ and $\Gamma \vdash \neg\phi$, which implies Γ is not consistent, a contradiction. By Zorn's lemma, there is a maximal consistent extension Γ. Γ is complete, by the last lemma. If $\Gamma \nvdash \neg\phi$, then $\Gamma \cup \{\neg\phi\}$ is consistent, implying $\Gamma \cup \{\neg\phi\} = \Gamma$, so $\phi \in \Gamma$, implying $\Gamma \vdash \phi$. □

It is important to note this theorem as the first non constructive theorem in these notes. There is effectively no way to make a general version of this theorem constructive, because we will find that there are certain limitations to constructive arguments in first order logic, which manifest in undecidability results, like Gödel's incompleteness theorem.

A theory Γ is a *scapegoat theory* if for any formula ϕ which only has one free variable x, there is a closed term t for which

$$\Gamma \vdash (\exists x : \neg\phi(x)) \Rightarrow \neg\phi[t/x]$$

Scapegoat theories are useful for proving the completeness theorem, for it is easier to move constants into interpretations than with plain formulas.

Lemma 3.19. *Every consistent theory Γ has a consistent scapegoat extension Γ which has the same cardinality as Γ if Γ is infinite, or is denumerable if Γ is finite.*

Proof. The deductive closure of Γ has the same cardinality as Γ in the infinite case, or is denumerable in the finite case. Since the deductive closure extends Γ, we may, without loss of generality, assume Γ is deductively closed. Let $\mathcal{X}$ be the set of all variables in all formulas of Γ which have exactly one free variable, and biject them with a set C disjoint from all preexisting characters in the first order language of Γ. Let c_x be the element of C in bijection with $\mathcal{X}$. Let Γ_∞ be the theory obtained from Γ by adding C as constants to the theory, together with the formulas $(\exists x : \neg\phi) \Rightarrow \neg\phi(c_x)$. Γ_∞ is surely a scapegoat by construction. We claim that Γ_∞ is consistent and scapegoat. It suffices, since every proof is finite, to show that every finite subextension is consistent. Consider any particular variables $x_1, x_2, \ldots, x_n \in \mathcal{X}$, and let $\Gamma_0 = \Gamma$, and Γ_k be the theory obtained from Γ_{k-1} be adding the contant c_{x_k} and the axiom $(\exists x_k : \neg\phi_k(x_k)) \Rightarrow \neg\phi_k(c_x)$. We prove consistency by induction on k. Suppose Γ_{k-1} is consistent, and Γ_k is inconsistent. Then we may prove all statements in Γ_{k-1} in Γ_k. In particular, $\Gamma_k \vdash \neg((\exists x_k : \neg\phi_k(x_k)) \Rightarrow \neg\phi_k(c_{x_k}))$. Since this formula is closed (because x is the only free variable in ϕ_k), we may apply the deduction theorem to conclude

$$\Gamma_{k-1} \vdash ((\exists x_k : \neg\phi_k(x_k)) \Rightarrow \neg\phi_k(c_{x_k})) \Rightarrow \neg((\exists x_k : \neg\phi_k(x_k)) \Rightarrow \neg\phi_k(c_{x_k}))$$

which allows us to conclude (by the tautology $(A \Rightarrow \neg A) \Rightarrow \neg A$), that $\Gamma_{k-1} \vdash \neg((\exists x_k : \neg\phi_k(x_k)) \Rightarrow \neg\phi_k(c_{x_k}))$, hence

$$\Gamma_{k-1} \vdash (\exists x_k : \neg\phi_k(x_k)) \qquad \Gamma_{k-1} \vdash \phi_k(c_{x_k})$$

Since c_{x_k} does not occur in the axioms of Γ_{k-1}, we conclude that $\Gamma_{k-1} \vdash \phi_k(y_k)$, by replacing all occurences of c_{x_k} in the proof of $\phi_k(c_{x_k})$ by a new variable y_k which does not occur in the proof. But then $\Gamma_{k-1} \vdash (\forall y_k : \phi_k(y_k))$, so $\Gamma_{k-1} \vdash \phi_k(x_k)$, hence $\Gamma_{k-1} \vdash (\forall x_k : \phi_k(x_k))$, contradicting $(\exists x_k : \neg\phi_k)$, and implying that Γ_{k-1} is inconsistent. Thus we have shown Γ_∞ is consistent. □

Lemma 3.20. *Let* Γ *be a consistent, complete, scapegoat theory. Then* Γ *has a model* M *whose universe of discourse is the set of closed terms in* Γ.

Proof. For each constant c in the language, let $c^M = c$. For each function f, let $f^M(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ (by assumption, each t_i is closed). For each predicate P, let $(t_1, \ldots, t_n) \in P^M$ if and only if $\vdash P(t_1, \ldots, t_n)$. We shall show that $M \models \phi$ if and only if $\Gamma \vdash \phi$, for any closed formula ϕ. This implies that M is a model of Γ, for then, if ϕ is any axiom of Γ, then $\Gamma \vdash (\forall x_1, \ldots, x_n : \phi)$, hence $M \models (\forall x_1, \ldots, x_n : \phi)$, which occurs if and only if $M \models \phi$. Thus M models all axioms. We will prove our statement by structural induction.

1. If ϕ is $P(t_1, \ldots, t_n)$, the statement is trivial by construction.

2. Suppose ϕ is $\neg\psi$. If $M \models \neg\psi$, then $M \not\models \psi$, so $\Gamma \not\vdash \psi$, and so $\Gamma \vdash \neg\psi$ by completeness. Conversely, if $M \not\models \neg\psi$, then $M \models \psi$, so by induction $\Gamma \vdash \psi$, hence by consistency $\Gamma \not\vdash \neg\psi$.

3. If ϕ is $\psi \Rightarrow \eta$, since ϕ is closed, ψ and η are closed. If $M \not\models \psi$, then $M \models \psi \Rightarrow \eta$, and by induction $\Gamma \not\vdash \psi$, so $\Gamma \vdash \neg\psi$, and then $\Gamma \vdash \psi \Rightarrow \eta$ by tautology. The remaining cases are again treated by tautology and completeness.

4. If ϕ is $(\forall x : \psi)$, then either ψ is closed, or $\psi(x)$ has a single free variable x. If ψ is closed, the statement follows from tautologies and completeness fairly easily. We shall treat the other case in detail. Suppose that $M \models (\forall x : \psi(x))$, yet $\Gamma \not\vdash (\forall x : \psi(x))$. Thus $\Gamma \vdash \neg(\forall x : \psi(x))$. Since Γ is scapegoat, there is a constant c such that $\Gamma \vdash \neg\psi(c)$, from which we conclude $M \models \neg\psi(c)$ by induction, contradicting that $M \vdash (\forall x : \psi(x))$. Conversely, suppose $M \not\models (\forall x : \psi(x))$, yet $\Gamma \vdash (\forall x : \psi(x))$. Then $\Gamma \vdash \psi(t)$ for any term t, so by induction, $M \models \psi(t)$ for all closed terms t. Let f be an assignment on M such that which does not satisfy ψ. Let $f(x) = t$. Since t is a closed term in the interpretation, $f(t) = t$, and $f = f[f(t)/x]$, so a previous lemma implies that f cannot satisfy $\psi(t)$. Thus $M \not\models \psi(t)$, a contradiction.

We have addressed all cases, so our proof is complete. □

Theorem 3.21. *Every consistent theory has a model whose cardinality is the same as the theory itself, unless the theory is finite, in which the model is denumerable.*

Proof. Let Γ be a consistent theory. Extend Γ to a consistent, complete, scapegoat theory Γ', which is denumerable if Γ is finite or denumerable, or else Γ has the same cardinality as Γ'. But then Γ' has a model consisting of closed terms in Γ', whose cardinality is the same as the Γ. □

We've built up enough theory to prove the first fundamental result about first order logic: Gödel's completeness theorem (named after Kurt Gödel, who first proved the theorem in 1930). Our version of the proof is quite different to Gödel's first proof, our version being based on ideas of Leon Henkin.

Theorem 3.22 (Gödel's Completeness Theorem)**.** *A formula is provable in a theory if and only if it is true under all interpretations.*

Proof. Let Γ be a theory. Without loss of generality, assume Γ is consistent, because otherwise Γ has no models, in which case the completeness theorem holds vacously. If $\Gamma \vdash \phi$, then we have already shown ϕ is true in all models. Conversely, if $\Gamma \nvdash \phi$, then $\Gamma \cup \{\neg\phi\}$ is consistent, so $\Gamma \cup \{\neg\phi\}$ has a model M. But then M is a model for Γ such that $M \models \neg\phi$. □

Thus syntax and semantics coincide in the classical case for first order logic. We actually proved something much stronger, that a formula is provable in a theory if and only if it is true for all countable models, if the theory is finite, or equal to the theories cardinality if the theory is infinite. This is important, because it hints at further results that first order logic is not very good at distinguishing between models of different cardinalities. It is also important to note that our proof is non-constructive. We did not construct a formal proof given that the formula was true under all interpretations. Indeed, we will later see that the completeness theorem is equivalent to the axiom of choice, so must involve some non-constructive procedures.

3.5 The Compactness Theorem

The fact that every consistent theory has a model is incredibly important, because we can now combine a fundamental 'finiteness' of syntactic proofs, with semantic results, to obtain some powerful consequences.

Theorem 3.23 (The Compactness Theorem)**.** Γ *is a consistent theory if and only if every finite subset of* Γ *is consistent. Conversely,* Γ *has a model if and only if every finite subset of* Γ *has a model.*

Proof. If Γ is consistent, than a finite subset is consistent. Conversely, suppose Γ is inconsistent, and consider proofs of two statements $\Gamma \vdash \phi$ and $\Gamma \vdash \neg\phi$. These proofs only use finitely many axioms in Γ, so there is some finite subset Δ of Γ such that $\Delta \vdash \phi$ and $\Delta \vdash \neg\phi$, so some finite subset is inconsistent. □

Remark. Let Γ be a consistent theory, and suppose Λ is a theory which is *closed under conjunction*. We claim that the compactness theorem is equivalent to the fact that if $\Gamma \cup \Lambda$ is inconsistent, then there is $\psi \in \Lambda$ such that $\Gamma \vdash \neg\psi$. Indeed, suppose $\Gamma \cup \Lambda$ is inconsistent, then compactness implies that $\Gamma \cup \{\phi_1, \ldots, \phi_n\}$ is inconsistent for some $\{\phi_1, \ldots, \phi_n\} \subset \Lambda$. But then if $\psi = \phi_1 \wedge \cdots \wedge \phi_n$, then $\Gamma \cup \{\psi\}$ is inconsistent, which implies that for every model M of Γ, $M \models \neg\psi$, and thus by the completeness theorem, $\Gamma \vdash \neg\psi$.

To see the equivalence with the compactness theorem, suppose Λ is a theory, such that every finite subset is consistent. Without loss of generality, let us assume that Γ is closed under conjunction, so that the finiteness condition is equivalent to the fact that $\{\phi\}$ is consistent for each $\phi \in \Gamma$. If Λ is inconsistent, the result above would imply that there was $\phi \in \Lambda$ such that $\vdash \neg\phi$. But then $\{\phi\}$ would be inconsistent.

There are applications to the compactness theorem in widely varying areas of mathematics. Here is a classic example, which uses the theorem to construct an *algebraic closure* of any field.

Example. *Let* L_{ring} *be the language of rings. The theory of fields can be introduced by translating the axioms of field theory into the language of first order logic. For instance, the associativity of addition can be written in the language* L *as the statement*

$$(\forall a, b, c : (a + b) + c = a + (b + c)).$$

Putting all the axioms together, we can find a theory Γ *over* L *which encapsulates the theory of fields, i.e. such that any normal model (the interpretation of the equality operator is the normal equality operator) of* Γ *has the structure of a field via the interpretation of the constants* 0 *and* 1*, and binary operators* $+$*, and* $\cdot$*. We will use this model to show that* any field has an algebraic closure*, i.e. for any field* k*, there exists an algebraically closed field* $k_{\mathfrak{a}}$ *containing* k *as a subfield, i.e. a field such that every non constant univariate polynomial has a root.*

Given a field k, augment the language L to a language $L(k)$, containing all elements of k as constants. Also enlarge the theory Γ to a theory $\Gamma(k)$, which contains all algebraic relations in k as axioms (the new theorems correspond to what we call the diagram *of k in more advanced model theory). This theory is consistent, because k can be used to produce a model of the theory, in the natural way. The models of $\Gamma(k)$ correspond precisely to* field extensions *of k, i.e. because for any other model k' of $\Gamma(k)$, the interpretation of constants in the language $L(k)$, together with the extra axioms in $\Gamma(k)$, implies that we have a natural morphism of fields $k \to k'$. Now consider the theory $\Gamma(k)_\mathfrak{a}$, which contains $\Gamma(k)$, as well as all statements of the form*

$$(\exists x : p(x) = 0),$$

for any univariate non-constant polynomial $p \in k[x]$. We claim this theory is consistent. Indeed, any finite subtheory *of $\Gamma(k)_\mathfrak{a}$ is consistent, because for any finite family of polynomials $p_1, \ldots, p_n$, we can find a field extension k' of k containing at least one root of each of these polynomials, and so any finite subtheory of $\Gamma(k)_\mathfrak{a}$ contains a model. But this means that $\Gamma(k)_\mathfrak{a}$ has a model k', i.e. there exists a field extension of k containing a root of every non-constant polynomials in $k[x]$.*

Now we iterate this process. We let $k = k_0$, $k' = k_1$, and then find a field k_2 extending k_1, which contains roots for all polynomials in $k_1[x]$. We then consider k_3, k_4, and so on. The union $k_\mathfrak{a} = \bigcup k_i$ is a field, and all non-constant polynomials in $k_\mathfrak{a}[x]$ have roots, because any polynomial in $k_\mathfrak{a}[x]$ actually lies in $k_n[x]$ for some large n, another manifestation of syntactic compactness. Thus $k_\mathfrak{a}$ is algebraically closed.

Example. *The theory of fields can be extended to the theory of fields of characteristic p, for some prime $p > 0$, by adding the axiom $p = 0$, where $p = 1 + \cdots + 1$, to the theory of fields. The theory of fields of characteristic zero is obtained by adding the axioms $n \neq 0$, for any integer n (where n is expressed syntactically as the n-fold sum $1 + 1 + \cdots + 1$). Now if s is any theorem which is true in the theory of fields of characteristic zero, then it can only use finitely many of the axioms of this theory, i.e. there exists n_0 such that the proof only uses the axioms $n \neq 0$ for $n \leqslant n_0$. But this means that the theorem remains true in fields of characteristic p, for any $p \geqslant n_0$. Thus theorems expressed in the first order theory of fields of characteristic zero* must also be true *for fields of suitably large characteristic.*

Example (De Bruijn Erdös Theorem)**.** *A graph G with possibly infinitely many vertices V is n-colorable if there is a map $f : V \to \{1, \ldots, n\}$ with $f(v) \neq f(w)$ if (v, w) is an edge in G. We claim that if every finite subgraph of a graph is*

n-colorable, then the entire graph is n-colorable. Consider the first order theory with a binary equality predicate =, a binary 'edge' predicate E, and n predicates $P_1, \ldots, P_n$. We interpret the universe of this theory as representing the vertices of a n-colored graph, with the predicate $E(x, y)$ being true when x and y have an edge between them, and the predicate $P_i(x)$ being true if the vertex x has color i. In addition to the usual axioms of equality (see the next chapter), we consider the additional axioms

$$\neg E(x, x) \qquad E(x, y) \Rightarrow E(y, x)$$

$$A_1(x) \vee A_2(x) \vee \cdots \vee A_n(x)$$

$$(A_i(x) \wedge A_i(y)) \Rightarrow \neg E(x, y)$$

The models of this theory are precisely the n-colorable graphs. Given a graph G, augment this theory by adding constants a_v for each node $v \in G$, as well as adding the axioms $a_v \neq a_w$ for $v \neq w$. This theory is consistent if and only if G is n-colorable. By the compactness theorem, the theory is consistent if and only if every finite sub-theory is consistent. But every finite sub-theory has a model, namely, some finite n-colorable subgraph of G. Thus the theory is consistent. Alternative proofs of this theorem without mathematical logic require much more sophisticated techniques, e.g. relying on topological results such as Tychonoff's theorem.

Consider the space of all consistent theories in a first order logic, and in particular, take the family $U_\phi = \{\Gamma : \Gamma \vdash \phi\}$. Then U_ϕ defines a basis for a topology, for $U_\phi \cap U_\psi = U_{\phi \wedge \psi}$. A net $\{\Gamma_\alpha\}$ converges to some theory Γ in this topology if and only if every statement provable in Γ is eventually provable in the net. Every element of the basis is clopen, for $U_\phi^c = U_{\neg\phi}$, so the space is completely disconnected. The compactness theorem is equivalent to the fact that the set C of complete, consistent theories being topologically compact. Indeed, any open cover of C by a family of open sets generated by formulas $\{\phi_\alpha\}$; we claim that the theory $\{\neg\phi_\alpha\}$ is inconsistent. Otherwise, the theory would have a complete, consistent extension, and then $\{\phi_\alpha\}$ would not cover C. Conversely, for any inconsistent theory Γ, $\{U_{\neg\phi} : \phi \in \Gamma\}$ is a cover of C, since Γ cannot be contained in any complete consistent theory.

We can similarly consider a topology on the class $\mathcal{M}$ of all models of a first order system, by considering the family $V_\phi = \{M \in \mathcal{M} : M \models \phi\}$ as open neighbourhoods. Then $M_\alpha \to M$ only when any formula ϕ satisfied by M is eventually satisfied in M_α for sufficiently large α. For any model M, define $\mathrm{Th}(M)$ to be the set of all *closed* formulas ϕ for which $M \models \phi$. Then $\mathrm{Th}(M)$ is a consistent

theory, which we claim is also complete. Let ϕ be any formula, and let ϕ' be its closure. Then either $M \models \phi'$ or $M \models \neg\phi'$, which implies either ϕ or ϕ' is in $\mathrm{Th}(M)$, but then either $\mathrm{Th}(M) \vdash \phi$ or $\mathrm{Th}(M) \vdash \neg\phi$. We claim that the map $M \mapsto \mathrm{Th}(M)$, from $\mathcal{M}$ to C, is continuous. Given a formula ϕ, consider all M such that $\mathrm{Th}(M) \vdash \phi$. If ϕ' is the closure of ϕ, then $U_\phi = U_{\phi'}$, so we may assume ϕ is closed. But then either $M \models \phi$ or $M \models \neg\phi$, and since $\mathrm{Th}(M)$ is consistent, we must have $M \models \phi$. If $M \models \phi$, then obviously $\mathrm{Th}(M) \vdash \phi$, so $\mathrm{Th}^{-1}(U_\phi) = V_\phi$. The surjectivity of Th is equivalent to the fact that every complete consistent theory has a model. However, we shall see in the next section that it is impossible for Th to be injective, even if we reduce $\mathcal{M}$ to isomorphism classes of theories, unless we limit our models to only having a certain cardinality. Since $\mathrm{Th}(U_\phi) = V_\phi$ for each closed formula ϕ, the map is also open, implying $\mathcal{M}$ is compact. A useful corollary is that

Theorem 3.24. *Any sequence of models contains a convergent subsequence.*

This theorem has interesting consequences. First, note that if Γ is a theory, then the set of all models of a theory Γ is closed in $\mathcal{M}$, since the set of all models M which model Γ are also all models such that $\mathrm{Th}(\Gamma)$ proves all of Γ, which is

$$\mathrm{Th}^{-1}\left(\bigcap_{\phi\in\Gamma} U_\phi\right)$$

and each U_ϕ is closed, since its complement is open.

Example. *Let Γ_{peano} be the Peano theory of arithmetic. Supplement the standard definition of Peano arithmetic by adding an additional constant c to the theory. For each natural number n, add the axiom $c > n$ to the theory. By the compactness theorem, the theory $\Gamma_{peano} \cup \{c > n : n \in \mathbb{N}\}$ is consistent. Thus it must have a model M, a model of Peano arithmetic which contains a 'number' which is larger than any natural number, despite the fact that Γ_{peano} contains axioms given by the induction schema, which implies that properties of c can be inferred purely from properties of 0, and properties implied by the successor function.*

3.6 Axiomatizations

Given a class C of structures on a language L, we say C is *axiomatizable* if there is a theory Γ such that the models of Γ are precisely the elements of C (such a theory

is called an *axiomatization* of C). A class is *finitely axiomatizable* if there is an axiom system describing the class with only finitely many axioms.

Theorem 3.25. *If $\mathcal{A}$ is finitely axiomatizable, and $\mathcal{A}$ is the class of models of some theory Γ, then some finite subset of Γ axiomatizes Γ.*

Proof. Let Γ_0 be a finite theory such that $\mathcal{A}$ is the family of models for Γ_0. For each statement $\phi \in \Gamma_0$, ϕ is true in every model of Γ, which by the completeness theorem implies that $\Gamma \vdash \phi$. But the proof of each ϕ in Γ_0 can only use finitely many axioms in Γ. Thus there is a finite subset Γ' of Γ such that every statement in Γ_0 is deducible from the axioms in Γ'. But this means that Γ' axiomatizes Γ. □

Theorem 3.26. *$\mathcal{A}$ is finitely axiomatizable iff $\mathcal{A}$ and $\mathcal{A}^c$ are axiomatizable.*

Proof. If Δ axiomatizes $\mathcal{A}$, and Π axiomatizes $\mathcal{A}^c$, then $\Delta \cup \Pi$ axiomatizes

$$\mathcal{A} \cap \mathcal{A}^c = \varnothing.$$

Thus $\Delta \cup \Pi$ is inconsistent, which, by the compactness theorem, implies that there is some $\phi_1, \ldots, \phi_n \in \Delta$, and $\psi_1, \ldots, \psi_m \in \Pi$ such that $\{\phi_1, \ldots, \phi_n\} \cup \{\psi_1, \ldots, \psi_m\}$ is inconsistent. The models of $\{\phi_1, \ldots, \phi_n\}$ and $\{\psi_1, \ldots, \psi_m\}$ must therefore be disjoint from one another. But the models of $\{\phi_1, \ldots, \phi_n\}$ contain $\mathcal{A}$, and the models of $\{\psi_1, \ldots, \psi_m\}$ contain $\mathcal{A}^c$, which implies in fact that $\{\phi_1, \ldots, \phi_n\}$ gives an axiomatization of $\mathcal{A}$, and $\{\psi_1, \ldots, \psi_m\}$ gives an axiomization of $\mathcal{A}^c$. Conversely, if $\mathcal{A}$ has a finite axiomatization $\{\phi_1, \ldots, \phi_n\}$, then $\mathcal{A}^c$ has an axiomization given by $\{\neg\phi_1 \vee \cdots \vee \neg\phi_n\}$. □

Example. *The theory of fields is finitely axiomatizable, as is the theory of fields of characteristic $p > 0$, for some specific p – we just add the equation $p = 0$ to the usual field axioms. The theory of fields of characteristic 0 is also axiomatizable; we just add the infinite sequence of axioms $\{p \neq 0\}$ to the usual field axioms. However, unlike the theory of fields of characteristic p, the theory of fields of characteristic zero is* not *finitely axiomatizable. Otherwise, there would be a finite subset of the axioms added to the field axioms which characterizes the theory of fields of characteristic zero. However, we clearly see that any finite subset of the axioms we've given for the theory of fields of characteristic zero also has models which have finite characteristic.*

Example. *The theory of infinite sets can be axiomatized over the language L which contains no constants, and only the equality predicate, with the theory $\Gamma =$*

$\{\phi_n : n \geqslant 1\}$*, where* ϕ_n *is the statement expressing that the set must have at least n elements, i.e.* ϕ_n *is the statement* $(\exists x_1, \ldots, x_n : x_i \neq x_j$ *for* $1 \leqslant i < j \leqslant n)$*. This theory is* not finitely axiomatizable*, because any finite subtheory of* Γ *has a finite model, namely, if the finite subtheory only contains* ϕ_n *for* $n \leqslant n_0$*, then* $\{1, \ldots, n_0\}$ *is a model for the theory. It follows that the* theory of finite sets *does not have an axiomatization in first order logic – there are first order theories which are only modelled by finite sets, but there is no first order theory with the additional property that every finite set is a model.*

3.7 Skolem-Löwenheim and Systems With Equality

We have seen that Lindenbaum's technique allows us to construct models from any consistent theory. We will now discuss the Skolem-Löwenheim Theorem, which shows that we can construct *arbitrarily large models*, and also *fairly small models*, for any consistent theory with an infinite model. Our first proof relies on a cheat, which we will fix by introducing *first order systems with equality*, and *normal models*.

Theorem 3.27 (Naive Skolem-Löwenheim Theorem)**.** *Any consistent theory has a model whose cardinality is the same as the theory, or is denumerable if the theory is finite. More generally, if* κ *is a cardinal greater than or equal to the cardinality of a consistent theory* Γ*, then there is a model of* Γ *with cardinality* κ*.*

Proof. For the first result, it suffices to consider the model obtained in Lindenbaum's Lemma. For the second result, let M be a model of Γ. Fix $x \in U^M$. Extend U^M to a set U^N of cardinality ω. Each new element will behave exactly like x. We define $f^N(t_1, \ldots, t_n) = f^M(t'_1, \ldots, t'_n)$, where $t'_k = t_k$ if $t_1 \in U^M$, else $t'_k = x$. Similarily, let $(t_1, \ldots, t_n) \in P^N$ if and only if $(t'_1, \ldots, t'_n) \in P^M$. Define constants the same as constants are defined in U^M. Then N is a model of Γ, with cardinality ω. □

The reason this proof seems like a cheat is because first order logic is not powerful enough to make terms distinct; we can hide various model-theoretic 'elements' of a theory inside a single syntactic element of the theory. It might be hoped that if we introduce an idea of equality, and force models to interpret equality as actual equality, then we might prevent this cardinality problem from occuring. But we will see that this is *not* the case.

We now specialize our study to the study of *first order theories with equality*. First order logic with equality always has a binary predicate $=$, possesses

all the axiom schemata of vanilla first order logic, but in addition, possesses two additional axiom schemata, namely, the additional axiom $x = x$, which whose application we will denote by (A6), and axioms $(x = y) \Rightarrow (\phi(t) \Rightarrow \phi(s))$, where x and y are variables, and t and s are terms, where s is obtained from t by swapping some numbers of occurrences of x with occurrences of y. Application of this result will be denoted by (A7).

Example. *In any theory with equality, and any term t, we have*

$$\vdash t = t.$$

This follows from $(A6)$*, by applying a substitution. Similarly, we have*

$$\vdash (t = s) \Rightarrow (s = t),$$

by applying (A7), whose formal proof we denote below:

$(\mathsf{t} = \mathsf{s} \Rightarrow \mathsf{s} = \mathsf{t})$	
1. $(\mathsf{x} = \mathsf{y})$	
2. $(\mathsf{x} = \mathsf{y}) \Rightarrow ((\mathsf{x} = \mathsf{x}) \Rightarrow (\mathsf{y} = \mathsf{x}))$	*(A7)*
3. $(\mathsf{x} = \mathsf{x}) \Rightarrow (\mathsf{y} = \mathsf{x})$	*(1),(2),(MP)*
4. $(\mathsf{x} = \mathsf{x})$	*(A6)*
5. $(\mathsf{y} = \mathsf{x})$	*(3),(4),(MP)*
6. $(\mathsf{x} = \mathsf{y}) \Rightarrow (\mathsf{y} = \mathsf{x})$	*(1-6),(DT)*
7. $(\forall \mathsf{x}, \mathsf{y} : (\mathsf{x} = \mathsf{y}) \Rightarrow (\mathsf{y} = \mathsf{x}))$	*(UG)*
8. $(\mathsf{t} = \mathsf{s} \Rightarrow \mathsf{s} = \mathsf{t})$	*(7),(A4),(MP)*

We shall also need the theorem $\vdash (t = s) \Rightarrow (s = r \Rightarrow t = r)$*, which is an instance of (A7) obtained after some universal generalization and specification.*

We have already seen the theory of groups as a first order theory with equality. Here is another example.

Example. *The theory of fields is built on a first order theory with constants* 0 *and* 1*, and functions* $+$ *and* $\cdot$*. The new axioms (in addition to the axioms of equality) are*

$$x + (y + z) = (x + y) + z \qquad x + 0 = x \qquad (\forall x, \exists y : x + y = 0)$$

$$x + y = y + x \qquad x \cdot (y \cdot z) = (x \cdot y) \cdot z \qquad x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$x \cdot y = y \cdot x \qquad x \cdot 1 = x \qquad x \neq 0 \Rightarrow (\exists y : x \cdot y = 1) \qquad 0 \neq 1$$

If we add another binary predicate symbol $<$, and add axioms

$$x < y \Rightarrow x + z < y + z \qquad x < y \wedge 0 < z \Rightarrow x \cdot z < y \cdot z$$

Then we obtain the theory of ordered fields.

Example. *One of the most important historical axiom systems was a system for geometry. The new predicates of the system are I, P, and L. We intuitively interpret $P(x)$ to mean that x is a point, we interpret $L(x)$ to mean x is a line, and we interpret $I(x, y)$ to mean that x lies on y (is incident to), provided that x is a point and y is a line, and otherwise, the predicate is false. The new axioms are*

$$P(x) \Rightarrow \neg L(x) \quad I(x, y) \Rightarrow P(x) \wedge L(y)$$

$$L(x) \Rightarrow (\exists y, z : y \neq z \wedge I(y, x) \wedge I(z, x))$$

$$[P(x) \wedge P(y) \wedge x \neq y] \Rightarrow (\exists z : L(z) \wedge I(x, z) \wedge I(y, z))$$

$$(\exists x, y, z : P(x) \wedge P(y) \wedge P(z) \wedge \neg C(x, y, z)),$$

where $C(x, y, z)$ is the collinear relation

$$(\exists u : L(u) \wedge L(x, u) \wedge L(y, u) \wedge L(z, u).$$

The parallel postulate, nor it's negation, is provable in this system, because the theory has Euclidean, projective, and hyperbolic geometries as models. Thus this theory is not complete.

In theories of equality it is possible to define new symbols which are useful for abreviating formulae. We define $(\exists! x : \phi(x))$ to means that there is only one element x satisfying ϕ. That is, the symbol is short for

$$(\exists x : \phi(x)) \wedge (\forall x, y : \phi(x) \wedge \phi(y) \Rightarrow x = y)$$

Thus any model possesses exactly one element with the property ϕ.

In any model M of a theory with equality, the relation $=^M$ is an equivalence relation. The model M is *normal* if the equivalence relation is trivial, i.e. no two distinct elements in the model are equal to one another. Any model M can be contracted into a normal model M' by identifying elements that are equal to one

another. Given a model M, we quotient U^M by $=^M$ to obtain $U^{M'}$. For a function f and predicate P, define

$$f^{M'}([t_1],[t_2],\ldots,[t_n]) = [f^M(t_1,\ldots,t_n)]$$

$$([t_1],[t_2],\ldots,[t_n]) \in P^{M'} \quad \text{iff} \quad (t_1,\ldots,t_n) \in P^{M'}$$

These definitions are well defined, since the model interprets equality correctly. All axioms are correctly interpreted by the model as well. Since every consistent theory with equality has a normal model, the completeness theorem holds even when restricted to theories with equality, and only to normal models. In the sequel, *all theories* we discuss will be assumed to have equality, and *all models* will be assumed normal. We can now discuss the downward and upward Skolem-Löwenheim theorems.

Proposition 3.28 (The Downward Skolem-Löwenheim Theorem)**.** *Any consistent theory of equality has a normal model whose cardinality is less than or equal to the cardinality of the theory in question.*

Proof. Take any (not necessarily normal) model of the theory, whose cardinality is equal to the cardinality of the theory, and then contract the model to obtain a normal model. □

This theorem appears to contradict various standard results in mathematics, for instance, that the field of real numbers is the only complete, ordered field up to isomorphism. *However*, we must remember that first order theories are only a model of mathematics, and thus do not sufficiently encapsulate all mathematical techniques. We therefore learn from the Skolem-Löwenheim theorem that it is impossible to *axiomatize* the theory of complete ordered fields in first order logic.

We can obtain a more powerful version of the downward Skolem-Löwenheim Theorem which allows one to identify substructures of models of large cardinality that are elementarily equivalent to the overall model. To begin, we must discuss the Tarski-Vaught criterion.

Theorem 3.29. *Let M and N be structures of a first order language L, where $N \subset M$. Suppose that for any statement $\phi(x_1,\ldots,x_n,y)$ in L, and any $a_1,\ldots,a_n \in M$, $M \models (\exists y : \phi(a,y))$ implies that $N \models (\exists y : \phi(a,y))$. Then the injective map $N \to M$ gives an elementary equivalence (we call N an* elementary substructure *of M).*

Proof. We must prove that for any sentence $\phi(x_1, \ldots, x_n)$ in L, and any $a_1, \ldots, a_n \in U_N$, $M \vDash \phi(a)$ if and only if $N \vDash \phi(a)$. We will prove this by induction on the complexity of ϕ. It is true for atomic predicates. If it is true for ψ and η, it is also clearly true for $\neg\psi$ and $\psi \circ \eta$, for any propositional connective $\circ$. Finally, it suffices to show that if the claim is true for ψ, it is also true for $(\exists x : \psi)$. If $N \vDash (\exists x : \psi)$, then there exists $a \in U_N$ such that $N \vDash \psi[a/x]$. By the induction hypothesis, $M \vDash \psi[a/x]$, and thus $M \vDash (\exists x : \psi)$. Conversely, if $M \vDash (\exists x : \psi)$, then the assumption of the theorem entails that $N \vDash (\exists x : \psi)$. Thus the result is proved. □

Theorem 3.30 (The Downward Skolem-Löwenheim Theorem V2)**.** *If M is a structure over a first order language L, then for any $V \subset M$, there exists an elementary substructure N of M containing V, and such that $\#(N) \leqslant \max(\#(V), \#(L))$.*

Proof. We define N inductively. We first define $V_0 = V$. Given that V_n is defined for some integer n, let Γ_n be the set of all sentences in the language $L(V_n)$ of the form $(\exists x : \phi)$, such that $M \vDash (\exists x : \phi)$. For each $(\exists x : \phi) \in \Gamma_n$, we can find $a_\phi \in M$ such that $M \vDash \phi[a_\phi/x]$, and form a set A_n out of the set of all such elements. Then let X_{n+1} be $X_n \cup A_n$. Then define N as the substructure of M with the underlying universe $\bigcup_n V_n$. It is clear that if $\#(X_n) \leqslant \max(\#(V), \#(L))$, the same is true for $\#(X_{n+1})$, and thus, since $\#(L)$ is infinite, that $\#(N) \leqslant \max(\#(V), \#(L))$. Moreover, it is simple to see that Vaughn's criterion is satisfied, so that N is an elementary substructure of M. □

Now we can state the true upward Löwenheim-Skolem theorem, since the other proof relied on a big cheat, duplicating elements that are syntactically equal, which we cannot rely on in the theory of normal models, we must do some extra work.

Corollary 3.31 (Upward Löwenheim-Skolem)**.** *Any theory of equality which has a normal model containing infinitely many elements, must also have models of any greater cardinality.*

Proof. Let Γ be a consistent theory with an infinite normal model M. Given any cardinal $\omega \geqslant \#(M)$, add to Γ new constants c_α for $\alpha \in \omega$, with axioms $c_\alpha \neq c_\beta$ if $\alpha \neq \beta$, forming a new theory Γ'. We claim Γ' is consistent. Suppose that $\Gamma' \vdash \phi \wedge \neg\phi$ for some statement ϕ. The proof of this statement would only use finitely many axioms $c_{\alpha_1} \neq c_{\beta_1}, \ldots, c_{\alpha_m} \neq c_{\beta_m}$. Since M is infinite, we may consider a model M', with the same universe of discourse, but interpreting c_{α_i} and c_{β_i} in such a way that $c_{\alpha_i} \neq c_{\beta_i}$ is satisfied in M. But then M' is a model for

$\Gamma \cup \{c_{\alpha_1} \neq c_{\beta_1}, \ldots, c_{\alpha_m} \neq c_{\beta_m}\}$, which therefore must be consistent. This gives a contradiction. Thus Γ' is consistent, and, by the completeness theorem, must have a normal model M' with $\#(M') \leqslant \#(\Gamma') = \omega$. But also the equality axioms of Γ' imply that $\#(M') \geqslant \omega$, so that we have equality here. □

We *must* have applied the axiom of choice in proving the Löwenheim-Skolem theorems in some manner, because the theorems are actually equivalent. In fact, this shows that a number of results about first order logic are equivalent to the axiom of choice, because our argument proved the following sequence of claims:

$$\text{AC} \Rightarrow \text{Completeness} \Rightarrow \text{Compactness} \Rightarrow \text{Löwenheim-Skolem} \Rightarrow \text{AC}.$$

To see how Löwenheim-Skolem implies the axiom of choice, consider the theory Γ containing the sentence

$$(\forall x, y, x', y' : (f(x, y) = f(x', y')) \Rightarrow (x = x' \wedge y = y'))$$

and the sentence

$$(\forall z : \exists x, y : f(x, y) = z).$$

A model of this theory is precisely a set U, and a bijection $f : U \times U \to U$. The theory has infinite models: simply consider the natural numbers $\mathbb{N}$, and the bijection $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ given by the standard construction (note this *doesn't* use the axiom of choice, but instead just an inductive definition). But if Γ has a countably infinite model, it *must*, by the upward Löwenheim-Skolem theorem, have models of any infinite cardinality. But now we see how we can prove the axiom of choice. Given any infinite set X, the theorem implies that there must exist a bijection $f : X \times X \to X$, and this statement is equivalent to the axiom of choice. Thus there is *no constructive way* to prove the completeness of first order logic, whereas we have proved the completeness of propositional logic constructively. TODO: Mendelson seems to believe that completeness can be reduced to a weaker form of the axiom choice - I should check to see which one is right.

Here's a cute application of the compactness theorem, which relates back to the fact that the theory of finite sets is *not* axiomatizable in first order logic.

Theorem 3.32. *If a theory Γ has normal models of arbitrarily large cardinality, then it has an infinite normal model.*

Proof. For each n, consider the theory

$$\Gamma_{n_0} = \Gamma \cup \{\phi_n : 1 \leqslant n \leqslant n_0\},$$

where ϕ_n is the statement that there exists at least n distinct elements, as considered in previous examples. Then by assumption, Γ_{n_0} is consistent for all n_0. But this means that $\Gamma_\infty = \bigcup \Gamma_{n_0}$ is a consistent theory by the compactness theorem. And a normal model of Γ_∞ must have infinite cardinality. □

Remark. Similarily, the upward Skolem-Löwenheim theorem implies that we cannot have an axiomatization of a theory of countably infinite sets in first order logic.

Skolem's paradox, a consequence of the downward Skolem-Löwenheim theorem, is that ZFC, expressible as a first order theory, has a *countable model*. This seems to contradict Cantor's theorem, i.e. that the set of subsets of the natural numbers is uncountable. Resolving this paradox requires us to be careful with how ZFC works: to say that the set of subsets of the natural numbers $\mathbb{N}$ is uncountable is to say that there *does not* exist a surjective function f from $\mathbb{N}$ to $2^{\mathbb{N}}$, and under the ZFC interpretation of a function f as a set of ordered pairs, that there does not exist a set f of ordered pairs that corresponds to a surjective function. We might write this as the following statement:

$$\phi = \neg(\exists f : F(f) \wedge S(f))$$

where F abbreviates the predicate 'f is a function from $\mathbb{N}$ to $2^{\mathbb{N}}$', and S abbreviates the predicate 'f is onto $2^{\mathbb{N}}$', i.e.

$$F(f) = (\forall x \in \mathbb{N} : \exists y \in 2^{\mathbb{N}} : (x, y) \in f) \wedge (\forall z \in f : \exists x \in \mathbb{N}, y \in 2^{\mathbb{N}} : z = (x, y)),$$

and

$$S(f) = (\forall y \in 2^{\mathbb{N}} : \exists x \in \mathbb{N} : (x, y) \in f).$$

(see the discussion in the next section as to why introducing the constants $\mathbb{N}$ and $2^{\mathbb{N}}$ into our language does not change the situation). It is entirely possible for this statement to be true in a countable interpretation M of the language of ZFC: one need only interpret $\mathbb{N}$ and $2^{\mathbb{N}}$ as elements of M, while not having any element of M that might be interpreted as a surjective function from $\mathbb{N}$ to $2^{\mathbb{N}}$.

3.8 Redundancy of Constants and Functions

Another application of theories of equality is to enable us to introduce new constants and formulas to a logic, and add new axioms to the logic related to these constants and formulas, without increasing the number of theorems that can be proved about the original logic.

Theorem 3.33. *Let Γ be a theory with equality over a language L, consider a tuple of variables $y = (y_1, \ldots, y_n)$, and suppose that we can prove that*

$$\Gamma \vdash (\exists! x : \phi(x, y)).$$

Consider a new theory Γ_f, over a language $L(f)$ obtained by augmenting L with a new n-ary function f, and such that Γ_f has the additional family of axioms

$$\phi(f(t), t),$$

for any terms $t = (t_1, \ldots, t_n)$ with the property that the string $f(t)$ is free for substitution for x in $\phi(x,t)$. Then there is a constructive method, which takes any formula η in the language $L(f)$, and finds a formula ν in the language L, such that

$$\Gamma_f \vdash \eta \Leftrightarrow \nu$$

and there is a constructive procedure for taking a proof that $\Gamma \vdash \nu$, and converting it into a proof that $\Gamma_f \vdash \phi$.

Proof. Define a *simple* f term to be a term of the form $f(t)$, where f does not occur in any of the terms t. Consider the procedure which takes a formula η, finds the leftmost occurence of a simple f term in the formula, and replaces it with a variable x not yet occuring in ϕ, obtaining a formula η_0. Consider the formula $\psi = (\exists x : \phi(x,t) \wedge \eta_0(x))$. Then we can prove $\Gamma_f \vdash \eta \Leftrightarrow \psi$, because $\eta = \eta_0(f(t))$, and we have a proof

$\eta_0(\mathsf{f}(\mathsf{t})) \Leftrightarrow (\exists \mathsf{x} : \phi(\mathsf{x},\mathsf{t}) \wedge \eta_0(\mathsf{x}))$

1. $\eta_0(\mathsf{f}(\mathsf{t}))$
2. $\phi(\mathsf{f}(\mathsf{t}),\mathsf{t}) \wedge \eta_0(\mathsf{f}(\mathsf{t}))$
3. $(\exists \mathsf{x} : \phi(\mathsf{x},\mathsf{t}) \wedge \eta_0(\mathsf{x}))$

4. $\eta_0(\mathsf{f}(\mathsf{t})) \Rightarrow (\exists \mathsf{x} : \phi(\mathsf{x},\mathsf{t}) \wedge \eta_0(\mathsf{x}))$
5. $(\exists \mathsf{x} : \phi(\mathsf{x},\mathsf{t}) \wedge \eta_0(\mathsf{x}))$
   6. $\phi(\mathsf{c},\mathsf{t}) \wedge \eta_0(\mathsf{c})$
   7. $(\exists! \mathsf{x} : \phi(\mathsf{x},\mathsf{t}))$
   8. $\mathsf{c} = \mathsf{f}(\mathsf{t})$
   9. $\eta_0(\mathsf{f}(\mathsf{t}))$
10. $\eta_0(\mathsf{f}(\mathsf{t}))$

11. $(\exists \mathsf{x} : \phi(\mathsf{x},\mathsf{t}) \wedge \eta_0(\mathsf{x})) \Rightarrow \eta_0(\mathsf{f}(\mathsf{t}))$

Continuing this process $\eta_1, \eta_2, \ldots, \eta_n$, each successively containing one less instance of the formula f, until we end up with a formula ψ which contains no instances of f. By transitivity, we know that $\Gamma_f \vdash \phi \Leftrightarrow \psi$. It will suffice to prove the theorem assuming that η contains no instances of f to being with, because if $\Gamma_f \vdash \eta_n$ implies $\Gamma \vdash \eta_n$, we may work backwards to conclude that if $\Gamma_f \vdash \eta$, then $\Gamma_f \vdash \eta_n$, and therefore that $\Gamma \vdash \eta_n$. We also assume η is a closed formula. Let M be any normal model of Γ. Then for any $a_1, \ldots, a_n \in M$, there is a unique x such that $M \vDash \phi_M(x, a)$. If we define $f : M^n \to M$ by $f(a) = x$, then we may extend M to an interpretation of Γ_f. If $\Gamma_f \vdash \eta$, this implies that $M \vDash \eta$, and therefore, since M has the same domain as M, that $M \vDash \eta$. Since every normal model of Γ thereby satisfies η, we may apply the completeness theorem to conclude that $\Gamma \vdash \eta$. □

It is important to note that if ϕ has length n, then the formula obtained by removing occurences of f converts ϕ to a formula of length $O(n)$, hence the process is not only computable, but efficiently computable.

Example. *In the theory of groups, we can prove that* $(\exists! y : xy = e)$. *We can therefore construct an equivalent theory by introducing a new function* f *which satisfies* $xf(x) = e$, *but we often denote* $f(x)$ *by* x^{-1}, *or in the theory of abelian groups, by* $-x$. *We can also augment the theory of groups by considering the equation* $(\exists! z : x = yz)$, *which leads us to introduce a function* g *such that the theorem* $x = yg(x, y)$ *holds; it is then possible to derive that* $g(x, y) = y^{-1}x$.

Example. *In the theory of fields, we do not have the theorem* $(\exists! y : xy = 1)$, *though we do have* $(\exists! y : x + y = 0)$, *so we can introduce the function which takes* x *to* $-x$. *First note that* $0x = 0$ *is a theorem of the theory of fields, because* $0x + 0x = (0 + 0)x = 0x$, *and also* $0x + 0 = 0x$, *hence* $0 = 0x - 0x = 0x$. *If* $(\exists! y : xy = 1)$ *was a theorem of the theory of fields, then we could find introduce a constant* c *such that* $0c = 1$, *then* $x = 1x = (0c)x = 0(cx) = 0$, *hence* $x = 0$, *and by substitution we find that* $1 = 0$. *Since the theory of fields is consistent (it has a model), we find that* $\neg(\exists y : 0y = 1)$. *Nonetheless, we do have the theorem* $(\exists! y : (x = 0 \wedge y = 0) \vee xy = 1)$, *hence we could define an inversion operation* x^{-1}, *such that* x^{-1} *really is the multiplicative inverse of* x *if* x *is nonzero, and* $x^{-1} = 0$ *if* $x = 0$. *The problem is that the functions of standard first order logic can be applied to arbitrary terms of the logic. To obtain a more natural view of 'partial functions', we would need to take a detour down a theory of* type classes.

A nice result of the last theorem is that in theories with equality, one can replace functions and constants by predicates, so the objects are essentially redun-

dant. Given any n-ary function f, we can introduce an n-ary predicate P, remove f from the theory, and add the axiom $(\exists! y : P(y, x_1, \ldots, x_n))$, and replacing axioms involving f with axioms involving the new predicate P.

Example. *In group theory, we can remove $\cdot$ and e from the theory of groups by adding a trinary predicate P and a unary predicate Q, where we interpret $P(x, y, z)$ as "$z = xy$", and $Q(x)$ as "$x = 0$". We add the new axioms $(\exists! z : P(x, y, z))$ and $(\exists! x : Q(x))$, and convert the old axioms such that*

- *$x(yz) = (xy)z$ becomes*

$$P(x, y, a) \wedge P(y, z, b) \wedge P(a, z, c) \wedge P(x, b, d) \Rightarrow c = d$$

- *$ex = x$ becomes*

$$Q(x) \wedge P(x, y, z) \Rightarrow y = z$$

  Similarily, $xe = x$ becomes

$$Q(x) \wedge P(y, x, z) \Rightarrow y = z$$

- *The associative law $x(yz) = (xy)z$ becomes*

$$P(x, y, a) \wedge P(a, z, b) \wedge P(y, z, c) \wedge P(x, c, d) \Rightarrow b = c$$

- *The inversion law $(\exists y : xy = e)$ becomes*

$$(\exists y : Q(a) \wedge P(x, y, z) \Rightarrow a = z)$$

Thus the theory involves no functions or constants to specify.

This reduction is very important in the theory of logic programming languages, which attempts to reduce the computation of functions by converting this calculation to manipulation of predicates in a suitable formal language, so that computation becomes expressed as a series of inference rules, rather than as a description of the functions in the computation.

3.9 Prenex and Skolem Normal Form

A formula ϕ is in *prenex normal form* if it can be written as

$$(Q_1 x_1, \ldots, Q_n x_n : \psi),$$

where each Q_i is either $\forall$ or $\exists$, and ψ contains no quantifiers. We shall use our theory of equality to construct a prenex normal form equivalent to any given formula ϕ. More precisely, we will prove that there is an algorithm for transforming any formula ϕ into a formula ψ in prenex normal form, such that $\vdash (\phi \Leftrightarrow \psi)$, and moreover, there is an algorithm to compute the formal proof of this statement. If we are working in a *pure predicate calculus*, i.e. a first order theory with no constants or functions, and infinitely many predicate symbols, then we can go even further, and construct a *Skolem normal form* for any formula ϕ, i.e. a formula ψ which is a prenex normal form, in which all occurences of $\exists$ occur before occurences of $\forall$. TODO (See Section 2.10 of Mendelson or Section 3.2 of Hedman).

3.10 Diagrams

In this section, we discuss the technique of using *diagrams* of models to understand their structure. Given a structure M defined over a first order language L, the *diagram* $\Delta(M)$ is the theory over the language $L(M)$ containing all literals that are satisfied by M. If N is a model of $\Delta(M)$, then there is a natural morphism $M \to N$; in other words, the diagram of a model is a set of sentences which ultimately amounts to saying 'M can be embedded in any model of this theory'. Similarily, the *elementary diagram* $\Delta_E(M)$ ofa model M is the larger theory containing all sentences of $L(M)$ satisfied by M. If N is a model of $\Delta_E(M)$, then there is a natural elementary embedding of M in N, so that the elementary diagram is a theory saying 'M can be elementarily embedded in any model of N'.

Let us consider some consequences of this technique. A subset of an interpretation of a first order language is called a *substructure* if it is also an interpretation, i.e. if it contains all constants, and is closed under applications of functions. From a categorical perspective, a substructure of an interpretation N is an injective morphism $f : M \to N$ such that for any atomic predicate P, $P_M(a)$ is true if and only if $P_N(f(a))$ is true. If Γ is a theory *only* containing *universal sentences*, i.e. sentences of the form $(\forall x_1, \ldots, x_n : \phi(x_1, \ldots, x_n))$, where ϕ is quantifier free, then a substructure of any model is a model. This condition is quite common for many

first order theories: for instance, the theory of equivalence relations, partial orders, linear orders, rings, fields, and so on. It turns out this is essentially a necessary and sufficient condition.

Theorem 3.34. *Let Γ be a theory, such that a substructure of any model is a model. Then there exists an equivalent axiomatization Γ_0 of Γ, such that Γ_0 only contains universal sentences.*

Proof. Let Γ_0 be the set of all universal sentences such that $\Gamma \vdash \Gamma_0$. Then any model of Γ is a model of Γ_0. It suffices to show the converse. Suppose M is a model of Γ_0. Let $\Delta(M)$ be the diagram of M. If $\Gamma \cup \Delta(M)$ has a model M', then M is isomorphic to a substructure of M', and thus by assumption, is a model of Γ. So let's assume $\Gamma \cup \Delta(M)$ does not have a model. If it didn't, by the compactness theorem, $\Gamma \cup \Delta(M)$ would have a finite, inconsistent subset, i.e. there is a finite set $\Sigma \subset \Delta(M)$ such that $\Gamma \cup \Sigma$ is inconsistent. Let's consider $a_1, \ldots, a_n \in M$ such that

$$\Sigma = \{\phi_1(a_1, \ldots, a_n), \ldots, \phi_n(a_1, \ldots, a_n)\},$$

for some statements $\phi_1, \ldots, \phi_n$, which are either atomic predicates, or the negations of atomic predicates. Now Γ is a subset of the original language, before we introduced the constants $\{a_1, \ldots, a_n\}$. This means that the theory $\Gamma \cup \{\exists x : \phi_1(x) \wedge \cdots \wedge \phi_n(X)\}$ is inconsistent. This means that

$$\Gamma \vdash (\forall x : \neg\phi_1(x) \vee \cdots \vee \neg\phi_n(x)).$$

The sentence $\psi = (\forall x : \neg\phi_1(x) \vee \cdots \vee \neg\phi_n(x))$ is a universal sentence provable by Γ, so $\psi \in \Gamma_0$. But this gives a contradiction, since M is a model of Γ_0. □

A similar argument shows that if a theory is conserved under *superstructures*, then there exists an equivalent axiomatization of the theory only containing *existential statements*.

TODO: Fix above in line with Hedman Section 4.5.1, and dicuss when a statement is Γ equivalent to a quantifier free equivalent, plus example of a theory conserved under supersructures and substructures, but without quantifier elimination. Then discussion Section 4.6, that discusses that statements preserved under unions of chains are equivalent to $\forall_2$ statements.

Let's also use the method of diagrams to prove the existence of certain 'fiber-product' type constructions in the category of models of a given theory satisfying suitable assumptions. Our first result is about joint embeddings of models of a complete theory.

Theorem 3.35. *Given any family $\{M_\alpha\}$ of a complete theory Γ over a language L, there exists a model M and a family of elementary embeddings $M_\alpha \to M$.*

Proof. It suffices to show that $\bigcup_\alpha \Delta_E(M_\alpha)$ is inconsistent. By compactness, we can assume without loss of generality that the index set is finite, and we will prove the result by induction on the cardinality of the index set. So suppose that $\Lambda_{\alpha_0} = \bigcup_{\alpha<\alpha_0} \Delta_E(M_\alpha)$ is consistent. If $\bigcup_{\alpha\leqslant\alpha_0} \Delta_E(M_\alpha)$ was inconsistent, the fact that $\Delta_E(M_\alpha)$ is closed under conjunction implies that there is $\phi(x,y)$ in L, and constants a such that $\phi(x,a) \in \Delta_E(M_\alpha)$, but $\Lambda_{\alpha_0} \vdash \neg\phi(x,a)$. Since Λ_{α_0} does not contain any formulas containing the constants a, $\Lambda_{\alpha_0} \vdash \neg\phi(x,y)$, i.e. $\Lambda_{\alpha_0} \vdash \neg(\exists y : \phi(x,y))$. But by completeness, $\Gamma \vdash \neg(\exists x : \phi(x,y))$, hence $M_{\alpha_0} \vDash \neg(\exists x : \phi(x,y))$, which contradicts that $M_{\alpha_0} \vDash \phi(x,a)$. □

More generally, we can take fiber products of elementary embeddings.

Theorem 3.36. *Given any family $\{M_\alpha\}$ of a complete theory Γ over a language L, and let S be a set, together with a family of maps $S \to M_\alpha$. Expand the language L to include the elements of S as constants, and suppose that the natural extensions of the models $\{M_\alpha\}$ to structures over $L(S)$ are all elementarily equivalent. Then there exists a model M, together with elementary equivalences $M_\alpha \to M$, which all commute with the maps $S \to M_\alpha$. The assumptions are satisfied, for instance, if S is itself a model of Γ, and the maps $S \to M_\alpha$ are elementary equivalences.*

Proof. Augment the language L to a language $L(S)$ by including including the elements of S as constants, and consider the deductive closure of Γ in $L(S)$. Since all the structures are elementarily equivalent, the theory $\bigcup \Delta_E(M_\alpha)$ is consistent, and thus has a model, as in the last proof, which gives rise to the required map. □

TODO: Amalgamation of Models of Complete Theories: Section 4.4 of Hedman.

TODO: Section 4.5-4.6 of Hedman.

3.11 Categoricity

First order theories do not have unique models, because we can obtain a new model M' from any model M by taking a different underlying set, considering a bijection onto this set, and then pushing the constants and formulas on one set to the other set. The notion of categoricity discusses the phenomenon of unique

models *given a fixed cardinality* and *modulo isomorphisms*, which is the strongest criterion we can have to having unique models.

For a cardinality $\mathfrak{a}$, a theory Γ is $\mathfrak{a}$ *categorical* if there exists a unique model of Γ of cardinality $\mathfrak{a}$, up to isomorphism. Categoricity is the closest phenomenon we can obtain in the semantic theory of first order logic, when $\mathfrak{a}$ is infinite, because the Löwenheim-Skolem theorem guarantees the existence of models of infinite cardinality, provided that there is at least one model with infinite cardinality. One can think of such a theory as *categorically defining* a given mathematical object, up to a given cardinality, giving a complete description of properties that uniquely define the object up to isomorphism.

Example. *Let Γ be the pure theory of equality with the additional axiom $(\exists x, y : x \neq y \wedge (\forall z : x = z \vee y = z))$. Then every normal model contains two elements, and every bijection between two normal models is a Γ-isomorphism, so the theory is 2-categorical. Similarily, we can find extensions of the pure theory of equality which are n-categorical for any finite n.*

Example. *The theory of dense linear orderings with neither a first nor a last element is $\aleph_0$ categorical, but not $\mathfrak{a}$ categorical for any cardinality $\mathfrak{a} > \aleph_0$. Let M and N be two countable dense ordered sets, and consider two enumerations*

$$M = \{x_1, x_2, \dots\} \quad N = \{y_1, y_2, \dots\}$$

We shall construct an order preserving invertible map between M and N, which will therefore be an isomorphism between the two models. Define $f(x_1) = f(y_1)$, and assuming that f has been defined for $x_1, x_2, \dots, x_n$, let $f(x_{n+1}) = y_i$, where i is the smallest index such that if $x_j < x_{n+1}$ for $j < n$, then $f(x_j) < y_i$, and if $x_j > x_{n+1}$, then $f(x_j) > y_i$. It is always impossible to find such an index i, since the sets are densely ordered. After finding x_{n+1}, we pick the smallest index i such that y_i is not in the range of f, find the smallest index j such that $f(x_j)$ is not yet defined, and such that f remains order preseriving if $f(x_j) = y_i$. By induction, we find that f can be defined for all elements of M, that f is injective, and that f is also surjective.

Example. *The theory of vector spaces over the rational numbers can be defined as a simple extension of the theory of abelian groups such that if we define $ny = y + y + \cdots + y$, then we add the axioms $(\exists! y : ny = x)$ for each positive integer $n \geqslant 2$. The models of this theory are exactly the vector spaces over the rational numbers, and the morphisms are the vector space morphisms. We know from basic*

linear algebra that these models are classified by the dimension of the underlying vector space, so that the theory is not $\aleph_0$ *categorical, but the dimension of a vector space is always the cardinality of the vector space for uncountable cardinalities, so the theory is* $\mathfrak{a}$ *categorical for all uncountable cardinalities* $\mathfrak{a}$.

Example. *If we add the axiom* $x + x = 0$ *to the theory of abelian groups, we instead obtain the theory of vector spaces over* $\mathbf{F}_2$*, and we find the theory if* $\mathfrak{a}$ *categorical for all cardinalities upon which an* $\mathbf{F}_2$ *vector space can be defined (any infinite cardinality, and any power of 2).*

The examples show that the following properties are true of a theory:

- There is a theory which is $\aleph_0$ categorical, but not $\mathfrak{b}$ categorical for any other cardinality $\mathfrak{b} > \aleph_0$.
- There is a theory which is not $\aleph_0$ categorical, but is $\mathfrak{b}$ categorical for any cardinality $\mathfrak{b} > \aleph_0$.
- There is a theory which is $\mathfrak{a}$ categorical for any infinite cardinality $\mathfrak{a}$. Such theories are called *totally categorical*.
- There is a theory which is not $\mathfrak{a}$ categorical for any infinite cardinality $\mathfrak{a}$.

It is a result of Morley (1965) that these are the *only* four possibilities, but we will not prove that here (it involves some sophisticated techniques which gave rise to the field of *stability theory*). As to some other more sophisticated results in the area, there is a result of Zil'ber, wihch states that totally categorical theories are not finitely axiomatizable. On the other hand, a result of Hrushovski says that a totally categorical theory is always *quasi-finitely axiomatizable*, i.e. there exists a finite family of statements $\{\phi_1, \ldots, \phi_n\}$ containing free variables $x_1, \ldots, x_m$, such that the theory is axiomatized by

$$\left\{ \left(\exists x_1, \ldots, x_m : (\bigwedge_{i \neq j} x_i \neq x_j) \wedge (\bigwedge_i \phi_k(x_i) \right) : 1 \leqslant k \leqslant n, m \geqslant 1 \right\},$$

i.e. the axioms are that 'infinitely many elements of the universe satisfy the statements $\phi_1, \ldots, \phi_n$'.

We now discuss the *Łos-Vaught test* (1954), which can be used to show certain categorical theories are complete.

Theorem 3.37. *If a theory* Γ *has* $\mathfrak{a}$ *symbols, and is* $\mathfrak{b}$ *categorical for some infinite cardinality* $\mathfrak{b} \geqslant \mathfrak{a}$*, and has no finite models, then* Γ *is complete.*

Proof. If $\Gamma \nvdash \phi$ and $\Gamma \nvdash \neg\phi$, then $\Gamma \cup \{\phi\}$ and $\Gamma \cup \{\neg\phi\}$ are consistent, and therefore both have normal models of cardinality $\mathfrak{b}$ by the Löwenheim-Skolem theorems. Assuming without loss of generality that ϕ is a closed formula, it is clear that M and N cannot be Γ isomorphic, because $M \models \phi_k(x)phi$ and : $N \models \neg\phi$. □

In particular, any theory of equality, which has no finite models, but is categorical for any infinite cardinality, is a complete theory. The theory of fields of characteristic zero and the theory of planar geometry with infinitely many points are both not complete theories, and thus not categorical for any infinite cardinality. On the other hand, the Łos-Vaught test does show that the theory of dense linear orderings, the theory of algebraically closed fields of any fixed characteristic, and the theory of vector spaces over the rational numbers are all complete theories.

This rules out the categoricity of the theory of fields of characteristic zero and the theory of geometries with infinitely many points, for instance, since these theories are not complete. On the other hand, the Łos-Vaught test does apply to the theory of dense linear orderings, which is $\aleph_0$ categorical, which shows this theory is complete.

Example. *We use the theory of categoricity to prove a* 0-1 Law For Infinite Random Graphs*. Namely, we will prove that if* ϕ *be a statement in* L_{graphs}*, and if* G_n *is the random graph on n vertices obtained by including each edge with probability* $1/2$*, then either*

$$\lim_{n\to\infty} \mathbb{P}(G_n \models \phi) = 1 \quad or \quad \lim_{n\to\infty} \mathbb{P}(G_n \models \phi) = 0.$$

We consider the theory Γ_{RG}*, which is the theory of graphs, extended with the axioms*

$$\left\{ \left(\exists x_1, \ldots, x_n : \bigwedge_{i\neq j} (x_i \neq x_j) \right) : n \geqslant 1 \right\},$$

which ensure any model has infinitely many vertices, as well as the set $\{\rho_{kl} : k, l \geqslant 1\}$*, where* ρ_{kl} *is the statement*

$$\left(\forall x_1, \ldots, x_k, y_1, \ldots, y_l : \left(\bigwedge_{i,j} x_i \neq y_j \right) \Rightarrow \left(\exists z : \left(\bigwedge_i A(x_i, z) \right) \wedge \left(\bigwedge_j \neg(A(z, y_j) \vee z = y_j) \right) \right) \right).$$

The family of sentences says that for any two disjoint sets of vertices V and W in a graph of the same size, we can find a new vertex z, adjacent to all vertices in V

but not adjacent to any vertex in W. The theory Γ_{RG} is clearly consistent, since any finite subset of statements from the theory has a model. We claim the theory is actually $\aleph_0$ categorical. Indeed, the statements $\{\rho_{kl}\}$ allow us to construct an isomorphism f between any two countable models of the theory: given two such graphs, with vertices enumerated as $V = \{v_1, \dots\}$ and $W = \{w_1, \dots\}$. Given that we have defined the isomorphism f for a set S_C of V, mapping injectively onto a set S_W of W, we alternate between the following two methods:

- *Pick the smallest v_i not contained in S_V. Applying the statements $\{\rho_{kl}\}$, we can find w_j, not contained in S_W such that $v \in S_V$ is adjacent to v_i if and only if $f(v)$ is adjacent to w_j. Define $f(v_i) = w_j$.*
- *Pick the smallest w_j not contained S_W, and perform the analogous construction, i.e. using $\{\rho_{kl}\}$ to find v_i not contained in S_V such that w_j is adjacent to $f(v)$ if and only if v_i is adjacent to v. Define $f(v_i) = w_j$.*

Iterating this procedure constructs a bijection from V to W preserving adjacency, and thus we have constructed an isomorphism. It follows from this theory that there is a unique countable 'random graph', and by Vaughn's test, the theory is complete.

The axioms of graphs clearly hold in any finite random graph. And the axioms added hold asymptotically with probability one *in any sufficiently large random graph. This is easy to see with the first set of axioms ensuring the random graph has enough vertices. Verifying that $\mathbb{P}(G_n \vDash \rho_m) \to 1$ for each m is somewhat harder. For any disjoint vertex sets V_1 and V_2 with size m, the probability that any given vertex v works as a choice of z is 2^{-2m}. Thus some boolean operations and independence show that the probability that no such z exists is k^{n-2m}, where $k = 1 - 2^{-2m}$. A union bound tells us therefore that*

$$\mathbb{P}(G_n \vDash \rho_m) \leqslant \binom{n}{m}\binom{n-m}{m}k^{n-2m} = \frac{n!}{(m!)^2(n-2m)!}k^N \lesssim_m n^{2m}k^{n-2m}.$$

But this quantity converges to zero for a fixed m as $n \to \infty$. Thus

$$\lim_{n\to\infty} \mathbb{P}(G_n \vDash \rho_m) = 1.$$

If Γ_{Graphs} is the theory of graphs, and

$$\Gamma_{Graphs} \vdash (\phi \Rightarrow \psi),$$

then $\mathbb{P}(G_n \vDash \phi) \leqslant \mathbb{P}(G_n \vDash \psi)$. Thus, applying the completeness of Γ_{RG}, the $0-1$ law for infinite random graphs immediately follows.

3.12 Definability

Let L be a first order language, and let M be a structure over the language. For $m \geqslant 1$, a set $S \subset M^n$ is *definable* if there exists a formula $\phi(x) = \phi(x_1, \ldots, x_n)$ in L such that

$$S = \{a \in L^m : M \vDash \phi(a)\}.$$

In this circumstance, we will also say the n-ary relation R defined by S is definable. An element $a \in M$ is definable if $\{a\}$ is definable, a function $f : M^n \to M^m$ is definable if it's graph $\Gamma(f) = \{(x, y) \in M^{n+m} : y = f(x)\}$ is definable. The study of definability measures the complexity of sets that are discussable in a given language.

Example. *All natural numbers are definable in* $(\mathbb{N}, <)$. *To start with,* 0 *is definable, because it is the* smallest number*, that is* 0 *is the unique element* n *of the structure* $(\mathbb{N}, <)$ *which satisfies the formula* $\phi(n)$*, where*

$$\phi(x) = \neg(\exists y : y < x).$$

If n *is definable, then we claim* $n + 1$ *is definable. Indeed, we consider the formula*

$$\psi(x, y) = (x < y) \wedge \neg(\exists z : x < z < y),$$

which is true precisely when y *is the immediate successor of* x*. If* $\phi_n(x)$ *defines an integer* n*, then the formula*

$$\phi_{n+1}(y) = (\exists x : \phi_n(x) \wedge \psi(x, y))$$

defines $n + 1$.

If $f : M \to M$ is an automorphism of a model M, and S is a definable subset of M, then $f(S) = S$. Indeed, if $S = \{x \in M : M \vDash \phi(x)\}$, then the fact follows from the property that $M \vDash \phi(s)$ if and only if $M \vDash \phi(f(s))$. This fact can be used to show certain sets are *undefinable*, i.e. because they are not preserved by automorphisms.

Example. *The only definable subsets of* $(\mathbb{Z}, <)$ *are* $\varnothing$ *and* $\mathbb{Z}$*. Indeed, if we consider the order automorphism* $f_n : \mathbb{Z} \to \mathbb{Z}$ *given by* $f_n(x) = x + n$*, then we see that any definable set* S *must satisfy* $S + n = S$ *for all* $n \in \mathbb{Z}$*. But the only such sets that can satisfy this property are* $S = \varnothing$ *and* $S = \mathbb{Z}$*.*

On the other hand, all integers are definable in the structure $(\mathbb{Z}, <, n)$, for any fixed integer $n \in \mathbb{Z}$. This follows by similar techniques to the last example, i.e. because the immediate successor and immediate predecessor of any definable integer is definable. For similar reasons, every integer is definable in $(\mathbb{Z}, +)$ and $(\mathbb{Z}, \cdot)$, because 0 and 1 can be shown to be definable in these languages (they are the additive and multiplicative identities respectively).

The order relation $<$ is not definable in $(\mathbb{Z}, +)$, because $x \mapsto -x$ is an automorphism of $(\mathbb{Z}, +)$. It is also undefinable in $(\mathbb{Z}, \cdot)$, because we can obtain many automorphisms of this structure by 'permuting primes', and such automorphisms do not preserve the ordering of $\mathbb{Z}$. On the other hand, $<$ is definable in $(\mathbb{Z}, +, \cdot)$. We can obtain a defining statement by applying Lagrange's four squares theorem, which says every non-negative positive integer can be written as the sum of at most four integers. Thus the natural numbers are a definable subset of $\mathbb{Z}$ in this structure, defined by the statement

$$\phi(x) = (\exists y_1, y_2, y_3, y_4 : x = y_1^2 + y_2^2 + y_3^2 + y_4^2).$$

We can then define the order structure by the statement

$$\psi(x, y) = (x \neq y) \wedge \phi(y - x),$$

implicitly using the fact that substraction is definable in $(\mathbb{Z}, +)$.

Example. *Let $(k, +, \cdot)$ be a field of characteristic zero. Then every element of $\mathbb{Q}$ is definable in $(k, +, \cdot)$. Indeed, 0 and 1 are definable as the additive and multiplicative identities, and one can then use $+$ to show every integer is definable. Division is then defined as the partial function $\{(x, y, z) : (y \neq 0) \wedge x = z \cdot y\}$, and combining these definitions can then be used to produce definitions for all rational numbers.*

Example. *Despite our intuitions about multiplication, the operation is not definable in $(\mathbb{N}, +)$. To see this, we rely on a result of Ginsburg and Spanier (1966), which says that all definable subsets of $(\mathbb{N}, +)$ are* ultimately periodic*, i.e. for any such definable $S \subset \mathbb{N}$, there exists $k \in \mathbb{N}$ such that the symmetric difference $S \Delta\, k\mathbb{N}$ is finite. The set of perfect squares is definable would be definable in $(\mathbb{N}, +)$ if multiplication was definable, which gives a contradiction since such a set is not ultimately periodic.*

Let M be a structure over a language L. If we augment L by adding constants for each element of M, producing a new language L^*, then we increase the power

of the language L, that allows us to define more subsets of M. We say things are *parametrically definable* if they are definable over L^*. Unlike with definability, every element of M is parametrically definable, and in fact, by taking disjunctions, every finite subset of M is definable. By taking negations, every *cofinite* set is definable (a set whose complement is finite). We say a structure is *minimal* if these are the only parametrically definable subsets.

Remark. The automorphism technique above used to verify if a set was definable cannot be used to show a set is not parametrically definable, since there are no nontrivial automorphisms of a structure M over the augmented language L^*, since such an automorphism would have to fix every constant.

Linearly ordered sets with no minimal or maximal element can never be minimal. Indeed, if X is such a set, then all intervals are parametrically definable, and for each $x \in X$, $[x, \infty)$ will be neither finite nor cofinite. The next hope for such sets is that one cannot produce more complicated sets; we say a linearly order set X is *o-minimal* (*order minimal*) if the only parametrically definable sets are finite unions of intervals.

Example. *The structure $(\mathbb{Z}, +, <)$ is not o-minimal. Indeed, the set of even numbers is definable, and such a set cannot be written as a finite union of intervals. Neither is $(\mathbb{Z}, \cdot, <)$, because the set of perfect squares, a definable subset, is not a finite union of intervals.*

Example. *The structure $(\mathbb{R}, \sin, <)$, where $\sin : \mathbb{R} \to \mathbb{R}$ is the usual sinusoidal function, is not o-minimal, because then the set $\sin^{-1}(0) = 2\pi\mathbb{Z}$ is definable, but not a finite union of intervals. In fact, one can show that, perhaps suprisingly,* all Borel sets *are definable in this structure.*

We thus see that o-minimality prevents 'infinite oscillation', like which are present with sinusoidal functions over the real line, or the set of even integers in the set of all integers.

3.13 Decidability

A deductive theory Γ is *decidable* if there is an effective procedure that determines whether any formula is an element of Γ or not.

Theorem 3.38. *If Γ is a complete, consistent theory, with an effectively enumerable axiomatization, then Γ is decidable.*

Proof. If a theory is axiomatic, there is a way to effectively enumerate all theorems of the theory. We first find an effective enumeration of all axioms, and enumerate all variables of the theory $x_1, x_2, \ldots$. We will build the set of all theorems constructively, beginning with an empty set $S = \varnothing$. At the kth step,

1. Add the kth axiom to S.
2. Attempt Modus Ponens on all elements of S, and all applications of universal generalization on elements of S, using only the variables $x_1, \ldots, x_k$.

Eventually, we will add each axiom to the set, and we will eventually be able to apply any application of universal generalization, hence we will eventually add all theorems to the set. Given any formula ϕ, either $\Gamma \vdash \phi$ or $\Gamma \vdash \neg\phi$, but not both, and we can decide whether ϕ is provable by waiting for either ϕ or $\neg\phi$ to end up in the enumeration. Thus the algorithm is guaranteed to terminate given any input. □

The fact that the theory has an effectively enumerable axiom system is doing a *lot of legwork here*. Given any structure M, $\mathrm{Th}(M)$ is a complete, consistent theory, but this theory *does not* necessarily have an effectively enumerable axiomatization. The seminal *incompleteness theorem* of Gödel shows that $\mathrm{Th}(\mathbb{N})$ is *not* decidable. Therefore, it follows that any effectively enumerable listing of theorems of arithmetic is necessarily *incomplete*: it does not contain axioms necessary to prove all results which are true of the natural numbers.

Example. *Consider the theory of densely ordered sets with no first nor last element. The theory is $\aleph_0$ categorical, and thus complete, and so decidable. But let's consider a more efficient algorithm, since the last algorithm has no time guarantees. This algorithm will take a statement ϕ of length l, and produce a proof of ϕ, or it's negation, in time $O(2^{O(l)})$.*

Given any formula ϕ in the theory (which we may assume is closed), there is an effective procedure to convert ϕ into prenex normal form

$$(Q_1 x_1, \ldots, Q_n x_n : \psi(x)),$$

where ψ contains no quantifiers, i.e. in Poly(l) time. We may assume all the quantifiers are existential quantifiers, by replacing $(\forall x : \eta)$ with $\neg(\exists x : \neg\eta)$, since we are trying to find a procedure to prove either the statement above or it's negation. Thus ϕ is without loss of generality of the form

$$(\exists x_1, \ldots, x_n : \psi(x)),$$

where ψ contains no quantifiers, and only contains terms in $\{x_1, \ldots, x_n\}$. Our goal is to successively replace ϕ with formulas ϕ', such that $\vdash \phi \Rightarrow \phi'$, and moreover, *the proof this statement is* computable. *We will not carry out all the details, but one should check throughout the proof that each time we perform this replacement, the implicit implication statement is provable by an explicit proof.*

To begin with, we can put ψ into disjunctive normal form, replacing occurences of $t \neq s$ with $t < s \vee s < t$, and replacing occurrences of $\neg(t < s)$ with $t = s \vee t < s$. There are at most $O(2^l l!)$ different ways to arrange n variables in a linear ordering, so there can be at most $O(2^l l!)$ different disjunctive clauses in ψ, and each clause has length at most $O(l)$. Now we can write

$$\psi = \eta_1 \vee \cdots \vee \eta_m,$$

where each formula η_i is a conjunction of atomic predicates, each of the form $t < s$ or $t = s$ for some $t, s \in \{x_1, \ldots, x_n\}$. The statement

$$(\exists x_n : \eta_1 \vee \cdots \vee \eta_m)$$

can be replaced with

$$(\exists x_n : \eta_1) \vee \cdots \vee (\exists x_n : \eta_m).$$

We will now replace each of the statements in this disjunction with an equivalent formula only containing the variables $\{x_1, \ldots, x_{n-1}\}$. Let's focus on one such statement, and let's write it as $(\exists x_n : \eta)$. We can perform a replacement that factors out terms in η that do not contain x_n, so we may assume all terms in the conjunction defining η contain x_n. We consider four possibilities:

- *η is a single atomic predicate, of the form $x_n = x_n$: Replace $(\exists x_n : \eta)$ with any tautology, i.e. $x_{n-1} = x_{n-1}$.*
- *η contains a term of the form $x_n = t$ or $t = x_n$, where $t \neq x_n$: Replace $(\exists x_n : \eta)$ with $\eta[t/x_n]$.*
- *If η contains a term of the form $x_n < x_n$, replace $(\exists x_n : \eta)$ with any contradiction, i.e. $x_{n-1} < x_{n-1}$.*
- *If the three previous cases do not hold, we can rearrange the atomic predicates that form η so that η is equal to*

  $$(x_n < t_1) \wedge \cdots \wedge (x_n < t_{k_1}) \wedge (s_1 < x_n) \wedge \cdots \wedge (s_{k_2} < x_n).$$

  We then simply replace $(\exists x_n : \eta)$ with $\bigwedge_{i,j}(s_i < t_j)$.

We have thus replaced $(\exists x_n : \eta)$ with an equivalent statement in the language of densely ordered fields, that no longer contains x_n as a variable. Collecting together all these calculations, we can thus replaces $(\exists x_1, \ldots, x_n : \psi(x_1, \ldots, x_n))$ with $(\exists x_1, \ldots, x_{n-1} : \psi'(x_1, \ldots, x_{n-1}))$, where ψ' is in disjunctive normal form. Iterating this process $l-1$ times, we obtain an equivalent statement $(\exists x_1 : \psi(x_1))$. Now ψ is in disjunctive normal form, with each atomic predicate being of the form $x_1 < x_1$ or $x_1 = x_1$. Each of these terms is either a contradiction or a tautology, i.e. of the form $x_1 = x_1$ or $x_1 < x_1$, and we can just check the truth functional properties of the statement to find a proof or disproof of the original statement. Noting that if we remove duplicate disjunctive clauses at each stage of the algorithm, then we can always guarantee at each stage that ψ has length $O(2^{O(l)})$, and so each iteration of the algorithm takes this amount of time. Iterating $l-1$ times, and then computing the truth table at the end, which takes $O(2^l(n!2^l))$ time, we conclude the complete algorithm takes $O(2^{O(l)})$ time. A similar method, due to Tarski (1951), verifies the theory of real closed fields is decidable, as does a result of Szmielew (1955), which shows the theory of abelian groups is decidable.

3.14 Quantifier-Elimination

A theory Γ has *quantifier elimination* if for any formula $\phi(x)$, there is a quantifier-free formula $\psi(x)$ such that

$$\Gamma \vdash \phi(x) \Leftrightarrow \psi(x).$$

A structure M has quantifier elimination if it's theory $\mathrm{Th}(M)$ has quantifier elimination. The study of the *definable sets* of a model of a theory with quantifier elimination are greatly simplified by this fact: we need only need to take the Boolean algebra generated by the definable sets corresponding to atomic predicates.

3.15 Non-Standard Analysis

The model theory of first order logic can be used to construct a very interesting set of models, which allows one to construct rigorous theories of infinitisimals in analysis. Let $\mathbb{R}$ be the set of real numbers, and consider a language by taking all elements of $\mathbb{R}$ as constants, and, for each n, all subsets of $\mathbb{R}^n$ as n-ary relations. The set $\mathbb{R}$ is naturally a structure for this language, and we let Γ denote the complete theory of all sentences which hold in $\mathbb{R}$. The upper Löwenheim-Skolem theorems

tell us that $\mathbb{R}$ is *not* the only model of Γ. Indeed, there exists models of Γ with arbitrarily large cardinality. Such models must be ordered fields, i.e. because Γ contains statements such as

$$(\forall x, y : x + y = y + x) \quad \text{and} \quad (\forall x, y : (0 < x) \wedge (0 < y) \Rightarrow (0 < x + y))$$

and so on. A proper extension of the theory of Γ is called a *non-standard model of analysis*.

Since Γ contains the diagram of $\mathbb{R}$, given any non-standard model $\mathbb{R}^*$, we have a proper embedding $\mathbb{R} \to \mathbb{R}^*$ which allows us to naturally think of $\mathbb{R}$ as a subset of $\mathbb{R}^*$.

Lemma 3.39. *Any non-standard model of analysis is non-archimidean.*

Proof. Let $\mathbb{R}^*$ be a non-standard model, and fix $x \in \mathbb{R}^* - \mathbb{R}$. Let

$$S = \{a \in \mathbb{R} : x < a\}.$$

If S is empty, then x is larger than every real number, and so $\mathbb{R}^*$ is non-archimidean. Otherwise, S is non-empty. If $S = \mathbb{R}$, then x is smaller than every real number, and then $-x$ is larger than every real number, so $\mathbb{R}^*$ is non-archimidean. Otherwise, we can consider a_*, the infinum of S. For any real number $a > 0$, $a_* - a < x$ and $a_* + a > x$. If $x < a_*$, then it follows that $a_* - x$ is a positive number smaller than every other positive real number, and thus $(a_* - x)^{-1}$ is larger than every positive real number, so $\mathbb{R}^*$ is non-archimidean. If $x > a_*$, then $x - a_*$ is smaller than every other positive real number, and so $(a_* - x)^{-1}$ is larger than every positive real number. In each circumstance, we conclude that $\mathbb{R}^*$ is non-archimidean. □

Given any non-standard model $\mathbb{R}^*$, define

$$M_0 = \{x \in \mathbb{R}^* : (\exists a \in \mathbb{R} : |x| < a)\}$$

to be the set of all *finite elements* in R, and let

$$M_1 = \{x \in \mathbb{R}^* : (\forall x \in \mathbb{R} : |x| < a)\}$$

be the set of *infinitesimal elements*. Then M_1 is a prime ideal of M_0, and M_0/M_1 is isomorphic to $\mathbb{R}$. Given a subset S of R, we let

$$O(S) = \bigoplus_{a \in S} M_0 a.$$

Then $O(1)$ is the set of all finite elements. We say two numbers $x, y \in R$ are infinitely close if $x - y \in M_1$. Let us write $x \approx y$ for this proposition.

Any relation on $\mathbb{R}$ extends to a relation on R. In particular, the set of integers in $\mathbb{R}$ can be defined by a unary relation

$$\{x \in \mathbb{R} : Z(x)\},$$

for some unary predicate Z. Applying the same predicate to $\mathbb{R}^*$, we obtain a set of *nonstandard* integers

$$\mathbb{Z}^* = \{x \in \mathbb{R}^* : Z(x)\}$$

Such a set must contain infinite elements, because the proposition

$$(\forall x, \exists n : Z(n) \wedge x < n)$$

holds in $\mathbb{R}$, and thus must also hold in $\mathbb{R}^*$. If $x \in \mathbb{R}^*$ is infinitely large, then there must exist $n \in \mathbb{Z}^*$ such that $x < n$, and this n must also therefore be infinitely large. In fact, $\mathbb{Z}^*$ is a non-standard extension of the integers in the same way that $\mathbb{R}^*$ is a non-standard extension of the real numbers; let ϕ be any statement expressible in the language of the integers (all integers as constants, and all subsets as relations as above), such that $\mathbb{Z} \vDash \phi$. We might as well assume ϕ is in prenex normal form, i.e. we can write ϕ as $Q_1 x_1 \ldots Q_n x_n \psi(x_1, \ldots, x_n)$. Working for $k = n$ to $k = 1$, we perform the following construction:

- If $Q_k = \forall$, we replace

  $$Q_1 x_1 \ldots \forall_k x_k : \psi_k(x_1, \ldots, x_k)$$

  with

  $$Q_1 x_1 \ldots \forall_k x_k : (Z(x_k) \Rightarrow \psi_k(x_1, \ldots, x_k)).$$

- If $Q_k = \exists$, we replace

  $$Q_1 x_1 \ldots \exists_k x_k : \psi_k(x_1, \ldots, x_k)$$

  with

  $$Q_1 x_1 \ldots \exists_k x_k : (Z(x_k) \wedge \psi_k(x_1, \ldots, x_k)).$$

Iterating the algorithm, we obtain a statement $\tilde{\phi}$ such that $\mathbb{R} \vDash \tilde{\phi}$, which implies $\mathbb{R}^* \vDash \tilde{\phi}$, and, now working backwards, removing the predicate Z we've introduced, we conclude that $\mathbb{Z}^* \vDash \phi$.

Remark. Similarily, we can consider a non-standard model $\mathbb{N}^*$ of the natural number, and a non-standard model $\mathbb{Q}^*$ of the rational numbers. It can be shown that every transcendental number is infinitely close to an element of $\mathbb{Q}^*$.

3.16 Logic Programming

In its general form, deduction in first order logic is uncomputable. That is, there is no algorithm which returns a proof of any given provable formula in first order logic, or rejects the input if there is no proof. However, it is easy to see that there is an algorithm which, given some provable formula, constructs a proof of this formula – we can simply enumerate all proofs in a clever manner, and then just check, given each proof, whether the conclusion of the proof is the statement we needed to prove. This means that it is not impossible to find algorithms which can find proofs of statements, and reject a large majority of statements which have no proof. The incompleteness theorem just implies that this algorithm must run into an infinite loop on some inputs. In logic programming, we find efficient ways of deciding whether some formula is provable, while trying to avoid infinite loops in a large majority of cases.

A *definite clause* (also known as a *Horn clause* after logician Alfred Horn) is a formula of the form $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \Rightarrow \psi$, where $\phi_1, \ldots, \phi_n, \psi$ are atomic predicates (so all the quantifiers are global over the entire formula). ψ is known as the *head* of the formula, and $\phi_1 \wedge \cdots \wedge \phi_n$ the *body*. Since $\phi_1 \wedge \cdots \wedge \phi_n \Rightarrow \psi$ is logically equivalent to $\psi \vee \neg\phi_1 \vee \cdots \vee \neg\phi_n$, we find that if a variable x does not occur in ψ, then $(\forall x : \phi_1 \wedge \cdots \wedge \phi_n \Rightarrow \psi)$ is equivalent to $(\exists x : \phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow \psi$. When we express a definite clause as a disjunction of clauses, it is possible to consider clauses with an empty body, or an empty head, and we shall treat these as definite clauses. Thus, by $\Rightarrow \psi$ we mean the logical formula ψ, and by $\phi_1, \ldots, \phi_n \Rightarrow$ we mean $\neg\phi_1 \vee \ldots \neg\phi_n$. We shall call a definite clause with an empty body a *fact*, and a clause with an empty head a *definite goal*. We will find that this particular subfamily of first order logic is somewhat effectively computable.

The problem of logic programming is to determine if a formula ψ of the form $\neg(\phi_1 \wedge \cdots \wedge \phi_n)$, known as a *definite goal*, is the logical consequence of a finite set of definite clauses, known as a *definite program*. If we universally quantify this formula over the variables $x_1, \ldots, x_n$, and then carry through a negation, we find this formula is equivalent to the formula $\neg(\exists x_1, \ldots, x_n : \phi_1 \wedge \cdots \wedge \phi_n)$, so the problem is equivalent to determining if ϕ_i are simultaneously satisfiable by some $x_1, \ldots, x_n$. We could even consider this problem in a more complex, constructivist form, which is to find a particular set of closed terms $t_1, \ldots, t_m$ such that $\phi_1(t_1, \ldots, t_m), \ldots, \phi_n(t_1, \ldots, t_m)$ all hold simultaneously. Surprisingly, it turns out that for a theory consisting only of finitely many Horn clauses, the problem is solvable constructively.

Given any theory Γ, assumed to be defined over a language containing one or

more constant symbols, the *Herbrand universe* of the theory is the set of all closed terms in the theory (in this context, a closed term is also called a *ground term*). On any Herbrand universe, we have standard interpretations of the functions and constants of the theory, and thus of all the terms of the theory. The set of all atomic predicates formed from the ground terms (also known as a *ground formula*) is known as the *Herbrand base* of the theory. A particular interpretation of the predicates gives what is called a *Herbrand interpretation* of the theory, and a *Herbrand model* of a theory is a Herbrand interpretation that is actually a model. Herbrand universes are particularly useful to the semantics of definite programs. First, they give an easy proof that all definite programs are consistent theories, because a model for any definite program can be given over the Herbrand universe by interpreting each atomic predicate P as true for all inputs.

Example. *Consider the definite program* Γ *formed from a language containing the constant* 0*, a unary successor function* S*, and the predicate Odd, where*

$$\Gamma = \{Odd(S(0)), Odd(x) \Rightarrow Odd(S(S(x)))\}.$$

The Herbrand universe H *of this program is precisely*

$$\{0, S(0), S(S(0)), \dots\},$$

which we can identify with the set $\mathbb{N}$ *of natural numbers. A Herbrand interpretation is just a specification of the behaviour of the predicate Odd over* H*. The Herbrand models can be identified with the set of all subsets of* $\mathbb{N}$ *that contain all odd natural numbers. In particular, if we let all natural numbers satisfy the predicate Odd, we get a specific case of the model considered at the end of the last paragraph.*

Theorem 3.40. *If* $\phi_1, \dots, \phi_n$ *is a definite program, and* ψ *is a goal, then for any model* M *of* $\{\phi_1, \dots, \phi_n, \psi\}$*, if we consider the interpretation* P_H *on the Herbrand universe* H *for each predicate* P *given by setting* $P_H(t)$ *to be true if* $M \models P(t)$*, then we obtain a Herbrand model of* $\{\phi_1, \dots, \phi_n, \psi\}$.

Proof. If ϕ_i is $\eta_1(x_1, \dots, x_k) \wedge \cdots \wedge \eta_m(x_1, \dots, x_k) \Rightarrow \nu(x_1, \dots, x_k)$, then $M \models \phi_i$, hence in particular if $t_1, \dots, t_k$ are closed terms of the theory, then we find $M \models \eta_1(t_1, \dots, t_k) \wedge \cdots \wedge \eta_m(t_1, \dots, t_k)$, hence in particular if H is the Herbrand universe with the interpretation given above, then $H \models \phi_i(t_1, \dots, t_k)$ for all closed terms $t_1, \dots, t_k$, hence $H \models \phi_i$. If ψ is the formula $\neg(\psi_1(x_1, \dots, x_k) \wedge \cdots \wedge \psi_n(x_1, \dots, x_k))$, then $M \models \psi$, so in particular $M \models \neg(\psi_1(t_1, \dots, t_k) \wedge \cdots \wedge \psi_n(t_1, \dots, t_k))$ for all closed terms $t_1, \dots, t_n$, hence we find that $H \models \neg(\psi_1 \wedge \dots \psi_n)$. □

Corollary 3.41. *If a goal ψ is satisfiable given $\phi_1, \ldots, \phi_n$ (which is equivalent to finding a model of the program and the goal), then there is a Herbrand model of $\phi_1, \ldots, \phi_n$ in which ψ is satisfiable.*

This is a fact which *does not* hold for general theories. That is, to verify a goal is satisfiable in a general theory, it is not sufficient to check all the Herbrand models.

Example. *Consider a first order language with a single constant c, no functions, and a single predicate P. Consider the theory*

$$\Gamma = \{\neg P(c), (\exists x : P(x))\}.$$

There are no Herbrand models of Γ, because the Herbrand universe of the language is precisely $\{c\}$. On the other hand, Γ has a model - we can simply consider $\{c, c'\}$, where $P(c)$ is false, and $P(c')$ is true.

An important property of Herbrand models of definite programs is that if $\{M_\alpha\}$ is a family of Herbrand models, then $\bigcap M_\alpha$ is also a Herbrand model. A Herbrand model exists, because we can always take the full relation on any given family of formulas. Thus by taking the intersection of all Herbrand models of a theory, we can consider a *least Herbrand model*, which contains all the necessary semantics of the definite program.

Theorem 3.42. *If M is the least Herbrand model of a given definite program Γ, then $\Gamma \vdash \phi$ holds if and only if $M \models \phi$.*

Proof. First, note that since M is a model of Γ, then $\Gamma \vdash \phi$ implies $M \models \phi$. Conversely, define a Herbrand interpretation H by letting $H \models P(t)$ if $\Gamma \vdash P(t)$. Then H is actually a Herbrand model of Γ, because for any formula $\phi \in \Gamma$, it is trivial to show $\Gamma \vdash \phi$. But this implies that since M it the *least* Herbrand model, if $M \models P(t)$ then $H \models P(t)$, hence $H \vdash P(t)$. □

Example. *Consider the formula $P(a) \vee Q(b)$. We have two Herbrand models H_1, H_2 of this formula, where $H_1 \models P(a)$ and $H_1 \models \neg Q(b)$, and $H_2 \models \neg P(a)$ but $H_2 \models Q(b)$. The intersection of these two models is not a Herbrand model of the formula, so the model property does not hold. Note that this formula cannot be specified as part of a definite program.*

Though the model intersection property does hold for Herbrand models of any logical theory, we cannot necessarily take the intersection of arbitrary models of

a theory because they may be defined over a particular set, and we have seen that we cannot reduce the study of the semantics of that logical theory to the semantics defined over a Herbrand universe. This is what makes the study of definite programs so special. In order to verify that $\phi_1 \vee \cdots \vee \phi_n$ is satisfiable in some model, we just need to check if $\neg(\phi_1 \vee \cdots \vee \phi_n)$ is *not* provable in the theory, which means that the minimal Herbrand model has closed terms $t_1, \ldots, t_m$ such that $\phi_1(t_1, \ldots, t_m) \vee \cdots \vee \phi_n(t_1, \ldots, t_m)$. Thus, given $\phi_1 \vee \cdots \vee \phi_n$ and a definite program Γ, the final piece of the puzzle is to give an algorithm to find closed terms in the minimal model of Γ which satisfy the formula.

Our algorithm to find these closed terms will be given recursively, which will correspond to a constructive method of finding the minimal Herbrand model. We cannot simply use provability to construct the model, because the whole point of our discussion was to verify a way to prove facts in definite programs, and we have no constructive way to prove particular statements. In turn, the inductive construction will give a recursive way to verify if an arbitrary definite goal is in the minimal model.

- If the definite program contains a fact $\psi(x_1, \ldots, x_k)$, then $\psi(t_1, \ldots, t_k)$ is in the minimal model for any closed terms $t_1, \ldots, t_k$.
- If the definite program contains a rule of the form

  $$\phi_1(x_1, \ldots, x_k), \ldots, \phi_k(x_1, \ldots, x_k) \Rightarrow \psi(x_1, \ldots, x_k),$$

  and $\phi_1(t_1, \ldots, t_k), \ldots, \phi_k(t_1, \ldots, t_k)$ are in the minimal model, then

  $$\psi(t_1, \ldots, t_k)$$

  is in the minimal model.

We claim that these two rules, applied inductively, suffice to construct a Herbrand interpretation which is the minimal Herbrand model. In fact, we see that this recursive algorithm can actually generate an infinite list containing *all* closed terms which satisfy a given goal $\psi_1, \ldots, \psi_n$. We just iterate through each rule of the definite program of the form $\phi_1 \wedge \cdots \wedge \phi_m \Rightarrow \eta$, where there is some terms $t_1, \ldots, t_m$ with $\eta(t_1, \ldots, t_m) = \psi_1(t_1, \ldots, t_m)$, and then we find all closed terms which satisfy the recursive goal

$$\phi_1(t_1, \ldots, t_m), \ldots, \phi_k(t_1, \ldots, t_m), \psi_2(t_1, \ldots, t_m), \psi_n(t_1, \ldots, t_m)$$

This is the process of *unification*, which we will algorithmically specify shortly. This is sufficient to find all the required closed terms. Unfortunately, there can be infinitely many terms which satisfy the formula, and this iteration process is not guaranteed to terminate, but in certain cases it gives a surefire way of constructing closed terms.

Example. *We should not expect the process to terminate in the general case (or even to determine if a particular computable subset of closed terms satisfies the set), because we can reduce the Post correspondence problem into constructing closed terms of a definite program, which we know to be uncomputable. Indeed, the problem is given an alphabet Λ, and finitely many words $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ over the alphabet, the correspondence problem is to determine whether there are indices $i_1, \ldots, i_m$ with $\alpha_{i_1} \ldots \alpha_{i_n} = \beta_{i_1} \ldots \beta_{i_n}$. If we write $\alpha_i = x^i_1 \ldots x^i_{i_N}$, and we consider the definite program*

$$\begin{aligned}
&\vdash x = x\\
&\vdash c(x, c(y,z)) = c(c(x,y),z)\\
&\vdash \alpha_i = c(x^i_1, c(x^i_2, \ldots, c(x_{i_N-1}, x_{i_N})))\\
&\vdash \alpha_i = c(s)\\
y = x &\vdash x = y\\
x = y, y = z &\vdash x = z\\
x_0 = x_1, y_0 = y_1 &\vdash c(x_0, x_1) = c(y_0, y_1)
\end{aligned}$$

with addition constants given by the alphabet Λ, and where we think of c as the composition of two words, then determining if there are closed terms t_0, t_1 over the universe consisting of the variables α_i and β_j and composition such that $t_0 = t_1$ is equivalent to the Post correspondence problem.

Example. *Consider the definite program*

$$\begin{aligned}
&\vdash odd(s(0))\\
odd(x) &\vdash odd(s(s(x)))
\end{aligned}$$

To find all terms satisfying $odd(y)$, we first reduce the term to the fact $odd(s(0))$, hence $s(0)$ satisfies the terms. Next, we find that letting $y = s(s(x))$ makes the goal unify with the head of the second axiom of the program, hence all other closed terms satisfying the term $odd(y)$ are of the form $odd(s(s(x))$, where $odd(x)$

is true. Continuing recursively, we find that $s(s(s(0)))$ is a closed term satisfying the clause, then $s(s(s(s(s(0)))))$, and so on and so forth. The algorithm never terminates, but successfully enumerates all the terms satisfying the clause. That is, the algorithm enumerates all the odd natural numbers.

We now formally define unification, and define an algorithmic way to unify terms. It will be helpful here to introduce some new notation. A substitution will be viewed as a map σ taking the variables in the logic and returning some term. Then σ can be uniquely extended to be defined on the set of all terms. Often we only need to understand the definition of some subset of some variables $X_1, \ldots, X_m$, in which case we write $\sigma = \{t_1/X_1, \ldots, t_n/X_n\}$ (this does not uniquely specify σ, but it specifies the map 'uniquely enough' for most purposes). Given two substitutions σ and τ, the composition $\sigma \circ \tau$ is just the composition of the two maps extended to terms. A substitution is idempotent if $\sigma^2 = \sigma$. If $t_1, \ldots, t_n$ and $u_1, \ldots u_n$ are two families of terms, then we say they unify if there is a substitution σ such that $\sigma(t_i) = \sigma(u_i)$ holds for each i, and we call σ a *unifier*. There is also an inductive definition which will correspond to a (non deterministic) algorithm to unify two lists of terms.

- For any two constants c and k, c unifies with k if and only if $c = k$.
- For any variable X and term t which does not contain X, X unifies with t. If t contains X, then t does not unify with X.
- If f and g are formulas, then $f(t_1, \ldots, t_n)$ and $g(u_1, \ldots, u_n)$ unify if and only if $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$ unify, and $f = g$.
- $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$ unify if and only if t_i unifies with u_i for each i, and the unification is the same for each i.

A unification σ between $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$ is more general than another unification τ if the corresponding substitution can unify if $\tau = \eta \circ \sigma$ for some other substitution η. This gives a partial ordering on the set of all unifications, and a maximum element is called the *most general unifier*. The idea is that if we try to unify $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$ by constructing a most general unifier between each t_i and u_i, then it is still possible to concatenate this unifier with the other unifiers to obtain a universal unifier, unless such a unification process is impossible.

- For any two formulas $f(t_1, \ldots, t_n)$ and $g(u_1, \ldots, u_n)$, we cannot unify if $f \neq g$, and if $f = g$ then it suffices to find a most general unifier for $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$.

- A most general unifier for the identity unification $X = X$ is the identity substitution.
- A most general unifier for $t = x$ or for $x = t$, where $t \neq x$ is impossible to construct if t contains an occurence of X, and otherwise it is obtained by swapping X with t.
- To unify $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$, obtain a most general unifier σ between t_1 and u_1, and τ between $t_2, \ldots, t_n$ and $u_2, \ldots, u_n$, and then consider the concatenation $\sigma \circ \tau$, which will be a most general unifier.

If we can construct a most general unifier TODO: See Section 3.3-3.6 of Hedman.

3.17 Limitations of First Order Logic

The weakness of expression in first order logic is both a boon and a curse. It prevents us from formalizing certain theories involving more infinitary characteristics. But it's simplicity is a boon, because it means we can find a sound, complete formal system for deriving theorems in first order logic, which have the model theoretic consequence of compactness, which gives us many useful consequences, like the Löwenheim-Skolem theorem. In this section, we discuss mathematical phenomena that *cannot* be encapsulated in a first order system.

Example. *Consider the theory of graphs, recalling that the language for the theory has no constants or formulae, and with a single binary predicate A denoting adjacency. For each n, we can form a sentence $\psi_n(x.y)$ in this language which can be interpreted as "The graph has a path of length n from x to y". But we* cannot *formalize a sentence which can be interpreted as "The graph has a path from x to y". Indeed, suppose it were possible to find such a sentence $\phi(x,y)$. Augment the language with two constants a and b, and consider the theory*

$$\Gamma = \{(\forall x, y : \phi(x,y)), \neg\psi_1(a,b), \psi_2(a,b), \ldots\}.$$

Then any finite subset of this theory is satisfiable (i.e. for each n, we can consider a connected graph with two vertices a and b which are not connected by a path of length $< n$). But Γ itself is not satisfiable. Thus we conclude by contradiction that ϕ cannot exist.

Clearly, for any formal system which can provide an axiomatiziation of connectedness, the analogue of the compactness theorem *must fail*. In particular, such a formal system cannot have a sound, complete deductive theory behaving analogously to the formal system for first order logic. Indeed, connectedness can be formalized using *infinitary logic* (in particular a form of infinitary logic often denoted $\mathcal{L}_{\omega_1,\omega}$), i.e. logic that allows for infinite conjunctions and disjunctions of first order formulas, or in the more powerful variant of *second order logic*, which is incredibly powerful system allowing for quantification over *relations*, i.e. *subsets of the universe of discourse*. The power of second order logic can make it's model theory often quite boring; given any model M of a theory of second order logic, then M is the *only* model of $\mathrm{Th}(M)$, up to isomorphism.

There is a theory of Lindström, which says that the theory of first order logic is the most powerful logical system defined on the language of first order logic, and possessing compactness and the downward Löwenheim-Skolem theorem. So in order to have a powerful theory of models, we are often stuck with first order logic.

Chapter 4

Combinatory Logic

Standard propositional logic formalizes the mathematical process of proof, pincering mathematical statements in a formal system where they can be examined in detail. Rather than looking at statements which can be verified true or false, Combinatory logic instead analyzes higher order functions, and our limitations in defining them. These functions are universally found in mathematics. The statement $(\forall x : \neg P(x) \vee Q(x))$ contains the functions $\neg$, $\vee$, P, Q, and $\forall$, and it is semantically interesting to interpret the statement not as a schema which can be applied to assert truths about objects of some domain of discourse, but instead as a single truth about the logical function obtained by composing the atomic logical operations. Combinatory logic studies the formal representation of these higher order functions, revealing the limitations in the functions we can define.

The primal nature of the substitution operator was discovered early on in the theory of combinatory logic. It is fundamental tool for constructing functions. For instance, if we represent numbers as tallies, so that 1 is $\cdot$, 2 is $\cdot\cdot$, and 7 is $\cdot\cdot\cdot\cdot\cdot\cdot\cdot$, then we can add 3 to any number by substituting it into the expression $\cdot\cdot\cdot x$ for the variable x. Similarily, we may multiply a number by two by substituting it into the equation xx for x. Addition and multiplication is thus a special case of substitution, and we shall find that a more complicated specification of this process will suffice to represent any function. Substitution is so important to combinatory logic that the field is often seen as the formal analysis of substitution.

Combinatorial logic was initially designed to provide a foundation for mathematics, where we can view inference rules as operators on formulas which can be formally analyzed. The operation of substitution plays a subtle role in much of this process, and this encouraged Combinatorial logic forerunner Moses Schönfinkel to introduce substitution as an explicit connective in a formal system. It turns out

that unrestricted substitution is an incredibly volatile operator, especially in the context of first order logic. While it provides expressive power, it also opens the floodgate to paradoxes which can easily lead to an inconsistent system. Even if these careless systems are refined, the systems are likely not expressive enough to handle all mathematical concepts. Nonetheless, the formal systems of combinatory logic are sufficiently expressive to describe all realistic forms of computation, and it is this application which makes the theory useful to modern computing science.

4.1 The λ Calculus

The λ-calculus is the most famous formalization used to study combinatory logic. Every formula in the calculus represents an operator over the other formulas in the calculus, and there are only two fundamental connectives we can use to manipulate these operators. The first is *function application*, which takes a function f and applies it to an argument a, denoted (fa). The second is *function abstraction*, which transforms a term t into a function $(\lambda x.t)$, which takes some argument a and substitutes it for x in t. We then use substitution rules on terms to express computation, and this is where the real fun of λ calculus begins.

Terms of the λ calculus can be applied arbitrarily to other terms, include the term itself. This distinguishes the view of functions in set theory, where it is impossible to consider a function application $f(f)$ without contradicting the axiom of regularity. Thus terms of the λ calculus model something very different to standard set theoretic functions. To distinguish this form of the calculus from other formal systems where the operators have particular domains, we call this the *untyped* version of the λ calculus. Nonetheless, unrestricted application is not a deficiency in the calculus – it has a reasonable interpretation in mathematics, which we will discuss later, and is required to construct the class of all computable functions.

So lets define the terms of λ calculus formally. First, we consider a symbol set consisting of variables, constants, the abstractor symbol λ, parenthesis, and a period as a separator. We define terms to be the smallest set of strings over these symbols such that

- All variables and constant are terms.
- If M and N are terms, then (MN) are terms.

- If M is a term, and x is a variable, then $(\lambda x.M)$ is a term.

If there are no constants, we call this a *pure* version of the λ calculus, because there are no constants, so every term of the calculus is a pure function. There is essentially no distinction between constants and variables in the λ calculus, except that we can perform λ abstraction on variables, so we won't consider constants in these notes.

We introduce some shorthand to ensure we don't get overwhelmed by parenthesis. First, we let terms associate to the left, so that $N_1 \dots N_n$ is shorthand for the term $(\dots((N_1 N_2)N_3)\dots)N_n)$, and we write $(\lambda x_1 \dots x_n.M)$ for the term $(\lambda x_1.(\lambda x_2.(\dots(\lambda x_n.M)\dots)))$. This shorthand is meant to let of think of a series of λ abstractions as representing a multidimensional function. This is the technique of Currying: we can think of a function which takes two arguments as a function which takes a single argument, and then returns a function which takes another argument to calculate the overall result!

On the set of terms of the λ-calculus, we introduce *reduction rules*, which not only simplify formulas, but act as the formal model of computation required for the theory.

- α *reduction* is a safe way to rename variables. If M is a term, and y is a variable not occuring anywhere in M, then we have a one-step α reduction $(\lambda x.M) \rhd_{\alpha,1} (\lambda y.M[y/x])$. More generally, if M contains a subterm of the form $(\lambda x.N)$, and if M' is the operation formed by replacing an occurence of $(\lambda x.N)$, then we will let $M \rhd_{\alpha,1} M'$. The transitive closure of the $\rhd_{\alpha,1}$ relation will be denoted $\rhd_\alpha$.

- β *reduction* introduces a semantic computational step into the calculus. Given a β *redex*, a term which can be written $(\lambda x.M)N$, we write $(\lambda x.M)N \rhd_{\beta,1} M[N/x]$, provided that N is 'safe to substitute' for x in M. This means that every free occurrence of x in M occurs in a subterm of the form $(\lambda y.M_0)$, where y is a free variable in N. Similarily, we will $\rhd_{\beta,1}$ be allowed on subterms of a general term, and the transitive closure will be denoted $\rhd_\beta$.

- A relation $\rhd$ satisfies the ξ *rule* if $M \rhd N$ implies $\lambda x.M \rhd \lambda x.N$.

- A relation $\rhd$ is closed under composition if $M_0 \rhd N_0$ and $M_1 \rhd N_1$ implies $(M_0 M_1) \rhd (N_0 N_1)$.

The general reduction relation, which we shall denoted by $\rhd$ or $\rhd_\lambda$, is the smallest transitive relation $\rhd$ containing α and β reduction. It satisfies the ξ rule, and is

closed under composition, because α and β reduction are allowed over subterms. Conversely, $\rhd$ is also the smallest transitive relation containing α and β reduction *not* over subterms, closed under composition, and satisfying the ξ rule. As an example, since

$$(\lambda yx.xy)xy \rhd_\alpha (\lambda y_0 x_0.x_0 y_0)xy \rhd_\beta yx$$

we find that $(\lambda yx.xy)xy \rhd yx$. If we enlarge reduction to be the smallest reflexive, symmetric relation, we obtain the notion of equivalence, which we write as $M =_\lambda N$, or if we want to emphasize the proof theoretic aspects, as $\lambda \vdash M = N$.

As $\rhd$ is the smallest transitive relation containing α and β reduction, we obtain an inductive method to prove that a property $R(x, y)$ over the terms of the λ calculus satisfies $R(M, N)$ for any M which reduces to N. It suffices to verify that

- R is transitive.
- $R((\lambda x.M)N, M[N/x])$, where N is safe for substitution for x in M.
- If $R(M_0, M_1)$ and $R(N_0, N_1)$ hold, then $R(M_0N_0, M_1N_1)$ also holds.
- If $R(M, N)$ holds, then $R(\lambda x.M, \lambda x.N)$ holds.

because then R is a transitive relation containing α and β reduction, and satisfying the ξ and composition rule, hence R contains the relation of reduction, and so if $M \rhd N$, then $R(M, N)$ necessarily holds. If R is symmetric and reflexive, then we can also conclude that if $M =_\lambda N$, then $R(M, N)$ holds.

It is important to consider how important it is that we only perform β reduction on terms safe for substitution. If not, then we could conclude that

$$(\lambda xy.x)y \rhd_\beta (\lambda y.y)$$

$$(\lambda xy.x)y \rhd_\alpha (\lambda xy_0.x)y \rhd_\beta \lambda y_0.y$$

We would like to interpret reduction as contracting a function definition to a shorter definition which defines an equivalent function. However, the two functions above certainly should not be equivalent. Indeed, the first represents the identity function $f(x) = x$, and the second represents the constant function $g(x) = y$, for some y. Even without a semantic interpretation, we can further use these reductions to conclude that

$$\lambda \vdash M = (\lambda y.y)M = (\lambda y_0.y)M = y$$

so we find that any two terms of the λ calculus are equal. This is clearly not desirable in an actual theory for representing interesting classes of operators.

We say a term M is in β *normal form* if it contains no subterms which form a β redex. This means exactly that the only reductions possible from M are α reductions: If $M \rhd N$, then $M \rhd_\alpha N$. Since α conversion isn't much of an interesting calculation, we view β normal forms as forms which have been completely computed. We can then interpret β normal forms as terms representing algorithms which eventually halt.

Example. *The term* $(\lambda x.(\lambda y.yx)z)a$ *has normal form* za*, because*

$$(\lambda x.(\lambda y.yx)z)a \rhd_\beta (\lambda x.zx)a \rhd_\beta za$$

and za *does not contain any* β *redexes.*

Example. *The term* $\Omega = (\lambda x.xx)(\lambda x.xx)$ *has no normal form, because* α *reduction only changes the term to* $(\lambda y.yy)(\lambda z.zz)$*, for some variables* y *and* z*, and* β *reduction on any term of this form results in* $(\lambda z.zz)(\lambda z.zz)$*. Thus the set of all compositions of terms of this form is closed under* β *and* α *reduction, and all of them contain a* β *redex, so we conclude that* Ω *has no normal form. It is singular because it is the only term up to* α *conversion which cannot be reduced to some other term.*

There is a more formal way to define the class of all normal forms. We take the following inductive definition:

- All variables are in normal form.
- If M is in normal form, and x is a variable, then xM is in normal form.
- If M is in normal form, then $\lambda x.M$ is in normal form.

Surely any element of this grammar is in normal form. Conversely, if xM is in normal form, then so too is M because the subterms of M cannot be β redexes, and if $(\lambda x.M)$ is in normal form, then so too is M because M is in normal form. Thus the problem of deciding whether a term of the λ calculus is in β normal form can be expressed as a context free grammar, a computable operation. However, the problem of determining whether a term of the λ calculus can be *reduced* to a term in β normal form is undecidable. It is essentially the same problem as proving a Turing machine halts.

4.2 Consistency and Church-Rosser

We can express the λ calculus as an equational theory of logic. However, the theory is not a first order logic, because the terms of the calculus contain variables which may be bound, and this is not possible in vanilla first order logic. In order to get around this, we form a formal theory with only a single predicate, so that the propositions of the system are exactly of the form $M = N$, for some terms M and N, and we consider the reduction rules as inferences from the basic equality axioms. Because the λ calculus is not modelled as a first order theory, the model theory will have to be slightly different. The axioms of the system are

$$\begin{aligned} (\alpha) \quad & \lambda x.M = \lambda y.M[y/x] \\ (\beta) \quad & (\lambda x.M)N = M[N/x] \\ (\mathrm{id}) \quad & M = M \end{aligned}$$

and the inference rules are

$$\frac{M = N}{LM = LN}\ (\mu) \qquad \frac{M = N \qquad N = L}{M = L}\ (\tau) \qquad \frac{M = N}{LM = LN}\ (\nu)$$

$$\frac{M = N}{N = M}\ (\sigma) \qquad \frac{M = N}{\lambda x.M = \lambda x.N}\ (\xi)$$

An equation is provable in this axiom system if and only if the equation is true for the λ terms.

Since the λ calculus is equational, there is no way to talk about the standard consistency of the inference rules – there is no such thing as a contradiction, because we don't necessarily have a negation connective. However, the syntax does give rise to some predicates in the metalanguage, those being the equality predicates $\lambda \vdash M = N$. Since formal systems using classical logic are inconsistent precisely when every statement in the language can be proved, we will say that the λ calculus is consistent if *not every* equation can be proved. That is, there is some M and N such that $M = N$ cannot be proved.

The Church-Rosser theorem is the central result to proving the consistency of the λ calculus. It is clear from the result that a β normal form is unique up to α reduction. Thus if two terms M and N are in normal form, and cannot be α converted into one another, then we cannot prove that $M = N$. We surely have two terms which are not α equivalent, so we can conclude that the λ calculus is a consistent theory, as a direct result.

Theorem 4.1 (Church-Rosser for $\triangleright$)**.** *If $M \triangleright M_0$ and $M \triangleright M_1$, then there is a term N such that $M_0 \triangleright N$ and $M_1 \triangleright N$.*

Proof. Annoyingly technical, so I'll prove it some other time. □

The property that the Church-Rosser theorem proves for $\triangleright$ is called *confluence*. There is an analogous result for equality of terms. Note that for any symmetric relation, confluence is trivial, so the theorem has to be strighly stronger than this to be interesting.

Theorem 4.2. *If $\lambda \vdash M = N$, then there is a term L such that $M \triangleright L$ and $N \triangleright L$.*

Proof. First, note that if $\vdash M = N$, then there is $L_1, \ldots, L_n$ with $L_0 = M$, $L_n = N$, and either $L_k \triangleright L_{k+1}$ or $L_{k+1} \triangleright L_k$. We shall prove this theorem by induction on n. If $n = 1$, the theorem is trivial, because M is syntactically equal to N. If $n = 2$, then either $M \triangleright N$ or $N \triangleright M$, and then the theorem is just the Church-Rosser theorem for $\triangleright$. Otherwise, by induction, we may assume that there is L with $L_1 \triangleright L$ and $L_{n-1} \triangleright L$. If $L_n \triangleright L_{n-1}$, then $L_n \triangleright L$, hence the theorem is complete. Thus we may assume $L_{n-1} \triangleright L_n$. But then by applying Church-Rosser for $\triangleright$, we conclude that there is K such that $L \triangleright K$ and $L_n \triangleright K$, and this completes the proof, because $L_1 \triangleright L \triangleright K$. □

Essentially, what we have proved is that if R is any transitive confluent relation, and we take the smallest symmetric extension R', then if $R'(x, y)$, then $R(x, z)$ and $R(y, z)$ for some z.

Corollary 4.3. *If $\lambda \vdash M = N$, and N is in β normal form, then $M \triangleright N$.*

Proof. The last theorem shows that $M \triangleright L$ and $N \triangleright L$ for some term L. But then L is alpha congruent to N, hence $L \triangleright N$, and so $M \triangleright N$. □

Corollary 4.4. *If $\lambda M = N$, and M and N are both in β normal form, then M and N are α equivalent.*

Terms without a normal form correspond to algorithms which don't terminate. Such terms are essentially meaningly in the λ calculus. But the fact that a term has a normal form does not imply that subterms of the term have a normal form. For instance, $(\lambda x.y)\Omega$ has a normal form y, whereas we know Ω does not have a normal form. This means that a 'meaningful' term may have meaningless subterms. This seems undesirable, so it is of interest to reduce the terms of the λ calculus so that subterms of terms with normal forms have normal forms. Church found the system λI with this property. It is defined in essentially the same way as the standard λ calculus except that when forming the terms of the system, we only let $(\lambda x.M)$ be a term when x is a free variable in M.

4.3 Combinators

An alternative formal system in which to discuss combinatory logic is the theory of combinators. One can see this theory as the subtheory of the λ calculus generated by the closed terms. Suprisingly, this subtheory turns out to be equivalent in expressive power to the entire λ calculus theory. This may seem unreasonable in the context of λ calculus, for we need free variable terms in order to form future functions. However, we can represent λ abstraction via the composition of closed λ terms, so that there is a formal system for combinatory logic with composition as the only connective. We will introduce variables into combinatory logic for convenience, but we note they are not required for most discussions of the calculus, and are much simpler than variables in the λ calculus because there is no way to bind variables.

Combinatory logic was invented a decade before the λ calculus, and can be developed with no mention of Curry's theory at all. One starts with a set of primitive combinators, and a set of variables, and inductively constructs the set of all combinators.

- Every primitive combinator and variable is a combinator.
- If A and B are combinators, then (AB) are combinators.

Equivalence of terms is generated based on substitution rules for the primitive combinators. We associate with each base combinator C a substitution rule of the form $Cx_1 \dots x_n \rhd_A A_C$, where A_C is another term in the λ calculus. We can then define one step reduction as the relation formed by the rules $CB_1 \dots B_n \rhd_{C,1} A_C[B_1/x_1, \dots, B_n/x_n]$, or more generally, if this reduction occurs in a subterm of a combinator. The union of all $\rhd_{C,1}$ is one step reduction $\rhd_1$, and the transitive closure is general reduction $\rhd$. We can then consider equivalences by taking the reflexive symmetric closure, and we denote this by $M =_{\mathrm{CL}} N$, or $\mathrm{CL} \vdash M = N$.

Example. *There are some classical combinators which are universally known.*

$$\begin{array}{ccc} Ix \rhd x & Bxyz \rhd x(yz) & Sxyz \rhd xz(yz) \\ Kxy \rhd x & Cxyz \rhd xzy & Wxy \rhd xyy \\ Mx \rhd xx & B'xyz \rhd y(xz) & Yx \rhd x(Yx) \end{array}$$

Note that all the combinators but the Y combinator have axioms which are formed from the variables in the definition. We call these types of combinators proper combinators*, rather than* improper combinators*.*

Combinators can be classified by five features, based on how the substitution relations work. *Identity combinators* are those which operate by reductions of the form $Cx_1 \ldots x_n \rhd_C x_1 \ldots x_n$. The classical I combinator above is an identity combinator of arity one. The combinator BI satisfies $BIxy \rhd_B I(xy) \rhd_I xy$, hence BI acts as an identity combinator of arity two. An *associator* is a combinator which groups the input terms. The B combinator is an associator, and is the only non-trivial associator with arity three (the other associator would be $Bxyz \rhd (xy)z$, but this is just an identity combinator of arity three). An example of a more complicated combinator is a combinator C with the axiom $Cxyzvw \rhd_C x((yz)vw)$. *Cancellators* remove variables from an input, like $Kxy \rhd_K x$, and *permutators* permutes arguments. Finally, *duplicators*, as expected, duplicate arguments.

Since combinatory logic is an equational theory, we can also talk about consistency. It is often useful to use a sequent calculus for describing the theory. The only new axiom is primitive reduction

$$(\rho) \qquad CB_1 \ldots B_n = A_C[B_1/x_1, \ldots, B_n/x_n]$$

in addition to the axiom $M = M$. We then use the standard inference rules

$$\frac{M = N}{LM = LN}\ (\mu) \qquad \frac{M = N \qquad N = L}{M = L}\ (\tau) \qquad \frac{M = N}{LM = LN}\ (\nu)$$

$$\frac{M = N}{N = M}\ (\sigma)$$

One interesting difference between combinatory logic and the lambda calculus is that, because combinatory logic has no variable binding operations, the system can be formalized as a first order equational theory. This will have interesting consequences for the differences between the model theory of combinatory logic and the model theory of the λ calculus.

A *weak redex* is a combinator of the form $AB_1 \ldots B_n$, on which we can perform a reduction. If a term contains no weak redexes, we say it is in *weak normal form*, which is essentially the end result of a calculation. There is a version of the Church Rosser theorem for reduction in combinatory logic, so that $\rhd$ is a confluent operation, and the system of combinatory logic is consistent. That is, weak normal forms are unique if they exist. Furthermore, if $M =_{\mathrm{CL}} N$, there is a combinator L such that $\mathrm{CL} \vdash M \rhd L$ and $\mathrm{CL} \vdash N \rhd L$.

Intuitively, any base term in combinatory logic can be modelled in the λ calculus, because the axioms for primitive combinatory logic are analogous to closed λ abstractions. For instance, the standard combinator axioms

$$\begin{array}{ccc} Ix \rhd x & Bxyz \rhd x(yz) & S\,xyz \rhd xz(yz) \\ Kxy \rhd x & Cxyz \rhd xzy & Wxy \rhd xyy \\ Mx \rhd xx & B'xyz \rhd y(xz) & Yx \rhd x(Yx) \end{array}$$

are analogous to

$$\begin{array}{ccc} \lambda x.x & \lambda xyz.x(yz) & \lambda xyz.xz(yz) \\ \lambda xy.x & \lambda xyz.xzy & \lambda xy.xyy \\ \lambda x.xx & \lambda xyz.y(xz) & \lambda xyzv.xy(xvz) \end{array}$$

Conversely, any closed λ term with all the λ terms to one side is analogous to an axiom. The position of the λ terms is important to the theory, which hints at why the syntactic theory of combinatory logic and the λ calculus is different. However, we should expect that sufficiently powerful combinators should lead to a form of logic expressive enough to represent all substitution rules.

We will say a combinatory logic is *combinatorially complete* if, for any axiom rule of the form $Cx_1 \ldots x_n = A_C$, there is a combinator C' (not necessarily primitive) such that $C'B_1 \ldots B_n \rhd A_C[B_1/x_1, \ldots, B_n/x_n]$ for all combinators B_i (we need only show this for a set of variables). The most well known combinatorially complete logic consists of the primitive combinators $\{S, K\}$, but other combinatory basis, like $\{I, B, C, W, K\}$ and $\{I, J, K\}$, exist. We shall actually prove that $\{S, K, I\}$ is combinatorially complete, rather than $\{S, K\}$, but since $SKKx \rhd I$ for all combinators x, the combinatorial completeness of $\{S, K, I\}$ implies the combinatorial completeness of $\{S, K\}$. Our method to do this is to introduce a form of 'lambda abstraction' which forms a part of the metatheoretic language to express all substitution rules.

Given a variable x and a combinator M, we shall define a combinator $[x].M$ such that $[[x].M]y \rhd M[y/x]$ for all combinators y. This will enable us to write arbitrary substitution rules in the calculus.

- If none of the x occur in M, define $[x].M = KM$. Then $KMy \rhd M$, and $M = M[y/x]$, hence $[x].M$ satisfies the required property.
- If none of the x occur in M, defined $[x].Mx = M$. Then $My \rhd My$, and $My = [Mx][y/x]$. This is not necessary, because it is covered by other cases, but leads to a simple formula for λ abstraction.
- Let $[x].x = I$. Then $Iy \rhd y$, and $y = x[y/x]$.
- Otherwise, let $[x].(MN) = S\,[[x].M][[x].N]$. Then by induction,

$$S\,[[x].M][[x].N]y \rhd ([[x].M]y)([[x].N]y) \rhd M[y/x]N[y/x]$$

and $M[y/x]N[y/x] = (MN)[y/x]$.

Note that $[x].M$ does not contain the variable x. Thus if we extend substitution multidimensionally, so that $[x_1, \ldots, x_n].M$ is defined recursively as $[x_1].[[x_2, \ldots, x_n].M]$. By induction, we find

$$[[x_1, \ldots, x_n].M]x_1 \ldots x_n \rhd M$$

And because $[[x_1, \ldots, x_n].M$ does not contain the variables x_i, we find by substitution that $[[x_1, \ldots, x_n].M]y_1 \ldots y_n \rhd M[y/x]$. This uses the general principle that if $M \rhd N$, then $M[y/x] \rhd N[y/x]$. So now, if we have an axiom of the form $Cx_1 \ldots x_n \rhd_C A_C$, we let $B = [x].A_C$. Then we have shown that $By_1 \ldots y_n \rhd A_c[y/x]$, hence $\{K, S\}$ is combinatorially complete. Essentially, this is the main technique to proving a set of combinators is complete, albeit reducing the set to a basis which is already proved complete.

Since we have a form of 'lambda abstraction' for combinators, it is an interesting question to ask whether the ξ rule holds in our calculus. That is, if $\vdash_{\mathrm{CL}} M = N$, then does it necessarily hold that $\vdash_{\mathrm{CL}} [x].M = [x].N$. One difference between combinatory logic and the lambda calculus is that the ξ rule need not hold in combinatory logic.

Example. *If $M = S\,xyz$ and $N = xz(yz)$, then $M \rhd_S N$, yet*

$$[x].M = S\,(S\,(S\,(KS\,)I)(Ky))(Kz)$$

$$[x].N = S\,(S\,I(Kz))(S\,(Ky)(Kz))$$

and both elements are in normal form, hence they cannot be CL equivalent.

The lack of the ξ rule is normally no problem, but sometimes it does cause issue, which means λ calculus becomes a more applicable theory. Conversely, combinatory logic has a more elegant theory of substitution, so we must make a tradeoff between the two. The ξ rule will return when we analyze the similarities between the formal theory of the λ calculus and the formal theory of combinatory logic.

4.4 Extensionality

An intensional process is one which is defined by its description, whereas an extensional process is one uniquely defined by its inputs and outputs. The functions

of set theory are extensional, because two functions are equal if and only if they agree on all inputs. Conversely, the terms of the λ calculus and the combinators of combinatory logic are viewed as intensional descriptions of processes, because they describe the process of substitution, and just because Mx is equivalent to Nx for all inputs x does not imply that M is equivalent to x. Algorithms in computing science are intensional, because two algorithms which solve the same problem are not necessarily viewed as equivalent, especially if the runtime of the algorithms is under scrutiny.

It turns out that the theory of the λ calculus is equivalent to combinatory logic as a formal system, provided we introduce extensionality to the system. Recall that a definition of function is extensional if functions are identified by their inputs and outputs, as in set theory. That is, if $fx = gx$ for all x, then $f = g$. The λ calculus is an intensional theory because this need not hold. For instance, if $f = y$, and $g = (\lambda x.yx)$, then

$$\lambda \vdash gx = (\lambda x.yx)x = yx = fx$$

yet we cannot prove that $g = f$ in the λ calculus, since both terms are in normal form. Similar results hold for combinatory logic – we can define two primitive terms with the same substitution rule, yet the two terms will be unequal in the theory.

We extend the formal theory of the λ calculus to be intensional by adding an additional inference scheme to the proof system. If x is a variable, and M and N are terms not containing x as a free variable, we add the rule

$$\frac{Mx = Nx}{M = N}\ (\zeta)$$

The extended theory is known as the $\lambda\zeta$ calculus, and we write $\lambda\zeta \vdash M = N$ for entailment of the equation theory. We can also consider adding the axiom

$$(\eta) \quad (\lambda x.Mx) = M$$

where x is not a free variable of M, and we write $\lambda\eta \vdash M = N$ for this formal theory.

Theorem 4.5. *The inference rule ζ is correct in the $\lambda\eta$ theory, in the sense that if $\lambda\eta \vdash Mx = Nx$, and M and N do not contain x as a free variable, then $\lambda\eta \vdash M = N$. Conversely, the axiom η is derivable in the $\lambda\zeta$ theory.*

Proof. The first part of the theorem is proved by the following sequent tree in the $\beta\eta$ calculus.

$$\dfrac{\dfrac{(\lambda x.Mx) = M \quad (\eta)}{M = (\lambda x.Mx)}\ (\sigma) \qquad \dfrac{\dfrac{Mx = Nx}{(\lambda x.Mx) = (\lambda x.Nx)}\ (\xi) \qquad (\lambda x.Nx) = N \quad (\eta)}{(\lambda x.Mx) = N}\ (\tau)}{M = N}\ (\tau)$$

The second is a very simple derivation of the axiom η in the $\beta\zeta$ calculus.

$$\dfrac{(\lambda x.Mx)x = Mx \quad (\beta)}{(\lambda x.Mx) = M}\ (\zeta)$$

Hence the $\beta\eta$ and $\beta\zeta$ systems have the same theorems. $\beta\eta \vdash M = N$ if and only if $\beta\zeta \vdash M = N$. □

The ξ rule is essential to proving that $\beta\zeta$ and $\beta\eta$ are equivalent systems, and this is one reason why ξ is known as the principle of weak extensionality. Another reason is that ξ becomes redundant in an extensional system, in the sense that it is provable in the $\beta\zeta$ if we remove the ξ rule. Conversely, ζ is not provable if we remove the ξ rule from the $\beta\eta$ formal system.

Since η gives us an additional reductions $\rhd_\eta$ of the form $(\lambda x.Mx) \rhd_\eta M$, certain β normal forms can be further reduced. The induced reduction operation is still confluent – if $M \rhd M_0$ and $M \rhd M_1$, there is N such that $M_0 \rhd N$ and $M_1 \rhd N$. A term is in $\beta\eta$ normal form if it is in β normal form, and if there are no subterms of the form $(\lambda x.Mx)$, where M does not contain x as a free variable. This is essentially the termination point of computations in the $\beta\eta$ calculus. Since there is more than one $\beta\ \eta$ normal form, even modulo α congruence, the $\beta\eta$ calculus is consistent.

Theorem 4.6. *A term has a β normal form iff it has a $\beta\eta$ normal form.*

Proof. We use the inductive construction of β normal forms to prove that every β normal form has a $\beta\ \eta$ normal form.

- If M is a variable, then M is already in $\beta\eta$ normal form.
- If $M = xN$, and $N \rhd N_0$, where N_0 is in $\beta\eta$ normal form, then $M \rhd xN_0$, and xN_0 is in $\beta\eta$ normal form.
- If $M = \lambda x.N$, and $N \rhd N_0$, where N_0 is in normal form, then $M \rhd \lambda x.N_0$, and either $\lambda x.N_0$ is in $\beta\eta$ normal form, or $N_0 = N_1 x$, where N_1 is in $\beta\eta$ normal form, and $M \rhd \lambda x.N_0 \rhd_\eta N_1$, and so M is reducable to η normal form.

Essentially, η reduction always reduces the complexity of a term, so the induction step works. □

By the same process, we find that if $M \rhd_{\beta\eta} N$, then $M \rhd_\beta M_0$, and $M_0 \rhd_\eta N$, so we can postpone η reduction to the end of calculations.

Combinatory logic is also a non-extensional system. For instance, two prime combinators may have equal axioms without being formally equal in the system. Remember that CL does not satisfy the ξ rule, and thus does not possess the principle of weak extensionality. There is no problem with adding the ξ rule as an axiom, however.

$$\frac{M = N}{[x].M = [x].N} \ (\xi)$$

What's more, we could think of adding the extensionality principles as axioms of CL.

$$\frac{Mx = Nx}{M = N} \ (\xi)$$

$$[x].Mx = M \quad (\eta)$$

where M and N do not contain x as a free variable. The formal theory of equality found by adding the ξ rule is denoted CLξ, and the theory of equality found by adding ζ is denoted CLζ.

Example. *In CL, we find that*

$$SKyx \rhd (Kx)(yx) \rhd x$$

for any combinator y. Therefore in CLζ, we conclude that $SKy = SKz$ for all combinators y and z. Similarily, we find that $KIyx \rhd x$, hence $SKy = KIy$ in CLζ, hence $SK = KI$.

In λ calculus, the ξ rule was essential to proving that $\lambda\beta\eta$ was equivalent to $\lambda\beta\zeta$. This was not so important, because the ξ rule was imbedded in the basic theory. Conversely, in combinatory logic the η rule already holds by definition of the term $[x].Mx$. Using the same techniques as for the extensionality of the λ calculus, we can therefore prove that CLζ is an equivalent theory of equality to CLξ. We remark that we could have left out the special case $[x].Mx$ in the definition of substitution, in which ase the η rule would not hold by definition, and then CLξ would be a strictly weaker theory than CLζ. Various definitions of substitution, and the corresponding ξ rules will result in different systems of equality.

Now to prove the consistency of CLζ, we apply the standard technique: find a system of reduction which specifies the equality, prove confluence, and show that more than one normal form exists. We define *strong reduction* $\rhd_s$ on combinators to be the standard theory of reduction, with an additional result that if $M \rhd_s N$, then $[x].M \rhd_s [x].N$ (and conversely, we sometime denote normal reduction as *weak reduction* $M \rhd_w N$). I know of no proof of the Church Rosser property for strong reduction which is directly proven from this definition. The main way that strong reduction is proved confluent is by relating the notion directly to the extensional λ calculus. The normal forms here are called *strongly irreducible*. We note, however, that strong reduction is very difficult to work with, which is one reason why little attention to it has been considered.

4.5 Equivalence of Combinatory Formal Systems

Both combinatory logic and the λ calculus accurately model functions obtained by substituting terms. They seem to have very similar properties, albeit from a few small differences. The ξ rule fails in Combinatory logic, whereas the η rule fails in λ calculus. It turns out that if we add these rules to the formal systems, thereby considering their extensional forms, the two systems will have very strong equivalence properties.

First, we shall begin by considering a slight modification to the λ calculus. The terms will not consist of strings of symbols, but rather an equivalence class of strings defined modulo α conversion. In this form of the λ calculus, α reduction does not even need to be considered in the theory, because it is just equality in this modified system. We will let the set of equivalence classes of terms in this calculus be denoted by Λ.

Now given a particular combinatory logic with combinators, with the same variable set as a corresponding λ calculus we shall define a λ *transform* $M \mapsto M_\lambda$ which takes combinators in the logic to α identified terms in Λ which preserves the operation of reduction. This will be the first form of equivalence. If a primitive combinator C has the axiom $Cx_1 \ldots x_n \rhd_C A_C$, then we shall define $C_\lambda = \lambda x_1 \ldots x_n.A_C$. We can then define the transform of general combinators by letting $(MN)_\lambda = M_\lambda N_\lambda$. Each M_λ is a *closed* term of the λ calculus, and we call such closed terms combinators, because of this. As a first result, we note that $[M[N/x]]_\lambda = M_\lambda[N_\lambda/x]$.

Lemma 4.7. *If $M \rhd_w N$ in CL, then $M_\lambda \rhd_\beta N_\lambda$. Thus if $M =_{CL} N$, then $M_\lambda =_\beta N$. Conversely, if $M =_{CL\zeta} N$, then $M_\lambda =_{\beta\zeta} N_\lambda$.*

This is just proved by induction on the length of a proof of $M \rhd_w N$ and $M =_{\mathrm{CL}\zeta} N$, and is left to the reader.

To obtain an if and only if result, we require a combinatory logic with a basis of combinators expressive enough to represent all possible solutions in the λ calculus. Thus from now on, we assume our combinatory logic contains only the primitive operators S, K, and I. We note that the λ transform of this form of combinatory logic is specified by

$$I_\lambda = (\lambda x.x) \quad K_\lambda = (\lambda xy.x) \quad S_\lambda = (\lambda xyz.xz(yz))$$

We can then define $[x].M$ for any combinatory term M, and this allows us to form an inverse λ transform. We define the CL transform $M \mapsto M_{\mathrm{CL}}$ of any term in Λ. We start by letting $x_{\mathrm{CL}} = x$ for variables x, let $(MN)_{\mathrm{CL}} = M_{\mathrm{CL}}N_{\mathrm{CL}}$, and let $(\lambda x.M)_{\mathrm{CL}} = [x].[M_{\mathrm{CL}}]$. Because $[x].[M_{\mathrm{CL}}]$ does not contain any instances of the variable x, the term is well defined up to α congruence.

Lemma 4.8. *For any combinator M, $[M_\lambda]_{CL} = M$.*

Thus the λ transform has a left inverse. Note, however, that the CL transform is not injective, because

$$(\lambda x.yx)_{\mathrm{CL}} = [x].yx = S\,(Ky)I$$

and

$$[S\,(Ky)I]_\lambda = ((\lambda uvw.uw(vw))((\lambda uv.u)y))(\lambda u.u)$$

Though it is surjective.

Lemma 4.9. *If $M =_{\beta\zeta} N$, then $M_{CL} =_{CL\zeta} N_{CL}$.*

As a corollary, we see that $M = N$ in CLζ if and only if $M_\lambda = N_\lambda$ in the $\lambda\beta\zeta$ theory, and $M = N$ in the $\lambda\beta\zeta$ theory if $M_{\mathrm{CL}} = N_{\mathrm{CL}}$ in CLζ.

We note that the correspondence for the reduction rules of CL and λ are nowhere near as elegant. It is easy to prove that if $M \rhd_{\beta\zeta} N$, then $M_{\mathrm{CL}} \rhd_s N_{\mathrm{CL}}$, but the converse does not hold. Since reduction is really only used to form an equivalence of terms, this isn't too much of an issue.

4.6 The Power of λ calculus

The core problem with the λ calculus, and a combinatorially complete combinatory logic, is that it is too expressive in its full form. One result is the following paradox, known as the fixed point theorem.

Theorem 4.10. *For any term M, there is a term N such that* $MN =_\lambda N$.

Proof. Let $N = \lambda x.f(xx)$, and let $M = NN$. Then

$$M = NN = (\lambda x.f(xx))N \rhd_\beta f(NN) = f(M)$$

Thus we have found a fixed point. □

If we are to interpret terms of the λ-calculus as real functions, we must be very careful, because otherwise this theorem would imply every theorem has a fixed point! This is clearly not true for all functions – a particular example in logic is the negation operator, and this theorem would imply $\neg x = x$ for some x. We will address these problems once we have developed a syntax theory for the calculus.

4.7 Models of the λ Calculus and Combinatory Logic

The formal theories of combinatorial logic are fun to play around with, but it is an interesting question what the theory actually *represents*. Combinatory logic is strange among formal systems in the sense that it has no immediate semantic interpretation, because it is immediately too powerful to be represented by set theoretic functions.

Lambda abstraction is a difficult concept to try and model, so let's begin by focusing only on composition, via models of combinatory logic. We cannot model combinators as functions directly, but there is a common method in mathematics to apply elements of sets to the set themselves. For instance, in Hilbert space theory, we can associate with each point x in the space the functional $\langle x|$, which operates on the space by the inner product. In the theory of groups, we can compose elements together by a multiplication operation. Thus we would expect a suitable environment to model combinatory logic as elements of a set X together with a composition operation $X \times X \to X$. The composition operation will be written $(x, y) \mapsto xy$ except where this becomes ambiguous, and we assume terms are left associative, so we can consider products of the form $x_1 x_2 \dots x_n$ for $x_i \in X$. Another way of getting around the fact is to represent functions on X as points $x \in X$. We shall let the function associated with $x \in X$ be denoted $f_x : X \to X$. We can then define composition of functions as $(f_x f_y) = f_{f_x(y)}$. This is essentially the same method as with the abstract composition operation, since by currying an association $F : X \to X^X$ of points with functions on sets naturally reduces to

$F : X \times X \to X$. We shall call a set X with a composition operation an *applicative structure*.

A function $f : X^n \to X$ is *representable* on an applicative structure X if there is an element $a \in X$, such that $f(x_1, \ldots, x_n) = ax_1 \ldots x_n$ for all $x_i \in X$. Just because we have an applicative structure on a set X, does not imply that *all* functions on the set are representable as elements of the set. In fact, we can guarantee that this does not occur, because Cantor proved that X^X always has a cardinality strictly greater than X, so there cannot exist a surjective map $X \to X^X$. However, we shouldn't expect a model of combinatory logic to model all functions on a set, because one reason for combinatory logic's existence was to model only the *computable functions*.

We shall define a model of combinatory logic with primitive combinators **B** to be an applicative structure X with more than one element together with an association $\rho : \mathbf{B} \to X$, where we denote $\rho(C)$ as x_C. If a general combinator C has free variables $y_1, \ldots, y_n$, we will let $x_C[x_1/y_1, \ldots, x_n/y_n]$ be the element of X fromed by mapping y_i to x_i, and then considering the term closed under composition. For a model of combinatory logic to be a true model, the association $x_C x_1 \ldots x_n = A_C[x_1/y_1, \ldots, x_n/y_n]$ for any $x_i \in X$, where C has a substitution axiom $Cy_1 \ldots y_n \rhd A_B$. Thus the composition rule of the applicative structure naturally represents the substitution rule for each primitive combinator. Now give a model X, and two terms M and N in combinatory logic, we write $X \models M = N$ if, for all choices of $x_i \in X$, $x_M[x_1/y_1, \ldots, x_n/y_n] = x_N[x_1/y_1, \ldots, x_n/y_n]$, where the y_i are free variables in M and N. This gives us a semantic theory for combinatory logic.

Note that what we have done is not really any more general than the model theory for first order logic, since we can write the terms of combinatory logic as terms of a first order system with a single equality predicate, where the primitive combinators are constants. A model is then just a set X together with a map f from the primitive combinators to X, and we can define an applicative structure on X by taking the interpretation of composed terms in the first order logic.

Two terms in $a, b \in X$ are *extensionally equivalent* if $ax = bx$ for all $x \in X$, or equivalently, if $f_a = f_b$. An applicative structure X is extensional if $f_a = f_b$ holds if and only if $a = b$. A model of extensional combinatory logic should naturally be an extensional applicative structure, and this is certainly true if we interpret a model of this logic as a normal model of the first order theory. By consistency, there exists a model of extensional combinatory logic.

Part II

Set Theory

Almost all mathematicians use the following axioms of set theory, specified by Zemerlo-Fraenkel. The axioms of set theory consider only a universe consisting only of a single type of objects, the *sets*, and a single primitive binary relation of *elementhood*, denoted as $A \in B$:

- *Axiom of Extensionality*: If two sets X and Y have the same elements, then $X = Y$.
- *Axiom Scheme of Separation*: If P is a property of sets, then for any set X, there is a set
$$Y = \{x \in X : P(x) \text{ is true}\},$$
i.e. for any s, $s \in Y$ if and only if $s \in X$ and $P(s)$ is true.
- *Axiom of Union*: For any set $\mathcal{X}$, there exists a set
$$\bigcup \mathcal{X} = \left\{ s : \text{there is } X \in \mathcal{X} \text{ such that } s \in X \right\}$$
known as the *union* of X.
- *Axiom of Power Set*: For any X, there exists a set
$$\mathcal{P}(X) = \{S : S \subset X\},$$
known as the *power set of* X, where $S \subset X$ is the property that for any s, if $s \in S$, then $s \in X$.
- *Axiom of Infinity*: There exists an infinite set.
- *Axiom Scheme of Replacement*: If F is a function, then for any X there exists a set $Y = F[X] = \{F(x) : x \in X\}$.
- *Axiom of Regularity*: Every nonempty set has a $\in$-minimal element.
- *Axiom of Choice*: Every family of nonempty sets has a choice function.

The theory with all these axioms is denoted ZFC, and ZF without the axiom of choice.

Part III

Computability

In 1931, Kurt Gödel proved all sufficiently complicated axiomatic systems had unprovable theorems, but a fundamental question remained: by what method could we decide whether a theorem could be proved? It took a decade for Alonzo Church and Alan Turing to deduce the impossibility of such a claim. Fifty years later, 'theoretical computation' had become a common reality. In this section, we introduce the mathematical models which formed the foundation for the computer, as well as more modern models which analyze the limitations of various computational methods.

Turing and Church's major breakthrough was precisely defining what a 'computational procedure' is. It is often the case that precise definitions give rise to easy proofs of the most surprising consequence. We shall spend many chapters contemplating what power a computational procedure should have. Philosophically, one should be able to define such a procedure without reference to a computer, for humans computed long before microchips. On the other hand, models should reflect physical reality, since one needs a physical mechanism in order to compute, whether electronic or mental. If your computational model is too strong or too weak, it will not accurately represent the limitations of real life.

We will begin by analyzing the automaton, a model of computation without stored memory. We will expand the amount of expression of the automaton by considering context-free grammars. Finally, we add memory by considering a Turing machine. It is the Church Turing thesis that this is the ultimate model of computation – any real world computation can be modelled as an action on a Turing machine. There has been no evidence in the past century to contradict this thesis, and every realistic model of computation is not as powerful as that of the Turing machine, so we accept the claim. From this model, and with the hypothesis of Church and Turing, we can make precise, philisophically interesting statements about the nature of computation in the real world, addressing theorems about the uncomputability and complexity of certain problems.

As in mathematical logic, the objects of study are strings of symbols over a certain alphabet. One studies the notion of computation syntactically. One of the main ideas of computability theory is that a mental decision can be modelled as a *decision problem* – find a computational model which will 'accept' certain strings over an alphabet. Suppose our problem is to verify whether the addition of two numbers is correct. We are given a, b, and c, and we must decide whether $a+b=c$. Our symbol set is $\{0,1,\ldots,9,:\}$, and we wish to model a computation which accepts all strings of the form $\mathsf{a}:\mathsf{b}:\mathsf{c}$, where a, b, and c are decimal strings for which $a+b=c$. Thus we must design machine to accept strings in a specified language, and determining whether a problem is solvable reduces to studying the

structure of languages over a finite alphabet. We shall find that, obviously, this problem is possible to compute on a Turing machine, but it is not so simple – there are some models of computation which are unable to decide whether addition is correct.

As a more dynamic discipline than mathematical logic, we need more operations on strings to obtain languages from other languages. We obviously need concatenation, but also *reversal*, which will be denoted s^R. These operations are extended to languages by applying the operations on a component by component basis:

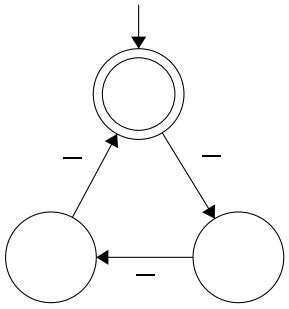
$$S \circ W = \{s \circ w : s \in S, w \in W\} \qquad S^R = \{s^R : s \in S\}$$

A *palindrome* is a string s for which $s^R = s$. If Σ is a set of strings, we shall let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

Chapter 5

Finite State Automata

Our initial model of computability is a computer with severely limited, finite amount of memory. Surprisingly, we shall still be able to compute a great many things. The idea of this model rests on explicitly representing memory as a finite amount of states, whose behaviour is uniquely determined by the state upon which it stands at a point of time. We can represent this process via a state diagram. Suppose we would like to describe an algorithm determining if a number is divisible by three. We shall represent a number by a string of dashes. For instance, $- - - - -$ represents the number 5. We describe the algorithm in a flow chart below



The algorithm proceeds as follows. Begin at the top node. we proceed clockwise around the triangle, moving one spot for each dash we see. If, at the end of our algorithm, we end up back at the top node, then the number of dashes we have seen is divisible by three. The basic idea of the finite automata is to describe computation via these flow charts – we follow a string around a diagram, and if we end up at a 'accept state', then we accept the string. A mathematical model for this description is a finite state automaton.

A *deterministic finite state automaton* is a 5-tuple $(Q, \Sigma, \Delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is a start state, $F \subset Q$ are the accept states, and $\Delta : Q \times \Sigma \to Q$ is the transition function. A finite state machine 'works' exactly how our original algorithm worked. Draw a directed graph whose nodes are states, and draw an edge between a state q and each related state $\Delta(q, \sigma)$, for each symbol $\sigma \in \Sigma$. Take a string $s \in \Sigma^*$. We begin at the start state q_0. Sequentially, for each symbol in s, we follow the directed edge from our current state to the state on the edge related to the current symbol in s. If, we end at an accept state in F, then s is an 'accepted' string. Formally, we define this method by extending Δ to $Q \times \Sigma^*$. We define, for $s \in \Sigma^*$, $t \in \Sigma$,

$$\Delta(q, \varepsilon) = q \quad \Delta(q, st) = \Delta(\Delta(q, s), t)$$

A state machine M *accepts* a string s in Σ^* if $\Delta(s, q_0) \in F$. We call the set of all accepting strings the *language* of M, and denote the set as $L(M)$. A subset of Σ^* which is a language of a deterministic finite state automata is known as a *regular language*.

Example. *Consider $\Sigma = \{-\}$. Then the set of all 'dashes divisible by three' is regular, as in the introductory diagram. Formally, take*

$$Q = \mathbb{Z}_3 \quad \Delta(x, -) = x + 1 \quad q_0 = 0 \quad F = \{0\}$$

then $(Q, \Sigma, \Delta, q_0, F)$ recognizes dashes divisible by three. The 'graph' of the automata is exactly the graph we've already drawn.

Arithmetic is closed under certain operations. Given two numbers, we can add them, subtract them and multiplication, and what results is still a number. In the theory of computation, the operations have a different flavour, but are nonetheless just as important. We shall find that all regular languages can be described from very basic languages under certain compositional operators, under which the set of regular languages is closed.

Theorem 5.1. *If $A, B \subset \Sigma^*$ are regular languages, then $A \cup B$ is regular.*

Proof. let $M = (Q, \Sigma, \Delta, q_0, F)$ and $N = (R, \Sigma, \Gamma, r_0, G)$ be automata recognizing A and B respectively. We shall define a finite automata recognizing $A \cup B$. Define a function $(\Delta \times \Gamma) : (Q \times R) \times \Sigma \to (Q \times R)$, by letting

$$(\Delta \times \Gamma)(q, r, \sigma) = (\Delta(q, \sigma), \Gamma(r, \sigma))$$

Consider

$$H = \{(q, r) \in S : q \in F \text{ or } r \in G\}$$

We contend that

$$((Q \times R), \Sigma, \Delta \times \Gamma, (q_0, r_0), H)$$

recognizes $A \cup B$. By induction, one verifies that for any $s \in \Sigma^*$,

$$(\Delta \times \Gamma)(q, r, s) = (\Delta(q, s), \Gamma(r, s))$$

Thus $(\Delta \times \Gamma)(q_0, r_0, s) \in H$ if and only if $\Delta(q_0, s) \in F$ or $\Gamma(r_0, s) \in G$. □

Theorem 5.2. *If A is a regular language, then A^c is regular.*

Proof. If $M = (Q, \Sigma, \Delta, q_0, F)$ recognizes A. Then define a new machine $N = (Q, \Sigma, \Delta, q_0, F^c)$. The transition $\Delta(q_0, s)$ is in F^c if and only if $\Delta(q_0, s)$ is not in F. □

Corollary 5.3. *If A and B are regular languages, then $A \cap B$ are regular.*

Proof. $A \cap B = (A^c \cup B^c)^c$. □

5.1 Non Deterministic Automata

An important concept in computability theory is the introduction of non-determinism. Deterministic machines must follows a set protocol when understanding input. Non deterministic machines can execute one of many different specified protocols. If any of the protocols accepts the input, then the entire machine accepts the input. Thus non-deterministic machines are said to multitask, for they can be seen to run every protocol specified at once, checking one of a great many protocols to see a pass. An alternative viewpoint is that the machines make a lucky guess – they always seem to choose the write protocol which results in an accepted string.

A *non-deterministic finite state automaton* is a 5-tuple $(Q, \Sigma, \Delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, q_0 is the start state, $F \subset Q$ are the accept states, and $\Delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the non-deterministic transition function.

In a non-deterministic finite state automata, we *accept* a string s if $s = s_1 \dots s_n$, where each $s_i \in \Sigma_\varepsilon$, and there are a sequence of states $t_0, \dots, t_{n+1}$, with $q_0 = t_0$ and $t_n \in F$, such that $t_{k+1} \in \Delta(t_k, s_k)$. The set of accepting strings of a machine M form the language $L(M)$. We draw a graph with nodes Q, and with directed edges v to w if $w \in \Delta(v, \Sigma_\varepsilon)$. We begin at q_0. For a string s, we attempt to find a path

from q_0 to an accept state, by following edges whose corresponding symbol is in s (or whose symbol is ε, in which we get for free). The string is accepted if such a path is possible. Some call non-deterministic methods a lucky guess methods, since they always make a lucky guess of which deterministic path to take to accept a string.

There is a nicer criterion of acceptance than described above, which is easier to work with in proofs. First, assume there are no ε-transitions in a machine M; that is, $\Delta(q, \varepsilon) = \varnothing$ for all states q. We may then extend $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ to $\Delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$ recursively by

$$\Delta(X, \varepsilon) = X \quad \Delta(X, st) = \Delta(\Delta(X, s), t)$$

where $\Delta(X, s) = \{\Delta(x, s) : x \in X\}$. A string s is accepted by M if and only if an accept state is an element of $\Delta(q_0, s)$. If $s = s_1 \dots s_n$, and there are $t_0, \dots, t_n$ with $t_0 = q_0$, t_n an accept state, and $t_{k+1} \in \Delta(t_k, s_k)$, then by induction one verifies that $t_n \in \Delta(q_0, s)$. Conversely, an induction on s verifies that if $q \in \Delta(q_0, s)$, and if $s = s_1 \dots s_n$, then there is a state q' with $q' \in \Delta(q_0, s_1 \dots s_{n-1})$, $q \in \Delta(q', s_n)$. But this implies that if there is an accept state in $\Delta(q_0, s)$, then s is accepted by M. We can always perform this trick, for we may always remove ε transitions.

Lemma 5.4. *Every non-deterministic automata is equivalent to a non deterministic automata without ε transitions, in the sense that they both recognize the same language.*

Proof. Consider a non-deterministic automata.

$$M = (Q, \Sigma, \Delta, q_0, F)$$

Define a state u to be ε reachable from t if there is a sequence of states $q_0, \dots, q_n$ with $q_0 = t$, $q_n = u$, and $q_{i+1} \in \Delta(q_i, \varepsilon)$. Let $E(u)$ be the set of all states ε reachable from u. Define

$$N = (Q, \Sigma, \Delta', q_0, F)$$

Where

$$\Delta'(q, s) = \begin{cases} \bigcup_{u \in E(q)} \Delta(u, s) & s \neq \varepsilon \\ \varnothing & s = \varepsilon \end{cases}$$

Then it is easily checked that $L(N) = L(M)$, for we may skip over ε transitions in N. □

It seems to be a much more complicated procedure to find if a string is accepted by a non-deterministic automata, but it turns out that every non-deterministic automata can be converted into a deterministic automata. The proof relies on the fact that we may exploit the operations of non-determinism, described using power sets of a set.

Theorem 5.5. *A non-deterministic finite state automata language is regular.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, F)$ be a non-deterministic automata. Assume M has no ε transitions, without loss of generality. Let

$$N = (\mathcal{P}(Q), \Sigma, \Gamma, \{q_0\}, \{S \in \mathcal{P}(Q) : S \cap F \neq \varnothing\}$$

where

$$\Gamma(S, t) = \Delta(S, t)$$

We have already verified this deterministic machine recognizes $L(M)$, for $\Delta(\{q_0\}, s)$ contains an accept state if and only if s is accepted. □

It is now fair game to use non-deterministic automata to understand regular languages, for the language of every non-deterministic automata is regular.

Theorem 5.6. *If A and B are regular languages, then $A \circ B$ is regular.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, F)$ and $N = (R, \Sigma, \Gamma, r_0, G)$ be deterministic automata accepting A and B respectively. Without loss of generality, assume Q is disjoint from R. Consider the non-deterministic machine

$$O = (Q \cup R, \Sigma, \Pi, q_0, G)$$

Define

$$\Pi(s, t) = \begin{cases} \{\Delta(s, t)\} & s \in Q, \text{ and } t \neq \varepsilon \text{ or } s \notin F \\ \{\Delta(s, t), r_0\} & s \in F, t = \varepsilon \\ \{\Gamma(s, t)\} & s \in R \\ \varnothing & \text{otherwise} \end{cases}$$

We quickly verify that if $s \in L(M)$ and $w \in L(N)$, then $sw \in L(O)$. If $v \in L(O)$, there is some substring k such that $r_0 \in \Pi(q_0, k)$ (for this is the only path from q_0 to G). If we choose k to be the shortest such string, then k must also be an accept string in $L(M)$, since any substring of k cannot map to states in R, and thus k must map via the ε transition from an accept state of M. If $v = kr$, then $\Pi(r_0, r)$ is an accept state in N, so $r \in L(N)$. Thus $v \in L(M) \circ L(N)$. □

Theorem 5.7. *If A is a regular language, A^* is regular.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, F)$ be a deterministic language which accepts A. Form a non-deterministic automata $N = (Q \cup \{i\}, \Sigma, \Gamma, i, F)$, where

$$\Gamma(q,s) = \begin{cases} \{\Delta(q,s)\} & s \neq \varepsilon \\ \{q_0\} & s = \varepsilon, q = i \\ \{i\} & s = \varepsilon, q \in F \\ \varnothing & \text{otherwise} \end{cases}$$

Then $L(N) = A^*$, for if $s = a_1 \dots a_n$, with $a_i \in A$, then we may go from i to q_0, then to an accept state via a_1, return to i with a ε transition, and continue, so s is accepted. If we split a string accepted to $L(N)$ into when ε transitions to i are used, then we obtain strings in A. □

In turns out that the operations of concatenation, union, and 'kleene starification' are enough to describe all regular languages. Thus we may take an algebraic approach to understanding regular languages, using the symbology of regular expressions.

5.2 Regular Expressions

Automata are equivalent to a much more classical notion of computation – regular expressions.

Definition 5.1. *A* regular expression *over an alphabet Σ is the smallest subset of $(\Sigma \cup \{\varnothing, \cup, *, (,)\})^*$ such that*

1. *$\varnothing$ is a regular expression, as are $s \in \Sigma^*$.*
2. *If λ and γ are regular expressions, then so are λ^*, $(\lambda \cup \gamma)$, and $(\lambda \circ \gamma)$.*

Every regular expression describes a language. A string $s \in \Sigma^*$ is recognized by an regular expression λ if

1. $\lambda \in \Sigma^*$, and $s = \lambda$.
2. $\lambda = (\lambda \cup \gamma)$, and s is recognized by λ or by γ.
3. $\lambda = \gamma \circ \delta$, and $s = wl$, where w is recognized by γ, and δ recognizes l.

4. $\lambda = \gamma^*$, and $s = \varepsilon$ or $s = s_1 \dots s_n$, where s_i is recognized by γ.

The regular language corresponding to a regular expression λ is $L(\lambda)$, the set of strings recognized by t. The set of all regular expressions on an alphabet Σ will be denoted $R(\Sigma)$.

It is a simple consequence of our discourse that every language recognized by a regular expression *is actually* a regular language. In fact, we can show that every regular language is described by a regular expression. We shall describe an algorithm for converting a finite state automaton to a regular expression. We shall make a temporary generalization, by allowing non deterministic finite automata to have regular expressions in their transition functions. A *generalized non-deterministic finite state automaton* is a 5-tuple $M = (Q, \Sigma, \Delta, q_0, f_0)$, where $\Delta : Q - \{f_0\} \times Q - \{q_0\} \to R(\Sigma)$ is the generalized transition function, and q_0 is the start state, f_0 is the end state. A generalized automaton accepts $s \in \Sigma^*$ if we may write $s = s_1 \dots s_n$, and there are a sequence of states $q_0, \dots, q_n$ where $q_n = f_0$, and $\Delta(q_i, q_{i+1})$ recognizes s_i.

Theorem 5.8. *Any generalized finite-state automaton describes the language of a regular expression.*

Proof. If the generalized automaton has two states, then there is only one transition from start state to begin state, and this transition describes a regular expression for the automaton. We will reduce every automaton to this form by induction. Suppose an automaton has n states. Fix a state $q \in Q$, which is neither the beginning or accepting state. Define a new automaton

$$N = (Q - \{q\}, \Sigma, \Delta', q_0, f_0)$$

where $\Delta'(a, b) = (\Delta(a, b) \cup \Delta(a, q)\Delta(q, q)^*\Delta(q, b))$ Then N is equivalent to M, and has one fewer state, and is thus equivalent to a regular expression. □

Corollary 5.9. *Every regular language is described by a regular expression.*

Proof. Clearly, every DFA and NFA is equivalent to a generalized NFA, for by adding new states, we may ensure no states map back to the start state, and that there is only one end state. □

5.3 Limitations of Finite Automata

We've discovered a menagerie of different problems we can solve with finite automata, but it has already been foreshadowed that better machines await. Here we

discover methods which attack languages, showing that they cannot be recognized by regular expressions, not finite automata.

Theorem 5.10 (Pumping Lemma)**.** *Let L be a regular language. Then there is a number p, called the pumping length, such that any $s \in L$ which satisfies $|s| \geqslant p$, then we may write $s = wuv$, where $|u| > 0$, $|wu| \leqslant p$, and $wu^i v \in L$ for all $i \geqslant 0$.*

Proof. Let L be a regular language, and M a deterministic automata recognizing L with p states. Let s be a string with $|s| \geqslant p$, with $s \in L(M)$. Write $s = s_1 \dots s_n$ and let $q_k = \Delta(q_0, s_1, \dots, s_k)$. Then we obtain $|s| + 1$ states $q_0, q_1, \dots, q_{|s|}$. By the pidgeonhole principle, since q_i equals some q_j, for $i < j$. Let $w = s_1 \dots s_i$, $u = s_{i+1} \dots s_j$, $v = s_{j+1} \dots s_{|s|}$. Then $\Delta(\Delta(q_0, w), u) = \Delta(q_0, w)$, so

$$\Delta(q_0, wu^i v) = \Delta(q_0, wuv)$$

So $wu^i v \in L$ for all i. □

Example. *$L = \{0^k 10^k : k \geqslant 0\}$ is not regular. If it was regular, it would have a pumping length p. Since $0^p 10^p$ is in L, so we may write $0^p 10^p = wuv$, where $|wu| \leqslant p$, and $wu^i v \in L$. Then $u = 0^k$, for $k > 0$, and $wv = 0^{p-k} 10^p \in L$, which is clearly not in the language, contradicting regularity.*

Example. *$L = \{1^{n^2} : n \in \mathbb{N}\}$ is not regular. Suppose we had a pumping length p. Then $1^{p^2} \in L$, so there is $0 < k \leqslant p$ with $1^{p^2+k} \in L$. But*

$$(p+1)^2 = p^2 + 2p + 1 > p^2 + k$$

] And there is no perfect square between p^2 and $(p+1)^2$, a contradiction.

Example. *The language $L = \{0^i 1^j : i > j\}$ is not regular. If we had a pumping lenght p, then $0^p 1^{p-1} \in L$. But then there is $0 < k \leqslant p$ such that $0^{p+(i-1)k} 1^{p-1} \in L$ for all natural numbers i. In particular, for $i = 0$, we find $0^{p-k} 1^{p-1} \in L$ But $p - k \leqslant p - 1$, a contradiction.*

There is a much more mathematically elegant and complete way of separating regular languages from non-regular ones, discovered by John Myhill and Anil Nerode. Consider an alphabet Σ, and a particular language $L \subset \Sigma^*$. Call two strings a and b in Σ^* L-indistinguishable if $az \in L$ if and only if $bz \in L$ for any $z \in \Sigma^*$. This forms an equivalence relation on Σ^*. We shall define the index of L, denoted Ind L, to be the cardinality of the partition.

Theorem 5.11 (Myhill-Nerode)**.** *L is regular if and only if Ind $L < \infty$, and Ind L is the number of states of the smallest finite state machine to recognize L.*

Proof. Let M be a deterministic finite state machine recognizing L with n states, transition function Δ, and start state q_0. Let $a_1, \ldots, a_{n+1}$ be $n+1$ strings in Σ^*. We claim that at least one pair is indistinguishable. if we take $q_i = \Delta(q_0, a_i)$, then some $q_i = q_j$. These two strings are then indistinguishable in L. Conversely, suppose that Ind $L < \infty$. Let $A_1, \ldots, A_n$ be the equivalence classes of Σ^*. If $s \in A_i$ is contained in L, then every other $w \in A_i$ is in L, for otherwise s and w can be distinguished. Build a finite state machine whose states are A_i, whose start state is $[\varepsilon]$, and whose transition function is

$$\Delta([s], t) = [st]$$

$[s]$ is accepted if and only if $s \in L$. This finite state machine recognizes s. □

In some circumstances, the Myhill Nerode theorem is very powerful.

Example. *For $\Sigma = \{\mathsf{a}, \mathsf{b}, \ldots, \mathsf{z}\}$, consider the set L of words w whose last letter has not appeared before. For example, the words* apple*,* google*,* k*, and ε are in L, but the words* potato *and* nutrition *are not. Is this language regular? We apply Myhill Nerode. If the letters in one word are different to the letters in another word, these words are distinguishable. If the letters in one word are the same as the letters in another word, and both are not accepted or both are not accepted, these words are indistinguishable. Thus the index of the language in consideration is the same as the number of different subsets of the set of letters in a word, counted twice for repeated and non-repeated characters. Subtracting one from the fact that ε need only be counted once, we find that $2 \cdot 2^{26} - 1 = 2^{27} - 1$. Since this is finite, the language is regular, and this is the minimal number of sets in a finite state machine recognizing the language.*

5.4 Function representation

Decision problems are sufficient to model a large variety of computational models, but for completeness, we should at least mention how we determine whether problems with multiple outputs can be. The counterpart of a finite-state automata is a finite-state transducer. A *finite state transducer* is a 5 tuple $(Q, \Sigma, \Lambda, \Delta, \Gamma, q_0)$, where Q is the set of states, Σ is the input alphabet, Λ is the output alphabet, $\Delta : Q \times \Sigma \to Q$ is the transition function, $\Gamma : Q \times \Sigma \to \Lambda_\varepsilon$ is the transduction

function, and q_0 is the start state. Each finite-state transducer M gives rise to a function $f_M : \Sigma^* \to \Lambda^*$, defined by

$$f_M(\varepsilon) = \varepsilon \quad f_M(st) = f_M(s) \circ \Gamma(\Delta(q_0, s), t)$$

A *regular function* is one computable by a finite state transducer.

Example. *Consider an alphabet* $\mathbf{F}_2$ *and consider the function* f *taking and returning strings over* $\mathbf{F}_2$*, inverting the strings on even positions, and leaving the strings on the odd positions. Then* f *is a regular function, for it may be computed by the automata below.*

Chapter 6

Context Free Languages

Finite state machines are useful on a specific set of problems, but have limitations. We would like our notion of computability to decide on a much more complicated set of problems. Thus we must define new classes of computable languages.

6.1 Context Free Grammars

Definition 6.1. *A* Context Free Grammar *is* (V, Σ, R, S)*, where* V *is a set of variables,* Σ *is a character set, disjoint from* V*,* R *is a relation between* V *and* $(V \cup \Sigma)^*$*, and* $S \in V$ *is the start variable. We call elements of* R *derivation rules, and write* $(a, s) \in R$ *as* $a \to s$*.*

A string of the form uvw is *directly derivable* from uAw if $A \to v$ is a derivation rule. The smallest transitive relation containing the 'directly derivable' relation is the derivability relation. To reiterate, a string u is *derivable* from w if there is a sequence of direct derivations

$$w \to s_0 \to \cdots \to s_n \to u$$

Such a sequence is known as a *derivation*. The language of a grammar is the set of all strings in Σ^* derivable from the start state S. As in finite automata, the language of a grammar G will be denoted $L(G)$. A language is *context-free* if it is the language of a context free grammar.

Lemma 6.1. *Let* $G = (V, \Sigma, R, S)$ *and* $H = (V, \Sigma, R', S)$ *be two different context free languages. If every* $(v, s) \in R'$ *is a derivation in* G*, then* $L(H) \subset L(G)$*.*

Proof. If we can show that every direct derivation in H is a derivation in G, then all derivations in H are derivations in G, since derivations form the smallest transitive relation containing the direct derivations in H, and the derivations in G certainly satisfy this. If w is directly derived from s in H, then there is a rule (B, u), $s = rBt$, and $w = rut$. There is a sequence $s_0 \dots s_n$ deriving u from B in G. But then $(rs_0t) \dots (rs_nt)$ is a derivation of w from s, so w is derivable from s in G. □

A leftmost derivation is a derivation which always swaps out the leftmost variable. A string is *ambiguous* if it has two different leftmost derivations. A grammar is ambiguous if its language contains an ambiguous string. Ambiguity is unfortunate when parsing a language, since it means we may be able to interpret elements of the language in two different ways.

Example. *First order logic can be defined as an unambiguous grammar. To develop the language, we took a bottom up approach, but a top up approach can also be taken. We must take a finite alphabet to develop the language, which we take to be*

$$\{(,), \forall, \exists, \neg, \wedge, \vee, \Rightarrow, x, f, P, 0, 1, \dots, 9\}$$

we cannot take an 'infinite number of variables', in the sense of an infinite number of symbols, for then formal language theory does not apply. We must instead assume our variables, predicates, and functions are themselves words in a finite alphabet. For instance, we will enumerate our variables

$$\Lambda = \{x, x_0, x_{00}, \dots, x_{0000000000}, \dots\}$$

The trick to forming functions and predicates is to put n ones after an f or a P to denote that is is an n-ary function, so

$$\mathcal{F}^n = \{f^{111\dots11}, f^{111\dots11}{}_0, \dots, f^{111\dots11}{}_{000000000}, \dots\}$$
$$\mathcal{P}^n = \{P^{1111\dots11}, \dots\}$$

Then we may form predicate logic as a context free grammar. The variables are easiest to form

$$X \to x \mid X_0$$

Terms are tricky to define because we must form functions as well. The trick is to introduce new variables Y and Z which add enough terms to the function.

$$T \to X \mid f^1U)$$
$$U \to V(T \mid {}^1U, T$$
$$V \to \varepsilon \mid V_0$$

Finally, we form the formulas of the calculus. Again, the only trick part are the atomic formulae

$$F \to (F \wedge F) \mid (F \vee F) \mid (\neg F) \mid (F \Rightarrow F) \mid (\forall x : F) \mid (\exists x : F) \mid P^1 Z)$$
$$Y \to Z(T \mid {}^1 Y, T$$
$$Z \to \varepsilon \mid Z_0$$

We showed the formulas are derived unambiguously, but this took a lot of hard work. It is impossible to find a general procedure to decide whether a language is ambiguous, which is what makes verifying ambiguity so difficult.

It is useful to put grammars in a simple form for advanced theorems. A grammar (V, Σ, R, S) is in *Chomsky Normal Form* if the only relations in R are of the form

$$S \to \varepsilon \quad A \to BC \quad A \to a$$

where $A, B, C \in V$ and $B, C \neq S$, and $a \in \Sigma$.

Theorem 6.2. *Every context-free language can be recognized by a context-free grammar in Chomsky normal form.*

Proof. We shall reduce any context free grammar $G = (V, \Sigma, R, S)$ to a context free grammar in normal form in a systematic fashion, adding each restriction one at a time.

1. *No derivation rules map to the start variable*:
   Create a new start variable mapping onto the old start variable.

2. *There are no ε-transition rules except from the start variable*:
   Define a variable $A \in V$ to be nullable if we may derive ε from A. Let W be the set of all nullable variables. Define a new language $G'(V, \Sigma, R', S)$ such that, if $A \to A_1 \ldots A_n$ is a derivation rule in G, and $A_{i_1}, \ldots, A_{i_m}$ are nullable, then we add 2^m new rules to G' by removing some subset of the A_{i_k}. Then $L(G') = L(G) - \{\varepsilon\}$, so that if $\varepsilon \in L(G)$, we need only add an ε rule to S to make the two languages equal.

   We will prove that if A is a variable in G', then A can derive $w \in \Sigma^*$ in G' if and only if it can derive it in G and $w \neq \varepsilon$. One way is trivial, the other a proof by induction on the length of the derivation. Suppose we have a derivation in G

   $$A \to s_0 \to \cdots \to s_n \to w$$

Let $s_0 = A_1 \dots A_n$. Then each A_i derives w_i in G, where $w = w_1 \dots w_n$. We can choose such a derivation to be shorter than the derivation of G. But this implies that A_i derives w_i in G', provided $w_i \neq \varepsilon$. Let $w_{i_1} \dots w_{i_m} \neq \varepsilon$. Then $m \neq 0$, since $w \neq \varepsilon$. We have a corresponding production rule $A \to A_{i_1} \dots A_{i_n}$ in G', since the other variables are nullable. Thus, by induction, A can derive w.

3. *There are no derivation rules $A \to B$, where B is a variable*:
   Call B unitarily derivable from A if there are a sequence of derivation rules

   $$A \to V_1 \to \cdots \to V_n \to B$$

   Define a new grammar $G' = (V, \Sigma, R', S)$. If B is directly derivable from A, and B has a derivation rule $B \to s$, then G' has a production rule $A \to s$, provided that s is not a variable. Then G' has no rules of the form $A \to B$, and generates the same language. This is fairly clear, and left to the reader to prove.

4. *Every rule is of the form $A \to AB$ or $A \to a$*:
   If we have a rule in G of the form $A \to s$, where $s = s_1 \dots s_n$, and $s_{i_1}, \dots, s_{i_m} \in \Sigma$, then add new unique variables $A_{s_{i_k}}$ for each $s_i \in \Sigma$, and replace the rule with new rules of the form

   $$A \to s_1 \dots A_{s_{i_1}} \dots A_{s_{i_m}} \dots s_n$$

   $$A_{s_{i_m}} \to s_{i_m}$$

   Thus we may assume every rule of the form $A \to A_1 \dots A_n$ (where we may assume $n \geqslant 2$) only maps to new variables. But then we may add new variables $V_2 \dots V_{n-1}$, and swap this rule with rules of the form

   $$A \to V_{n-1} A_n$$

   $$V_k \to V_{k-1} A_k$$

   $$V_2 \to A_1 A_2$$

Now every derivation rule is in the correct form, and we have reduced every grammar to Chomsky normal form. □

Chomsky normal form allows us to prove a CFG pumping lemma.

Theorem 6.3. *If L is a context free language, then there is $p > 0$, such that if $s \in L$, and $|s| \geqslant p$, then we may write $s = uvwxy$, where $|vwx| \leqslant p$, $v, x \neq \varepsilon$, and for all $i \geqslant 0$, $uv^i wx^i y \in L$.*

Proof. Let A be the language of the grammar G, which we assume to be in Chomsky normal form. Let there be v variables in G. If the parse tree of $s \in A$ has height k, then $|s| \leqslant 2^{k-1}$, which follows because the tree branches in two except at roots, so there is at most 2^{k-1} roots. If $|s| \geqslant 2^{3v}$, then every parse tree of s has height greater than v. Pick a particular parse tree of smallest size. There is a sequence of variables $A_1, A_2, \ldots, A_{3v}$, such that A_i is the parent of A_{i+1}. Because of how many variables there are, some variable must occur at least 3 times (for otherwise we may remove the variables in pairs, to conclude that $3v - 2v = v$ variables contain no variables, a contradiction). What's more, they much occur within a height of $3v$ of each other. Let $A_i = A_j = A_k$, for $i < j < k$. Let A_i produce aA_jb, let A_j produce xA_ky, and let A_k produce r. Write $s = maxrybn$. By virtue of the minimality of the tree, we may assume that a or b is nonempty, and one of x or y is nonempty. First, if both ax and yb are assumed non-empty, then we may pump these strings up, and have proved our lemma. So suppose ax is empty. Then b and y are nonempty, and $s = mrybn$. Since A_i produces A_jb, and A_j produces A_ky, A_i may be pumped to produce A_jb^i, A_j may be pumped to produce A_ky^i, and mry^ib^in is in the context free language, so we have non-empty strings to pump. The proof is simlar if by is empty, for then a and x are non-empty. The constraint $|vxy| \leqslant k$ is satisfies for yb and ax, for the A_i and A_j lie within $3v$ of each, so the string produces in this production is at most as long as 2^{3v}, which is less than or equal to the pumping length. □

6.2 Pushdown Automata

Regular languages have representations as the languages of regular expressions or as finite automata. Context-free languages also have dual representations, as 'machines' or as abstraction operations. It is good to represent a language as a machine for it may hint as to what hardware capabilities a computer must have to be able to solve problems related to the languages. The key machine component for a context-free language is a stack. A pushdown automata is a finite state automata with the addition of a stack.

Definition 6.2. *A* (non-deterministic) pushdown automata *is a tuple* $(Q, \Sigma, \Lambda, \Delta, q_0, F)$, *where Q is a finite set of states, Σ is an alphabet, Λ is the stack alphabet, Δ :*

$Q \times \Sigma_\varepsilon \times \Lambda_\varepsilon \to \mathcal{P}(Q \times \Lambda_\varepsilon)$ is the state transition function, $q_0 \in Q$ is the start state, and $F \subset Q$ are the accept states.

It turns out that deterministic pushdown automata are less powerful than non-deterministic automata, so we do not discuss deterministic automata. It is interesting to note that the languages of deterministic automata are connected to unambiguous grammars, though we will not have time to discuss this further.

Let us describe a pushdown automata intuitively. The automata has a stack of symbols from Λ, which it can push and pull from when deciding how to move through the machine. A stack is a string in Λ^*. Thus, formally, a string s is *accepted* by a push-down automata M if there are a sequence of states $q_0, \ldots, q_n$, and stacks $w_0, \ldots, w_n \in \Lambda^*$ such that $q_n \in F$, $w_0 = \varepsilon$, and if we write $w_i = w\lambda$, with $\lambda \in \Lambda_\varepsilon$, then $w_{i+1} = w_i\lambda'$, with $(q_{i+1}, \lambda') \in \Delta(q_i, \lambda)$. Thus we pop and pull off the rightmost character in the string when moving between states.

Pushdown automata have enough versatile memory to recognize context free languages. The stack can 'remember' variables it has yet to parse, and check when symbols are used. We shall allow a mild generalization of pushdown automata, which can push multiple symbols to the stack at a time. This is fine, without loss of generality, because we could have instead introduced new states that take nothing from the stack, and push the symbols on one at a time. We shall also assume a pushdown automata starts with a \$ symbol at the bottom of its stack, which is fine, because we could have added another start state to the automata which pushes the \$ on as we begin running the machine.

Theorem 6.4. *Every context free language is accepted by a pushdown automata.*

Proof. Consider a context free language (V, Σ, R, S). Consider a pushdown automata $(Q, \Sigma, \Lambda, \Delta, q_0, F)$, with the stack language $\Lambda = \Sigma \cup V$, and Q just two states q_0 and f_0. For each derivation $A \to s \in R$ we have

$$(q_0, s) \in \Delta(q_0, \varepsilon, A)$$

And for each $a \in \Sigma$, we have

$$(q_0, \varepsilon) \in \Delta(q_0, a, a)$$

And a finale transition

$$(f_0, \varepsilon) \in \Delta(q_0, \varepsilon, \$)$$

It is clear that this automata parses the context free language. □

A converse also holds, so that pushdown automata are equivalent computers of context free languages. To do this, we assume that the automata pushes everything off its stack before it finishes, and has a single accept state f_0. In addition, we shall assume that a state only pops and pulls in one action, and doesn't do both at the same time. Adding additional states means this is no loss of generality.

Theorem 6.5. *Each pushdown automata language is context free.*

Proof. The gist of our approach is as follows. Let $(Q, \Sigma, \Gamma, \delta, q_0, \{f_0\})$ be a pushdown automata. We shall define a context grammar with variables A_{pq}, with $p, q \in Q$. This variable should be able to generate all possible strings which can start in p with an empty stack, and end up in q with an empty stack. Our start variable will then be A_{q_0,f_0}. The first rules are most basic

$$A_{pp} \to \varepsilon$$

Such a path may end up empty halfway through the path, so we have these rules, for each $p, q, r \in Q$,

$$A_{pq} = A_{pr}A_{rq}$$

We can also pop something on the stack, and save it for a long time later. If $t \in \Sigma$, and $(r, t) \in \Delta(p, a, \varepsilon)$, and $(q, \varepsilon) \in \Delta(s, b, t)$, then we add the derivation rule

$$A_{pq} = aA_{r,s}b$$

We claim these rules describe all possible derivations we could make in the pushdown automata. It is clear that all such derivations in this context language are accepted in the pushdown automata.

We shall prove that if we can move from p to q using a string x, both with an empty stack, then $A_{pq} \to x$. This is done by induction on the number of steps to accept the string in the automata. If we do this in one step, then the string is empty, and we have a rule $(q, \varepsilon) \in \Delta(p, \varepsilon)$, or the string consists of a single letter, and we have a rule $(q, \varepsilon) \in \Delta(p, t)$. In the first case, we have a derivation $A_{p,q} \to \varepsilon$, and in the second, we have a derivation

$$A_{p,q} \to tA_{q,q}\varepsilon \to t\varepsilon\varepsilon = t$$

Now consider a machine that runs for a length n. Suppose the stack empties at some state k, after running through x_1 of the string, for $x = x_1x_2$. Then we have, by induction, a derivation

$$A_{pq} \to A_{pk}A_{kq} \to x_1x_2$$

Thus we may assume that the stack never empties except at beginning and end. Then the first action must be to push a symbol t to the stack, and the last action to remove t. If we move from p to r in the first action by reading a, and from s to q in the last action by reading b, then we may write $x = acb$, and by induction, we have the derivation

$$A_{pq} \to aA_{rs}b \to acb = x$$

Thus we have verified the equivalence of the pushdown automata and context free language. □

Pushdown automata are easy to connect to their finite state cousins.

Corollary 6.6. *Every regular language is context free.*

Chapter 7

Turing Machines and Uncomputability

Finite state machines are good at modelling machines with a small amount of memory, and pushdown automata are good at modelling stack processes. Nonetheless, there are many problems that 'should be computable', since we can solve them which these automata cannot solve. In this chapter, we introduce a much more robust model, the *Turing machine*, which satisfies this criteria. Every known algorithm can be implemented in this model.

The basic idea is that a Turing machine runs off of an infinite tape, which the machine can scan through, swerving left and right to read off the tape. The tape begins with the input

$$q_0 x_1 x_2 \dots x_n$$

A notation which implies that the machine is in state q_0, and the tape head is looking at x_1, and the tape consists of the letters $x_1, x_2 \dots, x_n$, and then the rest of the tape consists of blank spaces.

The Turing machine has finitely many states, which describe how the machine reacts when it sees a current character – it chooses to swap the current character with a different character, and moving either left or right one space. We may also choose to accept the string or reject the string at any time.

Formally, a turing machine can be described as a tuple

$$(Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

where Q is a set of states, Σ is an input alphabet, Γ is a tape alphabet (which we assume includes the blank space $-$), $q_0 \in Q$ is the start state, $q_{\text{accept}} \in Q$ is the

accept state, and $q_{\text{reject}} \in Q$ is the reject state, and $\Delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$. We interpret this as given a state and a currently read letter, we move to a new state, we replace the letter with a new character, and we move left or right.

To compute the operation of a Turing machine on a string, we recursively define a computation on a string. A specific state of the machine will be represented by a string sqw, where $q \in Q$, $s, w \in \Gamma^*$. We say that *sqtw yields suq′w* if $(q', u, R) = \Delta(q, t)$.

Chapter 8

Complexity Theory

We have now discovered what it means to solve a problem algorithmically. We formalize the problem into a certain language over an alphabet, then find a Turing machine over that alphabet which terminates on every input, and accepts exactly theres strings that are an element of a language. But just because a problem is solvable, does not mean that it is *feasibly solvable*; there may be an algorithm which will eventually solve a problem, but if it takes centuries to find the answer, the solution is not effective. In this section, we restrict our attention to the computable problems, and describe the degree of difficulty of certain problems. First, we must find a measure of a problem's difficulty, so that we can classify problems based on how difficult they are.

8.1 Measuring Complexity

Let M be a turing machine over an alphabet Σ, which halts on all inputs. Then, for each string $s \in \Sigma^*$, M deterministically executes a certain number of steps, eventually terminating. Define $t_M(s) \in \mathbb{N}$ to be the number of configurations M steps through before it halts on input s. The *time complexity* of M is the function

$$f_M(n) = \max\{t_M(s) : s \in \Sigma^*, |s| = n\}$$

It could be argued that a better notion of complexity is the average running time over inputs of a certain length, but we rarely know how often certain inputs will occur. Often, in practice, slow inputs occur much more often that fast inputs, so worst time analysis better reflects an algorithm's speed. Furthermore, worst

time analysis gives us an elegant and useful theory which gives meaningful results, which without this simplification would be impossible. We should not be dogmatic in this approach, because some algorithms do benefit from an average case analysis (for instance, the ellipsoid algorithm in linear programming), but the worst-case approach has turned out to be the most useful.

We rarely specify a turing machine exactly, and thus have difficulty expressing f_M as an exact number. Even if we described a machine exactly, f_M likely will not have a simple formula specifying an algorithms speed. Thus we apply the theory of asymptotics. Recall that a real-valued function f, the 'big O' set $O(f)$ consists of all functions g for which $|g| \leqslant c|f|$ *eventually* holds, where c is some positive constant. Similarily, the 'little o' set $o(f)$ consist of all functions g satisfying the sharper relation

$$\lim_{x\to\infty} \left|\frac{g(x)}{f(x)}\right| = 0$$

Less important to our needs, but canonically include are the $\Omega(f)$ and $\omega(f)$ sets, which consist of functions g such that $f \in O(g)$ and $f \in o(g)$ respectively. We define $\Theta(f) = \Omega(f) \cap O(f)$, the class of functions which are asymptotically equal to f.

Example. *Let $P \in \mathbb{R}[X]$ be a polynomial of degree m. For $x \geqslant 1$, and $i \leqslant j$, the inequality $x^i \leqslant x^j$ holds, so if P is expressed in coefficients as*

$$P = \sum c_i X^i$$

then for $x \geqslant 1$,

$$|P(x)| \leqslant \sum |c_i| x^i \leqslant \left(\sum |c_i|\right) x^m$$

and we have shown that $P \in O(x^m)$.

Example. *Since $O(f)$ is a subset of a space of functions, they can be manipulated under real-valued operations. As an example, we consider the set $2^{O(\log n)}$, which consists of all functions of the form 2^f, where $f \in O(\log n)$. We contend this is just the set of all functions g in $O(n^c)$ for some c. First note that if*

$$f(x) \leqslant c \log x$$

then because the exponential function is monotone, we obtain that

$$2^{f(x)} \leqslant 2^{c \log x} = x^c$$

so if the first inequality eventually holds, the second inequality eventually holds. Conversely, suppose $f \in O(x^c)$. Suppose that eventually

$$f(x) \leqslant Kx^c$$

Then if $x \geqslant e$,

$$\log_2 f(x) \leqslant \log_2 K + c \log_2 x \leqslant \frac{c + 2\log_2 K}{\log 2} \log x$$

which implies $\log_2 f \in O(\log n)$, and $f = 2^{\log_2 f} \in 2^{O(\log n)}$.

Example. *A commonly used relation is that $O(f) + O(g) = O(f + g)$. Let h and k be functions with a constant K and n_0, such that*

$$h \leqslant Kf \quad k \leqslant Kg$$

eventually holds. Then

$$h + k \leqslant K(f + g)$$

eventually holds as well, so $h + k \in O(f + g)$. Conversely, suppose $h \in O(f + g)$, and let

$$h \leqslant K(f + g)$$

Define

$$h_1 = \min(h, Kf) \quad h_2 = h - h_1$$

Then $h_1 + h_2 = h$. It is easy to see $h_1 \in O(f)$. Now if $n \geqslant n_0$, then

$$h(n) - h_1(n) \leqslant Kf + Kg - h_1(n) \leqslant Kg$$

Thus $h_2 \in O(g)$, and we have verified the equality. Similar arguments show

$$O(fg) = O(f)O(g)$$

Certainly, a machine M with $f_M(n) = n^2$ runs much faster than a machine N with $f_N(n) = 2^8 n^2$, and certainly a machine L with $f_L(n) = 2^n$ runs faster than f_N for the first 20 input sizes, but regardless of the constant, the exponential machine will quickly become much slower even if we choose a much larger constant ($2^{100}n^2$ becomes smaller than 2^n only after the input size increases to 125). What's more, we have a theorem that, by constructing a new machine, we can always decrease the constant in an algorithm, with at most a linear increase in time.

Theorem 8.1. *For any two-tape Turing machine M, and $\varepsilon > 0$, there is an equivalent two-tape machine N such that*

$$f_N(x) \leqslant \varepsilon f_M(x) + (1+\varepsilon)n$$

eventually holds.

Proof. Consider the following strategy. Let M have tape alphabet Σ and tape alphabet $\Sigma \cup \Sigma^n$. Construct N with tape alphabet $\Sigma \cup \Sigma^m$, and with states $Q \times \Sigma^m \cup Q'$. Our machine takes the first m characters $w_1, \ldots, w_m$ in input, and places $(w_1, w_2, \ldots, w_m) \in \Sigma^m$ on the second tape. This takes m steps. If the input does not contain m strings, we assume that we write in whitespace instead. Continue this until we reach whitespace. After $m\lceil n/m \rceil$ steps, we have a condensed input. After this preprocessing, we can execute m steps of the original machine in 6 steps of our algorithm. Our state in N will now be of the form (q,t), where q is a state in M, and t is the current position in the tuple $(w_1, \ldots, w_m)$ we are pointing at in the second string that the simulated machine should be at. We move left once, and then right twice, in order to see the states to the left and right of our current states. Given this information, we can predict whether we will end up on the left or right after m steps of the simulated machines, and what the other states will look like. Only states in the current tuple, and the left or right tuple (but not both) will be changed. Based on our computation, we move left or right, changing the states we are currently in, and, if needed, move again to change the states in our new tuple. Thus in six steps, we have simulated m steps. If M takes k steps to halt on some input, N takes $m\lceil n/m \rceil + \lceil k/m \rceil$. For n large enough,

$$m\lceil n/m \rceil \leqslant (1+\varepsilon)n$$

which implies that eventually

$$f_N \leqslant \lceil f_M/m \rceil + m\lceil n/m \rceil$$

If we choose m large enough, we obtain the inequality desired. □

If we only use one tape, then the increase becomes $\varepsilon f_M + 2n^2 + 2$. We also note that programs which run in linear time can only improved to $(1+\varepsilon)n$, for some ε. This makes sense, for if some program runs in time bounded by cn for $c < 1$, then for large inputs the program must eventually not even look at all the input, which is unfeasible in most problems.

Since we do not describe Turing machines formally, we rely on heuristic requirements to determine upper bounds for certain algorithms. Our intuition, guided by our knowledge of formal Turing machines, should carry us through, provided we describe algorithms in enough detail that the underlying Turing machine can be seen in enough detail.

We now have the formality to classify problems by how long it takes to compute them. Given a function g, define $\mathbf{TIME}(g)$ to be the class of all languages L which are recognized by a machine M, and $f_M \in O(g)$. Thus $\mathbf{TIME}(g)$ consists of all problems which can be computed 'roughly in time with g'. This is the first example of a *complexity class*, a set of languages characterized by how slow it takes to decide upon the language.

Example. *Consider the language* $A = \{0^k 1^k : k \geqslant 0\}$. *Determine a turing machine deciding the language from the algorithm*

1. *Scan across input, checking whether the input is of the form* $0^i 1^j$ *for some* i, j. *We ensure the string only contains 0s and 1s, and that the string contains no 0s after it contains 1s. Reject in any other circumstance. Afterwords, return to the beginning of the tape.*

2. *While there are still unmarked 0s and 1s on the tape, tick off a new 0 on the tape and a new 1. If there are no 0s but still 1s, or 0s but no 1s, reject the input.*

3. *We have ticked off one 0 for each 1, so there are the same number of zeroes as ones. Accept the input.*

We analyze the three steps individually, putting the asymptotics together once we're done. Step 1 takes roughly a constant number of steps for each element of the input, for we perform the same operation on each character. Thus the time to compute step 1 is in $O(n)$. *The number of times step 2 is executed is at most half the characters in the input, and each iteration is in* $O(n)$, *for the iteration involes moving from the beginning to the end of the tape a finite number of times. Thus step 2 is in*

$$O(n/2)O(n) = O(n)^2 = O(n^2)$$

Finally, step 3 is a constant time operation, and is therefore in $O(1)$. *Thus the entire algorithm is in*

$$O(n) + O(n^2) + O(1) = O(n^2 + 2n + 1) = O(n^2)$$

Thus $A \in \mathbf{TIME}(n^2)$. *A divide and conquer variation of this algorithm shows that* $A \in \mathbf{TIME}(n \log n)$, *left as an exercise.*

8.2 Models of Complexity

We have considered various variants of Turing machines. All turn out to decide the same class of languages, hence the adoption of the machine as an applicable model of real world computation. But we run into issues when applying this argument to complexity theory, for using a different model of turing machine may reduce the time complexity of an algorithm.

Example. *Consider the 0^k1^k problem. The best algorithm we found ran in $O(n \log n)$ time. But consider a 2-tape Turing machine, which first shifts all 1s in the input to a 2nd tape, then procedurally checks off 0s and 1s. This runs in $O(n)$ time, for we need only scan over the tape a constant number of times. Later, we shall show that the asymptotics of the 0^k1^k problem cannot be improved on a single tape machine, so multi-tape turing machines are asymptotically faster than one tape machines.*

We may take comfort in discovering that changing the model does not *drastically* affect the computation time of an algorithm. This is what this section sets out to address.

Theorem 8.2. *If a multi-tape turing machine has time complexity f, where $f \geqslant n$, then the multi-tape turing machine has an equivalent single-tape turing machine with time complexity in $O(f^2)$.*

Proof. We have already described a procedure which simulates a k-tape turing machine. We shall compute an asymptotic analysis of this simulation. Suppose the multitape turing machine takes $g(n)$ steps before terminating on a particular input of size n. Let us analyze each step

1. The algorithm first takes the input $w_1 \dots w_n$, and manipulates it into the form

   $$\#\dot{w}_1 \dots w_n \#\dot{\sqcup}\#\dot{\sqcup}\#\dot{\sqcup}\# \dots \#\dot{\sqcup}$$

   where we have k hashtags. This takes $O(n+k)$ steps.

2. For each of the $g(n)$ steps the multitape machine takes, we must pass through our simulation tape, recognizing the current state. We then perform a second pass to move dots left or right when needed. In each step, we add at most k new elements to our tape, so the size of the tape is always bounded by $n + kg(n)$. Therefore the number of steps in the first pass is in $O(n + kg(n))$. The second pass moves through the tape, and pushes at most k new symbols

into the tape, otherwise just moving the dots in the tape back and forth. A push moves at most $n + kg(n)$ symbols to the right, and thus the number of steps we take is in $O(k(n + kg(n)))$.

Step 2 is applied $g(n)$ times, so overall, the speed of the algorithm is in

$$O(n + k) + O(n + kg(n)) + g(n)O(n + kg(n)) = O(ng(n) + g^2(n))$$

assuming that $n \in O(g)$, then the speed is in $O(g^2)$. □

Thus, though multitape machines may compute faster, they only introduce do things quadratically faster than single tape machines. One of the biggest issues with complexity theory is that this is not true of non-deterministic machines.

First off, we must debate how long a non-deterministic machine takes to compute. We cannot simply count all steps the non-deterministic machine takes, because the machine takes many branches, some of which may be infinite. Since we can see non-deterministic computation as some sort of parallel processing, we could define the time to be the length of the shortest branch of processesing which yields termination. Mathematically, however, we will see that it is more convenient to define the run time to be the length of the longest branch. We are then able to define the time complexity of a non-deterministic automata.

Theorem 8.3. *If a non-deterministic machine runs in $O(f)$ time, then there is an equivalent deterministic automata which runs in $2^{O(f)}$ time.*

Proof. Perform a time analysis on the turing machine which simulates a non-deterministic machine. □

Thus non-deterministic algorithms are fundamentally connected to exponential deterministic algorithms. This makes sense, for in general, if a problem can be divided into exponentially many cases to check, each verifiable in a linear amount of time, then a non-deterministic algorithm can split into each possible case, exponentially dividing, and then check each case in a polynomial amount of time, giving us a polynomial time algorithm. Thus the time of Non-deterministic turing machines is bounded by the time it takes to verify a single case.

Now we get to the fun stuff. Define the complexity class

$$\mathbf{P} = \bigcup_{k=0}^{\infty} \mathbf{TIME}(n^k)$$

these are all problems computable in polynomial time on a deterministic automata.

Chapter 9

Pseudorandomness

Consider a binary string of n bits. What does it mean for this string to be random, when a random string of n bits takes every value with equal probability? What it means for a string to be random is realized by the modern theory of *algorithmic randomness*.

9.1 Statistical Testing

Let us come up with conditions 'random strings' should satisfy from a statistical approach. Consider an infinite string, i.e. an element $S = \{S_i\} \in \Sigma^{\mathbb{N}}$, where Σ is some finite alphabet. If $\{S_i\}$ were a family of independent random variables, uniformly distributed on Σ, then the law of large numbers tells us that, almost surely, for each $\sigma \in \Sigma$, if $N_S(\sigma, n)$ is the number of times that σ occurs in the first n digits of the sequence S, then by the law of large numbers, almost surely,

$$\lim_{n\to\infty} \frac{N_S(\sigma, n)}{n} = \frac{1}{\#(\Sigma)}.$$

Such a sequence is called *simply normal*. More generally, for *any* finite string $s \in \Sigma^*$ of length m, if we let $N_S(s, n)$ denote the number of occurences of s as a substring of the initial first n characters of S, then the Kolmogorov zero-one law implies that almost surely,

$$\lim_{n\to\infty} \frac{N_S(s, n)}{n} = \frac{1}{\#(\Sigma)^m}.$$

Such a sequence is called *normal*. A *number* $x \in \mathbb{R}$ is *normal in base b* if it's expansion in base b is a normal sequence, and *normal* if it is normal in every base.

Almost all numbers are normal, but it is very difficult to prove that particular numbers, like $\sqrt{2}$, e, or π, are normal.

Certainly a 'random sequence' should be normal. More generally, one might expect that a 'random sequence' $\{S_i\}$ has the property that for any increasing function $f : \mathbb{N} \to \mathbb{N}$, the sequence $\{S_{f(i)}\}$ is normal. However, there is an obvious problem here. The pidgeonhole principle implies that there exists $\sigma \in \Sigma$ and an increasing function $f : \mathbb{N} \to \mathbb{N}$ such that $X_{f(i)} = \sigma$ for all i. Certainly, the sequence $\{S_{f(i)}\}$ is not normal. To fix this problem, we should choose f in a way that does not depend on X, in some sense. An approach suggested by Von Mises, and later established by Alonso Church, was to restrict to the study of *computable functions*. But in 1939, Ville showed that von Mises' approach would not give 'fully random' sequences - he constructed, for any countable set of functions $\mathcal{F}$, a binary sequence $\{S_i\}$ such that $\{S_{f(i)}\}$ is normal for all $f \in \mathcal{F}$, but such that for every n, the subsequence $S_1 \dots S_n$ has more zeroes than ones - this never happens since almost surely,

$$\liminf_{n\to\infty} \frac{N_S(s,n) - n}{\sqrt{2n \log\log n}} = -1 \quad \text{and} \quad \limsup_{n\to\infty} \frac{N_S(s,n) - n}{\sqrt{2n \log\log n}} = 1.$$

One can immediately fix this by declaring a number to be random if it passes the law of the iterated logarithm in addition to the law of large numbers. But how do we know there is not another random test that could cause us problems here.

A fix to this problem emerged in the 1960s, due to Martin-Löf. In probability theory, a σ-algebra is defined on $\Sigma^{\mathbb{N}}$ by taking the Borel topology, which is that topology generated by the open sets $[s]$, for each $s \in \Sigma^*$, which is the set of all infinite strings which have s as an initial string. We define a Borel probability measure μ on $\Sigma^{\mathbb{N}}$, such that

$$\mu([s]) = \frac{1}{\#(\Sigma)^n}.$$

We have previously tried to define a sequence S to be random by finding sets $E \subset \Sigma^{\mathbb{N}}$ with $\mu(E) = 0$, and declaring that S should not be an element of E. This is of course not possible for all such E, since $\{S\}$ is a null set containing S for any sequence S, so we instead specialize to the *effectively null sets*.

Recall that a set $S \subset \Sigma^*$ is *computably enumerable* if there is a Turing machine which is able to list out all elements of S (and only elements of S). The set is *computable* if there is a Turing machine with an input tape such that, on input $n \in \Sigma^*$, the Turing machine is able to decide in finite time whether n is an element of S, or not an element of S. These notions are not equivalent: the family of

all descriptions of Turing machines which halt is not a computable set (Turing's famous proof), but the set is computably enumerable - simply set up a growing queue which eventually enumerates all possible Turing machines, and iteratively swap between adding a new Turing machine to this queue, and simulating a single step of all Turing machines in the queue, removing them when they eventually halt). These notions are closely related, however. A set is computable if and only if it is computably enumerable *in increasing order of length*.

Now let S be an arbitrary sequence. Now we say a sequence $\{T_n\}$ of open subsets of $\Sigma^{\mathbb{N}}$ is *uniformly computably enumerable* if there exists a computably enumerable set $G \subset \mathbb{N} \times \Sigma^*$ such that for each n,

$$T_n = \bigcup \{[\sigma] : (n, \sigma) \in G\}.$$

In addition, if a uniformly computably enumerable sequence $\{T_n\}$ satisfies $\mu(T_n) \leqslant 2^{-n}$ for all n, then the intersection is an *effectively null set*.

The method should now be clear. For any effectively null set $E \subset \Sigma^{\mathbb{N}}$, we consider a *Martin-Löf test* associated with E; a sequence S passes a Martin-Löf" test if $S \notin E$. If a sequence passes *all possible* Martin-Löf tests, it is called *Martin-Löf random*. Since there are only countably many Martin-Löf tests, and any particular Martin-Löf test is almost surely passed by a random sequence of symbols, we conclude that *almost every* infinite sequence is Martin-Löf random.

Remark. Roughly speaking, this is how a Martin-Löf test: we have a Turing machine that outputs a sequence of tuples $\{(n_k, s_k) : n_k > 0 \text{ and } s_k \in \Sigma^{\mathbb{N}}\}$, with the a-priori guarantee that

$$\mathbb{P}\left(\bigcup_{n_k = n} [\sigma_k]\right) \leqslant 2^{-n}.$$

We begin by writing down each positive integer. Every time a tuple (n_k, s_k) emerges from the Turing machine, we check whether s_k is the initial string of S - if it is, we cross off the integer n_k. The string S passes the test if there exists a number which is not eventually crossed off in the running of the algorithm.

As an example, consider the Von Mises test, i.e. testing, for a given string $s \in \Sigma^*$ of length m, whether a given infinite sequence S satisfies

$$\lim_{n \to \infty} \frac{N_S(s, n)}{n} = \frac{1}{\#(\Sigma)^m}.$$

Here's a Martin-Löf test which actually gives a strong statement. If $\{S_n\}$ are independent random variables, then we can write $N_S(s, n) = Z_1 + \cdots + Z_m$, where

each Z_i is the sum of m different random quantities, each a sum of n/m independent $\mathrm{Ber}(p)$ random variables, where $p = 1/\#(\Sigma)^m$. Hoeffding's inequality guarantees that

$$\left\| \frac{N_S(s,n)}{n} - \frac{1}{\#(\Sigma)^m} \right\|_{\psi_2} \leqslant n^{-1} \sum_i \|Z_i - \mathbb{E}[Z_i]\|_{\psi_2} \lesssim (m/n)^{1/2}.$$

Thus there is some universal computable constant $c_1 > 0$ such that for any $t > 0$,

$$\mathbb{P}\left(\left| \frac{N_S(s,n)}{n} - \frac{1}{\#(\Sigma)^m} \right| \geqslant t \right) \leqslant 2e^{-c_1 n t^2/m}.$$

In particular, there exists a universal computable constant $c > 0$ such that for $t \gtrsim 1$,

$$\mathbb{P}\left(\left| \frac{N_S(s,n)}{n} - \frac{1}{\#(\Sigma)^m} \right| \geqslant cm^{1/2} n^{-1/2} \log(t/2)^{1/2} \right) \leqslant 1/t.$$

Now consider a computable sequence of positive numbers $\{\delta_k\}$ that is summable, and suppose there exists a computable sequence $\{k_n\}$ such that

$$\sum_{k_n \leqslant k < k_{n+1}} \delta_n \leqslant 2^{-n}.$$

Consider the following Martin-Löf test; we enumerate all pairs (n, s), where s is a string of length $k \geqslant k_n$, and

$$\left| \frac{N_s(s,n)}{n} - \frac{1}{\#(\Sigma)^m} \right| \geqslant cm^{1/2} n^{-1/2} \log(1/2\delta_n)^{1/2}.$$

It is clear that the associated open set T_n has $\mu(T_n) \leqslant 2^{-n}$ because of the choice of k_n. But this means that if S is Martin-Löf randm, then there exists n_0 such that for $n \geqslant n_0$,

$$\left| \frac{N_s(s,n)}{n} - \frac{1}{\#(\Sigma)^m} \right| = cm^{1/2} n^{-1/2} \log(1/2\delta_n)^{1/2}.$$

Thus we have a quantitative rate of convergence of these averages, as compared to the qualitative test of Von Mises.

9.2 Martingales

Suppose S is a random sequence of symbols. A *Martingale* is an assign of fair betting odds to each outcome of the sequence, i.e. a function $f : \Sigma^* \to [0, \infty)$

such that for any string s,

$$f(s) = \frac{1}{\#(\Sigma)} \sum_{\sigma \in \Sigma} f(s\sigma).$$

If $S = \{S_i\}$ is a random sequence of symbols, then $\mathbb{E}(f(S_1 \ldots S_n)) = \mathbb{E}(f(\varepsilon))$ for any $n > 0$. The fact that f is non-negative allows us to apply the Martingale convergence theorem to conclude that there exists a function $g : \Sigma^{\mathbb{N}} \to [0, \infty)$ such that, almost surely, $f(S_1 \ldots S_n) \to g(S)$. In particular, almost surely, we know that

$$\limsup_{n \to \infty} f(S_1 \ldots S_n) < \infty.$$

In this case, we say f *suceeds* on the sequence S. We say a sequence S is *computably random* if there does not exist a computable Martingale f which succeeds on S.

Part IV

Descriptive Set Theory

Chapter 10

S

Consider a class $C(d)$ of functions from $\mathbb{R}^d$ to $\mathbb{R}$, for each positive integer d. We can then consider a family $\mathcal{S}_0(d)$ of subsets E of $\mathbb{R}^d$ which can be written as

$$E = \{x \in \mathbb{R}^d : f_1(x) = 0, \ldots, f_n(x) = 0, g_1(x) > 0, \ldots, g_m(x) > 0\},$$

for $f_1, \ldots, f_n, g_1, \ldots, g_m \in C(d)$. Recursively define a sequence $\mathcal{S}_n(d)$ of subsets of $\mathbb{R}^d$, for $n > 0$, by letting the set by the union of $\mathcal{S}_{n-1}(d)$ together with:

- All pairwise unions, intersections of elements of $\mathcal{S}_{n-1}(d)$.
- All complements and topological closures of elements of $\mathcal{S}_{n-1}(d)$.
- All cartesian productions of elements of $\mathcal{S}_{n-1}(k)$ with $\mathcal{S}_{n-1}(d-k)$, for $1 \leqslant k \leqslant d-1$.
- All projections of elements of $\mathcal{S}_{n-1}(k)$, for $k > d$, into any arrangement of coordinates.

If we keep repeating this process, there are two possible circumstances: either the sequence *stabilizes*, i.e. there exists n_0 such that $\mathcal{S}_n(d) = \mathcal{S}_{n+1}(D)$ for $n \geqslant n_0$, or the sequence *never stabilizes*. In the former case, we often have a rich geometric structure for all constructible sets. In the latter case, we can obtain very strange sets, such as Cantor like structures. The focus of these notes is determining the general theory which follows from the consequence of this stability, the main initial cases being obtained by letting C be the class of linear functions (the class of sets obtained are the *semilinear sets*), and when C is the class of polynomial functions (the class of sets here being called *semialgebraic*). It is a result

of Tarski and Seidelberg than the projection of any semialgebraic set is semialgebraic. Similarily, the closure, interior, and convex hull of a semialgebraic set is semialgebraic. Each semialgebraic set has finitely many connected components, and each component can be triangulated into finitely many semialgebraic real analytic manifolds. This book will show these results generalize to more general 'o-minimal structures'.

10.1 *o*-minimality

An *o-minimal structure* is a sequence $\{\mathcal{S}_d\}$, where for each d, $\mathcal{S}_d$ is a Boolean subalgebra of $\mathbb{R}^d$ with unity (i.e. a family of subsets of $\mathbb{R}^d$ containing $\mathbb{R}^d$ and closed under union, intersections, and complements), such that:

- If $A \in \mathcal{S}_d$, then $A \times \mathbb{R}$ and $\mathbb{R} \times A$ are elements of $\mathcal{S}_{d+1}$.
- If $\pi : \mathbb{R}^{d+1} \to \mathbb{R}^d$ is the projection map obtained by removing the last coordinate, and if $A \in \mathcal{S}_{d+1}$, then $\pi(A) \in \mathcal{S}_d$.
- For any d, and any $1 \leqslant i \leqslant j \leqslant d$, the set $\{x \in \mathbb{R}^d : x_i = x_j\}$ lies in $\mathcal{S}_d$.
- $\mathcal{S}_1$ is the set of all finite unions of points and intervals.

There are many interesting *o*-minimal structures: semialgebraic sets, semilinear sets. But they also have much structure, as we now see.

Given an *o*-minimal structure, we say a set is *definable* if it lies in the structure. We say a function $f : A \to \mathbb{R}$ is definable if it's graph is definable. We will see that the closure and interior of any definable set is definable, and, perhaps more suprisingly, and for *any* definable function $f : [a, b] \to \mathbb{R}$, we can decompose $[a, b]$ into a disjoint union of intervals, such that f is continuous on the interior of each interval, and is either constant on the interior of the interval, strictly increasing on the interior of the interval, or strictly decreasing on the interior of the interval. More generally, any definable set can be split into the disjoint union of *cells*, i.e. either graphs of unions of cells

Bibliography

[1] Mendelson, *Introduction to Mathematical Logic*, Chapman and Hall, 1997.

[2] Terrence Tao, *The Completeness and Compactness Theorems of First Order Logic*, 2009.

[3] Douglas Hofstadter, *Gödel, Escher, Bach: an Eternal Golden Braid*, 1979.

[4] I, Grattan-Guinness, *How Bertrand Russell Discovered His Paradox*, 1978.