Classical Metalogic

An Introduction to Classical Model Theory & Computability

Notes by R.J. Buehler

January 5, 2015

Preface

What follows are my personal notes created in preparation for the UC-Berkeley Group in Logic preliminary exam. I am not a computability theorist, nor a model theorist; I am a graduate student with some knowledge who is–alas–quite fallible. Accordingly, this text is made available as a convenient reference, set of notes, and summary, but without even the slight hint of a guarantee that everything contained within is factual and correct (indeed, some areas are entirely unchanged from the moment I copied them off the blackboard). This said, if you find a serious error, I would greatly appreciate it if you would let me know so that it can be corrected. The material for these notes derives from a wide variety of sources:

- Lectures by Wes Holliday
- Lectures by Antonio Montalban
- Lectures by John Steel
- Kevin Kelly's computability theory notes
- David Marker's "Model Theory: An Introduction"
- Wilfrid Hodge's "A Shorter Model Theory"
- Robert Soare's "Recursively Enumerable Sets and Degrees"
- Richard Kaye's "Models of Peano Arithmetic"
- Chang and Keisler's "Model Theory"

While I certainly hope my notes are beneficial, if you're attempting to learn the contained material for the first time, I would highly suggest picking up (at least) a copy of Marker and Soare's texts in addition.

To those Group in Logic students who may be using these notes to help themselves prepare for their preliminary exam, chapters 1-5, 7, and 9-18 contain relevant material, as well as chapter 8, section 3.

Contents

Chapter 1

First-Order Logic

Although our aim is, quite clearly, to speak eventually of full first-order logic, we begin much more simply–with the basic building blocks thereof. The first of these basic 'building blocks' is the notion of a signature or vocabulary.

1.1 Signatures

A signature is simply a listing of the symbols with which we will be working–our vocabulary, so to speak. This vocabulary of symbols is divided into three sets based on what kind of thing a given symbol is intended to represent.

Signature
A signature is a (possibly empty) collection of the following sets:

i. The set of constant symbols; symbols which represent a particular entity

ii. For every $n \in \mathbb{N}$, a set of n-ary relation symbols; symbols which represent a relation which either holds or does not hold between tuples of n entities.

iii. For every $n \in \mathbb{N}$, a set of n-ary function symbols; symbols which represent functions mapping n tuples of entities to entities.

A signature L with no constants or function symbols is, for obvious reasons, called a *relational signature*; a signature with no relation symbols is, for less obvious reasons, sometimes called an *algebraic signature*.

1.2 Languages

In addition to the symbols provided by the signature, every language also has a stock of *variables*. Any symbol can be used as a variable provided that it is not already being used. The actual choice of variables is never important, and so it's common to restrict them to $x_0, x_1, \ldots$ with natural number or cardinal subscripts.

1.2.1 Terms

The terms of a signature will eventually play a critical role in our work; intuitively, a term is a combination of function symbols, constants, and variables–something which would denote an individual if all the variables were assigned.

Term
The terms of a signature L are the strings of symbols defined as follows:

- Every variable is a term of L
- Every constant is a term of L
- For $n > 0$, if f is an n-ary function symbol of L, and $t_1, \ldots, t_n$ are terms of L, then $f(t_1, \ldots, t_n)$ is a term of L as well.
- Nothing else is a term of L.

In other words, the terms of a signature (eventually, language) are simply the constants and variables closed under the application of function symbols.

Closed Term
A term with no variables

If a term t is introduced as $t(\bar{x})$ this means that $\bar{x}$ is a sequence $(x_0, x_1, \ldots, x_n)$ of variables, and every variable which occurs in t is among the variables in $\bar{x}$.

1.2.2 Atomic Formulas and Sentences

Atomic Formula
The atomic formulas of a signature L are the strings of symbols generated by:

- If s and t are terms of L, then '$s = t$' is an atomic formula of L.
- For $n > 0$, if R is an n-ary relation symbol of L, and $t_1, \ldots, t_n$ are terms of L, then the expression $R(t_1, \ldots, t_n)$ is an atomic formula of L.

Atomic Sentence
An atomic formula in which there are no variables or–equivalently–all of whose terms are closed.

Just as with terms, if we introduce an atomic formula φ as $\varphi(\bar{x})$, then $\varphi(\bar{s})$ means the atomic formula got from φ by putting terms from the sequence $\bar{s}$ in place of all occurrences of the corresponding variables in $\bar{x}$.

At times, it will be useful to refer to not only that atomic formulas of a signature, but also negated atomic formulas:

Literal
An atomic formula or negated atomic formula; paralleling earlier use, closed literals are those literals without variables.

1.2.3 Formal Languages

Let L be an arbitrary signature. To achieve a (canonical) formal language $\mathcal{L}$ from the signature L, the symbols of L will be supplemented by the 'normal logical connectives'; that is,

$\neg$	'not'
$\wedge$	'and'
$\vee$	'or'
$=$	'equals'
$\forall$	'for all'
$\exists$	'there exists'

The terms, closed terms, atomic formulas, atomic sentences, etc. of the formal language $\mathcal{L}$ are defined to be the same as those of the signature L. Finally, we define $\mathcal{L}$ itself as the class of all its well-formed formulas. The well-formed formulas φ of $\mathcal{L}$ include, for any atomic formula A of L and any well-formed formula φ, both

$$A \text{ and } \neg\varphi$$

which is quite standard. In addition to these, however, we also allow possibly infinitary constructions with a given cardinal upperbound α on conjunction/disjunction and β on quantification:

- If $\Phi \subseteq \mathcal{L}$ and $|\Phi| < \alpha$, then
$$\bigwedge\Phi \in \mathcal{L} \text{ and } \bigvee\Phi \in \mathcal{L}$$
- if $\varphi \in \mathcal{L}$ and $\bar{x}$ is a tuple of variables such that $|\bar{x}| < \beta$, then
$$(\forall\bar{x})\,[\varphi] \in \mathcal{L} \text{ and } (\exists\bar{x})\,[\varphi] \in \mathcal{L}$$

In addition to the *well-formed formulas* of the language, we also distinguish a particular class of these formulas: *sentences*.

Sentence
A formula of a language $\mathcal{L}$ with no free variables.

Some thought quickly shows that every language has a unique signature; since our primary topic will be languages, we will often conflate the two. In addition, although only occasionally observed elsewhere, given a language $\mathcal{L}$ generated from a signature L with conjunction/disjunction and quantification upperbounds α / β respectively, we will use $\mathcal{L}_{\alpha\beta}$ as the name of this language.

Example 1.

$\mathcal{L}_{\infty\infty}$	Any number of disj./conj. and any number of quantifiers
$\mathcal{L}_{\kappa\infty}$	Less than κ formulas disj./conj. and any number quantifiers
$\mathcal{L}_{\aleph_0\aleph_0}$	Only finitely many formulas disj./conj. and only finitely many quantifiers
$\mathcal{L}_{\aleph_0 0}$	Only finitely many formulas disj./conj. and no quantifiers (called *quantifier-free*)

Although the term *first-order* technically refers to logics quantifying only over objects–a criterion which all of our languages meet, regardless of the particular α, β chosen–we will specifically use the term for the language $\mathcal{L}_{\aleph_0\aleph_0}$ which is our primary concern. If an infinitary language is being considered, it will be made explicit. In the $\mathcal{L}_{\aleph_0\aleph_0}$ case, the more complex definition of language given above simplifies to the one typically given in undergraduate courses:

$$\text{Wffs} = \begin{cases} R(\bar{t}) & \text{for } \bar{t} \text{ terms} \\ \neg\varphi \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid \forall x\varphi \mid \exists x\varphi & \text{for } \varphi, \psi \in \text{Wffs} \end{cases}$$

We add the following definitions to our repertoire in order to mirror our natural intuitions on the similarity of various formulas:

Variant
For a language $\mathcal{L}$ and corresponding signature L, we say that two n-ary formulas of $\mathcal{L}$, $\varphi(\bar{x})$ and $\psi(\bar{y})$, are variants of one another if $\varphi(\bar{x})$ may be obtained from $\psi(\bar{y})$ via a uniform replacement of variables (that is, $\psi(\bar{x}) = \varphi(\bar{x})$) and vice-versa.

Note that the relation of variance over the formulas of a language is an equivalence relation; the cardinality of a first-order language $\mathcal{L}$ is actually defined as the number of variant equivalence classes of its formulas. In simpler terms, the actual choice of variables is immaterial; for our purposes $R(x_1, x_2)$ may as well be $R(x_3, x_9)$.

1.2.4 Standard Abbreviations

The following is a quick survey of standard abbreviations used here and elsewhere with the formulas of languages.

$x \neq y$	for	$\neg x = y$
$(\varphi_0 \wedge \cdots \wedge \varphi_n)$	for	$\bigwedge\{\varphi_0, \ldots, \varphi_n\}$
$(\varphi_0 \wedge \cdots \wedge \varphi_n)$	for	$\bigvee\{\varphi_0, \ldots, \varphi_n\}$
$\bigwedge_{i \in I} \varphi_i$	for	$\bigwedge\{\varphi_i \mid i \in I\}$
$\bigvee_{i \in I} \varphi_i$	for	$\bigvee\{\varphi_i \mid i \in I\}$
$(\varphi \rightarrow \psi)$	for	$(\neg\varphi) \vee \psi$
$(\varphi \leftrightarrow \psi)$	for	$(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
$\forall x_1 x_2 \ldots x_n$ or $\forall \bar{x}$	for	$\forall x_1 \forall x_2 \ldots \forall x_n$
$\exists x_1 x_2 \ldots x_n$ or $\exists \bar{x}$	for	$\exists x_1 \exists x_2 \ldots \exists x_n$
$\bot$	for	$\bigvee\emptyset$

Normal conventions for parenthesizing are also in effect; $\neg$ binds tighter than all the other logical symbols while $\wedge$ and $\vee$ bind tighter than $\rightarrow$ and $\leftrightarrow$ in turn.

1.3 Normal Forms

Propositional and first-order logic possess a number of normal forms; the most important are briefly canvassed below:

Conjunctive Normal Form (CNF)
Any truth functional formula φ of a first-order language $\mathcal{L}$ (i.e. no quantifiers) is equivalent to a formula which is a conjunction of disjunctions of literals, i.e.

$$[\bigvee \Phi_0] \wedge [\bigvee \Phi_1] \wedge \ldots \wedge [\bigvee \Phi_n]$$

where the Φ_i are finite sets of literals.

Disjunctive Normal Form (DNF)
Any truth functional formula φ of a first-order language $\mathcal{L}$ (i.e. no quantifiers) is equivalent to a formula which is a disjunction of conjunctions of literals, i.e.

$$[\bigwedge \Phi_0] \vee [\bigwedge \Phi_1] \vee \ldots \vee [\bigwedge \Phi_n]$$

where the Φ_i are finite sets of literals.

Prenex Normal Form
Any formula φ of a first-order language $\mathcal{L}$ is equivalent to a formula which has all its quantifiers preceding it, i.e.

$$\forall x_0 \exists x_1 \ldots \forall x_n \psi(\bar{x})$$

where ψ is quantifier-free.

Part I

Classical Model Theory

Chapter 2

The Basics of Models

The most basic example of a model theory is that of propositional logic: Emile Post's much vaunted truth table.

Example 2.
A simple truth table.

p	q	$\neg q$	$p \wedge \neg q$
T	T	F	F
T	F	T	T
F	T	F	F
F	F	T	F

This simple model theory may be taken as our paradigm; it elegantly represents all facts necessary to determining the truth or falsity of a given sentence in a formal and concise manner. If more generality were desired (as it may very well be), imagine assigning truth values to all the atomic propositions available for use. Each of these assignments gives a different model, a different situation.

2.1 Models

So-called models, interpretations, or structures are the primary topic of model theory; a model will play the role of a particular instantiation of our signature, one way the constant symbols could have been assigned, one definition each of the relation symbols could have been given, and one function each of our function symbols could have named. All of these could, of course, have been different; a model simply gives one way the 'world' could have been.

Model
For a given signature L, a model A is an object made up of the following four elements:

i. A set called the *domain* of A, written '$\mathrm{dom}(A)$' or '$\mathrm{dom}\ A$'; occasionally also referred to as the *universe* or *carrier* of A. The elements of $\mathrm{dom}(A)$ are called the *elements* or *entities* of the model A. The cardinality of A, symbolized $|A|$, is defined to be the cardinality of $\mathrm{dom}(A)$ or $|\mathrm{dom}(A)|$.

ii. A referent for each of the *constant symbols* of the signature L. If c is a constant symbol of the signature, we write c^A for the element named by the constant symbol 'c' in the model A.

iii. For each positive integer n and each n-ary relation symbol of the signature, an extension for this relation symbol; that is, an n-ary relation on $\mathrm{dom}(A)$ (subsets of $(\mathrm{dom}(A))^n$) which the n-ary relation symbol from the signature names. If R is a relation symbol, we write R^A for the relation named by the symbol 'R' in the model A.

iv. For each positive integer n and n-ary function symbol of the signature, an n-ary function on $\mathrm{dom}(A)$ (i.e. a map from $(\mathrm{dom}(A))^n$ to $\mathrm{dom}(A)$) which this symbol names. If f is a function symbol, we write f^A for the function named by 'f' in the model A.

As the definition implies, a model is always associated with a signature; to make this connection explicit, given a signature L, a model A of that signature is called an L-model. Crucially, the domain of a first-order model is not allowed to be empty, but the extensions of relations are. Finally, we adopt the following notation:

- Models are symbolized by capital letters: $A, B, C, \ldots$

- Sequences of elements from a model are represented by placing a bar: $\bar{a} = a_1, a_2, \ldots, a_n$ where $a_1, a_2, \ldots, a_n \in \mathrm{dom}(A)$. A *tuple in/from* A is a finite sequence of elements of A; it is an *n-tuple* if it has exactly length n. In a minor breach of this notation, we will occasionally make use of an infinite sequence of elements and also denote it with $\bar{a}$, $\bar{b}$, $\bar{c}$, etc; to avoid confusion, whenever we do so it will be made explicit that the tuples being used are possibly infinite.

For better or worse, there is no standard way of representing a model; it often falls to the reader to make educated guesses as to what a writer intended.

2.2 Models and Formal Languages

From a given signature L, we may construct a number of formal languages, most notably $\mathcal{L}_{\aleph_0\aleph_0}$. Similarly, a signature L determines a class of L-models. As foreshadowed above, L-models and a formal language $\mathcal{L}$ generated from L share a very special relationship, namely that the class of L-models exhausts all possible interpretations of the formal language $\mathcal{L}$, it represents every different way things could be with respect to the language $\mathcal{L}$.

To briefly recap, an L-model A provides an interpretation of the proper type for every symbol in the signature L. For any constant symbols c, this means c^A is some element of $\mathrm{dom}(A)$; for any n-ary relation symbol R or n-ary function symbol f, this means an n-ary relation over $\mathrm{dom}(A)$, denoted R^A, and an n-ary function $f^A : \mathrm{dom}(A)^n \to \mathrm{dom}(A)$, respectively. In doing so, however, a model does far more–assuming standard first-order logic, it sets a truth value for every well-formed formula of any language $\mathcal{L}$ generated from the signature L. Mathematicians tend to symbolize this relationship with $\models$, writing '$A \models \varphi$' whenever the formula φ is made true by the model A and saying A *models* φ. We retain their phrasing, but reserve the symbol $\models$ for the model-theoretic consequence relation. In lieu, we adopt $\Vdash$, following the usage of modal logicians.

For a model A, $|A|$ is the cardinality of A. For a signature L, $|L|$ denotes the cardinality of L if and only if L is infinite; otherwise, if L is finite, we'll say $|L| = \aleph_0$.

Truth in a Model
Let L be a signature, $\mathcal{L}$ a language generated from L, and A an L-model. Furthermore, let $t(\bar{x})$ be a term of $\mathcal{L}$ and $\bar{a}$ a sequence of elements from A at least as long as $\bar{x}$. Then, by recursion on the complexity of the term t,

- If $t(\bar{x})$ is c for some constant c of the signature L, then $t^A[\bar{a}]$ is c^A
- If $t(\bar{x})$ is x_i for some variable x_i in $\bar{x}$, then $t^A[\bar{a}]$ is a_i; that is, the corresponding entry in the tuple $\bar{a}$.
- If $t(\bar{x})$ is $f(\bar{s})$ for some function symbol f and tuple of terms $\bar{s}$, then $t^A[\bar{a}]$ is $f^A(s_1^A[\bar{a}], \ldots, s_n^A[\bar{a}])$

Note that these three clauses fix the denotation of any term in $\mathcal{L}$ when it is paired with a tuple of elements from $\mathrm{dom}(A)$ of appropriate length. This done, we may now define *truth in a model* or the $\Vdash$ relation. Again, the definition is recursive, but now on the complexity of the formula. First, suppose that $\varphi(\bar{x})$ is atomic. Then, given a tuple of elements $\bar{a}$ at least as long as $\bar{x}$,

- $A \Vdash R(t_1, \ldots, t_n)[\bar{a}]$ is true for some relation symbol R in L if and only if $\langle t_1^A[\bar{a}], \ldots, t_n^A[\bar{a}]\rangle \in R^A$
- $A \Vdash t \approx s\ [\bar{a}]$ if and only if $t^A[\bar{a}] = s^A[\bar{a}]$

Finally, the recursive step:

- $A \Vdash \neg\varphi[\bar{a}]$ if and only if $A \nVdash \varphi[\bar{a}]$
- $A \Vdash \bigwedge\Phi[\bar{a}]$ if and only if for every formula $\psi(\bar{x}) \in \Phi$, $A \Vdash \psi[\bar{a}]$
- $A \Vdash \bigvee\Phi[\bar{a}]$ if and only if for some formula $\psi(\bar{x}) \in \Phi$, $A \Vdash \psi[\bar{a}]$
- $A \Vdash \forall y\, \varphi[y, \bar{a}]$ if and only if for every element b of A, $A \Vdash \varphi[b, \bar{a}]$
- $A \Vdash \exists y\, \varphi[y, \bar{a}]$ if and only if for some element b of A, $A \Vdash \varphi[b, \bar{a}]$

Formula Equivalence in a Model
For a language $\mathcal{L}$, corresponding signature L, and L-model A, we say that two n-ary formulas of $\mathcal{L}$, $\varphi(\bar{x})$ and $\psi(\bar{y})$, are equivalent in A if and only if $A \Vdash \forall\bar{x}\ [\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})]$.

2.2.1 Model-Theoretic Logical Consequence

Model-Theoretic Logical Consequence
Let $\mathcal{L}$ be a language, L the corresponding signature, T a set of sentences of $\mathcal{L}$, and φ a sentence of $\mathcal{L}$. T entails φ or φ is a model-theoretic consequence of T–in symbols, $T \models \varphi$–if and only if every model of T is also a model of φ.

Entailment in hand, a host of familiar logical concepts follow closely behind:

Validity
A sentence of a language $\mathcal{L}$ is valid if it is entailed by the empty set; that is, $\models \varphi$. φ is also called a *logical theorem*.

Satisfiability
A sentence φ of a language $\mathcal{L}$ is called satisfiable or consistent if there exists an L-model A such that $A \Vdash \varphi$; similarly, a set of sentences Φ of a language $\mathcal{L}$ is called satisfiable or consistent if and only if there exists an L-model A such that $A \Vdash \varphi$ for every $\varphi \in \Phi$.

2.3 Theories

While models are, unsurprisingly, the primary topic of model theory, theories represent a close second place. Theories–along with the model theoretic consequence relation–will bridge the gap between the formal languages and models.

Theory
A set of sentences of a particular formal language $\mathcal{L}$

Extension of a Theory
An *extension of a theory* T (or supertheory of T) is a theory T' such that $T \subseteq T'$.

Modeling a Theory
Let $\mathcal{L}$ be a language, L the corresponding signature, A an L-structure, and T a theory in $\mathcal{L}$. $A \Vdash T$ if and only if $A \Vdash \varphi$ for every sentence $\varphi \in T$.

Note that it makes sense to speak of a model of a theory (with no element assignment) since every formula under consideration has no free variables.

2.3.1 Theories and Diagrams of Models

In moving from models to theories, we distinguish three theories of particular importance:

Theory of a Model
For an L-model A, the theory of A in the language $\mathcal{L}$ or $\mathrm{Th}_{\mathcal{L}}(A)$ is the set of all $\mathcal{L}$-sentences φ such that $A \Vdash \varphi$. That is, all $\mathcal{L}$-sentences true in A.

Atomic Diagram
For an L-model A, the *atomic diagram of* A or $\mathrm{Diag}_{\mathrm{at}}(A)$ is the theory containing all closed literals φ of the language $\mathcal{L}^+$ such that $A^+ \Vdash \varphi$, where $\mathcal{L}^+$ is the language generated by expanding the signature L to include a new constant for every element of the domain of A and A^+ is A with each of these new constants representing the intended element of the domain.

Couched in less formal terms, the expansion from L to L^+ gives the ability to name every element of the model A. To make this connection clear, the added constants are often subscripted with the names of the elements themselves, e.g. we add c_a for every $a \in A$. Of course, simply expanding the signature–and by extension the language–with new constants is not enough; the movement from A to A^+ guarantees that each of these constants refers to its intended element of A. A itself is, after all, not an L^+-model.

Elementary Diagram
For an L-model A, the *elementary diagram of* A or $\mathrm{Diag}_{\mathrm{el}}(A)$ is the theory containing all sentences φ of the language $\mathcal{L}^+_{\aleph_0\aleph_0}$ such that $A^+ \Vdash \varphi$, where $\mathcal{L}^+_{\aleph_0\aleph_0}$ is the first-order language generated by expanding the signature L to include a new constant for every element of the domain of A and A^+ is A with each of these new constants representing the intended element of the domain.

Just as above, the expansion from L to L^+ gives the ability to name every element of the model A, while the movement to A^+ ensures these constants refer as intended. The new stipulation of $\mathcal{L}^+_{\aleph_0\aleph_0}$ above is the reason for the term *elementary*–a common synonym in model theory for *first-order*.

Finally, while it's clear that, for any model A, $\mathrm{Diag}_{\mathrm{at}}(A) \subset \mathrm{Diag}_{\mathrm{el}}(A)$, $\mathrm{Th}_{\mathcal{L}}(A)$ is constructed using a different (smaller) language. Thus, it is not true that $\mathrm{Diag}_{\mathrm{at}}(A) \subset \mathrm{Th}_{\mathcal{L}}(A)$–although, from the perspective of $\mathcal{L}^+$, $\mathrm{Th}_{\mathcal{L}}(A) \subset \mathrm{Diag}_{\mathrm{el}}(A)$. All three of the theories above will prove useful later, but even now they suggest a distinction in the

'sameness' of models. In particular, note that for two L-models A and B, $B \Vdash \mathrm{Th}_{\mathcal{L}}(A)$ is a strictly weaker condition than $B^+ \Vdash \mathrm{Diag}_{\mathrm{at}}(A)$ (for some extension B^+ of B) which is in turn strictly weaker than $B^+ \Vdash \mathrm{Diag}_{\mathrm{el}}(A)$ (for some extension B^+ of B).

> **$\mathcal{L}_{\alpha\beta}$-Equivalent**
> Two $\mathcal{L}_{\alpha\beta}$-models A and B are *$\mathcal{L}_{\alpha\beta}$-equivalent* if and only if $Th_{\mathcal{L}_{\alpha\beta}}(A) = Th_{\mathcal{L}_{\alpha\beta}}(B)$, written $A \equiv_{\mathcal{L}_{\alpha\beta}} B$.

In practice, however, we will constrain ourselves to $\mathcal{L}_{\aleph_0\aleph_0}$, and so we will almost exclusively use:

> **Elementary Equivalent**
> Two L-models A and B are *elementary equivalent* if and only if $\mathrm{Th}_{\mathcal{L}_{\aleph_0\aleph_0}}(A) = \mathrm{Th}_{\mathcal{L}_{\aleph_0\aleph_0}}(B)$, written simply as $A \equiv B$.

2.3.2 Canonical Models

The reverse direction–from theory to model–is, as it turns out, a significantly harder task. Here we consider only the most simplistic case–although we return to the topic in our discussion and proof of compactness. Given a theory T of atomic sentences, the following gives a canonical model A such that $A \Vdash T$.

> **Equality Closed**
> A theory T of atomic sentences is *equality closed* if and only if
>
> - For every closed term t, $t = t$ is in T
> - If $\varphi(x)$ is an atomic formula, s and t are terms, $\varphi(s) \in T$, and $s = t \in T$, then $\varphi(t) \in T$.

► **Theorem 2.1** (Henkin Models for Atomic Theories)**.**
Let T be a consistent theory containing only atomic sentences. Then, there is a model A such that

- $A \Vdash T$
- T is exactly the set of atomic sentences in A.
- Every element of A is of the form t^A for some closed term t.

Proof Sketch.
First, generate the equality closure T' of T. Let the domain of A be the equivalence classes of $=$ over the closed terms of T'. For any n-ary function f and tuple $\bar{t}$ of n terms, set $f(\bar{t})^A = [f(\bar{t})]$; that is, set $f(\bar{t})^A$ to be the equivalence class which includes the closed term $f(\bar{t})$. Similarly, for any n-ary relation R, set R^A to be the set of tuples $\bar{t}$ such that $R(\bar{t})$ is in T'.

The method of proof sketched above is a *Henkin construction*; later, we'll see that the same process can be expanded to construct models for arbitrary consistent theories.

Chapter 3

Relations Between Models

3.1 From Homomorphism to Isomorphism

By earlier, $\bar{a}$ is $\langle a_0, \ldots, a_{n-1} \rangle$; for a function δ, we adopt the notation $\delta\bar{a}$ for the tuple $\langle \delta(a_0), \ldots, \delta(a_{n-1}) \rangle$.

Homomorphism
Let L be a signature and let A, B be L-models. A *homomorphism* is a function $\delta : \mathrm{dom}(A) \to \mathrm{dom}(B)$ such that:

- For each constant symbol c in L, $\delta(c^A) = c^B$
- For each $n > 0$, each n-ary relation symbol R in L, and each n-tuple $\bar{a}$ from A, if $\bar{a} \in R^A$, then $\delta\bar{a} \in R^B$.
- For each $n > 0$, each n-ary function symbol f in L, and n-tuple $\bar{a}$ from A, $\delta(f^A(\bar{a})) = f^B(\delta\bar{a})$

In other words, the existence of a homomorphism from a model A to a model B implies that some portion of B looks like A–albeit in a very weak sense. In particular, over that portion of B the functions of L behave as they do in A and the relations of B contain at least everything they do in A, although a single element of B may represent multiple elements in A. This definition can clearly be strengthened, and–in fact–we do precisely this:

Embedding
Let L be a signature and let A, B be L-models. A function $\delta : \mathrm{dom}(A) \to \mathrm{dom}(B)$ is an *embedding* if and only if it is injective, a homomorphism, and satisfies a strengthened second condition:

- For each $n > 0$, each n-ary relation symbol R of L and each n-tuple $\bar{a}$ from A, $\bar{a} \in R^A \Leftrightarrow \delta\bar{a} \in R^B$.

Less formally, the existence of an embedding from a model A to a model B means that B contains an exact copy of A somewhere inside of it.

Isomorphism
Let L be a signature and let A, B be L-models. A function $\delta : \mathrm{dom}(A) \to \mathrm{dom}(B)$ is an *isomorphism* if and only if it is a surjective embedding. We say A is isomorphic to B if an isomorphism $\delta : A \to B$ exists, symbolized $A \cong B$.

Isomorphism is, as usual, taken to mean that the models A and B are essentially the same; that they don't differ in any interesting respect. In the future, for the sake of brevity, we denote functions $\delta : \mathrm{dom}(A) \to \mathrm{dom}(B)$ simply by $\delta : A \to B$. In addition, the following terminology will occasionally be used: homomorphisms $\delta : A \to A$ are called *endomorphisms* of A; isomorphisms $\delta : A \to A$ are called *automorphisms* of A. The identity map on $\mathrm{dom}(A)$, I_A, is an automorphism for any model A.
The following follows easily from the definitions given.

► **Theorem 3.1** (Basic Properties)**.**
Let L be a signature.

(i) If A, B, and C are L-models and $\delta : A \to B$, $\mu : B \to C$ are homomorphisms / embeddings / isomorphisms, then the composed map $\mu \circ \delta$ is a homomorphism / embedding / isomorphism from A to C

(ii) Let A and B be L-models. If $\delta : A \to B$ is an isomorphism, then the inverse map $\delta^{-1} : \mathrm{dom}(B) \to \mathrm{dom}(A)$ exists and is an isomorphism from B to A

(iii) The relation $\cong$ is an equivalence relation on the class of L-models.

(iv) If A, B are L-models, $\delta : A \to B$ is a homomorphism, and there are homomorphisms $\mu : B \to A$ and $\epsilon : B \to A$ such that $\mu \circ \delta = I_A$ and $\delta \circ \epsilon = I_B$, then δ is an isomorphism and $\mu = \epsilon = \delta^{-1}$.

Proof Sketch.
Simply leverage the definitions above

► **Theorem 3.2** (Basics of Homomorphism)**.**
If $\delta : A \to B$ is a homomorphism,

(i) For every term $t(\bar{x})$ and tuple $\bar{a} \in A$ of sufficient length, $\delta(t^A[\bar{a}]) = t^B(\delta\bar{a})$

(ii) For every atomic formula $\varphi(\bar{x})$ and tuple $\bar{a} \in A$ of sufficient length, if $A \Vdash \varphi[\bar{a}]$, then $B \Vdash \varphi[\delta\bar{a}]$

Proof Sketch.
Simply leverage the definitions above

3.2 Submodels

Submodel
A is a submodel of B, often notated $A \subseteq B$, if and only if

- A and B are both L-models
- $dom(A) \subseteq dom(B)$
- For all constants in L, $c^A = c^B$
- For each $n > 0$, each n-ary relation R in L, and each n-tuple $\bar{a}$ in A, $\bar{a} \in R^A \Leftrightarrow \bar{a} \in R^B$.
- For each $n > 0$, each n-ary function symbols f in L, and each n-tuple $\bar{a}$ in A, $f^A(\bar{a}) = f^B(\bar{a})$.

or, equivalently, A and B are L-models with $\mathrm{dom}(A) \subseteq \mathrm{dom}(B)$ and the inclusion map $i : \mathrm{dom}(A) \to \mathrm{dom}(B)$ is an embedding.

► **Lemma 3.1.**
Let B be an L-model and $X \subseteq \text{dom}(B)$. X is the domain of a submodel of B if and only if

- For all constants c, $c^B \in X$.
- For every $n > 0$, every n-ary function symbol f of L, and every n-tuple $\bar{a}$ from X, $f^B(\bar{a}) \in X$.

Proof Sketch.
Simply leverage definitions for both directions.

Generated Submodel
Let A be a submodel of B and Y a set of elements of B. *A is the submodel of B generated by Y*, notated $A = \langle Y \rangle_B$, if and only if it is the smallest submodel of B which contains Y. $\langle Y \rangle_B$ is also called the *hull* of Y in B; we call Y itself a *set of generators* for A.

Less formally, a generated submodel is the result of taking all the elements in the provided set, all the constants, and closing under functions (apply the functions a step at a time to generate a hierarchy, the union is the generated submodel).

Finitely Generated
A submodel B is finitely generated if and only if it is of the form $\langle Y \rangle_A$ for some finite set Y of elements and L-model A .

► **Theorem 3.3** (A Bound on Generated Submodels)**.**
Let B be an L-model and $Y \subseteq \text{dom}(B)$. Then, $|\langle Y \rangle_B| \leq |Y| + |L|$.

Proof Sketch.
It's easy to show that Y union an additional individual for every constant symbol in L has cardinality at worst $|Y|+|L|$; similarly, the set of functions from L applied to elements of Y can be seen to have cardinality at most $|Y| + |L| + \aleph_0$. Altogether, then, the size of the domain of $\langle Y \rangle_B$ is–using the lemma above–bounded by $|Y| + |L| + |Y| + |L| + \aleph_0$. But this is equivalent to, $\max(|Y|, |L|, |Y|, |L|, \aleph_0)$ which is, of course, simply $\max(|Y|, |L|, \aleph_0)$. Recall, however, that $|L| \geq \aleph_0$, and so this is equivalent to $\max(|Y|, |L|) = |Y| + |L|$

3.3 Preserving Formulas

Thus far, our relations between models have focused on connections between the various parts of a model–not the fact the models make any sentence of the corresponding language either true or false. Taking this approach offers a new way of looking at several old relations, as well as a trio of entirely new ones. On the very broadest level, we could classify maps as to whether or not they *preserve* particular sets of formulas; that is, given a language $\mathcal{L}$, two L-models A and B, a set of formulas $\Phi(\bar{x})$, and a function $\delta : A \to B$, δ is called a Φ-map if and only if it preserves all the formulas in $\Phi(\bar{x})$. Of course, not all sets of formulas are equally interesting; we begin by formalizing the notion of preservation and presenting the most ubiquitous classes of formulae.

Preservation
Let $\mathcal{L}$ be a language and A, B L-structures. A function $\delta : A \to B$ preserves a formula $\varphi(\bar{x})$ of $\mathcal{L}$ if and only if for every $\bar{a} \in A$, $A \Vdash \varphi[\bar{a}] \Rightarrow B \Vdash \varphi[\delta\bar{a}]$

Using our previous work, It follows immediately that homomorphisms preserve atomic formulas while embeddings preserve both atomic formulas and their negations (literals). Isomorphisms, naturally, preserve all formulae.

3.3.1 ∀ and ∃ Preservation

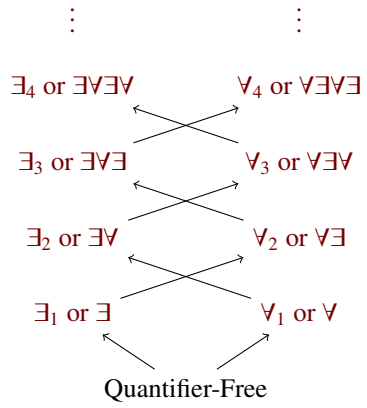
Existential
A formula is *existential*, an $\exists_1$-formula (pronounced 'E-1'), or simply an $\exists$-formula if and only if it is built out of quantifier free formulas using only $\exists$, $\wedge$, and $\vee$. In other words, it can be written in prenex normal form as $\exists \bar{x}\, [\varphi(\bar{x})]$.

Universal
A formula is *universal*, an $\forall_1$-formula (pronounced 'A-1'), or simply an $\forall$-formula if and only if it is built out of quantifier free formulas using only $\forall$, $\wedge$, and $\vee$. In other words, it can be written in prenex normal form as $\forall \bar{x}\, [\varphi(\bar{x})]$.

As the subscripts hint, $\exists_1$ and $\forall_1$ formulas are, in fact, members of a natural hierarchy on formulas; while we will use these two most often, the full context may make our work easier to understand:

Quantifier Hierarchy

- $\exists_0$ and $\forall_0$ are defined as the quantifier-free formulas.
- A formula is in $\forall_{n+1}$ if and only if it is in the smallest class of formulas which contains the $\exists_n$ formulas and is closed under $\wedge$, $\vee$, and adding $\forall$ to the front.
- A formula is in $\exists_{n+1}$ if and only if it is in the smallest class of formulas which contains the $\forall_n$ formulas and is closed under $\wedge$, $\vee$, and adding $\exists$ to the front.



► **Theorem 3.4** (Embeddings Preserve ∃-Formulas Upwards)**.**
Let $\mathcal{L}$ be a first-order language and L the signature associated with it. If A, B are L-models and $\delta : A \to B$ is an embedding, then δ preserves all $\exists$-formulas.

Proof Sketch.
By induction on the complexity of formulas; note that any existential formula can be written in the form $\exists \bar{x} \varphi(\bar{x})$ where φ is in CNF or DNF. The base case of the induction is atomic and negated atomic formulas; from here, it's easy to see that $\wedge$ and $\vee$ are also preserved. The $\exists$ case is slightly harder and relies on the definition of an embedding.

► **Corollary** (Submodels Preserve ∀-Formulas Downwards)**.**
Let $\mathcal{L}$ be a first-order language, L the signature associated with it, and A, B L-models. Then,

(i) If A is a submodel of B and φ is an $\forall$-formula, then for every $\bar{a} \in A$, $B \Vdash \varphi[\bar{a}] \Rightarrow A \Vdash \varphi[\bar{a}]$.

(ii) If T is an $\forall$-theory, $B \Vdash T$, and A is a submodel of B, then $A \Vdash T$ as well.

■ **Notation.** For a theory T in a language $\mathcal{L}$, the theory containing all of the $\forall$-sentences of $\mathcal{L}$ which are consequences of T is denoted $T_\forall$.

► **Lemma 3.2** ($T_\forall$-Models are Submodels of T-models)**.**
If T is a theory in a first-order language L, then the $T_\forall$-models are precisely the submodels of the T-models.

Proof Sketch.
($\Leftarrow$)
Simply leverage the previous preservation results.

($\Rightarrow$)
Suppose $A \Vdash T_\forall$ for some theory T. We show the existence of a B such that $B \Vdash T$ and $A \subseteq B$. Let U be the set of all $\exists$-sentences φ such that $A \Vdash \varphi$. Assume $T \cup U \cup \mathrm{Diag}_{\mathrm{at}}(A)$ doesn't have a model. Then, by compactness, there is some finite set $F \subseteq U$ such that $T \cup \mathrm{Diag}_{\mathrm{at}}(A) \models \neg\bigwedge F$. Using compactness again, there is some finite set $E \subseteq \mathrm{Diag}_{\mathrm{at}}(A)$ such that $T \models \bigwedge E \rightarrow \neg\bigwedge F$. But, by the definition of U, $\neg\bigwedge F$ is logically equivalent to an $\forall$-formula θ, and so $T \models \bigwedge E \rightarrow \theta$–but then $\bigwedge E \rightarrow \theta$ as a whole is a $\forall$-sentence (recall, $\bigwedge E$ must be quantifier-free). Thus, $\bigwedge E \rightarrow \theta \in T_\forall$ and, by assumption, $A^+ \Vdash \bigwedge E \rightarrow \theta$. Therefore, $A^+ \Vdash \theta$ and $A \Vdash \neg\bigwedge F$–contradicting the definition of U. By reductio, $T \cup U \cup \mathrm{Diag}_{\mathrm{at}}(A)$ must have a model, and the reduct of this model to our original language fulfills the stipulations for our B.

► **Theorem 3.5** (Łoś-Tarski Preservation Theorem)**.**
Let $\mathcal{L}$ be a first-order language, T a theory in $\mathcal{L}$, and $\Phi(\bar{x})$ a set of L-formulas ($\bar{x}$ need not be finite); then, the following are equivalent:

(i) For any two L- models A, B such that $A \Vdash T$, $B \Vdash T$, and $A \subseteq B$, $\Phi(\bar{x})$ is preserved downward from B to A; that is, for every $\bar{a} \in A$, if $B \Vdash \bigwedge\Phi[\bar{a}]$, then $A \Vdash \bigwedge\Phi[\bar{a}]$ as well

(ii) For some set of $\forall$-formulas $\Psi(\bar{x})$ ($\bar{x}$ need not be finite), $T \models \forall\bar{x}[\bigwedge\Phi(\bar{x}) \leftrightarrow \bigwedge\Psi(\bar{x})]$; that is, Φ is equivalent modulo T to a set of $\forall$-formulas

Proof Sketch.
($\Leftarrow$)
Simply leverage the earlier $\forall$-preservation results.

($\Rightarrow$)
The case where $\Phi(\bar{x})$ is a theory follows easily by the previous proposition. Consider, then, the case where $\Phi(\bar{x})$ is not a theory. Let $\bar{c}$ be a tuple of new constants (expanding the language) and note that $\Phi(\bar{c})$ is now a theory. Note further that, like the more general $\Phi(\bar{x})$, $\Phi(\bar{c})$ is preserved downwards between all models A, B such that $A \Vdash T$, $B \Vdash T$, and $A \subseteq B$. Let $\Psi(\bar{c})$ denote $\Phi(\bar{c})_\forall$. It's trivial that $T \cup \Phi(\bar{c}) \models \Psi(\bar{c})$. Consider $T \cup \Psi(\bar{c})$, by the definition of Ψ and the previous lemma, any model of $T \cup \Psi(\bar{c})$ is submodel of a $\Phi(\bar{c})$ model–and thus itself a $\Phi(\bar{c})$ model. It follows immediately that $T \cup \Psi(\bar{c}) \models \Phi(\bar{c})$, and so

$$T \models \Phi(\bar{c}) \leftrightarrow \Psi(\bar{c})$$

Noting that $\bar{c}$ occurs nowhere in T,

$$T \models \forall\bar{x}[\Phi(\bar{x}) \leftrightarrow \Psi(\bar{x})]$$

Note that, in the theorem above, $\bigwedge\Phi(\bar{x})$, $\bigwedge\Psi(\bar{x})$, and $\forall\bar{x}[\bigwedge\Phi(\bar{x}) \leftrightarrow \bigwedge\Psi(\bar{x})]$ need not actually be well-formed formulas of $\mathcal{L}$. Nonetheless, their meanings are still clear. More colloquially, the Łoś-Tarski preservation theorem states that the formulas preserved downwards by the submodel relation are always exactly the $\forall_1$ formulas (any other formula preserved downwards by the relation is just equivalent to an $\forall_1$ formula). This, of course, quickly gives rise to the following corollary:

► **Corollary** (Embeddings Preserve Exactly $\exists_1$ Upwards)**.**
If $\mathcal{L}$ is a first-order language, T is an L-theory, and $\varphi(\bar{x})$ is an L-formula, then the following are equivalent:

(i) $\varphi(\bar{x})$ is preserved upwards by embeddings between models of T

(ii) $\varphi(\bar{x})$ is equivalent modulo T to an $\exists$-formula

3.3.2 Elementary Preservation

Arguably the most important class for preservation, however, isn't found in the quantifier hierarchy; rather, it is the class of all first-order formulas of a language.

Elementary Embedding
For a first-order language $\mathcal{L}$, a function $\delta : A \to B$ is an *elementary embedding* if and only if for every formula $\varphi(\bar{x})$ of $\mathcal{L}$ and every $\bar{a} \in A$, $A \Vdash \varphi[\bar{a}] \Leftrightarrow B \Vdash \varphi[\delta\bar{a}]$. In less formal terms, an elementary embedding preserves the truth of all formulas applied to elements of A.

Elementary Submodel
Let A, B be L-structures. A is an *elementary submodel* of B or B is an *elementary extension* of A–notated $A \preceq B$–if and only if $A \subseteq B$ and the inclusion map is an elementary embedding (that is, an *elementary inclusion*)

► **Theorem 3.6** (Vaught-Tarski Criterion)**.**
Let $\mathcal{L}$ be a first-order language, A, B L-structures, and $A \subseteq B$, then the following are equivalent:

(i) $A \preceq B$

(ii) For every formula $\varphi(\bar{x}, y)$ of $\mathcal{L}$ and every tuple $\bar{a}$ from A, if $B \Vdash \exists y\, \varphi(\bar{a}, y)$, then $B \Vdash \varphi(\bar{a}, d)$ for some element d from A

Proof Sketch.
($\Rightarrow$)
Simply leverage the definitions

($\Leftarrow$)
Induction on formulas; the un-quantified cases all follow easily using vacuous existentials. For the $\forall$ case, note that (ii) is easily extended to an if and only if and route through existentials.

That is, being an elementary submodel–a submodel which preserves all formulas–is encompassed simply by the inclusion map preserving, for every n, the $\exists_n$ formulas downwards.

3.4 Chains of Models

Chain
Let L be a signature, γ a cardinal, and $(A_i : i < \gamma)$ a sequence of L-structures. The sequence $(A_i : i < \gamma)$ is a *chain* if and only if for every i, j such that $i < j < \gamma$, $A_i \subseteq A_j$.

Union of a Chain
Given a chain of models $(A_i : i < \gamma)$, the *union of* $(A_i : i < \gamma)$ is the L-model $A_\infty = \bigcup_{i<\gamma} A_i$ defined as follows:

- $\mathrm{dom}(A_\infty) = \bigcup_{i<\gamma} \mathrm{dom}(A_i)$
- For any constant symbol c, $c^{A_\infty} = c^{A_i}$, for any $i < \gamma$
- For any n-ary relation symbol R, $\bar{a} \in R^{A_\infty}$ if and only if there is an A_i such that $\bar{a} \in R^{A_i}$
- For any n-ary function symbol f, define $f^{A_\infty}(\bar{a}) = f^{A_i}(\bar{a})$ where A_i is any model in the chain containing $\bar{a}$

Preservation Under Chains
Given a chain $(A_i : i < \gamma)$ of L-models, a formula $\varphi(\bar{x})$ is preserved under the chain $(A_i : i < \gamma)$ if and only if whenever there is $\bar{a} \in A_0$ such that $A_i \Vdash \varphi[\bar{a}]$ for every $i < \gamma$, then $A_\infty \Vdash \varphi[\bar{a}]$ as well.

► **Theorem 3.7** (Chains Preserve ∀∃)**.**
Let $\mathcal{L}$ be a first-order language and L the associated signature; if $\varphi(\bar{x})$ is an $\forall\exists$ formula, then $\varphi(\bar{x})$ is preserved under chains.

3.4.1 Elementary Chains of Models

Elementary Chain
A chain of L-structures $(A_i : i < \gamma)$ such that the inclusion maps are all elementary inclusions.

► **Theorem 3.8** (Unions of Elementary Chains are Elementary Extensions)**.**
If $(A_i : i < \gamma)$ is an elementary chain of L-structures, then $\bigcup_{i<\gamma} A_i$ is an elementary extension of every A_i.

Proof Sketch.
Use the Łoś-Tarski theorem and induction on formulas; by the construction of A_∞, one can argue that the witness appears at some point in the chain and by the transitivity of elementary models/induction hypothesis, that it a witness appears in the one under consideration. Thus, by the inductive hypothesis, this last witness appears in A_∞.

3.5 Reduction and Expansion

Reduct, Expansion
Let L, L^+ be signatures such that $L \subseteq L^+$. Let A be an L-model, B an L^+-model such that the interpretations of the symbols in L coincide in A and B. A is called an *L-reduct of B*–in other words, A is $B \upharpoonright L$–while B is a L^+-expansion of A.

► **Lemma 3.3** (Elementary Diagram Lemma)**.**
Suppose $\mathcal{L}$ is a first-order language, A, B are L-structures, $\bar{c}$ is a tuple of distinct constants not in L, $(A, \bar{a})$ and $(B, \bar{b})$ are $L(\bar{c})$ structures, and $\bar{a}$ generates A. Then the following are equivalent:

- For every formula $\varphi(\bar{x})$ of $\mathcal{L}$, if $(A, \bar{a}) \Vdash \varphi(\bar{c})$, then $(B, \bar{b}) \Vdash \varphi(\bar{c})$
- There is an elementary embedding $\delta : A \to B$ such that $\delta\bar{a} = \bar{b}$

Note that, by the forward direction from the first to second statement, for any model D of the elementary diagram of A in $\mathcal{L}$, there is an embedding of A in the reduct $D|L$.

> $\forall_1$ **Preservation**
> Let $\mathcal{L}$ be a formal language, L the associated signature, and A, B L-models. $A \preceq_1 B$ if and only if every $\forall_1$ formula $\varphi(\bar{x})$ is preserved from A to B.

► **Lemma 3.4.**
Let $\mathcal{L}$ be a formal language, L the associated signature. For any theory U and L-model G, if $G \Vdash U$, then there exists an L-model D such that $D \Vdash T$ and $G \preceq_1 D$.

► **Lemma 3.5.**
Let $\mathcal{L}$ be a formal language, L the associated signature, and G, D L-models; if $G \preceq_1 D$, then there exists an L-model E such that $D \subseteq E$ and $G \preceq E$.

Proof Sketch.
Let T' be the union of the quantifier free formulas of B and elementary diagram of A. Suppose that T' is not consistent; noting that constants from one do not appear in the other, derive a contradiction. By compactness, T' has a model.

► **Theorem 3.9.**
Let T be a theory. The following are equivalent:

- For every chain of structures $(B_i : i < \gamma)$ if $B_i \Vdash T$ for all i, then $B_\infty \Vdash T$ as well.
- There is an $\forall_2$-theory U such that $\models \bigwedge T \leftrightarrow \bigwedge U$.

Proof Sketch.
($\to$) Let $U = \{\psi$ is a $\forall_2$ sentence such that $T \models \psi\}$. Clearly, $T \models \bigwedge U$. From here, we'll build, starting with $A \models U$ and using the lemmas above, an elementary chain $A \preceq A_1 \preceq A_2 \ldots$ where each $A_i \preceq_1 B_i \subseteq A_{i+1}$. Since T is preserved under chains, $A_\infty \models T$ and $A \preceq A_\infty$, $A \models T$ by the theorem above.

Chapter 4

Compactness and Strong Completeness

As stated above, every structure has a first order theory–namely, $\mathrm{Th}(A)$; but, quite obviously, not every theory has a first order structure (simply consider a theory with two contradictory sentences). Earlier, canonical models provided a means for constructing a model of any set of atomic sentences; in this section, we first consider what constraints are needed to guarantee the existence of a model of an arbitrary theory, then use this result to prove two major theorems for first-order languages.

4.1 Hintikka Sets

Hintikka Set
A theory T in a language $\mathcal{L}$ is a Hintikka set if and only if

(H1) For every atomic sentence φ, if $\varphi \in T$, then $\neg\varphi \notin T$.

(H2) For every closed term t of L, $t \approx t \ \in T$

(H3) For all closed terms s, t, and atomic formula $\varphi(\bar{x})$ if $\varphi[s] \in T$ and $s \approx t \ \in T$, then $\varphi[t] \in T$ as well.

(H4) If $\neg\neg\varphi \in T$, then $\varphi \in T$

(H5) If $\bigwedge\Phi \in T$, then $\Phi \subseteq T$. If $\neg\bigwedge\Phi \in T$, then $\neg\varphi \in T$ for some $\varphi \in \Phi$.

(H6) If $\bigvee\Phi \in T$, then there is a $\varphi \in \Phi$ such that $\varphi \in T$. If $\neg\bigvee\Phi \in T$, then $\neg\varphi \in T$ for all $\varphi \in \Phi$.

(H7) If $\forall x\, \varphi(x) \in T$, then $\varphi(t) \in T$ for every closed term t of $\mathcal{L}$. If $\neg\forall x\, \varphi(x) \in T$, then $\neg\varphi(t) \in T$ for some closed term t of $\mathcal{L}$

(H8) If $\exists x\, \varphi(x) \in T$, then $\varphi(t) \in T$ for some closed term t of $\mathcal{L}$. If $\neg\exists x\, \varphi(x) \in T$, then $\neg\varphi(t) \in T$ for every closed term t of $\mathcal{L}$

The condition (H7) above is known as the *Henkin property*–after a famous logician–and the closed term t in the property which makes $\exists x\varphi(x)$ true is commonly called a *witness* to $\exists x\varphi(x)$; it's often said in this context that t witnesses the truth of $\exists x\varphi(x)$.

► **Theorem 4.1** (Hintikka Sets have Models)**.**
If T is a Hintikka set, then the canonical model A of the set of atomic sentences in T is a model of T.

Proof Sketch.
Let U be the set of atomic sentences in T. Observe that, by definition, U is equality closed. So, by a previous theorem, the canonical model A is such that $A \Vdash U$. By construction U is the set of atomic sentences in A and every $a \in A$ is of the form t^A for some closed term. To show that $A \Vdash T$, argue by an induction on formulas using the Hintikka properties.

► **Theorem 4.2** (Finite Satisfiability, Completeness, and Witnesses)**.**
Let T be a theory in a first-order language $\mathcal{L}$. If

(i) Every finite subset of T has a model

(ii) For every sentence φ of $\mathcal{L}$, either φ or $\neg\varphi$ is in T

(iii) For every sentence $\exists x\, \psi(x) \in T$, there exists a closed term t of $\mathcal{L}$ such that $\psi[t] \in T$ (that is, t is a witness for $\exists x\, \psi(x)$)

Then, T is a Hintikka set.

Proof Sketch.
Simply prove that each clause of the Hintikka set definition holds.

The conditions $(i) - (iii)$ above are commonly referred to as finite satisfiability, completeness, and having witnesses respectively.

4.2 Compactness for First-Order Logic

Among model theorists, few theorems are as revered or as useful as compactness; compactness itself establishes a fundamental limit to the expressive capabilities of first-order logic: only finite things are distinguishable. The infinite all blurs together.

► **Theorem 4.3** (Compactness for First-Order Logic)**.**
Let T be a theory of a first-order language $\mathcal{L}$. If every finite subset of T has a model, then T has a model.

Proof Sketch.
Show the following:

- Extension Lemma: If a theory Γ is finitely satisfiable, then Γ has an extension Γ^+ in an expanded first-order language $\mathcal{L}^+$ which is finitely satisfiable, complete, and has witnesses.

Then, argue as follows. Let Γ be finitely satisfiable. Then, by the lemma, there is a Γ^+ in an expanded language which extends it and is Hintikka (an earlier theorem). By previous work, Γ^+ has a model A because it is a Hinikka set; thus, Γ, a subtheory of Γ^+, also has a model–namely $A \restriction \mathcal{L}$.

► **Lemma 4.1** (Extending Finitely Satisfiable Theories)**.**
If a theory Γ in a first-order, countable language is finitely satisfiable, then Γ has an extension Γ^+ which is finitely satisfiable, complete, and has witnesses.

Proof Sketch.
Add $\aleph_0$ many new constants to L, the signature in use; enumerate the sentences of $\mathcal{L}(\aleph_0)$, noting that it is possible to be so since $|\mathcal{L}| \leq |\mathcal{L}(\aleph_0)| = \aleph_0$. Induct starting with T, adding either a sentence or its negation at each step according to whichever is finitely satisfiable with the current theory (note that you must prove it is always possible to add at least one these). In addition, if the added sentence is an existential $\exists x[\varphi(x)]$, also add $\varphi[c_i]$, where c_i is the least constant of the $\aleph_0$ that were originally added which has not yet appeared in any sentence of our theory.

4.3 Strong Soundness and Completeness for First-Order Logic

Although our focus thus far has been exclusively on the model-theoretic logical consequence relation, $\models$, it will be useful to occasionally pair it with a proof theoretic counterpart, $\vdash$. This proof theoretic counterpart can be derived from any of the standard systems; the actual choice doesn't matter for our purposes.

> $\Gamma \vdash \varphi$
> For a theory Γ and sentence φ, $\Gamma \vdash \varphi$–read 'Γ proves φ'–if and only if there is a proof from the sentences of Γ to φ.

► **Theorem 4.4** (Soundness)**.**
If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.

Proof Sketch.
Soundness, as usual, is easily established simply by showing that each of the proof rules of a given system preserves the truth in a model relation, $\Vdash$.

> **Consistent**
> A set of sentences Γ is consistent if and only if $\Gamma \nvdash \bot$ where $\bot$ can be taken as either a primitive that is always false or an abbreviation for $\exists x(x = x) \wedge \neg\exists x(x = x)$.

► **Theorem 4.5** (Completeness)**.**
If $\Gamma \models \varphi$, then $\Gamma \vdash \varphi$.

Proof Sketch.
With clever usage of compactness, it is enough to show that whenever a theory is consistent, it is satisfiable. Assume $\Gamma \models \varphi$; if Γ is inconsistent, then $\Gamma \vdash \varphi$ trivially. To see this, assume, then, that Γ is consistent. If whenever a theory is consistent, it is satisfiable, we now have that Γ is satisfiable. Consider, then, $\Gamma \cup \{\neg\varphi\}$. By assumption, this is unsatisfiable, and so–by compactness–there is a finite subtheory Γ^- of Γ such that $\Gamma^- \cup \{\neg\varphi\}$ is unsatisfiable. Then, by above, $\Gamma^- \cup \{\neg\varphi\}$ is inconsistent. Utilizing the reductio rule of the proof system (or whatever combination of rules is equivalent to it), we now have a proof establishing $\Gamma \vdash \varphi$.

To prove that whenever a theory is consistent, it is satisfiable, first prove that if Γ is consistent, then Γ has an extension Γ^+ in an expanded language which is consistent, complete, and has witnesses. From here, we have that Γ^+ is Hintikka, and thus has a model–making it satisfiable. It follows immediately that Γ is satisfiable (as earlier, simply take the reduct).

Chapter 5

Definability

In model theory, definability comes in two distinct forms: the definability of a relation and the definability of a class of models. Each notion is important in its own right, and so we consider each in turn. The primary distinction between the two is simply that the former encompasses relations relative to a particular model while the latter describes classes of models, and is thus relative to only a particular language.

5.1 Definability of Relations

To say an n-ary relation is definable in a some $\mathcal{L}$-model M simply means that there is a formula of $\mathcal{L}$ such that all and only those tuples of elements in the relation make the formula true; more colloquially, there is a formula that *describes* the relation. Notationally, $\varphi(A^n)$ is intended as shorthand for the following set: $\{\bar{a} \mid A \Vdash \varphi[\bar{a}]\}$. Of course, this notation naturally extends to include parameters as well: if $\psi(x_0, \ldots, x_{n-1}, \bar{y})$ is an L-formula and $\bar{b}$ is a tuple from A, then $\psi(A^n, \bar{b}) = \{\bar{a} \mid A \Vdash \psi[\bar{a}, \bar{b}]\}$. Using the above, a formal definition for definable relations is given below:

> **First-Order Definable Relation without Parameters**
> Let $\mathcal{L}$ be a first-order language and A an L-model. An n-ary relation R is *first-order definable without parameters* or $\emptyset$*-definable* (pronounced 'zero-definable') if and only if there exists an $\mathcal{L}$-formula $\varphi(\bar{x})$ such that $R = \varphi(A^n)$

Expanding on this basic definition, it's also possible to define a weaker notion:

> **First-Order Definable with Parameters**
> Let $\mathcal{L}$ be a first-order language, A an L-structure, and $X \subseteq \mathrm{dom}(A)$ a set of parameters. An n-ary relation R is *first-order definable with parameters* or X*-definable* if and only if there exists an $\mathcal{L}$-formula $\varphi(\bar{x}, \bar{y})$ such that $R = \varphi(A^n, \bar{b})$ for $\bar{b} \in X$.

It may be easiest to think of first-order definability with paramters as simply definability in an expanded signature with constants for all the elements in the parameter set S. In general, proving that a relation is definable is accomplished by exhibiting the formula which defines the relation; proving that a relation is not definable by invoking a connection between automorphisms and definable relations:

► **Proposition 5.1** (Fixed by Automorphisms ⇔ Definable Relation)**.**
Let M be an $\mathcal{L}$-model and R an n-ary relation over M. R is first-order definable if and only if R is fixed set-wise by every automorphism of M

Thus, to show a relation is nondefinable, it suffices to exhibit an automorphism which either moves an element of relation out of the relation or moves an element not in the relation into it. As one last note, an element a of a model is said to be definable if and only if the relation $\{a\}$ is. All of the above, thus, carries over directly into talk about particular elements.

Example 3.
For simple examples of definable relations, consider:

(i) $\emptyset$ and the domain in any model

(ii) Zero in $\langle \mathbb{Z}, + \rangle$

(iii) $>$ in $\langle \mathbb{N}, +, 0 \rangle$

For simple examples of nondefinable relations, consider:

(i) Zero in $\langle \mathbb{Z}, < \rangle$

(ii) $x_1 + 3 = x_2$ in $\langle \mathbb{N}, +, S, 0 \rangle$

5.2 Definable Classes of Models

One of the topics of preeminent concern in logic is what can be said in a particular logical system–the topic of this section.

Basic Elementary Definable
A class of models K is *basic elementary definable* (abbreviated, EC) if and only if there is a sentence φ such that K is the class of all models which make φ true; that is, $K = \text{Mod}(\varphi)$.

Elementary Definable/Axiomatizable
A class of models K is *elementary definable* or *elementary axiomatizable* (abbreviated, EC_Δ) if and only if there is a theory T such that K is the class of all models which make T true; that is, $K = \text{Mod}(T)$.

Closely connected to these is the theory of a class of models:

Theory of a Class of Models
Let K be a class of L-models. The *theory of* K in the language $\mathcal{L}$–notated $\text{Th}_{\mathcal{L}}(K)$–is the set of all sentences of $\mathcal{L}$ that are true in every model in K.

Note that, for any class K, $\text{Th}(K)$ is the largest possible theory which could define K; by simply leveraging the definitions above, it's easy to show that if any theory defines K, then $\text{Th}(K)$ does.

While the primary method for showing definability (either elementary or basic elementary) remains simply exhibiting the defining sentence or theory, showing non-definability proves much more difficult. Given a class K, the standard strategy is to show that $\text{Th}(K)$ union a theory which–as a whole–guarantees a model is not in K is finitely satisfiable by models in K. A simple application of compactness then gives the desired result. The most famous and important result along this line is owed to Thoralf Skolem.

5.3 The Skolem Theorems

Thoralf Skolem was a Norwegian logician who wrote prolifically during the early parts of the 20th century. Skolem was, among other things, one of the detractors from the use of so-called 'uncountable' infinites and worked to dissuade others from using them. One of Skolem's major results in this vein was showing that there existed countable models of the 'uncountable' real numbers; how Skolem did so is our topic for this section.

5.3.1 Skolemization

Skolemization
Let T be a theory of a first-order language $\mathcal{L}$. The *Skolemization* of T is a theory T^+ in a first-order language $\mathcal{L}^+$ where $T \subseteq T^+$, $L \subseteq L^+$, and

(i) Every L-model A such that $A \Vdash T$ can be extended to an L^+-model A^+ such that $A^+ \Vdash T^+$.

(ii) For every formula $\varphi(\bar{x}, y)$ of $\mathcal{L}^+$ with $\bar{x}$ non-empty, there is a term $t(\bar{x})$ of $\mathcal{L}^+$ such that $T^+ \models \forall \bar{x}[\exists y\, \varphi(\bar{x}, y) \to \varphi(\bar{x}, t(\bar{x}))]$

Skolem Terms
The terms $t(\bar{x})$ of $\mathcal{L}^+$ in condition (ii) above. These terms are usually generated by simply adding, for every formula $\varphi(\bar{x}, y)$, a new function f_φ to the signature which takes $\bar{x}$ and returns the witness y if it exists; in this case, the f_φ are called *Skolem functions*

Skolem Theory
A theory T is a *Skolem theory* or–equivalently–*has Skolem functions* if and only if T is a Skolemization of itself.

Note that being a Skolem theory is relative to the language in use; if T is a Skolem theory under $\mathcal{L}$, it will usually not be a Skolem theory under $\mathcal{L}^+$ where $\mathcal{L} \subseteq \mathcal{L}^+$.

► **Theorem 5.1** (Skolemization Theorem)**.**
If $\mathcal{L}$ is a first-order language, then there exists $\mathcal{L}^\Sigma$ a language such that $\mathcal{L} \subseteq \mathcal{L}^\Sigma$ and Σ a theory of $\mathcal{L}^\Sigma$ such that

- Every L-model A can be expanded to a model A^Σ of Σ
- Σ is a Skolem theory in $\mathcal{L}^\Sigma$
- $|\mathcal{L}^\Sigma| = |\mathcal{L}|$

Proof Sketch.
For every formula $\varphi(\bar{x}, y)$, create a new function symbol f_φ with arity $|\bar{x}|$ and add these new functions to the signature. Make $\Sigma = \{\forall \bar{x}[\exists y \varphi(\bar{x}, y) \to \varphi(\bar{x}, f_\varphi(\bar{x}))] : \varphi(\bar{x}, y)$ is an L-formula$\}$. Union and repeat.

► **Corollary.**
Let T be a theory in a first-order language $\mathcal{L}$. Then, T has a Skolemization T^+ in a first-order language $\mathcal{L}^+$ with $|\mathcal{L}^+| = |\mathcal{L}|$.

5.3.2 The Downward Löwenheim-Skolem Theorem

Skolem Hull
Let T be a Skolem theory in a first-order language $\mathcal{L}$, A a T-model, and X a subset of the elements of A. The *Skolem Hull* of X is $\langle X \rangle_A$, the submodel of A generated by X.

► **Theorem 5.2** (Skolem Hulls are Elementary Submodels)**.**
Let T be a Skolem theory in a first-order language $\mathcal{L}$. If A is an T-model and X is a set of elements from A such that the Skolem hull of X, $\langle X \rangle_A$, is non-empty, then $\langle X \rangle_A \preceq A$.

Proof Sketch.
Use the Tarski-Vaught criterion, noting that Skolemization is specifically designed to force the presence of witnesses in the submodel.

► **Theorem 5.3** (Downward Löwenheim-Skolem Theorem)**.**
Let $\mathcal{L}$ be a first-order language, B be an infinite L-structure, and $X \subseteq B$. Then, there is an L-structure A such that:

- $X \subseteq \text{dom}(A)$
- $A \preceq B$
- $|A| = \max(|X|, |L|, \aleph_0)$

Proof Sketch.
Simply leverage the work above; expand A to A^Σ use the Skolemization theorem, then take the Skolem hull around X.

5.3.3 The Upward Löwenheim-Skolem Theorem

Rather than simply proceed to the theorem immediately, we first consider two results of gradually increasing strength, but similar technique to our eventual result.

► **Proposition 5.2** (Bigger Models Always Exist)**.**
Let T be a theory which has at least one infinite model. Then, for every κ, there is a model of T of size $\geq \kappa$.

Proof Sketch.
Just add κ constants to the signature and sentences saying that no two constants name the same element (pairwise distinct) to the theory T. Prove this expanded theory is finitely satisfiable and invoke compactness; finally, take the reduct to the original signature.

- **Fact.** If $B \Vdash \text{Diag}_{\text{at}}(A)$, then A can be embedded into B; simply consider $f(a) = c_a^B$.

► **Proposition 5.3** (Bigger Extensions Always Exist)**.**
Let T be a theory and let A be an infinite model such that $A \Vdash T$. Then, for every cardinal κ, there is a T-model extending B ($A \subseteq B$) such that $|B| \geq \kappa$.

Proof Sketch.
As earlier, introduce κ new constants to the signature, then add sentences asserting these constants are all pairwise distinct from one another to T. In addition, however, also add $\text{Diag}_{\text{at}}(A)$ to T. Prove finite satisfiability, invoke compactness, take the reduct.

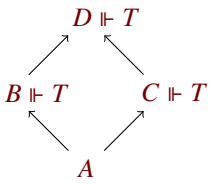
► **Theorem 5.4** (Upward Löwenheim-Skolem)**.**
For any infinite L-structure A and any cardinal κ such that $|A| \leq \kappa$ and $|L| \leq \kappa$, there is a model B such that $A \preceq B$ and $|B| = \kappa$.

Proof Sketch.
As earlier, add κ new constants to L, generating a new language $\mathcal{L}^+$. Take $\text{Diag}_{\text{el}}(A)$ union sentences asserting that the new constants are pairwise disjoint and show that this is finitely satisfiable. Next, prove that $A \preceq B \upharpoonright L$. While it's very possible that $B \upharpoonright L$ is too big, simply leverage the downward Löwenheim-Skolem theorem.

Chapter 6

Amalgamation

Amalgamation is a process centered on the following diagram:



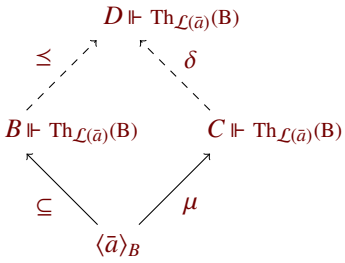
Each of A, B, C, and D represents a model while the arrows signify embeddings. The model of primary interest in the diagram is A; in order to investigate a model, it's natural to view the ways in which it can be extended–hence B and C. But even more information can be gained by looking not only at the extensions themselves, but also how those extensions of A relate to one another in a model containing both of them: hence, D.

Of course, B and C are seldom allowed to be any extension whatsoever; we will assume there is a theory T such that $B \Vdash T$ and $C \Vdash T$ which we desire to preserve in the amalgam of B and C–D above. A final constraint imposed on the process of amalgamation is, again, quite natural; since the entire process is used to investigate the model A, the embeddings from B and C into D had better agree on A. In this context, D is said to be an *amalgam of B and C over A*. As we will show, first-order logic has a number of amalgamation properties, general processes which–under specific assumptions–guarantee the existence of an amalgam with certain properties. As it turns out, these properties can be leveraged to a variety of uses beyond the investigation of A, the interpolation and definability theorems in the last section among them.

6.1 Elementary Amalgamation Property

The most basic amalgamation process takes, for $\bar{a}$ a set of generators of A, the theory T above to be the set of all sentences in the language $\mathcal{L}(\bar{a})$–immediately giving that B, C, and D are all $\mathcal{L}(\bar{a})$-equivalent to one another. In fact, we will guarantee an even stronger condition: the embeddings from B and C into D will be elementary. Because we can, we may as well take A to be a submodel of B and B to be a submodel of D, making the relations along the left side of the diagram $\subseteq$ and $\preceq$ respectively. Note, however, that doing so only makes things slightly simpler and doesn't constitute an important extension of our original diagram.

► **Theorem 6.1** (Elementary Amalgamation Property)**.**
Let $\mathcal{L}$ be a first-order language, let B, C be L-structures, and let $\bar{a}$, $\bar{c}$ be (possibly infinite) tuples of elements of B, C such that $(B, \bar{a}) \equiv (C, \bar{c})$. Then, there exists an amalgam D of B and C over $\langle \bar{a} \rangle_B$ such that $B \preceq D$ and the embedding $\delta : C \to D$ is elementary. In a picture,



where $\mu : \langle \bar{a} \rangle \to C$ is the unique embedding which takes $\bar{a} \mapsto \bar{c}$

Proof Sketch.
Do it without the constants first. Let T be the union of the elementary diagram of B and the elementary diagram of C. Suppose T is not consistent; noting that constants from one do not appear in the other, derive a contradiction.

6.2 Strong Elementary Amalgamation Property

An amalgamation is said to be *strong* if the overlap between the two extensions B and C of A have minimal overlap in the amalgam D; that is, the embedded images of B and C in D have only the domain of A in common with one another.

► **Theorem 6.2** (Strong Elementary Amalgamation Property)**.**
Let A, B, C be L-structures such that $A \preceq B$ and $A \preceq C$. Furthermore, let $\bar{a}$ (not necessarily finite) name all the elements of A. Then, there is an L-structure D such that $B \preceq D$ and there exists an elementary embedding $\delta : C \to D$ such that $\mathrm{dom}(B) \cap \delta(\mathrm{dom}(C)) = \mathrm{dom}(A)$.

Proof Sketch.
Let $T = \mathrm{Diag}_{\mathrm{el}}(B) \cup \mathrm{Diag}_{\mathrm{el}}(C) \cup \{\text{`}b \neq c\text{'}$: for all $b \in B - A$ and all $c \in C - A\}$. Show that T is finitely satisfiable; in particular, suppose not and note that we need only consider one element from each elementary diagram (elementary diagrams are closed under conjunction). From here, simply use the universalization trick: $\mathrm{Diag}_{\mathrm{el}}(B) \models \varphi(\bar{a}, \bar{c}) \to \bigvee_{i,j} c_i = b_j$, and so $A \models \varphi(\bar{a}, \bar{d})$, but then $B \models (\bar{a}, \bar{d})$ (use the earlier).

6.3 The Strong Amalgam Property and Algebraicity

In the previous section we showed that, in first-order logic, two elementary extensions of a model always have a strong amalgam; it's natural to ask whether this result can be weakened. It turns out that it can–but not completely. A strong

amalgam does not, in general, exist for any two extensions of a model. In order to guarantee existence, it's necessary to borrow a notion from mathematics: algebraicity.

Algebraic
Let B be an L-structure, $X \subseteq B$, and b an element of B. b is *algebraic over X in B* if and only if there is a formula $\varphi(x, \bar{y})$, tuple $\bar{a} \in X$, and $n \in \mathbb{N}$, such that $B \Vdash \varphi(b, \bar{a}) \wedge \exists_{\leq n} y\, \varphi(y, \bar{a})$

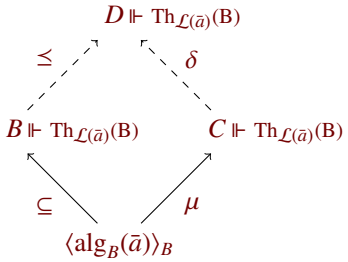
Algebraic Closure
The *algebraic closure of X in B*, $\text{alg}_B(X)$, is the set of all b from B which are algebraic over X. A set X is *algebraically closed* if and only if $X = \text{alg}_B(X)$

Proving each of the following is a good exercise in fleshing out an intuitive understanding of algebraic closures:

- **Fact.** For any model B,
- $X \subseteq \text{alg}_B(X)$; for $a \in X$, simply consider the formula $x \approx y$
- If $B \preceq C$, then $\text{alg}_B(X) = \text{alg}_C(X)$; use the definition of elementary extension
- If $Y \subseteq \text{alg}_B(X)$, then $\text{alg}_B(Y) \subseteq \text{alg}_B(X)$;
- $\text{alg}_B(\text{alg}_B(X)) = \text{alg}_B(X)$
- $\text{dom}(\langle \text{alg}_B(X) \rangle_B) = \text{alg}_B(X)$

The major result in this area is, unsurprisingly, a strong amalgamation theorem dealing with algebraically closed sets:

► **Theorem 6.3** (Strong Elementary Amalgamation over Algebraically Closed Sets)**.**
Let B and C be L-structures and $\bar{a}$ a sequence of elements in both B and C such that $(B, \bar{a}) \equiv (C, \bar{a})$. Then, there exist an elementary extension D of B and an elementary embedding $\delta : C \to D$ such that $\delta\bar{a} = \bar{a}$ and $\text{dom}(B) \cap g(\text{dom}(C)) = \text{alg}_B(\bar{a})$.



Presented slightly differently,

► **Lemma 6.1.**
Given a model B, a tuple $\bar{a} \in \text{dom}(B)$, and $b \in \text{dom}(B) - \text{alg}_B(\bar{a})$,

- There exists an L-structure A such that $B \preceq A$, $c \in \text{dom}(A) - \text{dom}(B)$, and $(A, \bar{a}, c) \equiv (B, \bar{a}, b)$
- There exists L-structures D and C such that $\bar{a} \in B \preceq D$, $\bar{a} \in C \preceq D$, and $b \notin \text{dom}(C)$.

Proof Sketch.
It's easier if we absorb X into the language and add a constant c_c for c. Let $T = \mathrm{Diag}_{\mathrm{el}}(B) \cup \{\varphi(c_c) : \varphi(x) \in \mathrm{tp}_B(b/\emptyset)\} \cup \{\text{`}c \neq d\text{'}: d \in B\}$. If this is inconsistent, then $\mathrm{Diag}_{\mathrm{el}}(B) \vdash \varphi(c_c) \to \bigvee_{i=1}^{k} \text{`}c_c = d_i\text{'}$, thus $B \models \forall x(\varphi(x) \to \bigvee_{i=1}^{k} \text{`}x = d_i\text{'})$. Prove that b is algebraic–a contradiction.

Note that, as a special case of the theorem above,

► **Corollary** (Strong Amalgamation for Elementary Equivalent Models)**.** Let B, C be L-models. If $B \equiv C$, then there exists a L-models D and C' such that B
$B \cap C' = \mathrm{alg}_B(\emptyset) = \mathrm{alg}_C(\emptyset)$.

Proof Sketch.
We want D to satisfy $T = \mathrm{Diag}_{\mathrm{el}}(C) \cup \mathrm{Diag}_{\mathrm{el}}(B) \cup \{\text{`}b \neq c\text{'}: b \in B - \mathrm{alg}_B(\emptyset), c \in C - \mathrm{alg}_C(\emptyset)\}$. Suppose it's inconsistent. Then there are finite sets $Y \subseteq B - \mathrm{alg}_B(\emptyset)$, $Z \subseteq C - \mathrm{alg}_C(\emptyset)$ such that $\mathrm{Diag}_{\mathrm{el}}(B) \cup \mathrm{Diag}_{\mathrm{el}}(C) \vdash \bigvee_{y \in Y, z \in Z} \text{`}y = z\text{'}$. Thus we have that $Y \cap g(Z) \neq \emptyset$ where g is the elementary embedding from C to D.

6.4 Interpolation and Definability

► **Theorem 6.4** (Elementary Extensions across Language Expansions)**.**
Let B be an L_1-model and C an L_2-model for languages $\mathcal{L}_1$ and $\mathcal{L}_2$ such that $L = L_1 \cap L_2$. If $B \restriction L \equiv C \restriction L$, then there is an $L_1 \cup L_2$-model D such that $B \preceq D \restriction L_1$ and there is an L_2-elementary embedding from C into $D \restriction L_2$.

Proof Sketch.
Observe that if $L_2 = L$, then $L_2 - \mathrm{Diag}_{\mathrm{el}}(B) \cup L - \mathrm{Diag}_{\mathrm{el}}(C)$ is consistent because if $\varphi(\bar{b}) \vdash \neg\psi(\bar{z})$ for $\varphi \in \mathrm{Diag}_{\mathrm{el}}(B)$, $\psi \in \mathrm{Diag}_{\mathrm{el}}(C)$, then $\varphi(\bar{b}) \vdash \forall \bar{x} \neg\psi(\bar{x})$, $B \models \forall \bar{x} \neg\psi(\bar{x})$, and $C \models \forall \bar{x} \neg\psi(\bar{x})$–contradicting $C \models \psi(c)$.
Let B_0 be an L_1 structure such that $B \preceq_{L_1} B_0$ and there is an embedding $C \restriction_L \preceq_L B_0$. Since $C \restriction_L \equiv_L B_0 \restriction_L$, there is C_0 an L_2 structure such that $B_0 \restriction_L \preceq_L C_0 \restriction_L$ and $C \restriction_L \preceq_{L_2} C_0 \restriction_L$. Let $D = \bigcup_{n=1}^{\omega} B_n$, an $\mathcal{L}_1$ structure; we have $B \preceq_{L_1} D$. Observe that $\tilde{D} = \bigcup_{n=1}^{\omega} C_n$, an $\mathcal{L}_2$ structure, and thus $C \preceq_{\mathcal{L}_2} \tilde{D}$. Observe that g_{n+1}

► **Theorem 6.5** (Craig's Interpolation Theorem)**.**
Assume $\mathcal{L}_1, \mathcal{L}_2$ are languages with $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$. Let φ be an L_1-sentence and χ an L_2-sentence such that $\varphi \vdash \chi$. Then, there is an L-sentence ψ such that $\varphi \vdash \psi$ and $\psi \vdash \chi$.

Proof Sketch.
Let $\Psi = \{\psi$ is an L-sentence such that $\varphi \vdash \psi\}$ = the set of L-implications of φ. We show first that $\Psi \models \chi$. Let C be an L_2 structure such that $C \models \Psi$. Consider $\varphi \cup \mathrm{Th}_L(C \restriction_L)$. Assume it is not consistent; then $\varphi \vdash \neg\theta$ where $\theta \in \mathrm{Th}_L(C \restriction_L)$, but then $\neg\theta \in \Psi$ and $C \models \Psi$, so $\neg\theta \in \mathrm{Th}(C)$–a contradiction. Let B be an L_2 structure such that $B \models \varphi \cup \mathrm{Th}_L(C \restriction_L)$. We have $C \equiv_L B$, so there is an $\mathcal{L}_1 \cup \mathcal{L}_2$ structure D such that $B \preceq_{L_1} D \restriction_{L_1}$ and $C \preceq_{L_2} D \restriction_{L_2}$. By assumption, $B \models \varphi$, so $D \models \varphi$, so $D \models \chi$, so $C \models \chi$.

► **Theorem 6.6** (Beth's Definability Theorem)**.**
Let L and L^+ be signatures such that $L \subseteq L^+$. If $\varphi(\bar{x})$ is an L^+-formula, then the following are equivalent:

- For A, B L^+-structures, if $A \restriction L = B \restriction L$, then for $\bar{a}$ a tuple of elements from A and B, $A \Vdash \varphi[\bar{a}]$ if and only if $B \Vdash \varphi[\bar{a}]$
- There is an L-formula $\psi(x)$ such that $\mathrm{Th}_L(B) \vdash \forall x[\varphi(x) \leftrightarrow \psi(x)]$.

Proof Sketch.

Chapter 7

Types, Saturation, and Categoricity

7.1 Types

A type is–at base–a description of a possible element or sequence of elements in a model, often with the use of parameters from some distinguished set. More formally, let M be an $\mathcal{L}$-structure and $A \subseteq \text{dom}(M)$. $\mathcal{L}(A)$ refers to the language $\mathcal{L}$ augmented by constants for each element of A while $\text{Th}_{\mathcal{L}_A}(M)$ is, of course, the theory of M (or, more properly, that extension of M wherein the added constants refer as intended) in $\mathcal{L}_A$.

> **Type**
> A set of $\mathcal{L}(A)$ formulas p with $x_1, \ldots, x_n$ free variables is an *n-type in M over A* if and only if $\text{Th}(M)_{\mathcal{L}_A} \cup \exists \bar{x} \bigwedge p(\bar{x})$ is satisfiable

An n-type p is *complete* if and only if for any formula $\varphi(x_1, \ldots, x_n)$, either $\varphi \in p$ or $\neg\varphi \in p$. The set of all complete n type in M over A is denoted $S_n^M(A)$

► **Proposition 7.1** (Type ⇔ Finitely Satisfiable)**.**
A set of $\mathcal{L}(A)$ formulas p with $x_1, \ldots, x_n$ free variables is an n-type in M over A if and only if $\text{Th}(M)_{\mathcal{L}_A} \cup \exists \bar{x} \bigwedge p(\bar{x})$ is finitely satisfiable

---

The simplest way to generate a complete type is, of course, to describe an element that actually exists in the model. If $\bar{a} \in \text{dom}(M)$, then $\text{tp}_M(\bar{a}/A)$ is the set of all n-ary $\mathcal{L}(A)$-formulas $\varphi(\bar{x})$ such that $M \Vdash \varphi[\bar{a}]$; $\text{tp}_M(\bar{a}/A)$ is, as our notation hints, obviously itself a complete type. Not all complete types correspond, however, to a tuple from the model itself. Those that do are said to be *realized* in the model; those that do not are *omitted*.

► **Proposition 7.2** (Types are Realized in Elementary Extensions)**.**
Let $p(\bar{x})$ be an n-type in M over A. Then, there is a model A such that $M \preceq A$ and A realizes p

---

> ■ ***Nota bene.*** If $M \preceq N$, then $\text{Th}_{\mathcal{L}_A}(M) = \text{Th}_{\mathcal{L}_A}(N)$, and thus $S_n^M(A) = S_n^N(A)$. That is, the set of complete types is constant along the elementary substructure relation.

7.1.1 Tuples with the Same Type

If a complete type is a complete description of a tuple of possible elements in a given model, then two tuples having the same complete type is, quite naturally, understood as the tuples being possibly identical for the model. Indeed, this intuition can be made precise:

► **Proposition 7.3** (Same Complete Types ⇒ Automorphism in Elementary Extension)**.**
If $\mathrm{tp}_M(\bar{a}/A) = \mathrm{tp}_M(\bar{b}/A)$, then there is a model N such that $M \preceq N$ and M has an automorphism δ such that $\delta\bar{a} = \bar{b}$

Proving the result above requires a new bit of terminology and a clever lemma:

Partial Elementary Map
A *partial elementary map* from a model N to a model M is a function $\delta : B \subseteq \mathrm{dom}(N) \to \mathrm{dom}(M)$ such that, for any $\bar{b} \in B$ and $\varphi \in \mathcal{L}$,

$$N \Vdash \varphi[\bar{b}] \Leftrightarrow M \Vdash \varphi[\delta\bar{b}]$$

► **Lemma 7.1** (Partial Elementary Maps Always Extend in an Elementary Extension)**.**
Let M and N be $\mathcal{L}$-models with $B \subseteq \mathrm{dom}(N)$ and $\delta : B \to \mathrm{dom}(M)$ partial elementary. For any $b \in \mathrm{dom}(M)$, there is an elmentary extension M' of M and $\delta' : B \cup \{b\} \to M'$ extending δ

► **Corollary** (Partial Elementary Maps Extend Completely in Some Elementary Extension)**.**
Let M and N be $\mathcal{L}$-models with $B \subseteq \mathrm{dom}(N)$ and $\delta : B \to \mathrm{dom}(M)$ partial elementary. There is a model M' such that $M \preceq M'$ and δ extends to an elementary embedding $\delta^* : N \to M$

7.1.2 Principal and Supported Types

Let $\mathcal{L}$ be a countable language and M an $\mathcal{L}$-model.

Support
Let $p(\bar{x})$ be a type of M over $A \subseteq \mathrm{dom}(M)$. $p(\bar{x})$ has *support* $\varphi(\bar{x})$ if and only if $\varphi(\bar{x})$ is an $\mathcal{L}(A)$-formula(?) such that $\mathrm{Th}_{\mathcal{L}(A)}(M) \cup \{\exists\bar{x}\ \varphi(\bar{x})\}$ is satisfiable and for every $\psi(\bar{x}) \in p(\bar{x})$, $T \Vdash \forall\bar{x}(\varphi(\bar{x}) \to \psi(\bar{x}))$; $\varphi(\bar{x})$ itself is called a *support* of the type $p(\bar{x})$.

Note that the clause '$\mathrm{Th}_{\mathcal{L}(A)}(M) \cup \{\exists\bar{x}\ phi(\bar{x})\}$ is satisfiable' simply guarantees that $\varphi(\bar{x})$ doesn't generate a contradiction, but rather legitimately implies the formulas of $p(\bar{x})$ against the backdrop of $\mathrm{Th}_{\mathcal{L}(A)}(M)$. A special case of support arises when the supporting formula $\varphi(\bar{x})$ is itself a member of the type it supports; in such cases, $\varphi(\bar{x})$ is called a *generator* for p and the type $p(\bar{x})$ is said to be *principal*.

Principal
Let $p(\bar{x})$ be a type of M over $A \subseteq \mathrm{dom}(M)$. $p(\bar{x})$ is *a principal type* if and only if it has support $\varphi(\bar{x})$ such that $\varphi(\bar{x}) \in p(\bar{x})$.

In a striking continuity of definition, a type is said to be *unsupported* or *non-principal* if it is not supported or not principal, respectively. From only the definitions given, several basic facts about supported and principal types are immediate:

■ **Fact.** If a type $p(\bar{x})$ over a set A is complete (with respect to all formulas with parameters from A), then it has support if and only if it is principal over T.

7.1.3 Towards Saturation

All this talk of types and possible elements of a model leads naturally to the construction of models which realize as many types as possible; our first step in this direction and towards the notion *saturation* is the following:

► **Theorem 7.1** (Elementary Extensions which Realize all Types of the Model)**.**
For any L-model A, there is an L-model B such that $A \preceq B$ and every type of A is realized in B.

Proof Sketch.
List all of the complete types over A: $\{\Phi_\sigma : \sigma \in \kappa\}$. Let $B_0 = A$. By definition, there is a B_1 such that $B_0 \preceq B_1$ and B_1 realizes Φ_0. Note further that Φ_1 is finitely realized in $A = B_0$ and $B_0 \preceq B_1$–thus, Φ_1 is finitely realized in B_1 as well. Φ_1 is thus a complete type of B_1, and there is thus a B_2 such that $B_1 \preceq B_2$ and B_2 realizes Φ_1 (and Φ_0). Generalize to an inductive argument.

7.2 Omitting Types

Before moving to models which realize as many types as possible, the opposite process bears mentioning. In particular, by restricting to the countable case it becomes possible to exert near perfect control over which types of a theory T are and are not omitted in elementary extensions of some model A–with a single exception.

► **Theorem 7.2** (Countable Omitting Types Theorem)**.**
Let $\mathcal{L}$ be a countable, first-order language. If T is an $\mathcal{L}$-theory and $\Phi_1, \Phi_2, \Phi_3, \ldots$ are unsupported types over T, then there is a countable model of T which omits all the types Φ_i for $i \in \mathbb{N}$.

Proof Sketch.
Let $L^+ = L \cup \{c_1, c_2, c_3, \ldots\}$. Generate T^+, an L^+ theory, that is finitely satisfiable, complete, has witnesses, $T \subseteq T^+$, and for every $m \in \omega$ and every c_i there is $\varphi(x) \in \Phi_m$ such that $T^+ \vdash \neg\varphi(c_i)$. The term model of T^+ is a model of T with the required property.

To construct T^+, build a sequence $T_0 \subseteq T_1 \subseteq \ldots$ of finite sets of L^+ sentences such that $\forall i T \cup T_i$ is consistent and –at the end– we let $T^+ = T \cup \bigcup_{i=0}^{\omega} T_i$. At stage s, suppose we have T_s and need T_{s+1}. If $s = 3e$, work toward completeness (add the next sentence or its negation). If $s = 3e + 1$, work towards adding witnesses. If $s = 3e + 2$, take the next pair $\langle i, m\rangle$. Let $\vartheta = (c_i, \bar{d}) = \bigwedge T_s$. T is consistent with $\vartheta(c_i, \bar{d})$ and so $\exists x \exists \bar{y}\ \vartheta(x, \bar{y})$. Since Φ_m is not supported by $\exists \bar{y} \vartheta(x, \bar{y})$, there is $\varphi(x) \in \Phi_m$ such that $T \nvdash (\exists \bar{y} \vartheta(x, \bar{y})) \to \varphi(\bar{x})$. Thus, $T \nvdash \vartheta(c, \bar{d}) \to \varphi(c)$. So, $T \cup T_s \cup \neg\varphi(c_i)$ is consistent. Let $T_{s+1} = T_s \cup \{\neg\varphi(c_i)\}$.

We argue along the lines of the omitting types theorem. First, extend $\mathcal{L}$ to $\mathcal{L}^+$ by adding a constant for every element (countably many) and countably many new constants. Let T_{-1} be the empty theory and note that, by assumption, $T \cup T_{-1}$ has a model which is an $\mathcal{L}^+$ model.

Given T_{i-1}, T_i is generated as follows. Noting that T_{i-1} must be finite and that the formulas of $\mathcal{L}^+$ are countable, consider the next formula in the sequence of L^+ formulas which hasn't yet been considered, say φ_i. If $T \cup T_{i-1} \cup \{\varphi_i\}$ has a model, define $T_i = \{\varphi_i\} \cup T_{i-1}$; if not, then every model of $T \cup T_{i-1}$ is a $\neg\varphi$ model, and so define $T_i = \{\neg\varphi_i\} \cup T_{i-1}$. In either case, $T \cup T_i$ has a model.

Next, we search through T_i for any formulas of the form $\exists x \varphi(x)$. Every time such a formula is found, take one of our new constants (of which only finitely have been used), say c_φ, to name an element which satisfies φ (this must be possible since $T \cup T_i$ has a model. Redefine $T_i = T_i \cup \{\varphi(c_\varphi)\}$.

Finally,

Example 4.

Let $T = \mathrm{Th}(\mathbb{N})$. Note that $\Phi(\bar{x}) = \{$'$x \neq 0$', '$x \neq s0$', '$x \neq ss0$',$\ldots\}$ is finitely realizable and so a type. It cannot be supported, however, because $\mathbb{N}$ omits Φ.

7.3 Atomic and Prime Models

Throughout this section, it's assumed that $\mathcal{L}$ is countable.

Prime Model
A model M is *prime* if and only if whenever $M \equiv N$, $M \preceq N$

■ **Remark.** It's easy to show with Lowenheim-Skolem that–supposing a countable language–any prime model must be countable.

Example 5.

- The standard model of arithmetic $\mathcal{N}$ is prime
- Every finite model M is trivially prime

Atomic Model
A model M is *atomic* if and only if $\text{tp}_M(\bar{a})$ is principal for every $\bar{a} \in \text{dom}(M)$

► **Proposition 7.4** (Prime ⇒ Atomic)**.**
If a model M is a prime model of $\text{Th}(M)$, then M is atomic.

■ ***Nota bene*.** The converse does not hold generally, just in the countable case; consider, for example, uncountable dense linear orders without endpoints

---

► **Proposition 7.5** (Element-wise Definable ⇒ Atomic)**.**
If every element of a model M is definable, then M is atomic

► **Theorem 7.3.**
Let $\mathcal{L}$ be a countable, first-order language and T a complete $\mathcal{L}$-theory with infinite models.

- If for every $n < \aleph_0$, $S_n(T)$ is countable, then T has a countable atomic model
- If an L-model A is countable, atomic, and models T, then A is a prime model of T

Proof Sketch.
(1) Let $\Phi_1, \Phi_2, \ldots$ be a lits of all non-principle, complete types over T. This list is countable, so use the omitting types theorem; (2) Argue generally, noting the assumption of completeness, and construct the embedding piece by piece. In particular, prove and use that if $\bar{a}$ from A and $\bar{b}$ from B and $\text{tp}_A(\bar{a}) = \text{tp}_B(\bar{b})$, then for every $c \in A$, there is $d \in B$ such that $\text{tp}_A(\bar{a}c) = \text{tp}_B(\bar{b}d)$ (note that the types in A are principle). To do this, not that $\text{tp}_A(\bar{a}c)$ is generated by some $\theta(\bar{x}y)$. Thus, $T \models \forall\bar{x}\forall y\ (\theta(\bar{x}y) \leftrightarrow \bigwedge_{\varphi(\bar{x}y)\in \text{tp}_A(\bar{a}c)} \varphi(\bar{x}y)$. $A \models \exists y\theta(\bar{a}, y)$, so $B \models \exists y\theta(\bar{b}, y)$. Let d be that y. If we iterate this principle, we obtain an embedding; moreover, it must be elementary since the elements were chosen to have the same types and thus satisfy the same formulas.

► **Lemma 7.2.**
Let $\mathcal{L}$ be a countable, first-order language. If $A = \text{alg}_A(\varphi)$, then A is atomic.

Proof Sketch.
Take $\bar{a}$ from A such that there is a formula $\Theta(\bar{x})$ and $n \in \omega$ such that $A \models \theta(\bar{a})$ and $\exists_{\leq n}\bar{x}\ \theta(\bar{x})$. Pick θ so that n is minimized. Show that $A \models \forall\bar{x}(\theta(\bar{x}) \to \varphi(\bar{x}))$ whenever $\varphi(\bar{x}) \in \text{tp}_A(\bar{a})$. Let $\varphi(\bar{x}) \in \text{tp}_A(\bar{a})$. Let $\bar{a}_1, \ldots, \bar{a}_n$ be that tuples where θ is true. Assume $A \not\models \varphi(\bar{a}_i)$ for some i. Then, $\theta(\bar{x}) \wedge \varphi(\bar{x})$ is a formula that $\bar{a}$ satisfies, but which has fewer than n tuples which satisfy it total–contradicting our choice of θ.

► **Theorem 7.4.**
Let $\mathcal{L}$ be a countable, first-order language. Suppose A and B are atomic, countable L-models. Then, if $A \equiv B$, $A \cong B$.

Proof Sketch.
We adapt the argument showing 'If A is countable and atomic, then A is prime' to go back and forth between the models and generate an isomorphism. Pick $a_1 \in A$, $b_1 \in B$ such that $\text{tp}_A(a_1) = \text{tp}_B(b_1)$. Pick $a_2 \in A$, $b_2 \in B$ such that $\text{tp}_A(a_1, a_2) = \text{tp}_B(b_1, b_2)$. The end result of this procedure is a bijection $g(a_i) = b_i$ which preserves types and is thus an isomorphism.

7.4 Saturation

The end of the previous section saw us realizing all the types of some starting model in one of its elementary extensions; *saturation* takes this tactic one step further: realizing the types of a model in that very model.

Saturated
Let λ be a cardinal. A model M is *λ-saturated* if and only if for every $A \subseteq \text{dom}(M)$ with $|A| < \lambda$, every complete 1-type $p(x) \in S_n^M(A)$ is realized in M. A model M is *saturated* simpliciter if and only if M is $|M|$-saturated

Example 6.

- $(\mathbb{Q}, <)$ is saturated
- $(\mathbb{R}, <)$ is not saturated
- The countable random graph is saturated

► **Lemma 7.3** (Alternative Definition of Saturation)**.**
The following are equivalent:

(i) M is λ-saturated

(ii) For every $\mathcal{L}$-model B and (possibly infinite) tuples $\bar{a}$, $\bar{b}$ of length $< \lambda$ such that $(M, \bar{a}) \equiv (B, \bar{b})$, if $d \in B$, then there is a $c \in M$ such that $(M, \bar{a}, c) \equiv (B, \bar{b}, d)$.

► **Theorem 7.5** (Saturated Models are Unique)**.**
If M and N are saturated models of a theory T of the same size, then $M \cong N$.

7.4.1 Homogeneity

Homogeneity is, as the term suggests, a self-similarity property; for a model M to be λ-homogeneous for some cardinal λ, it must be that, when two parts of M of size $< \lambda$ are elementary identical (using parameters from these parts), any addition to one part can be mirrored by an addition on the other. That is, no difference occurs below the λ level. More formally,

> **Homogeneous**
> Let M be a model and λ a cardinal. M is *λ-homogeneous* if and only if for any possibly infinite tuples $\bar{a}$, $\bar{b}$ from M with length $< \lambda$, if $(M, \bar{a}) \equiv (M, \bar{b})$, then for every $d \in M$, there is a $c \in M$ such that $(M, \bar{a}, c) \equiv (M, \bar{b}, d)$. M is *homogeneous* simpliciter if and only if M is $|M|$-homogeneous.

■ **Remark.** Later, it will be popular to view this process as dealing in partial elementary embeddings; under this scheme, the above states that any partial elementary embedding $\delta : A \to \text{dom}(M)$ where $A \subseteq \text{dom}(M)$ and $|A| < \lambda$ can be extended by an additional element c so long as $|A \cup \{c\}| < \lambda$. The utility of this view clearly lies in the ability to eventually construct a full elementary embedding of M to itself.

The definition of homogeneity should, of course, be quite familiar; it is, after all, a slight weakening of the alternative definition proved above for saturation:

► **Proposition 7.6** (λ-Saturated $\Rightarrow$ λ-Homogeneous)**.**
If a model M is λ-saturated, then M is λ-homogeneous.

The notion of homogenity also has close ties to the notion of type; simply by definition:

► **Proposition 7.7.**
Suppose M is a $\aleph_0$-homogeneous model and $\bar{a}$, $\bar{b}$ from $\text{dom}(M)$. If $\text{tp}_M(\bar{a}) = \text{tp}_M(\bar{b})$, then for any $c \in \text{dom}(M)$, there is a $d \in \text{dom}(M)$ such that $\text{tp}_M(\bar{a}c) = \text{tp}_M(\bar{b}d)$

Pushing this idea further and leveraging partial elementary embeddings,

---

► **Theorem 7.6.**
If M and N are homogeneous, $M \equiv N$, $|M| = |N|$, and both realize the same n-types over $\emptyset$ for every n, then $M \cong N$.

In addition, $\aleph_0$-homogeneous models are rather ubiquitous entities:

► **Proposition 7.8** (Cardinality-Preserving, $\aleph_0$-Homogeneous Elementary Extensions)**.**
For any model M, there is a $\aleph_0$-homogeneous model N such that $M \preceq N$ and $|M| = |N|$.

Putting the facts above together, a characterization of not only $\aleph_0$-saturation generally, but also countable saturated models is possible:

► **Proposition 7.9** ($\aleph_0$-saturated $\Leftrightarrow$ $\aleph_0$-homogeneous and $S_n^M(\emptyset)$ Realized)**.**
A model M is $\aleph_0$-saturated if and only if M is $\aleph_0$-homogeneous and M realizes all types in $S_n^M(\emptyset)$ for every n

---

► **Theorem 7.7** (Countable, Saturated Model $\Leftrightarrow$ $S_n(T)$ is Finite)**.**
A complete theory T has a countable, saturated model if and only if $S_n(T)$ is finite for every n

---

7.4.2 Universality

It's clear that in moving from λ-saturation to λ-homogeneity something has been lost; luckily, that loss can be made explicit:

> **λ-Universal**
> A model M is *λ-universal* for a cardinal λ if and only if for every $B \equiv M$ with $|B| < \lambda$, $B \preceq M$. A model M is *universal* simpliciter if and only if M is $|M|^+$-universal.

Universality is, as it were, the obvious dual to the previously-defined *prime*; as many models as possible elementary embed into a universal model.

► **Theorem 7.8** (λ-saturated $\Leftrightarrow$ λ^+-universal and λ-homogeneous)**.**
Let λ be a cardinal. Then, the following are equivalent:

(i) M is λ-saturated

(ii) M is λ^+-universal and λ-homogeneous

Additionally, if $\lambda > \aleph_0$, then

(iii) M is λ-universal and λ-homogeneous

► **Corollary** (Saturated $\Leftrightarrow$ Homogeneous and Universal)**.**
A model M is saturated if and only if it is homogeneous and universal

---

► **Theorem 7.9** (λ-Saturated Models Always Exist)**.**
For every model A and every cardinal λ, there is a model B such that $A \preceq B$ and B is λ-saturated.

► **Theorem 7.10** ($\aleph_0$-Homogeneous Extensions)**.**
Let A be a model such that $|A| \geq \aleph_0$ and $A \preceq B$ for some $\aleph_0$-homogeneous B. Then, there is a model C which is $\aleph_0$-homogeneous, $A \preceq C \preceq B$, and $|C| = |A|$.

7.5 Categoricity

Categorical
A theory T is *λ-categorical* for a cardinal λ if and only if T is consistent and for all T-models A, B such that $|A|, |B| = \lambda$, $A \cong B$. As usual, a model A is *λ-categorical* just in case $\text{Th}(A)$ is.

A theory is said to be *categorical* simpliciter if and only if it is categorical for every cardinal λ–the next result, however, shows this to be a rather uninteresting result:

► **Theorem 7.11** (All Categorical Theories are Th(A) for a Finite Model A)**.**
If T is a categorical theory, then $T = \text{Th}(A)$ for some finite model A

Restricting to κ-categorical for some cardinal κ, however, proves much more nuanced. We already have two ready examples of $\aleph_0$-categorical models/theories:

Example 7.

- $(\mathbb{Q}, <)$ / dense linear orders without endpoints is $\aleph_0$-categorical
- The random graph is $\aleph_0$-categorical

For $\bar{a}, \bar{b} \in \text{dom}(A)$, $\bar{a}$ and $\bar{b}$ are automorphic (notated $\bar{a} \sim \bar{b}$) if and only if there is an automophism $\delta : A \to A$ such that $\delta\bar{a} = \bar{b}$. Putting together many of the results of this chapter,

---

► **Theorem 7.12** (Characterizations of $\aleph_0$-Categoricity)**.**
Let $\mathcal{L}$ be a countable, first-order language and T a complete $\mathcal{L}$-theory with infinite models; then, the following are equivalent:

(a) For every $A \Vdash T$ and every $n < \aleph_0$, $(A^n / \sim)$ is finite

(b) For some $A \Vdash T$ and every $n < \aleph_0$, $(A^n / \sim)$ is finite

(c) For some $A \Vdash T$ and every $n < \aleph_0$, A realizes finitely many n-types.

(d) For every $n < \aleph_0$, $S_n(T)$ is finite

(e) Given variables $\bar{x}$, there are only finitely many pairwise non-equivalent formulas $\varphi(\bar{x})$ modulo T

(f) For every $n < \aleph_0$, every type $p \in S_n(T)$ is principal over T

(g) T is $\aleph_0$-categorical

Proof Sketch.
$(a) \to (b)$ is trivial.
$(b) \to (c)$ since automorphic tuples realize the same type.
$(c) \to (d)$, fix n. A realizes k n-types. Suppose that $|S_n(T)| > K$. Let $p_0, \dots, p_k$ be types in $S_n(T)$. There is a formula $\varphi'_j(\bar{x})$ such that $\varphi'_j(\bar{x}) \in p_j$ and $\varphi'_j(\bar{x}) \notin p_i$, $i \neq j$, setting $\varphi_j = \varphi'_j \wedge \bigwedge_{i \neq j} \neg\varphi'_i \in p_j$. Not that $\varphi_{j,i} \in p_j - p_i \to \bigwedge_{i=1}^{k} \varphi_{j,i} \in p_j - p_i$. For every i, since $p_i \in S_n(T)$ and $\varphi_i \in p_i$, $T \vdash \exists \bar{x}\ \varphi_i(\bar{x})$, and thus $A \models \exists \bar{x}\ \varphi_i(\bar{x})$ with some $\bar{a}_i$. Show that each $\bar{a}_i$ realizes a different type for each i–a contradiction.
$(d) \to (e)$, we know there are k types in $S_n(T)$, $p_1(\bar{x}), \dots, p_k(\bar{x})$. Show that if $\varphi(\bar{x})$ and $\psi(\bar{x})$ are such that $\forall i = 1, \dots, k\ \varphi \in p_i \leftrightarrow \psi \in p_i$, then $T \vdash \forall \bar{x}(\varphi(\bar{x}) \leftrightarrow \psi(\bar{x}))$, possibly by contradiction. Argue combinatorially.
$(e) \to (f)$, let $\psi_1(\bar{x}), \dots, \psi_k(\bar{x})$ be a maximal set of inequivalent formulas. Given $p(\bar{x}) \in S_n(T)$, take $\theta(\bar{x}) := \bigwedge_{i \leq k, \psi_i \in p} \psi_i(\bar{x}) \in p(\bar{x})$. If $\varphi(\bar{x}) \in p(\bar{x})$, then for some i, $T \vdash \varphi \leftrightarrow \psi_i$ and $\psi_i \in p$, so $\theta(\bar{x}) \to \varphi(\bar{x})$.
$(f) \to (g)$, every model of T is atomic.
$(g) \to (f)$, every type that is not principle can be omitted (be careful with countability), so $\neg(f) \to \neg(g)$ since we can have one model realizing the non-principle type and another which does not–meaning they aren't isomorphic.
$(f) \to (d)$, Suppose $S_n(T)$ is infinite and every type is principle; let $p_1(\bar{x}), \dots$ be a listing of the types in $S_n(t)$. Each $p_i(\bar{x})$ has a generator $\varphi_i(\bar{x})$. Note that the set of the negations of these generators–say $\Phi(\bar{x})$–is finitely satisfiable (each generator must be false in every type it doesn't generate; every type is realized in some model). Then it can be extended to a complete type–but our list was supposed to contain all the types!
$(d) \to (a)$, Note that–by (f)–every model of T is atomic. Moreover, if A is atomic, if $\bar{a}, \bar{b}$ from A and $\mathrm{tp}_A(\bar{a}) = \mathrm{tp}_A(\bar{b})$, then $\forall c \exists d \mathrm{tp}_A(\bar{a}c) = \mathrm{tp}_A(\bar{b}d)$. Using this–as earlier–we can construct an automorphism $f : A \to A$ with $f(\bar{a}) = \bar{b}$. Noting that there are finitely many types, there must then by finitely many equivalent classes.

An interesting corollary of the theorem above is that $\aleph_0$-categorical models are *locally finite*:

Locally Finite
A model A is *locally finite* if and only if every finitely-generated submodel of A is itself finite.

► **Corollary** ($\aleph_o$-categorical ⇒ Locally Finite)**.**
If a model A is $\aleph_0$-categorical, then A is locally finite. In fact, there is a unique function $f : \mathbb{N} \to \mathbb{N}$ depending only on $\mathrm{Th}(A)$ with the property that for each $n < \aleph_0$, $f(n)$ is the least number m such that every n-generator submodel of A has at most m elements.

Proof Sketch.
Suppose that $a_1, \dots, a_n \in M$ generate an infinite model; if $b_1, b_2 \in \langle a_1, \dots, a_n \rangle$ and $b_1 \neq b_2$, $\mathrm{tp}(a_1, \dots, a_n, b_1) \neq \mathrm{tp}(a_1, \dots, a_n, b_3)$ because '$b_1 = t_1(a_1, \dots, a_n)$' ....

A deeper result dealing in categoricity–and beyond the scope of these notes–is Morley's categoricity theorem; while no proof is even attempted here, the statement is given and some of the basic notions of the theorem are developed in the next subsections.

► **Theorem 7.13** (Morley's Categoricity Theorem)**.**
If T is a theory in a countable $\mathcal{L}$ which is categorical for some uncountable cardinality, then T is categorical in all uncountable cardinalities

---

Chapter 8

Other Model-Theoretic Constructions

Fair warning, the section on Fraïssé limits below has never been edited while the section on ultrafilters is woefully incomplete.

8.1 Fraïssé's Construction

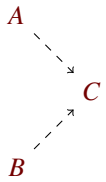
The countable case is a unique one; it borders on the finite–where things are, as a general rule, simpler–but also allows for infinite processes, useful techniques like chains and the models representing their unions. The combination of these proves powerful and allows for several results not generally applicable. Moreover, since the majority of logics fall into the countable case, it's well worth our time to explore the class of countable models.

This section details a construction technique which generates particularly nice countable models; indeed, much of our work in the following sections will make use of the properties these countable models possess. Perhaps the most basic notion we'll make use of is the *age of a model*.

Age
Let D be an L-model. The age of D is the class, up to isomorphism, of all finitely generated submodels of D. Thus, a class of models κ is an *age* if and only if it is the age of some model D.

It's easy to show that an age has the following two properties: *joint embedding* and *heredity*.

Joint Embedding Property
A class of models K has the *joint embedding property* (JEP) if and only if for every $A, B \in K$, there is $C \in K$ such that A and B embed in C.



Hereditary Property
A class of models K has the *hereditary property* (HP) if and only if for every $A \in K$ and every finitely-generated submodel B of A, an isomorphic copy of B is in K; in other words, K is closed under the operation of taking finitely-generated submodels.

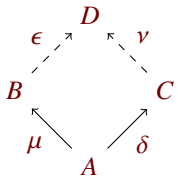
Of course, Fraïssé proved more than just that ages have the joint embedding and hereditary properties; ages are completely characterized by them:

► **Theorem 8.1** (Fraïssé's Characterization of Age)**.**
A countable class K of finitely-generated models is an age if and only if it has both the joint embedding and hereditary properties.

Proof Sketch.
The forward direction is easy, so the reverse is sketched here. Our strategy is simply to take an arbitrary class of finitely generated models κ with JEP and HP, then construct the model D for which it is an age. Assume $\kappa = \{A_0, A_1, \dots\}$ is an arbitrary class of finitely-generated models with HP and JEP. We build a chain of models as follows:
Let $B_0 = A_0 \in \kappa$. In general, let B_{n+1} be the model in κ such that both B_n and A_{n+1} embed in it (JEP). Let $D = \bigcup_{i<\omega} B_i$. If $A \in \kappa$, then A embeds in D. If C is a finitely generated submodel of D, then all of the generators must be present in some $B_n \in \kappa$, and thus $C \subseteq B_n$. It follows that $C \in \kappa$ and $C \subseteq D$.

Alongside the joint embedding and hereditary properties, we single out one last property which will ultimately prove necessary for our main result:

Amalgamation Property
A class of models K has the *amalgamation property* if and only if for every $A, B, C \in K$ such that $\delta : A \to C$, $\mu : A \to B$ are embeddings, there is a $D \in K$ and embeddings $\epsilon : B \to D$, $\nu : C \to D$ such that $\nu \circ \delta = \epsilon \circ \mu$



Although it is tempting to think so, the amalgamation property (AP) doesn't subsume the joint embedding property as a special case. In addition to these three properties of classes of models, another property of particular models will also prove useful:

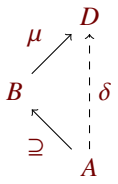
Ultrahomogeneous
A model D is *ultrahomogeneous* if and only if whenever A and B are finitely-generated submodels of D, every isomorphism $A \to B$ extends to an automorphism of D.

The primary result of this section is the construction of a particular model–the Fraïssé limit–from a class of countable models K which is itself countable; this particular construction is important because the Fraïssé limit is not only unique up to isomorphism (i.e. categorical), but also possess two other important properties:

► **Theorem 8.2** (Fraïssé's Theorem)**.**
Let K be a countable class of finitely-generated, countable models with the hereditary, joint embedding, and amalgamation properties. Then K is the age of a model D, called *the Fraïssé limit of* K, which is countable, ultrahomogeneous, and $\aleph_0$-categorical.

8.1.1 Constructing the Fraïssé Limit

Weakly Homogeneous
A model D is *weakly homogeneous* if and only if whenever A and B are finitely-generated submodels of D such that $A \subseteq B$ and $\delta : A \to D$ is an embedding, then there is an embedding $\mu : B \to D$ extending δ.



► **Proposition 8.1** (Ultrahomogeneous ⇒ Weakly Homogeneous)**.**
If a model A is ultrahomogeneous, then A is also weakly homogeneous

Proof Sketch.
Build a sequence of models $B_0 \subseteq B_1 \subseteq \ldots$ all in κ such that for every $A \in \kappa$, there is a j where A embeds in B_j and for every $C \subseteq A \in \kappa$, every $i \in \omega$, and embedding $g : C \to B_i$, there is a j and an embedding $g' : A \to B_j$. If we can establish this, then the conclusion follows easily. Note that $\mathbb{N}^4 \approx \mathbb{N}$, let $(i, j, k_0, k_1) \mapsto \langle i, j, k_0, k_1 \rangle$. At stage s we will build B_{s+1}. If $s = 2e$, let B_{s+1} be obtained using JEP with B_s and A_e. If s is odd, it's of the form $2\langle i, j, k_0, k_1 \rangle + 1$. Suppose $i, j, k_0, k_1 < s$. Let C be the jth finitely generated submodel of B_i. Let f' be the k_0th finitely generated submodel of A_{k_1}. If $C \cong C'$, let B_{s+1} be obtained by using AP on C, A_{k+1}, and B_s. If $C \ncong C'$, do nothing.

► **Lemma 8.1.**
Let C, D be L-models. If the age$(C) \subseteq$ age(D) and D is weakly homogeneous, then C embeds in D.

Proof Sketch.
Consider increasingly large elements of age(C) and note that, while each element may be disjoint inside of D, we can construct an extended embedding around the smaller element which is isomorphic to the larger. Building successively, we obtain an embedding of C.

► **Lemma 8.2** (Back and Forth Argument)**.**
Let C, D be L-models. If the age$(C) =$ age(D) and C,D are weakly homogeneous, then for any finitely -generated submodels A, B of C, D where $A \cong B$ by δ, δ extends to an isomorphism $\delta^+ : C \to D$.

Proof Sketch.
Let A, B be arbitrary submodels such that $A \cong B$. Let A_1 be the model generated by A_0 and c_0–an arbitrary element of C. Since the ages of C and D are identical, there is a A_1' in D such that $A_1 \cong A_1'$. Note, however, that A_0 embeds in A_1', and so–by weakly homogeneous–there is a B_1 such that $B_1 \cong A_1'$ and $B \subseteq B_1$. Let B_2 be the model generated by B_2 and d_0 an arbitrary element of D. Repeat the argument above, going from D to C.

► **Corollary** (Weakly Homogeneous ⇔ Ultrahomogeneous)**.**
A model is weakly homogeneous if and only if it is ultrahomogeneous

Simply use the lemma above with the same model for C and D.

► **Corollary** (Fraïssé Limits are Unique)**.**
The Fraïssé limit of a class of models K is $\aleph_0$-categorical.

8.1.2 Countable Categoricity and Fraïssé Limits

$\aleph_0$-categorical models are handy; luckily, we have a powerful means of constructing them: Fraïssé's method. The only difficulty to simply importing the Fraïssé limit construction covered earlier wholesale is ensuring that the sizes of the models in the class κ are kept under control by the number of generators which give rise to them, a concern which gives rise to the following property:

Uniformly Locally Finite
A model A is *uniformly locally finite* if and only if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for every n, if $\bar{a}$ is an n-tuple from A, $|\langle \bar{a} \rangle| \leq f(n)$.

► **Corollary** ($\aleph_0$-Categorical ⇒ Uniformly Locally Finite)**.**
If a model A is ω-categorical, then A is uniformly locally finite.

Proof Sketch.
Note that if $\text{tp}(a_1, \ldots, a_n) = \text{tp}(b_1, \ldots, b_n)$, then $\langle a_1, \ldots, a_n \rangle \cong \langle b_1, \ldots, b_n \rangle$ (think about what happens if they're not). Since $S_n(T)$ is finite (say size m), there are at most m distinct submodels generated from n elements–so set $f(n) = m$ for all n.

► **Lemma 8.3.**
Let $\bar{a}, \bar{b}$ be tuples of A. Then the following are equivalent:

(a) $\langle \bar{a} \rangle \cong \langle \bar{b} \rangle$

(b) $\text{qf-tp}_A(\bar{a}) = \text{qf-tp}_A(\bar{b})$

Proof Sketch.
$(a) \to (b)$; Take an atomic formula, say $R(t_1(\bar{a}), t_2(\bar{a}), \ldots, t_n(\bar{a}))$. Show that $M \models R(t_1(\bar{a}), t_2(\bar{a}), \ldots, t_n(\bar{a}))$ if and only if $M \models R(t_1(\bar{b}), t_2(\bar{b}), \ldots, t_n(\bar{b}))$, and so all quantifier-free formulas have the same truth values.

$(b) \to (a)$; We want to produce f. Take $t(\bar{a})$, define $f(t(\bar{a})) = t(\bar{b})$. Consider atomic formulas and show that f is well-defined, surjective, and injective.

► **Lemma 8.4.**
If $\langle \bar{a} \rangle$ is finite, then there is a q.f. formula $\varphi(\bar{x}) \in \text{qf-tp}(\bar{a})$ such that $M \models \forall x(\varphi(\bar{x}) \leftrightarrow \wedge \text{qf-tp}(\bar{a}))$ (i.e. if $\bar{b} \in M$ and $M \models \varphi(\bar{b})$, then $\langle \bar{a} \rangle \cong \langle \bar{b} \rangle$).

Proof Sketch.
Let $\langle \bar{a} \rangle = \{t_1(\bar{a}), \ldots, t_k(\bar{a})\}$. Let $\varphi(\bar{x})$ be the conjunctions of all formulas '$f(t_{i_1}(\bar{a}), \ldots, t_{i_l}(\bar{a})) = t_s(\bar{a})$' for all of the l-ary function symbols f and all tuples $(i_1, \ldots, i_l) \in k^l$. Repeat this for both R and $\neg R$ as well.

Homogeneous
A model M is homogeneous if and only if $\forall \bar{a}, \bar{b}$ if $\mathrm{tp}_M(\bar{a}) = \mathrm{tp}_M(\bar{b}) \to \bar{a} \sim \bar{b}$.

► **Theorem 8.3.**
If M is atomic, then M is homogeneous

Proof Sketch.
Assume $\mathrm{tp}(\bar{a}) = \mathrm{tp}(\bar{b})$. By atomic, there is $\theta(\bar{x}, y)$ generating $\mathrm{tp}(\bar{a}c)$. $M \models \exists y \theta(\bar{a}, y)$, and so $M \models \exists y \theta(\bar{b}, y)$. Let d be such that $M \models (\bar{b}d)$. Note that $\theta(\bar{b}d)$ now generates the whole type.

► **Corollary.**
If M is $\aleph_0$-categorical, then M is homogeneous

► **Theorem 8.4.**
Let $\mathcal{L}$ be finite. Let κ be a class of finitely generated models which is uniformly locally finite and has the HP, the JEP, and the AP. Let M be the Fraïssé limit of κ, $T = \mathrm{Th}(M)$. Then,

(a) T is $\aleph_0$-categorical

(b) T has quantifier elimination

Proof Sketch.

- M is ultrahomogeneous if and only if $\forall \bar{a}, \bar{b} \in M$ if $qf-\text{tp}(\bar{a}) = qf-\text{tp}(\bar{b}) \rightarrow \bar{a} \sim \bar{b}$ if and only if $\forall \bar{a}, \bar{b}$ if $qf-\text{tp}(\bar{a}) = qf-\text{tp}(\bar{b}) \rightarrow \forall c \exists d$ such that $qf-\text{tp}(\bar{a}, c) = qf-\text{tp}(\bar{b}, d)$.
- $qf-\text{tp}(\bar{a}) = qf-\text{tp}(\bar{b}) \leftrightarrow (\langle \bar{a} \rangle, \bar{a}) \cong (\langle \bar{b} \rangle, \bar{b})$
- Let L be finite. If $\langle \bar{a} \rangle$ is finite, there is a q.f. formula $\varphi(\bar{x})$ such that $M \models \forall \bar{x}(\varphi(\bar{x}) \leftrightarrow \bigwedge qf-\text{tp}(\bar{a}))$

(a) Note that the Fraïssé limit, M is countable, ultrahomogeneous, and age$(M) = \kappa$. We want to define a theory T such that the countable models of T are exactly the Fraïssé limit of κ (because then $M \models T$ and $T \subseteq \text{Th}(M)$ and if $A \models \text{Th}(M) \rightarrow A \models T \rightarrow A \cong M$).

To say that age$(M) \supseteq \kappa$, for each A in κ generated by $\bar{a} \in A$ (look at the qf-formula generated above for model with finite generators). For each $\langle \bar{a} \rangle$ in κ, there is a qf formula $\theta_{\langle \bar{a} \rangle}(\bar{x})$ such that $B \models \theta(\bar{b}) \leftrightarrow \langle \bar{b} \rangle \cong \langle \bar{a} \rangle$. Let $T_0 = \{\exists \bar{x} \theta_{\langle \bar{a} \rangle}(\bar{x}) : \langle \bar{a} \rangle \in \kappa\}$.
We want T_1 such that if $A \models T_1$, then age$(A) \subseteq \kappa$. The number of models in κ with n generators is $|\langle a_1, \ldots, a_n \rangle| \leq f(n)$. The number of models in κ with n generators is less than the total number of L-models with $\leq f(n)$ elements which is finite. Let $B_1, \ldots, B_l$ be all the models in κ with n generators. For each B_i , there is a formula $\theta_{B_i}(\bar{x})$ such that $M \models \theta_{B_i}(\bar{x}) \leftrightarrow \langle \bar{x} \rangle \cong B_i$. Let $T_1 = \{\text{`}\forall \bar{x} \bigvee_{i=1}^{l} \theta_{B_i}(\bar{x})\text{'} : \text{for } n \in \omega\}$.
To get ultrahomogeneous, use M is ultrahomogeneous if and only if $\forall \bar{a}, \bar{b}$ if $qf-\text{tp}(\bar{a}) = qf-\text{tp}(\bar{b}) \rightarrow \forall c \exists d$ such that $qf-\text{tp}(\bar{a}, c) = qf-\text{tp}(\bar{b}, d)$ and note that all the $qf-$tps are principal. That is $qf-\text{tp}(\bar{a}) = qf-\text{tp}(\bar{b})$ if and only if $\bigvee_{i=1}^{l} \theta_{B_i}(\bar{a}) \wedge \theta_{B_i}(\bar{b})$

Example 8.
Let κ be the class of finite graphs with no function symbols. The Fraïssé limit is a graph that has the property if $X, Y \subseteq R$ are finite, there is $v \in R$ such that $\forall w \in X$, $(v, w) \in E$ and $\forall w \in Y$, $(v, w) \notin E$.

8.2 Ultraproducts

Ultraproducts–although considerably more complicated than the procedures given thus far–are simply yet another means of generating new models from existing ones while preserving certain first-order characteristics. The method originates with Skolem in the 1930's and has been used extensively since Łos' work in 1955. The fundamental theorem of ultraproducts simply establishes the connection between ultraproducts and first-order characteristics; before proceeding to it, however, it's necessary to establish the notion of an *ultrafilter*.

8.2.1 Ultrafilters

Recall that $\mathcal{P}(I)$ gives the powerset of I–that is, the set of all subsets of I.

Filter
A *filter D over a non-empty set I* is a set $D \subset \mathcal{P}(I)$ such that

(i) $I \in D$

(ii) $X, Y \in D \Rightarrow X \cap Y \in D$

(iii) $X \in D$ and $X \subset Y \subset I \Rightarrow Y \in D$

That is, a filter D over I contains I, is closed under intersection, and is closed upwards under the subset relation. Common examples of filters over a set I are:

- The trivial filter $D = \{I\}$
- The improper filter $D = \mathcal{P}(I)$ (not actually a filter)
- For each $Y \subset I$, the filter $D = \{X \subset I : Y \subset X\}$; called *the principal filter generated by Y*
- The *Fréchet filter* $D = \{X \in \mathcal{P}(I) : X \text{ is cofinite}\}$

Proper Filter
A filter D over a set I is a *proper filter* if and only if it is not the improper filter

Generated Filter
Given a non-empty set I and any subset X of $P(I)$, the *filter generated by X* is the filter D given by

$$\cap\{F : X \subset F \text{ and } F \text{ is a filter over } I\}$$

A filter D is said to have the *finite intersection property* if and only if the intersection of any finite number of elements of D is non-empty.

► **Proposition 8.2.**
Let X be any subset of $\mathcal{P}(I)$ and let D_X be the filter generated by X. Then,

(i) D_X is a filter over I

(ii) D is the set of all $Y \in \mathcal{P}(I)$ such that either $Y = I$ or for some $Z_1, \ldots, Z_n \in X$, $Z_1 \cap \cdots \cap Z_n \subset Y$

(iii) D is a proper filter if and only if X has the finite intersection property

---

Ultrafilter
A filter D over a non-empty set I is an *ultrafilter over* I if and only if for all $X \in \mathcal{P}(I)$,

$$X \in D \Leftrightarrow I - X \notin D$$

It's common to simply say that D is an ultrafilter and assume that $I = \cup D$.

► **Proposition 8.3** (Ultrafilters are Maximal Proper Filters)**.**
The following are equivalent:

(i) D is an ultrafilter over I

(ii) D is a maximal proper filter over I

Proof Sketch.
Simply leverage the definition of ultrafilter

► **Theorem 8.5** (Ultrafilter Theorem)**.**
Let I be a non-empty set. If $X \subset \mathcal{P}(I)$ and X has the finite intersection property, then there exists an ultrafilter D over I such that $X \subset D$

► **Corollary** (Proper Filters Extend to Ultrafilters)**.**
Let I be a non-empty set. Any proper filter over I can be extended to an ultrafilter over I

8.3 Ehrenfeucht-Mostowski Models

At base, Ehrenfeucht-Mostowski models boil down to the thought that, given a particular theory T, by building a model of T around a particular group of elements called an indiscernible sequence, the properties of the resulting model as a whole can be controlled. In this area, a model A contains a linear ordering X just in case $X \subseteq \text{dom}(A)$, no further connection with $<_X$ is required.

8.3.1 Indiscernible Sequences

Taking a linearly order set $(X, <)$ and some $k \in \mathbb{N}$, we may generate a new set $[(X, <)]^k$, the set of all $<$-increasing k-tuples of elements from X. Let $f : [(X, <)]^k \to B$ be a partition of this set (elements in the same partition are mapped to the same element of B). It may help to think of f as a coloring of $[(X, <)]^k$ using $|B|$ distinct colors. Define

> **f-Indiscernible**
> A subset Y of X is *f-indiscernible* if and only if for any two $<$-increasing k-tuples $\bar{a}$, $\bar{b}$ from Y, $f\bar{a} = f\bar{b}$. In other words, f is constant across $[(Y, <)]^k$.

Put slightly differently, an f-indiscernible set is simply a set of elements from the original X all of whose $<$-increasing k-tuples reside in the same partition, all of whose $<$-increasing k-tuples have received the same color. Although it is far from obvious, the situation described above and combinatorial results guaranteeing the existence of f-indiscernible subsets of certain sizes paves the way to many model theoretic results.

To introduce a bit more notation, suppose that it can be shown in the situation above that there exists an f-indiscernible set Y of size α. Since all of the above were arbitrary, this holds irrespective of the actual sets/colors involved and merely in virtue of their relative sizes. Accordingly, let $|X| = \lambda$, $|Y| = \alpha$, and $|B| = \beta$. Then,

$$\lambda \to (\alpha)^k_\beta$$

That is, for any linearly order set X of size λ and set of β colors, a coloring of $[(X, <)]^k$ is guaranteed to have a set Y of size at least α such that all of $[(Y, <)]^k$ has the same color. This notation was chosen by Erdős and Rado so that if a result of the form above holds, then it remains true for any increase of λ on the left or decrease of α, β, or k on the right. Both the earliest and most important result for our purposes is:

► **Theorem 8.6** (Ramsey's Theorem - Infinite Form)**.**
For all $k, n \in \mathbb{N}$,

$$\aleph_0 \to (\aleph_0)^k_n$$

The key move in adapting these results to model theory is to think of our partitions, our colors, as determined by whether or not the given $<$-increasing k-tuple satisfies a particular combination of formulas.

> **Φ-Indiscernible Sequence**
> Let A be an L-structure and Φ a set of L-formulas all with k free variables. Suppose $Y \subseteq \text{dom}(A)$ and $<$ is a linear ordering on Y. Y is a *Φ-indiscernible sequence* in A if and only if for every formula $\varphi(\bar{x}) \in \Phi$ and every pair of $<$-increasing k tuples $\bar{a}, \bar{b}$,
> $$A \Vdash \varphi[\bar{a}] \leftrightarrow \varphi[\bar{b}]$$

Again, the above has little intuitive force in isolation, and so we tip our hand a bit. Given an ordering $<$ on $\text{dom}(A)$, note that for any tuple $\bar{a} \in [(\text{dom}(A), <)]^k$, either $A \Vdash \varphi[\bar{a}]$ or $A \Vdash \neg\varphi[\bar{a}]$ for each $\varphi \in \Phi$. Form a partition, then, on the tuples in $[(\text{dom}(A), <)]^k$ so that tuples in the same part all behave the same with respect to every $\varphi \in \Phi$. Each part in this partition is Φ-indiscernible. In truth, Φ-indiscernability isn't of considerable interest. It is, however, a stepping stone to something which is:

Indiscernible Sequence
Let A be an L-structure, $Y \subseteq \text{dom}(A)$, and $<$ a linear ordering on Y. $(Y, <)$ is an *indiscernible sequence in A* if and only if $(Y, <)$ is simultaneously a $\{\varphi\}$-indiscernible sequence for every φ in $\mathcal{L}$.

► **Theorem 8.7** (Constructing Models with Indiscernible Sequences)**.**
Let T be a theory with infinite models. For any infinite linear order $(I, <)$, there is a model M such that $M \Vdash T$ and M contains an indiscernible sequence $(x_i : i \in I)$

Proof.
Unsurprisingly, by compactness and Ramsey's theorem. Expand $\mathcal{L}$ to include constants c_i for every $i \in I$; take

$$T' := T \cup \{c_i \neq c_j : i, j \in I \wedge i \neq j\} \cup \{\varphi[c_{i_1}, \ldots, c_{i_m}] \to \varphi[c_{j_1}, \ldots, c_{j_m}] : \bar{i}, \bar{j} \in [(I, <)]^m, \varphi \text{ any } \mathcal{L}\text{-formula}\}$$

Any model of T' is as requested, and so it remains only to show finite satisfiability. Consider any finite subset of T'; we may assume with no loss that it contains T and sentences dictating that all the mentioned constants are not equal. All that remains are a finite number of formulas from

$$\{\varphi[c_{i_1}, \ldots, c_{i_m}] \to \varphi[c_{j_1}, \ldots, c_{j_m}] : \bar{i}, \bar{j} \in [(I, <)]^m, \varphi \text{ any } \mathcal{L}\text{-formula}\}$$

It follows that there is a maximum arity that occurs among the finite number of φ, say k. Take a countable model A of T and consider $[(\text{dom}(A), <)]^k$ where $<$ is an arbitrary linear ordering on $\text{dom}(A)$. Let Φ denote the union of the φ and note that we may think of each of the φ as taking k free variables. Thus,By Ramsey's theorem, there exists a countably infinite Φ-indiscernible sequence $Y \subseteq \text{dom}(A)$. Assign the constants mentioned so that the ordering $<_I$ is preserved by the chosen ordering on A–the result is a model of the finite subset of T'. □

8.3.2 E-M Models

Given any consistent theory T, recall that it can be Skolemized; that is, it can be extended to a Skolem theory T^* in an extended language $\mathcal{L}^*$ such that, for any model M of T, there is an $\mathcal{L}^*$-extension of M such that $M^* \Vdash T^*$. Furthermore, any generated submodel of M^* must be an elementary submodel. Last section showed that, given a theory, producing models with particular indiscernible sequences is a surprisingly easy task–so long as the theory has infinite models. This one not only defines, at long last, E-M models, but also details the results which give them such importance.

Ehrenfeucht-Mostowski Models
An *Ehrenfeucht-Mostowski* or *E-M model* is the Skolem hull of an indiscernible sequence

---

► **Lemma 8.5** (Order Indiscernible Automorphisms Induce a Submodel Automorphisms)**.**
Let T^* be a Skolem theory in $\mathcal{L}^*$, and suppose that $M \Vdash T^*$. If $Y = (y_i : i \in I)$ is an infinite indiscernible sequence for M and $\sigma : I \to I$ is an automorphism for the linear order I, then there is an automorphism $\sigma^* : \langle Y \rangle_M \to \langle Y \rangle_M$ extending σ.

Type of an Indiscernible Sequence
The *type of an indiscernible sequence X* in a model M is the set of all φ such that $M \Vdash \varphi[\bar{a}]$ for $\bar{a}$ from X

► **Lemma 8.6** (Type Preserving Order Changes)**.**
Let T^* be a Skolem theory in $\mathcal{L}^*$. Suppose that $Y = (y_i : i \in I)$ is an infinite sequence of order indiscernibles in a model M of T^*. If J is any infinite linear order, there exists a T-model N containing a sequence of order indiscernibles $X = (x_j : j \in J)$ and $\text{tp}(X) = \text{tp}(Y)$.

► **Lemma 8.7** (Order Embeddings Induce Submodel Embeddings)**.**
Let T^* be a Skolem theory in $\mathcal{L}^*$. Suppose that $Y = (y_i : i \in I)$ is a sequence of order indiscernibles in a model M of T^* and $X = (X_j : j \in J)$ is a sequence of order indiscernibles in a model N of T^* with $\text{tp}(X) = \text{tp}(Y)$, then any embedding $\tau : I \to J$ extends to an elementary embedding $\tau^* : \langle Y \rangle_M \to \langle X \rangle_N$.

Part II

Computability Theory

Chapter 9

Of Algorithms

Computability theory is, at base, simply an attempt to characterize the intuitive notion of an *algorithm*, a finite set of simple, step-by-step directions. Although both computability theory and the notion of algorithm existed long before computers, both are–to the non-logician–most manifest in these little boxes of metal and plastic. As any computer scientist will happily tell you, algorithms are wonderfully powerful things–especially when they can be executed at incredibly high speeds. Everything from watching a tv show, to playing a video game, to opening a word document is governed by step-by-step algorithms encoded into your computer; incredibly complex algorithms to be sure, but still the step-by-step sets of instructions that constitute our subject matter.

A natural first step towards formally characterizing the intuitive notion of *algorithm* is to generate a set of intuitive conditions on how algorithms behave; the list below will serve as our basis:

Algorithm

1. Uniquely determined by a finite set of instructions
2. Takes inputs from some set I and may produce outputs from some set O
3. Given input $i \in I$, an algorithm A determines a sequence of execution steps called *the computation of A on input I*
4. Given input $i \in I$, an algorithm A *may* reach a halting step with an output $o \in O$

In this last and only this last case, an algorithm A is said to output $o \in O$ on input $i \in I$ and we write '$f_A(i) \downarrow = o$' to convey this fact. Generating a partial function f_A in this manner gives *the function computed by A*, abbreviated of course to simply f_A. The function f_A is *partial* because nothing requires the algorithm A to have an output for every $i \in I$; thus, $\mathrm{dom}(f_A) \subseteq I$. As a last bit of symbolism, stating that g is a partial function taking a tuple of k inputs from a set X and possibly returning an output from a set Y will often be abbreviated, following standard mathematical practice, as $g : X^k \twoheadrightarrow Y$. Future references to g will often make the arity of the function explicit as a superscript, e.g. g^k.

To push forward and fully formalize the notion of algorithm, it's necessary to specify a 'programming language', a means with which to specify algorithms. There are a number of natural choices, and–historically–early research in computability theory gave rise to a myriad of suggestions. Luckily, they are all equivalent.

9.1 Turing Machines

Perhaps the best known of these suggestions is the *Turing machine*. A Turing machine M is:

- A finite alphabet A with distinguished symbol $* \in A$
- A finite set of S of states with both a start and a stop state in S
- A 2-way infinite tape divided into cells and a reading head

- Each cell has some $a \in A$ written in it (the $*$ from earlier can be interpreted as a blank)
- A finite set of quintuples of the form: $\langle q, a, b, R, s\rangle$ or $\langle q, a, b, L, s\rangle$ where $q, s \in S$, $a, b \in A$, and L/R specify movement right or left of the read/write head. Interpreted, in state q, reading a replace it with b and move one square right/left, going into state s.
- Additionally, if $\langle q, a, b, M, s\rangle, \langle q, a, c, N, t\rangle \in M$, then $b = c$, $M = N$, and $s = t$.

For those unfamiliar, the quintuples specify the behavior of the Turing machine, and–although tedious in practice–all known algorithms can, in principle, be carried out by an appropriately defined Turing machine.

Turing Computable
A partial function $f : O^k \twoheadrightarrow I$ is *Turing computable* if and only if there exists a Turing machine M such that $f = f_M$

9.1.1 The Church-Turing Thesis

The much affirmed Church-Turing thesis is the claim that an algorithm is computable in the intuitive sense if and only if it is Turing computable; that is, for any algorithm A there is a corresponding Turing machine M such that $f_A^k = f_M^k$. In support of this idea are the closure properties of the class of Turing computable functions, the lack of a counterexample, and the equivalence of Turing computability to other means of characterizing computability (register machines, μ-recursive derivation, etc.). We will, as usual, assume the thesis throughout our work.

9.2 Functions and Recursion

Studying computability theory via Turing machines is possible, but ultimately proves tedious. Instead, we adopt a different and more mathematically tractable strategy. The beginnings of this formulation appeared when the output o of an algorithm A on input i was written earlier as

$$f_A(i) = o$$

The standard function notation of mathematics, it turns out, carries over rather effortlessly into talk of algorithms, and so we leverage this conceptualization as the basis of our analysis. Notice, moreover, that while the function f_A doesn't uniquely specify the algorithm A–maybe a different set of instructions produces all the same input-output pairs!–it does capture the fact that an algorithm exists which produces such and such output on such and such input. Some thought shows, however, that this is all we truly care about; whether there exists any particular set of instructions with such and such behavior is the intriguing question–and easily captured within our function notation. The fact that we give up the ability to distinguish between particular sets of instructions doing the same thing is a very small loss in return for a great deal of simplicity.

Thinking of algorithms as functions naturally leads to the question of what sets the functions map things between; for complete generality, we might wish to always have the ability to specify an input set I and an output set O such that the 'algorithm' f under consideration is a partial map of the form $I \twoheadrightarrow O$. This is certainly doable, but a better alternative is readily available. Rather than allow wildly shifting input and output sets, assume that every function maps from $\mathbb{N}^k$ to $\mathbb{N}$ for some k; at first glance, this might seem to unnaturally limit the algorithms we may consider. As we'll soon see, however, the limitations imposed by this assumption are actually remarkably weak; simple encoding schemes will allow us to effortlessly represent any algorithm with both a countable input and output set.

Having solidified the interpretation of algorithms as functions from $\mathbb{N}^k$ to $\mathbb{N}$, the task is to lay out those function operations which allow us to construct all and only the computable functions. This task will carry us through the next three chapters, and centers on the notion of *recursion*. Recursion is, at base, simply defining something in terms of its previous values (often called an *inductive* definition in mathematics)–a process anyone with programming experience knows well. As an intuitively computable operation, recursion–alongside function composition and several simple 'starting' functions–will give us a first attempt at defining the computable functions, yielding a class known as the *primitive recursive functions*.

Chapter 10

The Primitive and μ-Recursive Functions

To sum the previous discussion, the primitive recursive functions are an attempt to generate formally a class of functions which corresponds exactly to the intuitively computable functions; in order to do so, we select three types of functions as obviously computable: the constant functions, the successor function, and the projection functions. Next, we select two means of combining these functions into more complicated functions which are also intuitively computable: function composition and primitive recursion.

10.1 The Primitive Recursive Functions

Formally,

Primitive Recursive Functions
The class of primitive recursive functions is the smallest class C of partial functions $f : \mathbb{N}^k \nrightarrow \mathbb{N}$ for any finite k containing

- $f(x_1, \ldots, x_k) = n$ for every $k > 0$ and $n \geq 0$; the *constant functions*
- $f(n) = n + 1$; the *successor function*
- $f(x_1, \ldots, x_k) = x_i$ for $1 \leq i \leq k$; the *projection functions*

and closed under

i. *Function Composition:* Let f be a k-ary function and $g_1, \ldots, g_k$ l-ary functions all in C, then the l-ary function $h(\bar{y}) = f(g_1(\bar{y}), \ldots, g_k(\bar{y}))$ is in C as well.

ii. *Primitive Recursion:* If h is a k-ary function, g a $k + 2$-ary function, and both are in C, then

$$f(0, \bar{x}) = h(\bar{x})$$

$$f(y + 1, \bar{x}) = g(y, f(y, \bar{x}), \bar{x})$$

Primitive Recursive
A function $f : \mathbb{N}^k \to \mathbb{N}$ is *primitive recursive* if and only if there exists a *primitive recursive derivation* $g_1, \ldots, g_k$ with $f = g_k$ where each g_i is either one of the basic primitive recursive functions or obtained from a previous g_j ($j < i$) by composition or primitive recursion.

10.1.1 Primitive Recursive Relations

A k-ary predicate or relation in $\mathbb{N}$ is a subset of $\mathbb{N}^k$; the characteristic function of a k-ary predicate/relation R is, contrary to the typical definition in other areas of mathematics, a function $\chi_R : \mathbb{N}^k \to \{0, 1\}$ such that

$$\chi_R(\bar{x}) = \begin{cases} 0 & \text{if } \bar{x} \in R \\ 1 & \text{if } \bar{x} \notin R \end{cases}$$

Primitive Recursive Relation
A relation R is *primitive recursive* if and only if the characteristic function of the relation, $\chi_R(\bar{x})$, is primitive recursive.

10.2 Primitive Recursive Operations

► **Lemma 10.1.**
If $h(\bar{z}, x)$ is a primitive recursive function, then $g_1(\bar{z}, y) = \prod_{x<y} h(\bar{z}, x)$ and $g_2(\bar{z}, y) = \sum_{x<y} h(\bar{z}, x)$ are primitive recursive functions as well.

Proof Sketch.
Induction

► **Lemma 10.2** (Primitive Recursive Operations)**.**
The class of primitive recursive relations is closed under $\neg$, $\wedge$, $\vee$, substitution of primitive recursive functions, permutation of variables, adding dummy variables, and bounded quantifiers.

Proof Sketch.

- Composition; let P be a k-ary primitive recursive relation and $f_1, \ldots, f_k$ primitive recursive functions. Define,

  $$Q(\bar{y}) \text{ if and only if } P(f_1(\bar{y}), \ldots, f_k(\bar{y}))$$

  Note that $\chi_Q(\bar{y}) = \chi_P(f_1(\bar{y}), \ldots, f_k(\bar{y}))$, and thus is primitive recursive.
- Bounded Quantifiers; Let R be a primitive recursive relation and define

  $$Q(\bar{z}, y) \text{ if and only if } \exists x < y, R(x, \bar{z})$$

  Note that $\chi_Q(\bar{z}, y) = \prod_{x<y} \chi_R(x, \bar{z})$, and thus is primitive recursive.

10.2.1 Piecewise Definition

► **Lemma 10.3** (Piecewise Definitions with Primitive Recursive Components)**.**
If

$$f(\bar{x}) = \begin{cases} h_1(x) & \text{if } R_1(\bar{x}) \\ h_2(\bar{x}) & \text{if } R_2(\bar{x}) \\ \vdots & \\ h_k(\bar{x}) & \text{if } R_k(\bar{x}) \end{cases}$$

where each h_i is a primitive recursive function, each R_i is a primitive recursive relation, and the R_i are both mutually exclusive and exhaustive, then f is itself primitive recursive.

Proof.
Define f by

$$f(\bar{x}) = h_1(\bar{x})\overline{\text{sg}}(\chi_{R_1}(\bar{x})) + h_2(\bar{x})\overline{\text{sg}}(\chi_{R_2}(\bar{x})) + \cdots + h_k(\bar{x})\overline{\text{sg}}(\chi_{R_k}(\bar{x}))$$

and note that this is primitive recursive. □

10.2.2 Bounded Search

Continuing the trend, we now show that the primitive recursive functions are closed under bounded search. The μ or search operator takes on the least value satisfying some relation; thus, $\mu z(R(z))$ is 'the least z such that $R(z)$'; in this section, the μ operator will be bounded, represented by $\mu_{<y}$ for a given bound y.

► **Theorem 10.1** (Bounded Search is Primitive Recursive)**.**
Let R be a primitive recursive relation; if

$$f(\bar{x}, y) = \begin{cases} (\mu_{<y} z)[R(z, \bar{x})] & \text{if such a } z \text{ exists} \\ y & \text{otherwise} \end{cases}$$

then f is a primitive recursive function.

Proof.

$$f(\bar{x}, 0) = 0$$

$$f(\bar{x}, n+1) = \begin{cases} f(\bar{x}, n) & \text{if } R(f(\bar{x}, n), \bar{x}) \\ n+1 & \text{otherwise} \end{cases}$$

□

10.2.3 Coding Sequences

Given a finite sequence $\langle a_0, \ldots, a_k \rangle$, we may encode it in a single number as

$$\ulcorner \langle a_0, \ldots, a_k \rangle \urcorner = p_0^{a_0+1} p_1^{a_1+1} \ldots p_k^{a_k+1}$$

where p_i denotes the ith prime number. In addition, we set

$$\ulcorner \emptyset \urcorner = 0$$

Note that, for every k,

$$f_k(a_0, \ldots, a_k) = \ulcorner \langle a_0, \ldots, a_k \rangle \urcorner$$

is a primitive recursive function. More generally, the following primitive recursive predicates and functions are constructed to make working with encoded sequences easy.

$Seq(n)$ if and only if n is the encoding of a sequence

$$lh(n) = \begin{cases} k & \text{if } n = \ulcorner \langle a_0, \ldots, a_{k-1} \rangle \urcorner \text{ for some sequence } \langle a_0, \ldots, a_{k-1} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$(n)_i = \begin{cases} a_i & \text{if } Seq(n) \text{ and } i < lh(n) \\ 0 & \text{otherwise} \end{cases}$$ s

Less formally, Seq indicates whether a number encodes a sequence, lh gives the length of the sequence a number encodes if it encodes one and 0 otherwise, while $(n)_i$ returns the ith entry (starting at 0) from the encoded sequence if i is less than the length of the sequence, 0 otherwise.

► **Proposition 10.1.**

i. Seq is a primitive recursive predicate

ii. lh is a primitive recursive function

iii. The function $d(n, i) = (n)_i$ is primitive recursive

Proof.

$Seq(n)$ if and only if $n = 0 \vee \forall p \leq n[(Prime(p) \wedge Div(n, p)) \rightarrow (\forall q < p[Prime(q) \rightarrow Div(q, n)])$

$$lh(n) = \begin{cases} \sum_{x \leq n} \bar{sg}(\chi_{prime(x) \wedge Div(n,x)}(x)) & \text{if } Seq(n) \\ 0 & \text{otherwise} \end{cases}$$

Finally, taking $f(n)$ to give the nth prime:

$$(n)_i = \begin{cases} \mu_{<n} x[Div(n, f(i)^x) \wedge \neg Div(n, f(i)^{x+1})] - 1 & \text{if } i < lh(n) \\ 0 & \text{otherwise} \end{cases}$$

□

10.2.4 Simultaneous and Course of Values Recursion

► **Theorem 10.2** (Simultaneous Recursion is Primitive Recursive)**.**
Suppose that

$$f(0) = m_0$$

$$g(0) = n_0$$

$$f(n + 1) = h_1(f(n), g(n))$$

$$g(n + 1) = h_2(f(n), g(n))$$

If h_1, h_2 are primitive recursive, then f, g are as well.

Proof.
Let f, g, h_1, and h_2 be as above. Let f_2 be the length 2 encoding function and define φ as follows:

$$\varphi(0) = f_2(m_0, n_0)$$

$$\varphi(n+1) = f_2(h_1((\varphi(n))_0, (\varphi(n))_1), h_2((\varphi(n))_0, (\varphi(n))_1))$$

Note that not only is φ primitive recursive, but $\varphi(n)$ also encodes the values $f(n), g(n)$ for every n. Thus, $f(n)$ and $g(n)$ can be retrieved as follows:

$$f(n) = (\varphi(n))_0$$

$$g(n) = (\varphi(n))_1$$

Both of which are, of course, primitive recursive constructions. By the arbitrary nature of f, g, h_1 and h_2 the theorem is established. □

► **Theorem 10.3** (Course of Values Recursion is Primitive Recursive)**.**
If $f(n+1)$ is always definable via a single primitive recursion procedure Q given singleton access to all of its preceding values $f(0), \ldots, f(n)$, then f itself is primitive recursive.

Proof.
Let $p(n)$ be the primitive recursive function which produces the nth prime. Define

$$\varphi(0) = (p(0))^{f(0)}$$

$$\varphi(n+1) = \varphi(n)[(p(n+1))^{Q'(\varphi(n))}]$$

where Q' is the primitive recursive procedure Q with singleton access to all of f's preceding values given by the decoding function $(n)_i$ on $\varphi(n)$, which–by construction–encodes each of them. Note that, by assumption, $f(0)$ is definable by a primitive recursive procedure (it has no previous values), and so the brief use above is not illicit. It follows immediately that φ is primitive recursive. Using φ, it's now possible to give a primitive recursive construction of f itself:

$$f(n) = (\varphi(n))_n$$

It follows by the arbitrary nature of f and Q that course of values recursion is primitive recursive, and thus the theorem is proved. □

10.3 The Computable Functions?

The primitive recursive functions, then, are a surprisingly large class; including the vast majority of everyday mathematical functions and operations. It's natural, then, to wonder whether we've already succeeded; perhaps all *algorithm* means is 'primitive recursive function'! Life, of course, is seldom so compliant.

► **Theorem 10.4** (Primitive Recursive ⊂ Computable)**.**
There exists a computable function which is not primitive recursive.

Proof Sketch.
Utilizing the encoding scheme already defined, we may encode *derivations* of primitive recursive functions; more importantly, however, we may encode them in such a way that it can be checked–primitive recursively–whether or not a given number e encodes a valid derivation of a primitive recursive function. Note that, by simply proceeding through the natural numbers, we now have a primitive recursive enumeration within which every last primitive recursive function's encoding must appear. We may go further, however. Once a derivation is fixed, it's certainly algorithmically possible to run the derived function on any appropriate input; that is, we should be able to construct a computable function Θ such that

$$\Theta(e, x) = \begin{cases} \vartheta_e(x) & \text{if } e \text{ encodes a primitive recursive function } \vartheta_e \\ 0 & \text{otherwise} \end{cases}$$

But, with such a computable Θ in hand, we may channel the spirit of Cantor and define $\gamma : \mathbb{N} \twoheadrightarrow \mathbb{N}$ by

$$\gamma(x) = \Theta(x, x) + 1$$

A moment's thought shows that not only should γ be computable, but it also couldn't have been listed in our enumeration! It follows immediately that there must exist computable functions which are not primitive recursive, as asserted.

10.4 Unbounded Search & μ-Recursion

μ-recursion grants the ability for unbounded search, to simply continue looking until a value is found–or even to never stop looking at all. This operation represents a fundamental increase in the strength of the operations we allow ourselves, but at a serious cost; for the very first time, the functions under consideration need not be total, with μ-recursion in play its possible that a function is simply undefined on some inputs. In particular, there are two sources of incompleteness to be wary of; the first arises quite naturally from the unbounded μ operator alone:

Unbounded μ-Operator
For $R(x_1, \ldots, x_k, y)$ a $k+1$-ary relation, the expression

$$\mu y[R(\bar{x}, y)]$$

gives the least number y such that, for a given $\bar{x}$, $R(\bar{x}, y)$ holds; if no such y exists, the expression is undetermined.

That is, an expression containing the μ-operator may be undefined because a satisfying y cannot be found. The definition of μ-recursion, however, hints at a more subtle difficulty:

μ-Recursion
Given $\theta(x_1, \ldots, x_k, y)$ a $k+1$-ary function, the result of *μ-recursion* on θ with respect to y is the k-ary function,

$$\rho(\bar{x}) = \mu y[\theta(\bar{x}, y)\downarrow = 0 \wedge (\forall z \leq y)[\theta(\bar{x}, z)\downarrow]]$$

Note the shift from talk of relations in the definition of the μ-operator to discussion of a partial function in the definition of μ-recursion. μ-recursion is not restricted to primitive recursive or even μ-recursive relations; rather, we allow μ-recursion over any μ-recursive function–*even though many are partial*. This allowance is necessary to achieve the class of functions we desire, but requires users of μ-recursion to keep the '$\wedge(\forall z \leq y)[\theta(\bar{x}, z)\downarrow]$' clause in mind; while there might be a least y satisfying $\theta(\bar{x}, y)$, μ-recursion on θ will be undefined if $\theta(\bar{x}, z)$ isn't defined for all lesser natural numbers.

Despite this, it's common to omit the '$\wedge(\forall z \leq y)[\theta(\bar{x}, z)\downarrow]$' clause for μ-recursion on primitive recursive or μ-recursive relations; since these are defined everywhere, the result always behaves as expected. Although it is tempting to do so, the same liberties must not be taken with relations that are not μ-recursive; indeed, doing so represents an unwarranted expansion of the class of functions under consideration.

> **μ-Recursive**
> A function f is called *μ-recursive* if and only if f can be built using primitive recursive functions by means of composition, primitive recursion, and μ-recursion.

With this last addition, it's possible to prove that–for any Turing machine M–the k-ary function f_M^k is μ-recursive; in other words, the μ-recursive functions are the computable functions, assuming the Church-Turing thesis.

► **Theorem 10.5** (Turing-Computable = μ-Recursive)**.**
Given a partial function $f : \mathbb{N}^k \to \mathbb{N}$, then f is Turing computable if and only if f is μ-recursive.

Proof Sketch.

($\to$)
Fix a Turing machine t and the partial function it defines $f_t : \mathbb{N} \to \mathbb{N}$. Define a primitive recursive predicate $T.Comp_t(\ulcorner(x, y, L)\urcorner)$ where $C = \ulcorner(x, y, L)\urcorner$ with L coding a terminating Turing computation with machine t, input x, and output y:

$$T.Comp_t(\ulcorner(x, y, L)\urcorner) \text{ if and only if } (\mu C[T.Comp_t(C) \wedge (C)_0 = x])_1$$

Informally, a terminating Turing comp with (t, x, y) is a sequence L of machine status descriptions. A status description is a sequence $l = \ulcorner(T, h, i)\urcorner$ where T specifies the tape (a finite sequence of 0's and 1's) , h the head ($< lh(T)$), and i (in $\{0, 1, \ldots, s\}$ if machine has states 0 through s) the Turing machine's state. Thus, $L = \ulcorner(\ulcorner(T, h_0, i_0)\urcorner, \ldots, \ulcorner(T, h_m, i_m)\urcorner)\urcorner$ and initial case $i_0 = 1$, $T_0 = (0, 1, 1, \ldots, 1)(x \text{ many } 1\text{'s})$, and $h_0 = 0$ and end case $i_k = 0$ (termination state), $T_k = (\ldots, 1, 1, 1, \ldots, 1, 0 \ldots)$ (y many 1's) i_k case $\forall i < lh(L)$ and how $(T_{i+1}, h_{i+1}, i_{i+1})$ relates to (T_0, h_i, i_i)

10.5 The Computable Functions

> **Computable Functions**
> The class of functions which can be computed by algorithm; according to the Church-Turing thesis, the term is synonymous with the μ-recursive/Turing-computable functions.

It's important to note that a member of the computable functions may be partial; to distinguish the set of total functions within the set of computable functions, we define

> **Total Computable Functions**
> The subset of the computable functions containing all and only the total functions; that is, functions defined for every $n \in \mathbb{N}$; also known as the total recursive or recursive functions

10.5.1 Computable Relations

As earlier, a k-ary predicate or relation in $\mathbb{N}$ is a subset of $\mathbb{N}^k$; the characteristic function of a k-ary predicate/relation R is a function $\chi_R : \mathbb{N}^k \to \{0, 1\}$ such that

$$\chi_R(\bar{x}) = \begin{cases} 0 & \text{if } \bar{x} \in R \\ 1 & \text{if } \bar{x} \notin R \end{cases}$$

Computable Relation
A relation R is *computable* if and only if the characteristic function of the relation, $\chi_R(\bar{x})$, is a total computable function; also called *recursive*.

As a special case, note that for $k = 1$ we have also given a definition for what it means to say that a set $A \subseteq \mathbb{N}$ is *computable*.

Chapter 11

The Fundamental Theorems of Computability Theory

As the section on primitive recursion argued, the computable functions should, intuitively, be capable of enumerating themselves–proving that they actually can will occupy most of our time in this section. The constructions of this section not only give a host of basic results about the computable functions, but also will showcase a number of recurring tools in computability theory.

11.1 The Padding Lemma

As earlier with the primitive recursive functions, it's easy to define a primitive recursive predicate which indicates whether a given natural number encodes (according to our scheme) a valid derivation of a μ-recursive function; these natural numbers are called *indices* or *Gödel numbers*.

► **Lemma 11.1** (The Padding Lemma)**.**
Each computable function ϑ_e has $\aleph_0$ many indices, and for each e, we may construct an infinite set A_e such that for every $y \in A_e$, $\vartheta_y = \vartheta_e$.

Proof Sketch.
Simply add extraneous steps to the derivation encoded by e

11.2 The Normal Form Theorem

The normal form theorem is the first major step in constructing an enumeration of the computable functions and does precisely what its name indicates; it provides a standard format capable of representing every possible computable function. The key to this form is Kleene's T-predicate–a primitive recursive relation $T(e, \bar{x}, C)$ which verifies C encodes a computation of the function encoded by e on input x.

► **Theorem 11.1** (The Normal Form Theorem)**.**
For every $k \geq 1$, there exists a primitive recursive predicate $T^k(e, \bar{x}, C)$–the Kleene T-predicate–with $\bar{x} = (x_0, \ldots, x_{k-1})$ and a 1-ary primitive recursive function U such that

$$\vartheta_e(\bar{x}) = U(\mu C[T^k(e, \bar{x}, C)])$$

As we'll see, it is Kleene's T predicate and the μ-recursion which do the work above; the primitive recursive U does little more that output the appropriate value.

Proof Sketch.
The key to defining Kleene's T predicate is the notion of a *computation*; given a derivation $f_0, \ldots, f_n$, a computation that $f_n(\bar{x}) = y$ is a list of statements of the form:

$$f_{i_0}(\bar{x}_0) = y_0$$

$$f_{i_1}(\bar{x}_1) = y_1$$

$$\vdots$$

$$f_n(\bar{x}) = y$$

where each line $f_{i_a}(\bar{x}_a) = y_a$ is justified by the derivation of f_n and earlier equations on the computation list. Less formally, a computation should be thought of as a proof that $f_n(\bar{x}) = y$ for some $\bar{x}, y$. Just as with derivations, constructing a primitive recursive predicate to test for valid computations is little more than picking a representation scheme and working through the details. Once this is accomplished, Kleene's T predicate is defined as

$$T^k(e, \bar{x}, C) = \begin{cases} 0 & C \text{ encodes a valid computation of } \vartheta_e(\bar{x}) \\ 1 & \text{otherwise} \end{cases}$$

while U is defined to simply pick out the value of $\vartheta(\bar{x})$ from C.

11.3 The Enumeration Theorem

A function $\Theta : \mathbb{N}^{k+1} \twoheadrightarrow \mathbb{N}$ is k-ary *universal* if and only if for any k-ary computable function f, there exists $e \in \mathbb{N}$ such that $\Theta(e, \bar{x}) = f(\bar{x})$.

► **Theorem 11.2** (The Enumeration Theorem)**.**
For every $k \in \mathbb{N}$, there exists a computable k-ary universal function, Θ^k.

Proof Sketch.
For any k, simply define

$$\Theta^k(e, \bar{x}) = U(\mu C[T(e, \bar{x}, C)])$$

using the k-ary Kleene T-predicate.

Note that in $\Theta^k(e, \bar{x}) = U(\mu C[T(e, \bar{x}, C)])$ above U and T are primitive recursive relations and μ appears just once. It follows immediately that only a single application of the μ-operator in a derivation suffices to produce any μ-recursive function.

► **Corollary.**
Let $f : \mathbb{N} \to \mathbb{N}$ be computable, total, and a bijection. Then, f^{-1} is also computable

---

Proof.
Define μ-recursive $g : \mathbb{N} \to \mathbb{N}$ by, informally, fix $e^* \in \mathbb{N}$ such that $f = \Phi^1_{e^*}$ (by enum. thm., e^* exists). Given $y \in \mathbb{N}$, to compute $g(y)$, search for some computation C and $x \in \mathbb{N}$ such that C 'says' '$f(x) = y$'. Then, output x. (that defines $g(y) = x$).
Formally,

$$\Big((\mu C[\exists x, L \leq C[C = \ulcorner(e^*, \ulcorner(x)\urcorner, y, L)\urcorner \wedge Comp(C)]])_1\Big)_0$$

or, since we have total, $\mu x[f(x) = y]$ (the first construction will skip non-total portions of functions, the latter may stall). □

11.4 The s^m_n Theorem

Also known as the parameter or parameterization theorem, the s^m_n theorem establishes, for any n and m, the existence of a primitive recursive function s^m_n which–given the encoding of an $m+n$-ary function f and m 'parameters'–picks out an encoding which represents f, but with its first m inputs fixed to the given parameters. At first blush, this seems like a rather trivial task for such a vaunted theorem; implicit in this operation, however, is that the s^m_n functions effectively allow traversal through an infinite hierarchy of enumeration functions $\Theta^1, \Theta^2, \Theta^3, \ldots$ which are based in the same encoding scheme. The uses for such traversal are, as we will see, quite extensive.

► **Theorem 11.3** (s^m_n Theorem)**.**
For every $m, n \geq 1$, there is a primitive recursive, $m+1$-ary function s^m_n such that for every $e, a_1, \ldots, a_m$, and $x_1, \ldots, x_n$

$$\Theta^{m+n}(e, \bar{a}, \bar{x}) = \Theta^n(s^m_n(e, \bar{a}), \bar{x})$$

Proof Sketch.

An especially useful variant of this theorem is the *index function theorem*:

► **Theorem 11.4** (Index Function Theorem)**.**
If $f(a_1, \ldots, a_m, x_1, \ldots, x_n)$ is a $m+n$-ary primitive recursive function, then there is a particular primitive recursive function $s(a_1, \ldots, a_m)$–called the index function of f–such that for every $\bar{a}$ and $\bar{x}$,

$$\vartheta^n_{S(\bar{a})}(\bar{x}) = f(\bar{a}, \bar{x})$$

Thus, given $f(\bar{a}, \bar{x})$, we may construct a list of parameterized functions $f_{\bar{a}}(\bar{x}) = f(\bar{a}, \bar{x})$ by varying the parameters $\bar{a}$. Using the theorem above, it follows that there is a primitive recursive function s that will allows us to convert each set $\bar{a}$ of parameters into the index of the corresponding parameterized function.

Proof Sketch.

11.5 The Halting Problem

The halting problem for a computable k-ary function ϑ_e and input $\bar{x}$ asks whether or not ϑ_e halts on input $\bar{x}$, which of $\vartheta_e(\bar{x})\downarrow$ and $\vartheta_e(\bar{x})\uparrow$ obtains. Of course, this question by itself isn't incredibly interesting; whether there is, for a given arity k, a $k+1$-ary, computable function which is capable of solving the halting problem for any encoding e of a k-ary computable function and any input $\bar{x}$, however, is. That is, we desire a computable ϑ_H such that

$$\vartheta_H(e,\bar{x}) = \begin{cases} 0 & \Theta^k(e,\bar{x})\downarrow \\ 1 & \text{otherwise} \end{cases}$$

As it turns out, this is provably impossible; that is, there exist functions which are demonstrably non-computable.

► **Theorem 11.5** (The Halting Problem is not Computable)**.**
For any $k > 0$, there is no computable function which solves the k-ary halting problem.

Proof Sketch.
Suppose $\vartheta_H(e,\bar{x})$ solves the k-ary halting problem. Define a new k-ary function f as

$$f(\bar{x}) = \begin{cases} 0 & \vartheta_H(x_0,\bar{x})\downarrow = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

and note that f is clearly computable; it follows that $f = \vartheta_{e_f}$ for some e_f. Consider, then, $f(e_f,\ldots,e_f)$; if $\vartheta_H(e_f,\ldots,e_f)\downarrow = 1$, then $f(e_f,\ldots,e_f)\downarrow$–contra $\vartheta_H(e_f,\ldots,e_f)$. If not $\vartheta_H(e_f,\ldots,e_f)\downarrow = 1$, then $f(e_f,\ldots,e_f)\uparrow$–contra $\vartheta_H(e_f,\ldots,e_f)$ yet again. Thus, in either case, $\vartheta_H(e_f,\ldots,e_f)$ answers incorrectly.

► **Corollary.**
Let f be any computable function. Then, there is an 'inverse' g for f such that g is computable, where an 'inverse' is taken to mean

$$g(y)\downarrow = x \text{ if and only if } y \in \mathit{range}(f) \wedge \forall y \in \mathit{range}(f)[f(g(y)) = y]$$

Proof Sketch.
See the earlier proof for f total, bijective

11.6 Kleene's Fixed Point Theorem

The following result, owed like much of our work to Kleene, is important enough to have earned the monikers *the 2nd recursion theorem* and even just *the recursion theorem*. v u

---

► **Theorem 11.6** (The Kleene Fixed Point Theorem)**.**
Suppose $f : \mathbb{N} \to \mathbb{N}$ is a total computable function; then for every k, there exists an e such that

$$\vartheta^k_e \simeq \vartheta^k_{f(e)}$$

In other words, e is a fixed point of f vis-à-vis the enumeration function Θ^k.

Proof Sketch.

Suppose, for the moment, that we can computably construct a table of k-ary computable functions (by specifying their indices in our K-ary enumeration) that has the following property:

($\star$) For every computable function f, there is a corresponding row r_f such that for every $n \in \mathbb{N}$,

(i) if $f(n)\downarrow$, the function $\vartheta_{f(n)}$ is the nth entry in r_f

(ii) if $f(n)\uparrow$, the nth entry in r_f is instead the function which halts on no input

Given such a table and the closure properties of the computable functions, it's easy to see that the diagonal of the table must itself appear as a row of the table. We thus have the following picture:

$$
\begin{array}{lccccccc}
 & \boldsymbol{\vartheta_{e_{1,1}}} & \vartheta_{e_{1,2}} & \vartheta_{e_{1,3}} & \cdots & & & \\
 & \vartheta_{e_{2,1}} & \boldsymbol{\vartheta_{e_{2,2}}} & \vartheta_{e_{2,3}} & \cdots & & & \\
 & \vartheta_{e_{3,1}} & \vartheta_{e_{3,2}} & \boldsymbol{\vartheta_{e_{3,3}}} & \cdots & & & \\
 & \vdots & \vdots & \vdots & \ddots & & & \\
\text{Diagonal, row } n\text{:} & \boldsymbol{\vartheta_{e_{1,1}}} & \boldsymbol{\vartheta_{e_{2,2}}} & \boldsymbol{\vartheta_{e_{3,3}}} & \cdots & \boldsymbol{\vartheta_{e_{n,n}}} & \cdots & \\
 & \vdots & \vdots & \vdots & & & \ddots &
\end{array}
$$

Moreover, since our construction of the table is computable, we have a means of computably generating the indices of just the diagonal of our table. Taking an arbitrary total computable f, then, we may construct a new row f(Diagonal) using our procedure for generating the diagonal row (recall, by ($\star$), there must be a corresponding row). Note, however, that this new row must intersect the diagonal, giving a fixed point of precisely the type required:

$$
\begin{array}{lccccccccc}
 & \boldsymbol{\vartheta_{e_{1,1}}} & \vartheta_{e_{1,2}} & \vartheta_{e_{1,3}} & \cdots & & & & & \\
 & \vartheta_{e_{2,1}} & \boldsymbol{\vartheta_{e_{2,2}}} & \vartheta_{e_{2,3}} & \cdots & & & & & \\
 & \vartheta_{e_{3,1}} & \vartheta_{e_{3,2}} & \boldsymbol{\vartheta_{e_{3,3}}} & \cdots & & & & & \\
 & \vdots & \vdots & \vdots & \ddots & & & & & \\
\text{Diagonal, row } n\text{:} & \boldsymbol{\vartheta_{e_{1,1}}} & \boldsymbol{\vartheta_{e_{2,2}}} & \boldsymbol{\vartheta_{e_{3,3}}} & \cdots & \boldsymbol{\vartheta_{e_{n,n}}} & \cdots & & & \\
 & \vdots & \vdots & \vdots & & & \ddots & & & \\
\text{f(Diagonal), row } m\text{:} & \vartheta_{f(e_{1,1})} & \vartheta_{f(e_{2,2})} & \vartheta_{f(e_{3,3})} & \cdots & \vartheta_{f(e_{n,n})} & \cdots & \boldsymbol{\vartheta_{e_{m,m}} \simeq \vartheta_{f(e_{m,m})}} & \cdots &
\end{array}
$$

As it turns out, it's easy to generate a table meeting the requirements above; the $k = 1$ case is given explicitly, but the strategy easily generalizes. Consider

$$t(x,y,z) = \begin{cases} \vartheta^1_{\Theta^1(x,y)}(z) & \Theta^1(x,y)\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Taking x as our row variable and y as our column, the t above generates a table of the required type; moreover, the process is computable. Define:

$$t(x,y,z) = \Theta^1(\Theta^1(x,y),z)$$

and so the s^m_n theorem guarantees a total computable s^2_1 such that, for any r and c,

$$t(r,c,z) \simeq \vartheta^3_{e_t}(r,c,z) \simeq \vartheta_{s^2_1(e_t,r,c)}(z)$$

This $s^2_1(e_t,x,y)$ allows, then, the effective generation of our table, while $s^2_1(e_t,x,x)$ generates the diagonal indices. We may carry out the remainder of the argument as above; since $s^2_1(e_t,x,x)$ generates the diagonal indices, m is–by the construction of t–the index of $f(s^2_1(e_t,x,x))$, while the actual fixed point is given by $s^2_1(e_t,m,m)$.

$$\vartheta^1_{s^2_1(e_t,m,m)}(z) \simeq \Theta^3(e_t,m,m,z)$$

By the definition of e_t,

$$\simeq \Theta^1(\Theta^1(m,m),z)$$

and, finally, by the definition of m,

$$\simeq \Theta^1(f(s^2_1(e_t,m,m)),z)$$
$$\simeq \vartheta^1_{f(s^2_1(e_t,m,m))}(z)$$

An equivalent way of viewing the construction above is to notice that the fixed point e^* above, viewed as a function, *forces* itself to be a fixed point relative to f by computing the function $\vartheta_{f(e^*)}$ and then mimicking this function's behavior.

► **Corollary.**
Let f be total computable; for every k there exists e such that

$$W^k_e = W^k_{f(e)}$$

Example 9.
As an alternative to our original proof of Rice's Theorem, suppose I is a computable index set that is non-trivial ($\neq \mathbb{N}, \emptyset$). Get e using recursion theorem, define

$$\varphi_{f(e)}(y) = \begin{cases} \varphi_b(y) & \text{if } I(e) \\ \varphi_a(y) & \text{if } \bar{I}(e) \end{cases}$$

where $a \notin I$, $b \in I$. If I is computable, then there is a primitive recursive f such that $\varphi(e) = \varphi_{f(e)}$.

Example 10.
Is there a partial recursive ψ such that for every e, if W_e is finite, then $\psi(e)\downarrow$ and $|W_e| \leq \psi(e)$? Define

$$W_e = \begin{cases} \emptyset & \text{if } \psi(e) \uparrow \\ \{0, \ldots, \psi(e)+1\} & \text{if } \psi(e)\downarrow \end{cases}$$

By the S_n^m theorem, we may obtain such an e. Define f so that

$$W_{f(e)} = \begin{cases} \emptyset & \text{if } \psi(e) \uparrow \\ \{0, \ldots, \psi(e)+1\} & \text{if } \psi(e)\downarrow \end{cases}$$

$\psi(e)\downarrow$ since otherwise $W_e = \emptyset$ is finite, and $\psi(e)\downarrow$. But then, $|W_e| = \psi(e)+1$.

► **Theorem 11.7** (Uniform Fixed Point Theorem)**.**
Let $f(x,y)$ be total computable. Then, for any k, there is a total computable, unary function $\vartheta_e^1(y)$ such that

$$\vartheta^k_{\vartheta^1_e(x)} \simeq \vartheta^k_{f(x,\vartheta^1_e(x))}$$

► **Corollary** (Double Fixed Point Theorem)**.**
Suppose $f(x,y)$ and $g(x,y)$ are total computable; then, there is a pair of indices a, b such that

$$\vartheta_a = \vartheta_{f(a,b)}$$

$$\vartheta_b = \vartheta_{g(a,b)}$$

In other words, there are a, b such that $W_a = \{b\}$ and $W_b = \{a\}$.

Proof.
Get a total recursive $a = a(b)$ such that $\varphi_{a(b)} = \varphi_{f(a(b),b)}$ for every b (by URT). Now, fix b such that

$$\varphi_b = \varphi_{g(a(b),b)}$$

by the ordinary recursion theorem. The pair $(b, a(b))$ then works. □

Example 11.
Suppose ψ is computable and $A \subseteq \mathbb{N}$, and for all e,

$$W_e \subseteq A \Rightarrow \psi(e)\downarrow \text{ and } \psi(e) \in A - W_e$$

Claim: A is productive. The productive function is f where $f(y) =?$ There is a total computable $e(y)$ such that for every y,

$$W_{e_y} = \begin{cases} \emptyset & \psi(e_y) \uparrow \\ W_y & \psi(e_y)\downarrow \end{cases}$$

$$W_{s_y(z)} = \begin{cases} \emptyset & \psi(z) \uparrow \\ W_y & \psi(z)\downarrow \end{cases}$$

where $s(y,z)$ is total computable by S_n^m. Our productive function is $f(y) = \psi(e_y)$.

Chapter 12

Non-Computable Relations

Two chapters ago, the definition of a computable relation was given; it has been reproduced below for ease of reference:

> **Computable Relation**
> A relation R is *computable* if and only if the characteristic function of the relation, $\chi_R(\bar{x})$, is a total computable function; also called *recursive* or *decidable*.

Computable relations, then, have a computable decision procedure for whether any particular input is or is not in the set; rather appropriately, computable relations are also known as decidable, a computable function can always decide whether a given input is or is not in the set. This is, undoubtedly, a nice property for a relation to have, but it's far from necessary. Indeed, it's natural to immediately distinguish to other classes of relations:

> **Computably Verifiable**
> A relation R is *computably verifiable* if and only if it has a computable verifying function; also known as *semi-recursive* or *semi-decidable*

> **Computably Refutable**
> A relation R is *computably refutable* if and only if it has a computable refuting function; also known as *co-semi-recursive*

A *verifying function* (also known as a *positive characteristic function*) for a relation R is

$$\chi^{+}_{R}(\bar{x}) = \begin{cases} 0 & \text{if } R(\bar{x}) \\ \uparrow & \text{otherwise} \end{cases}$$

Similarly, a *refuting function* (also known as a *negative characteristic function*) for a relation R is

$$\chi^{-}_{R}(\bar{x}) = \begin{cases} \uparrow & \text{otherwise} \\ 1 & \text{if } \neg R(\bar{x}) \end{cases}$$

Before moving forward, we introduce two further notational abbreviations:

$$W_e = \text{domain}(\vartheta^1_e)$$

$$E_e = \text{range}(\vartheta^1_e)$$

and offer a warning. Up until this point, all relations have been well-behaved, i.e. computable, and so it's easy to give little thought to familiar operations on them. From here forward, however, the relations under consideration need not be well-behaved, and thus–for example–μ-recursion is no longer always admissible on whichever relation currently holds our interest. Failure to internalize this expansion in our topic matter is liable to lead to large errors, and it may serve the reader well to refamiliarize themselves with the limitations of the operations covered thus far vis-a-vis relations.

► **Theorem 12.1** (Computably Verifiable Relations are Domains)**.**
A relation R is computably verifiable if and only if R is domain(f) for some computable function f; that is, every R is W_e for some e

Proof Sketch.
By definition, a computable relation R has a computable verifying function ϑ_e–but then R simply is W_e

► **Corollary** (Computably Refutable Relations are the Complements of Domains)**.**
A relation R is computably refutable if and only if R is $\mathbb{N} -$ domain(f) for some computable function f; that is, every R is $\mathbb{N} - W_e$ for some e

12.1 A Three-Fold Distinction

Throughout the next few sections we'll explore various different ways of viewing and manipulating relations; thus far, a three-fold distinction has been introduced: only computably verifiable relations, only computably refutable relations, and both computably verifiable and refutable relations (that is, computable relations). This is one way of viewing relations, but it is by no means the only reasonable one. An alternative tact–and one which ultimately proves quite valuable–is to group relations via the formulas which define them.

Of course, definability is a relative concept; in this case, the language of choice is the *language of first-order arithmetic*; our signature contains a constant symbol 0, relation symbol $<$, and function symbols for multiplication, addition, and successor. In addition, we assume that all of our work is being carried out in the standard model of the natural numbers, denoted $\mathcal{N}$. We, of course, make significant use of common abbreviations like $\leq$, $>$, $(\exists x < y)$, 102, 5, etc. In short, we work within–hopefully!–the reader's intuitive understanding of the natural numbers.

Just as in model theory, a relation $R(\bar{x})$ on the natural numbers is *defined* by a formula $\varphi(\bar{x})$ in the language of first-order arithmetic if and only if the relation contains all and only those tuples of natural numbers which satisfy the formula; that is,

$$R(\bar{n}) \Leftrightarrow \mathcal{N} \Vdash \varphi(\bar{n})$$

Using this identification of formulas with relations, it's natural to generate both the *arithmetical hierarchy of formulas* and the *arithmetical hierarchy of relations*, although only the first level will concern us for the moment. First, we construct the hierarchy for formulas; call a first-order, $\mathcal{L}_A$ formula Σ_0^0 or Π_0^0 if and only if it is first-order equivalent to some $\varphi(\bar{x})$ using only the functions/relations of the signature, bounded quantification, and the standard truth-functional connectives. Base case established, let ψ, φ be formulas of $\mathcal{L}_A$ and define:

$$\psi(\bar{x}) \text{ is } \Sigma_1^0 \Leftrightarrow \psi(\bar{x}) \text{ is first-order equivalent to } \exists \bar{y}[\varphi(\bar{x}, \bar{y})] \text{ where } \varphi(\bar{x}, \bar{y}) \text{ is } \Pi_0^0 \text{ and } \bar{x} \text{ is finite}$$

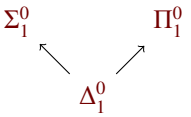
$$\psi(\bar{x}) \text{ is } \Pi_1^0 \Leftrightarrow \psi(\bar{x}) \text{ is first-order equivalent to } \forall \bar{y}[\varphi(\bar{x}, \bar{y})] \text{ where } \varphi(\bar{x}, \bar{y}) \text{ is } \Sigma_0^0 \text{ and } \bar{x} \text{ is finite}$$

This establishes the first level of the arithmetical hierarchy of formulas; now, for its relation counterpart. If a relation on the natural numbers is defined by a Σ_1^0/Π_1^0 formula, we will call the relation itself Σ_1^0/Π_1^0. Next, define:

$$\text{A relation } R \text{ is } \Delta_1^0 \Leftrightarrow R \text{ is defined by both a } \Sigma_1^0 \text{ and a } \Pi_1^0 \text{ formula}$$

Alternatively, a relation R is Δ_1^0 if and only if both R and $\neg R$ are Σ_1^0/Π_1^0. This gives the basis of the arithmetical hierarchy of relations; in picture form, we have:

Formula Hierarchy



Relation Hierarchy

In both cases, the arrows denote set inclusion upwards. Our emphasis will, of course, be on the relation hierarchy; it's important, however, to keep the formula hierarchy in mind since it represents the basis for the former.

Some basic ways of classifying relations–computably verifiable, computably refutable, Σ^0_1, Π^0_1, Δ^0_1–in hand, it's natural to ask how these classifications relate to one another and our previous work; uncovering and clarifying these relationships is our next task.

► **Lemma 12.1** (Σ^0_0/Π^0_0-Definable is Primitive Recursive)**.**
Let R be a relation over $\mathbb{N}$; then, the following are equivalent:

(i) R is primitive recursive

(ii) R is defined by a Σ^0_0/Π^0_0-formula in $\mathcal{N}$

Proof Sketch.
($\Rightarrow$)
By induction; show that all of the starting primitive recursive functions are definable, then that composition and primitive recursion are as well.

($\Leftarrow$)
By induction on the complexity of φ; note, in particular, that all of the operations we need to close under have been shown to be primitive recursive.

► **Theorem 12.2** (Σ^0_1 is Computably Verifiable)**.**
Let R denote a relation on $\mathbb{N}$; then, the following are equivalent:

(i) R is Σ^0_1

(ii) R is computably verifiable

Proof Sketch.
$(i) \Rightarrow (ii)$
Suppose R is Σ^0_1; that is, R is defined by $\exists \bar{y} \varphi(\bar{x}, \bar{y})$ for $\varphi(\bar{x}, \bar{y})$ a Σ^0_0/Π^0_0 formula. By the lemma above, $\varphi(\bar{x}.\bar{y})$ defines a primitive recursive relation Q. Thus, we may explicitly define a verifying function for R by:

$$\chi^+_R(\bar{x}) = \mu y[\chi_Q(\bar{x}, (y)_0, \ldots, (y)_k)\downarrow = 0 \wedge (\forall z \leq y)[\chi_Q(\bar{x}, (z)_0, \ldots, (z)_k)\downarrow]]$$

$(ii) \Rightarrow (i)$
Suppose $R = \mathrm{dom}(\vartheta^k_e)$ for some e, k. Then, by our earlier work, $R(\bar{x})$ if and only if there is an encoded computation C such that $T^k(e, \bar{x}, C)$. Recall, however, that $T^k(e, \bar{x}, C)$ is primitive recursive, and thus, defined by a Σ^0_0/Π^0_0 formula of $\mathcal{L}_A$, $\varphi(y, \bar{x}, w)$. Since e is fixed, it follows immediately from the preceding that R is defined in $\mathcal{N}$ by $\exists w\, \varphi(\underline{e}, \bar{x}, w)$, a Σ^0_1-formula.

Example 12.
Examples of Σ_1^0 Relations:

(i) Every W_e under our enumeration scheme

(ii) Kleene's K-predicate; defined by: $e \in K$ if and only if $\Theta^1(e, e)\downarrow$

(iii) The halting predicate; defined by $\langle i, j\rangle \in H$ if and only if $\Theta^1(i, j)\downarrow$

► **Corollary** (Π_1^0 is Computably Refutable)**.**
Let R denote a relation over $\mathbb{N}$; then, the following are equivalent:

(i) R is Π_1^0

(ii) R is computably refutable

Example 13.
Examples of Π_1^0 Relations

(i) Every $\mathbb{N} - W_e$ under our enumeration scheme

(ii) Kleene's $\bar{K}$-predicate; defined by: $e \in K$ if and only if $\Theta^1(e, e)\uparrow$

(iii) The un-halting predicate; defined by $\langle i, j\rangle \in H$ if and only if $\Theta^1(i, j)\uparrow$

► **Corollary** (Δ_1^0 is Computable)**.**
A relation R over $\mathbb{N}$ is Δ_1^0 if and only if R is computable.

2

The three-fold distinction present at the beginning of this section thus survives the initial characterization of relations via formulas; eventually, this simple picture will collapse in the face of an infinite hierarchy of relations, but–for the moment–we may bask in the ability to sort relations into one of three intuitive classes.

12.1.1 Closure Properties and Connections

With the identification of the Σ_1^0 relations and computably verifiable relations, Π_1^0 relations and computably refutable relations, and finally Δ_1^0 relations and computable relations, it remains to detail both which operations each class is closed under and which–if any–operations carry us from one class to another.

From previous work,

► **Theorem 12.3** (Δ_1^0 Closed Operations)**.**
Δ_1^0 or computable relations are closed under all boolean operations, bounded quantifiers, and substitution o
computable functions.

In a similar vein, it's easy to show

► **Theorem 12.4** (Σ_1^0, Π_1^0 Closed Operations)**.**
Both Σ_1^0 and Π_1^0 are closed under $\wedge, \vee$, bounded quantification, and substitution of partial computabl
addition, Σ_1^0 is closed under existential quantification, while Π_1^0 is closed under universal quantificatio

Proof Sketch.
Leverage either the definition in terms of computable verifiability or definability by a certain class of formula.

Moreover, as might be readily surmised from either the opposing natures of verifiability and refutability or existential and universal quantification,

► **Theorem 12.5** (Σ^0_1 and Π^0_1 Relations are Complements)**.**
For any Σ^0_1 relation R, the relation $\neg R$ is Π^0_1.

Proof Sketch.
Simply leverage definitions

12.2 Computably Enumerable Relations

Before moving forward, there is a last class of relations which naturally distinguish themselves: the *computably enumerable* relations. The insight behind this class is, quite simply, that–in studying computability–relations whose members can be computably listed deserve special attention.

Computably Enumerable
A set/relation $R \subseteq \mathbb{N}$ is called *computably enumerable* (often abbreviated to 'c.e.') if and only if $R = \emptyset$ or $R = range(f)$ for some total computable function f; in other words, there exists a total computable function $f : \mathbb{N} \to R$ which enumerates the members of R. Also known as *recursively enumerable*.

A bit of works shows that, while computable enumerability is a new consideration, the class of computably enumerable relations is one with which we're already quite familiar.

Graph of a Function
For a k-ary partial function $f : A^k \twoheadrightarrow B$, the graph of f is $\{\langle \bar{x}, f(\bar{x})\rangle : \bar{x} \in A \text{ and } f(\bar{x})\downarrow\}$; that is, the $k+1$-ary relation relating elements of $\text{dom}(f)$ to their corresponding range element.

► **Theorem 12.6** (Computable Function ⇔ Computably Enumerable Graph)**.**
A partial function $f : \mathbb{N}^k \twoheadrightarrow \mathbb{N}$ is computable if and only if $\text{graph}(f)$ is computably verifiable/Σ^0_1.

Proof Sketch.
($\Rightarrow$)
Suppose $f = \varphi^k_e$ for some k, e; then, the relation $f(\bar{x}) = y$ has positive characteristic function

$$\mu w[T^k(e, \bar{x}, w) \wedge y = U(T^k(e, \bar{x}, w))] \cdot 0$$

which is μ-recursive, and thus computable.

($\Leftarrow$)
Assume $\text{graph}(f)$ is defined in $\mathcal{N}$ by a Σ^0_1-formula of $\mathcal{L}_A$, say $\exists \bar{w}\varphi(\bar{x}, y, \bar{w})$. Then, $\varphi(\bar{x}, y, \bar{w})$ is a Σ^0_0/Π^0_0-formula, and so defines a primitive recursive relation $Q(\bar{x}, y, \bar{w})$. Thus,

$$f(\bar{x}) = \mu w[\chi_Q(\bar{x}, y, (w)_0, \ldots, (w)_k)\downarrow = 0 \wedge (\forall z \leq w)[\chi_Q(\bar{x}, y, (z)_0, \ldots, (z)_k)\downarrow]] \cdot 0 + y$$

By inspection, f is computable.

► **Theorem 12.7** (Selection Theorem)**.**
If $R(\bar{x}, y)$ is a $k+1$-ary, computably verifiable / Σ_1^0 relation, then there is a computable f such that

(i) $\text{dom}(f) = \{\bar{x} \mid \exists y R(\bar{x}, y)\}$

(ii) For a given $\bar{x}$, if $\exists y R(\bar{x}, y)$, then $R(\bar{x}, f(\bar{x}))$

In other words, the computable f always picks out a witness to $\exists y R(\bar{x}, y)$ for any input $\bar{x}$ and is only defined when such a witness exists.

Proof Sketch.
Simply leverage the previous results and build f

► **Theorem 12.8** (Computable Enumerability is Computable Verifiability)**.**
$R \subseteq \mathbb{N}$ is computably enumerable if and only if R is computably verifiable / Σ_1^0.

Proof Sketch.
($\Rightarrow$)
Construct an existential formula defining R using the Tarski T predicate

($\Leftarrow$)
Define the computable f such that $R = \text{ran}(F)$ by using recursion and the μ operator

We have, then, a plethora of names, but only three distinct classes of relations; since writing 'computably verifiable / Σ_1^0 / computably enumerable' invariably becomes unwieldy, we adopt the convention of referring to our classes of relations by the names Σ_1^0, Π_1^0, and Δ_1^0. The alternative characterizations in terms of verifiability, refutability, computability, and enumerability should not, of course, be forgotten–but, one hopes, constant reminders are not particularly necessary.

► **Proposition 12.1** (Infinite Σ_1^0 Relations are the Ranges of Computable, Bijective Functions)**.**
If $A \subseteq \mathbb{N}$ is infinite and Σ_1^0, then there exists $f : \mathbb{N} \to A$ such that f is both bijective and total computable.

► **Theorem 12.9** (Infinite, C.E., and $<$-Preserving $\Leftrightarrow \Delta_1^0$)**.**
If $A \subseteq \mathbb{N}$ is infinite, then the following are equivalent:

i. $A = \text{ran}(f)$ where $f : \mathbb{N} \to \mathbb{N}$ is total computable and $<$-preserving, $n < m \Rightarrow f(n) < f(m)$

ii. A is a Δ_1^0 relation

■ **Remark.** Note that any finite set is computable; thus, if $f : \mathbb{N} \to \mathbb{N}$ is total computable and $<$-preserving, then $\text{ran}(f)$ is computable.

12.3 Reduction and Computable Separability

Reduction for a class of sets is the property that, for any two sets of the class, each can be reduced to a subset of itself such that the subsets are disjoint and the union of the subsets is unchanged from the union of the original sets; as it turns out, the set of Σ_1^0 relations over $\mathbb{N}$ is one such class:

► **Theorem 12.10** (Reduction for Σ_1^0)**.**
If $A, B \subseteq \mathbb{N}$ are Σ_1^0, then there exists $A' \subseteq A$ and $B' \subseteq B$ both Σ_1^0 such that

(i) $A' \cup B' = A \cup B$

(ii) $A' \cap B' = \emptyset$

Proof.
Let $A = range(f)$, $B = range(g)$. Let $n \in A' \Leftrightarrow \exists k(n = f(k) \wedge \forall j < k(n \neq g(j))$, $n \in B' \Leftrightarrow \exists k(n = g(k) \wedge \forall j \leq k(n \neq f(j))$; that is, we simply look at which function gets to an element first. □

In the context of the arithmetical hierarchy, the natural and immediate companion to the reduction property is the *separation property*; since Σ_1^0 has reduction, we may immediately show that the class of its complements–Π_1^0–has separation:

► **Corollary** (Separation for Π_1^0)**.**
If A, B are disjoint Π_1^0 sets, then there is a Δ_1^0 set C such that $A \subseteq C$ and $C \cap B = \emptyset$; that is, C *separates* A and B (note that which Π_1^0 set is taken as A and which B is of no consequence).

Proof Sketch.
Apply the reduction theorem to $\bar{A}$ and $\bar{B}$

Strangely enough, the primary utility of this section comes not from the positive results already given, but rather the following negative result:

► **Theorem 12.11** (Σ_1^0 does not have Separation)**.**
Σ_1^0 does not have the separation property.

Proof Sketch.
We define two disjoint Σ_1^0 sets, A and B, as follows:

$$e \in A \Leftrightarrow \Theta^1(e, e)\downarrow = 0$$

$$e \in B \Leftrightarrow \Theta^1(e, e)\downarrow = 1$$

Suppose that there is a computable C such that $A \subseteq C$ and $C \cap B = \emptyset$. Then, the following defines a computable function, and so has some index e_0 in our enumeration:

$$\vartheta_{e_0}(x) = \begin{cases} 1 & \text{if } x \in C \\ 0 & \text{if } x \notin C \end{cases}$$

Note that $e_0 \in A \cup B$. If $e_0 \in A$, then $\Theta^1(e_0, e_0)\downarrow = 0$, and so $e_0 \notin A$–meaning $A \not\subseteq C$. Suppose, then, that $e_0 \in B$; then, $\Theta^1(e_0, e_0)\downarrow = 1$, and so $e_0 \in C$–meaning $e_0 \notin B$. It must be, then, that no such C exists; that is, A and B are *inseparable*.

► **Corollary** (Not All Computable Functions can be Extended)**.**
There is a computable function with no total computable extension

Proof Sketch.
Taking A, B from the proof above, define

$$g(x) = \begin{cases} 0 & \text{if } x \in A \\ 1 & \text{if } x \in B \end{cases}$$

If $g \subseteq f$ and f is total computable, then we may use f to define a separating Δ^0_1 C:

$$x \in C \Leftrightarrow f(x)\downarrow = 0$$

■ **Fact.** If a total computable function sends every member of K into a set A and every member of $\bar{K}$ into a set B,

12.4 Many-One Reducibility

All our distinctions vis-a-vis relations have thus far collapsed into the basic three-way distinction between verifiable relations (Σ^0_1; computably enumerable), refutable relations (Π^0_1), and relations which are both verifiable and refutable (Δ^0_1). Nonetheless, these classes cannot be entirely homogeneous since not all Σ^0_1, nor all Π^0_1 relations are Δ^0_1. Is this the only dimension along which relations within each class differ? In other words, are all Σ^0_1, but not Δ^0_1 relations essentially the same? Are all Π^0_1, but not Δ^0_1 relations essentially the same? Answering these questions, of course, requires a notion of what it means for relations to be 'the same'. After presenting an intuitive notion of reduction–for one relation being at least as hard as another–we begin the long process of showing that, surprisingly enough, not all $\Sigma^0_1 - \Delta^0_1$ relations were created equal and, of course, the same for $\Pi^0_1 - \Delta^0_1$ relations.

The first–and most important–of our steps is formalizing precisely what it means for one relation to be at least as hard as another; it's intuitively obvious that such a relation should be both reflexive and transitive. A natural additional criterion is that the process of reduction should itself be computable; indeed, it's a short leap from these basic considerations to our most basic notion of reducibility:

Many-One Reducible
For two relations $A, B \subseteq \mathbb{N}$, A is *many-one reducible* to B–in symbols, $A \leq_m B$–if and only if there is a total computable function f such that for every e,

$$e \in A \Leftrightarrow f(e) \in B$$

i.e., $f^{-1}(B) = A$. Also known as *m-reducible*.

Although implicit in much of work until now, it's worth noting that only considering 1-ary relations doesn't result in a loss of generality–encoding sequences is, after all, a primitive recursive operation. The notion of m-reducibility in hand, it's easy to see that it behaves as intended vis-a-vis computable relations:

► **Proposition 12.2** (Reducible to Computable ⇒ Computable)**.**
If $A \leq_m B$ and B is computable, then A is computable

Proof Sketch.
Simply note that $\chi_A(x) = \chi_B(f(x))$ where f is the function which gives the reduction $A \leq_m B$

Before moving forward, we introduce two additional pieces of terminology; although defined in terms of Σ_1^0, identical definitions hold for the terms Π_1^0-hard and Π_1^0-complete.

Σ_1^0-Hard
A relation $A \subseteq \mathbb{N}$ is Σ_1^0-*hard* if and only if for every Σ_1^0 relation $B \subseteq \mathbb{N}$, $B \leq_m A$; i.e., A is at least as hard to compute as any Σ_1^0 relation.

Σ_1^0-Complete
A relation $A \subseteq \mathbb{N}$ is Σ_1^0-*complete* if and only if A is both Σ_1^0-hard and Σ_1^0.

Our original question, then, asks whether every relation in $\Sigma_1^0 - \Delta_1^0$ / $\Pi_1^0 - \Delta_1^0$ is, in fact, Σ_1^0/Π_1^0-complete.

12.4.1 The m-Degrees

To generalize even further, recall that for two relations R and S, if both $R \leq_m S$ and $S \leq_m R$, R and S are–intuitively–of the same computational difficulty; moreover, this relationship is precisely what interests us. We may readily symbolize such a relationship by dictating the following definition:

$$R \equiv_m S \text{ if and only if both } R \leq_m S \text{ and } S \leq_m R \text{ for unary relations } R, S$$

It follows immediately from our previous work that $\equiv_m$ is an equivalence relation, and so divides relations over $\mathbb{N}$ into equivalence classes of computational difficulty–the *m-degrees*.

m-Degree
The m-degree of a relation A is its equivalence class under $\equiv_m$; that is, $\deg_m(A) = \{B : A \equiv_m B\}$

Our goal of completely characterizing the basic relation classes Δ_1^0, Σ_1^0, and Π_1^0 may then be readily rephrased into understanding the m-degrees of the relations in these classes; in particular, whether or not there are precisely three m-degrees below the Σ_1^0/Π_1^0 level.

12.4.2 Characterizing the Σ_1^0/Π_1^0-Complete Sets

Although we've already encountered several Σ_1^0-complete and Π_1^0-complete relations, the most useful and intuitive of these will prove to be Kleene's K predicate and its complement $\bar{K}$:

► **Proposition 12.3** (K is Σ_1^0-Complete)**.**

► **Corollary** ($\bar{K}$ is Π_1^0-Complete)**.**

Index Sets and Rice's Theorem

Index Set
$I \subseteq \mathbb{N}$ is an *index set* if and only if for every a, b,

$$\vartheta_a = \vartheta_b \Rightarrow (a \in I \Leftrightarrow b \in I)$$

In other words, I is an index set if and only if, for some class F contained in the set of computable functions, $I = \{a \mid \vartheta_a \in F\}$. There are two index sets that, one hopes, leap immediately to mind: $\emptyset$ and $\mathbb{N}$. For obvious reasons, we refer to these two as the *trivial* index sets.

► **Theorem 12.12** (Non-Trivial Index Sets are at least Σ^0_1/Π^0_1-Hard)**.**
For any non-trivial index set $I \subseteq \mathbb{N}$, either $K \leq_m I$ or $K \leq_m \bar{I}$

Proof Sketch.
Suppose I is a non-trivial index set. Note that there exists a computable function f such that $f(x) \uparrow$ for all inputs x; let e_f denote the index of this function. Then, $e_f \in I$ or $e_f \in \bar{I}$; noting the index sets are closed under complementation, assume without loss of generality that $e_f \in \bar{I}$. By assumption, $I \neq \emptyset$, so there must exist some $e_0 \in I$–and, by I an index set, $\vartheta_{e_0} \neq \vartheta_{e_f}$. Consider the following function:

$$g(x,y) = \begin{cases} \vartheta_{e_f}(y) & x \in K \\ \uparrow & \text{otherwise} \end{cases}$$

Note, in particular, that g is computable; it follows by the s^m_n theorem that there exists a computable function $s^1_1(x)$ such that

$$\vartheta_{s^1_1(x)}(y) = g(x,y)$$

Take the reduction function to be this s^1_1

► **Corollary** (Rice's Theorem)**.**
The only computable index sets are the trivial index sets: $\emptyset$ and $\mathbb{N}$.

► **Corollary.**
If I is an index set, $I \neq 0$, $I \neq \mathbb{N}$, and $\emptyset$ is indexed in $\bar{I}$, then I is Σ^0_1-hard.

► **Theorem 12.13** (Rice-Shapiro)**.**
For any n, if $I \subseteq \mathbb{N}$ is an n-ary, Σ^0_1 index set, then

$$e \in I \Leftrightarrow \exists k \in I \text{ such that } \vartheta^n_k \subseteq \vartheta^n_e \text{ and } \vartheta^n_k \text{ is a function with finite domain}$$

.

Proof Sketch.

That is, any Σ^0_1, k-ary index set contains the index of a function just in case it also contains an index for a finite function which extends into it.

Productive and Creative Relations

Productive
A relation $B \subseteq \mathbb{N}$ is *productive* if and only if there is a total computable function f–called the *productive function for* B–such that for every e,

$$W_e \subseteq B \Rightarrow f(e) \in B - W_e$$

Note that if B is productive, then B is not computably enumerable since, for any c.e. set W_e, there is always another element $f(e)$ in B, but not in W_e; in other words, *productive* simply means 'effectively not computably enumerable'

Completely Productive
B is completely productive if and only if there is a total computable f such that

$$f(e) \in B \Leftrightarrow f(e) \notin W(e)$$

► **Theorem 12.14.**
B is productive if and only if B is completely productive.

Proof.
($\rightarrow$)
($\leftarrow$) tricky proof with recursion thm. or (a) $\bar{K}$ is completely productive $f(e) = e \in \bar{K}$ iff $e \notin W_e$ (f witnesses comp. prod.) (b) if $\bar{K} \leq_m B$, then B comp. prod. (as for prod.) □

► **Proposition 12.4** ($\bar{K}$ is productive)**.**
$\bar{K}$ is a productive relation.

Proof Sketch.
Set $f(e) = e$. Then need, if $W_e \subseteq \bar{K}$, $e \in \bar{K}$ and $e \notin W_e$. But, if $e \in W_e$, $e \in \bar{K}$–but then $e \notin W_e$. Thus, $e \notin W_e$, and so $e \in \bar{K}$.

► **Proposition 12.5** (Productive Relations contain Infinite, Σ^0_1 Relations)**.**
If $B \subseteq \mathbb{N}$ is a productive relation, then B has an infinite, Σ^0_1 subset.

Proof Sketch.
Note that, since B is productive, it has a productive function f; note further that $\emptyset$ is a computably enumerable subset of B. Taking $e_\emptyset$ to be its index, $f(e_\emptyset)$ is another element of B–meaning $\{f(e_\emptyset)\}$ is a c.e. subset of B! In general, the pattern above can be computably cycled, giving an infinite, c.e. subset of B.

► **Theorem 12.15** ($\bar{K} \leq_m \Leftrightarrow$ Productive)**.**
The following are equivalent:

(i) A relation $R \subseteq \mathbb{N}$ is productive

(ii) $\bar{K}$ is m-reducible to R

[Myhill 1955]

Proof.
Let f_m be the many-one reduction of B to C and f_B the productive function for B. Define $f_C(x) = f_m(f_B(s_1^1(x)))$ where s_1^1 is defined as follows. Set

$$g(e, x) = \Theta^1(e, f_m(x))$$

Noting that g is computable, by the s_n^m theorem there must exist a function s_1^1 such that

$$\vartheta_{s_1^1(e)}(x) = g(e, x)$$

Noticing that $W_{s_1^1(e)} = f_m^{-1}(W_e)$ makes it easy to show that f_C is, in fact, a productive function for C. □

► **Corollary** (Π_1^0-Hard ⇔ Productive)**.**
Every Π_1^0-complete relation is a productive relation

Creative
A relation $A \subseteq \mathbb{N}$ is *creative* if and only if A is Σ_1^0 and $\bar{A}$ is productive.

■ **Fact.** By definition, no creative relation can be computable; indeed, the name 'creative' derives from this built in non-decidability

■ **Fact.** Not every productive set has a creative complement

Computably Isomorphic
Two relations A and B are *computably isomorphic* if and only if there exists a total computable, bijective function from A to B

Being computably isomorphic means that two sets are isomorphic from the limited perspective of the computable functions; that is, there is no difference between the computational difficulty of the two sets according to any of the measures considered thus far.

► **Theorem 12.16** ($K \leq_m$ ⇔ Creative)**.**
The following are equivalent:

(i) A relation $R \subseteq \mathbb{N}$ is creative

(ii) K is m-reducible to R

(iii) R is computably isomorphic to K

[Myhill 1955]

Proof Sketch.

► **Corollary** (Σ^0_1-Complete ⇔ Creative)**.**
Every Σ^0_1-complete relation is a creative relation; moreover, all of these are computably isomorphic to one another. They are all 'the same' according to our current measures.

Chapter 13

Relative Computation and Turing Reducibility

Having explored the bottom of the arithmetical hierarchy, we now set our sights on the hierarchy as a whole; it's not clear, however, that the rest of the arithmetic hierarchy is *worth* studying. For the lowest level, we had a clear motivation in the notion of computability; the higher levels, though, don't at first seem to admit anything similar. In this section, not only is the remainder of the hierarchy generated, but natural problems are shown to reside at levels beyond Σ^0_1 and Π^0_1; in the next section, we take up the task of giving an intuitive framework for thinking about these higher levels.

13.1 Higher Complexity Relations

As before, call a first-order formula of $\mathcal{L}_A$, the language of first-order arithmetic, Σ^0_0 or Π^0_0 if and only if it is logically equivalent to some φ using only bounded quantification and the standard truth-functional connectives. Inductively,

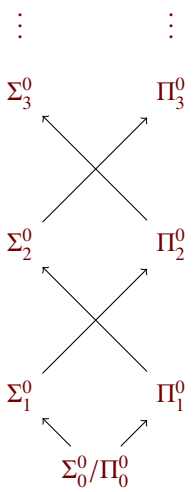
$$\psi \text{ is } \Sigma^0_{n+1} \Leftrightarrow \psi \text{ is logically equivalent to } \exists \bar{x} \varphi \text{ where } \varphi \text{ is } \Pi^0_n \text{ and } \bar{x} \text{ is finite}$$

$$\psi \text{ is } \Pi^0_{n+1} \Leftrightarrow \psi \text{ is logically equivalent to } \forall \bar{x} \varphi \text{ where } \varphi \text{ is } \Sigma^0_n \text{ and } \bar{x} \text{ is finite}$$

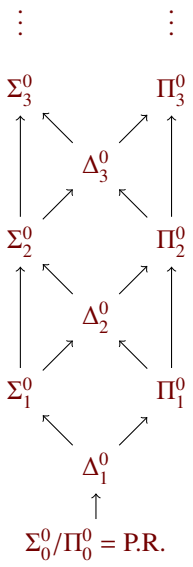
This gives rise to an infinite hierarchy of $\mathcal{L}_A$ formulas; as earlier, we again associate the relations on the natural numbers defined by each of these formulas in $\mathcal{N}$ with the formulas themselves, generating a hierarchy of relations over $\mathbb{N}$. We also define,

$$\text{A relation } R \text{ is } \Delta^0_n \Leftrightarrow R \text{ is defined by both a } \Sigma^0_n \text{ and a } \Pi^0_n \text{ formula}$$

Alternatively, a relation R is Δ^0_n if and only if both R and $\neg R$ are Σ^0_n-relations.



Formula Hierarchy



Relation Hierarchy

13.1.1 Closure Properties & Basic Results

Σ^0_n, Π^0_n, and Δ^0_n relations are closed under $\vee$, $\wedge$, bounded quantification, substitution of total computable functions, and substitution of partial computable functions after the $n = 1$ level. In addition, Δ^0_n relations are closed under negation, Σ^0_n relations are closed under existential generalization, and Π^0_n relations are closed under universal generalization. All of these results follow easily by leveraging the formula-based definition of the hierarchy.

Universal Relation
A $k+1$-ary relation $U(\bar{x})$ is *universal for some class of relations D* if and only if

i. U is itself in D

ii. Whenever a relation R is in D and k-ary, then there exists an $e \in \mathbb{N}$ such that

$$R = U(e, \bar{y}) = \{\bar{y} \ : \ U(e, \bar{y})\}$$

► **Theorem 13.1** (Σ^0_n-Universal Relations Exist)**.**
For any $n \geq 1$ and any k, there is a Σ^0_n-universal $k+1$-ary U.

Proof Sketch.
Proceed by induction on n, noting that the $n = 1$ case has already been established. For the inductive step, suppose the result holds for n and consider $n+1$. Given k, let V be Π^0_n-universal for $k+1$-ary (note that Σ^0_n-universal immediately guarantees a Π^0_n-universal, namely its negation); define:

$$U(e, \bar{x}) \Leftrightarrow \exists y V(e, \bar{x}, y)$$

► **Corollary** (Π^0_n-Universal Relations Exist)**.**
For any $n \geq 1$ and any k, there is a Π^0_n-universal $k+1$-ary U.

► **Theorem 13.2** (Hierarchy Theorem)**.**
If a k-ary relation U is Σ_n^0-universal, then U is not Π_n^0

Proof Sketch.
Suppose that a $k+1$-ary Σ_n^0-universal U is Π_n^0. Consider the k-ary relation R defined by:

$$\mathbb{R}(e, \bar{n}) \Leftrightarrow \neg U(e, e, \bar{n})$$

Since U is Π_n^0, R is both Σ_n^0 and k-ary.

Note that the hierarchy theorem guarantees something we've tactily assumed thus far; the arithemtic hierarchy defined above never collapses at some level. Going higher always introduces new, previously unseen relations.

Hard
A relation $A \subseteq \mathbb{N}$ is D-hard for some class of relations D if and only if for every relation $B \subseteq \mathbb{N}$ in D, $B \leq_m A$

► **Corollary** (Hard Relation ⇔ Hard Complement)**.**
A relation $R \subseteq |N$ is Σ_n^0-hard if and only if $\neg R$ is Π_n^0-hard

13.1.2 Canonical Problems

Just as K and $\bar{K}$ are used as the standard bearers for the Σ_1^0 and Π_1^0 relations generally, there are a host of canonical problems which are widely known and utilized as the de facto problem in their complexity class:

$$\begin{array}{rcl}
K(e) & \iff & \vartheta_e(e)\downarrow \\
\mathrm{Tot}(e) & \iff & W_e = \mathbb{N} \\
\mathrm{Und}(e) & \iff & W_e = \emptyset \\
\mathrm{Inf}(e) & \iff & W_e \text{ is infinite} \\
\mathrm{Fin}(e) & \iff & W_e \text{ is finite} \\
\mathrm{Cof}(e) & \iff & W_e \text{ is cofinite} \\
\mathrm{Onto}(e) & \iff & E_e = \mathbb{N} \\
\mathrm{Rec}(e) & \iff & W_e \text{ is computable} \\
\mathrm{Comp}(e) & \iff & W_e \text{ is Turing complete}
\end{array}$$

Using our work with index sets, it's easy to see that all of these–with the exception of K–are non-computable. Classifying these relations further, however, can become quite involved. Several of these results are presented below:

► **Proposition 13.1** (Tot is Π_2^0-Complete)**.**
Tot is a Π_2^0-complete relation

Proof Sketch.
For hardness, let $B(n)$ be an arbitrary Π_2^0-relation. B is, by definition, defined by $\forall m \exists n R(x, m, n)$ for some primitive recursive R. To reduce B to Tot, consider the total computable f which gives

$$\vartheta_{f(e)}(m) = \mu n R(e, m, n)$$

► **Proposition 13.2** (Fin is Σ^0_2-Complete)**.**
Fin is a Σ^0_2-complete relation

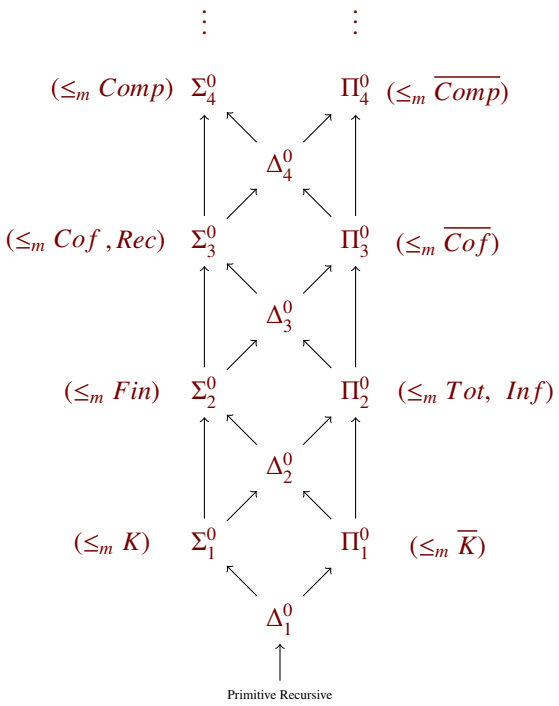
Proof Sketch.
For hardness, let $B(n)$ be an arbitrary Σ^0_2-relation. B is, by definition, defined by $\exists n \forall m R(x, n, m)$ for some primitive recursive R. To reduce B to Fin, consider the total computable f which gives

$$\varphi_{f(e)}(y) = \begin{cases} 0 & (\forall m \leq y)(\exists n)\bar{R}(e, m, n) \\ \uparrow & \text{otherwise} \end{cases}$$

► **Proposition 13.3** (Cof is Σ^0_3-Complete)**.**
Cof is a Σ^0_3-complete relation

Proof Sketch.

In general, then, we have the following picture of the arithmetic hierarchy when all of the canonical problems have been assigned to the appropriate level of the hierarchy:



Limit Computation

13.1.3 Trees and Tree Arguments

Tree
A *tree* on a set X is a set $T \subseteq \cup_{n\in\mathbb{N}} {}^nX$ where $\cup_{n\in\mathbb{N}} {}^nX$ is the set of sequences of values $\langle a_0, \ldots, a_{n-1}\rangle$ where each $a_i \in X$ such that if $t \in T \Rightarrow \forall k(t \upharpoonright k \in T)$

We let $[T] = \{f : \mathbb{N} \to X | \forall k f \upharpoonright k \in T\}$

► **Lemma 13.1** (Konig's Infinity Lemma)**.**
If T is an infinite, finitely branching tree, then $[T] \neq \emptyset$

Finitely Branching
T is finitely branching iff $\forall s \in T$, $\{a | s^\frown\langle a\rangle \in T\}$ is finite.

Proof? ? ? of the defin? above
T: define $s_i \in T, dom(s_i) = i$ such that $\{t \in T | s_i \leq t\}$ is infinite. (Some Stuff)
If $T \subseteq \bigcup_{n\in\mathbb{N}} {}^n\mathbb{N}$ is computable (i.e. $\{\ulcorner s \urcorner | s \in T\}$ is computable) and infinite finitely branching, then

i. $\exists f \in [T]$ such that $graph(f)$ is Δ^0_3

ii. There need not be a Δ^0_2 branch.

► **Theorem 13.3** (Consider $T \subseteq \bigcup_{n\in\mathbb{N}} {}^n\{0,1\}$ Suppose T is computable, finitely branching. T has a Δ^0_2 branch.)**.**
A

---

ll we really need is that we have a recursive function bounding the size of levels?

Proof.
Let f be the leftmost branch, i.e. $f \upharpoonright o = \emptyset$, $f(n)$ =least $i \in \{0,1\}$ such that $f \upharpoonright n \frown \langle i\rangle$ has infinitely many extensions in T. Note that what follows the 'such that' in the previous sentence is equivalent to $\forall k > dom(t) \exists s \in {}^k\{0,1\}[s \in T \wedge t \subseteq S]$–$\Pi^0_1$. Gives that f is Δ^0_2. $f(k) = i$ iff $\exists s(seq(s) \wedge lh(s) = k+1 \wedge \forall j \leq lh(s) \wedge (somethingaboutthepreviousconditiononS \upharpoonright j, (s)) \wedge (s)_k = i)$. □

► **Theorem 13.4.**
There is a recursive tree T on $\{0,1\}$ such that T is infinite and T has no infinite recursive branch.

Proof.
Let A, B be c.e., $A \cap B = \emptyset$ such that there is no recursive C with $A \subseteq C$ and $C \cap B = \emptyset$. T will be the tree of all attempts to build χ_C, for some C such that $A \subseteq C$ and $C \cap B = \emptyset$. Formally, let $h : \mathbb{N} \to A$, $g : \mathbb{N} \to B$ both surjective, recursive, total. Put $s \in T$ iff s is a finite sequence of 0's and 1's and $\forall n \leq lh(s)[h(n) \in dom(s) \Rightarrow s(h(n)) = 0 \wedge g(n) \in dom(s) \Rightarrow s(g(n)) = 1]$. T recursive and check that $[T] = \{\chi_C | A \subseteq C \wedge C \cap B = \emptyset\}$. □

13.2 Relative Computability

Post believed that many-to-one reducibility fails to accurately capture our intuitive notion of reducibility; he proposed several different notions instead, the most important of which is based in Turing's own notion of *relative computability* (Soare 47; Cutland 167). At its core, Post's suggestion was to explain reducibility in terms of *relative difficulty*; the general thrust of this analysis is that a problem X is reducible to a problem Y if and only if having a solution for Y also gives a solution for X. This initial idea is, of course, in desperate need of further expansion, but the intuition behind it should be a natural one.

To phrase this new vision slightly differently–and draw on terminology we eventually adopt–we may imagine having an 'oracle' for a particular problem, call it Y. This oracle, like a rather peculiar character out of Greek mythology, tells us instantly whether or not any item we present to it is a solution to Y; beyond this, however, our oracle is a proverbial black box. We do not question how it operates, why it is so magnificently fast, etcetera; we simply know that it is always correct and always gives its answer instantly. Oracle in hand, the problems that are reducible to Y are simply classified as all of the problems we may solve with the use of our oracle for Y.

Returning to the formalism we've developed thus far, the analog to a problem is a k-ary relation on $\mathbb{N}^k$; an oracle O for a k-ary relation R (the 'problem' above) simply says 'yes' or 'no' for any k-tuple of $\mathbb{N}$ according to whether the tuple is or is not in the relation. This characterization immediately gives rise to a number of possibilities for what, in particular, we should take an oracle to be.

i. The relation R itself

ii. The characteristic function of R, χ_R

iii. An 'oracle tape' added to a Turing machine with χ_R printed on it

iv. A total function $O : \mathbb{N} \to \mathbb{N}$

It's not hard to see that all of these are reducible to one another; in a by now stereotypical move, it's not necessary to consider k-ary relations at all–simply encode it into a 1-ary relation in the standard way; similarly, taking χ_R is a very natural choice, but restricting ourselves to a characteristic function still gives the same power as a total function, so why bother? The oracle tape suggestion is out if only because of our function-based analysis up until now; the choice, then, is between representing oracles as 1-ary relations–known, more colloquially, as sets–or total functions from $\mathbb{N}$ to $\mathbb{N}$. We take the latter tact, but build in aspects of the others:

> **Oracle**
> An *oracle* O is a any total function $O : \mathbb{N} \to \mathbb{N}$

In starting our study of computability, we selected a set of functions which were intuitively computable, and then proceeded; now, we take those same functions and simply add one more: the oracle function, O. Just as we constructed a notion of computable, this immediately gives rise to, for any oracle O, a notion of O-computability.

> **O-Computable**
> A partial k-ary function $f : \mathbb{N}^k \twoheadrightarrow \mathbb{N}$ is *computable in an oracle* O or *O-computable* if and only if there is a derivation of f using the primitive recursive functions, O, and the operations of primitive recursion, composition, and μ-recursion

Just as earlier, we must invoke the following with only suggestive evidence:

► **Thesis 13.1** (Relativized Church-Turing Thesis)**.**
A function f is algorithmically computable relative to an oracle O if and only if f is O-computable

---

13.2.1 The Relativized Theorems

The major theme of this section is that the most important results established for computability generally carry over wholesale to the notion of O-computability; in most cases the proofs of these theorems are almost verbatim those of the previous result–note, in particular, that all are independent of the particular oracle in use.

► **Theorem 13.5** (The Relativized Enumeration Theorem)**.**
There exists an $k+1$-ary, O-computable function $\Theta^{O,k}$ with the property that for any oracle O and any O-computable, k-ary f, there is an $e \in \mathbb{N}$ for which

$$\Theta^{O,k}(e, \vec{x}) \simeq f(\vec{x})$$

► **Theorem 13.6** (The Relativized s_n^m Theorem)**.**
For every $m, n \geq 1$ there exists an injective, computable, $m+1$-ary function s_n^m such that for all oracles O, and any $e, a_1, \ldots, a_m \in \mathbb{N}$,

$$\Theta^{O,m+n}(e, \vec{a}, \vec{x}) \simeq \vartheta^{O,n}_{s_n^m(e,\vec{a})}(\vec{x})$$

► **Theorem 13.7** (Kleene's Relativized Fixed-Point Theorem)**.**
For any oracle O, if $f(x, y)$ is an O-computable function, then there is a 1-ary, computable function n such that for any y and any k,

$$\vartheta^{O,k}_{n(y)} \simeq \vartheta^{O,k}_{f(n(y),y)}$$

Moreover, the function n does not depend on the oracle O used.

■ **Notation.** $\vartheta_e^\sigma(x)\!\downarrow\!< s$, sometimes written à la Soare as $\{e\}_s^\sigma(x)\downarrow$, indicates that $\vartheta_e^\sigma(x)\downarrow$ without recourse to any number greater than or equal to s; thus, $s > 0$, $e, \sigma, x < s$, and no number used in the computation is greater than or equal to s

► **Theorem 13.8** (Use Principle)**.**
For any oracle O and any $e \in \mathbb{N}$,

(i) If $\vartheta_e^O(x) = y$, then $(\exists s)(\exists \sigma \subset O)[\vartheta_e^\sigma(x) = y \wedge \vartheta_e^\sigma(x)\!\downarrow\!< s]$

(ii) For any $s \in \mathbb{N}$ and sequence σ, if $\vartheta_e^\sigma(x) = y$ and $\vartheta_e^\sigma(x)\!\downarrow\!< s$, then $(\forall t \geq s)(\forall \tau \supseteq \sigma)[\vartheta_e^\tau(x) = y \wedge \vartheta_e^\tau(x)\!\downarrow\!< t]$

(iii) For any sequence σ, if $\vartheta_e^\sigma(x) = y$, then $(\forall O \supseteq \sigma)[\vartheta_e^O(x) = y]$

Proof Sketch.
Although quite strong in the abstract, each of the above follows easily from how we've defined oracle computation

13.2.2 The Relative Arithmetical Hierarchy

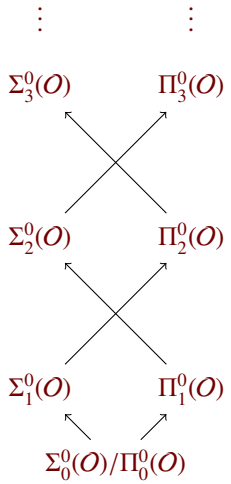
Mimicking our development of computability, we may, for any oracle O, identify a relativized arithmetical hierarchy. Paralleling our earlier work, we work in the language of first-order arithmetic expanded with our oracle relation O, notated $\mathcal{L}_A(O)$. The hierarchy of formulas is again inductively defined by:

$$\begin{aligned}
\text{A formula } \psi \text{ of } \mathcal{L}_A(O) \text{ is } \Sigma^0_0(O)/\Pi^0_0(O) &\Leftrightarrow \psi \text{ is logically equivalent to a formula using only the boolean} \\
&\qquad \text{connectives and bounded quantification} \\
\text{A formula } \psi \text{ of } \mathcal{L}_A(O) \text{ is } \Sigma^0_{n+1}(O) &\Leftrightarrow \psi \text{ is logically equivalent to } \exists \bar{x} \varphi \text{ where } \varphi \text{ is } \Pi^0_n(O) \text{ and } \bar{x} \text{ is finite} \\
\text{A formula } \psi \text{ of } \mathcal{L}_A(O) \text{ is } \Pi^0_{n+1}(O) &\Leftrightarrow \psi \text{ is logically equivalent to } \forall \bar{x} \varphi \text{ where } \varphi \text{ is } \Sigma^0_n(O) \text{ and } \bar{x} \text{ is finite}
\end{aligned}$$

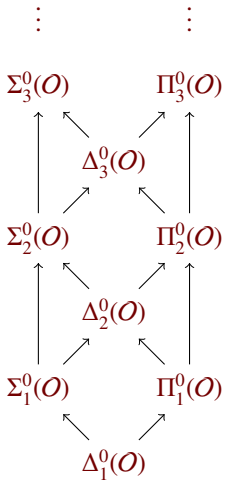
As before, we associate the relation defined by a given formula in the standard model $\mathcal{N}$ with that formula's position in the hierarchy, defining

$$\text{A relation } R \text{ is } \Delta^0_n(O) \Leftrightarrow R \text{ is defined by both a } \Sigma^0_n(O) \text{ and a } \Pi^0_n(O) \text{ formula}$$

This is, of course, equivalent to: a relation R is $\Delta^0_n(O)$ if and only if both R and $\neg R$ are $\Sigma^0_n(O)$-relations. We thus have:



Formula Hierarchy



Relation Hierarchy

The closure rules associated with the arithmetical hierarchy carry over verbatim to the relativized arithmetical hierarchy:

i. The class of $\Sigma^0_1(O)$ predicates is closed under $\wedge$, $\vee$, substitution of total O-computable functions, substitution of O-computable functions past the $n = 1$ level, and existential generalization

ii. The class of $\Pi^0_1(O)$ predicates is closed under $\wedge$, $\vee$, substitution of total O-computable functions, substitution of O-computable functions past the $n = 1$ level, and universal generalization

iii. The class of $\Delta^0_1(O)$ predicates are closed under $\wedge$, $\vee$, $\neg$, substitution of total O-computable functions, and substitution of O-computable functions past the $n = 1$ level.

13.3 Turing Reducibility

Having refined the notion of relative computability, it's time to return to the impetus for its development.

> **Turing Reducible**
> Given two unary relations $A, B \subseteq \mathbb{N}$, *$A$ is Turing reducible to B*, notated $A \leq_T B$, if and only if χ_A is χ_B-computable.

That is, $A \leq_T B$ just in case A is decidable given an oracle for B; the following are easy to establish:

► **Proposition 13.4** (Characteristics of $\leq_T$)**.**
The following all hold for Turing reducibility:

(i) $\leq_T$ is a preorder

(ii) $\leq_m \subset \leq_T$

(iii) For any relation A, $A \leq_T \bar{A}$

(iv) For any computable relation B, $B \leq_T A$ for any relation A

(v) O-Computability is preserved downward under $\leq_T$

(vi) The Σ/Π distinction is not preserved downward under $\leq_T$

Turing Degree
The equivalence classes generated by the relation of mutual Turing reducibility; that is, for any relation $A \subseteq \mathbb{N}$,

$$[A]_T = \{B \subseteq \mathbb{N} \,:\, B \leq_T A \text{ and } A \leq_T B\}$$

Two relations with the same Turing degree are, for obvious reasons, said to be *Turing equivalent*, often notated by $\equiv_T$.

Conventionally, Turing degrees are written in boldface, e.g. **a**, **b**, and **c**, while the class of all Turing degrees is denoted **D**. The following should be reminiscent of the terminology used with many-one reducibility: for a class of relations Γ,

a is a Γ-*degree*	$\iff$	there is a relation of Turing degree **a** in Γ
a is Γ-*hard*	$\iff$	every relation in Γ has a Turing degree $\leq_T$ **a**
a is Γ-*complete*	$\iff$	**a** is both a Γ-degree and Γ-hard

► **Corollary** (Characteristics of Turing Degrees)**.**
The following all hold for Turing degrees:

(i) There is a unique computable degree, denoted **0**

(ii) For a relation $A \subseteq \mathbb{N}$, $[A]_m \subseteq [A]_T$

(iii) $[K]_T$ is Σ^0_1-complete

(iv) The Turing degrees **D** form a partially ordered set
(**a**<**b** just in case a relation in **a** $\leq_T$ a relation in **b**)

13.3.1 The Turing Jump Operator

Denoted by the addition of an apostrophe, the *Turing jump* or simply *jump* A' (read 'A-jump' or 'A-prime') of a relation A is Kleene's K relation relative to an oracle for A, often abbreviated K^A. In other words,

$$A' = K^A = \{x \,:\, \Theta^{A,1}(x,x)\downarrow\} = \{x \,:\, x \in W^A_x\}$$

As above, there is a unique Turing degree for the computable relations, $\mathbf{0}$, and so the unrelativized version of K is $\emptyset'$ This notation can clearly be extended and often is:

$$A^{(n+1)} = (A^{(n)})'$$

► **Proposition 13.5** ($K^A \equiv_T K_0^A \equiv_T K_1^A$)**.**
For any $A \subseteq \mathbb{N}$, K^A is Turing equivalent to each of the following:

(i) $K_0^A = \{\langle x, y\rangle : \Theta^{A,1}(x, y)\downarrow\}$

(ii) $K_1^A = \{x : W_x^{A,1} \neq \emptyset\}$

Proof Sketch.
Use the relativized s_n^m theorem and Myhill's isomorphism theorem

---

► **Theorem 13.9** (Characteristics of the Turing Jump)**.**
For any relations A, B, and C on $\mathbb{N}$,

(i) A' is $\Sigma_1^0(A)$

(ii) $A' \not\leq_T A$

(iii) B is $\Sigma_1^0(A)$ if and only if $B \leq_m A'$

(iv) If A is $\Sigma_1^0(B)$ and $B \leq_T C$, then A is $\Sigma_1^0(C)$

(v) $B \leq_T A$ if and only if $B' \leq_m A'$

(vi) If $B \equiv_T A$, then $B' \equiv_m A'$

(vii) A is $\Sigma_1^0(B)$ if and only if A is $\Sigma_1^0(\bar{B})$

Proof Sketch.
pg. 53 Soare

► **Corollary** (Post Hierarchy Theorem)**.**
For every $n \geq 1$,

i. $\emptyset^{(n)}$ is a Σ_n^0-complete set.

ii. A relation $A \subseteq \mathbb{N}$ is Δ_{n+1}^0 if and only if $A \leq_T \emptyset^{(n)}$

Proof Sketch.

Using the Turing jump operator, we define a parallel operation on Turing degrees:

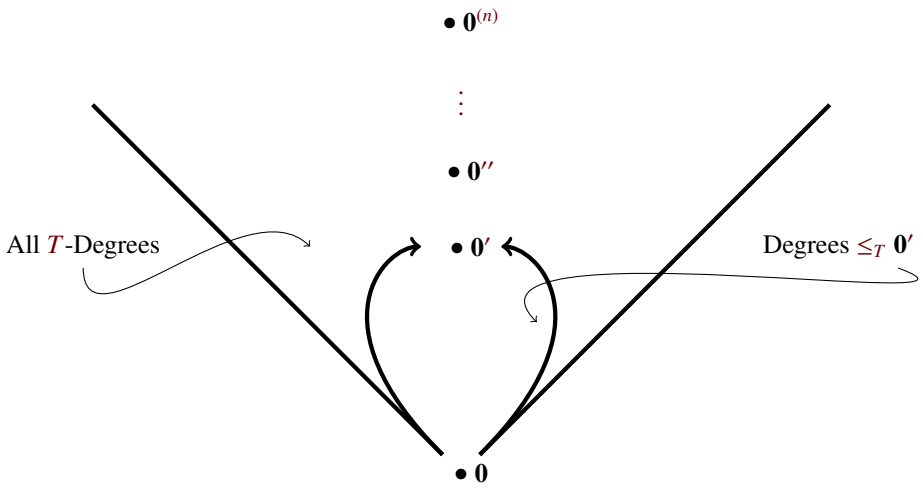
$$\mathbf{a}' = \deg_T(A') \text{ for any } A \in \mathbf{a}$$

Using our results above, the jump is well-defined on degrees, and–by definition–$\mathbf{0}$ is the minimal degree. We thus have an infinite hierarchy of degrees:

$$\mathbf{0} < \mathbf{0}' < \mathbf{0}'' < \mathbf{0}''' < \cdots < \mathbf{0}^{(n)} < \ldots$$

In fact, this hierarchy is closely related to another, as the first few levels make clear:

$$\begin{aligned}
\mathbf{0} &= \deg(\emptyset) = \{A \,:\, A \text{ is computable}\} \\
\mathbf{0}' &= \deg(\emptyset') = \deg_T(K) = \deg_T(\bar{K}) \\
\mathbf{0}'' &= \deg(\emptyset'') = \deg_T(\text{Fin}) = \deg_T(\text{Tot}) \\
\mathbf{0}''' &= \deg(\emptyset''') = \deg_T(\text{Cof}) = \deg_T(\text{Rec}) \\
\mathbf{0}'''' &= \deg(\emptyset'''') = \deg_T(\text{Comp}) = \deg_T(\overline{\text{Comp}})
\end{aligned}$$



$\aleph_0$-Jump

$$A^{(\aleph_0)} = \{\langle n, k\rangle | k \in A^{(n)}\}$$

This is the *computable join* of all $A^{(n)}$ for $n \in \mathbb{N}$.

■ **Fact.** $\emptyset^{(\aleph_0)} \equiv_m \text{Th}^{\mathcal{N}}$

Chapter 14

Post's Problem & Oracle Constructions

Much of our work thus far has been foundational, establishing the basic results and basic notions of computability theory. In this chapter, we leverage the results and notions of the last few chapters in order to prove deep results about the structure of both the Turing and m-degrees.

14.0.2 Simple Relations

Immune
$B \subseteq \mathbb{N}$ is *immune* if and only if both B is infinite and B has no infinite Σ^0_1-subset.

■ **Fact.** There are no Σ^0_1 immune sets.

► **Proposition 14.1** (Π^0_1, Immune Sets Exist)**.**
There is a Π^0_1 set that is immune

Proof Sketch.
Priority?

Simple
A set $A \subseteq \mathbb{N}$ is simple if and only if A is Σ^0_1 and $\bar{A}$ is immune.

► **Theorem 14.1** (Simple Sets $\not\leq_T \emptyset$)**.**
For any simple set $A \subseteq \mathbb{N}$, $A \not\leq_T \emptyset$

► **Theorem 14.2** (Simple Sets $\not\leq_T K$)**.**
For any simple set $A \subseteq \mathbb{N}$, $A \not\leq_T K$

By the theorem above, we have a simple set A.

► **Theorem 14.3** (Post)**.**
There is a simple set.

Proof.
Let f be total recursive, 1-1, and $ran(f)$ is not recursive. Put $D(n) \Leftrightarrow \exists k > n(f(k) < f(n))$ (deficiency stages in the enumeration of $ran(f)$). Claim: D is simple Proof: D is Σ^0_1; $\bar{D}$ is infinite: if $\bar{D}$ has one element, it has infinitely many since there must always be a next smallest and, by definition, this is also in $\bar{D}$. Note further that there must always be at least one element in $\bar{D}$ since there must be a smallest element (which, by definition, is in $\bar{D}$).
$\bar{D}$ is Immune: if not, let $B \subseteq \bar{D}$ be infinite r.e. Then, $m \notin ran(f)$ iff $\exists n(n \in B \wedge m < f(n) \wedge \forall k < n(f(k) \neq m))$. Note that the right side is Σ^0_1. □

Recall that f_k is the finite function coded by k as a table. $F_k = ran(f_k)$.

Hyper-Immune
A set A is *hyper-immune* iff

i. A is infinite

ii. Whenever g is total recursive $\mathbb{N} \to \mathbb{N}$, and $F_{g(m)} \cap F_{g(n)} = \emptyset$ for all $m \neq n$, then $\exists n\ F_{g(n)} \cap A = \emptyset$.

Hyper-Simple
A set A is *hyper-simple* if and only if A is Σ^0_1 and $\bar{A}$ is hyper-immune.

14.1 Incomparable Turing Degrees

Incomparable
Two degrees a, b are *incomparable*, notated $a|b$, if and only if $a \not\leq b$ and $b \not\leq a$

► **Proposition 14.2.**
If a function f is total, extends every partial function in some A-computable sequence, and is completely determined by the sequence, then f is A-computable.

Proof Sketch.
To determine any value, simply search the sequence until a partial function which defines it is found (such a partial function must exist).

► **Theorem 14.4** (Incomparable Turing Degrees below $\mathbf{0}'$)**.**
There are Turing degrees $\mathbf{a}$,$\mathbf{b} \leq_T \mathbf{0}'$ such that $\mathbf{a}$ and $\mathbf{b}$ are incomparable [Kleene-Post].

Proof Sketch.
Our proof method is the first in a set of standard proof techniques in computability theory: a *finite extension, oracle construction*. Put simply, we build step-by-step the characteristic functions for two relations A and B, intending to let $\mathbf{a}$,$\mathbf{b}$ be the Turing degrees of A, B respectively. Crucially, however, we build χ_A and χ_B in both a $\mathbf{0}'$-computable manner (guaranteeing $\mathbf{a}$,$\mathbf{b} \leq_T \mathbf{0}'$) and so that

(i) No B-computable function computes χ_A

(ii) No A-computable function computes χ_B

which, of course, together guarantee incomparability. To guarantee (i) and (ii), we identify a pair of constraints for every $e \in \mathbb{N}$:

$$(S_e)\ \chi_A \neq \vartheta_e^B$$

$$(R_e)\ \chi_B \neq \vartheta_e^A$$

We need simply show that A, B can, in fact, be constructed in a $\mathbf{0}'$ manner to meet both R_e and S_e for every $e \in \mathbb{N}$. To start, set $A = B = \emptyset$. At any odd stage $2e+1$, requirement S_e is satisfied. Taking $\chi_A^{2e+1}, \chi_B^{2e+1}$ to be our initial approximations of χ_A, χ_B (encoded as sequences), we query our $\emptyset'$-oracle for the truth or falsity of the following Σ_1^0, $\mathcal{L}_A$ sentence:

$$\exists \sigma \exists t[BinSeq(\sigma) \wedge \chi_B^{2e+1} \subset \sigma \wedge \vartheta_e^{\sigma}(lh(\chi_A^{2e+1}))\downarrow < t]$$

where $BinSeq$ is true if and only if the input is a number encoding a sequence of 0's and 1's. Put less formally, we ask whether there is an acceptable extension of χ_B^{2e+1} such that

Case 1: If the sentence is true, there must be a least pair $\langle \sigma', t' \rangle$ witnessing this; set $\chi_B^{2e+1+1} = \sigma'$ and define χ_A^{2e+1+1} as χ_A^{2e+1}, but with the addition of

$$\chi_A^{2e+1+1}(lh(\chi_A^{2e+1})) = \begin{cases} 0 & \text{if } \vartheta_{t'}^{\sigma'}(lh(\chi_A^s)) > 0 \\ 1 & \text{otherwise} \end{cases}$$

Case 2: If the sentence is false, define χ_A^{2e+1+1} as χ_A^{2e+1} and χ_B^{2e+1+1} as χ_B^{2e+1}, but with the addition of

$$\chi_A^{2e+1+1}(lh(\chi_A^{2e+1})) = 0$$

$$\chi_B^{2e+1+1}(lh(\chi_B^{2e+1})) = 0$$

Reversing the treatment of χ_A and χ_B in the even case gives a complete procedure of the type required.

► **Theorem 14.5** (Incomparable Turing Degrees)**.**
For any degree $\mathbf{c}$, there are incomparable Turing degrees $\mathbf{a}$,$\mathbf{b}$ such that $\mathbf{c} \leq_T \mathbf{a}$, $\mathbf{b}$ and $\mathbf{a}$,$\mathbf{b} \leq_T \mathbf{c}'$.

Proof Sketch.
Generalize the proof above

► **Theorem 14.6** (An Incomparable Degree Always Exists)**.**
For any degree $\mathbf{b} >_T \mathbf{0}$, there exists an incomparable degree $\mathbf{a} <_T \mathbf{b}'$ for $\mathbf{b}$.

Proof Sketch.
Another finite extension, oracle construction; the two requirements necessary correspond to avoiding the lower cone of degrees with reduce to **b** and the upper cone of degrees which reduce **b**.

■ **Remark.** If $A \equiv_T B$, then $A' \equiv_m B'$. So we can define $[A]'_T = [A']_T$

$$c \leq_T d \Rightarrow c' \leq_T d'$$

So, $0' \leq_T d'$ for all d.

14.2 Friedburg Completeness

Since **0** is the unique smallest Turing degree, $\mathbf{0}'$ is the unique smallest jump-degree. The next theorem establishes that not only do jumps take us to progressively higher Turing degrees, but they also exhaust all possible Turing degrees above $\mathbf{0}'$:

► **Theorem 14.7** (Friedburg Completeness Theorem)**.**
For every Turing degree **b** such that $\mathbf{0}' \leq_T \mathbf{b}$, there is a Turing degree **a** such that $\mathbf{a}' = \mathbf{b}$

Proof Sketch.
Finite extension, oracle construction; we need to construct an A meeting two general constraints: $A' \leq_T B$ and $B \leq_T A'$. In a now standard move, we'll ensure the first on odd steps and the second on even. As we'll see, the latter of these is quite easy, while the former requires more thought. Turning to this problem first, we wish to provide a means for B to decide A'; since B is arbitrary, this information must somehow be built into A–which, of course, is only useful if B can compute A. Our strategy, then, is to make our entire construction of A B-computable, then to somehow encode whether or not $e \in A'$ into the construction.

Moving to the details of this procedure, suppose we are at an odd step $2e+1$ with an initial approximation χ_A^{2e+1} of χ_A. We need a B-decidable question which will somehow indicate whether or not $e \in A'$ which–since all we know of B is $\emptyset' \leq_T B$–this means all we may ask is a strictly existential question. As usual, we invoke the use properties of relative computation, querying:

$$(\exists \sigma \supset \chi_A^{2e+1})(\exists t)[\vartheta_e^{\sigma}(e)\downarrow < t]$$

If true, we set $\chi_A^{2e+2} = \sigma$; if not, we set $\chi_A^{2e+2} = \chi_A^{2e+1}$. Thus, for B to decide whether or not $e \in A'$, it need simply run the construction of A until the $2e+1$-th step, then check the formula above for truth or falsity (note that, if false, $e \notin A'$).

After all the excitement above, building A (B-computably) so that $B \leq_T A'$ is as easy as, for any even step $2e+2$, always extending χ_A^{2e+2} by $\chi_B(e)$; that is, set

$$\chi_A^{2e+3}(x) = \begin{cases} \chi_A^{2e+2} & \text{if } x < lh(\chi_A^{2e+2}) \\ B(e) & \text{if } x = lh(\chi_A^{2e+2}) \end{cases}$$

► **Theorem 14.8** (Relative Friedburg Completeness)**.**
For every Turing degree **c** and any Turing degree $\mathbf{b} \geq_T \mathbf{c}'$, there is a Turing degree **a** such that $\mathbf{c} \leq_T \mathbf{a}$ and $\mathbf{a}' = \mathbf{b}$

Proof Sketch.
Mimic the proof of the original, replacing **0** with **c**

► **Corollary** (Generalized Friedberg Completeness)**.**
For every $n \geq 1$ and every degree $\mathbf{c}$, any Turing degree $\mathbf{b} \geq_T \mathbf{c}^{(n)}$ is such that there is a Turing degree $\mathbf{a}$ where $\mathbf{a}^{(n)} = \mathbf{b}$.

Proof Sketch.
Induction on m

14.2.1 Minimal Degrees

Minimal
A Turing degree $\mathbf{a}$ is *minimal* if and only if $\mathbf{0} <_T \mathbf{a}$ and there is no Turing degree $\mathbf{b}$ such that $\mathbf{0} <_T \mathbf{b} <_T \mathbf{a}$

► **Theorem 14.9** (Spector 1957?)**.**
There is a noncomputable A such that for every B, $B <_T A (B \equiv_T \emptyset$ or $B \equiv_T A)$

Proof.
Construct χ_A by approximating χ_A with computable perfect ? or $\{0,1\}$ (i.e. T is a binary tree).

$$s \in T \Rightarrow \forall k s \upharpoonright k \in T$$

$$T \text{ is perfect} : \forall s \in T \exists u, v \in T[s \subseteq u \wedge s \subseteq v \wedge u \perp v]$$

where $r \perp s$ iff $\exists n \in dom(r) \cap dom(s)[r(n) \neq s(n)]$.

$$T \text{is computable}$$

Notation:
For T a tree, $s \in T$,

$$T_s = \{u \in T | u \not\perp s\}$$

Requirements on A: R_e: Either $\varphi_e^{\chi_A}$ is not total or $\varphi_e^{\chi_A}$ is computable or $\varphi_e^{\chi_A} \equiv_T A$.
Set $T_0 = \bigcup_{n \in \mathbb{N}} {}^n\{0,1\}$. Given T_e: want to ensure R_e.
Case 1: $\exists s \in T_e$ such that $\forall u, v \supseteq s$ with $u, v \in T$ $\forall k$ such that $\varphi_e^u(k)\downarrow$ and $\varphi_e^v(k)\downarrow$, then $\varphi_e^u(k) = \varphi_e^v(k)$. in that case, set $T_{e+1} = (T_e)_s$ for some such s. R_e is now guaranteed: suppose $\chi_A \in [T_{e+1}]$ and $\varphi_e^{\chi_A}$ is total. Then, $\varphi_e^{\chi_A}(k) = i$ iff $\exists u \in T_{e+1} \varphi_e^u(k) = i$ (note that this is Σ_1^0)
Case 2: Otherwise. We'll define a computable, perfect $T_{e+1} \subseteq T$ as follows: For $r \in 2^{<\omega}$, define $\pi(r) =\in T_e$ by induction of the length of r. Then, take $\{u | \exists r[u \subseteq \pi(r)]\}$. Set $\pi(\emptyset) = \emptyset$. Given $\pi(r)$, we need to define

$$\pi(r \frown \langle 0 \rangle)$$

$$\pi(r \frown \langle 1 \rangle)$$

Search for u, v, k showing Case 1 fails for $s = \pi(r)$. For the first ones you find, set $u = \pi(r \frown \langle 0 \rangle)$, $v = \pi(r \frown \langle 1 \rangle)$. Thus completes the definition of π. Note that π is computable. Thus, set

$$u \in T_{e+1} \Leftrightarrow \exists r(u \subseteq \pi(r))$$

$$\Leftrightarrow \exists r(r \in 2^{\text{dom}(u)+1} \wedge u \subseteq \pi(r))$$

Claim: If $\chi_A \in [T_{e+1}]$ and we're in case 2, then $A \leq_T \varphi_e^{\chi_A}$. To recover χ_A from $\varphi_e^{\chi_A}$. Suppose we know that $\pi(r) \subseteq \chi_A$. Q is $\pi(r \frown \langle 0 \rangle) \subseteq \chi_A$ or $\pi(r \frown \langle 1 \rangle) \subseteq \chi_A$. These two differ on φ_e, and so search for k such that for $u = \pi(r \frown \langle 0 \rangle)$, $v = \pi(r \frown \langle 1 \rangle)$, $\varphi_e^u(k)\downarrow, \varphi_e^v(k)\downarrow$, and $\varphi_e^u(k) \neq \varphi_e^v(k)$. Then, $\pi(r \frown \langle 0 \rangle) \subseteq \chi_A$ if and only if $\varphi_e^{\pi(r \frown \langle 0 \rangle)}(k) = \varphi_e^{\chi_A}(k)$ so $\varphi_e^{\chi_A}$ can tell us the answer to our question.
To finish, there is an A_s such that $\chi_A \in [T_e]$ for all e (if T_i is a subtree of $(2^{<\omega}, \subseteq)$ such that for every finite F such that

$$\cap_{i \in F}[T_i] \neq \emptyset$$

then,

$$\cap[T_i] \neq \emptyset$$

□

14.2.2 Constructing c.e. sets

Constructing c.e. with a prescribed $\leq_T$.

► **Theorem 14.10.**
There is a simple set A such that $A' \equiv_T 0'$

$0' = A'$ A $\emptyset'$

Proof.
We'll define finite sets A^s, $s \in \mathbb{N}$, $A^s \subseteq A^{s+1}$, $S \vdash' A'$ computable, $A = \bigcup_s A^s$.
Requirements: P_e: ensure $W_e \cap A \neq \emptyset$ (if W_e infinite) N_e: if you see $\varphi_e^{\chi_{A^s} \upharpoonright k}(e)\downarrow$, then preserve $\chi_{A^s} \upharpoonright k$ (i.e. restrain all $i < k$ such that if $i \notin A^s$, then i not in A).
To resolve conflicts: if P_e and N_l want to act in conflict at some stage s, give priority to lower index, breaking equality in favor of P. If an N is violated, throw it out since it doesn't do anything anymore. Ns can be issued repeatedly for a given e, each P_e acts at most once (achieving its goal). Each N_e acts finitely often ($\leq l + 1$).
Now, W_e inf $\Rightarrow W_e \cap A \neq \emptyset$. Take every $i \in W_e$ such that no N_l, $l < e$ ever restrains i. $i \in A$ unless some other $i' \in W_e \cap A \in T$. To see $\chi_{A'} \leq_T \emptyset'$, to decide whether $l \in A'$, $\varphi_e^{\chi_A}(l)\downarrow$ using $\emptyset'$ as oracle: go to a stage s such that if $e \leq l$ is such that P_e ever acts, it has acted already (this needs to ask 'did P_e ever act?'–a Σ_1^0 question). Then, $\varphi_l^{\chi^A}(l)\downarrow$ iff N_l has a restraint in force at some stage greater than s (again, a Σ_1^0 question) □

Get A, B r.e. such that $A \not\leq_T B$ and $B \not\leq_T A$
The above is called a finite injury priority argument.

Part III

Computability and First-Order Logic

Chapter 15

Languages, Theories, and Computability

Historically, logical interest has centered around a handful of first-order theories; in this chapter we present the basic terminology for working with both first-order logic and theories from a computability perspective, then canvas those theories which have proven most interesting, proving some basic results about each.

15.1 Languages, Theories, and Axioms

First, we bridge the gap between first-order languages–finite strings of symbols over varying signatures–on one hand and our theory of computation–functions from $\mathbb{N}$ to $\mathbb{N}$–on the other; to do so, we simply follow Gödel and generate an encoding scheme for the symbols which occur in any language of interest. Given all of our previous work, doing so is as simple as assigning each distinct symbol its own number via a computable function #; then, strings like '$\forall xR(x)$' can easily be encoded as a sequence of these numbers:

$$\langle \#(\text{`}\forall\text{'}), \#(\text{`}x\text{'}), \#(\text{`}R\text{'}), \#(\text{`(`}), \#(\text{`}x\text{'}), \#(\text{`)'}) \rangle$$

This sequence is, of course, itself representable as a single number in the familiar way. To save on space, we wrap this entire process into the placement of corner quotes around a well-formed formula of the language in question, as below:

$$\ulcorner \forall xR(x) \urcorner$$

Thus, $\forall xR(x)$ is a well-formed formula from some language of interest and $\ulcorner \forall xR(x) \urcorner$ is the natural number which encodes it according to some background encoding scheme. The specific encoding scheme used will be left unspecified; we merely suppose that it is a reasonable one which has the basic properties of traditionally popular schemes. The most important of these properties is that the language under consideration is *computable* (also, *recursively-presented*); that is, there exists a computable function which, given a natural number, decides whether or not that number encodes a well-formed formula of the language. As far as naturally occurring languages and theories, this is remarkably weak, but should–nonetheless–be kept in mind.

Deductively-Closed Theory
A *deductively-closed theory* in a language $\mathcal{L}$ is a set T of $\mathcal{L}$-sentences such that for all sentences φ of $\mathcal{L}$, if $T \models \varphi$, then $\varphi \in T$.

Axioms
A set of $\mathcal{L}$-sentences Γ is *a set of axioms for a deductively-closed $\mathcal{L}$-theory T* if and only if T is the deductive closure of Γ under some specific consequence relation

Computably Axiomatizable
A deductively-closed theory T in a computable language $\mathcal{L}$ is *axiomatizable* if and only if there exists a computable set of axioms for T; that is, there is a computable function which decides whether or not any natural number encodes an axiom

Historically, one of the largest areas of interaction between computability theory and logic is in proving or disporving the decidability of some particular, usually mathematical, theory.

15.2 Decidability and Undecidability

A theory is decidable if and only if, using our encoding of formulas into natural numbers, the relation defined by the theory is decidable; that is, determining whether or not any natural number encodes a formula which is in the theory is a computable process. A word of warning is necessary here; many authors equivocate between theories and deductively-closed theories or between the decidability of a theory and the decidability of its deductive closure; the terminology herein is unambiguous–other authors may not be so kind.

► **Theorem 15.1** (Craig's Trick)**.**
For a computable language $\mathcal{L}$, if a theory T is computably enumerable and deductively-closed, then T is computably axiomatizable.

Proof Sketch.
By assumption, we have a theory $T = \{\varphi_n : n \in \mathbb{N}\}$ such that, for some computable f, $f(n) = \ulcorner\varphi_n\urcorner$ for every $\mathbb{N}$. Using this f, define, for each n/φ_n, a new formula ψ_n as $\bigwedge_{i=1}^{n}\varphi_n$ (φ_n conjoined with itself n times). Define:

$$\Gamma = \{\psi_n | n \in \mathbb{N}\}$$

Γ is decidable since we may simply count the number of conjuncts in any well-formed formula, checking they are all the same, then use the f above to ensure that the conjunct is the correct one. Note finally that

$$T = \{\varphi : \Gamma \models \varphi\}$$

► **Theorem 15.2** (Complete and Computably Axiomatizable ⇒ Decidable)**.**
If a consistent theory T is both complete and computably axiomatizable, then T is decidable.

Proof Sketch.
Assume T is consistent since, otherwise, it is trivially decidable. To decide whether $\varphi \in T$, search for both a proof of φ from the computable axioms for φ and a proof of $\neg\varphi$ from the computable axioms for φ simultaneously, return whichever terminates first. Note that determining whether or not some encoding is a valid proof is computable so long as the process for determining whether or not a given premise is admissible is itself computable.

Of course, not all theories of interest are so well-behaved as to be both complete and computably axiomatizable; indeed, much work has gone into the question of determining the decidability or undecidability of a theory under more difficult circumstances; in this vein, we distinguish two notions stronger than simple undecidability:

Essentially Undecidable
A deductively-closed theory T in a computable $\mathcal{L}$ is *essentially undecidable* if and only if T has no consistent, decidable extension in $\mathcal{L}$.

Strongly Undecidable
A theory T in a computable $\mathcal{L}$ is *strongly undecidable* if and only if, for every deductively-closed $\mathcal{L}$-theory S such that $S \cup T$ is consistent, S is undecidable. A model A is called *strongly undecidable* if and only if $\mathrm{Th}(A)$ is strongly undecidable

Essential undecidability captures the idea that there is something about the sentences of the theory which forces undecidability; there is no way to 'repair' the undecidability of the theory. Strongly undecidable theories, in contrast, guarantee that no decidable theory is even so much as consistent with them–a distinctly stronger claim. Our picture, then, is as follows:

$$\text{Undecidable} < \text{Essentially Undecidable} < \text{Strongly Undecidable}$$

From just our previous work, we thus have:

► **Corollary** (Equivalences for Complete Theories)**.**
For a complete theory T, the following are equivalent:

(i) T is undecidable

(ii) T is essentially undecidable

(iii) T is not computably axiomatizable

This means, for example, that if a theory is known to be both computably axiomatizable and undecidable, then it must incomplete. The next proposition helps to demonstrate exactly how powerful a property strong undecidability really is:

► **Proposition 15.1** (Strongly Undecidable Theory ⇔ Strongly Undecidable Model)**.**
If T is any deductively-closed theory and A is some model such that $A \Vdash T$, then the following are equivalent:

(i) A is strongly undecidable

(ii) T is strongly undecidable

Proof Sketch.
Simply leverage definitions

► **Theorem 15.3** (Essentially Undecidable and Finitely Axiomatizable ⇒ Strongly Undecidable)**.**
If a theory T in a computable $\mathcal{L}$ is essentially undecidable and finitely axiomatizable, then T is strongly undecidable as well.

Proof Sketch.
Suppose S is a non-empty, deductively-closed, decidable theory consistent with an essentially undecidable T and Ψ is the finite axiomatization of T. A decidable, consistent extension of T can be constructed as follows: Proceed through all formulas φ, if $\bigwedge\Psi \rightarrow \varphi$ is in S, φ is in our extension; if not, it is not.

► **Theorem 15.4** (Incomplete, Decidable Theories have Consistent, Decidable, and Complete Extensions)**.**
Every incomplete, deductively-closed, and decidable theory T has a decidable, consistent, and complete extension [Tarski, 1949].

Proof Sketch.
By deductively-closed and incomplete, we have that the theory is consistent. We construct the consistent, complete, and decidable extension as follows: start with our theory and begin moving through the well-formed sentences of the language, starting with those having least Gödel number; for any given sentence, add it to our theory if its negation is not a consequence of the theory–otherwise, simply move to the next sentence. Leverage this process and the deductive-closure of the original theory to prove decidability.

► **Corollary** (Equivalences for Computably Axiomatizable, Deductively-Closed Theories)**.**
For a computably axiomatizable, deductively-closed theory T, the following are equivalent:

(i) T is essentially undecidable

(ii) T is consistent and every consistent, computably axiomatizable, deductively-closed extension of T is incomplete

(iii) T is consistent and no consistent, complete, deductively-closed extension of T is computably axiomatizable

---

15.3 Proving Decidability

A variety of equivalences and implications in hand, there are three main methods for proving that a particular, deductively-closed theory is, in fact, decidable:

(i) Quantifier Elimination

(ii) Model-Theoretic

(iii) Interpretations

The first two, quantifier elimination and model-theoretic, seek to leverage both the results above and in the model-theory portion of this text. In particular, we've shown that a deductively-closed theory which is complete, consistent, and computably axiomatizable must also be decidable. Since theories of interest are generally both consistent and computably axiomatizable, both of these methods are often used to demonstrate the last property: completeness.

15.3.1 Quantifier Elimination

Quantifier elimination is, as its name suggests, a method which attempts to eliminate the use of quantifiers in a language, to make them redundant and unnecessary. In general, this is, of course, impossible; quantifiers add expressiveness and–in the most general setting–it's impossible to eliminate them without the loss of this expressivity. Luckily, we often work in a setting far more limited than the most general; in particular, relative to some background theory T, it may very well be possible to eliminate the use of quantifiers. This is the most natural picture of quantifier elimination, and theories which behave in this manner are said to *have quantifier elimination*.

Formally, the *method of quantifier elimination* is slightly broader than detailed above; rather than the complete elimination of quantifiers, the method of quantifier elimination allows for a set of formulas, Φ, within which quantifiers are allowed, then reduces all formulas of the language to some boolean combination of these. That is, given a deductively-closed theory T in a language $\mathcal{L}$, we–in a moment of divine inspiration–identify some set of formulas, Φ, from $\mathcal{L}$ with a peculiar property:

($\star$) For any formula $\varphi(\bar{x})$ of $\mathcal{L}$, there is some boolean combination of formulas from Φ, $\psi(\bar{x})$, such that $T \models \forall\bar{x}[\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})]$

In other words, the set Φ–called *an elimination set for* T–can supply a boolean combination of its formulas which is T-equivalent to any formula of $\mathcal{L}$. In the particular case of sentences, then, all quantifiers except those occurring in the basic formulas of Φ are eliminated–we simply consider the T-equivalent formula from Φ instead!

Of course, the divine inspiration needed to pick Φ can be hard to come by; in general, there are four criterion 'good' choices tend to satisfy:

(1) Φ is reasonably small and non-redundant

(2) Formulas in Φ have straightforward meanings

(3) Constructing a boolean combination from Φ T-equivalent to a given sentence is a computable process

(4) Determining whether or not a boolean combination from Φ is a consequence of T is computable.

Given (3) and (4), it's easy to see that quantifier elimination shows the decidability of T; (1) and (2), in contrast, are empirical observations about sets for which logicians have established (3) and (4). Crucially, even without (3) and(4), all hope isn't lost for proving decidability; in particular, if it can be shown that T is complete with respect to all the sentences in Φ (for any sentence, either it or its negation are a consequence of T), then T must be complete–a property that, under many circumstances, readily gives decidability.

To achieve the full flavor of this method, we walk through a proof demonstrating the decidability of *Presburger arithmetic*:

■ **Fact.** To establish ($\star$), it's only necessary to consider an arbitrary formula of the form $\exists \bar{x}[\bigwedge \Psi(\bar{x})]$ where Ψ is a set containing only literals

Other natural theories whose decidability can be demonstrated using only the method of quantifier elimination are:

- The theory of the standard model of $\mathbb{N}$ under the signature containing only a successor function S and constant 0 [Herbrand, 1928]
- The theory of the standard model of $\mathbb{N}$ under the signature containing only a successor function S, multiplication function $\cdot$, and constant 0 [Skolem, 1930]
- Elementary Theory of Identity; the first-order validities of a language with empty signature [Lowenheim, 1915].
- Theory of Finitely Many Sets; the first-order validites of a language with only m unary relations [Lowenheim, 1915]
- Theory of Discrete Orders; axiomatized by,

$$\forall x \forall y[x \leq y \wedge y \leq x \rightarrow x = y]$$

$$\forall x \forall y[x \leq y \vee y \leq x]$$

$$\forall x \forall y \forall z[x \leq y \wedge y \leq z \rightarrow x \leq z]$$

$$\exists x \forall y[y \leq x \rightarrow x = y]$$

$$\forall x \exists y[x < y \wedge \forall z(x < z \rightarrow y \leq z)]$$

$$\forall x \forall y[y < x \rightarrow \exists z \forall w(z < x \wedge (z < w \rightarrow z \leq w))]$$

  [Langford, 1927]
- Theory of linear order in $\mathbb{Q}$; i.e. $\mathrm{Th}(\langle \mathbb{Q}, \leq \rangle)$
- Theory of Boolean Algebras [Tarski, 1949]
- Theory of $\mathbb{R}$; i.e. $\mathrm{Th}(\langle \mathbb{R}, 0, 1, +, -, \cdot, \leq \rangle)$ [Tarski, 1948]

15.3.2 Model Theory

Model theory, as we've seen, has a host of results about theories; thus, one way to show the decidability of a theory is to leverage these results along with those just established. Most commonly, this is done by showing a theory to be complete (usually via categoricity and the Łoś-Tarski theorem) or by systematically studying all possible extensions of a theory (and leveraging our earlier results in this direction).

It's worth noting that, while presented as separate methods, model theory and quantifier elimination often work hand-in-hand; the process of constructing elimination sets, in particular, benefits greatly from model-theoretic insights and results.

15.3.3 Interpretation

The method of interpretation is, at heart, the result of a simple, familiar idea: reduction. Suppose that, on one hand, we have a theory $T_?$ in a computable language $\mathcal{L}_?$ for which we wish to prove decidability; meanwhile, on the other hand, we have a provably decidable theory T_D formalized in a computable language $\mathcal{L}_D$. If there exists a computable function $f : \mathcal{L}_? \to \mathcal{L}_D$ such that, for every $\varphi \in L_?$,

$$\varphi \in T_? \Leftrightarrow f(\varphi) \in T_D$$

then $T_?$ must be decidable. Our decision procedure simply translates into $\mathcal{L}_D$ using f, then leverages the decision procedure for T_D. Of course, in line with our earlier work with reductions, reducibility is a two-way street. In particular, given two theories T_0, T_1 in two language $\mathcal{L}_0, \mathcal{L}_1$ and a computable $f : \mathcal{L}_0 \to \mathcal{L}_1$ such that

$$\varphi \in T_0 \Leftrightarrow f(\varphi) \in T_1$$

If T_0 is undecidable, then T_1 is undecidable as well–the contrapositive of our previous result.

From this general strategy a much more particular approach has developed; rather than work with theories, the focus is turned, instead, towards models. This gives the computable f needed for the reduction a great deal of structure and dovetails nicely with already existent model-theoretic practices (indeed, this is where the term 'interpretation' derives from). On the downside, however, this limits the applicability of interpretations to complete theories (which determine a model) or properties that interact nicely with models, e.g. strong undecidability and certain combinations of decidability/essential undecidability.

Formally, the method of interpretation focuses on defining one model within another; that is, given some model A, we construct an isomorphic, first-order definable copy of another model B within A. This construction is accomplished in a number of steps, starting with the interpretation function:

Interpretation Function
Given two computable languages $\mathcal{L}_0$ and $\mathcal{L}_1$, an *interpretation function* is a function $I : L_0 \to L_1$–that is, from the signature of $\mathcal{L}_0$ to the signature of $\mathcal{L}_1$–such that,

i. $I(c)$ for any constant symbol c is a well-formed formula $\varphi_c(v_0)$ of one free variable

ii. $I(R)$ for some n-ary relation symbol R is a well-formed formula $\varphi_R(v_0, \ldots, v_{n-1})$

iii. $I(f)$ for any n-ary function symbol f is a well-formed formula $\varphi_f(v_0, \ldots, v_n)$

In addition, with every interpretation function I we associate a well-formed formula $\varphi_D(v_0)$ of $\mathcal{L}_1$ with a single free-variable; in our scheme $\varphi_D(v_0)$ will determine the domain of the model we wish to interpret within A while I provides a simple translation of formulas from $\mathcal{L}_0$ to $\mathcal{L}_1$. Thus, given an $\mathcal{L}_1$-structure A, an interpretation function $I : L_0 \to L_1$, and an $\mathcal{L}_1$ formula $\varphi_D(v_0)$, we may construct an model-like entity A^I as follows:

A^I {

- Let $|A^I|$ be φ_D^A; that is, all elements a of A such that $A \Vdash \varphi_D[a]$
- For every constant symbol c of L_0, let c^{A^I} be the unique $b \in \varphi_D^A$ such that $A \Vdash \varphi_c[b]$
- For every relation symbol R of L_0, let R^{A^I} be $\varphi_R^A \cap (\varphi_D^A)^n$
- For every function symbol f of L_0, let f^{A^I} be defined by $f^{A^I}(\bar{a}) = b \Leftrightarrow A \Vdash \varphi_f(\bar{a}, b)$

In order for A^I to be a model, it must be the case that $A \Vdash \Phi_I$ where Φ_I is defined as follows:

Φ_I {

- $\exists v_0[\varphi_D(v_0)]$
- $\exists! v_0[\varphi_c(v_0) \wedge \varphi_D(v_0)]$ for every constant symbol c of L_0
- $\forall v_0 \ldots \forall v_{n-1}[\bigwedge_{i=0}^{n-1} \varphi_D(v_i) \to \exists! v_n[\varphi_f(v_0, \ldots, v_n) \wedge \varphi_D(v_n)]$ for every function symbol f of L_0

The first condition on Φ_I guarantees that the domain of A^I is nonempty, while the second and third guarantee that the constant and function definitions of A^I are respectively acceptable.

> **Interpretable**
> Let B be an $\mathcal{L}_0$ structure and A a $\mathcal{L}_1$ structure. B is *interpretable* in A if and only if $B \cong A^I$ for some interpretation function I such that $A \Vdash \Phi_I$; B is *computably interpretable* in A if and only if $B \cong A^I$ for some computable interpretation function I such that $A \Vdash \Phi_I$

At long last, we turn to how interpretability can be leveraged. If $\mathrm{Th}(A)$ or $\mathrm{Th}(B)$ is of interest, any interpretation function I can be used to generate a full, computable map $f : \mathcal{L}_0 \to \mathcal{L}_1$ reducing $\mathrm{Th}(B)$ to $\mathrm{Th}(A)$, just as in the introduction to this section.

► **Proposition 15.2** (Computably Interpretable, Strongly Undecidable Model ⇒ Strongly Undecidable)**.**
If A is computably interpretable in B, and A is strongly decidable, then B is strongly undecidable as well

---

► **Proposition 15.3** (Computably Interpretable, Essentially Undecidable ⇒ Essentially Undecidable)**.**
If A is computably interpretable in B, and $\mathrm{Th}(A)$ is essentially decidable, then $\mathrm{Th}(B)$ is essentially undecidable as well

---

► **Proposition 15.4** (Computably Interpretable in a Decidable Model ⇒ Decidable)**.**
If A is computably interpretable in a decidable model B, then $\mathrm{Th}(A)$ decidable

---

15.3.4 The Decidability of Validity

(i) Let $\mathcal{L}$ be recursively presented. Let E be the set of all $\mathcal{L}$-sentences of the form: $\exists \bar{v}\theta$ where θ is quantified or free. Then, SAT over E (that is, $\{\alpha \in E | Sat(\alpha)\}$) is computable since if $\alpha \in E$ if and only if α has a finite model (where this latter is a Σ^0_1 property).

(ii) Let $\mu = \{\neg\alpha | \alpha \in E\}$ or rather $\alpha \in \mu$ are of the form $\forall\bar{v}\gamma$ where γ is q.f., then SAT^μ is Π^0_1-complete for $\mathcal{L}$ having infinitely many function symbols of each arity and a binary relation symbol. $SAT^{LST} \leq_m SAT^\mu$ given θ a sentence of LST. Find γ such that

$$\models \theta \leftrightarrow \forall v_1 \exists v_2 \forall v_3 \ldots \gamma$$

where γ is q.f. This is called prenex form and is computable. We now successivbley replcae quatnifiers with skolem functions.

$$\beta^1 = \forall v_1 \forall v_3 \exists v_4 \ldots \gamma(v_1, f_2(v_1), v_3, \ldots)$$
$$\beta^2 = \forall v_1 \forall v_3 \forall v_5 \ldots \gamma(v_1, f_2(v_1), v_3, f_4(v_1, v_3), \ldots)$$

Then, $B^i \in SAT \Leftrightarrow \beta^{i+1} \in SAT$

(iii) Suppose that $\mathcal{L}$ has only unary predicate symbols and no function symbols. Then $SAT^{\mathcal{L}}$ is computable (this is again a result of the following property φ has a model iff φ has a finite model)

(iv) If $\mathcal{L}$ has 2 unary function symbols $SAT^{\mathcal{L}}$ is Π^0_1 complete; if there's just one and nothing else, $Sat^{\mathcal{L}}$ is computable.

■ **Fact.** (can use in exercises) Can use (iv) to get an essentially undecidable, not strongly undecidable, axiomatizable T again using inseparable r.e. sets.

15.4 Proving Undecidability

Chapter 16

Arithmetic & Incompleteness

16.1 True Arithmetic

True arithmetic is that understanding of arithmetic and the natural numbers which is taught to us from elementary through high school; the operations of multiplication, addition, etc. behave as usual while the only elements are those in $\mathbb{N}$, under the order-type ω. This is our inuitive picture of the natural numbers. We formalize this picture by selecting a first-order language in which we will work:

> **The Language of First-Order Arithmetic**
> The language of first-order arithmetic, abbreviated $\mathcal{L}_A$, is the first-order language generated from the following signature:
>
> (i) Binary Function Symbols: $+, \cdot$
>
> (ii) Constants: $0, 1$
>
> (iii) Binary Relation Symbol: $<$
>
> This signature is often referred to, for historical reasons, as the*signature of Peano arithmetic*.

■ **Fact.** Adding exponenetiation to the signature of $\mathcal{L}_A$ doesn't add expressivity; in other words, exponentiation is definable in $\mathcal{L}_A$

and then designating our intended model of $\mathcal{L}_A$ as $\mathcal{N}$, the *standard model*. To complete the picture and return to the topic of this section, we identify the term *true arithmetic* with $\mathrm{Th}(\mathcal{N})$ in the language $\mathcal{L}_A$. Recalling that much of our work with relations in computability theory is using true arithmetic (and, by extension, $\mathcal{L}_A$), it should now be quite obvious where the term *arithmetic* derives from in *the arithmetic hierarchy*:

► **Theorem 16.1** (Definable over $\mathcal{N} \Leftrightarrow$ Arithmetic)**.**
The following are equivalent:

i. A relation R is definable over $\mathcal{N}$ in $\mathcal{L}_A$

ii. A relation R is *arithmetic*; that is R appears in the arithmetic hierarchy at some level n

---

Indeed, as we saw earlier, the correspondence is even more exact; relations definable in $\mathcal{N}$ by a bounded formula are primitive recursive (although the converse is not true), the Σ^0_1-relations are exactly those definable in $\mathcal{N}$ by an existential formula, so on and so forth.

► **Corollary.**
Let

$$Th^{\mathcal{N}}_{\Sigma^0_1} = \{\ulcorner\alpha\urcorner \mid \alpha \text{ is an existential sentence of } \mathcal{L}^{ENT} \text{ and } \mathcal{N} \Vdash \alpha\}$$

Then, $K \leq_m Th^{\mathcal{N}}_{\Sigma^0_1}$

► **Theorem 16.2.**
$Th^{\mathcal{N}}_{\Sigma^0_1}$ is Σ^0_1-complete

Proof.
First, show it's Σ^0_1 by showing:

$$Sat(\ulcorner\varphi\urcorner, s) \Leftrightarrow \mathcal{N} \Vdash \varphi[(s)_0, \ldots, (s)_{lh(s)-1}] \wedge \varphi \text{ is bounded}$$

is primitive recursive (primitive recursion on $\ulcorner\varphi\urcorner$). Then,

$$\ulcorner\alpha\urcorner \in Th^{\mathcal{N}}_{\Sigma^0_1} \Leftrightarrow \alpha = \exists v \beta \text{ for some bounded } \beta \wedge \exists s Sat(\ulcorner\beta\urcorner, s)$$

Next, show $K \leq_m Th^{\mathcal{N}}_{\Sigma^0_1}$. Let

$$K(n) \Leftrightarrow \mathcal{N} \Vdash \alpha[n]$$

(α existential), then

$$K(n) \Leftrightarrow \ulcorner\alpha(\underline{n})\urcorner \in Th^{\mathcal{N}}_{\Sigma^0_1}$$

Then,

$$f(n) = \ulcorner\alpha(\underline{n})\urcorner$$

is primitive recursive. □

Let $\mathcal{N}^+ = (\mathbb{N}, +, \cdot, exp, S, 0, \leq, T^k \text{ for every } k \in \mathbb{N})$ where T^k is intended to represent the Kleene T-predicate. It follows that $Th^{\mathcal{N}^+}_{\Sigma^0_1}$ is, trivially, Σ^0_1-complete.

► **Corollary.**
$\mathcal{N}$ is strongly undecidable

Validities of a Language
Let $\mathcal{L}$ be a first-order language; the *validities of* $\mathcal{L}$, denoted $\text{Val}_{\mathcal{L}}$, is the theory

$$\text{Val}_{\mathcal{L}} = \{\varphi \,:\, \varphi \text{ is a sentence of } \mathcal{L} \text{ and } \vdash \varphi\}$$

► **Corollary** ($\text{Val}_{\mathcal{L}_A}$ is Σ^0_1-Complete)**.**
The validities of $\mathcal{L}_A$, $\text{Val}_{\mathcal{L}_A}$, form a Σ^0_1-complete theory.

Proof.

$\text{Thm}_Q \leq_m Val^{\mathcal{L}^{PA}}$: $\alpha \in Thm_Q iff \mapsto \bigwedge\!\!\bigwedge_{\beta \in Q} \beta \rightarrow \alpha$ which requires that Q is finitely axiomatized! □

We've already shown that any such theory is Σ^0_1-complete and thus undecidable.

► **Theorem 16.3.**
Suppose $\mathcal{N}$ is computably definable in some model D. Let T be axiomatizable and true in D. Then, T is Σ^0_1-complete and D is strongly undecidable

---

Proof.
Let I: symbols of $\mathcal{L}^{PA}$ into wffs in the language $\mathcal{L}$ of D be an interpretation function. Let Φ^I be the sentences of $\mathcal{L}$ saying this about I (i.e. $D \models \Phi^I$ iff $D^I$is an $\mathcal{L}^{PA}$ structure). Let $\alpha \in S$ iff $T \cup \Phi^I \vdash \alpha^I$. α^I is defined by induction on α in such a way that for any $D \models \Phi^I$, $D \models \alpha^I$ iff $D^I \models \alpha$. E.g.,

$$(\exists v\gamma)^I = \exists v(\theta^I_u(v) \wedge \gamma^I$$

$$(f(v_1, \ldots, v_n) = v_{n+1})^I = \alpha^I_f(v_1, \ldots v_{n+1})$$

S is a theory: Suppose $S \vdash \gamma$. Let $D \models T \cup \Phi^I$. $D^I \models S$ ($D \models \alpha^I$ for every $\alpha \in S$.) So, $D^I \models \gamma$, so $D \models \gamma^I$, and thus $T \cup \Phi^I \vdash \gamma^I$. Moreover, $\mathcal{N}^- \models S$ ($D^I \models S$). So S is Σ^0_1-complete (S is axiomatizable since T was). $S \leq_m T$ since $\alpha \in S$ iff $T \vdash \bigwedge\Phi^I \rightarrow \alpha^I$. The function f mapping α to $\bigwedge\Phi^I \rightarrow \alpha^I$ is recursive. □

16.2 First-Order Peano Arithmetic

First-order Peano arithmetic, often shortened to simply *Peano arithmetic*, is, in conception, a dodge; an attempt to circumvent the difficult philosophical and set-theoretic questions inherent in the original, second-order Dedekind-Peano induction axiom

$$\forall P[0 \in P \wedge \forall x[(x \in P \rightarrow x + 1 \in P) \rightarrow \forall y[y \in P]]]$$

by replacing it with an infinite number of weaker, first-order induction axioms. With the second-order axiom, the Dedekind-Peano axioms succeed in characterizing $\mathcal{N}$ up to isomorphism; as we saw above, the first-order attempt was doomed to failure from the start. Nonetheless, Peano arithmetic has become a first-order theory of significant import, characterizing a great deal of the most important aspects of $\mathcal{N}$. Intuitively, Peano arithmetic is true arithmetic–if you can only see the natural numbers; that is, on the natural numbers $\mathbb{N}$, Peano arithmetic is perfect–it simply fails to rule out the existence of elements not in $\mathbb{N}$.

Peano Arithmetic
Peano arithmetic, denoted PA, is the deductively-closed theory in $\mathcal{L}_A$ axiomatized by the following:

(Id1) $\forall x[(x + 0 = x) \wedge (x \cdot 0 = 0)]$

(Id2) $\forall x[x \cdot 1 = x]$

(A1) $\forall x\forall y\forall z[(x + y) + z = x + (y + z)]$

(A2) $\forall x\forall y[x + y = y + x]$

(M1) $\forall x\forall y\forall z[(x \cdot y) \cdot z = x \cdot (y \cdot z)]$

(M2) $\forall x\forall y[x \cdot y = y \cdot x]$

(D1) $\forall x\forall y\forall z[x \cdot (y + z) = x \cdot y + x \cdot z]$

(L1) $\forall x\forall y\forall z[x < y \wedge y < z \rightarrow x < z]$

(L2) $\forall x[x \not< x]$

(L3) $\forall x\forall y[x < y \vee x = y \vee y < x]$

(O1) $0 < 1 \wedge \forall x[x > 0 \rightarrow x \geq 1]$

(O2) $\forall x[x \geq 0]$

(O3) $\forall x\forall y\forall z[x < y \rightarrow x + z < y + z]$

(O4) $\forall x\forall y\forall z[0 < z \wedge x < y \rightarrow x \cdot z < y \cdot z]$

(O5) $\forall x\forall y[x < y \rightarrow (\exists z)[x + z = y]]$

along with an instance of

$$\forall \bar{y}[\varphi(0, \bar{y}) \wedge \forall x[\varphi(x, \bar{y}) \rightarrow \varphi(x + 1, \bar{y})] \rightarrow \forall x\varphi(x, \bar{y})]$$

for every $\varphi(x, \bar{y})$ in $\mathcal{L}_A$.

Comparing the original Dedekind-Peano second-order induction axiom with the replacement first-order induction schema, we trade induction over arbitrary subsets of $\mathbb{N}$–a philosophically and set-theoretically questionable act–for induction over the first-order, $\mathcal{L}_A$ definable subsets of $\mathbb{N}$–a safer, but weaker ability. It's precisely this loss which–as we will see–allows the formulation of so-called *non-standard models* of PA.

Note that, while not finitely axiomatizable, PA is computably axiomatizable, and thus, by earlier work, computably enumerable.

16.3 Robinson's Arithmetic

Robinson's arithmetic is the result of stripping all axioms from Peano arithmetic which are not strictly necessary for the establishment of PA's undecidability and incompleteness, thus giving a minimalist system in which Gödel's famous theorems still hold.

Robinson's Arithmetic
Robinson's arithmetic, denoted Q, is the deductively-closed theory in $\mathcal{L}_A$ axiomatized by the following:

($\mathrm{Id1}_1$) $\forall x[x + 0 = x]$

($\mathrm{Id1}_2$) $\forall x[x \cdot 0 = 0]$

($\mathrm{A1}^-$) $\forall x \forall y[x + (y + 1) = (x + y) + 1]$

($\mathrm{D1}^-$) $\forall x \forall y[x \cdot (y + 1) = (x \cdot y) + x]$

(L4) $\forall x \forall y[x < y + 1 \leftrightarrow x \leq y]$

(L5) $\forall x[x = 0 \vee \exists y[x = y + 1]]$

(L6) $\forall x \forall y[x + 1 = y + 1 \rightarrow x = y]$

(O6) $\forall x[x + 1 \neq 0]$

(O7) $\forall x \forall y[x < y + 1 \leftrightarrow \exists z[(x + z) + 1 = y]]$

There are a number of facts about Q which follow just from considering the axioms given; first, Q is finitely axiomatizable, and thus computably enumerable. Next, a bit of work shows that all the axioms of Q are either axioms or consequences of axioms of PA. That is, $Q \subset \mathrm{PA}$ and $\mathcal{N} \Vdash Q$. Finally, Q lacks any induction axiom, meaning it's possible to prove, for every 'natural number' every instance of a property $\varphi(x)$, but yet not obtain the universalization $\forall x \varphi(x)$.

16.4 Gödel's Incompleteness Theorems

16.4.1 Representability in a Theory

Numeral
The set of numerals of Q/PA is defined inductively as follows:

The constant 0 is a numeral

If α is a numeral, then the term $S(\alpha)$ is a numeral

The *numerals* of Q/PA, then, are simply the closed terms we instinctively associate with the natural numbers; e.g. the numeral for 2 is 'SS0'. Since actually writing numerals out is rather tiresome, it's common to either overline or underline a number in order to denote its numeral:

$$\overline{2} = \underline{2} = SS0$$

We adopt the latter convention.

Representability
Given a theory T in a language $\mathcal{L}$ with a suitable notion of numeral,

(i) A k-ary relation R is *weakly represented in* T by a well-formed formula φ of $\mathcal{L}$ with k free variables if and only if for any k-tuple of numbers $\bar{n}$,

$$R(n_1, \ldots, n_k) \Leftrightarrow T \vdash \varphi[\underline{n_1}, \ldots, \underline{n_k}]$$

(ii) A k-ary relation R is *strongly represented in* T by a well-formed formula φ of $\mathcal{L}$ with k free variables if and only if for any k-tuple of numbers $\bar{n}$,

$$R(n_1, \ldots, n_k) \Rightarrow T \vdash \varphi[\underline{n_1}, \ldots, \underline{n_k}]$$
$$\neg R(n_1, \ldots, n_k) \Rightarrow T \vdash \neg\varphi[\underline{n_1}, \ldots, \underline{n_k}]$$

(iii) A k-ary function f is *functionally represented in* T by a well-formed formula φ with $k+1$ free variables if and only if for any natural numbers $n_1, \ldots, n_k, m$,

$$f(\bar{n}) = m \Rightarrow T \vdash \varphi[\underline{n_1}, \ldots, \underline{n_k}, \underline{m}]$$
$$f(\bar{n}) = m \Rightarrow T \vdash (\exists! z)\varphi[\underline{n_1}, \ldots, \underline{n_k}, z]$$

► **Proposition 16.1** (Strong Representation in PA)**.**
If PA is consistent and R is strongly represented in PA, then:

$$R(n_1, \ldots, n_k) \Leftrightarrow \text{PA} \vdash \varphi[\underline{n_1}, \ldots, \underline{n_k}]$$

$$\neg R(n_1, \ldots, n_k) \Leftrightarrow \text{PA} \vdash \neg\varphi[\underline{n_1}, \ldots, \underline{n_k}]$$

Thus, a relation is strongly represented in PA if and only if it is weakly represented by some formula and its negation is weakly represented by the negation of the same formula.

► **Proposition 16.2** (Representation in PA)**.**
The following hold for the theory PA defined above:

(i) The identity relation is strongly represented in PA by the formula $x = y$

(ii) Addition of natural numbers is represented in PA by $x + y = z$

(iii) Multiplication of natural numbers is represented in PA by $x \cdot y = z$

(iv) The 'less than' relation on the natural numbers is strongly represented in PA by $x < y$

(v) A relation R is strongly representable in PA if and only if its characteristic function χ_R is representable in PA

(vi) For any $\mathcal{L}_A$-formula and any $n \in \mathbb{N}$,

$$\text{PA} \vdash \varphi(0) \wedge \varphi(1) \wedge \cdots \wedge \varphi(\underline{n}) \wedge x < \underline{n} + 1 \rightarrow \varphi(x)$$

(vii) For any $\mathcal{L}_A$-formula φ and any $n \in \mathbb{N}$, if for every $m < n$, $\text{PA} \vdash \neg\varphi(\underline{m})$ and $\text{PA} \vdash \varphi(\underline{n})$ as well, then

$$\text{PA} \vdash \forall x[\varphi(x) \wedge \forall y[y < x \rightarrow \neg\varphi(y)] \leftrightarrow x = \underline{n}]$$

Proof Sketch.
The above results, though sometimes tedious to prove, all straightforwardly rely on the axioms of PA, requiring only clever usage there of.

► **Theorem 16.4** (Computable Functions are PA-Representable)**.**
Every computable function is representable in PA; in fact, every computable function is representable by a Σ_1^0-formula in PA

Proof Sketch.
Leverage the definition of the computable functions; show that the starting functions are representable in PA, and that representability in PA is closed under the closure operations of the computable functions.

► **Corollary** (Computable Relations are Strongly Represented in PA)**.**
Every computable relation is strongly representable in PA; in fact, every computable relation is strongly representable by a Σ_1^0-formula in PA

Sound
A theory T is D-sound for a class of sentences D if and only if whenever φ is a D sentence and $T \vdash \varphi$, then φ is 'true' (what this means, of course, depends on context; usually truth in some standard model).

Of course, PA was built with the intent that $\text{PA} \subset \text{Th}(\mathcal{N})$; unfortunately, this hasn't been proven. Indeed, although there is a great deal of anecdotal evidence, it is–for all intents and purposes–an open question whether or not PA is even consistent. Soundness results with regard to PA, then, are not nearly as trivial as they may, at first, seem.

► **Theorem 16.5** (PA is Σ_1^0-Sound)**.**
If PA is consistent, then PA is Σ_1^0-sound; that is–assuming consistency–for any Σ_1^0, $\mathcal{L}_A$-formula φ such that $\text{PA} \vdash \varphi$, $\mathcal{N} \Vdash \varphi$

Proof Sketch.
Leverage the representability of computable relations, note that if PA $\vdash \varphi[\underline{n}]$, then PA $\vdash \exists x \varphi(x)$

► **Corollary** (PA is Π^0_1-Sound)**.**
If PA is consistent, then PA is Π^0_1-sound

► **Theorem 16.6** (Σ^0_1 and True in $\mathcal{N} \Rightarrow$ Provable in PA)**.**
If φ is a Σ^0_1 sentence of $\mathcal{L}_A$ such that $\mathcal{N} \Vdash \varphi$, then PA $\vdash \varphi$

Proof Sketch.
Note that PA strongly represents the formula $\varphi[a]$ by itself (our results above translate addition in $\mathcal{N}$ to addition in PA, etc.) where a is the witness to $\exists x \varphi(x)$. By our earlier results, PA $\vdash \varphi[\underline{a}]$, and thus PA $\vdash \exists x \varphi(x)$.

► **Theorem 16.7** (Computably Axiomatizable, Extends PA, and Σ^0_1-Sound $\Rightarrow$ Σ^0_1-Complete)**.**
If a deductively-closed theory T is computably axiomatizable, PA $\subseteq T$, and T is Σ^0_1-sound, then T is Σ^0_1-complete.

Proof Sketch.
Computable axiomatizability guarantees that T is Σ^0_1, while extending PA and being Σ^0_1-sound guarantee that T is consistent and still correctly proves Σ^0_1 facts about $\mathcal{N}$–making it Σ^0_1-hard.

► **Theorem 16.8** (PA is Essentially Undecidable)**.**
The theory PA is essentially undecidable.

Proof Sketch.
Let $A, B \subseteq \mathbb{N}$ be disjoint, Σ^0_1, and computably inseparable. Since A, B are Σ^0_1, PA strongly represents them; say by $\varphi_A(x)$ and $\varphi_B(x)$. Any decidable extension of Q can thus computably separate A and B using φ_A, φ_B–a contradiction.

16.4.2 Representability in Q

Although we present it in less detail, many of the representability results above carry over verbatim into Robinson's arithmetic; of course, this is because Robinson selected precisely the axioms needed to prove them.

► **Theorem 16.9** (Computable Functions are Q-Representable)**.**
Every computable function is representable in Q

► **Corollary** (Computable Relations are Strongly Represented in Q)**.**
Every computable relation is strongly representable in Q

► **Corollary** (Σ^0_1 Relations are Weakly Represented in Q)**.**
Every Σ^0_1 relation is weakly represented in Q by a Σ^0_1 formula

► **Theorem 16.10** (Q is Σ^0_1-Sound)**.**
If Q is consistent, then Q is Σ^0_1-sound; that is–assuming consistency–for any Σ^0_1, $\mathcal{L}_A$-formula φ such that $Q \vdash \varphi$, $\mathcal{N} \Vdash \varphi$

► **Theorem 16.11** (Σ^0_1 and True in $\mathcal{N} \Rightarrow$ Provable in Q)**.**
If φ is a Σ^0_1 sentence of $\mathcal{L}_A$ such that $\mathcal{N} \Vdash \varphi$, then $Q \vdash \varphi$

► **Corollary** (Q is Σ^0_1-Complete)**.**
The theory Q is Σ^0_1-complete

► **Theorem 16.12** (Computably Axiomatizable, Extends Q, and Σ^0_1-Sound $\Rightarrow$ Σ^0_1-Complete)**.**
If a deductively-closed theory T is computably axiomatizable, $Q \subseteq T$, and T is Σ^0_1-sound, then T is Σ^0_1-complete.

Proof Sketch.
Computable axiomatizability guarantees that T is Σ^0_1, while extending Q and being Σ^0_1-sound guarantee that T is consistent and still correctly proves Σ^0_1 facts about $\mathcal{N}$–making it Σ^0_1-hard.

► **Theorem 16.13** (Q is Essentially Undecidable)**.**
The theory Q is essentially undecidable.

Proof Sketch.
Let $A, B \subseteq \mathbb{N}$ be disjoint, Σ^0_1, and computably inseparable. Since A, B are Σ^0_1, Q strongly represents them; say by $\varphi_A(x)$ and $\varphi_B(x)$. Any decidable extension of Q can thus computably separate A and B using φ_A, φ_B–a contradiction.

16.4.3 The Diagonalization Lemma

Recall that, while it is often hidden, all of the results thus far make recourse to some fixed, well-chosen, and computable encoding of the language $\mathcal{L}_A$ wherein $\ulcorner\varphi\urcorner$ is the number encoding the $\mathcal{L}_A$-formula φ. The next result is one of the most important to the establishment of the incompleteness theorems; it shows the validity of the 'self-reference' which Gödel's proof relies so heavily on.

► **Lemma 16.1** (Diagonalization Lemma)**.**
Let T be an $\mathcal{L}_A$ theory that Σ_1^0-represents all computable functions. Given a well-formed formula $\psi(x)$ of one free variable from $\mathcal{L}_A$, there is a sentence φ of $\mathcal{L}_A$ such that

$$Q \vdash \varphi \leftrightarrow \psi(\underline{\ulcorner \varphi \urcorner})$$

Moreover, if $\psi(x)$ is Π_1^0, then φ may be taken to be equivalent to a Π_1^0 sentence also.

Proof.
Define

$$sub(m,n) = \begin{cases} \ulcorner \varphi[n] \urcorner & \text{if } m \text{ encodes a } \mathcal{L}_A\text{-formula } \varphi \text{ with one free variable} \\ 0 & \text{otherwise} \end{cases}$$

Note that *sub* is computable, and thus functionally represented in Q; suppose that the formula which does so is $\theta_{\text{sub}}(x,y,z)$. This established, let a well-formed formula of $\mathcal{L}_A$ with one free variable, $\psi(x)$, be given. Define $\gamma(v)$ as

$$\exists u(\theta_{\text{sub}}(v,v,u) \wedge \psi(u))$$

Colloquially, $\gamma(v)$ claims that there is a u such that $u = \ulcorner \varphi_v[\ulcorner v \urcorner] \urcorner$ ($v = \ulcorner \varphi_v(x) \urcorner$) and $\psi[u]$. Let φ be $\gamma(\ulcorner \gamma \urcorner)$. Thus, φ is

$$\exists u(\theta_{\text{sub}}(\ulcorner \gamma \urcorner, \ulcorner \gamma \urcorner, u) \wedge \psi(u))$$

and, since Q functionally represents *sub*,

$$T \vdash \forall u[\theta_{\text{sub}}(\underline{\ulcorner \gamma \urcorner}, \underline{\ulcorner \gamma \urcorner}, u) \leftrightarrow u = \underline{\ulcorner \gamma(\underline{\ulcorner \gamma \urcorner}) \urcorner}]$$

Simply using the definition of φ,

$$T \vdash \varphi \rightarrow \exists u(\theta_{\text{sub}}(\underline{\ulcorner \gamma \urcorner}, \underline{\ulcorner \gamma \urcorner}, u) \wedge \psi(u))$$

And, by the formula derived via Q functionally representing *sub*,

$$T \vdash \exists u(\theta_{\text{sub}}(\underline{\ulcorner \gamma \urcorner}, \underline{\ulcorner \gamma \urcorner}, u) \wedge \psi(u)) \rightarrow u = \underline{\ulcorner \varphi \urcorner}$$

Thus,

$$T \vdash \varphi \rightarrow \psi(\underline{\ulcorner \varphi \urcorner})$$

To establish the converse, by the definition of φ,

$$T \vdash \psi(\underline{\ulcorner \varphi \urcorner}) \rightarrow \theta_{\text{sub}}(\underline{\ulcorner \gamma \urcorner}, \underline{\ulcorner \gamma \urcorner}, \underline{\ulcorner \varphi \urcorner}) \wedge \psi(\underline{\ulcorner \varphi \urcorner})$$

And thus,

$$T \vdash \psi(\underline{\varphi}) \rightarrow \exists u(\theta_{\text{sub}}(\underline{\ulcorner \gamma \urcorner}, \underline{\ulcorner \gamma \urcorner}, u) \wedge \psi(u))$$
$$T \vdash \psi(\underline{\varphi}) \rightarrow \varphi$$

which gives,

$$T \vdash \psi(\underline{\varphi}) \leftrightarrow \varphi$$

□

There are a few interesting characteristics of the proof given above; first, the proof in no way depended on what $\psi(x)$ actually is. The fixed point, while depending on ψ, is not in any way based on the content of ψ. In every case, there is a fixed point of the same form as φ above. The fixed point itself, φ above, is often understood to claim 'I have property ψ'; while this is certainly a fair description of how φ behaves, the direct self-reference is misleading. φ actually 'says' "there is something which is both ψ and has properties X" where the properties X are enough for Q to uniquely determine that something as the encoding of that very sentence. The self-reference utilized above is thus

indirect. Finally, while T was used throughout the proof above, note that both Q and PA meet the criteria for T, and thus the fixed point lemma holds in both.

16.4.4 The Incompleteness Theorems

► **Theorem 16.14** (Gödel's First Incompleteness Theorem)**.**
Let T be a computably axiomatizable, consistent, deductively-closed $\mathcal{L}_A$-theory such that $Q \subseteq T$. Then, there is a Π^0_1 $\mathcal{L}_A$ sentence φ such that $\mathcal{N} \Vdash \varphi$ and $T \nvdash \varphi$.

Proof Sketch.
Given that T is computably axiomatizable, it's computable to check, given a reasonable encoding scheme, whether any given natural number encodes a proof of some particular statement. Thus, the following is a computable relation:

$$\text{Proof}_T(x, y) \Leftrightarrow x \text{ is a proof of } y \text{ from the axioms of } T$$

Since Q strongly represents all computable relations, there must be some $\mathcal{L}_A$ formula $\theta_{\text{Proof}_T}(x, y)$ with the requisite properties. Noting that $\neg\exists x[\theta_{\text{Proof}_T}(x, y)]$ is an $\mathcal{L}_A$-formula with only one free variable, the diagonalization lemma gives that there is a φ such that

$$Q \vdash \varphi \leftrightarrow \neg\exists x[\theta_{\text{Proof}_T}(x, \underline{\ulcorner\varphi\urcorner})]$$

Then, $T \nvdash \varphi$ since, if it did, the proof must have an encoding e, so that $\text{Proof}_T(e, \ulcorner\varphi\urcorner)$. Then,

$$Q \vdash \theta_{\text{Proof}_T}(\underline{e}, \underline{\ulcorner\varphi\urcorner})$$

$$Q \vdash \exists x[\theta_{\text{Proof}_T}(x, \underline{\ulcorner\varphi\urcorner})]$$

$$Q \vdash \neg\varphi$$

$$Q \vdash \varphi$$

Thus, T would be inconsistent, and so $T \nvdash \varphi$. But, $\mathcal{N} \Vdash \varphi$. Note finally that φ is, by construction, equivalent to a Π^0_1, namely $\neg\exists x[\theta_{\text{Proof}_T}(x, \underline{\ulcorner\varphi\urcorner})]$.

■ **Remark.** This is, of course, not quite the oft quoted statement of Gödel's first incompleteness theorem. In particular, it leaves open the possibility that T is, in fact, complete, and that $T \vdash \neg\varphi$. There are a number of ways to get around this and establish the incompleteness of T. Strengthening the assumptions to include Σ^0_1-soundness suffices, as well as Gödel's tact: ω-consistency instead of simply consistency.

> **ω-Consistency**
> A theory T which interprets the natural numbers is ω-consistent if and only if, for every $\varphi(x)$, if $T \vdash \varphi(0)$, $T \vdash \varphi(1)$, $T \vdash \varphi(\underline{2})$, ..., then $T \nvdash \exists x[\neg\varphi(x)]$

In truth, however, neither are necessary. A clever adjustment to Gödel's original proof, known as *Rosser's trick*, suffices to establish the stronger conclusion without strengthening the assumptions given above:

► **Theorem 16.15** (Gödel-Rosser Incompleteness Theorem)**.**
Let T be a computably axiomatizable, consistent, deductively-closed $\mathcal{L}_A$-theory such that $Q \subseteq T$. Then, T is incomplete. In particular, there is a Π^0_1 $\mathcal{L}_A$ sentence φ such that $\mathcal{N} \Vdash \varphi$, $T \nvdash \varphi$, and $T \nvdash \neg\varphi$.

Proof Sketch.
Given that T is computably axiomatizable, it's computable to check, given a reasonable encoding scheme, whether any given natural number encodes a proof of some particular statement. Thus, the following is a computable relation:

$$\text{Proof}_T(x, y) \Leftrightarrow x \text{ is a proof of } y \text{ from the axioms of } T$$

Since Q strongly represents all computable relations, there must be some $\mathcal{L}_A$ formula $\theta_{\text{Proof}_T}(x, y)$ with the requisite properties; similarly,

$$\text{neg}(x) = \begin{cases} \ulcorner \neg \psi \urcorner & x = \ulcorner \psi \urcorner \text{ for some } \mathcal{L}_A \text{ formula } \psi \\ 0 & \text{otherwise} \end{cases}$$

is also computable, and thus represented by some $\mathcal{L}_A$-formula, $\theta_{\text{neg}}(x, y)$. Note that the following, denoted $\psi(y)$ for short, is an $\mathcal{L}_A$-formula with one free variable:

$$\forall x[\theta_{\text{Proof}_T}(x, y) \rightarrow (\exists u < x)(\exists z)[\theta_{\text{Proof}_T}(u, z) \wedge \theta_{\text{neg}}(y, z)]]$$

Less formally, the formula $\psi(y)$ above states "for every proof of the formula encoded by y, there is a shorter proof of the negation of the formula encoded by y". It's this shift from a formula claiming its own unprovability in Gödel's proof to one claiming the earlier provability of its negation which comprises Rosser's contribution. From here, the rest of the proof follows easily.

■ **Notation.** The computability of proof for computably axiomatizable theories and the strong representability of this relation in theories extending Q allowed us, in the proofs above, to talk about the 'PA-provability' of various sentences *in PA itself*. We continue this trend by utilizing the formula which strongly represents provability to define a $\mathcal{L}_A$ sentence which claims that $T \supseteq Q$ is consistent. Let Con_T denote

$$\neg \exists x[\theta_{\text{Proof}_T}(x, \underline{\ulcorner \bot \urcorner})]$$

► **Theorem 16.16** (Gödel's Second Incompleteness Theorem)**.**
Let T be axiomatizable, consistent, deductively-closed, and $\text{PA} \subseteq T$. Then, $T \nvdash \text{Con}_T$

Proof Sketch.
Using our work in Gödel's first and the fixed point lemma, let φ be such that

$$Q \vdash \varphi \leftrightarrow \neg\exists x[\theta_{\text{Proof}_T}(x, \underline{\ulcorner\varphi\urcorner})]$$

We'll show

$$\text{PA} \vdash \varphi \leftrightarrow \text{Con}_T$$

which, by our previous work, implies $\text{PA} \nvdash \text{Con}_T$. Throughout the remainder of the proofs, we use $\Box_T\varphi$ as an abbreviation of $\exists x[\theta_{\text{Proof}_T}(x, \underline{\ulcorner\varphi\urcorner})]$ (the notation is, of course, remnant of modal logic; in particular, the provability logic *GL*; see Boolos' text on the subject). To start, note that

$$\text{PA} \vdash \varphi \to \neg\Box_T\varphi$$

and, by a reductio on the negation of the consequent,

$$\text{PA} \vdash \neg\Box_T\varphi \to \text{Con}_T$$

Thus,

$$\text{PA} \vdash \varphi \to \text{Con}_T$$

For the reverse, note that, by the construction of φ,

$$\text{PA} \vdash \Box_T\varphi \to \Box_T\neg\Box_T\varphi$$

Thus,

$$\text{PA} \vdash \Box_T\varphi \to \Box_T\bot$$

In other words,

$$\text{PA} \vdash \Box_T\varphi \to \neg\text{Con}_T$$

Contraposing,

$$\text{PA} \vdash \text{Con}_T \to \neg\Box_T\varphi$$

By the definition of φ,

$$\text{PA} \vdash \text{Con}_T \to \varphi$$

Putting the two together,

$$\text{PA} \vdash \text{Con}_T \leftrightarrow \varphi$$

► **Theorem 16.17** (Löb's Theorem)**.**
Suppose

$$\text{PA} \vdash \Box_{\text{PA}}\varphi \to \varphi$$

Then, $\text{PA} \vdash \varphi$.

Proof.
(Kripke)
Show $\text{PA} \cup \{\neg\varphi\}$ is inconsistent. By assumption and modus tollens,

$$\text{PA} \cup \{\neg\varphi\} \vdash \neg\Box_{\text{PA}}(\underline{\ulcorner\varphi\urcorner})$$

But then,

$$\text{PA} \cup \{\neg\varphi\} \vdash \neg\Box_{\text{PA}\cup\{\neg\varphi\}}\bot$$

By Gödel's second, however, it now follows that $\text{PA} \cup \{\neg\varphi\}$ is inconsistent. □

■ **Remark.** Note that PA only proves the soundness of its provability predicate with respect to a sentence φ when it 'knows' that φ holds, i.e. when the value of $\Box_{PA}\varphi$ doesn't matter. Given our previous work, this should make a great deal of sense; for all PA can prove, it is inconsistent, and so provability can't be depended on to entail truth generally.

16.4.5 The Consistency-Strength Hierarchy

The impetus for the so-called *Consistency Strength Hierarchy* is simply considering the result of adding successive consistency claims to a theory extending PA; that is, starting with a consistent, computably axiomatizable, deductively-closed T extending PA,

$$T_0 = T$$

$$T_{i+1} = T_i \cup \{Con_{T_i}\}$$

Some reflection shows, however, that if we'd like hierarchy which doesn't immediately collapse, we need another condition on T; after all, sometimes adding a consistency claim as above actually yields inconsistency, e.g. consider starting with $PA \cup \{\neg \mathrm{Con}_{PA}\}$. We thus stipulate that $\mathcal{N} \Vdash T$ or, less constrictively, T is Σ^0_1-sound.

Of course, this hierarchy over T can be extended transfinitely in the typical way:

$$T_\omega = \bigcup_{n<\omega} T_n$$

Note that if the original T is computably axiomatizable, so too is T_n for every n, as well as T_ω; in a similar vein, note that if T is Σ^0_1-sound, then so to is T_n for every $n \in \mathbb{N}$.

The *Consistency Strength Hierarchy* itself is the result of generalizing across the hierarchies generated by different extensions of PA:

Consistency Strength Hierarchy
For consistent, Σ^0_1-sound, deductively-closed theories T, S extending PA, $S \leq_{Con} T$ if and only if for every finite $F \subseteq S$, there is a finite $G \subseteq T$ such that $PA \vdash \mathrm{Con}_G \to \mathrm{Con}_F$. Note that in the finitely axiomatizable case this reduces to $PA \vdash \mathrm{Con}_T \to \mathrm{Con}_S$. The *consistency strength hierarchy* is the hierarchy over consistent, Σ^0_1-sound,deductively-closed theories extending PA generated by the ordering $\leq_{Con}$.

■ **Notation.** Define

$$(\Pi^0_1)_T := \{\varphi \,:\, \varphi \text{ is a } \Pi^0_1 \text{ sentence of } \mathcal{L}_A \text{ and } T \vdash \varphi\}$$

► **Theorem 16.18** (Conssitency Strength is Π^0_1-sentence Inclusion)**.**
For any consistent, Σ^0_1-sound, deductively-closed theories extending PA, $S \leq_{Con} T$ if and only if $(\Pi^0_1)_S \subseteq (\Pi^0_1)_T$

Proof.
($\Leftarrow$)
Let $S \vdash \varphi$ where φ is universal. Then have a finite $F \subseteq S$ such that $F \vdash \varphi$. Let G finite $\subseteq T$ such that $F \leq_{Con} G$. Then, $T \vdash Con_G$. (Use proof of PA proving consistency of all finite subsets, basically the same). So, $T \vdash Con_F$. Then, $T \vdash \varphi$ because $T \vdash \exists x Prov_F(x, \underline{\varphi})$. $T \vdash \exists x Prov_F(x, \underline{\varphi})$, and $PA \vdash \exists x Prov_F(x, \underline{\varphi}) \wedge Con_F) \Rightarrow \varphi$
($\Rightarrow$)
Finish this direction □

Example 14.
Let $PA \subseteq T$. $Rosser_T$ = "for every proof of me, there is a shorter (smaller Godel number) proof of my negation". $Rosser_T$ is universal. $T + Rosser_T \leq_{Con} T$

■ **Remark.** if $T + Con_T$ is Σ^0_1-sound, $T + Con_T \not\leq_{Con} T$ $(\Pi^0_1)_{T+Rosser_T} \not\subseteq (\Pi^0_1)_T$ since $Rosser_T$ is a counterexample

Proof.
Reason in PA. Assume Con_T. We need $Con_{T+Rosser_T}$. Suppose not. Then, $T \vdash \neg Rosser_T$, so $T \vdash Prov_T(\underline{x}, \underline{\neg Rosser_T})$ for some x. For every $y < x$, $T \vdash \neg Prov_T(\underline{y}, \underline{Rosser_T})$. So, $T \vdash Rosser_T$. So T is inconsistent. Contradiction. Thus, $Con_{T+Rosser_T}$. □

16.4.6

If T is Σ^0_1-sound, then T is Π^0_1-sound because it's consistent. The converse does not, however, hold (PA+ $\neg Con_{PA}$). Σ^0_1-soundness is actually equivalent to Π^0_1-soundness (Look at this).

► **Theorem 16.19.**
Peano Arithmetic is finitely axiomatizable

Proof.
Assume PA consistent. Let F finite such that $F \subseteq$ Axioms of PA. Then, $PA \vdash Con_F$; bound the complexity of these sentences (there must be some n such that for all $\varphi \in F$, φ is Σ^0_n-sentence). Note that satisfaction for Σ^0_n is definable over $\mathcal{N}^-$. We now have 'truth' for F.
Say by the wff σ,

$$PA \vdash \varphi \leftrightarrow \sigma(\underline{\varphi})$$

for all sentences φ of F. Then,

$$PA \vdash \sigma(\underline{\varphi})$$

Then, we use the formalized in PA soundness of some proof system. Thus, if $F \vdash PA$, then $F \vdash Con_F$, so F is inconsistent–a contradiction. □

■ **Remark.** Let T be axiomatizable in $\mathcal{L}^{PA}$, $PA \subseteq T$.

i. Suppose $T \vdash \varphi$ where φ is universal. Then, $PA + Con_T \vdash \varphi$ ($T \leq^*_{Con} PA + Con_T$)

ii. ($\neg Con_T$ is sterile) Let $T + \neg Con_T \vdash \varphi$, φ universal. Then, $T \vdash \varphi$.

Proof.
Have $PA + Con_{T+\neg Con_T} \vdash \varphi$ by (i). But, $PA \vdash PA \vdash Con_T \rightarrow Con_{T+\neg Con_T}$ (formalized Godel 2). So, $PA + Con_T \vdash \varphi$. So $T \vdash \varphi$. □

16.5 Nonstandard Models of Peano Arithmetic

16.5.1 PA$^-$ and End-Extensions

Let PA$^-$ denote the theory axiomatized by all the axioms of PA sans the induction schema.

> **Initial Segment / End-Extension**
> For two $\mathcal{L}_A$-models M and N with $M \subseteq N$, N is an end-extension of M (also, M is an initial segment of N)–notated $M \subseteq_e N$–if and only if for all $n \in N$ and $m \in M$, $N \Vdash n < m \Rightarrow n \in M$

► **Theorem 16.20** (All Models of PA^- are End-Extensions of $\mathcal{N}$)**.**
Let $M \Vdash \mathrm{PA}^-$. Then, the map $\mathbb{N} \to \mathrm{dom}(M)$ given by $n \mapsto \underline{n}^M$ is an embedding from the standard model onto an initial segment of M.

► **Proposition 16.3** (PA^- End-Extensions Preserve Δ_1^0 Formulas)**.**
If $M \subseteq_e N$ with $M \Vdash \mathrm{PA}^-$ and $N \Vdash \mathrm{PA}^-$, then $M \preceq_{\Delta_1^0} N$

16.5.2 Overspill and Underspill

Cut
For a model M of PA, a non-empty subset I of M is a *cut* if and only if for all $x \in I$ and all $y \in \mathrm{dom}(M)$, $M \Vdash y < x \Rightarrow y \in I$ and I is closed under the successor function.

► **Lemma 16.2** (Proper Cuts are Not Definable)**.**
If I is a proper cut of a PA model, then I is not definable (even with parameters).

Proof.
Taking such a formula, note that the model makes the antecedent of the induction schema true, and thus the consequent as well–a contradiction. □

► **Corollary** (Overspill)**.**
Let $M \Vdash \mathrm{PA}$ be non-standard and let I be a proper cut of M. Suppose $\bar{a} \in \mathrm{dom}(M)$ and $\varphi(x, \bar{a})$ is an $\mathcal{L}_A$ formula such that $M \Vdash \varphi(b, \bar{a})$ for all $b \in I$. Then, there is $c > I$ in M such that $M \Vdash (\forall x < c)\varphi(x, \bar{a})$

► **Corollary** (Underspill)**.**
Let $\bar{a} \in M$ with $M \Vdash \mathrm{PA}$, $I \subseteq \mathrm{dom}(M)$ a proper cut, and $\varphi(x, \bar{y})$ an $\mathcal{L}_A$ formula. Then,

(i) if $M \Vdash \varphi(c, \bar{a})$ for $c > I$ in M, then $M \Vdash (\forall x \geq b)\varphi(x, \bar{a})$ for some $b \in I$

(ii) if for all $c > I$ in M there exists $x > I$ with $M \Vdash \varphi(x, \bar{a}) \wedge x < c$, then for all $b \in I$ there exists $y \in I$ with $M \Vdash y \geq b \wedge \varphi(y, \bar{a})$.

16.5.3 Cofinal Extensions

Cofinal Extension
Suppose that M, N are PA^- models such that $M \subseteq N$. N is a cofinal extension of M–notated $M \subseteq_{cf} N$–if and only if for every $n \in \mathrm{dom}(N)$, there is an $m \in \mathrm{dom}(M)$ such that $N \Vdash n < m$

► **Theorem 16.21** (Cofinal Extension of PA Model and Δ_0^0 Preserving ⇒ Elementary Extension)**.**
If $M \subseteq_{cf} N$ such that $M \prec_{\Delta_0^0} N$, then if $M \Vdash \text{PA}$, $M \prec N$

► **Corollary** (PA-Model Inclusions are Δ_0^0-Elementary)**.**
Any inclusion between models of PA is Δ_0^0-Elementary

Part IV

Appendices and References

Chapter 17

Orderings

17.1 Linear Orders

$$\forall x[\neg x < x]$$

$$\forall x \forall y[x < y \vee x = y \vee y < x]$$

$$\forall x \forall y \forall z[x < y \wedge y < z \rightarrow x < z]$$

17.1.1 Discrete Linear Orders

There are four kinds of infinite, discrete linear order, all of which generate elementary classes:

(i) Infinite discrete linear orders with a starting point, but no endpoint; e.g., $(\mathbb{N}, <)$.

(ii) Infinite discrete linear orders with an endpoint, but no starting point; e.g., $(\mathbb{N}, <)$ with the reverse ordering.

(iii) Infinite discrete linear orders with both a starting and ending point; e.g. an order of type (i) followed by an order of type (ii).

(iv) Infinite discrete linear orders without endpoints

$$\forall x \exists y[x < y \wedge \neg \exists z[x < z \wedge z < y]]$$

$$\forall x \exists y[y < x \wedge \neg \exists z[z < x \wedge y < z]]$$

In all cases, quantifier elimination can be used to show that the theory is complete.

17.1.2 Dense Linear Orders

There are four kinds of dense linear orders, all of which generate elementary classes:

(i) Dense linear orders with a starting point, but no endpoint; e.g., $([0, 1), <)$.

(ii) Dense linear orders with an endpoint, but no starting point; e.g., $((0, 1], <)$

(iii) Dense linear orders with both a starting and ending point; e.g. $([0, 1], <)$

(iv) Dense linear orders without endpoints; e.g. $(\mathbb{Q}, <)$, $(\mathbb{R}, <)$

$$\forall x \forall y[x < y \rightarrow \exists z[x < z \wedge z < y]]$$

$$\forall x \exists y[y < x]$$

$$\forall x \exists y[x < y]$$

The theory of dense linear orders without endpoints is complete (use Łos-Tarski), $\aleph_0$-categorical (QE), but not categorical in any uncountable cardinal. Quantifier elimination is often a popular and productive proof tactic when working with DLO's; popular examples include

In cases (i)-(iii), quantifier elimination can easily be used to show that the theory is complete.

► **Corollary.**
The theory of partial orders is Σ^0_1-complete.

■ **Fact.** The theory of linear orders is decidable

Example 15.
$(\mathbb{Q}, \leq)$ is $\aleph_0$-saturated; $(\mathbb{Q}, \leq)$ is not ω_1 saturated (look at the type realized by i)

Example 16.
$(\mathbb{R}, \leq)$ is not ω_1-saturated (look at the type containing $n < x$ for all $n \in \mathbb{N}$.

Chapter 18

A Brief Synopsis of Set Theory

In the beginning, there was notation:

$\emptyset$	The empty set; {}
$\{x : \varphi(x)\}$	The set of elements x such that $\varphi(x)$ holds
$X - Y$	The set of all elements of X not in Y
$X \subseteq Y$	The set X is a subset (not necessarily proper) of Y
$\bar{x} = \langle x_1, \ldots, x_n \rangle$	By induction, $\langle x \rangle = x$ $\langle x_1, x_2 \rangle = \{x_1, \{x_1, x_2\}\}$ $\langle x_1, \ldots, x_k \rangle = \langle \langle x_1, \ldots, x_{k-1} \rangle, x_k \rangle$
$X \times Y$	The Cartesian product of X and Y; $\{\langle x, y \rangle : x \in X \text{ and } y \in Y\}$
$X_1 \times \cdots \times X_n$	$((X_1 \times X_2) \cdots \times X_n)$
X^n	By induction, $X^1 = X$ $X^k = X^{k-1} \times X$

Similarly, an *n-ary relation R over a set X* is a subset of X^n, while a *function f* is a binary, functional relation.

$f \restriction A$	The function f restricted to the set A
${}^X Y$	The set of all functions $f : X \to Y$
$\Pi_{i \in I} X_i$	The Cartesian product of a collection of sets $\{X_i : i \in I\}$; defined to be the set of all functions f with domain I such that $f(i) \in X_i$ for all $i \in I$.

Equivalence Relation
For some set X, a relation which is:

i. Reflexive - xRx for every $x \in X$

ii. Symmetric - $xRy \Rightarrow yRx$

iii. Transitive - xRy and $yRz \Rightarrow xRz$

Orderings
For some set X,

i. Partial Ordering - A binary relation R over X which is reflexive, transitive, and anti-symmetric (xRy and $yRx \Rightarrow x = y$)

ii. Simple Ordering - A partial ordering R of X which is connected (for all $x, y \in X$, either xRy or yRx)

iii. Well Ordering - A simple ordering R of X with the property that every nonempty subset of X has an R-least element (for every other y in the subset, xRy)

iv. Strict Well Ordering - A relation R over X which is irreflexive ($\neg xRx$ for every $x \in X$) such that adding $\{\langle x, x\rangle : x \in X\}$ gives a well ordering

Chains
For some set X,

i. X is a *chain* if and only if X is simply ordered by $\subseteq$

ii. X is a *well ordered chain* if and only if X is well ordered by $\subseteq$

18.1 Ordinals

Ordinal
An *ordinal* is a set α such that $\bigcup \alpha \subseteq \alpha$ and α is strictly well ordered by the $\in$ relation

Example 17.
In canonical set theory, the first three ordinals are:

$$0 = \emptyset$$
$$1 = \{\emptyset\}$$
$$2 = \{\emptyset, \{\emptyset\}\}$$

► **Proposition 18.1** (Basic Properties of Ordinals)**.**

i. Every element of an ordinal is itself an ordinal

ii. If α, β are ordinals, then $\alpha \subseteq \beta$ if and only if $\alpha \in \beta$ or $\alpha = \beta$

iii. Every ordinal is a well ordered chain

iv. Every set of ordinals is strictly well ordered by $\in$

v. Every set of ordinals is a well ordered chain

Successor Ordinal
For a given ordinal α, the *successor* of α is the ordinal

$$\alpha + 1 = \alpha \cup \{\alpha\}$$

Any ordinal which can be generated as the successor of another is called a *successor ordinal*.

Limit Ordinal
An ordinal which is not a successor ordinal

Example 18.
The least limit ordinal is ω, the order type of $\mathbb{N}$ under $<$

In addition, we define the *infimum* of a nonempty set X of ordinals is the least element of X, while the *supremum* is the least ordinal which is greater than or equal to every element of X; in fact,

---

► **Proposition 18.2** (Infimum and Supremum)**.**
Let X be a nonempty set of ordinals. Then, $\cap X$ and $\cup X$ are ordinals; in particular, the infimum and supremum respectively of X.

18.1.1 Transfinite Induction

Let φ be any property of ordinals. Assume that for all ordinals β, if φ holds for all $\gamma < \beta$, then $\varphi(\beta)$ holds. Then, φ holds of all ordinals.

18.1.2 Ordinals as Order Types

Order Type
A well ordering R over a set X has *order type* α for some ordinal α if and only if there is a bijection $f : X \to \alpha$ such that $xRy \Rightarrow f(x) \leq f(y)$; f is called an *isomorphism*

► **Proposition 18.3.**
Every well ordering has exactly one order type

18.1.3 Ordinal Arithmetic

Addition

For two ordinals α, β, $\alpha+\beta$ is defined as the ordinal γ such that $\alpha \leq \gamma$ and $\gamma - \alpha$ is isomorphic to the chain β. Transfinite induction shows that not only does $\alpha + \beta$ always exist, it is unique. Intuitively, adding the ordinals α and β can be pictured as the ordering generated by pasting α and β together–in that order.

Example 19.
As the above shows, ordinal addition is not commutative:

$$\omega + 1 \neq \omega$$

$$1 + \omega = \omega$$

Multiplication

For two ordinals α and β, $\alpha * \beta$ is defined as the ordinal given by the lexicographic ordering over the Cartesian product $\alpha \times \beta$. Graphically, this is equivalent to listing β distinct copies of the α ordinal:

Some thought shows that while ordinal multiplication is associative, it–like addition–isn't commutative.

Example 20.

$$\omega * 2 \neq \omega$$

$$2 * \omega = \omega$$

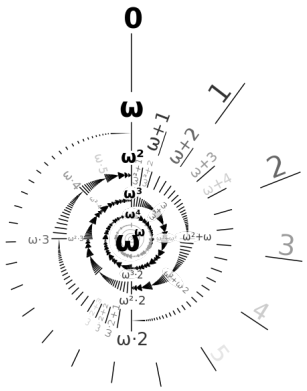
Exponentiation

For better or worse, ordinal exponentiation lacks an easy geometric presentation; for finite exponents, it's easiest to think of exponentiation in terms of multiplication, e.g. $\omega^2 = \omega * \omega$. The general case is given inductively by

$$\alpha^0 = 1$$

$$\alpha^{\beta+1} = (\alpha * \beta) * \beta$$

if δ is limit, then α^δ is the limit of $\alpha * \beta$ for all $\beta < \delta$



18.2 Cardinal Numbers

Cardinal numbers or cardinals are a generalization of the natural numbers used to measure the 'sizes' of sets; this is accomplished via the notion of bijection.

> **Cardinal**
> Two sets are said to have the same *cardinality* if and only if there exists a bijection between them; the *cardinal numbers* is the hierarchy of these 'sizes'. There are two standard ways of defining precisely what the 'cardinal numbers' are; the *von Neumann cardinal assignment* sets each cardinal as the least ordinal with that cardinality. Alternatively, we may identify cardinals as the equivalence class of sets with the given cardinality (note that this is not a set and so cannot be used in ZFC)

The ϑth infinite cardinal is denoted by $\aleph_\vartheta$; the first infinite cardinal is $|\mathbb{N}| = \aleph_0$.

18.2.1 Cardinal Arithmetic

Addition

Given the axiom of choice, every cardinal has a unique successor; set wise, given two disjoint sets X and Y with cardinalities α and β respectively, $\alpha + \beta$ is simply the cardinality of $X \cup Y$. Note that this correspond to standard addition if both α and β are finite, otherwise $\max(\alpha, \beta)$. Unlike ordinal addition, cardinal addition is both associative and commutative.

Multiplication

Cardinal multiplication is similarly well-behaved; given two disjoint sets X and Y with cardinalities α and β respectively, $\alpha\beta$ is simply the cardinality of the Cartesian product $X \times Y$. Finite numbers behave as usual and if both α and β are non-zero with at least one infinite, $\alpha\beta = \max(\alpha, \beta)$. Like cardinal addition, cardinal multiplication maintains both associativity and commutativity.

Exponentiation

Cardinal exponentiation is defined as follows: given two disjoint sets X and Y with cardinalities α and β respectively, α^β is $|X^Y|$ where X^Y is the set of all functions from Y into X, also commonly written as ${}^YX$. Finite numbers again behave as usual and cardinal exponentiation maintains all of the basic exponent manipulations characteristic of the finite setting.

18.2.2 Basic Results

The *successor* of a cardinal α, commonly denoted by α^+, is the least cardinal greater than α; that is, $\aleph_\vartheta^+ = \aleph_{\vartheta+1}$. Note, however, that $\aleph_\vartheta^+ = \aleph_{\vartheta+1} \neq \aleph_\vartheta + 1$. Paralleling the ordinal definition, a cardinal is a *limit cardinal* if and only if it is not the successor of a cardinal.

Strong Limit Cardinals

An important sequence of cardinals and one intimately connected with the notion of a *strong limit cardinal* is *the beths*; the beths are defined recursively by:

$$\beth_0 = \aleph_0$$

$$\beth_{\vartheta+1} = 2^{\beth_\vartheta}$$

and when ϑ is a limit ordinal,

$$\beth_\vartheta = \bigcup_{\beta<\vartheta} \beth_\beta$$

Meanwhile,

Strong Limit Cardinal
A cardinal α is a *strong limit cardinal* if and only if for all cardinals $\beta < \alpha$, $2^\beta < \alpha$

► **Proposition 18.4.**

(i) α is an infinite limit cardinal if and only if there is a limit ordinal ϑ such that $\alpha = \aleph_\vartheta$

(ii) α is an infinite strong limit cardinal if and only if there is a limit ordinal ϑ such that $\alpha = \beth_\vartheta$

(iii) For all ordinals ϑ, $\vartheta \leq \aleph_\vartheta \leq \beth_\vartheta$

(iv) There are arbitrarily large cardinals $\aleph_\vartheta$ such that $|\vartheta| = \aleph_\vartheta = \beth_\vartheta$

As a point of interest, the continuum hypothesis (CH) states that $\beth_1 = \aleph_1$, and the generalized continuum hypothesis (GCH) states that, for all ϑ, $2^{\aleph_\vartheta} = \aleph_{\vartheta+1}$. Thus, GCH implies that for all ϑ, $\beth_\vartheta = \aleph_\vartheta$ and that every limit cardinal is, in fact, a strong limit cardinal.

18.2.3 Regular and Cofinal Cardinals

Cofinality
Let ϑ be a limit ordinal; a set X is *cofinal* in ϑ if and only if $X \subset \vartheta$ and $\vartheta = \bigcup X$. The cofinality of ϑ, notated $\mathrm{cf}(\vartheta)$, is the least cardinal α such that a set of power α is cofinal in ϑ.

Regular and Irregular cardinals Cofinality Universes