Mathematical Logic II

Dag Normann

January 7, 2004

Contents

Introduction

This book is written as the curriculum text for the course *Mathematical Logic II* given at the University of Oslo, Norway. The presumption is that the reader has taken the course *Mathematical Logic*, or is familiar with logic up to the same level. In the *Appendix* we have collected some propositions without proofs. These are precise versions of theorems assumed to be known to the reader.

We will assume that the reader is familiar with first order logic at an elementary level, including the soundness theorem, the deduction theorem and the completeness theorem for countable theories. We will assume knowledge of the Henkin-style proof of this theorem. The theorem of constants will be used extensively. At some occations we will use the full completeness theorem, though a full proof use Zorn's Lemma or comparative means. Since most intended readers also will follow a course in axiomatic set theory, the verification of Zorn's Lemma from the Axiom of Choice is left for that course.

This textbook is essentially consisting of two parts, *Model Theory* and *Computability Theory*. In model theory we mainly restrict ourselves to the classical topics like quantifier elimination and omitting/realising types, but have included a brief introduction to finite model theory. In the computability theory part we use a Kleene-style introduction, and give an introduction to the recursion theorem, c.e. sets, Turing degrees, a basic priority argument and the existence of minimal degrees. We also include a chapter on generalized computability theory.

We will use the following terminology and conventions, all taken from the present textbook [1] in the course *Mathematical Logic*:

- L will be a first order language. Unless specified otherwise, L will be a language with equality.
- c_1, c_2 etc. will be names (or constant symbols) in a formal language.
- f, g, etc. will be function symbols in a formal language. Sometimes it will be convenient to consider a name as a function symbol of arity 0.
- R, P, Q etc. will be relation symbols in a formal language.
- v_1, v_2, . . . is the formal list of variables, but we will mainly use x, y etc. instead.
- t_1, t_2 etc. will denote terms.
- ϕ, ψ, ξ, etc. will denote formulas or sentences.
- A structure $\mathcal{A}$ for a language L will consist of a *domain* A and interpretations $l^{\mathcal{A}}$ for each name, function symbol and relation symbol l (l for 'letter') in L. Whenever we are given structures $\mathcal{A}$ and $\mathcal{B}$, we may use A and B for the domains of these structures without always mentioning it.

- s will denote an assignment function, a function assigning an element of a structure $\mathcal{A}$ to each variable v_i. $\phi[s]$ will be used intuitivly as the formula ϕ where all free variables are replaced by names for elements in A according to the assignment s.
- We will use the terminology $t^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$ and $\phi^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$ for instansiations of terms for free occurences of variables. In the latter case we assume without saying that the terms are substitutable for the variables in question.
- Since the notation above is both hard to read and to write, we will sometimes write $\phi(x_1,\ldots,x_n)$ when ϕ is a formula with at most $x_1,\ldots,x_n$ free. When we use this notation, we may write

  $$\phi(t_1,\ldots,t_n)$$

  instead of

  $$\phi^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}.$$

Each chapter is supplied with a set of exercises at the end, some simple and some hard. The exercises is an integrated part of the text, and at the end the students are assumed to have worked through most of them.

Chapter 5 will consist of just exercises. These will not represent integrated parts of the curriculum, but challenges the eager reader might like to face.

Chapter 1

Classical Model Theory

In model theory we investigate the connection between particular first order theories or first order theories of a special kind and the class of models for that theory. The completeness theorem is one of the most basic tools in model theory, and some of the applications will refer to the actual proof of the completeness theorem. We will also, without hesitation, use the axiom of choice.

1.1 Embeddings and isomorphisms

In this section we will discuss more systematically ways of comparing structures of a 1. order language L. Some of these concepts will be known to the reader, but for the sake of completeness we include all relevant definitions.

Definition 1.1 Let L be a first order language, $\mathcal{A}$ an L-structure with domain A.
A *substructure* $\mathcal{B}$ with domain B will be an L-structure such that

1. $B \subseteq A$.
2. $c^{\mathcal{B}} = c^{\mathcal{A}}$ whenever c is a name (constant symbol) in L.
3. $f^{\mathcal{B}}(a_1, \ldots, a_n) = f^{\mathcal{A}}(a_1, \ldots, a_n)$ whenever f is a function symbol in L of arity n and $a_1, \ldots, a_n \in B$.
4. $R^{\mathcal{B}} = R^{\mathcal{A}} \cap B^n$.

Definition 1.2 A first order theory T is *open* if all non-logical axioms are open, i.e. has no quantifiers.

Lemma 1.1 *Let $\mathcal{A}$ be an L-structure.*

a) Let $B \subseteq A$. Then B is the domain of a substructure $\mathcal{B}$ of $\mathcal{A}$ if and only if B is closed under all interpretations $f^{\mathcal{A}}$ where f is a function symbol (or a constant) in L.

b) *Let $\mathcal{B}$ be a substructure of $\mathcal{A}$. Let ϕ be an L-formula and s an assignment over $\mathcal{B}$ (and then also over $\mathcal{A}$.) Then $\phi[s]$ is true in $\mathcal{B}$ if and only if it is true in $\mathcal{A}$.*

Proof
The proof of a) is trivial and the proof of b) is trivial by induction on the sub-formula relation.

Given a structure $\mathcal{A}$, it is sufficient to consider the class of substructures for most purposes where this is relevant. However, sometimes we want to go the other way, we might like to extend a structure. This is done e.g. in moving from the natural numbers to the integers, further on to the rationals and maybe continuing to the algebraic numbers, the real nunbers or the complex numbers. Whenever convenient we may consider a natural number as a complex number, though in the set-theoretical model-building there is a far way to go. In the extreme, a natural number will be a finite set totally ordered by the $\in$-relation, an integer will be an equivalence class of pairs of natural numbers, a rational number will be an equivalence class of pairs of integers, a real will be a Dedekind cut of rational numbers (or even worse, an equivalence class of Cauchy sequences of rational numbers) and a complex number will be a pair of real numbers. Of course there is no mathematical gain in thinking of complex numbers as such monsters, it is easier to assume that $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{R} \subset \mathbb{C}$ and just base our mathematics on the intuition about these sets that we have.

When we in the context of a logic course want to extend a structure, it is standard procedure to use the completeness theorem in such a way that we ensure that a copy of the original structure will be a substructure. We have to do some set-theoretical hocus pocus in order to obtain an actual extension.

We will now develop the concepts needed in order to make this precise:

Definition 1.3 let L be a first order language and let $\mathcal{A}$ and $\mathcal{B}$ be L-structures.

a) An *embedding* from $\mathcal{B}$ to $\mathcal{A}$ is a map $\pi : B \to A$ that is one to one and such that

$\pi(c^{\mathcal{B}}) = c^{\mathcal{A}}$ for each name c.

$\pi(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(\pi(a_1), \ldots, \pi(a_n))$ for all $a_1, \ldots, a_n \in B$.

$R^{\mathcal{B}}(a_1, \ldots, a_n) \Leftrightarrow R^{\mathcal{A}}(\pi(a_1), \ldots, \pi(a_n))$ for all $a_1, \ldots, a_n \in B$.

b) An *isomorphism* will be an embedding that is onto.

Examples 1.1 a) If we consider $\langle \mathbb{R}, 0, +, < \rangle$ and $\langle \mathbb{R}^+, 1, \cdot, < \rangle$ as two structures for the language with one name, one binary function and one binary relation, then $\pi(x) = e^x$ is an isomorphism.

b) If we consider the reals as sets of Dedekind cuts of rationals with the inherited algebraic structure, then $\pi(q) = \{r \in \mathbb{Q} \ ; \ r < q\}$ will be an embedding of $\mathbb{Q}$ into $\mathbb{R}$.

c) If $\mathcal{Z} = \langle \mathbb{Z}, +, \cdot, 0, 1\rangle$ is a structure for the language of ring theory, then the set of non-negative integers will be a substructure (but not a subring) of $\mathcal{Z}$.

There are two traditions for defining the interpretation of a formula over a structure. One is using assignments as we did in the introduction. Then we define the truth value of any formula relative to a given assignment. The other tradition is to extend the language with names for each element in the structure, and then only interpret the sentences in this extended language. These approaches are equivalent, but it turns out that the latter approach is more easy to use for many of the purposes in model theory.

Definition 1.4 Let L be a first order language, $\mathcal{A}$ an L-structure.

a) By $L(\mathcal{A})$ we mean the language L where we add one new constant symbol c_a for each $a \in A$.

b) As a default, we will consider $\mathcal{A}$ as an $L(\mathcal{A})$-structure via the interpretation $c_a^{\mathcal{A}} = a$.

See Exercise 1.3 for the tedious, but simple observations that have to be made. Recall that a *literal* is a formula that is either an atomic formula or the negation of an atomic formula.

Definition 1.5 Let L be a first order language and let $\mathcal{A}$ be an L structure. By the *diagram* of $\mathcal{A}$, $\mathcal{D}(\mathcal{A})$, we mean the set of variable-free literals in $L(\mathcal{A})$ that are true in $\mathcal{A}$.

The diagram will play an important part when we want to show that we have extensions of a structure $\mathcal{A}$ with some desired properties; we just have to show that the diagram of $\mathcal{A}$ is consistent with the desired properties:

Lemma 1.2 *Let L be a first order language, $\mathcal{A}$ and $\mathcal{B}$ L-structures. Then the following are equivalent:*

i) There is an embedding π of $\mathcal{A}$ into $\mathcal{B}$.

ii) We may give an interpretation of the extra constants c_a from $L(\mathcal{A})$ in $\mathcal{B}$ such that $\mathcal{B}$ becomes a model for $\mathcal{D}(\mathcal{A})$.

Proof
Assume i). Let $(c_a)^{\mathcal{B}} = \pi(a)$. With this interpretation, ii) is verified.
Assume ii). Let $\pi(a) = (c_a)^{\mathcal{B}}$. Since L is a language with equality, and $\mathcal{B}$ is a model for the diagram of $\mathcal{A}$, π will be an embedding.

It will be a matter of taste if we talk about extensions of structures or of embeddings into other structures, see Exercise 1.4. Our first result is a typical modeltheoretical one, we connect the logical form of the axioms in a theory with the relations between models and their substructures. Recall that two first order theories are equivalent if they have the same theorems. (A consequence will be that they are based on the same language; why?)

Theorem 1.1 (Łos-Tarski)
Let T be a first order theory over a language L. Then the following are equivalent:

1. *If $\mathcal{B}$ is a model for T and $\mathcal{A}$ is a substructure for $\mathcal{B}$, then $\mathcal{A}$ is a model for T.*
2. *T is equivalent to an open theory T'.*

Proof
By the completeness theorem, two theories will be equivalent if and only if they have the same models. Thus 2. $\Rightarrow$ 1. is already proved.
Now assume that 1. holds. Let the non-logical axioms in T' be the open theorems of t . Then T' is an open theory, and all models for T will be models for T'. In order to prove that T and T' are equivalent, we must prove the converse. So, let $\mathcal{A}$ be a model for T'. In order to prove that T' is a model for T, consider the theory

$$T^* = \mathcal{D}(\mathcal{A}) \cup T.$$

Claim
T^* is consistent.

Proof
Assume not. Then there are $\phi_1, \ldots, \phi_n$ in the diagram of $\mathcal{A}$ such that $T \cup \{\phi_1, \ldots, \phi_n\}$ is inconsistent, or alternatively, that $T \vdash \neg(\phi_1 \wedge \cdots \wedge \phi_n)$.
Each ϕ_i will be an instance of some literal ψ_i in L, and we may let the same name c_a be substituted for the same variable z. Thus there are variables $z_1, \ldots, z_m$, new names $c_{a_1}, \ldots, c_{a_m}$ and literals $\psi_1, \ldots, \psi_n$ in L such that

$$\phi_i = (\psi_i)^{z_i, \ldots, z_m}_{c_{a_1}, \ldots, c_{a_m}}$$

for each $i \leq n$.
The names c_{a_j} will not occur in the non-logical axioms in T, so by the theorem of constants, $T \vdash \neg(\psi_1 \wedge \cdots \wedge \psi_n)$.
However, each ϕ_i is a true literal, so $\phi_1 \wedge \cdots \wedge \phi_n$ is a valid instance of $\psi_1 \wedge \cdots \wedge \psi_n$. This contradicts that $\neg(\psi_1 \wedge \cdots \wedge \psi_n)$ is valid in $\mathcal{A}$, and the claim is proved.

Let $\mathcal{B}$ be a model for T^*. There will be an embedding of $\mathcal{A}$ into $\mathcal{B}$, and by Exercise 1.2, $\mathcal{A}$ is a model for T.
This ends the proof of the theorem

Sometimes in model theory we will construct a structure by piecewise extensions, quite often using the completeness theorem at each step. In order to handle such constructions we need the concept of *direct limit*, a concept you find in other parts of mathematics as well.

Definition 1.6 Let $\langle I, < \rangle$ be a partial ordering.
The ordering is called *directed* if

$$\forall i \in I \forall j \in I \exists k \in I (i < k \wedge j < k).$$

A directed ordering is a generalisation of an increasing sequence.

Definition 1.7 Let L be a first order language.

a) A *directed system of L-structures* will consist of

1. A directed ordering $\langle I, < \rangle$
2. An L-structure $\mathcal{A}_i$ for each $i \in I$
3. An embedding $\pi_{ij} : \mathcal{A}_i \to \mathcal{A}_j$ whenever $i \leq j$ in I

such that π_{ii} is the identity function on each $\mathcal{A}_i$ and such that whenever $i \leq j \leq k$, then $\pi_{ik} = \pi_{jk} \circ \pi_{ij}$ where $\circ$ denotes composition.

b) Using the notation from a), a *directed limit* of the system will be an L-structure $\mathcal{A}$ and embeddings $\pi_i : \mathcal{A}_i \to \mathcal{A}$ for each $i \in I$ such that

- If $i \leq j$, then $\pi_i = \pi_j \circ \pi_{ij}$.
- If $\mathcal{B}$ is an L-structure and η_i is an embedding from $\mathcal{A}_i$ to $\mathcal{B}$ for each $i \in I$ such that $\eta_i = \eta_j \circ \pi_{ij}$ whenever $i \leq j$, then there is a unique embedding $\eta : \mathcal{A} \to \mathcal{B}$ such that $\eta_i = \eta \circ \pi_i$ for each $i \in I$.

Remark 1.1 The concept of a directded limit is more general. In a category theoretical context it is often known as a *colimit*.

Theorem 1.2 *Each directed system will have a directed limit, and this is unique up to isomorphisms.*

Proof
The reader should verify for her/himself that if $\mathcal{A}$ and $\mathcal{B}$ both satisfies the requirements of a directed limit, then $\mathcal{A}$ and $\mathcal{B}$ are isomorphic.
Let X be the set of ordered pairs (i, a) such that $a \in A_i$ where A_i is the domain of $\mathcal{A}_i$.
We let $(i, a) \approx (j, b)$ if for some $k \in I$ we have that $i \leq k$, $j \leq k$ and $\pi_{ik}(a) = \pi_{jk}(b)$.
This definition does not depend on the choice of k, and $\approx$ will be an equivalence relation.
Let A be the set of equivalence classes $[(a, i)]$, and let $\pi_i(a) = [(a, i)]$.
Whenever $C_1, \ldots, C_n$ are equivalence classes, we may find an $i \in I$ and $a_1, \ldots, a_n \in A_i$ such that $C_j = [(i, a_j)]$ for each $j \leq n$. Here we use that $\langle I, < \rangle$ is a directed ordering.
We let

$$f^{\mathcal{A}}(C_1, \ldots, C_n) = [(i, f^{\mathcal{A}_i}(a_1, \ldots, a_n))]$$

and

$$R^{\mathcal{A}}(C_1, \ldots, C_n) \Leftrightarrow R^{\mathcal{A}_i}(a_1, \ldots, a_n)$$

whenever f and R are n-ary.
This definition does not depend on i.
The details left out and the rest of the proof is left for the reader as Exercise 1.5.

1.2 Elementary embeddings

If two structures $\mathcal{A}$ and $\mathcal{B}$ for the same language L are isomorphic, they will have the same logical properties. In fact, this must be true as long as a logical property reflects the structure itself and not the underlying set-theoretical representation of the domains or of other parts of the interpretations. In paricular, they will share the same properties expressible in higher order logic.
Concepts like *finite*, *well ordering*, *countable* and *complete* are not expressible in first order logic. In order to make a discussion of this precise, we need a new concept:

Definition 1.8 Let L be a first order language, and let $\mathcal{A}$ and $\mathcal{B}$ be two L-structures.
We say that $\mathcal{A}$ and $\mathcal{B}$ are *elementary equivalent* if

$$\mathcal{A} \models \phi \Leftrightarrow \mathcal{B} \models \phi$$

whenever ϕ is a sentence in L.
Since it is useful to consider a structure as the same structure even if we interpret several additional names, we refer to L in the notation for elementary equivalence. We write

$$\mathcal{A} \equiv_L \mathcal{B}$$

when $\mathcal{A}$ and $\mathcal{B}$ are elementary equivalent with respect to sentences in L.

The Norwegian Logician *Thoralv Skolem* demonstrated the limitations of the ongoing program formalising mathematics.

Theorem 1.3 (Skolem)
Let L be the first order language of number theory, and let

$$\mathcal{N} = \langle \mathbb{N}, 0, 1, +, \cdot, exp, < \rangle$$

be the standard model.
Then there is another L-structure $\mathcal{N}'$ that is elementary equivalent to $\mathcal{N}$, but not isomorphic.

Proof
This is a consequence of the compactness theorem. Let c be a new name, let T be the set of sentences in L valid in $\mathcal{N}$, and let T^* be T extended by the axioms $c_k < c$ for each k, where c_k is the numeral in L denoting the number k.
Each finite subtheory of T^* may use $\mathcal{N}$ as a model, so by the compactness theorem, there is a model $\mathcal{N}'$ for T^*. This model will be elementary equivalent to $\mathcal{N}$ because both are models for T, but they are clearly not isomorphic.

Remark 1.2 The completeness theorem and the compactness theorem were not available to Skolem. His argument used another form of model theoretical construction, something we might call a *reduced ultraproduct* construction.

By a similar argument we may show that the mathematically important concept of a well ordering is not definable in first order logic.

Definition 1.9 Let $\langle X, < \rangle$ be a total ordering. $\langle X, < \rangle$ is a *well ordering* if every nonempty subset $Y \subseteq X$ has a least element.

Theorem 1.4 *There is no first order theory T with the class of well orderings as its models.*

Proof
Any such theory must be over the language with one binary relation symbol, we may as well denote it by $<$, and a sentence ϕ in L will be a theorem in T if and only if ϕ is true for all well orderings. So let T be the theory with all sentences ϕ true for all well orderings as its non-logical axioms. We will show that T has a model that is not a well ordering.

Extend L with new names c_k for each natural number k and let T^* be T extended with all axioms $c_{k+1} < c_k$ for $k \in \mathbb{N}$.
Each finite subtheory T_0 of T^* will have a model, since there are arbitrarily long finite well orderings. By the compactness theorem, T^* has a model

$$\mathcal{A} = \langle A, <^{\mathcal{A}}, \{c_k^{\mathcal{A}}\}_{k \in \mathbb{N}} \rangle.$$

Clearly the set $\{c_k^{\mathcal{A}} \ ; \ k \in \mathbb{N}\}$ has no least element, so $\langle A, <^{\mathcal{A}} \rangle$ is not a well ordering.
However

$$\langle A, <^{\mathcal{A}} \rangle \models T,$$

so this structure has all the first order properties shared by all well orderings.

Definition 1.10 A first order theory T over a language L is *complete* if T is consistent and $T \vdash \phi$ or $T \vdash \neg\phi$ for each sentence ϕ in L

The following is a direct consequence of the completeness theorem:

Lemma 1.3 *Let T be a first order theory over a language L. The following are equivalent:*

1. *T is complete.*
2. *$\mathcal{A} \equiv_L \mathcal{B}$ for all models $\mathcal{A}$ and $\mathcal{B}$ for T.*

If L is a first order language, $\mathcal{A}$ and $\mathcal{B}$ are L-structures, and π is an embedding from $\mathcal{A}$ to $\mathcal{B}$, we may view $\mathcal{B}$ as a structure for $L(\mathcal{A})$ using the interpretation

$$(c_a)^{\mathcal{B}} = \pi(a).$$

Definition 1.11 Let L be a first order language, and let $\mathcal{A}$ and $\mathcal{B}$ be two L-structures.

a) Assume that $\mathcal{A}$ is a substructure of $\mathcal{B}$. We say that $\mathcal{A}$ is *an elementary substructure* of $\mathcal{B}$ if $\mathcal{A} \equiv_{L(\mathcal{A})} \mathcal{B}$.

b) Assume that π is an embedding from $\mathcal{A}$ to $\mathcal{B}$ and view $\mathcal{B}$ as a structure for $L(\mathcal{A})$ as above.
We call π an *elementary embedding* if $\mathcal{A} \equiv_{L(\mathcal{A})} \mathcal{B}$.

Definition 1.12 Let L be a first order language, $\mathcal{A}$ an L-structure. The *theory of* $\mathcal{A}$, $Th(\mathcal{A})$, will have the sentences in $L(\mathcal{A})$ that are true in $\mathcal{A}$ as the non-logical axioms.

We have the following observation:

Lemma 1.4 *Let L be a first order language, $\mathcal{A}$ and $\mathcal{B}$ be two L-structures. The following are equivalent:*

1. *There is an elementary embedding π from $\mathcal{A}$ to $\mathcal{B}$.*
2. *There is an interpretation of each new constant in $L(\mathcal{A})$ as an element in $\mathcal{B}$ such that $\mathcal{B}$ becomes a model for $Th(\mathcal{A})$.*

Proof
Both directions are based on establishing the equality

$$\pi(a) = (c_a)^{\mathcal{B}}.$$

The details are left for the reader.

Theorem 1.5 *Let L be a first order theory, $\mathcal{A}$ and $\mathcal{B}$ two L-structures. Then the following are equivalent:*

1. *$\mathcal{A} \equiv_L \mathcal{B}$.*
2. *There is an L-structure $\mathcal{C}$ with elementary embeddings $\pi : \mathcal{A} \to \mathcal{C}$ and $\eta : \mathcal{B} \to \mathcal{C}$.*

Proof
2. $\Rightarrow$ 1. is trivial, so assume that $\mathcal{A}$ and $\mathcal{B}$ are elementary equivalent. We will extend L by adding names c_a for each $a \in A$ and a disjoint set of names d_b for $b \in B$.
Let $T = Th(\mathcal{A}) \cup Th(\mathcal{B})$. We will prove that T is consistent.
Assume not. We will use that the set of axioms in $Th(\mathcal{A})$ and the set of axioms in $Th(\mathcal{B})$ are closed under $\wedge$.
There are sentences ϕ in $Th(\mathcal{A})$ and ψ in $Th(\mathcal{B})$ such that $\vdash \neg(\phi \wedge \psi)$. Let $x_1, \ldots, x_n, y_1, \ldots, y_m$ be fresh and distinct variables and let ϕ' and ψ' be L-formulas such that ϕ is of the form

$$(\phi')^{x_1,\ldots,x_n}_{c_{a_1},\ldots,c_{a_n}}$$

and ψ is of the form

$$(\psi')^{y_1,\ldots,y_m}_{d_{b_1},\ldots,d_{b_m}}$$

.

By the theorem of constants

$$\vdash \neg(\phi' \wedge \psi'),$$

and using propositional logic and rules for quantifiers

$$\vdash \exists x_1 \cdots \exists x_n \phi' \to \forall y_1 \cdots \forall y_m \neg \psi'.$$

Now $\exists x_1 \cdots \exists x_n \phi'$ will hold in $\mathcal{A}$ since $a_1, \ldots, a_n$ makes ϕ' true. Since $\mathcal{A}$ and $\mathcal{B}$ are elementary equivalent, $\exists x_1 \cdots \exists x_n \phi'$ will be true in $\mathcal{B}$ as well.
As a consequence we must have that $\forall y_1 \cdots \forall y_m \neg \psi'$ is true in $\mathcal{B}$. But this is impossible, since there is an instance that makes ψ' true in $\mathcal{B}$, namely ψ. Thus we have observed a contradiction.

The assumption was that T is not consistent, and our conclusion will be that T is consistent and has a model $\mathcal{C}$. By Lemma 1.4 we see that 2. holds. This ends the proof of the theorem.

In Exercise 1.6 we give an application of this theorem.

In section 1.1 we discussed directed limits of directed systems. Such systems where all embeddings are elementary are of a particular interest:

Definition 1.13 Let L be a first order language, $\langle I, < \rangle$ be a directed set and let $\langle \{\mathcal{A}_i\}_{i \in I}, \{\pi_{ij}\}_{i<j} \rangle$ be a directed system of L-structures.
We call the system *elementary* if each π_{ij} is an elementary embedding.

The limit embeddings of an elementary system will be elementary, see Exercise 1.7.

We will prove two important theorems about elementary extensions, the so called Löwenheim-Skolem theorems. The theorems say that there is no way first order logic can distinguish between infinities that dominate the size of the language in question. One consequence will be the *Skolem's paradox*;

there is an elementary countable substructure of the reals.

This is not a real paradox, it only demonstrates that first order logic is not adequate for proving, or even stating, that $\mathbb{R}$ is uncountable.

Theorem 1.6 *Let $\mathcal{A}$ be an infinite first order structure over the language L. Let X be any set.*
Then there is an elementary extension $\mathcal{B}$ of $\mathcal{A}$ and an injective map $\delta : X \to B$. (We can find elementary extensions of $\mathcal{A}$ of arbitrarily large cardinality.)

Proof
Let L^* be $L(\mathcal{A})$ extended with a name d_b for each element $b \in X$. Let T be the theory $Th(\mathcal{A})$ extended with the axioms $d_b \neq d_c$ whenever b and c are distinct

elements of X.
Since $\mathcal{A}$ is infinite, $\mathcal{A}$ can be viewed as a model for each finite subtheory of T^*, so T^* has a model $\mathcal{B}$. Using Exercise 1.4 adjusted to elementary embeddings, we see that we may let $\mathcal{B}$ be an extension of $\mathcal{A}$. Let $\delta(b) = (d_b)^{\mathcal{B}}$. δ will be injective, and the theorem is proved.

This was the easy upwards Löwenheim-Skolem theorem. We will now face the downwards Löwenheim-Skolem theorem. Since we will not bother to get involved in too much cardinality arithmetics, we restrict ourselves to the countable case, which in any case is the most important and best known case.

Theorem 1.7 *Let L be a countable first order language, and let $\mathcal{A}$ be an L-structure.*
Then $\mathcal{A}$ has a countable elementary substructure.

Proof
Let ϕ be a formula in L, $a_1, \ldots, a_n$ elements of A. In order to improve the readability, we will let

$$\phi(x_1, \ldots, x_n)$$

mean that all free variables in ϕ are among $x_1, \ldots, x_n$, and we will write

$$\phi(a_1, \ldots, a_n)$$

instead of

$$\phi^{x_1,\ldots,x_n}_{c_{a_1},\ldots,c_{a_n}}.$$

Let $\phi(y, x_1, \ldots, x_n)$ be a formula in L. A *Skolem Function* for ϕ is a function $h : A^n \to A$ such that whenever $a_1, \ldots, a_n$ are in A then

$$\phi(h(a_1, \ldots, a_n), a_1, \ldots, a_n) \text{ is true in } \mathcal{A}$$

whenever

$$\text{there is some } b \in A \text{ such that } \phi(b, a_1, \ldots, a_n) \text{ is true in } \mathcal{A}.$$

For each such formula ϕ, we select one such Skolem function h_ϕ. If $n = 0$, h_ϕ will be a function of no variables, which will be just an element of A. In particular, $h_{y=y} \in A$.
If f is a function symbol in L, we find $f^{\mathcal{A}}$ as one of the Skolem functions: Let ϕ be the formula $y = f(x_1, \ldots, x_n)$. Then $f^{\mathcal{A}} = h_\phi$
Since L is countable, the set of Skolem functions will be countable.
Let B_0 be the empty set, and by recursion, let

$$B_{k+1} = \{h(a_1, \ldots, a_n)\ ;\ h \text{ is a Skolem function and } a_1, \ldots, a_n \in B_k\}.$$

By induction on k we see that each B_k will be countable, and that $B_k \subseteq B_{k+1}$. Let B be the union of the B_k's. Then B is a countable subset of A, and by construction, B is closed under all the Skolem functions. By Lemma 1.1, B is

the domain of a substructure $\mathcal{B}$ of $\mathcal{A}$. It remains to prove that $\mathcal{B}$ is an elementary substructure of $\mathcal{A}$, which amounts to prove that whenever $a_1, \ldots, a_n$ are in B, then $\phi(a_1, \ldots a_n)$ is true in $\mathcal{B}$ if and only if it is true in $\mathcal{A}$.
This is proved by induction on the complexity of ϕ.
If ϕ is atomic, the equivalence follows from the fact that $\mathcal{B}$ is a substructure of $\mathcal{A}$.
If $\phi = \neg\psi$ or $\phi = \psi_1 \vee \psi_2$, the induction step is trivial.
If $\phi(a_1, \ldots, a_n) = \exists y \psi(y, a_1, \ldots a_n)$, and $\phi(a_1, \ldots, a_n)$ is true in $\mathcal{B}$ via some b, then, by the induction hypothesis, it will be true in $\mathcal{A}$ via the same b.
If on the other hand $\phi(a_1, \ldots, a_n)$ is true in $\mathcal{A}$ via some b, it is true in $\mathcal{A}$ via $h_\psi(a_1, \ldots, a_n)$.
Since $h_\psi(a_1, \ldots, a_n) \in B$, we may use the induction hypothesis and see that $\phi(a_1, \ldots, a_n)$ is true in $\mathcal{B}$ via $h_\psi(a_1, \ldots, a_n)$.
This ends the proof.

1.3 Complete Theories

In the previous section, we defined a consistent first order theory to be complete if all sentences of the language can be proved or disproved. In this section we will look for criteria for a theory to be complete.

1.3.1 Categorical theories

A first order theory is *categorical* if all models are isomorphic. If a theory is categorical, it is a consequence of the uppwards Löwenheim-Skolem theorem that all models are finite. Thus this concept is of a rather limited interest. The following is more interesting:

Definition 1.14 Let L be a countable first order language, and T a theory over L. T is *ω-categorical* if all countable models are isomorphic.

It is easy the see that an ω-categorical theory must be complete, see Exercise 1.8. In Exercise 1.12 we see that the converse is not true.

Example 1.2 Let DO be the first order theory over the language L with equality and one binary relation symbol $<$, and with the following axioms:

DO-1 $\neg(x < x)$

DO-2 $x < y \wedge y < z \rightarrow x < z$

DO-3 $x < y \vee y < x \vee x = y$

DO-4 $x < y \rightarrow \exists u \exists v \exists w (u < x \wedge x < v \wedge v < y \wedge y < w)$

The first three axioms tells us that $<$ is a total ordering, while DO-4 implies that there is no largest element, no least elements, and that the elements in the ordering are densly ordered. *DO* stands for *dense ordering.*

We claim that DO is ω-categorical. We give an outline of the proof, and leave the details as Exercise 1.9
Let $\langle A, <_A \rangle$ and $\langle B, <_B \rangle$ be two countable models for DO. A *finite partial isomorphism* will be an order preserving map $p : K \rightarrow B$ where $K \subseteq A$ is finite. The set of finite partial isomorphisms have the following extension properties:

- If p is a finite partial isomorphism and $a \in A$, then p can be extended to a finite partial isomorphism q defined on a.
- If p is a finite partial isomorphism and $b \in B$, then p can be extended to a finite partial isomorphism q with b in its range.

Given enumerations $A = \{a_n \ ; \ n \in \mathbb{N}\}$ and $B = \{b_n \ ; \ n \in \mathbb{N}\}$ we can construct an increasing sequence of finite partial ismorphisms $\{p_n\}_{n \in \mathbb{N}}$ securing that $p_{2n}(a_n)$ is defined and that b_n is in the range of p_{2n+1}. The limit of these finite partial isomorphisms will be an isomorphism between the two structures.

We will discuss ω-categoricity in more depth in the paragraph on element types and in the chapter on finite model theory. The method of proof is for obvious reasons called a *back-and-forth construction.* Sometimes in the litterature, it is called a *zig-zag-construction.*

1.3.2 Elimination of Quantifiers

One of the key success stories of model theory is that of proving completeness of a theory via elimination of quantifiers, and thereby proving theorems about algebraic theories og genuine interest. Our key example will be field theory, but the main application is the theory of real closed fields. We have used traditional texts on model theory as sources for our exposition in this and later sections on model theory, in particular Sacks [2]. We will assume that the reader is familiar with field theory, but give a brief introduction for the sake of completeness.

Field Theory

The language of field theory that we will use, will consist of three constants, 0, 1 and -1 and two binary function symbols $+$ and $\cdot$. As logicians, we should make a clear distinction between these symbols and the symbols we use for the particular interpretations, but unless forced by the cirumstances, we will not do so.
We will base field theory on the following list of axioms:

F 01 $(x + y) + z = x + (y + z)$

F 02 $x + 0 = x$

F 03 $x + (-1 \cdot x) = 0$

F 05 $(x \cdot y) \cdot z = x \cdot (y \cdot z)$

F 06 $x \cdot 1 = x$

F 07 $x \neq 0 \to \exists y(x \cdot y = 1)$

F 08 $x \cdot y = y \cdot x$

F 09 $x \cdot (y + z) = x \cdot y + x \cdot z$

F 10 $0 \neq 1$

We will use the standard algebraic notation like e.g. xy and $-x$ for $x \cdot y$ and $(-1) \cdot y$ resp.
A *field* will be a model for this theory. Each natural number n will have an interpretation in a field $\mathcal{F}$. If $\mathcal{F}$ is a field and p is a prime, we say that the field have *characteristic* p if

$$\mathcal{F} \models p = 0.$$

$\mathbb{Z}_p = \{0, \ldots, p-1\}$ where addition and multiplication are carried out modulo p (where p is a prime) is an example of a field of characteristic p. If the reader is not familiar with this fact she/he should verify it by her/himself. A field will be of characteristic 0 if it is not of characteristic p for any prime p (a non-prime cannot be the characteristic of a field). The distinction between finite and infinite characteristics cannot be made in first order logic, neither can the distinction between finite and infinite fields. See Exercise 1.10 for more precise statements about this.

Other familiar examples of fields are $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$ with the standard algebraic structures. All these fields will have characteristic 0.

Definition 1.15 A field $\mathcal{F}$ is *algebraically closed* if each polynomial P of one variable and degree > 0 will have a root. This can be expressed by the infinite set of axioms

$$y_n \neq 0 \to \exists x(y_n x^n + y_{n-1}x^{n-1} + \cdots + y_0 = 0),$$

where x^n has its usual mathematical meaning, and $n \geq 1$.
We let ACF denote the theory of algebraic closed fields, and $ACF(p)$ or $ACF(0)$ denote the extension where the characteristic of the field is specified.

Lemma 1.5 *An algebraically closed field is never finite.*

Proof
Let $a_1, \ldots, a_n$ be elements of an algebraically closed field, and let

$$P(x) = (x - a_1) \cdots (x - a_n) - 1.$$

If $P(x) = 0$, then $x \neq a_1, \ldots, a_n$.

The isomorphisn property

Definition 1.16 Let T be a first order theory.
We say that T satisfies the *isomorphism property* if whenever $\mathcal{A}$ and $\mathcal{B}$ are models for T, $\mathcal{A}_0$ and $\mathcal{B}_0$ are substructures of $\mathcal{A}$ and $\mathcal{B}$ resp. and π_0 is an isomorphism from $\mathcal{A}_0$ to $\mathcal{B}_0$, then π_0 can be extended to an isomorphism between sub*models* $\mathcal{A}_1$ and $\mathcal{B}_1$ of $\mathcal{A}$ and $\mathcal{B}$.

Example 1.3 The theory DO satisfies the isomorphism property.

If we have two dense orderings and two isomorphic suborderings, either these suborderings are themselves dense, or they will have isomorphic gap structures. Using the proof of the ω-categoricity of DO we see that each pair of gaps can be filled by copies of $\mathbb{Q}$ taken from the given dense orderings. The result will be isomorphic submodels. We leave the details for the reader.

Example 1.4 The theory of fields satisfy the isomorphism property

A substructure of a field will contain 0, 1 and -1, and will be closed under summation and multiplication, so a substructure is actually a subring. So the task is to show that isomorphic subrings can be extended to isomorphic subfields inside the given fields. This is basic algebra, we have isomorphic "quotient-structures" of the two rings, and interpreting each quotioent in the fields will give isomorphic subfields.

We also use standard algebra to prove

Lemma 1.6 *The theory ACF will satisfy the isomorphism property*

Proof
Since this book is not a textbook in algebra, we will not give all the details.
Given a field $\mathcal{F}_0$ and an irreducible polynomial $P(x)$ over that field we may extend the field by a root of that polynomial in a purely algebraic way, by considering the field of all polynomials $Q(x)$ modulo $P(x)$. If $\mathcal{F}_0$ is a subfield of an algebraically closed field $\mathcal{F}$, and $a \in \mathcal{F}$ is a root of P, then the map $Q(x) \mapsto Q(a)$ is an isomorphism between the formal extension and a larger subfield of $\mathcal{F}$. (Euclied's algorithm for the largest common divisor applied to polynomial division is central here).
The algebraic closure of a field can be described as the result of a transfinite sequence of such formal extensions, so the algebraic closures of isomorphic fields within algebraically closed fields will be isomorphic.

In Exercise 1.11 we will give an alternative proof.

Lemma 1.7 *Let T be a first order theory over a language L and assume that T has the isomorphism property.*
Let T' be obtained from T by adding new constants to L, but no new non-logical axioms. Then T' has the isomorphism property.

The proof is trivial and is left for the reader.

The submodel property

Definition 1.17 Let ϕ be a formula in a first order language L. We call ϕ *simple* if ϕ is of the form $\exists\psi$ where ψ is open.

Definition 1.18 Let L be a first order language, T a theory over the language L.
We say that T has the *submodel property* if whenever $\mathcal{B}$ is a model of T, $\mathcal{A}$ is a submodel of $\mathcal{B}$ and ϕ is a simple sentence in $L(\mathcal{A})$, then ϕ is true in $\mathcal{A}$ if and only if it is true in $\mathcal{B}$.

Example 1.5 The theory *DO* satisfies the submodel property.

Dropping formalities, let ϕ be the sentence $\exists x\psi(x, a_1, \ldots, a_n)$. ψ can only express that x equals some a_i or that it is larger than some a_i's and smaller than others. If one such x can be found in some dense ordering extending $\{a_1, \ldots, a_n\}$ then it can be found in all such dense orderings.

Example 1.6 Field theory does not satisfy the submodel condition

Consider the statement $\exists x(x \cdot x = 2)$. This sentence is not true for the rationals, but for the reals.

Lemma 1.8 *The theory of algebraically closed fields satisfies the submodel condition.*

Proof
Let $\mathcal{B}$ be an algebraically closed field, and let $\mathcal{A}$ be an algebraically closed subfield. Let $\phi = \exists x\psi$ be a simple sentence in $L(\mathcal{A})$. Then ψ is a Boolean combination of atomic formulas $\xi(x, a_1, \ldots, a_n)$, where $a_1, \ldots, a_n$ are in $\mathcal{A}$.
The function symbols are denoting plus and times, so each term $t(x)$ in $L(\mathcal{A})$ will be equivalent to a polynomial in the variable x with coefficients from $\mathcal{A}$. Since $=$ is the only predicate, each atomic formula $\xi(x, a_1, \ldots, a_n)$ is equivalent to a polynomial equation

$$P_\xi(x) = 0$$

where P_ξ has coefficients from $\mathcal{A}$. This equation will have a root in $\mathcal{A}$ if and only if it has a root in $\mathcal{B}$.
If ϕ is true in $\mathcal{A}$, the ϕ is trivially true in $\mathcal{B}$. On the other hand, assume that ϕ is true in $\mathcal{B}$ via b, i.e. $\psi(b, a_1, \ldots, a_n)$ is true in $\mathcal{B}$. If one of the atomic subformulas ξ are true for $x = b$, then b is the root of the polynomial P_ξ with coefficients from $\mathcal{A}$, so b is in $\mathcal{A}$. On the other hand, if none of the subformulas ξ of ψ are true for $x = b$, we use that $\mathcal{A}$ is infinite (see Lemma 1.5), and that a finite set of polynomials only will have a finite set of roots. Thus let $x = a$, where a is in $\mathcal{A}$ and a is not making any of the subformulas ξ of ψ true. Then a and b satisfy the same atomic subformulas of ψ, so $\psi(a, a_1, \ldots, a_n)$ must be true. This means that ϕ is true in $\mathcal{A}$.

Lemma 1.9 *Let L be a first order language and let T be a theory over L that has the submodel property. Let T' be obtained from T by adding new names to L but no new non-logical axioms.*
Then T' has the submodel property.

The proof is left for the reader.

Elimination of quantifiers

Definition 1.19 Let L be a first order language, T a theory over L.
We say that T *accepts elimination of quantifiers* if we for each formula ϕ have an open formula ψ such that

$$T \vdash \phi \leftrightarrow \psi.$$

Lemma 1.10 *Let L be a first order language and T a theory over L. Then T accepts elimination of quantifiers if and only if each elementary formula in L is equivalent in T to an open formula.*

Proof
The 'only if'-direction is obvious, so assume that each elementary formula in L will be equivalent in T to an open formula.
By induction on the complexity of a formula ϕ, we prove that it is equivalent in T to an open formula, where we use the assumption to handle the quantifier case.

Lemma 1.11 *Let L be a first order language, T a theory over L.*
Let ϕ be a sentence in L, and assume that whenever $\mathcal{A}$ and $\mathcal{B}$ are models for T satisfying the same variable free sentences, then ϕ has the same truth values in $\mathcal{A}$ and in $\mathcal{B}$.
Then ϕ is equivalent in T to a variable free sentence in L.

Note
If a formula ϕ is variable free, it contains neither bound nor free variables, and thus no quantifiers. Another way of describing such formulas are as *open sentences*.

Proof
Let T_0 be T extended with all variable free theorems in $T \cup \{\phi\}$. It is sufficient to show that ϕ is a theorem in T_0.
Assume not, and let $\mathcal{A}$ be a model for $T_0 \cup \{\neg\phi\}$.
Let Δ be the set of variable free sentences true in $\mathcal{A}$, and let $\mathcal{B}$ be a model for $T \cup \Delta$. Then the same variable free sentences are true in $\mathcal{A}$ and in $\mathcal{B}$ (namely Δ), so $\neg\phi$ will hold in $\mathcal{B}$ as well.
$\mathcal{B}$ was an arbitrary model for $T \cup \Delta$, so by the completeness theorem,

$$T \cup \Delta \vdash \neg\phi.$$

Then there are $\xi_1, \ldots, \xi_n \in \Delta$ such that

$$T, \phi \vdash \neg(\xi_1 \wedge \ldots \wedge \xi_n).$$

Now $\xi_1, \ldots \xi_n$ will be simultaneous instances of formulas $\xi'_1, \ldots, \xi'_n$ in L, and by the theorem of constants

$$T, \phi \vdash \xi'$$

where

$$\xi' = \neg(\xi'_1 \wedge \cdots \wedge \xi'_n).$$

But then each instance of ξ' will be true in $\mathcal{A}$, contradicting that ξ_i is true in $\mathcal{A}$ for each i.
The assumption was that ϕ is not a theorem in T_0. This led to a contradiction, so we are through.

Theorem 1.8 *Let L be a first order language, T a theory over L that satisfies both the isomorphism condition and the submodel condition. Then T accepts elimination of quantifiers.*

Proof
By Lemma 1.10 it is sufficient to show that if $\phi = \exists x \psi(x, x_1, \ldots, x_n)$ where ψ is open, then ϕ is equivalent in T to an open formula. Extend T to T' by adding names $e_1, \ldots, e_n$, but no new non-logical axioms. By Lemmas 1.7 and 1.9, T' has the isomorphism and submodel properties.
We will show that $\exists x \psi(x, e_1, \ldots, e_n)$ is equivalent in T' to a variable free sentence $\xi(e_1, \ldots, e_n)$. Having done this, we conclude, using the theorem of constants, that $\exists x \psi(x, x_1, \ldots, x_n)$ is equivalent in T to $\xi(x_1, \ldots, x_n)$.

By Lemma 1.11 it is sufficient to show that whenever $\mathcal{A}$ and $\mathcal{B}$ are two models for T' satisfying the same variable free formulas, then $\exists x \psi(x, e_1, \ldots, e_n)$ will have the same truth values in $\mathcal{A}$ and in $\mathcal{B}$.
If $\mathcal{A}$ and $\mathcal{B}$ satisfy the same variable-free formulas, the respective minimal substructures $\mathcal{A}_0$ and $\mathcal{B}_0$ consisting of all interpretations of closed terms will be isomorphic. By the isomorphism property for T', this isomorphism can be extended to an isomorphism between two submodels $\mathcal{A}_1$ and $\mathcal{B}_1$ of $\mathcal{A}$ and $\mathcal{B}$ resp. Since T' has the submodel property, we have that $\exists x \psi(x, e_1, \ldots, e_n)$ will have the same truth values in $\mathcal{A}$ and $\mathcal{A}_1$ and in $\mathcal{B}$ and $\mathcal{B}_1$. Since $\mathcal{A}_1$ and $\mathcal{B}_1$ are isomorphic, we are through.

Prime Models

Definition 1.20 Let L be a first order language with at least one constant symbol. Let T be a first order theory over L, and let $\mathcal{A}$ be an L-structure. $\mathcal{A}$ is called a *prime model* for T if $\mathcal{A}$ can be embedded into any model $\mathcal{B}$ for T.

One important notice, a prime model for T is not neccessarily a model for T. Actually we have

Lemma 1.12 *Let L be a first order language, T a theory over L and $\mathcal{A}$ a prime model for T. Then $\mathcal{A}$ is a prime model for any extension T' of T over L.*

The (trivial) proof is left for the reader.

Example 1.7 Let L be the language of DO extended with the constants c_i for $i \in \mathbb{N}$.
Let DO^+ be the theory DO extended with the axioms $c_i < c_{i+1}$.
Then $\mathbb{N}$ with its usual ordering, and with i as the interpretation of c_i, is a prime model for DO^+.

In Exercise 1.12 We will see that DO^+ is a complete theory that is not ω-categorical.

Lemma 1.13 *Let T be a first order theory over a language L, and assume that L has at least one constant symbol. Then the following are equivalent:*

1. *T has a prime model.*
2. *Each variable free formula in L is decidable in T.*

Proof
First assume 1. Let $\mathcal{A}$ be a prime model for T. Then for any model $\mathcal{B}$ for T and any variable free ϕ (the constant symbol in L ensures that there are variable free formulas), the truth value of ϕ in $\mathcal{B}$ will be the same as in $\mathcal{A}$. Thus the existence of a prime model ensures that each variable free formula either is true in all models for T, and thus is a theorem, or is false in all models, and thus is disprovable in T. Thus they are all decidable in T, and 2. is proved.
Now assume 2. For any model $\mathcal{A}$ of T, let $\mathcal{A}_0$ be the submodel consisting of all interpretations of closed terms t in $\mathcal{A}$.
If $\mathcal{A}$ and $\mathcal{B}$ are two models for T, we define $\pi : \mathcal{A}_0 \to \mathcal{B}_0$ by $\pi(t^{\mathcal{A}}) = t^{\mathcal{B}}$. Since all variable free formulas are decidable, π is well defined and an isopmorphism.
This shows that $\mathcal{A}_0$ is a prime model for any $\mathcal{A}$ such that $\mathcal{A} \models T$.

A completeness criterion

Theorem 1.9 *Let L be a first order language and let T be a theory over L such that*

1. *T is consistent.*
2. *The language L of T has at least one constant symbol.*
3. *T has a prime model.*
4. *T satisfies the isomorphism property.*
5. *T satisfies the submodel property.*

Then T is complete.

Proof
By Theorem 1.8 we have that T accepts elimination of quantifiers.
Let ϕ be a sentence in L and let ψ be open such that $T \vdash \phi \leftrightarrow \psi$. We may assume that ϕ is variable free, since otherwise we may replace it by $\phi(c, \ldots, c)$, where c is a constant of the language.
Since T has a prime model, ψ is decidable in T by Lemma 1.13, so ϕ is decidable in T.

1.4 Element types

Throughout this section, we will let L be a fixed countable first order language, and we will let T be a complete theory over the language L.

Definition 1.21 An *n-type* is a maximal set X of formulas $\phi(x_1, \ldots, x_n)$ such that for all $\phi_1, \ldots, \phi_k$ in X we have that

$$T \vdash \exists x_1 \cdots \exists x_n(\phi_1 \wedge \cdots \wedge \phi_k).$$

We will use the expression *maximally consistent* for this property. Each consistent set of formulas with at most $x_1, \ldots, x_n$ free can be extended to an n-type. An n-type will be closed under $\wedge$, and for each formula ϕ with at most $x_1, \ldots, x_n$ free the maximality of X will ensure that $\phi \in X$ or $\neg\phi \in X$.

Definition 1.22 Let X be an n-type and $\mathcal{A}$ a model for T. We say that $\mathcal{A}$ *realises* X if for some $a_1, \ldots, a_n$ in A, $\mathcal{A} \models \phi(a_1, \ldots, a_n)$ for each $\phi \in X$.
If $\mathcal{A}$ does not realise X we say that $\mathcal{A}$ *omits* X.

Example 1.8 let $\mathcal{N}$ be the standard model for number theory and $T = Th(\mathcal{N})$. Let X be a maximally consistent extension of $\{c_n < x \ ; \ n \in \mathbb{N}\}$ where c_n is the numeral for n in the language. This is a 1-type that is omitted by the standard model. The construction of a nonstandard elementary extension of $\mathcal{N}$ used to prove Theorem 1.3 amounts to constructing a model realising this type.

The method used to prove Theorem 1.3 is quite general, and can be used to prove the following

Theorem 1.10 *Let X be an n-type. Then there is a model $\mathcal{A}$ for T realising X.*

Proof
The proof is left as Exercise 1.17. Add new names for the n variables and use the compactness theorem.

Some n-types will be realised by every model of T.

Definition 1.23 An n-type X is *principal* if for some $\phi \in X$ we have that

$$T \vdash \phi \rightarrow \psi$$

for all $\psi \in X$. We call ϕ a *generating element.* If X is not principal, we will use the term *nonprincipal.*

Lemma 1.14 *If X is a principal type, then X is realised in every model $\mathcal{A}$ of T.*

Proof
Any interpretation of $x_1, \ldots, x_n$ that satisfies the generating element will realise the type.

Lemma 1.15 *Let n be fixed.*
Then the following are equivalent:

1. *There are infinitely many n-types*
2. *There is a nonprincipal n-type.*

Proof
First assume that there is a nonprincipal n-type $X = \{\phi_i \ ; \ i \in \mathbb{N}\}$. For each $i \in \mathbb{N}$ there must be a $j_i > i$ such that

$$T \not\vdash \phi_1 \wedge \cdots \wedge \phi_i \rightarrow \phi_{j_i}$$

since otherwise the type will be principal. Thus, $\{\phi_1, \ldots, \phi_i, \neg\phi_{j_i}\}$ must be consistent, and there is an n-type X_i extending this set. Clearly, if $j \geq j_i$, then X_i and X_j are different, so there are infinitely many n-types.
Now assume that all types are principal. Since L is countable, we will only be able to use countably many generating formulas, so the set of n-types is at most countable.
With the aim of obtaining a contradiction, assume that this set is infinite. Let $\{X_i\}_{i\in\mathbb{N}}$ be an enumeration of all the n-types. Let ϕ_i be a generating formula for X_i. Then, if $i \neq j$ we must have that $\neg\phi_i \in X_j$. In particular, if $j > max\{i_1, \ldots, i_k\}$ we will have that $\neg(\phi_{i_1} \wedge \ldots \wedge \phi_{i_k}) \in X_j$.
It follows that $\{\neg\phi_i \ ; \ i \in \mathbb{N}\}$ is a consistent set (here we use the assumption that there are infinitly many principal n-types), and may thus be extended to an n-type different from all the X_i's. This contradicts the assumption that we have enumerated them all, and the lemma is proved.

If $\mathcal{A}$ is a model for T and $a_1, \ldots, a_n$ are elements in A (possibly with repetition), then $a_1, \ldots, a_n$ realises exactly one n-type,

$$X = \{\phi(x_1, \ldots, x_n) \ ; \ \mathcal{A} \models \phi(a_1, \ldots, a_n)\}.$$

This indicates that the set of types may give us information about the class of countable structures. One example of this is

Theorem 1.11 *Assume that all n-types are principal for all n.*
Then T is ω-categorical.

Proof
We will elaborate on the proof of the ω-categoricity for the theory DO.
Let $\mathcal{A}$ and $\mathcal{B}$ be two countable models for T with domains A and B. Let $A' = \{a_1, \ldots, a_n\} \subseteq A$ and $B' = \{b_1, \ldots, b_n\} \subseteq B$ be finite sets (without repetition) and let $p(a_i) = b_i$. We call p a*partial isomorphism* if $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ realise the same n-type.

Claim
If p is a partial isomorphism and $a \in A$, then p can be extended to a partial isomorphism q defined on a.

Let us first see how to prove the theorem from the claim. The statement is symetric in $\mathcal{A}$ and $\mathcal{B}$, so given $b \in B$ we may as well extend p to a q with b in its range. $\mathcal{A}$ and $\mathcal{B}$ will be elementary equivalent since T is complete. We may consider the empty function as a partial isomorphism defined on the empty set. Using the extensions obtained from the claim, we then use the back-and-forth strategy and build up a sequence of partial isomorphisms ensuring that each $a \in A$ will be in the domain of one of the partial isomorphisms, and each $b \in B$ will be in the range of one of the partial isomorphisms. In the end, we have constructed a total isomorphism.

Proof of claim
Let X be the $n+1$-type of $\{a_1, \ldots, a_n, a\}$. Since we have assumed that all $n+1$-types are principal, there is a generating $\phi(x_1, \ldots, x_n, x_{n+1})$ in X.
Since $\mathcal{A} \models \phi(a_1, \ldots, a_n, a)$ we have that

$$\mathcal{A} \models \exists x_{n+1} \phi(a_1, \ldots, a_n, x_{n+1}).$$

Since $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ realise the same n-type,

$$\mathcal{B} \models \exists x_{n+1} \phi(b_1, \ldots, b_n, x_{n+1}).$$

Choose $b \in B$ such that $\phi(b_1, \ldots, b_n, b)$ holds in $\mathcal{B}$ and let $q(a) = b$.
Let $\psi \in X$. Since

$$T \vdash \phi(x_1, \ldots, x_n, x_{n+1}) \to \psi(x_1, \ldots, x_n, x_{n+1})$$

we see that $\psi(b_1, \ldots, b_n, b)$ will hold in $\mathcal{B}$. This shows that X is the type realised by $\{b_1, \ldots, b_n, b\}$, and the claim is proved.

As a consequence we see that if the number of n-types is finite for each n, then T is ω-categorical. We will prove the converse to this, but in order to do so we need the theorem about omitting types.

Theorem 1.12 *Let X be a nonprincipal n-type. Then there is a model for T omitting X.*

Proof
The idea of the proof is as follows: We extend the language L of T with Henkin constants c_1, c_2, . . . and we extend T with Henkin axioms. Then, following the proof of the completeness theorem, we must make a complete extension of this extended theory and then form the term model. The term model will consist of equivalence classes of closed terms, and in fact, via the Henkin constant for $\exists x(x = t)$, we see that each equivalence class will contain one Henkin constant. Thus we must ensure, during the completion process, that for any collection $c_{i_1}, \ldots, c_{i_n}$ of Henkin constants there is a formula $\phi \in X$ such that $\neg\phi(c_{i_1}, \ldots, c_{i_n})$ is added to the theory.

In the proof of the completeness theorem we ad Henkin constants and Henkin axioms in waves. However, we prove that each wave is countable. Thus the set of new constants are countable, and we may just organise them in a list as above

such that if c_j is the Henkin constant of $\exists x\phi$ and c_i occurs in ϕ, then $i < j$.
Recall that the Henkin axiom is

$$\exists x\phi(x) \rightarrow \phi(c_j).$$

For the sake of simplicity, we will assume that $n = 1$.
For each Henkin constant c_k we will find a formula $\phi_k \in X$ such that

$$T \cup \{\neg\phi_1(c_1), \dots, \neg\phi_k(c_k)\}$$

is consistent with all Henkin axioms.
From Henkin's proof of the completeness theorem we know that adding a Henkin axiom introducing a new Henkin constant will preserve consistency. Thus, in the process of verifying that $\phi_k(c_k)$ is consistent with the extended theory so far and all the Henkin constants, we only need to be concerned with the Henkin axioms involving $c_1, \dots, c_k$.
The 'construction' is by recursion on k, so assume that

$$T \cup \{\neg\phi_1(c_1) \dots, \neg\phi_{k-1}(c_{k-1})\}$$

is consistent with all the Henkin axioms.
Let $\xi(c_1, \dots, c_k)$ be the conjunction of all $\neg\phi_j(c_j)$ for $j < i$ and all Henkin axioms introducing $c_1, \dots, c_k$.
Then $T, \xi(c_1, \dots, c_k)$ is consistent.
With the aim of obtaining a contradiction, assume that $T, \xi(c_1, \dots, c_k), \neg\phi(c_k)$ is inconsistent for all $\phi \in X$.
Then $T \vdash \xi(c_1, \dots, c_k) \rightarrow \phi(c_k)$ for all $\phi \in X$.
By the theorem of consants

$$T \vdash \xi(y_1, \dots, y_{k-1}, x) \rightarrow \phi(x)$$

and thus

$$T \vdash \exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x) \rightarrow \phi(x)$$

for all $\phi \in X$. Since X is maximally consistent, there are two possibilities

1. $\exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x) \in X$
2. $\neg\exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x) \in X$.

Case 1. contradicts that X is non-principal.
In case 2., we let $\phi = \neg\exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x)$ and see that

$$T \vdash \exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x) \rightarrow \neg\exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x).$$

By propositional logic it follows that

$$T \vdash \neg\exists y_1 \cdots \exists y_{k-1}\xi(y_1, \dots, y_{n-1}, x)$$

contradicting the induction hypothesis, i.e. that $T, \xi(c_1, \ldots, c_k)$ is consistent.

The asssumption was that $T, \xi(c_1, \ldots, c_k), \neg\phi(c_k)$ is inconsistent for each $\phi \in X$. This led to a contradiction, so there is a $\phi_k \in X$ such that

$$T, \xi(c_1, \ldots, c_k), \neg\phi_k(c_k)$$

is consistent.
The choice of ξ was such that $T, \neg\phi_1(c_1), \ldots, \neg\phi_k(c_k)$ will be consistent with all the Henkin axioms.
Let T^ω be T extended with all $\neg\phi_k(c_k)$ and all Henkin axioms. T^ω will be a consistent Henkin theory. Let T^c be a completion of T^ω. Then the term model of T^c will be a model for T omitting X.

In Exercise 1.19 we will discuss why the assumption that $n = 1$ is a harmless one.

Corollary 1.1 *If there are infinitely many n-types for some n, then T is not ω-categorical.*

Proof
If there are infinitely many n-types, one of them must be non-principal. Let X be non-principal Then there one model $\mathcal{A}$ realising X and one model $\mathcal{B}$ omitting X. These models cannot be isomorphic.

Suplementary material for the advanced reader

This section can only be read with complete understanding for a reader with a background from descriptive set theory and cardinality arithmetics.

An n-type is a subset of the set of formulas $\phi(x_1, \ldots, x_n)$, which is a countable set. Via e.g. a Gödel enumeration we may consider an n-type to be a subset of the natural numbers. The set of subsets of $\mathbb{N}$ forms a compact, metrizable topological space in a natural way, homeomorphic to the Cantor space.
There are two requirements that has to be satisfied by an n-type, given

$$\phi_1, \ldots, \phi_k$$

we must have that $T \vdash \exists x_1 \cdots \exists x_n(\phi_1 \wedge \cdots \wedge \phi_k)$, and X must be maximally consistent. The first requirement defines a G_δ-set and the second a closed set. Thus the set of n-types is a G_δ-set of sets of formulas.
By a standard and classical result of descriptive set theory, any G_δ-set is either finite, countable or has the cardinality of the continuum (this holds for a much more general class than the G_δ-sets). Thus, if there are uncountably many n-types for some n there will be as many isomorphism classes of countable models for T as there are reals, since κ many countable models will at most realise $\omega \times \kappa$ many n-types, and if $\kappa < 2^\omega$ then $\omega \times \kappa < 2^\omega$.

1.5 Saturated structures

In a sense we may say that the more n-types a model $\mathcal{A}$ for T realise, the richer the model is. The most generous would have been if $\mathcal{A}$ realises every n-type for every n. Of course, if there are uncountably many n-types for some n, this is impossible, a countable structure can realise at most countably many n-types.

Definition 1.24 Let $\mathcal{A}$ be a model for T, $A_0 \subseteq A$.
Let $L(A_0)$ be L extended with names for each $a \in A_0$. By abuse of notation we will use a both as the name and for the object. Let $T[A_0]$ be the complete theory of all $L(A_0)$ sentences true in $\mathcal{A}$.

Definition 1.25 Let $\mathcal{A}$ be a model for T. We call $\mathcal{A}$ *saturated* if all 1-types in $T[A_0]$ are realised in $\mathcal{A}$ whenever A_0 is finite.

Lemma 1.16 *Assume that T has only countably many n-types for each n. Let $\mathcal{A} \models T$. Let $A_0 \subseteq A$ be finite.*
Then $T[A_0]$ has only countably many 1-types.

Proof
Let $A_0 = \{a_1, \ldots, a_n\}$ and let X be a 1-type in $T[A_0]$.
Let $X' = \{\phi(x_1, \ldots, x_n, x) \ ; \ \phi(a_1, \ldots, a_n, x) \in X\}$.
Clearly, X' is a consistent set, so X' can be extended to an $n+1$-type.
If X and Y are different 1-types in $T[A_0]$, then for some $\phi(x)$ in $L(A_0)$, $\phi \in X \wedge \neg\phi \in Y$. It follows that X' and Y' cannot be extended to the same $n+1$-type. Since there are at most countably many $n+1$-types in T, there are at most countably many 1-types in $T[A_0]$.

Lemma 1.17 *Let $\mathcal{A}$ be a model for T, $A_0 \subseteq A$ a finite set and X a 1-type in $T[A_0]$. Then there is an elementary extension of $\mathcal{A}$ realising X.*

Proof
This is proved like Theorem 1.10.

Theorem 1.13 *Assume that T has at most countably many n-types for each $n \in \mathbb{N}$.*
Then T has a saturated model.

Proof
The final model will be the direct limit of an elementary directed system, and if the reader has not solved Exercise 1.7 so far it is about time to do so before further reading of this proof.
We will start with a countable model $\mathcal{A}_0$ with domain A_0. At stage k, assume that we have constructed a countable model $\mathcal{A}_k$ with domain A_k. Then we select a finite subset $B_k \subseteq A_k$ and a 1-type X_k in $T[B_k]$ and let $\mathcal{A}_{k+1}$ be an elementary extension of $\mathcal{A}_k$ realising X_k. Here we use Lemma 1.17.
In order to obtain a saturated model in the end, we must organise the selection of B_k and X_k such that the following are satisfied:

- Whenever $B \subseteq A_k$ is finite, there are infinitely many $k' \geq k$ such that $B = B_{k'}$.
- Whenever $B \subseteq A_k$ is finite and X is a 1-type in $T[B]$, then there is a $k' \geq k$ such that $B = B_{k'}$ and $X = X_{k'}$.

We leave the details here for the reader.

Theorem 1.14 *Let $\mathcal{A}$ and $\mathcal{B}$ be countable models for T where $\mathcal{B}$ is saturated. Then there is an elementary embedding from $\mathcal{A}$ to $\mathcal{B}$.*
If both $\mathcal{A}$ and $\mathcal{B}$ are saturated, they are isomorphic.

Proof
We show how to construct an elementary embedding in the first case. The isomorphism in the second case can be constructed by a back-and-forth construction using our construction at each step.
Let A and B be the domains of $\mathcal{A}$ and $\mathcal{B}$. Let $A = \{a_k \ ; \ k \in \mathbb{N}\}$, let $A_k = \{a_0, \ldots, a_{k-1}\}$ and let $\pi_k : A_k \to B$ be injective.
We say that π_k is a *partial elementary embedding* if for all sentences ϕ in $L(A_k)$,

$$\mathcal{A} \models \phi \Leftrightarrow \mathcal{B} \models \phi^{\pi_k}$$

where ϕ^{π_k} is the sentence in $L(\mathcal{B})$ obtained from ϕ by replacing each name for an element $a \in A_k$ by the name for $\pi_k(a)$. Observe that for $k = 0$, π_k is the empty function, and then it will be a partial elementary embedding just because $\mathcal{A}$ and $\mathcal{B}$ are elementary equivalent.
Now let $B_k = \{\pi_k(a) \ ; \ a \in A_k\}$.
Let X be the 1-type in $T[A_k]$ realised by a_k, and let X' be the corresponding 1-type in $T[B_k]$ using the transformation $\phi \mapsto \phi^{\pi_k}$.
Since $\mathcal{B}$ is saturated, X' is realised by some $b \in B$.
We extend π_k to π_{k+1} by $\pi_{k+1}(a_k) = b$ where $b \in B$ realises X'. Then π_{k+1} is a partial elementary embedding and the construction can go on.
At the end, the union π of all the π_k's will be an elementary embedding from $\mathcal{A}$ to $\mathcal{B}$.

The number of countable models

At this stage the reader is invited to write a small essay elaborating on the following:

The possible cardinalities of the set of isomorphism classes of countable models of a complete countable theory is fully determined, the cardinalities can be

$$1, 3, 4, \ \ldots, \aleph_0, \ \aleph_1, \ 2^{\aleph_0}.$$

That the number 2 is left out is not a misprint, but is one of the more peculiar results, which we will discuss further.
The theory DO^+ has three different models. Elaborating on this kind of construction we may find theories where the number of models are any number $\neq 2$.

The theory $ACF(0)$ has countably many models.
There are examples of theories with continuumly many models. One example is $DO!$, an extension of DO defined as follows:

- Let A_i be a new unary predicate symbol for each $i \in \mathbb{Z}$
- Ad axioms stating that A_i is an open interval for each $i \in \mathbb{Z}$
- Ad axioms stating that $A_i < A_j$ whenever $i < j$.

Then $DO!$ is a complete theory with $2^{\aleph_0}$ many non-isomorphic models.

In order to prove that other cardinalities are out of the question, we need to go beyond first order logic, and we need elaborate background theorems from descriptive set theory. This is outside the scope of this text. It is just recently established (2002) that $\aleph_1$ is a possible alternative, even in the case that the continuum hypothesis does not hold. The construction behind this argument is far beyond our ambitions here.
We do however have most tools available in order to show that there are never exactly 2 isomorphism classes of models.

Definition 1.26 A model $\mathcal{A}$ for T is *weakly saturated* if each n-type is realised in $\mathcal{A}$ for each $n \in \mathbb{N}$.

Lemma 1.18 *Assume that the number of models for T is finite, but > 1. Then T has a model that is weakly saturated, but not saturated.*

Sketch of proof
T has a countable, saturated model $\mathcal{A}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be the other models. Assume that neither of these are weakly saturated, and for each $i \leq k$, let X_i be an n_i-type not realised in $\mathcal{A}_i$. Without loss of generality, we may assume that the variables used for the types X_i are distinct, so $X_1 \cup \cdots \cup X_k$ can be viewed as a set of formulas in the variables e.g. $x_1, \ldots, x_n$ where $n = \sum_{i=1}^{k} n_i$. Let X be an n-type extending $X_1 \cup \cdots \cup X_k$. Then X cannot be realised in $\mathcal{A}_i$ for any $i \leq k$.
Let $c_1, \ldots, c_n$ be new names, and let T^* be T extended with the axioms $\phi(c_1, \ldots, c_n)$ for $\phi(x_1, \ldots x_n) \in X$. Since $\mathcal{A}$ is saturated, we may interpret $\mathcal{A}$ in such a way that it becomes a model for T^*. Moreover, two such models will be isomorphic. Thus T^* is ω-categorical and there are finitely many n-types in T^* for each n. It follows that there are only finitely many n-types in T for each n, which contradicts the assumption.
This ends the scetch.

Corollary 1.2 *There is no complete countable theory with exactly two non-isomorphic countable models.*

1.6 ω-logic

The prime object of consideration in this chapter has been the model theory of first order logic. We have seen that for many purposes, first order logic is inadequate. Logicians have been studying other forms of logic where some of the tools from first order logic can be used. There are at least three directions to go, we may extend the use of quantifiers beyond the setting of first order logic, we may consider infinitary proof trees and we may consider infinite connectives. The first alternative is natural if we want to form logics adequate for mathematical analysis or e.g. for the study of Noetherian rings. The third alternative was used when the possible numbers of models for a first order theory was characterised in full.
In this section we will consider a form of infinitary logic for models where the natural numbers is a definable substructure. This actually requires infinite proof trees, since we cannot have the compactness theorem to be valid in this case.

1.6.1 ω-logic

For the rest of this section, let L be a first order language with (among possibly other symbols) a unary predicate symbol Ω, a constant 0 and a unary function symbol S. We will let T be a first order theory over the language L.

Definition 1.27 Let $\mathcal{A}$ be a model for T.
We call $\mathcal{A}$ an *ω-model* if Ω is interpreted as $\mathbb{N}$, 0 as zero and S as the successor-function.

Definition 1.28 Let T be a theory as above. T is *ω-sound* if T has an ω-model. A formula ϕ in L is *ω-valid* if ϕ is valid in all ω-models for T.

We may formulate a logical system adequate for this concept of validity:

Definition 1.29 *ω-logic* will be first order logic extended with four axioms and one infinitary rule:

- Let k_0 be the constant 0, and let k_{n+1} be the term $S(k_n)$.
- $\Omega(0)$ and $\Omega(x) \to \Omega(S(x))$ are new axioms in ω-logic.
- $0 \neq S(x)$ and $S(x) = S(y) \to x = y$ are new axioms in ω-logic.
- If $\phi^x_{k_n}$ is an ω-theorem in T for all $n \in \mathbb{N}$, then $\forall x(\Omega(x) \to \phi)$ an ω-theorem in T.
- We will write $T \vdash_\omega \phi$ when ϕ is an ω-theorem in T.

Remark 1.3 We will not give a precise definition of an *ω-proof*, because in order to do so we need to develop a theory of infinite, wellfounded trees. The important fact is that we have used induction to define the class of ω-theorems, and then we can prove lemmas and theorems about them using induction over this construction.

ω-logic is of interest for several reasons. One reason is that the induction axiom in number theory is provable in ω-logic. Assume that we have a proof of

$$\phi(0)$$

and that we have a proof of

$$\forall x(\Omega(x) \land \phi(x) \rightarrow \phi(S(x))).$$

We then clearly have ω-proofs for each $\phi(k_n)$ and by the ω-rule we have a proof of

$$\forall x(\Omega(c) \rightarrow \phi(x)).$$

In some odd respects, this is a simplification; the ω-logic version of Peano arithmetic is better behaved from a proof-theoretical point of view, and analysing the ω-logic versions of proofs in number theory gives us information about the strength of number theory. This kind of application requires a much deeper involvement in proof theory than we are prepared for in this text.

1.6.2 ω-completeness

We will prove the completeness theorem for ω-logic. One motivation is to demonstrate the power of the proof method of the first order version. We will leave many of the details as exercises for the reader.

Definition 1.30 T is *ω-consistent* if there is no sentence $\phi \land \neg\phi$ that is an ω-theorem in T.

We have the standard equivalent versions of the completeness theorem for ω-logic:

Lemma 1.19 *The following are equivalent:*

1. *A formula ϕ is valid in all ω-models for T if and only if ϕ is an ω-theorem in ϕ.*
2. *A theory T is ω-consistent if and only if T has an ω-model.*

The proof is left as Exercise 1.20.

Lemma 1.20 *The theorem of constants holds for ω-logic.*

The proof is left as Exercise 1.21.

Lemma 1.21 *Let L be as above, T be an ω-consistent theory over L and $\exists x\phi$ be a sentence in L. Let r be a new name, not in L.*
Then $T, \exists x\phi \rightarrow \phi^x_r$ is also ω-consistent.

The proof is left as Exercise 1.22.

Using this lemma, we may construct the Henkin-extension of an ω-consistent theory such that all finite subtheories are ω-consistent as well. We cannot conclude that the full Henkin extension is ω-consistent, simply since ω-consistency is not closed under directed unions. We will see how we can avoid this obstacle.

From now on, T is a fixed ω-consistent theory. Let r_0, r_1, . . . be the new constants that we need in order to construct the Henkin extension H of T, and let L_∞ be the extended language. Let H_n be the subtheory of H where we only added the constants $r_0, \ldots, r_{n-1}$ to T and the corresponding Henkin axioms. Then r_n is not in the language of H_n.
Let $\{\phi_n\}_{n\in\mathbb{N}}$ be an enumeration of all sentences in L_∞ such that ϕ_n is a sentence in the language of H_n.

Lemma 1.22 *Let S be an ω-consistent theory, ϕ a sentence in the language of S. Then at least one of the theories S, ϕ and $T, \neg\phi$ are ω-consistent.*

The proof is left as Exercise 1.23.

Lemma 1.23 *Let S be an ω-consistent theory, t a closed term in the language of S and assume that $S, \Omega(t)$ is ω-consistent.*
Then there is a number n such that $S, t = k_n$ is ω-consistent.

Proof
Assume that $S, t = k_n$ is inconsistent for all n.
Then $t \neq k_n$ is an ω-theorem in S for all n.
Using the ω-rule we obtain

$$S \vdash_\omega \forall x(\Omega(x) \to x \neq t).$$

This means that $S \vdash_\omega \neg\Omega(t)$, contradicting the assumption. This ends the proof of the lemma.

We will now construct the theory T_∞.
Let $T_0 = H_0 = T$.
Assume that T_n is an ω-consistent extension of H_n within the language of H_n.
By Lemma 1.21, $T \cup H_{n+1}$ will be ω-consistent.
If $T_n \cup H_{n+1}, \phi_n$ is not ω-consistent, we let

$$T_{n+1} = T_n \cup H_{n+1}, \neg\phi_n.$$

If $T_n \cup H_{n+1}, \phi_n$ is ω-consistent, but ϕ_n is not of the form $\Omega(t)$, let

$$T_{n+1} = T_n, H_{n+1}, \phi_n.$$

If $T_n \cap H_{n+1}, \phi_n$ is ω-consistent, and ϕ_n is of the form $\Omega(t)$, we choose one m such that

$$T_{n+1} = T_n \cup H_{n+1}, t = k_m$$

is ω-consistent.

Let T_∞ be the union of all T_n. We cannot prove directly that T_∞ is ω-consistent, but T_ω is a complete Henkin theory in the traditional sense, so the term model $\mathcal{A}$ of T_∞ will be a model for T_∞, and then in particular for T. $\Omega^{\mathcal{A}}$ will be the set of equivalence classes of closed terms t such that $T_\infty \vdash \Omega(t)$. For each such t, there will be a number $n \in \mathbb{N}$ such that $T_\infty \vdash t = k_n$. Thus $\Omega^{\mathcal{A}}$ will be isomorphic to $\mathcal{N}$. This indirect argument shows that $\mathcal{A}$ is an ω-model after all, and the completeness theorem for ω-logic is proved. Thus, T_∞ is actually ω-consistent.

1.7 Exercises to Chapter 1

Ex 1.1 Show that if π is an isomorphism, then π has an inverse π^{-1} that is also an isomorphism.

Ex 1.2 Show that if T is an open theory, $\mathcal{A}$ is a model for T and $\pi : \mathcal{B} \to \mathcal{A}$ is an embedding, then $\mathcal{B}$ is a model for T.

Ex 1.3 Let L be a first order language, $\mathcal{A}$ an L-structure and ϕ an L-formula with at most $x_1, \dots, x_n$ free.

a) Let s be an assignment over $\mathcal{A}$. Then

$$\mathcal{A} \models \phi[s] \Leftrightarrow \mathcal{A} \models \phi^{x_1,\dots,x_n}_{c_{s(x_1)},\dots,c_{s(x_n)}}.$$

b) Let t be a term with variables among $x_1, \dots, x_n$, and let $u_1, \dots, u_n$ and $r_1, \dots, r_n$ be two sequences of closed terms such that $u_i^{\mathcal{A}} = r_i^{\mathcal{A}}$ for each i. Then $(t^{x_1,\dots,x_n}_{u_1,\dots,u_n})^{\mathcal{A}} = (t^{x_1,\dots,x_n}_{r_1,\dots,r_n})^{\mathcal{A}}$

c) Let ϕ be a formula with free variables among $x_1, \dots, x_n$, and let $u_1, \dots, u_n$ and $r_1, \dots, r_n$ be two sequences of closed terms such that $u_i^{\mathcal{A}} = r_i^{\mathcal{A}}$ for each i.
Then $(\phi^{x_1,\dots,x_n}_{u_1,\dots,u_n})^{\mathcal{A}} \Leftrightarrow (\phi^{x_1,\dots,x_n}_{r_1,\dots,r_n})^{\mathcal{A}}$.

Ex 1.4 Let $\mathcal{A}$ and $\mathcal{B}$
Show that the following are equivalent:

1. There is an embedding π from $\mathcal{A}$ to $\mathcal{B}$.
2. $\mathcal{A}$ is isomorphic to a substructure $\mathcal{A}'$ of $\mathcal{B}$.
3. $\mathcal{A}$ is a substructure of some $\mathcal{B}'$ isomorphic to $\mathcal{B}$.

Ex 1.5 Fill in all details in the proof of Theorem 1.2.

Ex 1.6 Let L be a first order language (with equality) and let T be a complete theory over L.

a) Show that either is each model of T infinite or otherwise will all models of T have the same finite cardinality.

b) Show that if T has a finite model, then all models are isomorphic.
Hint: Use Theorem 1.5.

Ex 1.7 Let $\langle \{\mathcal{A}_i\}_{i \in I}, \{\pi_{ij}\}_{i<j}\rangle$ be an elementary directed system with directed limit $\langle \mathcal{A}, \{\pi_i\}_{i\in I}\rangle$.
Uniformly in $i \in I$, show by induction on the complexity of a sentence ϕ in $L(\mathcal{A}_i)$ that π_i preserves the truth value of ϕ.

Ex 1.8 Show that if T is a countable, ω-categorical theory, then T is complete.

Ex 1.9 Prove in detail that DO is ω-categorical

Ex 1.10 a) Show that for each prime number p there is a finite extension $F(p)$ of field theory such that the models are exactly the fields of characteristic p.

b) Show that there is an extension $F(0)$ of field theory such that the models are exactly the fields of characteristic 0

c) Let ϕ be a formula in field theory. Show that if ϕ is true in fields of arbitrarily large finite characteristic, then ϕ is valid in some field of characteristic 0.

d) Let ϕ be a formula in field theory. Show that if ϕ is valid in finite fields of arbitrarily large cardinality, then ϕ is valid in some infinite field.

e) Show that the theory of finite fields or the theory of fields of finite characteristics cannot be expressed in first order logic.

f) Show that the theory $F(0)$ or the theory of infinite fields are not finitely axiomatizable.

Ex 1.11 (For readers familiar with Zorn's Lemma).
Let $\mathcal{A}$ and $\mathcal{B}$ be algebraically closed fields and let $\mathcal{A}_0$ and $\mathcal{B}_0$ be isomorphic substructures via the isomorphism π_0.

a) Use Zorn's lemma to show that π_0 can be extended to a maximal isomorphism π_1 between substructures $\mathcal{A}_1$ and $\mathcal{B}_1$.

b) Show that $\mathcal{A}_1$ and $\mathcal{B}_1$ are algebraically closed subfields of $\mathcal{A}$ and $\mathcal{B}$.

Ex 1.12 Let DO^+ be the theory in Example 1.7.

a) The theory DO^+ has three countable, non-isomorphic models. Find examples of these, and show that there are no more than three isomorphism classes of countable models.
Hint: When you have found tree non-isomorphic examples, you may use the proof of the ω-categoricity of DO to prove that there are no more.

b) Show that DO^+ does not satisfy the isomorphism property, but that every sentence nevertheless is equivalent to a variable free sentence.

c) Show that DO^+ is complete.

Ex 1.13 A theory T is called *model complete* if $T \cup \mathcal{D}(\mathcal{A})$ is complete whenever $\mathcal{A} \models T$.
Show that if T is consistent and has the isomorphism and submodel properties, then T is model complete.

Ex 1.14 Let T be a first order theory with both the isomorphism and submodel properties. Let $\mathcal{A}$ be a model for T.
Show that $T(\mathcal{A}) = T \cup \mathcal{D}(\mathcal{A})$ has the isomorphism and submodel properties.

Ex 1.15 Let $\mathcal{F}$ be a field, $P_i(x_1, \ldots, x_k)$ and $Q_j(x_1, \ldots, x_k)$ be polynomials over $\mathcal{F}$ for $i \leq n$ and $j \leq m$.
Show that there is a solution to the simultaneous set of equations

$$P_i(x_1, \ldots, x_k) = 0$$

and inequations

$$Q(x_1, \ldots, x_k) \neq 0$$

in any field extending $\mathcal{F}$ if and only if there is a solution in the algebraic closure. (This result is known as*Hilbert's Nullstellensatz.*)

Ex 1.16 A subset of $\mathbb{C}^k$ is called *algebraic* if it is set of solutions to a polynomial equation

$$P(x_1, \ldots, x_k) = 0.$$

Show that whenever $A \subseteq \mathbb{C}^k$ is definable by a first order formula in the language of field theory using parameters from $\mathbb{C}$, then A is a Boolean combination of algebraic sets.
What does this say about the expressive power of the language of field theory, in e.g. relation to complex analysis?

Ex 1.17 Prove Theorem 1.10 in detail.

Ex 1.18 Let T be the complete theory DO^+ considered in Exercise 1.12.

a) Find a nonprincipal 1-type as the completion of some infinite set of formulas.

b) **Hard** Describe all the 1-types and show that there is exactly one nonprincipal 1-type.

Ex 1.19 Fill in the details in the following sketch of a proof of the omitting type theorem for n-types in general:

- We may enumerate all ordered n-sequences of Henkin constants in a list $\{(c_{k_1}, \ldots c_{k_n})\}_{k \in \mathbb{N}}$.

- By induction on k, we may find a formula $\phi_k \in X$ such that

  $$T \cup \{\neg\phi_1(c_{1_1}, \ldots, c_{n_1})\}$$

  is consistent with all the Henkin constants.

Ex 1.20 Prove Lemma 1.19.

Ex 1.21 Prove the theorem of constants for ω-logic.
Warning: The proof of the theorem of constants in the first order case use a variable x that occurs nowhere in the proof. In this setting, we do not have proofs, only provability. Moreover, if we had proofs, we might risk that all variables are used in the proof. Discuss how we can overcome this obstacle.

Ex 1.22 Prove Lemma 1.21.

Ex 1.23 Prove Lemma 1.22.

Chapter 2

Finitary Model Theory

In Chapter 1 we have touched a little bit on the subject of model theory. To a large extent we have been interested in countable structures; implicitly we considered finite structures to be too simple and uncountable structures to be too advanced. The main reason why we did not go on with more model theory is that we need time and space for computability theory.
In some respects, the finite structures are simple, but in other respects they are the only interesting mathematical structures there are. For instance, a database is normally a finite, but dynamic, structure. Designing logical systems that are adequate for database theory is a challenging task.

In this chapter we will offer the reader a faint look at a growing field of logic, finite model theory. We are going to prove one of the classical results, the so called 0-1 -law. In a sense, this result tells us that first order logic without constants and function symbols have remarkably restricted expressive power, and that we need higher order logic to say something intelligent about a finite structure.
This chapter will have three sections. In the first section we will prove the 0-1 -law, in the second section we will extend the language to a 2. order language, and see how the expressive power increases. The final section will be exercises.

2.1 The 0-1 -law

For the rest of this chapter, L will be a fixed first order language without constant symbols and function symbols, and with finitely many predicate symbols.

Definition 2.1 a) A *complete open description* will be an open formula $M(x_1, \ldots, x_n)$ that is a maximal consistent conjunction of literals in the variables $x_1, \ldots, x_n$ displayed.

b) If $M(x_1, \ldots, x_n)$ and $N(x_1, \ldots, x_n, y_1, \ldots, y_k)$ are complete open descriptions, then N is *an extension of* N if $\models N \rightarrow M$.

Sometimes it is convenient to use an alternative notation for long conjunctions and disjunctions. In the litterature, the notations like $\bigwedge_{i\in I}\phi_i$ and $\bigvee_{i\in I}\phi_i$ are sometimes used to form infinitary formulas where conjunction and disjunctions are taken over infinitely many subformulas. We will only use the notation when the abreviations $\phi_{i_1}\wedge\cdots\wedge\phi_{i_k}$ and $\phi_{i_1}\vee\cdots\vee\phi_{i_k}$ are a bit aqward. We will also use the notation $\vec{x}$ to mean the list of variables $x_1,\ldots,x_n$, and $\exists\vec{x}$ will be an abreviation for $\exists x_1\cdots\exists x_n$.

With this notation, we will let T be the theory over L where the nonlogical axioms are all formulas of the form

$$\forall\vec{x}((\bigwedge_{i\neq j} x_i\neq x_j\wedge M(\vec{x}))\rightarrow\exists y(\bigwedge_i y\neq x_i\wedge N(\vec{x},y)))$$

whenever M is an open complete description and N is an open complete description extending M.

Theorem 2.1 *Let L and T be as above. Then T has no finite model, but is consistent and ω-categorical.*

Proof
The proof of consistency is left as the (nontrivial) Exercise 2.2.
If $\mathcal{A}$ and $\mathcal{B}$ are two countable models for T, we will construct an isomorphism using a back and forth construction like the one used to prove that DO is ω-categorical.
A *partial isomorphism* will, in this proof, be a map $p : A_0\rightarrow B_0$ where $A_0\subseteq A$, $B_0\subseteq B$, p is 1-1 and onto and such that whenever $a_1,\ldots,a_n$ are in A_0 and $R^{\mathcal{A}}(a_1,\ldots a_n)$ then $R^{\mathcal{B}}(p(a_1),\ldots,p(a_n))$.
Given $a_1,\ldots,a_n$ there is a unique complete open description $M(x_1,\ldots,x_n)$ (unique up to equivalence) such that $M(a_1,\ldots,a_n)$ holdes, and then we may rephrase the definition of a partial isomorphism to:
If $A_0=\{a_1,\ldots,a_n\}$, $B_0=\{p(a_1),\ldots,p(a_n)\}$ and M is the complete open description of $\{a_1,\ldots,a_n\}$ then M is also the complete open description of $\{p(a_1),\ldots,p(a_n)\}$. Let p and $a\notin A_0$ be given. Let M be the complete open description of A_0 described above, and let N be the complete open description of $A_0\cup\{a\}$. The axiom for M and N just implies that p can be extended to a q defined on a.
Then, as before, we can construct an isomorphism between $\mathcal{A}$ and $\mathcal{B}$ by piecewise extensions.

We will now restrict our attention to L-structures $\mathcal{A}$, where the domain A is a set $\{0,\ldots,n-1\}$. Then the number of L-structures will be finite, let $k(L,n)$ denote this number.

Definition 2.2 Let ϕ be a sentence in L.

a) Let $k(\phi,L,n)$ be the set of L-structures over $\{0,\ldots,n-1\}$ satisfying ϕ and let the *n-probability of ϕ* be

$$\mu_{L,n}(\phi)=\frac{k(\phi,L,n)}{k(L,n)}.$$

b) Let the *asymptotic probability* of ϕ be

$$\mu_L(\phi) = \lim_{n\to\infty} \mu_{L,n}(\phi)$$

provided the limit exists.

We will drop the subscript L from now on.

Lemma 2.1 *If ϕ is an axiom of T, then $\mu(\phi) = 1$.*

Proof
Let ϕ be the axiom

$$\forall\vec{x}(\bigwedge_{i\neq j} x_i \neq x_j \wedge M(\vec{x}) \to \exists y(\bigwedge_{i} y \neq x_i \wedge N(\vec{x}, y)))$$

where $\vec{x} = x_1, \ldots, x_m$.
Assume that $m < n$. We will estimate $\mu_n(\neg\phi)$ from above, i.e.

$$a_n = \mu_n(\exists x_1 \cdots \exists x_m(\bigwedge_{i\neq j} x_i \neq x_j \wedge M(\vec{x}) \wedge \forall y(\bigwedge_{i} y \neq x_i \to \neg N(\vec{x}, y)))).$$

We will consider formulas where we have used elements from $\{1, \ldots, n\}$ as instances, without making too much fuss about how to do this strictly according to the book. The local probability of a sentence of this kind is defined in the obvious way.
There are $n(n-1)\cdots(n-m+1)$ ways of selecting distinct elements for $x_1, \ldots, x_m$, and by symetry they are equally probable. Thus we have that $a_n =$

$$n(n-1)\cdots(n-m+1)\cdot\mu_n(M(0, \ldots, m-1)\wedge\forall y(\bigwedge_{i} y \neq i \to \neg N(0, \ldots, m-1, y)))$$

which is bounded by

$$b_n = n^m \cdot \mu_n(M(0, \ldots, m-1) \wedge \forall y(\bigwedge_{i} y \neq i \to \neg N(0, \ldots, m-1, y)))$$

which again is bounded by

$$c_n = n^m \cdot \mu_n(\forall y(\bigwedge_{i=0}^{m-1} y \neq i \to \neg N(0, \ldots, m-1, y))).$$

Let $\xi_1, \ldots, \xi_l$ be all the literals in N with an occurence of the variable y. Then

$$c_n = n^m \cdot \mu_n(\forall y(\bigwedge_{i=0}^{m-1} y \neq i \to \bigvee_{j=1}^{l} \neg\xi_j(0, \ldots, m-1, y))).$$

For each k with $m \leq k \leq n$ and $j \leq l$, the probability of $\xi_j(0,\dots,m-1,k)$ is exactly $\frac{1}{2}$. Thus we may evaluate

$$c_n = n^m \cdot \prod_{i=m}^{n} \mu_n(\bigvee_{j=1}^{l} \neg\xi_j(0,\dots,m-1,i)) = n^m(1-\frac{1}{2^l})^{n-m}.$$

Since m and l are fixed and $(1-\frac{1}{2^l}) < 1$, we have that

$$\lim_{n\to\infty} n^m(1-\frac{1}{2^l})^{n-m} = 0.$$

This proves the theorem.

Corollary 2.1 *Let L and T be as above, ϕ a sentence in L. Then $\mu(\phi) = 1$ or $\mu(\phi) = 0$.*

Proof
If $T \vdash \phi$, there are axioms $\phi_1, \dots \phi_k$ in T such that

$$\phi_1, \dots, \phi_k \vdash \phi.$$

Since

$$\mu_n(\phi_1 \wedge \cdots \wedge \phi_k) \geq 1 - \sum_{i=1}^{k}(1-\mu_n(\phi_i)$$

it follows that $\mu(\phi) = 1$. By the same argument, if $T \vdash \neg\phi$ we have that $\mu(\phi) = 0$ since $\mu(\neg\phi) = 1$.

In Exercise 5.2 the reader is invited to generalise some of these results to open theories.

2.2 Second order languages

We have shown that the first order theory of algebraically closed fields of a fixed characteristic is complete. We have shown that the natural numbers cannot be characterised up to isomorphism by any first order theory. And in this chapter we have shown the 0-1 -law of finite model theory for first order formulas.
These and other results indicate that first order languages are poor in expressive power.
These languages are called *first order* because we only accept quantifiers over the domain of the structure. A standard mathematical format of the induction axiom for number theory states

$$\forall A(0 \in A \wedge \forall x \in \mathbb{N}(x \in A \to x+1 \in A) \to \forall x \in \mathbb{N}(x \in A)).$$

Here we let A vary over all sets, in particular over all subsets of $\mathbb{N}$. Quantifiers ranging over all subsets of a domain, or more generally, over all predicates of some fixed arity on a domain, will be called *second order*. We will be more precise in a while, but notice already now that the natural numbers can be described up to isomorphism by second order formulas, see Exercise 2.4.

One problem with interpreting a second order quantifier is that the interpretation is not absolute. By this we mean that the power set of an infinite set is not fully understood. There are second order sentences in number theory where the truth value will depend on chosen axiomatisations of set theory. This is discussed at more depth in a course on axiomatic set theory. If we restrict our attention to finite structures, the same argument against second order logic in general is not valid, given a finite set we have, at least theoretically, a full control of the power set.

We will show that the expressive power of second order quantifiers is such that the 0-1 -law does not hold any more. A more systematical treatment of second order logic and other extensions of first order logic is beyond the scope of this text.

Definition 2.3 a) A *second order language* L^2 will be a first order language L extended with variables X_n^k for predicates of arity k.

b) If $t_1, \ldots, t_k$ are terms in L, then $X_n^k(t_1, \ldots, t_k)$ is a new atomic formula.

c) The class of second order formulas is closed under boolean connectives and first order quantifiers in the same way as the first order formulas are.

d) If ϕ is a second order formula, then $\exists X_n^k \phi$ and $\forall X_n^k \phi$ are second order formulas.

We will not go through the full definition of how second order formulas are interpreted over an L-structure. The point is that they can be interpreted, and that if $\mathcal{A}$ and $\mathcal{B}$ are isomorphic L-structures, then they will satisfy the same second order sentences.

From now on we will restrict ourselves to the first order languale $L_=$ of equality, and its second order extension $L_=^2$. Then there is exactly one structure with domain $\{0, \ldots, n-1\}$ for each n.

Theorem 2.2 *There is a second order sentence that is true exactly for the finite structures with an even number of elements.*

Proof
In the sentence below, X will be a predicate variable of arity 1 and Y will be a predicate variable of arity 2.
Let us first look at the sentence, and discuss why it works afterwards:

$$\exists X \exists Y [\forall x \exists y Y(x,y) \wedge \forall y \exists x Y(x,y)$$

$$\wedge \forall x \forall y \forall z (Y(x,y) \wedge Y(x,z) \rightarrow y = z)$$

$$\wedge \forall x \forall y \forall z (Y(x,y) \wedge Y(z,y) \rightarrow x = z)$$

$$\wedge\forall x\forall y(Y(x,y) \rightarrow (X(x) \leftrightarrow \neg X(y)].$$

The first and third conjuncts express that Y is the graph of a function f_Y. The second and fourth conjuncts express that f_Y is onto and one-to-one, i.e. a bijection. The last conjunct expresses that f_Y is a bijection between X and the complement of X. If the ground set is finite, we may find interpretations of X and Y satisfying this exactly when the total number of elements is an even one.

Clearly the sentence above does not satisfy the 0-1 -law, so we have proved

Corollary 2.2 *The 0-1 -law is not satisfied in general by second order sentences.*

In the proof of Theorem 2.2 we saw that we can express that Y is the graph of a function and other properties of functions using first order expressions in the variable Y. Other properties that we may express is

1. X^1 is finite.
2. X^1 is infinite.
3. $\langle X^1, Y^2\rangle$ is a well ordering
4. A ring is Noetherian (which means that any degreasing sequence of ideals is finite)

The verifications of the first three facts are left as Exercise 2.5.

2.3 Exercises to Chapter 2

Ex 2.1 Show that there is an upper bound on the length of a complete open description $M(x_1, \ldots, x_n)$ depending on n and the signature of the language, and that whenever $M(x_1, \ldots, x_n)$ is a complete open description and $y_1, \ldots, y_k$ are new variables, them then M has an extension to a complete open description $N(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

Ex 2.2 Let $\mathcal{A}$ be a structure for L.

a) Let $A_0 \subseteq A$ be finite, and let M be a complete open description satisfied by the enumeration $\{a_1, \ldots, a_n\}$ of the elements in A_0.
Let N be any extension of M, with one more variable.
Show that we may extend $\mathcal{A}$ to an L-structure $\mathcal{B}$ (by adding at most one element) such that the axiom relating M to N holds.

b) Use a) to show that T will have a model, by piecewise satisfying each instance of each axiom in T.

Ex 2.3 Let L be the language with $=$ as the only symbol.

a) Show that for every sentence ϕ in L and every n we have that $\mu_n(\phi) \in \{0,1\}$.

b) Show that the function $n \mapsto \mu_n(\phi)$ will be eventually constant for each sentence ϕ.

c) Evaluate from where the function $n \mapsto \mu_n(\phi)$ is eventually constant, in terms of some number related to ϕ, and use this to show that the pure theory of equality is decidable.

Ex 2.4 Find a set of first and second order sentences in the language of number theory that will have isomorphic copies of $\mathcal{N}$ as the only models.

Ex 2.5 A standard set-theoretical characterisation of infinity is that a set A is infinite exactly when there is a bijection between A and a proper subset of A. Use this to show that we may express that a set is finite by a second order formula.
Find a second order formula that express that Y^2 is a well ordering of X^1.
If you want to go on with harder problems, we suggest that you continue with Exercise 5.4.

Chapter 3

Classical Computability Theory

3.1 The foundation, Turing's analysis

In Leary [1] the recursive functions is defined as those that can be represented in elementary number theory. $f : \mathbb{N}^k \to \mathbb{N}$ is recursive if there is a formula $\phi(x_1, \ldots, x_k, y)$ such that for all $n_1, \ldots, n_k, m$ we have that $f(n_1, \ldots, n_k) = m$ if and only if

$$N \vdash \phi(c_{n_1}, \ldots, c_{n_k}, y) \leftrightarrow y = c_m.$$

Here c_n is the numeral for n, and N is elementary number theory.

The advantage of this definition is that it is well suited for proving Gödel's incompleteness theorem without introducing too many new concepts. The problem is that there is absolutely no conceptual analysis of the notion of computability behind this definition.

Gödel defines a class of recursive functions by recursion. His aim is to define a sufficiently rich class for handling algorithms for e.g. substitution of a term for a variable, and for coding the meta-theory of a formal theory, but sufficiently simple for us to be able to show that any recursive function will be definable, and actually, representable as described above.

We are going to base our study of computability on an approach due to Kleene, and we are going to restrict ourselves to computable functions defined on the natural numbers. In many respects, computing can be described as manipulation of symbols following a given set of rules. The symbols are not then natural numbers, and different ways of representing natural numbers (binary, decadic, via numerals, Roman figures etc.) might give different concepts of computing with numbers.

The best mathematical model for computability and computations is due to Alan Turing. He defined what is now known as *Turing machines*, small finite state machines operating on an infinite one-dimentional tape and doing symbol manipulation on this tape. The machine will, between each step of the computation, be in one of finitely many stages. It will read one of the symbols on the tape, and dependent of the stage it is in and the symbol on the tape that it reads, it will according to a fixed rule change its state, rewrite the symbol and move to the symbol to the right or to the left. It may of course stay in the same state, it may of course keep the symbol on the tape as it is, and sometimes it may be convenient to let it stay where it is. We think of the tape as consisting of squares, like an old fashioned movie-tape.

A Turing machine M is determined by

1. A finite alphabet Σ including one special symbol Bl for an empty square of the tape.
2. A finite set K of states, with one special state $s \in K$ called the *initial state*.
3. A partial function $\delta : (\Sigma \times K) \to (\Sigma \times K \times \{L, S, R\})$, where L means "left", R means "right" and S means "stay where you are".

In the litterature you will find many variations in the definition of a Turing machine, but they all have the same computational strength. We decided to let the function δ, which rules the action of the machine, be partial. If $\delta(\sigma, p)$ is undefined, the machine will be in a *halting situation*, which means that the computation commes to an end. We are not going to give a precise definition of the operational semantics for Turing machines, but are content with an intuitive description:

The operational semantics of Turing Machines

Let $M = \langle \Sigma, K, s, \delta \rangle$ be a Turing machine.
The starting configuration of M will consist of a word w (called *the input*) in Σ written on an otherwise blank tape that is infinite in both directions, a position on the tape and the initial state s.
At each step, the machine M may enter a new state, rewrite the symbol at its position on the tape and shift its position one square to the right or to the left, all according to the function δ.
If M is in stage p and reads the symbol σ, and $\delta(\sigma, p)$ is undefined, M halts. Then we who observe M will know this, and we will be able to read the content of the tape, which will be called *the output*.

Normally there will be a convention that there should be no blanks in the input word and that the machine will be started at the first blank square to the left or to the right of the input word. Then the following makes sense

Definition 3.1 Let Σ_0 be an alphabet not containing the symbol Bl
Let Σ_0^* be the set of finite words over Σ_0.
Let $f : \Sigma_0^* \to \Sigma_0^*$ be a partial function.
We say that f is *Turing Computable* if there is a Turing Machine M over an alphabet $\Sigma \supset \Sigma_0$ such that if M is started with $w \in \Sigma^*$ on the tape, then it halts if and only if $f(w)$ is defined, and then with $f(w)$ as the output word.

Turing claimed that a Turing machine can

- Search systematically for pieces of information.
- Remember pieces of information
- Rewrite contents

all according to fixed rules. As an example we may consider the map $n \mapsto n!$ and the map $m \mapsto m^m$. We all agree that these maps are in princple computable, but try to think about how a computation of $10^{10}!$ could be carried out: We would need a lot of book-keeping devices in order to be at the top of the situation at each stage, but nothing that is not covered by the three items above.

We follow Turing in claiming that his model is a good mathematical model for algorithmic computations. We are not going to make this book into a study of Turing Machines. The interested reader should consult other textbooks on the subject. The two key results are:

Theorem 3.1 *There is a fixed alphabet Σ such that for any alphabet Σ', any Turing Machine M over Σ' may be coded as a word "M" in σ and every word w in Σ' may be coded as a word "w" in Σ such that the partial function*

$$U(\text{"}M\text{""}w\text{"}) = \text{"}M(w)\text{"}$$

is Turing computable, where we write $M(w)$ for the output word if the input word is w.

This is known as the existence of a *Universal Turing Machine.*

The *Halting Problem* is the following:

Given a Turing machine M and an input w, will M eventually come to a halt when started on w?

Theorem 3.2 *There is no Turing machine H that solves the Halting Problem in the following sense:*
$H(\text{"}M\text{""}w\text{"})$ will always halt, and will halt with an empty output word if and only if M halts on w.

Our approach to computability will be more in the original style of Gödel, we will study functions defined for natural numbers only. However, the results that we obtain will be relevant for the more general approaches as well. This is based on what is known as the Church-Turing Thesis, which we phrase like this:

All algorithms can be simulated by a Turing Machine

and the fact that Turing-computability can be reduced to the notion we will be working with. This is left as one of the minor projects in Chapter 5.

3.2 Computable functions and c.e. sets

3.2.1 The primitive recursive functions

The basic definition

Recursion means 'backtracking', and in pre-Church/Kleene mathematics the term *recursive function* was used for the functions defined by iterated recursion. In this pre-Church/Kleene context a recursive definition will be a definition of a function on the natural numbers, where we give one initial value $f(0)$, and define $f(k+1)$ as a function of $f(k)$ and k. E.g. Skolem used the term 'recursive function' in this way. Following Kleene, we will call these functions *primitive recursive.*

We let $\vec{x}$ and $\vec{y}$ etc. denote ordered sequences of natural numbers of some fixed length. Normally the length will not be specified, but will be clear from the context.

Definition 3.2 *The primitive recursive functions* $f : \mathbb{N}^n \to \mathbb{N}$ will be the least class of functions satisfying:

i) $f(x, \vec{y}) = x + 1$ is primitive recursive.

ii) $f(x, \vec{y}) = x$ is primitive recursive.

iii) $f(\vec{x}) = q$ is primitive recursive for each $q \in \mathbb{N}$.

iv) If g is n-ary and primitive recursive, and $f_1, \ldots, f_n$ are m-ary and primitive recursive, then the composition

$$h(\vec{x}) = g(f_1(\vec{x}), \ldots, f_n(\vec{x}))$$

is primitive recursive.

v) If g is primitive recursive with arity $n + m$, and h_1 and h_2 are primitive recursive with arity n, then f will be primitive recursive, where f is defined by

$$f(\vec{x}, \vec{y}) = h_1(\vec{x}) \text{ if } g(\vec{x}, \vec{y}) = 0$$
$$f(\vec{x}, \vec{y}) = h_2(\vec{x}) \text{ if } g(\vec{x}, \vec{y}) \neq 0$$

We call this a *definition by cases.*

vi) If g is primitive recursive and n-ary, and τ is a permutation of $\{1, \ldots, n\}$, then f is primitive recursive, where

$$f(x_1, \ldots, x_n) = g(x_{\tau(1)}, \ldots, x_{\tau(n)}).$$

vii) If g and h are primitive recursive of arity n and $n + 2$ resp., then f is primitive recursive where

$f(0,\vec{y}) = g(\vec{y})$
$f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$

We say that f is defined by *primitive recursion* or *by the recursion operator* from g and h.

Remark 3.1 We have made no attempts to minimize this definition. It will be hard enough to establish a pool of primitive recursive functions rich enough to make our life easy. The establishment of this pool is the goal of this section. Most of the work has to be carried out as problem solving, since to a computability theorist this pool of primitive recursive functions must be an integrated part of oneself.

The pool of primitive recursive functions and sets

Standard number theoretical functions like addition, multiplication, exponentiation and factorial will all be primitive recursive. Subtraction is not primitive recursive for the simple reason that it leads us outside $\mathbb{N}$. We define a modified subtraction. This will be primitive recursive, see Exercise 3.1.

Definition 3.3 We let $\dot{-}$ be the modified subtraction defined by

$x \dot{-} y = x - y$ if $y \leq x$

$x \dot{-} y = 0$ if $y \leq x$.

Using parts ii) and vi) of the definition of the primitive recursive functions, we can show that all projection maps

$$I_{i,n}(x_1,\ldots,x_n) = x_i$$

are primitive recursive. We can use this to consider any function of a set of variables as a function of a larger set of variables, as in

$$f(x_1,x_2,x_3) = g(I_{1,3}(x_1,x_2,x_3), I_{3,3}(x_1,x_2,x_3)) = g(x_1,x_3).$$

Thus we will not need to be concerned with the requirement that every function in a composition must be of the same arity. This will simplify some of the descriptions of primitive recursive functions.

Definition 3.4 Let $A \subseteq \mathbb{N}^n$. The *characteristic function* of A will be the function

$$K_A : \mathbb{N}^n \to \mathbb{N}$$

that is 1 on A and 0 outside A.

Definition 3.5 A set $A \subseteq \mathbb{N}^n$ is *primitive recursive* if the characteristic function K_A is primitive recursive.

The primitive recursive sets will form a Boolean algebra for every dimension, see Exercise 3.2.

Every set defined from $=$ and $<$ using propositional calculus will be primitive recursive. We may also use functions known to be primitive recursive in the definition of primitive recursive sets, and primitive recursive sets when we describe primitive recursive functions. We will leave the verification of most of this to the reader, just state the properties of primitive recursion that are useful to us. The proofs are simple, and there is no harm leaving them as exercises. We will however prove one basic (and simple) lemma:

Lemma 3.1 *Let $f : \mathbb{N}^{1+n} \to \mathbb{N}$ be primitive recursive. Then the function g defined as the bounded product*

$$g(x, \vec{y}) = \prod_{z \leq x} f(z, \vec{y})$$

will be primitive recursive.

Proof
We define g by primitive recursion as follows
$g(0, \vec{y}) = f(0, \vec{y})$
$g(x+1, \vec{y}) = g(x, \vec{y}) \cdot f(x+1, \vec{y})$

By the same argument we can show that the primitive recursive functions will be closed under bounded sums. What is more important to us is that the primitive recursive sets will be closed under bounded quantification. This is an important strengthening of the language we may use to define primitive recursive sets and functions.

Lemma 3.2 *Let $A \subseteq \mathbb{N}^{1+n}$ be primitive recursive. Then the following sets are primitive recursive*

a) $B = \{(x, \vec{y}) \mid \exists z \leq y((z, \vec{y}) \in A)\}$

b) $C = \{(x, \vec{y}) \mid \forall z \leq y((z, \vec{y}) \in A)\}$

The proof is left as Exercise 3.3.

In computability theory, the μ-operator is important. Within primitive recursion theory we may often use *bounded search*, or a bounded μ-operator:

Lemma 3.3 *Let $f : \mathbb{N}^{1+n} \to \mathbb{N}$ be primitive recursive. Then*

$$g(x, \vec{y}) = \mu_{<x} z.(f(z, \vec{y}) = 0)$$

is primitive recursive, where $g(x, \vec{y})$ is the least z such that $f(z, \vec{y}) = 0$ if there is one such $z < x$, while $g(x, \vec{y}) = x$ otherwise.

g can be defined using primitive recursion and definition by cases. The details are left as Exercise 3.4.

The interplay between the primitive recursive functions and the primitive recursive sets is ruled by the following principles:

Theorem 3.3 *a) Every set definable from $=$ and $<$ using primitive recursive functions, boolean operators and bounded quantifiers will be primitive recursive.*

b) Every function defined by the schemes of primitive recursion, bounded search over a primitive recursive set and definition by cases over a finite partition of $\mathbb{N}^n$ into primitive recursive sets will be primitive recursive.

There is no need to prove this theorem, since it in a sense is the synthesis of what has been proved so far. The consequence is the level of freedom in defining primitive recursive functions and relations we have obtained. This freedom will be sufficient when we later claim that certain facts are trivial to prove.

Sequence numbers

One important use of primitive recursion is the coding of finite sequences. Gödel needed an elaborate way of coding finite sequences via the so called β-function. As we mentioned above, it was important for Gödel to show that the recursive functions are definable in ordinary number theory. Since this is not important to us to the same extent, we will use full primitive recursion in coding such sequences. All proofs of the lemmas below are trivial and can safely be left for the reader.

Lemma 3.4 *a) The set of prime numbers (starting with 2 as the least prime number) is primitive recursive.*

b) The monotone enumeration $\{p_i\}_{i\in\mathbb{N}}$ of the prime numbers is primitive recursive.

We now define the sequence numbers:

Definition 3.6 Let $x_0, \ldots, x_{n-1}$ be a finite sequence of numbers, $n = 0$ corresponding to the empty sequence.
We let the corresponding *sequence number* $\langle x_0, \ldots, x_{n-1}\rangle$ be the number

$$2^n \cdot \prod_{i=0}^{n-1} p_{i+1}^{x_i}$$

If $y = \langle x_0, \ldots, x_{n-1}\rangle$, we let $lh(y)$ (the *length of* y) be n and $(y)_i = x_i$

Lemma 3.5 *a) The set of sequence numbers is primitive recursive, and the sequence numbering is one-to-one (but not surjective).*

b) The function lh is the restriction of a primitive recursive function to the set of sequence numbers.

c) The function $coor(y, i) = (y)_i$ (the i'th coordinate of y) defined for sequence numbers y of length $> i$ is the restriction of a primitive recursive function of two variables.

d) For each n, the function

$$seq_n(x_0, \ldots, x_{n-1}) = \langle x_0, \ldots, x_{n-1} \rangle$$

is primitive recursive.

e) For all sequences $x_0, \ldots, x_{n-1}$ and $i < n$, $x_i < \langle x_0, \ldots, x_{n-1} \rangle$

f) 1 is the sequence number of the empty sequence.

It makes no sense to say that the full sequence numbering is primitive recursive. However any numbering satisfying Lemma 3.5 can be used for the purposes of this course, so there is no need to learn the details of this definition. Occationally it may be useful to assume that the sequence numbering is surjective. This can be achieved, see Exercise 3.5.

We will sometimes use an alternative coding of pairs that is both 1-1 and onto:

Definition 3.7 Let

$$P(x, y) = \frac{1}{2}((x + y)^2 + 3x + y)$$

It can be shown that $P : \mathbb{N}^2 \to \mathbb{N}$ is a bijection. Let π_1 and π_2 be the two projection maps such that for any x

$$P(\pi_1(x), \pi_2(x)) = x.$$

P, π_1 and π_2 are primitive recursive. The verifications are left for the reader as Exercise 3.5 e).

Ackermann proved that there is a total computable function in one variable that is not primitive recursive. His observation was that in the list of functions

$$f_0(x, y) = x + 1,\ f_1(x, y) = x + y,\ f_2(x, y) = xy,\ f_3(x, y) = x^y$$

each function but the first is defined as a y-iteration of the previous one.
For $n \geq 3$ we may then define

$$f_n(x, 0) = 1,\ f_n(x, y + 1) = f_{n-1}(f_n(x, y), x)$$

This defines the generalised exponentiations, or the *Ackermann-branches.* The important properties are given in Exercise 3.6.

3.2.2 The computable functions

The μ-operator

We extend the definition of the primitive recursive functions to a definition of the computable functions by adding one principle of infinite search. We will consider the construction

$$g(\vec{x}) = \mu x.f(x,\vec{x}) = 0.$$

In order to understand this definition, we must discuss what we actually mean, i.e. which algorithm this is supposed to represent.
Intuitivly we want $\mu x.g(x,\vec{x}) = 0$ to be the least x such that $g(x,\vec{x})$ is 0. If $g(x,\vec{x}) \in \mathbb{N}$ for all x, this is unproblematic, we search for this least x by computing $g(0,\vec{x})$, $g(1,\vec{x})$ and so on until we find one value of x giving 0 as the outcome. Now, if there is no such x we will search in vain, or in more technical terms, our procedure will be non-terminating. This forces us to introduce partial functions, i.e. functions being undefined on certain arguments. This also forces us to be careful about our interpretation of the μ-operator, we may in the search for the least x such that $g(x,\vec{x}) = 0$ turn into a z for which $g(z,\vec{x})$ is undefined before we find a z for which $g(z,\vec{x}) = 0$. In this case we will let $\mu x.g(x,\vec{x}) = 0$ be undefined, realizing that the algorithm for computing this number that we have in mind will be non-terminating.
With this clarification we ad the following

Definition 3.8 The *computable functions* is the least class of partial functions $f : \mathbb{N}^n \to \mathbb{N}$ satisfying i) - vii) in the definition of the primitive recursive functions, together with the extra clause

viii) If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is computable then

$$f(\vec{x}) = \mu x.g(x,\vec{x}) = 0$$

is computable.

A set is *computable* if the characteristic function is computable.
A *total computable function* is a computable function terminating on all inputs from the domain $\mathbb{N}^n$.

We must have in mind that the characteristic function of a set is total, so dealing with computable sets and with total computable functions will be much of the same. This is made precise in the following lemma:

Lemma 3.6 *Let $f : \mathbb{N}^n \to \mathbb{N}$ be total. Then the following are equivalent:*

i) f is computable.

ii) The graph of f, seen as a subset of $\mathbb{N}^{n+1}$, is computable.

Proof
Let A be the graph of f, K_A the characteristic function.
If f is computable,

$$K_A(\vec{x}, y) = K_=(f(\vec{x}), y)$$

and $K_=$ is primitive recursive, see Exercise 3.1, g) and e).
If K_A is computable, then

$$f(\vec{x}) = \mu y.1 \dot{-} K_A(\vec{x}, y) = 0$$

so f is computable.

Remark 3.2 This result will not hold for primitive recursion, there will be functions with primitive recursive graphs that are not primitive recursive.

There is an interplay between the total computable functions and the computable sets resembling Theorem 3.3 as follows

Theorem 3.4 *a) Any set definable from computable sets and total computable functions using boolean valued operators and bounded quantifiers will be computable.*

b) If $A \subseteq \mathbb{N}^{n+1}$ is computable, then g is computable, where
$g(\vec{x})$ is the least number z such that $(z, \vec{x}) \in A$.

The proof is trivial.

Kleene's T-predicate

The definition of the computable functions is technically an inductive definition of a class of partial functions. It is, however, important to be aware of the computational interpretation of this definition, the so to say 'operational semantics'. Every partial computable function is given by a term in a language with constants for each number, the +1 function and symbols for the recursion operator and the μ-operator.
This operational semantics then tells us that there are actual computations going on. Now we will define the concept of a *computation tree*. A computation tree will be a number coding every step in a computation up to the final outcome. In order to be able to do so, we need a Gödel numbering, or an indexing, of the computable functions. Beware that the numbering will not be 1-1, we will enumerate the algorithms or the terms, and then only indirectly enumerate the computable functions.

Definition 3.9 For each number e we define the partial function ϕ_e of n variables as follows:

i) If $e = \langle 1 \rangle$, let $\phi_e(x, \vec{y}) = x + 1$.

ii) If $e = \langle 2 \rangle$, let $\phi_e(x, \vec{y}) = x$.

iii) If $e = \langle 3, q\rangle$, let $\phi_e(\vec{x}) = q$.

iv) If $e = \langle 4, e', d_1, \ldots, d_m\rangle$ let

$$\phi_e(\vec{x}) = \phi_{e'}(\phi_{d_1}(\vec{x}), \ldots, \phi_{d_n}(\vec{x})).$$

v) If $e = \langle 5, d_1, d_2, d_3\rangle$ then
$\phi_e(\vec{x}, \vec{y}) = \phi_{d_1}(\vec{x})$ if $\phi_{d_3}(\vec{x}, \vec{y}) = 0$
$\phi_e(\vec{x}, \vec{y}) = \phi_{d_2}(\vec{x})$ if $\phi_{d_3}(\vec{x}, \vec{y}) \neq 0$
Here we mean that $\phi_{d_3}(\vec{x}, \vec{y})$ has a value that is different from 0. If $\phi_{d_3}(\vec{x}, \vec{y})$ is undefined, then $\phi_e(\vec{x}, \vec{y})$ is undefined.

vi) If $e = \langle 6, d, \langle\tau\rangle\rangle$ where $\langle\tau\rangle$ is the sequence number of a permutation τ of n variables, then

$$\phi_e(x_1, \ldots, x_n) = \phi_d(x_{\tau(1)}, \ldots, x_{\tau(n)}).$$

vii) If $e = \langle d_1, d_2\rangle$ then
$\phi_e(0, \vec{y}) = \phi_{d_1}(\vec{y})$
$\phi_e(x + 1, \vec{y}) = \phi_{d_2}(\phi_e(x, \vec{y}), x, \vec{y})$.

viii) If $e = \langle 8, d\rangle$ then

$$\phi_e(\vec{x}) = \mu z.\phi_d(z, \vec{x}) = 0$$

Otherwise If neither of i) - viii) above applies, let $\phi_e(\vec{x})$ be undefined.

Remark 3.3 We have defined $\phi_e(\vec{x})$ for every index e and every input $\vec{x}$, either as an undefined value or as a natural number. Indeed, if we get a natural number, this number will be unique, see Exercise 3.7.
We should have used different notation in order to distinguish between the various interpretatons of ϕ_e as the arity varies. It is quite common in the litterature to restrict the notation ϕ_e to the function of one variable, while we use ϕ_e^n for the function of n variables with index e.

Definition 3.10 We write $\phi_e(\vec{x})\downarrow$ if there is a y with $\phi_e(\vec{x}) = y$. We then say that $\phi_e(\vec{x})$ *terminates.*

We are now ready to use the sequence numbering and this indexing to define computation trees. Each terminating computation will have a unique computation tree, a number coding each step of the computation from the input to the output. We will actually be overloading this code with information, but for our purposes this is harmless. What is important is that information retrieval will be easy.

Definition 3.11 Let $\phi_e(\vec{x}) = y$. By primitive recursion on e we define the *computation tree* of $\phi_e(\vec{x}) = y$ as follows, assuming that the index e will be the index of the corresponding case:

i) $\langle e, x, \vec{y}, x + 1\rangle$ is the computation tree for $\phi_e(x, \vec{y}) = x + 1$.

ii) $\langle e, x, \vec{y}, x\rangle$ is the computation tree for $\phi_e(x, \vec{y}) = x$.

iii) $\langle e, \vec{x}, q\rangle$ is the computation tree for $\phi_e(\vec{x}) = q$.

iv) $\langle e, t, t_1, \ldots, t_n, y\rangle$ is the computation tree in this case, where each t_i is the computation tree for $\phi_{d_i}(\vec{x}) = z_i$ and t is the computation tree for $\phi_{e'}(\vec{z}) = y$

v) $\langle e, t_1, t_2, y\rangle$ is the computation tree in this case, where t_1 is the computation tree for $\phi_{d_3}(\vec{x}, \vec{y}) = z$ and t_2 is the computation tree for $\phi_{d_1}(\vec{x}) = y$ if $z = 0$ and for $\phi_{d_2}(\vec{x}) = y$ if $z \neq 0$.

vi) $\langle e, t_1, y\rangle$ is the computation tree in this case, where t_1 is the computation tree for $\phi_d(x_{\tau(1)}, \ldots, x_{\tau(n)}) = y$.

vii) $\langle e, 0, t, y\rangle$ is the computation tree for $\phi_e(0, \vec{y}) = y$ when t is the computation tree for $\phi_{d_1}(\vec{y}) = y$.
$\langle e, x+1, t_1, t_2, y\rangle$ is the computation tree for $\phi_e(x+1, \vec{y})$ when t_1 is the computation tree for $\phi_e(x, \vec{y}) = z$ and t_2 is the computation tree for $\phi_{d_2}(z, x, \vec{y}) = y$.

viii) $\langle e, t_0, \ldots, t_{y-1}, t_y, y\rangle$ is the computation tree in this case, where t_i is the computation tree for $\phi_d(i, \vec{x}) = z_i \neq 0$ for $i < y$ and t_y is the computation tree for $\phi_d(y, \vec{x}) = 0$.

We are now ready to define Kleene's T-predicate:

Definition 3.12 Let

$$T_n(e, x_1, \ldots, x_n, t)$$

if t is a computation tree for $\phi_e(x_1, \ldots, x_n)$

We will normally write T instead of T_1.

Theorem 3.5 *a) For each n, T_n is primitive recursive.*

b) There is a primitive recursive function U such that if t is a computation tree, then $U(t)$ is the output of the corresponding computation.

c) **(Kleene's Normal Form Theorem)**
For every arity n and all e we have

$$\phi_e(x_1, \ldots, x_n) = U(\mu t.T_n(e, x_1, \ldots, x_n)).$$

Proof
It is only a) that requires a proof. The proof of a) is however trivial, we construct the characteristic function of T_n by recursion on the last variable t. Monotonisity of the sequence numbering is important here. We leave the tedious, but simple details for the reader.

Corollary 3.1 *For each number n, the function*

$$f(e, x_1, \ldots, x_n) = \phi_e(x_1, \ldots, x_n)$$

is computable.

Remark 3.4 Corollary 3.1 is the analogue of the existence of a universal Turing Machine, we can enumerate the computable functions in such a way that each computable function is uniformly computable in any of the numbers enumerating it. Beware that this universal function is partial. There is no universal function for the total computable functions, see Problem 3.9.

The Recursion Theorem

The recursion theorem is one of the key insights in computability theory introduced by Kleene. In programming terms it says that we may define a set of procedures where we in the definition of each procedure refer to the other procedures in a circular way. The proof we give for the recursion theorem will be a kind of 'white rabbit out of the hat' argument based on the much more intuitive S^n_m-theorem. So let us first explain the S^n_m-theorem. Let f be a computable function of several variables. Now, if we fix the value of some of the variables, we will get a computable function in the rest of the variables. The S^n_m-theorem tells us that the index for this new function can be obtained in a primitive recursive way from the index of the original function and the values of the fixed variables. We have to prove one technical lemma:

Lemma 3.7 *There is a primitive recursive function ρ such that if*

$$\phi_e(x_1, \ldots, x_n) = t$$

and

$$1 \leq i \leq n$$

then

$$\phi_{\rho(e,i,x_i)}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = t$$

Proof
We define ρ by induction on e, considering the cases i) - viii). We leave some of the easy details for the reader, see Problem 3.11.
$\phi_e(x, \vec{y}) = x + 1$:
If $1 < i$ let $\rho(e, i, x_i) = e$, while $\rho(e, 1, x_1) = \langle 3, x_1 + 1\rangle$.
The cases ii) and iii) are left for the reader.
If $e = \langle 4, e', d_1, \ldots, d_n\rangle$, we simply let

$$\rho(e, i, x_i) = \langle 4, e', \rho(d_1, i, x_i), \ldots, \rho(d_n, i, x_i)\rangle.$$

Cases v) and vi) are left for the reader.
Case vii) splits into two subcases. If $1 < i$, this case is easy. For $i = 1$ we let

$\rho(e, 1, 0) = d_1$.
$\rho(e, 1, x+1) = \langle 4, d_2, \rho(e, 1, x), d_x, \vec{d}_{I_{i,n}} \rangle$
where d_x is the index for $f(\vec{y}) = x$ and $d_{I_{i,n}}$ is the index for the function selecting y_i from $\vec{y}$. In this case the primitive recursion is replaced by an iterated composition, the depth of which is determined by the value of x.
Case viii) is again easy, and is left for the reader.

Theorem 3.6 (The S^n_m-theorem)
Let $n \geq 1, m \geq 1$. There is a primitive recursive function S^n_m such that for all $e, x_1, \ldots, x_n, y_m, \ldots, y_m$

$$\phi_e(x_1, \ldots, x_n, y_1, \ldots, y_m) = \phi_{S^n_m(e, x_1, \ldots, x_n)}(y_1, \ldots, y_m)$$

Proof
Let ρ be as in Lemma 3.7.
Let $S^1_m(e, x) = \rho(e, 1, x)$
Let $S^{k+1}_m(e, x_1, \ldots, x_{k+1}) = \rho(S^k_{m+1}(e, x_1, \ldots, x_k), 1, x_{k+1})$. By an easy induction on k we see that this construction works for all m.

The S^n_m-theorem is a handy tool in itself, and we will use it frequently stating that we can find an index for a computable function uniformly in some parameter. Now we will use the S^n_m-theorem to prove the surprisingly strong

Theorem 3.7 (The Recursion Theorem)
Let $f(e, \vec{x})$ be a partial, computable function.
Then there is an index e_o such that for all $\vec{x}$

$$\phi_{e_0}(\vec{x}) = f(e_0, \vec{x}).$$

Proof
Recall that we by this equality mean that either both sides are undefined or both sides are defined and equal. Let

$$g(e, \vec{x}) = f(S^1_n(e, e), \vec{x})$$

and let $\hat{g}$ be an index for g. Let

$$e_0 = S^1_n(\hat{g}, \hat{g}).$$

Then

$$\phi_{e_0}(\vec{x}) = \phi_{S^1_n(\hat{g}, \hat{g})}(\vec{x}) = \phi_{\hat{g}}(\hat{g}, \vec{x}) = g(\hat{g}, \vec{x}) = f(S^1_n(\hat{g}, \hat{g}), \vec{x}) = f(e_0, \vec{x}).$$

Remark 3.5 Readers familiar with the fixed point construction in untyped λ-calculus may recognise this proof as a close relative, and indeed it is essentially the same proof. The recursion theorem is a very powerful tool for constructing computable functions by self reference. In Chapter 4 we will use the recursion theorem to construct computable functions essentially by transfinite induction. Here we will give a completly different application, we will prove that there is no nontrivial set of partial computable functions such that the set of indices for functions in the class is computable.

Theorem 3.8 (Riece)
Let $A \subseteq \mathbb{N}$ be a computable set such that if $e \in A$ and $\phi_e = \phi_d$ then $d \in A$. Then $A = \mathbb{N}$ or $A = \emptyset$.

Proof
Assume not, and let $a \in A$ and $b \not\in A$.
Let $f(e,x) = \phi_b(x)$ if $e \in A$ and $f(e,x) = \phi_a(x)$ if $e \not\in A$.
By the recursion theorem, let e_0 be such that for all x

$$f(e_0, x) = \phi_{e_0}(x).$$

If $e_0 \in A$, then $\phi_{e_0} = \phi_b$ so $e_0 \not\in A$.
If $e_0 \not\in A$, then $\phi_{e_0} = \phi_a$ so $e_0 \in A$.
This is a clear contradiction, and the theorem is proved.

Corollary 3.2 (Unsolvability of the halting problem)
$\{(e,x) \mid \phi_e(x)\downarrow\}$ *is not computable.*

Remark 3.6 Riece's theorem is of course stronger than the unsolvability of the Halting Problem, for which we need much less machinery.

3.2.3 Computably enumerable sets

Four equivalent definitions

An enumeration of a set is a function defined on all of $\mathbb{N}$ that is onto the set. For subsets of $\mathbb{N}$ we may ask for computable enumerations of a set. A set permitting a computable enumeration will be called *computably enumerable* or just *c.e.* The standard terminology over many years has been *recursively enumerable* or just *r.e.* , because the expression *recursive* was used by Kleene and many with him. We will stick to the word *computable* and thus to the term *computably enumerable.*
Of course there is no enumeration of the empty set, but we will call this set c.e. nevertheless.
In this section we will give some characterisations of the c.e. sets. One important characterisation will be as the semidecidable sets. A computable set will be decidable, we have an algorithm for deciding when an element is in the set or not. In a semidecidable set we will have an algorithm that verifies that an element is in the set when it is, but when the element is not in the set, this algorithm may never terminate. A typical semidecidable set is the solving set of the Halting Problem

$$\{(e,x) \mid \phi_e(x)\downarrow\}.$$

Another example is the set of theorems in first order number theory or any nicely axiomatisable theory, or the set of words in some general grammar. We will show that the semidecidable subsets of $\mathbb{N}$ will be exactly the c.e. sets. A third characterisation will be as the sets of projections of primitive recursive sets.

In the litterature this class is known as the Σ^0_1-sets. A fourth characterisation will be as the ranges of partial, computable functions.

This is enough talk, let us move to the definition:

Definition 3.13 Let $A \subseteq \mathbb{N}$. We call A *computably enumerable* or just *c.e.* if $A = \emptyset$ or A is the range of a total computable function.

Theorem 3.9 *Let $A \subseteq \mathbb{N}$. Then the following are equivalent:*

i) A is c.e.

ii) A is the range of a partial computable function.

iii) There is a primitive recursive set $S \subseteq \mathbb{N}^2$ such that

$$A = \{n \mid \exists m(n, m) \in S\}$$

iv) There is a partial computable function with domain A.

Proof
Since the empty set satisfies all four properties, we will assume that $A \neq \emptyset$.
i) $\Rightarrow$ ii):
Trivial since we in this volume consider the total functions as a subclass of the partial functions.
ii) $\Rightarrow$ iii):
Let A be the range of ϕ_e.
Then

$$n \in A \Leftrightarrow \exists y(T(e, \pi_1(y), \pi_2(y)) \wedge n = U(\pi_2(y)))$$

where $()_i$ is i'th coordinate of a sequence number, T is Kleene's T-predicate and U is the function selecting the value from a computation tree. The matrix of this expression is primitive recursive.
iii) $\Rightarrow$ iv): Let

$$n \in A \Leftrightarrow \exists m((n, m) \in S).$$

where S is primitive recursive. Then A is the domain of the partial computable function

$$f(n) = \mu m.(n, m) \in S.$$

iv) $\Rightarrow$ i):
Let A be the domain of ϕ_e and let $a \in A$ (here we will use the assumption that A is non-empty).
Let $f(y) = \pi_1(y)$ if $T(e, \pi_1(y), \pi_2(y))$, $f(y) = a$ otherwise. Then f will be computable, and A will be the range of f.
This ends the proof of the theorem.

Clearly characterisations ii) and iii) makes sense for subsets of $\mathbb{N}^n$ as well for $n > 1$, and the equivalence will still hold. From now on we will talk about c.e. sets of any dimension. The relationship between c.e. subsets of $\mathbb{N}$ and $\mathbb{N}^n$ is given in Exercise 3.14.

The following lemma will rule our abilities to construct c.e. sets:

Lemma 3.8 *Let $A \subseteq \mathbb{N}^n$ and $B \subseteq \mathbb{N}^n$ be c.e. Then*

a) $A \cap B$ and $A \cup B$ are both g.e.

b) If $n = m + k$ and both m and k are positive, then

$$\{\vec{x} \mid \exists \vec{y}(\vec{x}, \vec{y}) \in A\}$$

will be c.e., where $\vec{x}$ is a sequence of variables of length m and $\vec{y}$ is of length k.

Moreover, every computable set is c.e., the inverse image or direct image of a c.e. set using a partial computable function will be c.e.

All these claims follow trivially from the definition or from one of the characterisations in Theorem 3.9.

Let us introduce another standard piece of notation:

Definition 3.14 Let

$$W_e = \{n \mid \phi_e(n)\downarrow\} = \{n \mid \exists t T(e, n, t)\}.$$

Let

$$W_{e,m} = \{n \mid \exists t < m T(e, n, t)\}.$$

We let $\mathcal{K}$ be the *diagonal set*

$$\mathcal{K} = \{e \mid e \in W_e\}.$$

Lemma 3.9 *a) $\{(e, n) \mid n \in W_e\}$ is c.e.*

b) $\mathcal{K}$ is c.e.

c) Each set $W_{e,m}$ is finite.

d) $\{(e, n, m) \mid n \in W_{e,m}\}$ is primitive recursive.

All proofs are trivial.

Selection with consequences

From now on we will prove lemmas and theorems in the lowest relevant dimension, but clearly all results will hold in higher dimensions as well.

Theorem 3.10 (The Selection Theorem)
Let $A \subseteq \mathbb{N}^2$ be c.e. Then there is a partial computable function f such that

i) $f(n)\downarrow \Leftrightarrow \exists m(n, m) \in A$.

ii) If $f(n)\downarrow$ then $(n, f(n)) \in A$.

Proof
This time we will give an intuitive proof. Let A be the projection of the primitive recursive set $B \subseteq \mathbb{N}^3$ (characterisation iii).). For each n, search for the least m such that $(n, \pi_1(m), \pi_2(m)) \in B$, and then let $f(n) = \pi_1(m)$.

Intuitively we perform a parallell search for a witness to the fact that $(n, m) \in A$ for some m, and we choose the m with the least witness.

Corollary 3.3 *Let A and B be two c.e. sets. Then there are disjoint c.e. sets C and D with*

$$C \subseteq A,\ D \subseteq B \text{ and } A \cup B = C \cup D.$$

Proof
Let $E = (A \times \{0\}) \cup (B \times \{1\})$ and let f be a selection function for E.
Let $C = \{n \mid f(n) = 0\}$ and $D = \{n \mid f(n) = 1\}$.
Then C and D will satisfy the properties of the corollary.

Corollary 3.4 *A set A is computable if and only A and the complement of A are c.e.*

One way is trivial, since the complement of a computable set is computable and all computable sets are c.e. So assume that A and its complement B are c.e. Let E be as in the proof of the corollary above, and f the selection function. Then f is the characteristic function of A, so A is computable.

Corollary 3.5 *Let $f : \mathbb{N} \to \mathbb{N}$ be a partial function. Then the following are equivalent:*

i) f is computable.

ii) The graph of f is c.e.

Proof
If the graph of f is c.e., then f will be the selection function of its own graph, which is computable by the selection theorem. If f on the other hand is computable, then the graph of f will be the domain of the following function $g(n, m)$: Compute $f(n)$ and see if the result equals m.

Computably inseparable c.e. sets

In Exercise 3.16 we will see that two disjoint complements of c.e. sets can be separated by a computable set. Here we will show that a similar separation property does not hold for c.e. sets, and we will draw some consequenses of this fact.

Definition 3.15 Let A and B be two disjoint subsets of $\mathbb{N}$. We say that A and B are *computably separable* if there is a computable set C such that $A \subseteq C$ and $B \cap C = \emptyset$. Otherwise A and B are *computably inseparable.*

Theorem 3.11 *There is a pair of computably inseparable c.e. sets.*

Proof
Let $A = \{e \mid \phi_e(e) = 0\}$ and $B = \{e \mid \phi_e(e) = 1\}$.
Assume that C is a computable set that separates A and B, and assume that e_0 is an index for the characteristic function of C.
Then, if $e_0 \in C$, $\phi_{e_0}(e_0) = 1$. Then $e_0 \in B$ which is disjoint from C.
Likewise, if $e_0 \not\in C$ we show that $e_0 \in A \subseteq C$. In both cases we obtain a contradiction, so the existence of C is impossible.

Now this theorem has some interesting consequences concerning the difference between classical and constructive mathematics. We will end our general introduction to the basics of computability theory discussing some of the consequences.

Definition 3.16 a) A *binary tree* is a non-empty set D of finite 0-1 -sequences such that any initial segment of an element in D is also in D. A binary tree is computable if the set of sequence numbers of the sequences in D is computable.

b) An *infinite branch* in a binary tree D is a function $f : \mathbb{N} \to \{0, 1\}$ such that $(f(0), \ldots, f(n-1)) \in D$ for all n.

Lemma 3.10 (König's Lemma)
An infinite binary tree has an infinite branch.

The proof of König's lemma is trivial, but has very little to do with computability theory. One constructs a branch by allways extending the sequence in a direction where the tree is still infinite.
Well known theorems proved by similar arguments will be that a continuous function on a closed bounded interval will obtain its maximum and that any consistent first order theory has a complete extension.

Remark 3.7 The version of König's Lemma given above, restricting ourselves to binary trees, is often called *Weak Königs lemma*. The full lemma then refers to infinite trees with finite branching, not just binary branching. There are results showing that Weak König's lemma is equivalent, relative to some very weak theory, to the mentioned theorems from analysis and topology.

We will show a failure of a constructive version of König's lemma:

Lemma 3.11 *There is an infinite, computable binary tree without a computable, infinite branch.*

Proof
Let A and B be two computably inseparable c.e. sets and let $\{A_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$ be primitive recursive sequences of sets A_n and B_n contained in $\{0, \ldots, n-1\}$, with $A = \bigcup_{n\in\mathbb{N}} A_n$ and $B = \bigcup_{n\in\mathbb{N}} B_n$.
If σ is a 0-1 -sequence of length n, we let $\sigma \in D$ if for all $i < n$: $i \in A_n \Rightarrow \sigma(i) = 1$

and $i \in B_n \Rightarrow \sigma(i) = 0$. σ will be a characteristic function on $\{0, \ldots, n-1\}$ separating A_n and B_n.
Now the characteristic function of any set separating A and B will be an infinite branch in D. Since A and B are disjoint, D will be an infinite, binary tree. Moreover, D will be primitive recursive. A computable, infinite branch will on the other side be the characteristic function of a computable set separating A and B, something assumed not to exist. This ends the proof.

3.3 Degrees of Unsolvability

3.3.1 m-reducibility

Discussion

Having introduced the main concepts of computability theory there are several directions to move in. One direction will be to give a further analysis of the computable functions and subclasses of them. This will lead us to complexity theory or to subrecursive hierarchies (see Section 3.6. One key concept then will be that one set or function can be reduced to another in some simple way.
We will not consider such a fragmentisation of the computable functions into interesting subclasses yet. Instead we will ask for reductions between possible non-computable functions and sets, using the complexity of computability itself for defining the notion of reduction. The connection between this area and the ones mentioned above is that both the formation of interesting concepts and the methodology of those directions of computability theory are based on the experience gained from investigating computable reductions in general. Thus, in an introductory course, where one should learn the computability theorists way of thinking, this section is basic.

Definition 3.17 Let A and B be two subsets of $\mathbb{N}$.
We say that A *is m-reducible to* B, $A <_m B$, if there is a total computable function f such that for all n:

$$n \in A \Leftrightarrow f(n) \in B.$$

We read this as 'many-one-reducible', since f may be a many-one function. If we insist on f being injective, we will get 1-reducibility.

There is no way we can reduce $\mathbb{N}$ to $\emptyset$ and vice versa, and we will exclude those *trivial* sets from our further discussion. We then get the observations:

Lemma 3.12 *a) If A is computable and $B \neq \mathbb{N}, \emptyset$, then $A <_m B$.*

b) $A <_m B \Leftrightarrow (\mathbb{N} \setminus A) <_m (\mathbb{N} \setminus B)$.

c) $<_m$ is transitive.

The proofs are easy and are left for the reader.

Definition 3.18 We call two sets A and B m-equivalent if $A <_m B$ and $B <_m A$. We then write $A \equiv_m B$.

Clearly $\equiv_m$ will be an equivalence relation with a partial ordering inherited from $<_m$. We call the set of equivalence classes with this induced ordering *the m-degrees*. The m degrees have some properties easy to establish:

Lemma 3.13 *a) Let X be a finite set of m-degrees. Then X has a least upper bound.*

b) Let X be a countable set of m-degrees. Then X has an upper bound.

Proof
The computable sets will form the minimal m-degree. Thus the empty set has a least upper bound. In order to prove the rest of a) it is sufficient to show that $\{A, B\}$ will have a least upper bound. Let

$$A \oplus B = \{2n \mid n \in A\} \cup \{2n + 1 \mid n \in B\}.$$

This will define the least upper bound, see Problem 3.20.
In order to prove b), Let $\{A_n \mid n \in \mathbb{N}\}$ be a countable family of sets. Let

$$A = \Sigma_{n \in \mathbb{N}} A_n = \{\langle n, m\rangle \mid m \in A_n\}.$$

Then the m-degree of A will bound the m-degrees of A_n for each $n \in \mathbb{N}$.

We will invest most of our efforts in analysing the Turing degrees below. We will however prove one negative result, the m-degrees do not represent a linear stratification of all nontrivial sets into complexity classes.

Lemma 3.14 *There are two nontrivial sets A and B such that $A \not<_m B$ and $B \not<_m A$.*

Proof
We will construct two increasing sequences $\{\sigma_k\}_{k \in \mathbb{N}}$ and $\{\tau_k\}_{k \in \mathbb{N}}$ of 0-1 -functions approximating the characteristic functions of A and B resp. The construction will be in steps:
Let $\sigma_0 = \tau_0$ be the empty sequence.
Step $k = 2e$: If ϕ_e is not total, increase the lengths of σ_k and τ_k by one, letting $\sigma_{k+1}(x) = 1$ for the new entry x and $\tau_{k+1}(y) = 0$ for the new entry y, this to ensure that the sets we construct are nontrivial. If ϕ_e is total, let x be the least element for which $\sigma_k(x)$ is undefined.
We then extend σ_k to σ_{k+1} and τ_k to τ_{k+1} such that $\sigma_{k+1}(x)$ and $\tau_{k+1}(\phi_e(x))$ both are defined, and

$$\sigma_{k+1}(x) \neq \tau_{k+1}(\phi_e(x)).$$

This will ensure that A is not reducible to B via ϕ_e.
In step $k = 2e + 1$ we ensure in a similar way that B is not reducible to A via ϕ_e. This ends the proof of the lemma.

An m-complete c.e. set

We proved that the computable inverse of a c.e. set is c.e. A reformulation of this will be

Lemma 3.15 *Let $A <_m B$. If B is c.e., then A is c.e.*

Thus the c.e. sets form an initial segment of all sets preordered by m-reductions. First we will show that this set has a maximal element:

Lemma 3.16 *Let $\mathcal{K} = \{e \mid \phi_e(e)\downarrow\}$. Then any other c.e. set is m-reducible to $\mathcal{K}$.*

Proof
Let $A = \{d \mid \phi_e(d)\downarrow\}$. Adding a dummy variable we can without loss of generality assume that $A = \{d \mid \phi_e(d,x)\downarrow\}$ where the outcome of $\phi_e(d,x)$ is independent of x. Then $A <_m \mathcal{K}$ by

$$A = \{d \mid S^1_1(e,d) \in \mathcal{K}\}.$$

A natural question is now if there are c.e. sets that are not m-equivalent to $\mathcal{K}$ or the computable sets; are there more m-degrees among the c.e. sets? This was answered by Emil Post, when he classified a whole class of c.e. sets 'in between'.

Simple sets

Definition 3.19 A c.e. set A is *simple* if the complement of A is infinite, but does not contain any infinite c.e. sets.

A simple set cannot be computable (why?). We will prove that there exist simple sets, and that $\mathcal{K}$ cannot be reduced to any simple sets.

Lemma 3.17 *There exists a simple set.*

Proof
Let $B = \{(e,x) \mid 2e < x \wedge \phi_e(x)\downarrow\}$.
Let g be a computable selection function for B, i.e. $g(e)\downarrow$ when $(e,x) \in B$ for some x, and then $g(e)$ selects one such x.
Let A be the image of g. Then A is c.e.
Since $g(e) > 2e$ when defined, the complement of A will be infinite.
If W_e is infinite, W_e will contain a number $> 2e$, and $g(e)$ will be defined. Then $A \cap W_e \neq \emptyset$. This shows that A is simple.

Lemma 3.18 *Let $\mathcal{K} <_m A$.*
Then the complement of A contains an infinite c.e. set.

Proof
Let f be total and computable such that

$$e \in \mathcal{K} \Leftrightarrow f(e) \in A.$$

We will construct a computable sequence $\{x_i\}_{i\in\mathbb{N}}$ of distinct numbers outside A, and use the recursion theorem to glue the whole construction together.
The induction start will be the empty sequence.
Assume that $B_n = \{x_0, \dots x_{n-1}\}$ has been constructed, B_n disjoint from A.
Let $\rho(n)$ be such that $W_{\rho(n)} = \{e \mid f(e) \in B_n\}$. By the uniformity of the construction, ρ will be computable. We will let $x_n = f(\rho(n))$.
If $x_n \in B_n$ we have that $x_n \not\in A$, so $\rho(n) \not\in \mathcal{K}$, and $\rho(n) \not\in W_{\rho(n)}$. This contradicts the assumption that $x \in B_n$ and the definition of $W_{\rho(n)}$. Thus $x_n \not\in B_n$.
On the other hand, if $x_n \in A$, then $\rho(n) \in \mathcal{K}$, $\rho(n) \in W_{\rho(n)}$ and by construction, $x_n \in B_n$, which we agreed was impossible. Thus x_n is a new element outside A.
We can then continue the process and produce an infinite c.e. set outside A.

Corollary 3.6 *There is a non-computable c.e. set A such that $\mathcal{K}$ is not m-reducible to A.*

3.3.2 Turing degrees

Relativised computations

In the previous section we considered a notion of reducibility between sets base on the idea 'A is simpler than B if we can gain information about A by making one computable call about membership in B'. Now, if we imagine that we are given the superhuman ability to decide membership in B, we must also permit ourselves to ask more intricate questions about B in order to decide membership in A, and still claim that we use our superhuman power to make A decidable. There are several notions of reductions based on the idea that we may be allowed any computable set of questions about membership in B and draw the conclusions from the answers. We will not consider any of those intermediate notions of reduction in this introductory course, but take the most drastic approach: We extend the definition of the computable functions by adding some extra functions as initial ones. This is technically described in the following definition:

Definition 3.20 Let $f_1, \dots, f_k$ be a finite list of partial functions $f_i : \mathbb{N} \to \mathbb{N}$.

a) We extend the definition of $\phi_e(\vec{x})$ to a definition of

$$\phi_e^{f_1,\dots,f_k}(\vec{x})$$

by adding a nine'th clause
ix) If $e = \langle 9, i, k\rangle$ then $\phi_e^{f_1,\dots,f_k}(x, \vec{x}) = f_i(x)$. If $f_i(x)$ is undefined, then this computation does not terminate.

b) The computation tree will in this case be $\langle 9, i, k, x, f_i(x)\rangle$.

We call these algorithms *relativised algorithms* meaning that the concept of computation has been relativised to $f_1, \ldots, f_k$. One important aspect of this definition is the *finite use principle.* Even if we in reality accept functions as inputs in algorithms as well, no terminating algorithm will use more than a finite amount of information from the functions involved.

Lemma 3.19 *Assume that $\phi_e^{f_1,\ldots,f_k}(\vec{x}) = z$. Then there are finite partial sub-functions $f'_1, \ldots, f'_k$ of $f_1, \ldots, f_k$ resp. such that*

$$\phi_e^{f'_1,\ldots,f'_k}(\vec{x}) = z$$

with the same computation tree.

The proof is by a tedious but simple induction on e, using the fact that a sequence number is larger than all parts of the sequence.

We have given and used an enumeration of all ordered sequences of natural numbers. In the same fashion we can construct an enumeration $\{\xi_i\}_{i\in\mathbb{N}}$ of all finite partial functions such that the following relations will be primitive recursive:
$\xi_i(x)\downarrow$
$\xi_i(x) = y$
$dom(\xi_i) \subseteq \{0, \ldots, n\}$.

The reader is free to ad any other important properties, as long as they are correct. The properties above are sufficient to prove the extended Kleene T-predicate:

Lemma 3.20 *For each n and k the following relation is primitive recursive:*
$T_{n,k}(e, \vec{x}, i_1, \ldots, i_k, t) \Leftrightarrow t$ is the computation tree of $\phi_e^{\xi_{i_1},\ldots,\xi_{i_k}}(\vec{x})$.

In order to save notation we will from now on mainly consider computations relativised to one function. Using the coding

$$\langle f_0, \ldots, f_{k-1}\rangle(x) = f_{\pi_1(x)(mod\ \ k)}(\pi_2(x))$$

we see that there is no harm in doing this.

Lemma 3.21 *There is a primitive recursive function c such that for any partial functions f, g and h, if for all x, $f(x) = \phi_e^g(x)$ and for all x, $g(x) = \phi_d^h(x)$, then for all x, $f(x) = \phi_{c(e,d)}^h(x)$.*

Proof
$c(e, d)$ is defined by primitive recursion on e, dividing the construction into cases i) - xi).
For cases i) - iii), we let $c(e, d) = e$.
For the cases iv) - viii) we just let $c(e, d)$ commute with the construction of e from its subindices, i.e., if the case is $e = expr(e_1, \ldots, e_n)$ then $c(e, d) =$

$expr(c(e_1, d), \ldots, c(e_n, d))$, where $expr$ can be any relevant expression.
If $e = \langle 9, 1, 1\rangle$ we have $\phi^g_e(x, \vec{x}) = g(x) = \phi^h_d(x)$, so we let $c(e, d) = d'$ where $\phi^h_{d'}(x, \vec{x}) = \phi^h_d(x)$. d' is primitive recursive in d.

We can use the concepts introduced here to talk about computable *functionals*, and not just about computable functions.

Definition 3.21 Let $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$, a functional of type 2.
We call F *computable* if there is an index e such that for all total $f : \mathbb{N} \to \mathbb{N}$ we have

$$F(f) = \phi^f_e(0).$$

All computable functionals of type 2 will be continuous with respect to the canonical topologies. Kleene extended the notion of relativised computation to cover all functionals of any finite type as possible inputs. This will be discussed briefly in Chapter 4.

Turing Degrees

We will now restrict ourselves to total functions.

Definition 3.22 Let f and g be two functions. We say that f *is computable in* g if there is an index e such that for all x,

$$f(x) = \phi^g_e(x).$$

We write $f <_T g$. This relation is also called *Turing reducible to.*

The key properties of Turing reducibility is given by

Lemma 3.22 *a)* $<_T$ *is a transitive relation.*

b) If f is computable and g is any function, then $f <_T g$.

c) If f and g are functions, there is a function h such that for any other function h':

$f <_T h$ and $g <_T h$.
If $f <_T h'$ and $g <_T h'$ then $h <_T h'$.

Proof
a) is a consequence of Lemma 3.21, b) is trivial, and to prove c), let
$h(2n) = f(n)$ and $h(2n + 1) = g(n)$.

Definition 3.23 Let f and g be two functions. f and g are *Turing equivalent*, in symbols $f \equiv_T g$, if $f <_T g$ and $g <_T f$.

$\equiv_T$ will be an equivalence relation. The equivalence classes will be called *Turing degrees* or *Degrees of unsolvability.* We will simply call them *degrees.* We will let bold-face low case letters early in the alphabet , **a**, **b**, etc. denote degrees. The set of degrees has a canonical ordering $<$ inherited from $<_T$.

We can summarize what we have observed so far in

Lemma 3.23 *The ordered set of degrees is an upper semilattice with a least element such that every countable set is bounded, and every initial segment is countable.*

We leave the verifications for the reader. There is no maximal degree:

Lemma 3.24 *Let* $\mathbf{a}$ *be a degree. Then there is a degree* $\mathbf{b}$ *such that*

$$\mathbf{a} < \mathbf{b}.$$

Proof
Let $f \in \mathbf{a}$. Let $g(x) = \phi^f_{\pi_1(x)}(\pi_2(x)) + 1$ if $\phi^f_{\pi_1(x)}(\pi_2(x))\downarrow$, otherwise $g(x) = 0$. The proof of the unsolvability of the halting problem can be relativised to f so g is not computable in f. On the other hand, clearly f is computable in g.

The g constructed in the proof above is called f', *the jump of* f. The jump operator is indeed a degree-operator, see Exercise 3.21.

We have shown that there are incomparable m-degrees. The same method can be used to show that there are incomparable Turing degrees, see Exercise 3.22. We will prove a stronger result, showing that no strictly increasing sequence of degrees will have a least upper bound.

Theorem 3.12 *Let* $\{\mathbf{a}_i\}_{i\in\mathbb{N}}$ *be a strictly increasing sequence of degrees. Then there are two degrees* $\mathbf{b}$ *and* $\mathbf{c}$ *that are both upper bounds for the sequence, such that for any degree* $\mathbf{d}$*, if* $\mathbf{d} < \mathbf{b}$ *and* $\mathbf{d} < \mathbf{c}$*, then* $\mathbf{d} < \mathbf{a}_i$ *for some* $i \in \mathbb{N}$.

A pair $\mathbf{b}, \mathbf{c}$ as above is called *a perfect pair* for the sequence. The degrees in a perfect pair for a sequence will be incomparable. Further, the existence of a perfect pair shows that the sequence will have no least upper bound.

Proof
Let $\{f_i\}_{i\in\mathbb{N}}$ be a sequence of total functions of increasing Turing degrees. We will construct the functions g and h as the limits of approximations g_x and h_x, where we in the construction of g_{x+1} and h_{x+1} want to ensure that if $e_1 = \pi_1(x)$ and $e_2 = \pi_2(x)$ and $\phi^g_{e_1} = \phi^h_{e_2}$ are total, then $\phi^g_{e_1}$ is recursive in f_x. In order to simplicate notation, we let g and h be defined on $\mathbb{N}^2$, but this will alter nothing. In the construction we will preserve the following properties:

1. If $i < x$, then $g_x(i,n)$ and $h_x(i,n)$ are defined for all n.
2. If $i < x$, then $g_x(i,n) = f_i(n)$ for all but finitely many n.
3. If $i < x$, then $h_x(i,n) = f_i(n)$ for all but finitely many n.
4. $g_x(i,n)$ is defined only for finitely many (i,n) with $x \leq i$.
5. $h_x(i,n)$ is defined only for finitely many (i,n) with $x \leq i$.

This will ensure that g_x and h_x are equivalent to f_{x-1} for $x \geq 0$.
Let g_0 and h_0 both be the empty function.
Now, let $x \geq 0$ and assume that g_x and h_x are defined satisfying 1. - 5. above.

Let $e_1 = \pi_1(x)$ and $e_2 = \pi_2(x)$. What we will do next will depend on the answer to the following question:
Can we find an x and finite extensions g' of g_x and h' of h_x such that $\phi_{e_1}^{g'}(x) \neq \phi_{e_2}^{h'}(x)$ and both are defined?.
If the answer is 'no', we extend g_x to g_{x+1} by letting $g_{x+1}(x,n) = f_x(n)$ whenever this is not in conflict with the construction of g_x (a conflict that can appear at at most finitely many places), and we construct h_{x+1} from h_x and f_x in the same way.
If the answer is 'yes', we first choose two such finite extensions, and then we construct g_{x+1} and h_{x+1} from these extensions as above. This ends our construction.

We let $g(x,n) = g_{x+1}(x,n)$ and $h(x,n) = h_{x+1}(x,n)$. In the construction we have tried as hard as possible to avoid that $\phi_e^g = \phi_d^h$. The point is that we have tried so hard that if they after all turn out to be equal, they will both be computable in one of the f_i's.
Claim 1
f_i is computable in both g and h.
Proof
We have that $f_i(n) = g(i,n)$ except for finitely many i, so f_i is computable in g. The same argument holds for h.
Claim 2
For $x \geq 0$ we have that g_x and h_x both are computable in f_{x-1}
Proof
This is a trivial consequence of properties 1. - 5.
Claim 3
If $\phi_e^g = \phi_d^h$ and both are total, then ϕ_e^g is computable in f_x for some x.
Proof
Let $x = P(e,d)$ where P is the pairing from Definition 3.7. Then in the construction of g_{x+1} and h_{x+1} we ask for a y and finite extensions g' and h' of g_x and h_x such that $\phi_e^{g'}(y) \neq \phi_d^{h'}(y)$. If we had found some, we would let g and h be further extensions of one such pair of finite extensions, and then we would have preserved that $\phi_e^g(y) \neq \phi_d^g(y)$, contradicting our assumption on e and d. On the other hand, for every y we can, by the assumption and the finite use principle, find finite extensions such that $\phi_e^{g'}(y)\downarrow$ and $\phi_d^{h'}(y)\downarrow$. The point is that all these values must be equal, otherwise we could have found two extensions giving different values. Thus we can give the following algorithm for computing $\phi_e^g(y)$ from g_x which again is recursive in f_{x-1}: Search for any finite extension (by searching through the finite partial functions consistent with g_x) g' of g_x such that $\phi_e^{g'}(y)\downarrow$ The value we obtain will be the correct value.
This ends the proof of our theorem.

3.4 A minimal degree

3.4.1 Trees

In the constructions of degrees we have performed so far, we have been using brute force. For instance, when we want to construct a minimal pair, we start with three properties:

1. f is not computable.
2. g is not computable
3. If h is computable in both f and g, then h is computable.

We then split these properties into infinite lists of *requirements*, which we try to satisfy during the construction:

$R_{1,e}$ If ϕ_e is total, then $f \neq \phi_e$.

$R_{2,e}$ If ϕ_e is total, then $g \neq \phi_e$.

$R_{3,e,d}$ If $\phi_e^f = \phi_d^g$ is total, then ϕ_e^f is computable.

Now, for any of these requirements and any pair σ and τ of finite sequences there will be finite extensions σ' and τ' such that any further total extension f and g of σ' and τ' resp. will satisfy the requirement. Thus by a step-by-step construction we can construct f and g via finite approximations satisfying one requirement at the time. The reader is invited to work out the full proof, see Exercise 3.23.

We will now face a problem which we cannot solve by this simple method. We will show that there is a non-computable function f such that there is no function of complexity strictly between f and the computable functions. Again we will set up the relevant properties of f, and fragmentise them into a sequence of requirements we want to satisfy during the construction. The problem will be that we cannot ensure that these requirements are satisfied by considering just a finite approximation of f. Instead we will use trees, and we will satisfy the various requirements by insisting that f is a branch in a given binary tree. Before we can go into details with the argument, we will reconsider our definition of a binary tree, give a formulation that will be handy for this particular application. We will not distinguish between finite sequences and sequence numbers here, but whenever we say that a function defined from a set of finite sequences to the set of finite sequences is computable, we mean that the corresponding function on sequence numbers is computable.

Definition 3.24 Let D be the set of finite 0-1 -sequences.

a) If $\sigma \in D$ we let $\sigma * 0$ and $\sigma * 1$ be σ extended by 0 or 1 resp. We extend this to the concatenation $\sigma * \tau$ in the canonical way.

b) If σ and τ are two sequences, and $i < lh(\sigma)$, $i < lh(\tau)$ and $\sigma(i) \neq \tau(i)$, we say that σ and τ are *inconsistent*.

c) $f : D \to D$ is *monotone* if $f(\sigma)$ is a proper subsequence of $f(\tau)$ whenever σ is a proper subsequence of τ.

d) *A tree* is a monotone function $T : D \to D$ mapping inconsistent sequences to inconsistent sequences.

e) If S and T are trees, then S is a *subtree of* T if there is a tree T' such that $S = T \circ T'$.

f) If T is a tree and $f : \mathbb{N} \to \{0, 1\}$, then f *is a branch in* T if for every n there is a sequence σ of length n such that $T(\sigma)$ is an initial segment of g.

Remark 3.8 These trees will sometimes be called *perfect trees*, the set of branches will form a perfect subset of the set $\{0, 1\}^{\mathbb{N}}$ in the topological sense, i.e. a set that is closed and without isolated points.
Our intuition should be focused on the set $Set(T)$ of sequences that are initial seqments of the possible $T(\sigma)$'s. This will be a binary tree in the traditional sense, and we will have the same set of infinite branches. If S is a subtree of T, then $Set(S) \subseteq Set(T)$. The converse will not hold in general.

One key lemma is a simple topological result:

Lemma 3.25 *Let $\{T_n\}_{n\in\mathbb{N}}$ be a sequence of trees such that T_{n+1} is a subtree of T_n for all n. Then there is a function f that is a branch in all trees T_n.*

Proof
Consider the set X of σ such that $\sigma \in Set(T_n)$ for all n.
X will be a binary tree. The empty sequence is in X. If $\sigma \in X$, then for all n, $\sigma * 0 \in Set(T_n)$ or $\sigma * 1 \in Set(T_n)$. At least one of these has to hold for infinitly many n, and since we are dealing with subtrees, for all n. Thus X is not a finite tree and by König's Lemma has an infinite branch, which will be a common branch for all T_n's.

Remark 3.9 Using topology we might just say that the intersection of a decreasing sequence of nonempty compact sets is nonempty.

We will now see that certain computable trees can be used to meet natural requirements. As a first case, let us prove:

Lemma 3.26 *Let T be a computable tree and assume that ϕ_e is total. Then there is a computable subtree S of T such that ϕ_e is not a branch in S.*

Proof
If ϕ_e is not a branch in T, we can use $S = T$. If ϕ_e is a branch in T, one of $T(0)$ and $T(1)$ will be inconsistent with ϕ_e, since they are inconsistent themselves. (here 0 is the sequence of length 1 with entry 0). Assume that ϕ_e is inconsistent with $T(0)$.
Let $S(\sigma) = T(0 * \sigma)$. Then S is a subtree as desired. The other case is analogue.

3.4.2 Collecting Trees

Now we will describe a property on computable trees that will ensure that if f is a branch in the tree and ϕ_e^f is total, then ϕ_e^f is computable.

Definition 3.25 Let T be a computable tree, e an index.
T is *e-collecting* if for all finite sequences σ, τ and all $x \in \mathbb{N}$, if $\phi_e^{T(\sigma)}(x)\downarrow$ and $\phi_e^{T(\tau)}(x)\downarrow$, then

$$\phi_e^{T(\sigma)}(x) = \phi_e^{T(\tau)}(x).$$

Lemma 3.27 *Let T be a computable e-collecting tree and let f be a branch in T. If ϕ_e^f is total, then ϕ_e^f is computable.*

Proof
We will give an algorithm for computing $\phi_e^f(x)$ from x.
Since $\phi_e^f(x)\downarrow$, there will be a 0-1 -sequence σ such that $\phi_e^{T(\sigma)}(x)\downarrow$, and since T is e-collecting, the value $\phi_e^{T(\sigma)}(x)$ will be independent of the choice of σ. Thus our algorithm will be:
Search for a finite sequence σ such that $\phi_e^{T(\sigma)}(x)\downarrow$ and let the answer be the output of our algorithm.

Remark 3.10 There is more information to be gained from this proof. We see that the function ϕ_e^f itself is independent of f as long as it is total and f is a branch in an e-collecting tree.

3.4.3 Splitting Trees

We will now find a criterion that will ensure that f is computable in ϕ_e^f whenever f is a branch in a computable tree and ϕ_e^f is total:

Definition 3.26 Let T be a computable tree, and let e be an index.
We call T *e-splitting* if for all finite 0-1 -sequences σ and τ, if σ and τ are inconsistent, then there exists a number x such that

$$\phi_e^{T(\sigma)}(x)\downarrow\ ,\ \phi_e^{T(\tau)}(x)\downarrow \text{ with } \phi_e^{T(\sigma)}(x) \neq \phi_e^{T(\tau)}(x).$$

Lemma 3.28 *Let T be an e-splitting computable tree , let f be a branch in T and assume that ϕ_e^f is total. Then f is computable in ϕ_e^f.*

Proof
We will compute an infinite 0-1 -sequence $\{k_i\}_{i\in\mathbb{N}}$ from ϕ_e^f such that $T(\sigma_n)$ is an initial segment of f for all n, where $\sigma_n = (k_0, \ldots, k_{n-1})$. The empty sequence σ_0 of course satisfies this. Assume that σ_n is constructed. Then one of $T(\sigma_n * 0)$ and $T(\sigma_n * 1)$ will be an initial seqment of f. We will just have to determine which one. Now $\phi_e^{T(\sigma_n*1)}$ and $\phi_e^{T(\sigma_n*1})$ will be inconsistent, so exactly one of them will be inconsistent with ϕ_e^f. We can then use ϕ_e^f to find the inconsistent one, which means that we can decide which direction is along f and which is not. This provides us with the induction step, and we can move on. This ends the proof of the lemma.

Remark 3.11 In the case of T being an e-splitting tree, we see that ϕ_e^f is a one-to one-function of the branch f, and what we just argued for is that we can compute the inverse.

3.4.4 A minimal degree

Using Lemmas 3.25, 3.26, 3.27 and 3.28 we can show the existence of a minimal degree from the following:

Lemma 3.29 *Let T be a computable tree and let e be an index. Then there is a computable subtree S that is either e-collecting or e-splitting.*

Proof
Case 1: There is a sequence σ such that for all τ and τ' extending σ, $\phi_e^{T(\tau)}$ and $\phi_e^{T(\tau')}$ are equal where both are defined. Let $S(\tau) = T(\sigma * \tau)$. Then S will be a subtree of T and S will be e-collecting.
Case 2: Otherwise. Then for every σ there will be extensions τ_0 and τ_1 such that $\phi_e^{T(\tau_0)}$ and $\phi_e^{T(\tau_1)}$ are inconsistent. Further, we can find these τ_0 and τ_1 as computable functions $t_0(\sigma)$ and $t_1(\sigma)$.
We then define the subtree S by

$S(()) = T(())$, i.e. S and T are equal on the empty sequence.

If $S(\sigma)$ is defined, let $S(\sigma * 0) = T(t_0(\sigma))$ and $S(\sigma * 1) = T(t_1(\sigma))$.

This defines a subtree that will be e-splitting.

We have now proved all essential steps needed in the construction of a minimal degree:

Theorem 3.13 *There is a non-computable function f such that if $g <_T f$ then either g is computable or g is equivalent to f.*

Proof
Using the lemmas above we construct a family $\{T_n\}$ of computable trees such that T_{n+1} is a subtree of T_n for all n, and such that for all e

If ϕ_e is total, then ϕ_e is not a branch in T_{2e+1}

T_{2e+2} is either e-collecting or e-splitting.

Then by Lemma 3.25 there is an f that is a branch in all T_n's, and this f will have the property wanted.

3.5 A priority argument

3.5.1 C.e. degrees

In the constructions of minimal pairs and functions of minimal degrees we have not been concerned with the complexity of the sets and functions constructed. We can decide upper bounds on the objects constructed in the proofs by analysing the complexity of properties like ϕ_e is total and counting in depth how many number quantifiers we will need in order to write out a definition of the object constructed. If we, however, are interested in results about degrees with some bounded complexity, we must be more careful in our constructions. In this section we will be interested in degrees with at least one c.e. set in it:

Definition 3.27 Let $\mathbf{a}$ be a degree.
$\mathbf{a}$ is an *c.e. degree* if $\mathbf{a}$ contains the characteristic function of a c.e. set. We say that f is *of c.e. degree* if the degree of f is a c.e. degree.

There is a nice characterisation of the functions of c.e. degree. We leave the proof as an exercise for the reader, see Exercise 3.26

Theorem 3.14 *Let f be a function. Then the following are equivalent:*

i) f is of c.e. degree.

ii) There is a primitive recursive sequence $\{f_i\}_{i\in\mathbb{N}}$ converging pointwise to f such that the following function

$$g(x) = \mu n.\forall m \geq n(f_m(x) = f(x))$$

is computable in f.

3.5.2 Post's Problem

So far we only know two c.e. degrees, $\mathbf{O}$, the degree of the computable sets and functions, and $\mathbf{O}'$, the degree of the halting problem. Post's Problem asks if there are more c.e. degrees than those two. This is of course a nice, technical problem, but it has implications beyond that. One of the reasons why c.e. sets are so interesting is that the set of theorems in an axiomatisable theory is c.e. If there were no more c.e. degrees than those two known to us, a consequence would be that there are two kinds of axiomatisable theories, those that are decidable and those that share the complexity of Peano Arithmetic. As a consequence of Gödel's proof of the incompleteness theorem, the set of Gödel-numbers of theorems in Peano Arithmetic is a complete c.e. set, i.e. of even the same m-degre as $\mathcal{K}$.

Now, in 1957 two young mathematicians, Friedberg and Muchnic, independently constructed c.e. sets of inbetween degrees. They both developed what is now known as the priority method . The problem we have to face when constructing c.e. sets is that we must give an argorithm for adding elements to the set, but we cannot give an algorithm for keeping objects out of the set. If we did that, the set constructed would become computable. Thus when we have made an attempt to approximate a set with positive and negative information, we must be allowed to violate the negative information. However, we must not violate the negative information to such an extent that we ruin our global goal. We solve this dilemma by introducing priorities to our requirements. If an effort to satisfy one requirement will ruin the attempt to satisfy another requirement, we let the requirement with highest priority winn. This idea will work when two properties are satisfied by the construction: If we make an attempt to satisfy a requirement and we never ruin this attempt, we actually manage to satisfy the requirement. Further, if we after a stage in the construction never make an attempt to satisfy a requirement, the requirement will authomatically be satisfied.

Thus we are bound to satisfy the requirement of highest priority, either because we make an attemp which will not be ruined, or because there is no need to make an attempt.
Then we are bound to satisfy the next requirement, either because we make an attempt after the final attempt for the first requirement, or because there is no need to make such an attempt,
and so on.... In the finite injury lemma we will give a full proof along this line of arguing.

3.5.3 Two incomparable c.e. degrees

Theorem 3.15 *There are two c.e. sets A and B that are not recursive in each other.*

Remark 3.12 If A and B are not computable in each other, neither can be computable, because any computable set will be computable in any set. Moreover neither can have the same degree as $\mathcal{K}$, because every r.e. set is computable in $\mathcal{K}$. Thus we have not just produced an in-between degree, but two in-between degrees. In Exercise 3.27 we will see that there are infinitly many in-between degrees, and that any countable partial ordering can be embedded into the ordering of the c.e. degrees.

Proof
We will construct two c.e. sets A and B satisfying

R_{2e}: $\mathbb{N} \setminus A \neq W_e^B$

R_{2e+1}: $\mathbb{N} \setminus B \neq W_e^A$

or in other terms: The complement of A is not c.e. relative to B and vice versa. If we achieve this for all e, we will have proved the theorem, since a set A is computable in B if and only if both A and the complement of A are c.e. in B, and since A is c.e. we have that A is computable in B if and only if the complement of A is c.e. in B.

We will construct two primitive recursive increasing sequences $\{A_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$ of finite sets . We let

$$A_0 = B_0 = \emptyset.$$

We call each step in the process a *stage.* If $n = \langle 1, e, x\rangle$ we will consider to make an attempt to satisfy R_{2e} at stage n, while if $n = \langle 2, e, x\rangle$ we will consider to make an attempt to satisfy R_{2e-1}.
An attempt to satisfy R_{2e} will consist of selecting a $q \in A_{n+1} \cap W_e^{B_{n+1}}$, and then put up a *protection*, the set of points used negatively in the verification of $q \in W_e^{B_{n+1}}$. If we can keep all objects in this protection out of B throughout the construction, we will have

$$q \in A \cap W_e^B$$

and R_{2e} will be satisfied. We call the protection *active* at a later stage m if B_m is disjoint from this protection.

There is some little minor trick to observe, we will use disjoint infinite supplies of numbers that we may put into A (or B) in order to satisfy R_{2e} (or R_{2e+1}). We will use this to show that if we make only finitly many attempts, we will succeed after all.

Now let $n = \langle 1, e, x\rangle$ and assume that A_n and B_n are constructed. Assume further that we constructed certain protections, some of them active at stage n, others not. We write the following procedure for what to do next:
Let $B_{n+1} = B_n$.
Question 1: Is there a protection for R_{2e} active at stage n?
If the answer is 'yes', let $A_{n+1} = A_n$. and continue to the next stage.
If the answer is 'no', ask
Question 2: Is there a $y < n$ such that $\phi_{e,n}^{B_n}(\langle y, e\rangle)\downarrow$ and y is in no active protection for any requirement R_{2d+1} where $2d + 1 < 2e$?
If the answer is 'no', let $A_{n+1} = A_n$ and proceed to the next stage.
If the answer is 'yes', choose the least y, let $A_{n+1} = A_n \cup \{\langle y, e\rangle\}$, construct a protection $\{0, \dots, n\} \setminus B_n$ for R_{2e} and move on to the next stage.

If $n = \langle 2, e, x\rangle$ we act in the symmetric way, while for other n we just move on to the next stage, not adding anything to A or B.
This ends the construction.

Claim 1 *(The Finite Injury Lemma)*
For each requirement R_s there is a stage n_s after which we do not put up or injure any protection for R_s.
Proof
We prove this by induction on s, and as an induction hypothesis we may assume that there is a stage m_s after which we never put up a protection for any requirement R_t with $t < s$. Then we will never after stage m_s injure a protection for R_s. Thus if we never put up a protection for R_s after stage m_s we can let $n_s = m_s$, while if we construct a protection, this will never be injured and we can let n_s be the stage where this protection is constructed.

Now let $A = \bigcup_{n\in\mathbb{N}} A_n$ and $B = \bigcup_{n\in\mathbb{N}} B_n$. Then A and B are c.e. sets.
Claim 2
Each requirement R_s will be satisfied.
Proof
We prove this for $s = 2e$. There are two cases.
1. There is a protection for R_s active at stage n_s.
If this is the case, there will be a y such that $\langle y, e\rangle \in A_{n_s} \cap W_e^{B_n}$. Since the protection is not injured, we will have that $\langle y, e\rangle \in A \cap W_e^B$ and the requirement is satisfied, A and W_e^B are not complementary.
2. There is no such protection.
There is only finitely many objects of the form $\langle y, e\rangle$ in A, because we ad at most one such object for each stage before n_s, and never any at a stage after n_s.
On the other hand, there can be only finitly many objects of the form $\langle y, s\rangle$ in W_e^B, since otherwise we could choose one that is not in any protection for any R_t for $t < s$, and sooner or later we would at some stage after n_s make a new attempt to satisfy R_e, which we are not. Thus $A \cup W_e^B$ contains only finitly many objects of the form $\langle y, e\rangle$ and the sets are not the complements of each other. Thus the requirement will be satisfied in this case as well.

We have shown that all the requirements are satisfied in this construction, so the theorem is proved.

3.6 Subrecursion theory

3.6.1 Complexity

What is to be considered as complex will be a matter of taste. Actually, the same logician may alter her/his taste for complexity several times a day. This logician may give a class on aotomata theory in the morning, and then the regular languages will be the simple ones, while context free languages are more complex. Still, context free languages are decidable in polynomial time, and while our logician spends an hour contemplating on the **P** = **NP**-problem any context free language is much more simple than the satisfiability problem for propositional logic. If our logician is working mainly in classical computability theory, all decidable languages are simple, while the undecidable ones are the complex ones. If our logician is a set theorist, all definable sets are simple, may be all subsets of subsets of subsets of $\mathbb{R}$ are simle, we have to move high up in cardinality in order to find sets of a challenging and interesting complexity.
In this section, our view on complexity will be one shared by many proof theorists. One of the aims in proof theory is to characterise the functions and sets provably computable in certain formal theories extending elementary number theory. The idea is that if we know the functions provably computable in T, we know something about the strength of the theory T worth knowing.

We will not be concerned with proof theory in this book, and any references to proof-theoretical results should not be considered as a part of any curriculum based on this text.

3.6.2 Ackermann revisited

In Section 3.2.1 we defined the Ackermann branches. The idea of Ackermann was that each use of the scheme for primitive recursion involves an *iteration* of a previously defined function. Then diagonalising over a sequence of functions defined by iterated iteration would break out of the class of primitive recursive functions.

The Ackermann branches were defined using functions of two variables. We will be interested in pushing his construction through to the transfinite level, in order to describe more complex functions from below. Then it will be convenient, from a notational point of view, to use functions of one variable.

Definition 3.28

- Let $F_0(x) = x + 1$
- Let $F_{k+1}(x) = F_k^x(x)$

For the sake of illustration, let us compute $F_2(2)$:

$$F_2(2) = F_1(F_1(2)) = F_1(F_0^2(2)) = F_1(4) = F_0^4(4) = 8.$$

The reader is challenced to show that $F_2(x) = x \cdot 2^x$. In Exercise 3.30 we will see that diagonalising over the F_k's will lead us outside the class of primitive recursive functions.

From now on in this section we will let PA be Peano Arithmetic; i.e. elementary number theory with the first order induction axiom scheme. Our first order language L will be the language of PA.

Definition 3.29 A function $f : \mathbb{N} \to \mathbb{N}$ is *provably computable* if there is a formula $A(x, y)$ in L with only bounded quantifiers such that A defines the graph of f and

$$PA \vdash \forall x \exists y B(x, y).$$

This definition is extended in the obvious way to functions of several variables.

We will assume that the reader is familiar with the method of arithmetisation, sequence numbering and so forth.

Lemma 3.30 *Let $f(k, x) = F_k(x)$. Then f is provably computable.*

Outline of proof
Essentially we have to prove that F_0 is total and that if F_k is total then F_{k+1} is total as well. This induction step is proved by induction, where we actually must prove

$$\forall x \forall y \exists z F_k^y(x) = z$$

by induction on y.
In order to write a complete proof, we have to write down the formal definition of the graph of f and the establish a proof three for the formula required to be a theorem in PA. This is tedious and not very exiting; the exiting part is to decide how much induction that is required.

3.6.3 Ordinal notation

The concept of *an ordinal number* will be properly introduced in a course on set theory. We have defined the concept of well orderings. An ordinal number will be a set that in a canonical way represents an isomorphism class of well orderings. For our purposes it will be sufficient to think about order types of well orderings, and we will use a term language for such order types.

Definition 3.30 ω is the smallest infinite ordinal number, and it denotes the ordertype of $\mathbb{N}$ with the standard ordering.

All natural numbers will be ordinals as well.
We may extend the arithmetical operations *plus*, *times* and *exponentiation* to operations on orderings, or actually, on order types. The sum of two orderings $\langle A, <_A \rangle$ and $\langle B, <_B \rangle$ will be the set

$$C = A \oplus B = (\{0\} \times A) \cup (\{1\} \times B)$$

with the ordering $<_C$ defined by

- $\langle 0, a \rangle <_C \langle 1, b \rangle$ whenever $a \in A$ and $b \in B$.
- $\langle 0, a_1 \rangle <_C \langle 0, a_2 \rangle$ if and only if $a_1 <_A a_2$.
- $\langle 1, b_1 \rangle <_C \langle 1, b_2 \rangle$ if and only if $b_1 <_B b_2$.

The product of two orderings will for our purposes be the antilexicographical ordering on the product of the domains.

The formal definition of the exponential of orderings is less intuitive, but it helps to think of exponentiation as iterated multiplication. Our definition only works for special orderings $\langle A, <_A \rangle$, including all well orderings.

Definition 3.31 Let $\langle A, <_A \rangle$ and $\langle B, <_B \rangle$ be two orderings where A has a least element a_0.
Let $p \in C$ if p is a map from B to A such that $p(b) = a_0$ for all but finitely many $b \in B$.
If $p \neq q$ are in C, there will be a maximal argument b such that $p(b) \neq q(b)$.
We then let $p <_C q \Leftrightarrow p(b) < q(b)$ for this maximal b.

Lemma 3.31 *If $\langle A, <_A \rangle$ and $\langle B, <_B \rangle$ are two well orderings then the sum, product and exponential will be well orderings.*

The proof is left as Exercise 3.31

As a consequence, every arithmetical expression in the constant ω and constants for the natural numbers, and using 'plus', 'times' and 'exponents' will have an interpretation as an ordinal number.
The least ordinal number that cannot be described by an expression as above is baptized ϵ_0. As for ordinary arithmetics, $\omega^0 = 1$. This is a special case of the more general rule

$$\omega^\alpha \cdot \omega^\beta = \omega^{\alpha+\beta}.$$

A consequence is that each ordinal $\alpha < \epsilon_0$ can be written in a unique way as

$$\alpha = \omega^{\alpha_n} + \cdots + \omega^{\alpha_0}$$

where $\{\alpha_0, \cdots, \alpha_n\}$ is an increasing (not neccessarily strictly) sequence of ordinals less than α. We call this the *Cantor Normal Form* of α. If $\alpha_0 = 0$ the ordinal α will be a *successor ordinal*, otherwise it will be a *limit ordinal.*

We extended the Ackermann hierarchy to the first transfinite level by diagonalising over the Ackermann branches. One advantage with the Cantor normal form is that we may find a canonocal increasing unbounded sequence below every limit ordinal between ω and ϵ_0. Actually, it is possible to do so for ordinals greater than ϵ_0 too, but readers interested in how this is done and why are reccommended to follow a special course on proof theory and ordinal denotations.

Definition 3.32 Let

$$\alpha = \omega^{\alpha_n} + \cdots + \omega^{\alpha_0}$$

be given on Cantor normal form, and assume that $\alpha_0 > 0$.
We define the n'th element $\alpha[n]$ of the fundamental sequence for α as follows:

Case 1 $\alpha_0 = \beta + 1$:
Let $\alpha[n] = \omega^{\alpha_n} + \cdots \omega^{\alpha_1} + n \cdot \omega^{\beta}$.

Case 2 α_0 is a limit ordinal:
We may then assume that $\alpha_0[n]$ is defined, and we let

$$\alpha[n] = \omega^{\alpha_n} + \cdots \omega^{\alpha_1} + \omega^{\alpha_0[n]}$$

We may consider the map $\alpha \mapsto \alpha[n]$ as an extension of the predecessor function to limit ordinals:

Definition 3.33 Let $0 < \alpha < \epsilon_0$. We define the *n-predecessor of* α by

a) If α is a successor ordinal, the n-predecessor of α is the predecessor of α.

b) If α is a limit ordinal, the n-predecessor of α will be $\alpha[n]$.

c) We say that $\beta <_n \alpha$ if β can be reached from α by iterating the n-predecessor map.

Lemma 3.32 *Let $\alpha < \epsilon_0$, $\beta <_m \alpha$ and $m < n$. Then $\beta <_n \alpha$*

Proof
It is sufficient to prove that if $m < n$ and α is a limit ordinal, then $\alpha[m] <_n \alpha$. This is left as a non-trivial exercise for the reader, see Exercise 5.7.

Lemma 3.33 *Let $\alpha < \epsilon_0$ and let $\beta < \alpha$. Then there is an n such that $\beta <_n \alpha$.*

Proof
This is proved by induction on α with the aid of Lemma 3.32. The details are left as a nontrivial exercise for the reader, see Exercise 5.7

3.6.4 A subrecursive hierarchy

We will now extend the alternative Ackermann hierarchy to all ordinals less than ϵ_0:

Definition 3.34 Let $\alpha < \epsilon_0$. We define $F_\alpha(x)$ by recursion on α as follows:

- $F_0(x) = x + 1$
- $F_{\beta+1}(x) = F_\beta^x(x)$
- $F_\alpha(x) = F_{\alpha[x]}(x)$ when α is a limit ordinal.

Proof theorists have shown that if a function $f : \mathbb{N} \to \mathbb{N}$ is provably computable, then f will be bounded almost everywhere by one of the F_α's where $\alpha < \epsilon_0$. The proof involves the translation of a proof in PA to a proof in ω-logic, then a cut-elimination procedure in ω-logic and finally an analysis of cut-free proofs in ω-logic of the totality of computable functions. This analysis is far beyond the scope of this volume, and the reader is *not* recommended to work out the details.
In Exercises 5.8 and Exercise 5.7 the reader is challenced to prove that each F_α is provably computable and establish the hierarchial properties of the $\{F_\alpha\}_{\alpha<\epsilon_0}$-hierarchy. The main results are:

Lemma 3.34 *Let $n < x$ and $\beta <_n \alpha < \epsilon_0$.*
Then

$$F_\beta(x) \leq F_\alpha(x).$$

Theorem 3.16 *If $\beta < \alpha < \epsilon_0$, then*

$$\exists x_0 \forall x > x_0 (F_\beta(x) < F_\alpha(x)).$$

Remark 3.13 There are many aspects about subrecursive hierarchies that we have not discussed in this section. We have not discussed complexity classes. For instance, the class H_α of functions computable in polynomial time relative to a finite iteration of F_α represents a stratification of the set of all provably computable functions into a complexity hierarchy, where each complexity class is closed under composition and closed under polynomial time reductions. We will not discuss such matters further here.

3.7 Exercises

Ex 3.1 Prove that the following functions are primitive recursive:

a) $f(x, y, \vec{z}) = x + y$

b) $f(x, y, \vec{z}) = x \cdot y$

c) $f(x, y, \vec{z}) = x^y$

d) $f(x, \vec{y}) = x!$

e) $f(x, \vec{y}) = x \dot{-} 1$

f) $f(x, y, \vec{z}) = x \dot{-} y$

g) $f(x, y, \vec{z}) = 0$ if $x = y$

$f(x, y, \vec{z}) = 1$ if $x \neq y$

h) $f(x, y, \vec{z}) = 0$ if $x < y$

$f(x, y, \vec{z}) = 1$ if $x \geq y$

Ex 3.2 Prove the following facts:

a) $\mathbb{N}^n$ and $\emptyset$ are primitive recursive as subsets og $\mathbb{N}^n$.

b) The complement of a primitive recursive set is primitive recursive. Moreover, the union and intersection of two primitive recursive subsets of $\mathbb{N}^n$ will be primitive recursive.

Ex 3.3 a) Prove Lemma 3.2.

b) Prove that we can replace the inequalities by strict inequalities in Lemma 3.2.

Ex 3.4 Prove Lemma 3.3.

Ex 3.5 a) Prove Lemma 3.5.

b) Prove that the sequence numbering is monotone in each coordinate.

c) Prove that the monotone enumeration SEQ of the sequence numbers is primitive recursive.
Hint: Find a primitive recursive bound for the next sequence number and use bounded search.

d) Define an alternative sequence numbering as follows:
$\langle\langle x_0, \ldots, x_{n-1} \rangle\rangle$ is the number z such that

$$SEQ(z) = \langle x_0, \ldots, x_{n-1} \rangle.$$

Show that this alternative numbering is surjective and still satisfies Lemma 3.5

e) Prove that the pairing function P in Definition 3.7 is 1-1 and onto.

Ex 3.6 Let f_n be the n'the Ackermann branch.

a) Prove that for any primitive recursive function h there are numbers n and m such that $h(x) < f_n(m, x)$ for all x.

b) Show that the diagonal

$$f(x) = f_x(x, x)$$

is not primitive recursive.

Ex 3.7 Prove that if $\phi_e(\vec{x}) = y$ and $\phi_e(\vec{x}) = z$, then $y = z$.
Hint: Use induction on e.
Discuss why this is something that needs a proof.

Ex 3.8 Show that in the definitions of the primitive recursive functions and the recursive functions we can replace ii) and vi) by:
ii*) $I_{i,n}(x_1, \ldots, x_n) = x_i$ is primitive recursive (recursive)
for all i and n.

Ex 3.9 Let $\{f_n\}_{n \in \mathbb{N}}$ be a sequence of total functions such that

$$g(n, m) = f_n(m)$$

is computable.
Show that each f_n is computable, and that there is a total computable function not in the sequence.
Hint: Use a diagonal argument.

Ex 3.10 Show that there is a total computable function $\Phi(n, x)$ of two variables that enumerates all primitive recursive functions of one variable.
Is it possible to let Φ be primitive recursive?

Ex 3.11 Complete the proof of Lemma 3.7

Ex 3.12 Prove Corollary 3.2

Ex 3.13 a) Prove that every non-empty c.e. set is the image of a primitive recursive function (easy) and that every infinite c.e. set is the image of an injective computable function (not that easy, but still...).

b) Prove that the range of a strictly increasing total computable function is computable.

Ex 3.14 Let $A \subseteq \mathbb{N}^n$. Show that A is c.e. (by characterisation ii) or iii) in Theorem 3.9) if and only if

$$\{\langle x_1, \ldots, x_n \rangle \mid (x_1, \ldots, x_n) \in A\}$$

is c.e.

Ex 3.15 Give an explicit description of the selection function in the proof of Theorem 3.10.

Ex 3.16 Let A and B be two disjoint sets whose complements are c.e.
Show that there is a computable set C such that $A \subseteq C$ and $B \cap C = \emptyset$.
Hint:Use Corollary 3.3.

Ex 3.17 Let L be the language of propositional calculus over an infinite set $\{A_i\}_{i\in\mathbb{N}}$ of propositional variables. Discuss the following statement:
There is a primitive recursive consistent set of propositions with no computable completion.

Ex 3.18 a) Show that there is an enumeration $\{I_n\}_{i\in\mathbb{N}}$ of all rational intervals contained in $[0,1]$ such that the relations $I_n \subseteq I_m$, $I_n \cap I_m = \emptyset$ and $|I_n| < 2^{-m}$ are computable, where $|I_n|$ is the length of the interval.

A real number x is *computable* if there is a computable function h such that

i) $|I_{h(n)}| < 2^{-n}$

ii) For all n, $x \in I_{h(n)}$

b) Let $f : [0,1] \to [0,1]$ be a continuous function.
We say that f is *computable* if there is a total computable function $g : \mathbb{N} \to \mathbb{N}$ such that

i) $I_n \subseteq I_m \Rightarrow I_{g(n)} \subseteq I_{g(m)}$

ii) $x \in I_n \Rightarrow f(x) \in I_{g(n)}$

iii) For all x and m there is an n with $x \in I_n$ and $|I_{g(n)}| < 2^{-m}$.

Show that any computable function will be continuous. Show that there is a computable function that does not take its maximal value at any computable real.

Ex 3.19 Show that there is a non-trivial set A that is not m-reducible to its complement.
Hint: Let $\{f_n\}_{n\in\mathbb{N}}$ be an enumeration of all the total computable functions. Approximate A and its complement by finite characteristic functions to ensure at step n that f_n will not reduce A to its complement.

Ex 3.20 Show that if $A \equiv_m B$ and $C \equiv_m D$, then $A \oplus C \equiv_m B \oplus D$. Show that if $A <_m E$ and $B <_m E$ then $A \oplus B <_m E$. Show that $A \oplus B$ then represents the least upper bound of A and B in the m-degrees.

Ex 3.21 Show that if $f_1 \equiv_T f_2$, then the jumps f_1' and f_2' are also Turing equivalent.

Ex 3.22 Perform a direct construction and show that there are two total functions f and g such that $f \not<_T g$ and $g \not<_T f$.

Ex 3.23 Prove that there is a minimal pair of Turing degrees, i.e. a pair $\{\mathbf{a}, \mathbf{b}\}$ of degrees of non-computable functions, such that the degree $\mathbf{O}$ of computable functions is the greatest lower bound of $\mathbf{a}$ and $\mathbf{b}$.
Hint: We may use the main idea in the proof of Theorem 3.12, this theorem is simpler to prove. You may also get some ideas from the discussion of this proof in the text.

Ex 3.24 Let $\mathbf{O}$ be the degree of the computable functions.
Recall the definition of the jump operator cfr. Exercise 3.21. We define the arithmetical hierarchy as follows:

A Σ^0_1-set is an c.e. set (of any dimension).

A Π^0_1-set is the complement of a Σ^0_1-set

A Σ^0_{k+1}-set is the projection of a Π^0_k-set

A Π^0_{k+1}-set is the complement of a Σ^0_{k+1}-set

A Δ^0_k set is a set that is both Σ^0_k and Π^0_k.

Let $\mathbf{O}^{(n)}$ be the degree obtained from $\mathbf{O}$ using the jump-operator n times.

a) Prove that if A is Σ^0_k or A is Π^0_k, then (the characteristic function of) A has a degree $\mathbf{a} \leq \mathbf{O}^{(k)}$.

b) Show that for $k \geq 1$ we have that A is Δ^0_{k+1} if and only if the degree of A is bounded by $\mathbf{O}^{(k)}$.
Hint: Use the relativised version of the fact that a set is computable if and only if both the set and its complement are c.e.

Ex 3.25 Show that there are continuumly many minimal degrees.
Hint: Instead of constructing one tree T_n at level n we might construct 2^n trees $T_{\sigma,n}$ for $lh(\sigma) = n$, ensuring for each e that the branches of different trees will not be computable in each other via index e. The proof requires some clever book-keeping.

Ex 3.26 Prove Theorem 3.14.

Ex 3.27 Let $B \subseteq \mathbb{N}^2$ be a set. For each $n \in \mathbb{N}$, we let $B_n = \{m \mid (n,m) \in B\}$ and we let $B_{-n} = \{(k,m) \in B \mid k \neq n\}$.

a) Show that there is a c.e. set B such that for all n, B_n is not computable in B_{-n}.
Hint: Use an enumeration of $\mathbb{N}^2$ to give all the requirements

$$R_{(n,e)}: \ \mathbb{N} \setminus B_n \neq W_e^{B_{-n}}$$

a priority rank.

b) Consider a computable partial ordering $\prec$ on the natural numbers.
Show that there is an order-preserving map of $\prec$ into the c.e. degrees.
Hint: Use the construction in a), and let
$C_n = \{(k,m) \in B \mid k \preceq n\}$.

There is one computable partial ordering $\prec$ such that any other partial ordering of any countable set can be embedded into $\prec$, see Exercise 5.3. Thus this shows that any countable partial ordering can be embedded into the ordering of the c.e. degrees.

Ex 3.28 Fill in the details in the proof of the following theorem:

Theorem 3.17 *Let* $\mathbf{a} > \mathbf{O}$ *be an c.e. degree. Then there are two incomparable c.e. degrees* $\mathbf{b}$ *and* $\mathbf{c}$ *such that* $\mathbf{a} = \mathbf{b} \oplus \mathbf{c}$.

This theorem is called *The splitting theorem*. We split the c.e. degree $\mathbf{a}$ into two simpler c.e. degrees.
Proof
Let A be a nonrecursive r.e. set. It is sufficient to construct two disjoint r.e. sets B and C such that
$A = B \cup C$, A is not recursive in B and A is not recursive in C.
Let f be a 1-1 enumeration of A. At each stage n we will put $f(n)$ into B or $f(n)$ into C, but not into both.
Let B_n be the numbers put int B before stage n and C_n be the set of numbers put into C before stage n. Further, we let
$A_n = \{f(0), \ldots, f(n-1)\}$, so $A_n = B_n \cup C_n$.
We put up requirements

$R_{2e} \quad : K_A \neq \phi_e^B.$

$R_{2e+1} : K_A \neq \phi_e^C.$

which we give priorities in the usual way.
For each requirement R_s we define three auxilliary functions. For $s = 2e$ they will be:
The match function $m(s,n) = \mu k < n . \forall x \leq k \phi_{e,n}^{B_n}(x) = K_{A_n}(x)$.
The bar function $b(s,n) = max\{m(s,n') \mid n' \leq n\}$
The protection function $p(s,n) = \{y \mid y$ is used negativly in computing $\phi_{e,n}^{B_n}(x)$ for some $x \leq b(s,n)\}$.
In this case we call this *a protection of* B.

Now the construction at stage n is as follows: If $f(n) \notin p(s,n)$ for any $s \leq n$, put $f(n)$ into B. Otherwise, consider the requirement R_s of highest priority such that $f(n) \in p(s,n)$. If this is a protection of B, we put $f(n)$ into C, otherwise we put $f(n)$ into B.
When we put an element into a protection of B we *injure* that requirement. We will prove that for any requirement R_s there is a stage n_s after which we will never injure that requirement, and simultaneously that the bar-function $b(s,n)$ is bounded when s is fixed.
Assume that this holds for all $s' < s$. Then there is a stage n_s after which $f(n)$ is not in the protection for any $s' < s$, and then, after stage n_s, R_s will not be injured.
This in turn means that if $x < b(s,n)$ for $n \geq n_s$ and $\phi_{e,n}^{B_n}\downarrow$, then $\phi_e^B(x) = \phi_{e,n}^{B_n}$.
Now, if $\lim_{n\to\infty} p(s,n) = \infty$ we can use the increasing matching and the stability of $\phi_{e,n}^{B_n}$ to show that A is recursive, which it is not.
On the other hand, if $K_A = \phi_e^B$ we will get increasing matching. Thus at the same time we prove that the construction of the bar and protection for R_s terminates and that the requirement is satisfied at the end.

Ex 3.29 Post hoped to prove that a simple set cannot be of the same degree as the complete c.e. set K. This will however not be the case, which will be clear when you have solved this problem.

Let A be a non-computable c.e. set and f a total computable 1-1-enumeration of A. We know that f cannot be increasing (why?).
Let

$$B = \{n \mid \exists m > n(f(m) < f(n))\}$$

This is called the *deficiency set* of the enumeration.

a) Show that B is c.e. and that B is computable n A.

b) Show that A is computable in B.
Hint: In order to determine if $x \in A$ it is sufficient to find $n \notin B$ such that $f(n) > x$.

c) Show that B is simple.
Hint: If the complement of B contains an infinite c.e. set, the algorithm for computing A from B in b) can be turned into an algorithm for computing A.

Ex 3.30 Let F_k be the alternative Ackermann branch. Show that every primitive recursive function f is bounded almost everywhere by some F_k and that the diagonal function $F_\omega(x) = F_x(x)$ is not primitive recursive.

Ex 3.31 Prove Lemma 3.31

Ex 3.32 Let $\mathcal{P}$ be the set of polynomials $P(x)$ where we only use $+$ (not 'minus') and where all coefficients are natural numbers.
We order $\mathcal{P}$ by

$$P(x) \prec Q(x) \Leftrightarrow \exists n \forall m \geq n(P(m) < Q(m)).$$

Show that $\prec$ is a well ordering of ordertype ω^ω.

Chapter 4

Generalized Computability Theory

Chapter 5

Non-trivial exercises and minor projects

In this chapter we have collected some exercises or minor projects that is based either on the material in more than one chapter or that is too hard to be considered as an exercise in the ordinary sense. We call them exercises, though some of the items in this chapter are much more demanding than ordinary exercises.

Ex 5.1 Let L be a finite language. Show that the relation 'ϕ is true in all finite L-structures' is decidable if all predicates are unary, while that it in general is complete co-c.e.

Ex 5.2 Let L be a finite language void of constants and function symbols. Let T be an open theory over L. Show that there is an ω-categorical extension T^* of T.
Show that if T has only finitely many non-logical axioms, then T^* may be chosen to be decidable.
Discuss how your general construction is related to the step from the open theory of total orderings to DO.

Ex 5.3 In view (and improvement) of Exercise 5.2 show that there is a computable partial ordering $(\mathbb{N}, \prec)$ such that every other countable partial ordering $(X', <')$ can be embedded into $(\mathbb{N}, \prec)$.

Ex 5.4 For the sake of notational simplicity we extend the second order languages with variables for functions as well.

a) Find a second order formula that expresses that if X is a set, $f : X \to X$ and $x \in X$ then $Y = \{f^n(x) \; ; \; n \in \mathbb{N}\}$.

b) Discuss to what extent our formal definition of a finite set is in accordance with your intuition about set theory.

c) It is a fact of set theory that every set accepts a total ordering. Use this and our formal definition of infinity to show that the following are equivalent.

   1. X is finite.
   2. Every total ordering of X is a well ordering.

.

Ex 5.5 Let $L_=$ be the language of equality. We say that s subset of $\mathbb{N}$ is *definable in second order finitary logic* if there is a sentence ϕ in $L^2_=$ such that $n \in A$ if and only if $\{0, \ldots, n-1\} \models \phi$.
Discuss the properties of the class of sets definable in second order finitary logic in terms of decidability, complexity and closure properties.

Ex 5.6 Let Σ be a fixed alphabet. Show that there is an enumeration $\{M_i\}_{i\in\mathbb{N}}$ of the set of Turing Machines over Σ and a one-to-one enumeration $\{w_j\}_{j\in\mathbb{N}}$ of the set of words in Σ^* such that the function f satisfying the equation

$$f(i,j) \sim k \Leftrightarrow M_i(w_j) = w_k$$

is computable, where $\sim$ means that either is both sides undefined, or both sides are defined and with the same value.

Ex 5.7 In the text we have indicated how to establish some of the properties of the hierarchy $\{F_\alpha\}_{\alpha<\epsilon_0}$. Work out all the details and write a small essay about it. You may well combine this with Exercise 5.8

Ex 5.8 Show that if $\alpha < \epsilon_0$ then F_α is provably computable. Suggest a fundamental sequence for ϵ_0 and thus a function F_{ϵ_0}. If properly defined, F_{ϵ_0} will not be provably computable in PA, but in some seccond order extension. Discuss how we may formulate a proof of the totality of F_{ϵ_0} in some 2. order extension of PA.

Chapter 6

Appendix: Some propositions from a beginners course in logic

Proposition 1 The Completenes Theorem, two versions

a) Let T be a first order theory. Then T is consistent if and only if T has a model.

b) Let T be a first order theory over the language L and let ϕ be a formula in L. Then

$$T \vdash \phi \Leftrightarrow T \models \phi$$

Proposition 2 The Compactness Theorem
Let T be a first order theory. Then T has a model if and only if each finite subtheory T_0 of T has a model.

Proposition 3 The Deduction Theorem
Let T be a first order theory over a language L, let ϕ be a sentence in L and ψ a formula in L. Then

$$T, \phi \vdash \psi \Leftrightarrow T \vdash \phi \to \psi.$$

Proposition 4 The Theorem of Constants
Let T be a first order theory over a language L and let $c_1, \ldots, c_n$ be constants that are not in L. Let L' be the extended language.
Let T' be the theory over L' with the same nonlogical axioms as T, and let ϕ be a formula in L. Then for all lists $x_1, \ldots, x_n$ of variables

$$T \vdash \phi \Leftrightarrow T' \vdash \phi^{x_1,\ldots,x_n}_{c_1,\ldots,c_n}.$$

Index

Bibliography

[1] Christopher C. Leary, *A Friendly Introduction to Mathematical Logic*, Prentice hall, 2000.

[2] G. E. Sacks, *Saturated model theory*, W.A. Benjamin Inc. 1972.