Math 558, Foundations of Mathematics I
Lecture Notes

John D. Clemens

PSU Spring 2005

Contents

Part I

Computability

Chapter 1

Computable Functions

Functions

The notion of computability is fundamentally about functions. Since we will be interested in partial functions, we will not always require that each $x \in X$ have an image under the map f. When the domain of f is all of X, we say that f is a *total function*.

We will sometimes use λ-notation to define a function. For instance, writing

$$f = \lambda x_1 x_2 x_3.(x_1^2 + x_2^2 + x_3^2)$$

indicates that f is a function of the three variables x_1, x_2, and x_3, whose value on these inputs is $x_1^2 + x_2^2 + x_3^2$. More frequently we will simply write

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$$

to indicate the same thing.

We will primarily study functions of the natural numbers. For the record,

$$\mathbb{N} = \{0, 1, 2, \ldots\}.$$

We say a function is k-ary if its domain is $\mathbb{N}^k$.

1.1 Primitive Recursive Functions

We begin by considering a class of functions which may naturally be considered computable. This class contains many of the commonly used functions, but we will see that there are other functions which are intuitively computable and are not in this class.

We will start with a collection of very simple functions which are all intuitively computable. We then introduce operations for forming new functions, which preserve the property of being intuitively computable. All the functions we introduce here will be total functions.

Definition 1.1. The *initial functions* are:

1. the zero function: $Z(x) = 0$, and the 0-ary zero function 0. [1]
2. The successor function: $S(x) = x + 1$.
3. The projection functions: $P_i^k(x_1, \ldots, x_k) = x_i$ for $k \geq 1$ and $1 \leq i \leq k$.

Definition 1.2. The *primitive recursive operations* are:

- (Substitution or composition) Given $g : \mathbb{N}^m \to \mathbb{N}$ and $h_1, \ldots, h_m : \mathbb{N}^k \to \mathbb{N}$ with $m, k \geq 0$, we say that the function $f : \mathbb{N}^k \to \mathbb{N}$ is obtained from g and $h_1, \ldots, h_m$ by composition if, for all $x_1, \ldots, x_k \in \mathbb{N}$ we have

$$f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k)).$$

- (Primitive recursion) Given $g : \mathbb{N}^k \to \mathbb{N}$ and $h : \mathbb{N}^{k+2} \to \mathbb{N}$ with $k \geq 0$, we say that the function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ is obtained from g and h by primitive recursion if, for all $y, x_1, \ldots, x_k$ we have

$$\begin{aligned} f(0, x_1, \ldots, x_k) &= g(x_1, \ldots, x_k) \\ f(y+1, x_1, \ldots, x_k) &= h(y, f(y, x_1, \ldots, x_k), x_1, \ldots, x_k). \end{aligned}$$

  Note that primitive recursion produces a well-defined total function.

Example 1.3. A simple example of primitive recursion with $k = 0$ is the factorial function $f(y) = y!$. Let $g = 1$ and $h(y, z) = (y + 1) \cdot z$. Then f is obtained from g and h using primitive recursion, as

$$\begin{aligned} f(0) &= 1 \\ f(y+1) &= (y+1)! = (y+1) \cdot f(y) = h(y, f(y)). \end{aligned}$$

Definition 1.4. A function $f : \mathbb{N}^k \to \mathbb{N}$ is *primitive recursive* if it can be generated in finitely many steps from the initial functions using substitution and primitive recursion. Equivalently, the class of primitive recursive functions is the smallest class of functions containing the initial functions and closed under these two operations. We let $\mathcal{PR}$ denote the class of primitive recursive functions.

Formally, we can define a hierarchy of functions:

$$\begin{aligned} \mathcal{PR}_0 &= \text{ the initial functions} \\ \mathcal{PR}_{n+1} &= \text{ all functions obtained from functions in } \mathcal{PR}_n \\ &\qquad \text{using primitive recursive operations} \\ \mathcal{PR} &= \bigcup_{n \geq 0} \mathcal{PR}_n \end{aligned}$$

[1] A 0-ary function is one with no inputs, i.e. just a constant. This will be necessary for technical reasons.

The sequence of initial functions and primitive recursive operations we use to obtain a given primitive recursive function f is called a *primitive recursive derivation* of f.

Example 1.5. The following functions are all primitive recursive:

1. Addition: $f(x,y) = x + y$. To see this, let $g(y) = y = P_1^1(y)$ and $h(x,z,y) = z + 1 = S(z)$. Then
$$\begin{aligned} f(0,y) &= y = g(y) \\ f(x+1,y) &= x + 1 + y = (x+y) + 1 = h(x, f(x,y), y) \end{aligned}$$

2. Multiplication:
$$\begin{aligned} 0 \cdot y &= 0 \\ (x+1) \cdot y &= x \cdot y + y \end{aligned}$$

3. Exponentiation:
$$\begin{aligned} x^0 &= 1 \\ x^{(y+1)} &= x^y \cdot x \end{aligned}$$
Note that we have used recursion on the second coordinate, which does not fit our literal definition of primitive recursion; however, this is never a problem. Formally, we could define a function with the inputs flipped, and then recover the original function by composing the flipped function with projections.

4. The *predecessor* function $P(x) = \begin{cases} x - 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$.
This is given by the primitive recursion:
$$\begin{aligned} P(0) &= 0 \\ P(y+1) &= y \end{aligned}$$

5. *Restricted subtraction*: $x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$.
This is given by:
$$\begin{aligned} x \dot{-} 0 &= x \\ x \dot{-} (y+1) &= P(x \dot{-} y) \end{aligned}$$

6. $|x - y| = (x \dot{-} y) + (y \dot{-} x)$

Primitive recursive relations

We will study the computability of relations by considering their characteristic functions. Let R be a k-ary relation on $\mathbb{N}$ for some $k \geq 1$, i.e. $R \subseteq \mathbb{N}^k$. Then its characteristic function χ_R is defined as:

$$\chi_R(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } (x_1, \ldots, x_k) \in R \\ 0 & \text{if } (x_1, \ldots, x_k) \notin R \end{cases}$$

We can now say what it means for a relation to be primitive recursive.

Definition 1.6. We say a relation R on $\mathbb{N}^k$ is primitive recursive if its characteristic function χ_R is a primitive recursive function.

Example 1.7. The binary relation "$x = y$" is primitive recursive, since its characteristic function is

$$\chi_=(x, y) = 1 \dot{-} |x - y|.$$

Similarly the relations $x \leq y$, $x < y$, etc. are primitive recursive.

We have the following *closure properties* of primitive recursive functions and relations.

Lemma 1.8. *The collection of primitive recursive relations is closed under Boolean operations, i.e. if P and Q are two k-ary primitive recursive relations, then so are the relations $\neg P$, $P \wedge Q$, and $P \vee Q$.*

Proof: Let χ_P and χ_Q be the characteristic functions of P and Q. Then:

$$\begin{aligned} \chi_{\neg P} &= 1 \dot{-} \chi_P \\ \chi_{P \wedge Q} &= \chi_P \cdot \chi_Q \\ \chi_{P \vee Q} &= 1 \dot{-} (1 \dot{-} (\chi_P + \chi_Q)) \end{aligned}$$

□

Lemma 1.9. *(Iterated Sums and Products) If $f(x, y, z_1, \ldots, z_k)$ is a primitive recursive function, then so are*

$$g(y, z_1, \ldots, z_k) = \sum_{x=0}^{y-1} f(x, y, z_1, \ldots, z_k) \quad (= 0 \text{ if } y = 0)$$

$$h(y, z_1, \ldots, z_k) = \prod_{x=0}^{y-1} f(x, y, z_1, \ldots, z_k) \quad (= 1 \text{ if } y = 0)$$

Proof: We define the function g^* as

$$g^*(w, y, z_1, \ldots, z_k) = \sum_{x=0}^{w-1} f(x, y, z_1, \ldots, z_k)$$

Then g^* is obtained by primitive recursion:

$$\begin{aligned} g^*(0, y, z_1, \ldots, z_k) &= 0 \\ g^*(w+1, y, z_1, \ldots, z_k) &= g^*(w, y, z_1, \ldots, z_k) + f(w, y, z_1, \ldots, z_k) \end{aligned}$$

Hence g^* is primitive recursive, and $g(y, z_1, \ldots, z_k) = g^*(y, y, z_1, \ldots, z_k)$ is as well.

The function h is handled similarly. □

Lemma 1.10. *(Bounded Conjunction and Disjunction) If $R(x, y, z_1, \ldots, x_k)$ is a primitive recursive relation, then so are:*

$$\begin{aligned} P(y, z_1, \ldots, z_k) &\equiv \bigwedge_{x=0}^{y-1} R(x, y, z_1, \ldots, z_k) \\ Q(y, z_1, \ldots, z_k) &\equiv \bigvee_{x=0}^{y-1} R(x, y, z_1, \ldots, z_k) \end{aligned}$$

Proof: Since χ_R is primitive recursive, the previous lemma tells us that

$$\chi_P(y, z_1, \ldots, z_k) = \prod_{x=0}^{y-1} \chi_R(x, y, z_1, \ldots, z_k)$$

is primitive recursive as well, and χ_Q is handled similarly. □

Note 1.11. These are also referred to as *bounded quantification* since they are true, respectively, if the relation R holds for all x between 0 and $y - 1$, or if there exists an x in this range satisfying R.

Bounded quantification is useful in defining primitive recursive relations, since we can often bound the range of values we need to search.

Example 1.12. We can see that the relation

$$\text{Prime}(x) \equiv \ x \text{ is a prime number}$$

is primitive recursive, since we have

$$\text{Prime}(x) \equiv (x > 1) \wedge \neg \bigvee_{u=0}^{x-1} \bigvee_{v=0}^{x-1} (x = u \cdot v \wedge u > 1 \wedge v > 1)$$

and the relations $x = y$, $x < y$, etc. are primitive recursive.

The following is a similar general principle.

Lemma 1.13. *(Bounded minimization) If $R(x, y, z_1, \ldots, x_k)$ is a primitive recursive relation, then the function f is primitive recursive, where*

$$f(y, z_1, \ldots, z_k) = \begin{cases} \textit{the least } x < y \textit{such that } R(x, y, z_1, \ldots, z_k) \textit{ holds} \\ \qquad \textit{if such an } x \textit{ exists} \\ y \textit{ otherwise} \end{cases}$$

Proof: We multiply each of the possible output values by a test for whether this is the correct value:

$$f(y, z_1, \ldots, z_k) = \sum_{x=0}^{y-1} \left(x \cdot \chi_R(x, y, z_1, \ldots, z_k) \cdot \prod_{w=0}^{x-1} (1 \dot{-} \chi_R(w, y, z_1, \ldots, z_k)) \right) + y \cdot \prod_{x=0}^{y-1} (1 \dot{-} \chi_R(x, y, z_1, \ldots, z_k))$$

□

Note 1.14. The bound on the search is crucial to ensure that we produce a primitive recursive function (in fact, to ensure that we produce a total function). We will consider unbounded minimization later.

Example 1.15. Let $p(n) =$ the n-th prime, i.e. $p(0) = 2$, $p(1) = 3$, $p(2) = 5$, etc. Then the function $p(n)$ is primitive recursive. To see this, we use a theorem of Euclid which implies that $p(n+1) \leq p(n)! + 1$. We can define

$$f(y, z) = \begin{cases} \text{the least } x < y \text{ such that } x \text{ is prime and } x > z \\ \qquad \text{if such exists} \\ y \text{ otherwise} \end{cases}$$

so that f is primitive recursive. Then

$$p(0) = 2$$
$$p(n+1) = f(p(n)! + 2, p(n))$$

Example 1.16. We can see that the quotient and remainder functions for integer division are primitive recursive, where

$$\text{Quotient}(y, x) = \lfloor \frac{y}{x} \rfloor$$
$$\text{Remainder}(y, x) = y \mod x$$

i.e. $y = x \cdot \text{Quotient}(y, x) + \text{Remainder}(y, x)$. This holds since

$$\text{Quotient}(y, x) = \text{ the least } q \leq x \text{ such that } y < (q+1) \cdot x$$
$$\text{Remainder}(y, x) = y \dot{-} x \cdot \text{Quotient}(y, x)$$

Coding of finite sequences

We have been dealing with functions defined over $\mathbb{N}$; however, often we will be interested in dealing with other sorts of objects. We will handle this by coding the given objects as natural numbers. Here we consider finite sequences.

As a first example, we consider a coding of ordered pairs of natural numbers. Define

$$\text{Pair}(x, y) = 2^x \cdot 3^y$$

This allows us to code an ordered pair as a natural number (in a primitive recursive way). What is important is that we can uncode in a primitive recursive way as well: Given a code s for a pair, we have primitive recursive functions which compute the coordinates of the original pair. These are:

$$\begin{aligned}(s)_0 &= \text{ the exponent of 2 in the prime factorization of } s\\ &= \text{ the least } w < s \text{ such that Remainder}(s, 2^{w+1}) \neq 0\\ (s)_1 &= \text{ the exponent of 3 in the prime factorization of } s\end{aligned}$$

(where the bound works because $n < 2^n$ for all n, etc.)

We can generalize this to sequences of arbitrary length.

Definition 1.17. Given a finite sequence $\langle a_0, \ldots, a_{m-1}\rangle$, we define its *code* to be

$$\text{Code}(\langle a_0, \ldots, a_{m-1}\rangle) = \prod_{i=0}^{m-1} p(i)^{a_i+1}$$

Note that the empty sequence (with $m = 0$) has code 1.

Note that we really have a different coding function for each m. The exponent $a_i + 1$ is used in order to be able to determine the length of a coded sequence. We will generally simply use $\langle a_0, \ldots, a_{m-1}\rangle$ to denote the code of this sequence.

Note that we can decode in a primitive recursive way as with the pairing function. Let $f(s, i) = (s)_i$ be as follows:

$$\begin{aligned}(s)_i &= \text{the } i\text{-th digit of the sequence coded by } s\\ &= (\text{the exponent of } p(i) \text{ in the prime expansion of } s) \dot{-} 1\\ &= (\text{ the least } w < s \text{ such that Remainder}(s, p(i)^{w+1}) \neq 0) \dot{-} 1\end{aligned}$$

Most natural relations and operations on sequences are primitive recursive, such as the relation "s codes a sequence", Length(s) = the length of the sequence coded by s, the concatenation of two sequences, etc.

Note 1.18. We will often use $\langle a_0, \ldots, a_{m-1}\rangle$ to denote the code of a sequence of arbitrary length. When we do this, we mean that we apply the m-ary coding function consistent with the length of the given sequence. From the way we have chosen our coding, there will always be at most one sequence coded by a given s.

Simultaneous recursion

Definition 1.19. We say that the functions f_1 and f_2 on $\mathbb{N}$ are defined by *simultaneous recursion* from the functions g_1, g_2, h_1, h_2 if:

$$\begin{aligned} f_1(0, x_1, \ldots, x_k) &= g_1(x_1, \ldots, x_k) \\ f_2(0, x_1, \ldots, x_k) &= g_2(x_1, \ldots, x_k) \\ f_1(y+1, x_1, \ldots, x_k) &= h_1(y, f_1(y, x_1, \ldots, x_k), f_2(y, x_1, \ldots, x_k), x_1, \ldots, x_k) \\ f_2(y+1, x_1, \ldots, x_k) &= h_2(y, f_1(y, x_1, \ldots, x_k), f_2(y, x_1, \ldots, x_k), x_1, \ldots, x_k) \end{aligned}$$

The class of primitive recursive functions is closed under simultaneous recursion; this is left as an exercise. It is also closed under the following more general type of recursion:

Definition 1.20. We say that a function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ is defined by *course-of-values recursion* from $h : \mathbb{N}^{k+2} \to \mathbb{N}$ if:

$$f(y, x_1, \ldots, x_k) = h(y, \langle f(0, \vec{x}), \ldots, f(y-1, \vec{x}) \rangle, x_1, \ldots, x_k)$$

1.2 Ackermann's Function

We consider a function which will not be primitive recursive, yet is intuitively computable.

Definition 1.21. We define Ackermann's function $A(n, x)$ as follows. First, let:

$$\begin{aligned} A_0(x) &= x + 1 \\ A_{n+1}(x) &= A_n^{(x+1)}(1) \end{aligned}$$

where we mean the $x+1$-th iterate. Then define *Ackermann's Function* to be:

$$A(n, x) = A_n(x)$$

Note that these functions satisfy the following recursion:

$$\begin{aligned} A_0(x) &= x + 1 \\ A_{n+1}(0) &= A_n(1) \\ A_{n+1}(x+1) &= A_n(A_{n+1}(x)) \end{aligned}$$

Notice that each A_n is defined from the previous by primitive recursion, hence each A_n is primitive recursive. Also note that each A_n requires n iterations of primitive recursion, and that these functions have increasing growth rates. This suggests that $A(n, x)$ will not be primitive recursive (as we will see), since we would only be able to use finitely many operations of primitive recursion to define it if it were primitive recursive.

Example 1.22. We list the first few A_n functions:

$$\begin{aligned} A_0(x) &= x+1 \\ A_1(x) &= 1+(x+1) = x+2 \\ A_2(x) &= 1+2(x+1) = 2x+3 \\ A_3(x) &= 2^{x+3}-3 \\ A_4(x) &= 2 \uparrow (x+3) - 3 \text{ where } 2 \uparrow n = 2^{2^{\cdot^{\cdot^{2}}}} \ (n \text{ times}) \end{aligned}$$

Another point to note is that the type of recursive definition we have given is different from simultaneous recursion or course-of-values recursion. The difference is that we require potentially larger values of the second coordinate, whereas in all of the types of recursion we were allowed for primitive recursive functions would require that the second coordinate be fixed (or decrease).

Properties of Ackermann's function

We begin with by showing a few properties of Ackermann's function which will be used to show that it is not primitive recursive.

Lemma 1.23. *The functions A_n satisfy the following properties:*

1. *For all x, $A_n^{(x)}(1) \geq x+1$.*
2. *For all x, $x < A_n(x) < A_n(x+1)$.*
3. *For all x, $A_n(x+1) \leq A_{n+1}(x)$.*
4. *If $n < m$ then $A_n(x) < A_m(x)$ for all x.*
5. *For all x, $A_n(2x) < A_{n+2}(x)$.*

Proof: We leave (1) and (5) as exercises.

2. We use induction on n. First,

$$A_0(x+1) = x+2 > x+1 = A_0(x) > x$$

so this is true for $n=0$. Suppose it holds for n. Then:

$$A_{n+1}(x+1) = A_n(A_{n+1}(x)) > A_{n+1}(x) = A_n^{(x+1)}(1) \geq x+1 > x$$

using our inductive assumption and property (1).

3. $A_{n+1}(x) = A_n^{(x+1)}(1) = A_n(A_n^{(x)}(1)) \geq A_n(x+1)$ since A_n is increasing by (3) and $A_n^{(x)}(1) \geq x+1$ by (1).
4. This follows from (2) and (3), since $A_n(x) < A_n(x+1) \leq A_{n+1}(x)$.

□

Theorem 1.24. *For any primitive recursive function f there is an M such that for all $x_1, \ldots, x_n$,*

$$f(x_1, \ldots, x_n) < A_M(\max\{x_1, \ldots, x_n\})$$

Proof: We prove this by induction on a primitive recursive derivation of f. We begin with the initial functions.

$$\begin{aligned} Z(x) &= 0 < x + 1 = A_0(x) \\ S(x) &= x + 1 < x + 2 = A_1(x) \\ P_i^k(x_1, \ldots, x_k) &= x_i < x_i + 1 < A_0(\max\{x_1, \ldots, x_k\}) \end{aligned}$$

Next we consider composition. Let M and $N_1, \ldots, N_m$ be such that

$$\begin{aligned} g(x_1, \ldots, x_m) &< A_M(\max\{x_1, \ldots, x_m\}) \\ h_1(x_1, \ldots, x_n) &< A_{N_1}(\max\{x_1, \ldots, x_n\}) \\ &\ldots \\ h_m(x_1, \ldots, x_n) &< A_{N_m}(\max\{x_1, \ldots, x_n\}) \end{aligned}$$

Let $N = \max\{N_1, \ldots, N_m\}$, so that $h_i(x_1, \ldots, x_n) < A_N(\max\{x_1, \ldots, x_n\})$ for all $1 \leq i \leq m$. Then

$$\begin{aligned} f(x_1, \ldots, x_n) &= g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)) \\ &< A_M(A_N(\max\{x_1, \ldots, x_n\})) \\ &\leq A_{Q-1}(A_Q(\max\{x_1, \ldots, x_n\})) \\ &\quad \text{where } Q = \max\{N, M+1\} \\ &= A_Q(\max\{x_1, \ldots, x_n\} + 1) \\ &\leq A_{Q+1}(\max\{x_1, \ldots, x_n\}) \end{aligned}$$

Finally we handle primitive recursion. Let M and N be such that

$$\begin{aligned} g(x_1, \ldots, x_n) &< A_M(\max\{x_1, \ldots, x_n\}) \\ h(y, z, x_1, \ldots, x_n) &< A_N(\max\{y, z, x_1, \ldots, x_n\}) \end{aligned}$$

Let $Q = \max\{N + 1, M\}$. First we claim that

$$f(y, x_1, \ldots, x_n) < A_Q(y + \max\{x_1, \ldots, x_n\})$$

This is proved by induction on y. First,

$$\begin{aligned} f(0, x_1, \ldots, x_n) &= g(x_1, \ldots, x_n) < A_M(\max\{x_1, \ldots, x_n\}) \\ &\leq A_Q(0 + \max\{x_1, \ldots, x_n\}) \end{aligned}$$

Then, assuming this is true for some y,

$$\begin{aligned} f(y+1,\vec{x}) &= h(y, f(y,\vec{x}), x_1, \vec{x}) < A_N(\max\{y, f(y,\vec{x}), x_1, \ldots, x_n\}) \\ &< A_{Q-1}(\max\{A_Q(y+\max\{x_1,\ldots,x_n\}), y, x_1, \ldots, x_n\}) \\ &\leq A_{Q-1}(A_Q(y+\max\{x_1,\ldots,x_n\})) \\ &\quad \text{since } A_Q(y+\max\{x_1,\ldots,x_n\}) \geq y, x_1, \ldots, x_n \\ &= A_Q(y+1+\max\{x_1,\ldots,x_n\}) \end{aligned}$$

Now $y+\max\{x_1,\ldots,x_n\} \leq 2\max\{y, x_1, \ldots, x_n\}$, so

$$f(y, x_1, \ldots, x_n) < A_Q(2\max\{y, x_1, \ldots, x_n\}) < A_{Q+2}(\max\{y, x_1, \ldots, x_n\})$$

This shows that f is bounded by A_{Q+2} and finishes the case of primitive recursion. □

Note 1.25. In particular, using the properties from the lemma, for a unary primitive recursive function f there will be some n so that $A_n(x)$ is bigger than $f(x)$ for all x. We see from the proof of the theorem that this n will roughly correspond to how many steps are needed to define f. This relationship can be made more precise by studying the Axt Hierarchy and the Grzegorczyk Hierarchy of primitive recursive functions.

Corollary 1.26. *Ackermann's function is not primitive recursive.*

Proof: If Ackermann's function were primitive recursive, then so would be the function $g(x) = A(x,x) = A_x(x)$. Hence there would be some M such that, for all x, $g(x) < A_M(x)$. But then we would have $A_M(M) = g(M) < A_M(M)$, a contradiction. □

Note 1.27. We will later see that Ackermann's function is computable in a broader sense.

1.3 Register Machines

We present a second approach to characterizing computable functions. Our previous approach can be considered an *inductive definition*: We listed several simple functions which we asserted to be computable (the initial functions) and then described two closure operations which produced new computable functions from others. The next approach can be considered a *machine model*: We explicitly describe a procedure for computing a function (i.e. we make precise the notion of an algorithm), and say that a function is computable if some such procedure computes it. Here we will consider register machines. A number of other machine models are often studied, such as Turing Machines, Markov Algorithms, and other formulations of register machines; all of these turn out to give the same collection of machine-computable functions.

Definition 1.28. A *register machine* consists of a finite sequence of *registers* $R_0, \ldots, R_k$, and a finite sequence of *instructions* $I_0, \ldots, I_t$. We view each register as holding a natural number. Each instruction is of one of the three forms:

1. Increment instructions (R_i+, I_j): Add one to the contents of register R_i and proceed to instruction I_j.

2. Decrement instructions (R_i-, I_j, I_k): If register R_i contains 0, do not change it and proceed to instruction I_j; if R_i contains a number bigger than 0, subtract one from it and proceed to instruction I_k.

3. Halt instruction HALT: Stop execution of the algorithm.

A register machine operates as follows. It begins with some initial values in its registers. It then executes instructions, starting with I_0 and proceeding according to the above rules. The machine contains running until it reaches a HALT instruction (which need not happen) at which time it stops.

It will be clearer to present register machines using diagrams rather than lists of instructions. To do this, we dispense with the instruction numbers (although officially they are still numbered), only labeling I_0 with "START". We indicate the three type of instructions as in Figure 1.1. Each arrow points to another node, indicating the next instruction. The two arrows from a decrement instruction are labeled $= 0$ and > 0, according to the cases of R_i being 0 or not.

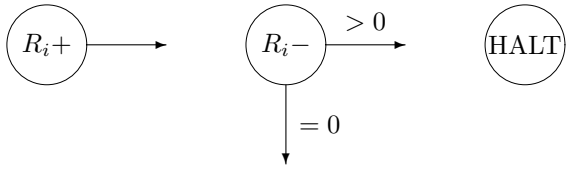


Figure 1.1: Register machine instructions

We can use register machines to compute functions of the natural numbers:

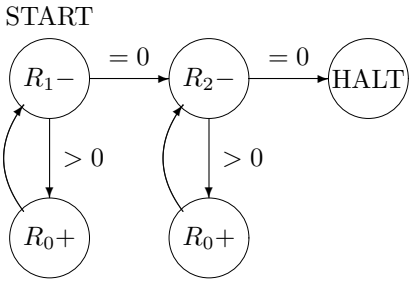
Definition 1.29. Let f be an n-ary partial function from $\mathbb{N}^n$ to $\mathbb{N}$, and let M be a register machine. We say that M *computes* f if the following hold for each n-tuple $(x_1, \ldots, x_n) \in \mathbb{N}^n$:

1. When the machine M is started running with each register R_i initially containing x_i for $1 \leq i \leq n$ and all other registers initially blank, the machine eventually halts if and only if the tuple $(x_1, \ldots, x_n)$ is in the domain of f.

2. If the machine eventually halts, it halts with the register R_0 containing the value $f(x_1, \ldots, x_n)$.

We say that a partial function is *register machine computable (RM-computable)* if there is some register machine M which computes it.

Example 1.30. We show that the addition function is register machine computable. That is, we find a function which, when started with x in R_1 and y in R_2 and all other registers blank, halts with $x + y$ in register R_0. We describe the machine with a diagram.

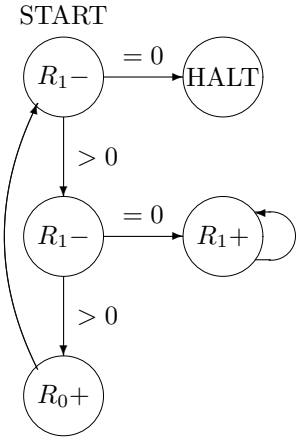


This machine proceeds by first adding the contents of R_1 to R_0, and then adding the contents of R_2 to R_0.

Note 1.31. In defining the function computed by a particular register machine, we really need to specify the arity of the function. For instance, the machine above computes the binary function of addition; however, it also computes the unary identity function, since if we specify that there is only one input then it is assumed that all other registers (in particular R_2) are initially 0. Hence, we need to refer to the k-ary function computed by the machine, and each machine computes k-ary functions for all k.

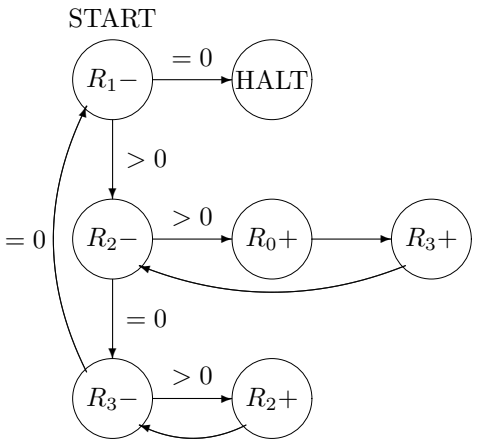
Example 1.32. We describe a register machine to compute the following partial function:

$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ \text{undefined} & \text{if } n \text{ is odd} \end{cases}$$



This machine successively subtracts 2 from R_1 and adds 1 to R_0; if it empties R_1 after an even number it halts; otherwise it goes into an infinite loop and never halts.

Example 1.33. We construct a register machine to compute the multiplication function:



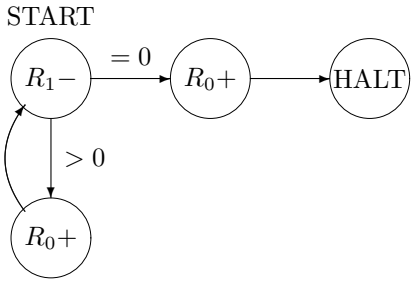
This machine repeatedly adds R_2 to R_0, R_1 many times. The register R_3 is used to keep track of the original value of R_2 to restore it at the end of each repetition.

We will see that this notion of computable function properly extends the class of primitive recursive functions. We first see that every primitive recursive function is register machine-computable.

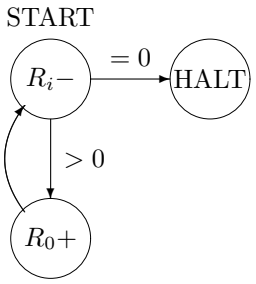
Lemma 1.34. *Each initial function is RM-computable.*

Proof: The zero function $Z(x) = 0$ is computed by the trivial machine:

The successor function $S(x) = x + 1$ is computed by the machine:



Each projection function $P_i^k(x_1, \ldots, x_k) = x_1$ is computed by the machine:



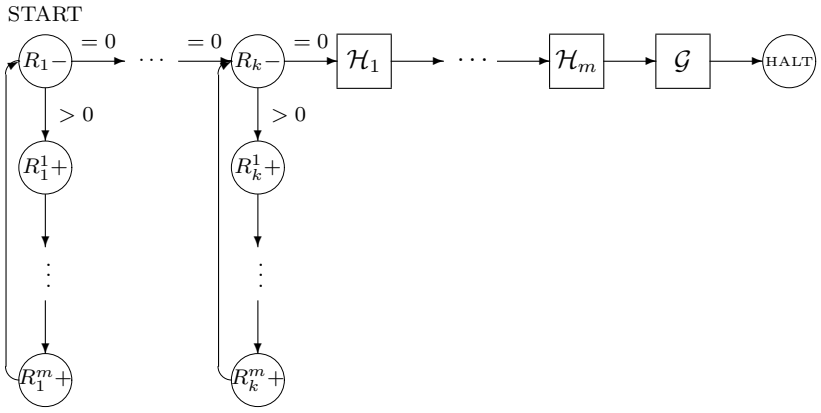
□

Lemma 1.35. *The class of RM-computable functions is closed under composition.*

Proof: Suppose the function $g(x_1, \ldots, x_m)$ and the functions $h_1(x_1, \ldots, x_k)$, $\ldots$, $h_m(x_1, \ldots, x_k)$ are computed by the register machines $\mathcal{G}$ and $\mathcal{H}_1, \ldots, \mathcal{H}_m$, respectively. We may assume that the registers used by each machine are disjoint. We will assume that the input registers for machine $\mathcal{H}_i$ are $R_1^i, \ldots, R_k^i$ and its output register is R_0^i; the input registers for $\mathcal{G}$ will be $R_0^1, \ldots, R_0^m$ and its out register is R_0. We will describe a machine with input registers $R_1, \ldots, R_k$ and output register R_0 which computes the composition function

$$f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k)).$$

The idea is to copy the input registers for f into the input registers for each of the h_i's, compute these functions using the machines $\mathcal{H}_i$, and then use these results in the machine $\mathcal{G}$. We illustrate this in the following diagram, where we replace the boxes for the various machines by the corresponding instructions for those machines, starting at each one's START instruction, and proceeding to the next instruction in the diagram instead of halting.

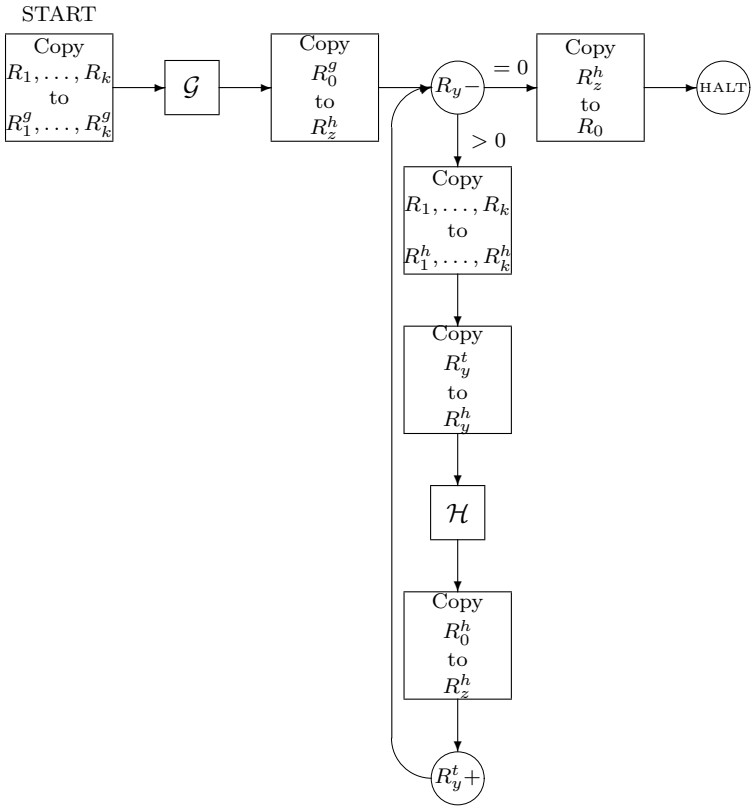


□

Lemma 1.36. *The class of RM-computable functions is closed under primitive recursion.*

Proof: Suppose we have functions $g(x_1, \ldots, x_k)$ computed by the machine $\mathcal{G}$ with input registers $R_1^g, \ldots, R_k^g$ and output register R_0^g, and $h(y, z, x_1, \ldots, x_k)$ computed by the machine $\mathcal{H}$ with input registers $R_y^h, R_z^h, R_1^h, \ldots, R_k^h$ and output register R_0^h. We will describe a machine to compute the function $f(y, x_1, \ldots, x_k)$ obtained from g and h by primitive recursion. We will let the input registers of this machine be $R_y, R_1, \ldots, R_k$.

We will use a "bottom-up" approach, i.e. we will begin by computing $f(0, x_1, \ldots, x_k)$, then use this to compute $f(1, x_1, \ldots, x_k)$ and so forth until we have computed $f(y, x_1, \ldots, x_k)$. We will use a register R_y^t to count up the current value of y we are computing, and count down R_y until we reach 0. This is illustrated in the following diagram.



□

The above three lemmas immediately give us the following.

Theorem 1.37. *Every primitive recursive function is RM-computable.*

We will next see that there are register machine-computable functions which are not primitive recursive. Specifically, we will show that Ackermann's Function is RM-computable.

The idea will be to use the recursive definition of Ackermann's function, and keep track of a finite sequence which stores the values we need to plug back into (in the way that most programming languages use a stack to handle recursive function calls). That is, when we use the recursion

$$A_{n+1}(x+1) = A_n(A_{n+1}(x))$$

to compute $A(n+1, x+1)$, we will first try to calculate $A(n+1, x)$, but add n to our sequence to remember that we need to plug this value back into A_n.

We begin with a few preliminary functions for manipulating sequences, which will essentially be pushing to and popping from a stack. We will show that several functions are primitive recursive, and hence are RM-computable by our previous result. Recall the following two primitive recursive functions introduced in the homework:

$$\text{Length}(s) = \begin{cases} k+1 & \text{if } s = \langle n_0, \ldots, n_k \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$R(s) = \begin{cases} n_k & \text{if } s = \langle n_0, \ldots, n_k \rangle \\ 0 & \text{otherwise} \end{cases}$$

The function $R(s)$ will output the last (rightmost) number in the sequence coded by s, and thus act as a Pop function. We need two more functions.

Lemma 1.38. *The following functions are primitive recursive:*

$$\mathit{Push}(s, n) = \langle n_0, \ldots, n_k, n \rangle \qquad \mathit{if}\ s = \langle n_0, \ldots, n_k \rangle$$
$$\mathit{Erase}(s) = \langle n_0, \ldots, n_{k-1} \rangle \qquad \mathit{if}\ s = \langle n_0, \ldots, n_k \rangle$$

Proof: Note that we only really care about applying these functions to s which code legitimate sequences. We can thus define:

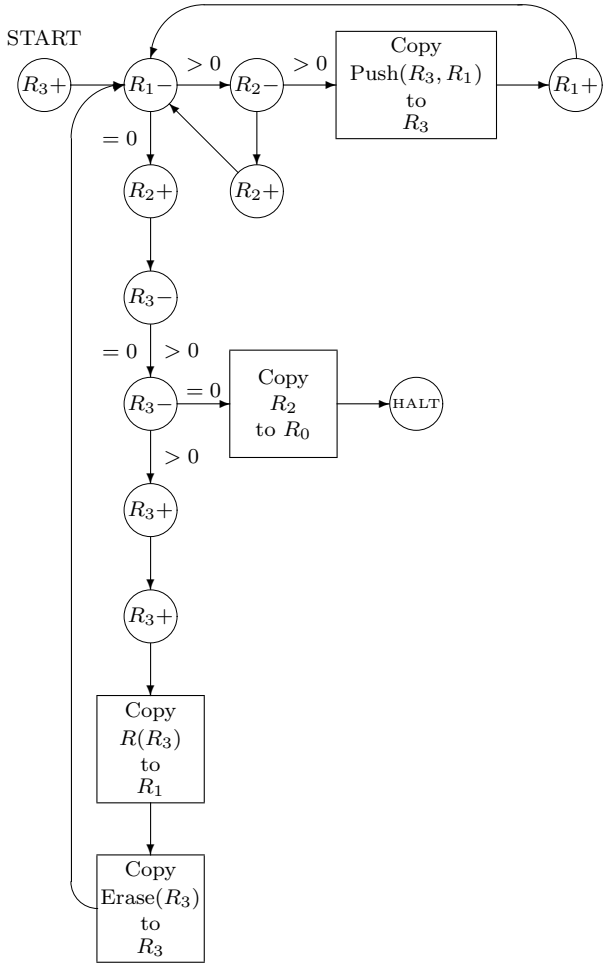
$$\text{Push}(s, n) = s \cdot p(\text{Length}(s))^{n+1}$$
$$\text{Erase}(s) = \text{Quotient}\left(s, p(\text{Length}(s) \dot{-} 1)^{R(s)+1}\right)$$

where as before $p(i)$ is the i-th prime. □

We can thus assume that we have register machines which compute these functions. We will use these to compute Ackermann's function.

Theorem 1.39. *Ackermann's function $A(n, x)$ is RM-computable.*

Proof: Our inputs for n and x are R_1 and R_2, respectively. We will use R_3 to hold our finite sequence (recursion stack). Note that by our conventions the empty sequence is coded by the number 1 (this was handled slightly differently in lecture). The diagram for our machine will be as follows.



The machine proceeds as follows. We first set up our sequence in R_3 to be the empty sequence (coded by 1). We then check if $n = 0$, which we can compute directly as $x + 1$. We do this, and then check if our sequence is empty (i.e. R_3 contains 1). If so, we can simply output $x + 1$ and halt. If not, we must plug this value into some other A_n; in this case we remove the last value from the sequence, use this as the next n value, and proceed.

If n was not 0, we check if $x = 0$; if so, we want to calculate $A_{n-1}(1)$; we set up this computation and proceed. Otherwise we are in the recursion case; we add $n - 1$ to our sequence, set up to compute $A(n, x - 1)$ and proceed. □

1.4 Partial Recursive Functions

We will now extend the class of primitive recursive functions in order to include more functions, such as Ackermann's function. As with the case of RM-

computable functions, this will involve considering partial functions.

Definition 1.40. Let $f : \mathbb{N}^k \to \mathbb{N}$ be a partial function. We say $f(x_1, \ldots, x_k)$ *converges* if $(x_1, \ldots, x_k) \in \text{dom}(f)$ and write $f(x_1, \ldots, x_k) \downarrow$. Similarly, we say $f(x_1, \ldots, x_k)$ *diverges* if $(x_1, \ldots, x_k) \notin \text{dom}(f)$ and write $f(x_1, \ldots, x_k) \uparrow$.

Note 1.41. When dealing with partial functions, we need to keep track of the domain. As usual, we have that $g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k)) \downarrow$ if and only if $h_i(x_1, \ldots, x_k) \downarrow$ for each i, and g is defined at these values. For primitive recursion, we have that $f(0, x_1, \ldots, x_k) \downarrow$ iff $g(x_1, \ldots, x_k) \downarrow$, and $f(y+1, x_1, \ldots, x_k) \downarrow$ iff $f(y, x_1, \ldots, x_k) \downarrow$ and $h(y, f(y, x_1, \ldots, x_k), x_1, \ldots, x_k) \downarrow$. In particular, for $f(y+1, x_1, \ldots, x_k)$ to be defined we must have $f(w, x_1, \ldots, x_k)$ defined for all $0 \leq w \leq y$.

Definition 1.42 (Minimization). Let $R(y, x_1, \ldots, x_k)$ be a relation. We say that the function $f(x_1, \ldots, x_k)$ is defined from R by *minimization* if

$$f(x_1, \ldots, x_k) = \begin{cases} \text{the least } y \text{ such that } R(y, x_1, \ldots, x_k) \text{ holds} \\ \qquad \text{if such a } y \text{ exists} \\ \uparrow \quad \text{if no such } y \text{ exists} \end{cases}$$

We write

$$f(x_1, \ldots, x_k) = \mu y[R(y, x_1, \ldots, x_k)]$$

when f is defined from R by minimization.

Minimization is also known as *μ-recursion*, *unbounded search*, and the *least number operator*. Note that this is similar to bounded minimization which we encountered earlier, except that the search can potentially go on forever, in which case the function is undefined. We will see that even when the search always succeeds, this is a more powerful operation than bounded minimization.

Example 1.43. Let $p(y)$ be a polynomial, and let $R(y, x)$ hold iff $p(y) = x$. Then the function

$$f(x) = \mu y[p(y) = x]$$

returns the smallest natural number which satisfies $p(y) = x$ if such exists, and is undefined if there is no y with $p(y) = x$.

Definition 1.44. The class of *partial recursive functions* is the smallest collection of partial functions containing the initial functions and closed under composition, primitive recursion, and minimization. More precisely, we define

$$\begin{aligned} \mathcal{R}_0 &= \text{the initial functions} \\ \mathcal{R}_{n+1} &= \text{all functions obtained from functions in } \mathcal{R}_n \text{ using composition} \\ &\qquad \text{or primitive recursion, or obtained by minimization from a} \\ &\qquad \text{relation } R \text{ with } \chi_R \text{ in } \mathcal{R}_n \\ \mathcal{R} &= \bigcup_{n \geq 0} \mathcal{R}_n \end{aligned}$$

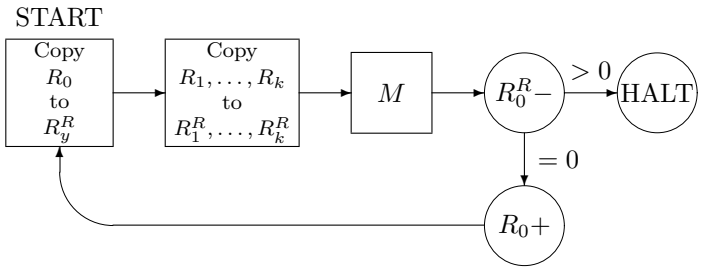
Then $\mathcal{R}$ is the class of partial recursive functions.

Definition 1.45. We say that a function is *(total) recursive* (or just *recursive*) if it is a partial recursive function which is total. We say that a relation R is *recursive* if χ_R is a (total) recursive function. Note that χ_R is always a total function. We can restate the previous definition to say that a function obtained by minimization from a recursive relation is partial recursive.

We will see that the class of partial recursive functions coincides with the class of RM-computable functions. This provides evidence that we have captured the intuitive notion of computability. We start by showing that each partial recursive function is RM-computable.

Lemma 1.46. *The class of RM-computable partial functions is closed under minimization, i.e. if R is a relation such that χ_R is RM-computable, and f is obtained from R by minimization, then f is RM-computable.*

Proof: Let the register machine M compute $\chi_R(y, x_1, \ldots, x_k)$ with input registers $R_y^R, R_1^R, \ldots, R_k^R$ and output register R_0^R. By assumption, χ_R is a total function. We describe a machine to compute $f(x_1, \ldots, x_k) = \mu y[R(y, x_1, \ldots, x_k)]$:



This machine successively computes values of χ_R starting with $y = 0$ until R is satisfied, at which point (if ever) if halts outputting the current value of y. □

Corollary 1.47. *Every partial recursive function is RM-computable.*

Showing that each RM-computable function is partial recursive will involve more work. This will be one consequence of a very useful result presented in the next section.

1.5 The Enumeration Theorem

In this section we will develop a system of coding register machines, and use it to show in particular that every RM-computable function is partial recursive. Each register machine M will be assigned a number $\ulcorner M \urcorner$ called the *Gödel number* of M. We will also refer to these numbers as indices for the functions computed by the machines.

Definition 1.48. Given a register machine instruction I, its *code* (or Gödel number) $\ulcorner I \urcorner$ is defined to be:

$$\ulcorner I \urcorner = \begin{cases} 2^0 \cdot 3^i \cdot 5^j & \text{if } I \text{ is an increment instruction } (R_i+, I_j) \\ 2^1 \cdot 3^i \cdot 5^j \cdot 7^k & \text{if } I \text{ is a decrement instruction } (R_i-, I_j, I_k) \\ 2^2 & \text{if } I \text{ is a HALT instruction} \end{cases}$$

Given a register machine M with instructions $I_0, \ldots, I_t$, its *Gödel number* (or code) $\ulcorner M \urcorner$ is the code of the sequence $\langle \ulcorner I_0 \urcorner, \ldots, \ulcorner I_t \urcorner \rangle$, i.e.

$$\ulcorner M \urcorner = \prod_{m=0}^{t} p(m)^{\ulcorner I_m \urcorner + 1}$$

where $p(m)$ is the m-th prime.

We can thus deal with register machines as natural numbers via their Gödel numbers. We first see that we can effectively recognize these codes.

Lemma 1.49. *The relation RM(e), where*

$$\text{RM}(e) \equiv e \textit{ is the Gödel number of some register machine } M$$

is primitive recursive.

Proof: We will need to check that e codes a finite sequence, and that each term in the sequence is a valid instruction code, i.e. any instructions to branch to are within the sequence. We start by noting that

$$\text{RM}(e) \equiv \bigvee_{l=0}^{e} \text{RM}(e, l)$$

where $\text{RM}(e, l)$ holds iff e is the Gödel number of a machine with instructions $I_0, \ldots, I_l$ (we can use e as an upper bound since as we have seen a code for a sequence of length l will be at least l). We will show that the relation $\text{RM}(e, l)$ is primitive recursive; $\text{RM}(e)$ is then primitive recursive since it is obtained using bounded disjunction.

For this, we observe that

$$\text{RM}(e, l) \equiv \text{Seq}(e) \ \wedge \ \text{Length}(e) = l + 1 \ \wedge \\ \bigwedge_{m=0}^{l} \bigvee_{i=0}^{e} \bigvee_{j=0}^{l} \bigvee_{k=0}^{l} \left[(e)_m = 2^0 \cdot 3^i \cdot 5^j \vee (e)_m = 2^1 \cdot 3^i \cdot 5^j \cdot 7^k \vee (e)_m = 2^2 \right]$$

which is a bounded disjunction of primitive recursive relations. □

Definition 1.50. For each $k \geq 0$ we let $\varphi_e^{(k)}$ be the k-ary function computed by the register machine with Gödel number e, or the trivial partial function with empty domain if e is not the Gödel number of any machine. More precisely:

$$\varphi_e^{(k)}(x_1, \ldots, x_k) = \begin{cases} \text{the final value of } R_0 \text{ if } e = \ulcorner M \urcorner \text{ for some register} \\ \qquad \text{machine } M \text{ which eventually halts when started} \\ \qquad \text{with } R_1 = x_1, \ldots, R_k = x_k \\ \uparrow \text{ if } e \text{ is not the Gödel number of a machine,} \\ \qquad \text{or } e = \ulcorner M \urcorner \text{ but } M \text{ does not halt when started} \\ \qquad \text{with } R_1 = x_1, \ldots, R_k = x_k \end{cases}$$

If f is a k-ary partial function and $e \in \mathbb{N}$ is such that $f = \varphi_e^{(k)}$, we say that e is an *index* for f. Note that a given function will have many different indices (infinitely many in fact), since multiple register machines will compute the same function.

Before we present the main theorem of this section, we prove a simple but useful technical lemma.

Lemma 1.51 (Definition by cases). *Let $R_1, \ldots, R_n$ be k-ary primitive recursive relations which are mutually exclusive and exhaustive, i.e. each $(x_1, \ldots, x_k)$ satisfies exactly one of these relations. Let $f_1, \ldots, f_n$ be k-ary primitive recursive functions, and define the function f by*

$$f(x_1, \ldots, x_k) = \begin{cases} f_1(x_1, \ldots, x_k) & \text{if } R_1(x_1, \ldots, x_k) \\ \vdots & \\ f_n(x_1, \ldots, x_k) & \text{if } R_n(x_1, \ldots, x_k) \end{cases}$$

Then f is primitive recursive.

Proof: We have $f(x_1, \ldots, x_k) = \sum_{i=1}^{n} f_i(x_1, \ldots, x_k) \cdot \chi_{R_i}(x_1, \ldots, x_k)$. □

Here is the main theorem about Gödel numbering:

Theorem 1.52 (The Enumeration Theorem). *For each $k \geq 0$ define the $(k+1)$-ary function U_k as*

$$U_k(e, x_1, \ldots, x_k) = \varphi_e^{(k)}(x_1, \ldots, x_k).$$

Then U_k is partial recursive.

Note 1.53. We may view U_k as a "universal" register machine. It is computed by some register machine, which is able to use the input e to simulate any other register machine. This is the idea behind a programmable computer.

Proof: The idea will be to encode the *state* of a register machine as it is running, i.e. the current values of the registers as well as the next instruction to execute. Together with the Gödel number of the machine, this will be sufficient information to continue the running of the machine. We will encode the state using the following function:

$$\text{State}(e, x_1, \ldots, x_k, n) = p(0)^m \cdot \prod_{i=0}^{\infty} p(i+1)^{z_i}$$

where, after having run n steps, the register machine with Gödel number e is about to execute instruction I_m and the value of each register R_i is z_i. Note that we have an infinite product; however, as any machine will use only finitely many register, all but finitely many of these terms will be 1 and hence it is well-defined.

We will first show that the State function is primitive recursion. We will use the recursion:

$$\begin{aligned} \text{State}(e, x_1, \ldots, x_k, 0) &= p(0)^0 \cdot p(2)^{x_1} \cdot \cdots \cdot p(k+1)^{x_k} \\ \text{State}(e, x_1, \ldots, x_k, n+1) &= \text{Step}(e, \text{State}(e, x_1, \ldots, x_k, n)) \end{aligned}$$

where $\text{Step}(e, z)$ will output the state of the machine with Gödel number e after running one step (instruction) starting in state z. Note that the initial value $\text{State}(e, x_1, \ldots, x_k, 0)$ is the starting state of the machine, with instruction 0 to be executed, the registers $R_1, \ldots, R_k$ set equal to $x_1, \ldots, x_k$, and all other registers equal to 0. We will show that the Step function is primitive recursive, which will show that State is primitive recursive.

We define the Step function by cases, depending on the type of the next instruction to be executed. Note that we will presume that e is actually the Gödel number of a register machine and z a code for a state; we will only actually use it in this case. We set:

$$\text{Step}(e, z) = \begin{cases} z \cdot p(0)^{j-m} \cdot p(i+1) & \text{if } ((e)_m)_0 = 0 \\ z \cdot p(0)^{j-m} & \text{if } ((e)_m)_0 = 1 \text{ and } (z)_{i+1} = 0 \\ z \cdot p(0)^{k-m} \cdot p(i+1)^{-1} & \text{if } ((e)_m)_0 = 1 \text{ and } (z)_{i+1} > 0 \\ z & \text{otherwise} \end{cases}$$

where we have used the abbreviations

$$m = (z)_0, \quad i = ((e)_m)_1, \quad j = ((e)_m)_2, \quad k = ((e)_m)_3$$

and we use $(n)_i$ here to denote the exponent of $p(i)$ is n (this is slightly different from the coordinate function for sequences, but essentially the same and primitive recursive). Note that $(e)_m$ is the code of the instruction to be executed, $((e)_m)_0$ tells which type of instruction it is, and $(z)_{i+1} = z_i$ is the value of the register being incremented or decremented. Note that when the machine reaches a HALT instruction the state does not change. All the conditions are primitive

recursive and are mutually exclusive and exhaustive, so by the previous lemma Step is primitive recursive.

We can now finish the proof by searching for the least n such that the machine has reached a HALT instruction. Let

$$\mathrm{Stop}(e, x_1, \ldots, x_k) = \mu n \left[(e)_{(\mathrm{State}(e,x_1,\ldots,x_k,n))_0} = 2^2 \wedge \mathrm{RM}(e) \right]$$

i.e. the least n such that the machine has reached a HALT instruction, or undefined if the machine never halts or e is not the Gödel number of a register machine. Stop is thus a partial recursive function. We lastly can set

$$U_k(e, x_1, \ldots, x_k) = (\mathrm{State}(e, x_1, \ldots, x_k, \mathrm{Stop}(e, x_1, \ldots, x_k)))_1$$

i.e. we find the value of R_0 in the code of the machine state once it has halted; the function is again undefined it the machine never halts or e does not code a machine. This shows that U_k is partial recursive, finishing the proof. □

Since we can substitute the Gödel number of any given register machine in for e in U_k, we obtain the following.

Corollary 1.54. *Every RM-computable function is partial recursive.*

Hence the class of RM-computable functions is the same as the class of partial recursive functions. The same sort of analysis used in the Enumeration Theorem can be applied to other notions of computable functions, such as Turing Machines, Markov Algorithms, etc. The idea is that any reasonable notion of an algorithm proceeds in steps, and we can develop an analogous coding for the state of the algorithm and the stepping function as we did above. This presents strong evidence that the class of partial recursive functions has captured the intuitive notion of computability. This (necessarily imprecise) idea is the content of what is known as *Church's Thesis*:

Church's Thesis. *Any intuitively algorithmically computable number-theoretic function is partial recursive.*

We will henceforth take this as our definition of a computable function, and study the properties of the class of partial recursive functions.

Another immediate consequence of the Enumeration Theorem is the following.

Corollary 1.55. *The class of partial recursive functions is the smallest class of partial functions containing the primitive recursive functions and closed under composition and minimization.*

Note, in fact, that we only need to apply minimization once in order to construct a given partial recursive function.

A natural question is whether we can avoid using partial functions if we want to study the total recursive functions. One approach would be to only allow minimization to be applied to a relation $R(y, x_1, \ldots, x_k)$ which has the

property that for each $(x_1, \ldots, x_k)$ there is a y satisfying $R(y, x_1, \ldots, x_k)$. As we will see in the next section, though, there will not be an effective (algorithmically computable) way of determining when a relation has this property. The next example gives more evidence that the use of partial functions in studying computability is somehow necessary.

Example 1.56. We give an example of a partial recursive function which can not be extended to a total recursive function. Let $g : \mathbb{N} \to \mathbb{N}$ be a partial function. We say that a total function $f : \mathbb{N} \to \mathbb{N}$ *extends* g if we have $f(x) = g(x)$ for any $x \in \text{dom}(g)$, i.e., f simply defines values for any numbers not originally in the domain of g.

Define the partial recursive function g by setting

$$g(x) = U_1(x, x) + 1 = \varphi_1^{(1)}(x) + 1$$

where, following usual conventions, $g(x)$ is undefined when $U_1(x, x)$ is undefined. Suppose f were a total recursive function which extended g. Let e_0 be an index for f, i.e. $f = \varphi_{e_0}^{(1)}$. But then

$$f(e_0) = \varphi_{e_0}^{(1)}(e_0) = U_1(e_0, e_0)$$

so we have that $g(e_0)$ is defined, namely

$$g(e_0) = U_1(e_0, e_0) + 1$$

contradicting that $f(e_0) = g(e_0)$ since $e_0 \in \text{dom}(g)$. Hence g can not be extended to a total recursive function. We will use this function g later.

Note that the above is an example of diagonalization. The function g itself is not problematic in this regard, since we can (and in fact must) have $g(e) \uparrow$ when e is an index for g. But this technique shows that we can not have a version of the enumeration theorem for the total recursive functions, i.e. there is no function $V_1(e, x_1)$ which is a total recursive function and enumerates all the total recursive unary functions in the way that U_1 does for the partial recursive functions.

Note 1.57. There is also a characterization of the primitive recursive functions in terms of machines. We can define a *Loop Program* as follows. We have a number of registers as before, and increment and decrement instructions. These instructions do not branch, though; execution always proceeds to the next instruction (decrementing a register with value 0 simply does nothing). In addition, we allow loops of the form "`for` R_i `do` S", where R_i is a register and S is a sequence of instructions (possible containing other loops); this is executed by repeating the instructions in S a number of times equal to the initial value of R_i (we can change R_i within the loop, but this does not affect how many times the loop is executed). The class of *Loop-computable* functions turns out to be equal to the class of primitive recursive functions.

1.6 Unsolvable Problems

We now turn our attention to which sets are computable.

Definition 1.58. A set $A \subseteq \mathbb{N}^k$ is *recursive* if its characteristic function χ_A is a (total) recursive function.

This is the same definition as for relations, and we will treat sets and relations identically.

Example 1.59. The set

$$A = \{n \in \mathbb{N} : n \text{ is prime}\}$$

is recursive, since we saw earlier that its characteristic function is in fact primitive recursive. Similarly the sets

$$\{s : s \text{ codes a sequence}\}$$

and

$$\{e : e \text{ is the Gödel number of a register machine}\}$$

are both recursive (primitive recursive, again).

There are, of course, many non-recursive sets, since there are continuum many subsets of the natural numbers but only countably many recursive functions. We are interested, though, in determining whether certain natural, explicitly defined sets are recursive. We give a first example of a non-recursive set here.

Example 1.60. Let g be the partial function constructed earlier, and define the set $K \subseteq \mathbb{N}$ as

$$K = \text{dom}(g) = \{x : \varphi_x^{(1)}(x) \downarrow\}.$$

We claim that K is non-recursive. If it were recursive, then the function

$$f(x) = \begin{cases} g(x) & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

would also be recursive, since we can produce a register machine which first computes $\chi_K(x)$ and halts with output 0 if $x \notin K$; otherwise, if $x \in K$ the machine computes $g(x)$ which is defined and outputs that value. But then f would be a total recursive function extending g, a contradiction.

We will be interested in determining when certain problems can be solved algorithmically. We introduce the following notation to make this precise.

Definition 1.61. A *problem* is a set $A \subseteq \mathbb{N}^k$ for some k. We say that the problem is *solvable* (or *decidable*) if the set is recursive; otherwise we say that it is *unsolvable* (or *undecidable*).

The idea is that to each such set we associate the problem of determining whether a given number is an element of that set; e.g., the set of primes is associated to the problem: Given a particular number, determine whether or not it is a prime. Thus, a problem is solvable if there is an algorithm which, when given a particular number, determines whether or not the number is an element of the set.

Example 1.62. As we have noted above, the set of primes and the set of codes for sequences are solvable, whereas the set K is unsolvable.

Certain natural problems occurring in mathematics have been shown to be unsolvable; we list several here.

Example 1.63 (Hilbert's Tenth Problem). Informally, Hilbert's Tenth Problem is to determine when a given Diophantine equation has a solution, i.e. given a polynomial $p(x_1, \ldots, x_n)$ with integer coefficients, to determine whether there are integers $a_1, \ldots, a_n$ such that $p(a_1, \ldots, a_n) = 0$. There are several ways of turning this into a precise problem; the following theorem of Matijasevič shows that it is unsolvable in a very strong way.

Theorem 1.64 (Matijasevič). *There is a polynomial $p(x_0, x_1, \ldots, x_9)$ with integer coefficients such that the set*

$$\{n \in \mathbb{N} : p(n, a_1, \ldots, a_9) = 0 \text{ for some } a_1, \ldots, a_9 \in \mathbb{Z}\}$$

is not recursive.

Hence, no reasonable algorithmic solution exists.

Example 1.65 (Word Problems for Groups). Let G be a group presented with finitely many generators and relations. The *word problem for G* is to determine, given a particular word w in the generators of G, whether or not $w = 1_G$ (the group identity element of G). The word problem of some groups is solvable, such as abelian groups and free groups; however, Novikov showed that there is a finitely presented group G with an unsolvable word problem.

Example 1.66 (The Halting Problem). A problem which is particularly relevant to our study is *The Halting Problem*, which we define to be the set

$$H = \{e \in \mathbb{N} : \varphi_e^{(1)}(0) \downarrow\}.$$

Thus, the problem is to determine whether a given register machine with code e eventually halts when started with all registers empty. We will see below that the set H is not recursive, i.e. the Halting Problem is unsolvable.

1.7 Reducibility

We have so far seen one concrete example of a non-recursive set, the set K. We wish to develop techniques for showing that other sets are not recursive which avoid direct arguments. The idea will be introduce a notion of reducing one problem to another which will give a gauge of relative complexity of problems.

Definition 1.67. let A and B be subsets of $\mathbb{N}$. We say that A is *many-to-one reducible* to B (or just *reducible*) if there is a (total) recursive function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $n \in \mathbb{N}$, $n \in A$ if and only if $f(n) \in B$. We write $A \leq_m B$ to indicate that A is reducible to B.

Thus, if $A \leq_m B$ then a reduction function f allows us to reduce the membership problem for A to that for B, as the next lemma makes precise.

Lemma 1.68. *Suppose $A \leq_m B$. If B is recursive, then so is A. If A is not recursive, then B is not recursive.*

Proof: Let f be a recursive function so that $n \in A$ iff $f(n) \in B$, and suppose B is recursive, so that χ_B is recursive. Then $\chi_A(n) = \chi_B(f(n))$, i.e. $\chi_A \; \chi_B \circ f$, so that χ_A and hence A is recursive. The second statement is just the contrapositive of the first. □

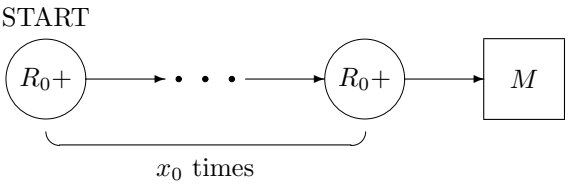
Note 1.69. The relation $\leq_m$ is a quasi-order, i.e. it is reflexive and transitive. It is not a linear order, though, there are sets A and B so that $A \not\leq_m B$ and $B \not\leq_m A$.

The following theorem will be a key technical tool in producing reductions.

Theorem 1.70 (The Parametrization Theorem). *Let $\theta(x_0, x_1, \ldots, x_k)$ be a $(k+1)$-ary partial recursive function. Then there is a primitive recursive function $f(x_0)$ such that for all $x_0, x_1, \ldots, x_k \in \mathbb{N}$ we have*

$$\theta(x_0, x_1, \ldots, x_k) = \varphi^{(k)}_{f(x_0)}(x_1, \ldots, x_k).$$

Proof: Since θ is partial recursive, there is some register machine which computes it. By an easy modification, we can set up the machine so that (unlike our normal convention) this machine uses registers $R_0, R_1, \ldots, R_k$ for input, and outputs in R_0. Call this machine M. Now, given x_0 we want $f(x_0)$ to produce an index for a machine M_{x_0} which takes inputs $x_1, \ldots, x_k$ in registers $R_1, \ldots, R_k$, sets register R_0 equal to x_0, and then runs machine M. Thus, M_{x_0} should look as follows:



Let the instructions of M be $\langle J_0, \ldots, J_t \rangle$. Then the instructions of M_{x_0} will be

$$\langle \underbrace{(R_0+, I_1), \ldots, (R_0+, I_t)}_{x_0 \text{ many}}, J'_0, \ldots, J'_t \rangle$$

where the instruction J'_i is the same as J_i except that the branching instructions have their indices increased by x_0 to account for their later position in the list,

e.g. if $J_0 = (R_i-, J_j, J_k)$ then $J'_0 = (R_i-, J_{j+x_0}, J_{k+x_0})$. Recalling that the code of a machine is the code for the sequence of codes for its instructions, we have

$$f(x_0) = \ulcorner M_{x_0} \urcorner = \prod_{m=0}^{x_0-1} p(m)^{2^0 \cdot 3^0 \cdot 5^{m+1}} \cdot \prod_{m=0}^{t} p(x_0+m)^{\ulcorner J'_m \urcorner}$$

where

$$\ulcorner J'_m \urcorner = \begin{cases} \ulcorner J_m \urcorner \cdot 5^{x_0} & \text{if } J_m \text{ is an increment instruction} \\ \ulcorner J_m \urcorner \cdot 5^{x_0} \cdot 7^{x_0} & \text{if } J_m \text{ is a decrement instruction} \\ \ulcorner J_m \urcorner & \text{if } J_m \text{ is a HALT instruction} \end{cases}$$

Note that the second product has a fixed number of terms which can be defined explicitly, so that the function f is primitive recursive. □

With a little more care we can prove the following version of the Parametrization Theorem, generally known as the *S-M-N Theorem*:

Corollary 1.71 (The S-M-N Theorem). *For each m and n there is a primitive recursive function $s_n^m(e, x_1, \ldots, x_m)$ such that*

$$\varphi_e^{(m+n)}(x_1, \ldots, x_m, y_1, \ldots, y_n) = \varphi^{(n)}_{s_n^m(e,x_1,\ldots,x_m)}(y_1, \ldots, y_n)$$

for all $e, x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{N}$.

The idea is to apply the above idea to the function U_k, and handle several input registers at once.

Note 1.72. We can view the Enumeration Theorem and Parametrization Theorem as saying that our indexing of partial recursive functions satisfies certain good properties. If we chose some arbitrary way of assigning indices to the partial recursive functions it would not necessarily satisfy these properties, although most any reasonable indexing will. These two theorems will be the chief properties we need, and later theorems we prove about the class of partial recursive functions will not need to refer to the actual coding of register machines, just the conclusions of these two theorems.

We can now prove the unsolvability of the Halting Problem.

Theorem 1.73. *The Halting Problem is unsolvable, i.e. the set*

$$H = \{e \in \mathbb{N} : \varphi_e^{(1)}(0) \downarrow\}$$

is not recursive.

Proof: We will show that the set K is reducible to H; since K is not recursive, this will show that H is not recursive. Consider the function

$$\theta(x, y) = \varphi_x^{(1)}(x) = U_2(x, x)$$

which is partial recursive. Recall that $K = \{x : \varphi_x^{(1)}(x) \downarrow\}$. Applying the Parametrization Theorem, there is a primitive recursive function f such that

$$\theta(x, y) = \varphi_x^{(1)}(x) = \varphi_{f(x)}^{(1)}(y)$$

for all x and y. Then if $x \in K$ we have $\varphi_x^{(1)}(x) \downarrow$, so $\varphi_{f(x)}^{(1)}(y) \downarrow$ for all y, so $\varphi_{f(x)}^{(1)}(0) \downarrow$, i.e. $f(x) \in H$. Similarly, if $x \notin K$ then $\varphi_{f(x)}^{(1)}(0) \uparrow$, so $f(x) \notin H$. Thus f is a reduction of K to H. □

1.8 The Recursion Theorem

We present an important and powerful theorem due to Kleene:

Theorem 1.74 (The Recursion Theorem or Fixed Point Theorem). *Let $\theta(e, x_1, \ldots, x_k)$ be a partial recursive function. Then there is an index e_0 such that, for all $x_1, \ldots, x_k$,*

$$\theta(e_0, x_1, \ldots, x_k) = \varphi_{e_0}^{(k)}(x_1, \ldots, x_k).$$

The following corollary makes the name clearer:

Corollary 1.75. *Let $f : \mathbb{N} \to \mathbb{N}$ be a total recursive function and $k \geq 1$. Then there is an index e_0 such that*

$$\varphi_{f(e_0)}^{(k)} = \varphi_{e_0}^{(k)}.$$

Thus, if we think of f as a procedure for transforming one register machine into another, the theorem says that there will be a machine so that the modified machine (although it may be a different machine) will still compute the same partial function, i.e. this function is a "fixed point" of f.

Proof of Corollary 1.75: Let $\theta(e, x_1, \ldots, x_k) = U_k(f(e), x_1, \ldots, x_k)$. Applying the Recursion Theorem we find an index e_0 so that

$$\varphi_{e_0}^{(k)}(x_1, \ldots, x_k) = \theta(e_0, x_1, \ldots, x_k) = U_k(f(e_0), x_1, \ldots, x_k) = \varphi_{f(e_0)}^{(k)}(x_1, \ldots, x_k)$$

for all $x_1, \ldots, x_k$, as desired. □

Proof of Theorem 1.74: The idea will be to consider register machines which somehow have access to their own indices; we will then try to find such a machine which, when given $x_1, \ldots, x_k$, evaluates the function θ on its own index and the inputs $x_1, \ldots, x_k$.

Applying the Parametrization Theorem to θ, we can find a primitive recursive function f such that

$$\theta(w, x_1, \ldots, x_k) = \varphi_{f(w)}^{(k)}(x_1, \ldots, x_k)$$

and applying it to the function $U_k(U_1(u,u), x_1, \ldots, x_k)$ we can find a primitive recursive d such that

$$\varphi^{(k)}_{d(u)}(x_1, \ldots, x_k) = \varphi^{(k)}_{\varphi^{(1)}_u(u)}(x_1, \ldots, x_k).$$

Let v be an index for the function $f \circ d$, i.e. $\varphi^{(1)}_v(u) = f(d(u))$ for all u. Then

$$\begin{aligned}\varphi^{(k)}_{d(v)} &= \varphi^{(k)}_{\varphi^{(1)}_v(v)}(x_1, \ldots, x_k) \\ &= \varphi^{(k)}_{f(d(v))}(x_1, \ldots, x_k) \\ &= \theta(d(v), x_1, \ldots, x_k)\end{aligned}$$

so taking $e_0 = d(v)$ finishes the proof. □

Example 1.76. We can use the Recursion Theorem to give another proof that Ackermann's function is recursive. We can rewrite the recursive definition of $A(n,x)$ as follows:

$$A(n,x) = \begin{cases} x+1 & \text{if } n = 0 \\ A(n \dot{-} 1, 1) & \text{if } n > 0 \text{ and } x = 0 \\ A(n \dot{-} 1, A(n, x \dot{-} 1)) & \text{if } n > 0 \text{ and } x > 0 \end{cases}$$

We consider a procedure which take some partial function (which is able to correctly compute some values of $A(n,x)$) and improves it by computing additional values for which the required data is available. Specifically, given a partial function $\psi(n,x) = \varphi^{(2)}_e(n,x)$, define $\psi(n,x)$ as

$$\psi(n,x) = \begin{cases} x+1 & \text{if } n = 0 \\ \varphi^{(2)}_e(n \dot{-} 1, 1) & \text{if } n > 0 \text{ and } x = 0 \\ \varphi^{(2)}_e(n \dot{-} 1, \varphi^{(2)}_e(n, x \dot{-} 1)) & \text{if } n > 0 \text{ and } x > 0 \end{cases}$$

Then, using the enumeration and Parametrization Theorems, there is a primitive recursive function $f(e)$ which computes an index for ψ from an index e for φ, i.e. $\psi = \varphi^{(2)}_{f(e)}$. Now let e_0 be a fixed point of f, i.e. $\varphi^{(2)}_{f(e_0)} = \varphi^{(2)}_{e_0}$. Then we have

$$\varphi^{(2)}_{e_0}(n,x) = \varphi^{(2)}_{f(e_0)}(n,x) = \begin{cases} x+1 & \text{if } n = 0 \\ \varphi^{(2)}_{e_0}(n \dot{-} 1, 1) & \text{if } n > 0 \text{ and } x = 0 \\ \varphi^{(2)}_{e_0}(n \dot{-} 1, \varphi^{(2)}_{e_0}(n, x \dot{-} 1)) & \text{if } n > 0 \text{ and } x > 0 \end{cases}$$

i.e. $\varphi^{(2)}_{e_0}$ satisfies the recursive characterization of Ackermann's function, so $A(n,x) = \varphi^{(2)}_{e_0}(n,x)$ is recursive as desired.

This same technique can be used to show that any well-defined function defined using recursion is partial recursive.

We next consider the distinction between a partial recursive function and its indices, i.e. the collection of register machines which compute the function.

Definition 1.77. We say that a set $I \subseteq \mathbb{N}$ is an *index set* if there is a collection $\mathcal{C}$ of partial recursive functions such that

$$I = I_{\mathcal{C}} = \{e : \varphi_e^{(1)} \in \mathcal{C}\}$$

i.e. I consists of all indices for functions in the class $\mathcal{C}$. Equivalently, I is an index set if and only if whenever $e \in I$ and i is such that $\varphi_e^{(1)} = \varphi_i^{(1)}$, then $i \in I$ as well.

Nontrivial index sets are complicated, as the following theorem shows.

Theorem 1.78 (Rice's Theorem). *If I is an index set and $I \neq \emptyset$ and $I \neq \mathbb{N}$ then I is not recursive.*

Proof: Since I is recursive iff $\mathbb{N} \setminus I$ is recursive, we may assume that the function ψ with empty domain is in $\mathcal{C}$ so that all of its indices are in $I_{\mathcal{C}}$. Let a be some index for ψ. We may also assume that there is some nontrivial partial function which is not in $\mathcal{C}$, so there is some b with $\varphi_b^{(1)} \neq \psi$ with $b \notin I_{\mathcal{C}}$. Now consider the function $f(x, y, z)$ given by

$$f(x, y, z) = \varphi_b^{(1)}(z) + \varphi_x^{(1)}(y) \dot{-} \varphi_x^{(1)}(y).$$

This is partial recursive, so there is an e with $\varphi_e^{(3)}(x, y, z) = f(x, y, z)$. From the S-M-N Theorem we then have

$$f(x, y, z) = \varphi^{(1)}_{s_2^1(e,x,y)}(z).$$

Let $h(x, y) = s_2^1(e, x, y)$. We then have: If $\varphi_x^{(1)}(y) \downarrow$ then $\varphi^{(1)}_{h(x,y)} = \varphi_b^{(1)}(z) \notin \mathcal{C}$, and if $\varphi_x^{(1)}(y) \uparrow$ then $\varphi^{(1)}_{h(x,y)} = \psi \in \mathcal{C}$. This reduces some question about halting to the set $I_{\mathcal{C}}$. We now make this precise.

Define the set $\bar{H}$ as

$$\bar{H} = \{\langle x, y\rangle : \varphi_x^{(1)}(y) \downarrow\}.$$

This is a fuller version of the Halting Problem. If we define the function $h'(s)$ to be:

$$h'(s) = \begin{cases} h((s)_0, (s)_1) & \text{if } s \text{ codes a sequence of length 2} \\ a & \text{otherwise} \end{cases}$$

then we have $s \in \bar{H}$ if and only if $h'(s) \notin I_{\mathcal{C}}$, i.e. h' reduces the set $\bar{H}$ to $\mathbb{N} \setminus I_{\mathcal{C}}$. If we can show that $\bar{H}$ is not recursive, we will then have that $\mathbb{N} \setminus I_{\mathcal{C}}$ is not recursive, hence $I_{\mathcal{C}}$ is not recursive.

To see this, we reduce the Halting Problem H to $\bar{H}$. Let $f(e) = \langle e, 0\rangle$. Then we have $e \in H$ if and only if $f(e) \in \bar{H}$, so we are done. □

This gives us the following immediate corollary:

Corollary 1.79. *The following index sets are not recursive:*

$$\begin{aligned}
\text{Tot} &= \{e : \varphi_e^{(1)} \text{ is a total function}\} \\
\text{Con} &= \{e : \varphi_e^{(1)} \text{ is a constant function}\} \\
\text{Inf} &= \{e : \text{dom}(\varphi_e^{(1)}) \text{ is infinite}\} \\
I_f &= \{e : \varphi_e^{(k)} = f\} \\
&\qquad \text{where } f \text{ is any } k\text{-ary partial recursive function}
\end{aligned}$$

In particular, we can not algorithmically determine whether a particular register machine computes a total function, further motivating or use of partial functions.

Also, the set of indices for a given function is not recursive; however, we can effectively produce infinitely many indices for a given function:

Lemma 1.80 (The Padding Lemma). *For each $k \geq 1$ there is a primitive recursive function $f(e, n)$ such that*

1. *For each e and every n, $\varphi_e^{(k)} = \varphi_{f(e,n)}^{(k)}$.*
2. *For each e, if $n < m$ then $f(e, n) < f(e, m)$.*

We skip the proof, as the idea is simply to append n irrelevant instructions to the machine with Gödel number e.

We can also use the Enumeration, Parametrization, and Recursion Theorems to show that all of the partial recursive operations are effective. We state two instances of this without proof:

Theorem 1.81. *For each k and m there is a recursive function f such that*

$$\varphi_{f(e,i_1,\ldots,i_m)}^{(k)}(x_1, \ldots, x_k) = \varphi_e^{(m)}(\varphi_{i_1}^{(k)}(x_1, \ldots, x_k), \ldots, \varphi_{i_m}^{(k)}(x_1, \ldots, x_k))$$

for all $e, i_1, \ldots, i_m, x_1, \ldots, x_k$.

This says that we can compute an index for a composition of functions from indices for the original functions. The analogous statement for primitive recursion is:

Theorem 1.82. *For each k three is a primitive recursive function f such that*

$$\begin{aligned}
\varphi_{f(e,i)}^{(k+1)}(0, x_1, \ldots, x_k) &= \varphi_e^{(k)}(x_1, \ldots, x_k) \\
\varphi_{f(e,i)}^{(k+1)}(y+1, x_1, \ldots, x_k) &= \varphi_i^{(k+2)}(y, \varphi_{f(e,i)}^{(k+1)}(y, x_1, \ldots, x_k), x_1, \ldots, x_k)
\end{aligned}$$

for all $e, i, y, x_1, \ldots, x_k$.

1.9 The Arithmetical Hierarchy

In this section we introduce a hierarchy of sets (relations) measuring the complexity of their definability.

Definition 1.83 (The Arithmetical Hierarchy). We define both the classes Σ_0^0 and Π_0^0 to be the class of primitive recursive relations (of any arity). Given $n \geq 0$, we define the class Σ_{n+1}^0 to be the class of all k-ary relations P such that P has the form

$$P(x_1, \ldots, x_k) \equiv \exists y R(x_1, \ldots, x_k, y)$$

where R is a $(k+1)$-ary relation in the class Π_n^0. Similarly, we define Π_{n+1}^0 to consist of all relations of the form

$$P(x_1, \ldots, x_k) \equiv \forall y R(x_1, \ldots, x_k, y)$$

where R is a $(k+1)$-ary relation in the class Σ_n^0. We define the class Δ_n^0 to equal $\Sigma_n^0 \cap \Pi_n^0$, i.e. those relations which are in both classes.

Note 1.84. The superscript of "0" refers to the fact that the quantifiers are applied to objects of level 0, i.e. natural numbers. We can similarly define a hierarchy of Σ_n^1 relations, etc., where quantifiers are applied to sets of natural numbers, but we will not study this here.

The level of a relation in the Arithmetical Hierarchy thus corresponds to the number of quantifiers needed to define it. For instance, if R is a primitive recursive relation, then the relation

$$P(x) \equiv \exists y \forall z \exists w R(x, y, z, w)$$

is a Σ_3^0 relation. We list some of the basic closure properties of these classes.

Theorem 1.85. *We have the following:*

1. *The classes Σ_n^0 and Π_n^0 are both contained in the classes Σ_{n+1}^0 and Π_{n+1}^0.*
2. *The classes Σ_n^0 and Π_n^0 are closed under (finite) conjunction and disjunction.*
3. *The classes Σ_n^0 and Π_n^0 are closed under bounded quantification.*
4. *For $n \geq 1$, the class Σ_n^0 is closed under existential quantification, and the class Π_n^0 is closed under universal quantification.*
5. *A relation P is in Σ_n^0 if and only if the relation $\neg P$ is in Π_n^0.*

Proof: Recall that we showed that the class of primitive recursive relations is closed under conjunction, disjunction, negation, and bounded quantification. These are all proved by induction on n.

1. This is immediate for $n = 0$ since $\Sigma^0_0 = \Pi^0_0$. For $n \geq 1$ we easily have that $\Sigma^0_n \subseteq \Pi^0_{n+1}$ since we can just add a vacuous quantifier (one applied to an unused variable), and to see that $\Sigma^0_n \subseteq \Sigma^0_{n+1}$ we can write P as $P(x_1, \ldots, x_k) \equiv \exists y R(x_1, \ldots, x_k, y)$ with R in Π^0_{n-1}; by inductive assumption R is then in Π^0_n and hence P is in Σ^0_{n+1}. Π^0_n is handled similarly.

2. Immediate for $n = 0$. Suppose P_1 and P_2 are Σ^0_n for $n \geq 1$, and we have $P_i(x_1, \ldots, x_k) \equiv \exists y R_i(x_1, \ldots, x_k, y)$ with R_i in Π^0_{n-1}. Then

$$\begin{aligned} P_1(x_1, \ldots, x_k) \wedge P_2(x_1, \ldots, x_k) &\equiv \exists y R_1(x_1, \ldots, x_k, y) \\ &\quad \wedge \exists y R_2(x_1, \ldots, x_k, y) \\ &\equiv \exists y R_1(x_1, \ldots, x_k, y) \wedge \exists z R_2(x_1, \ldots, x_k, z) \\ &\equiv \exists y \exists z (R_1(x_1, \ldots, x_k, y) \wedge R_2(x_1, \ldots, x_k, z)). \end{aligned}$$

   The quantified expression is Π^0_{n-1} by assumption, and part (4) will show that this expression is thus in Σ^0_n. The other cases are handled similarly.

3. This is handled similarly to part (2).

4. Let $P(x_1, \ldots, x_k, z)$ be Σ^0_n for $n \geq 1$ where

$$P(x_1, \ldots, x_k, z) \equiv \exists y P(x_1, \ldots, x_k, z, y)$$

   with R in Π^0_{n-1}. Then

$$\begin{aligned} \exists z P(x_1, \ldots, x_k, z) &\equiv \exists z \exists y P(x_1, \ldots, x_k, z, y) \\ &\equiv \exists s (\mathrm{Seq}(s) \wedge \mathrm{Length}(s) = 2 \wedge P(x_1, \ldots, x_k, (s)_0, (s)_1)) \end{aligned}$$

   i.e. we encode z and y as a pair; since coding and uncoding of finite sequences is primitive recursive, the part inside the quantifier is Π^0_{n-1}.

5. Immediate for $n = 0$, and for $n \geq 1$ with P in Σ^0_n and R in Π^0_{n-1} we have

$$\neg P(x_1, \ldots, x_k) \equiv \neg \exists y R(x_1, \ldots, x_k, y) \equiv \forall y \neg R(x_1, \ldots, x_k, y)$$

   so that $\neg R$ is in Σ^0_{n-1} by assumption and hence $\neg P$ is in Π^0_n.

□

Example 1.86. We can see that the set $H = \{e : \varphi^{(1)}_e(0) \downarrow\}$ belongs to the class Σ^0_1. We can write

$$\begin{aligned} H(e) &\equiv \mathrm{RM}(e) \wedge \exists n (\text{machine } e \text{ halts after at most } n \text{ steps on input } 0) \\ &\equiv \exists n (\mathrm{RM}(e) \wedge (e)_{(\mathrm{State}(e,0,n))_0} = 2^2) \end{aligned}$$

where $\mathrm{State}(e, 0, n)$ and $\mathrm{RM}(e)$ are primitive recursive, so that this is Σ^0_1.

Note 1.87. We often use the notation $\varphi^{(k)}_{e,n}(x_1, \ldots, x_k) \downarrow$ to indicate that e codes a register machine which halts on inputs $x_1, \ldots, x_k$ after at most n steps; as discussed above this is a primitive recursive relation of $e, n, x_1, \ldots, x_k$.

The class Σ_1^0 is of particular importance, and we will give several different characterizations of it.

Definition 1.88. We define the set $W_e^{(k)} \subseteq \mathbb{N}^k$ to be

$$W_e^{(k)} = \mathrm{dom}(\varphi_e^{(k)}).$$

Theorem 1.89. *A k-ary relation P is in Σ_1^0 if and only if $P = \mathrm{dom}(\psi)$ for some k-ary partial recursive function ψ, i.e. iff $P = W_e^{(k)}$ for some e.*

Proof: If P is Σ_1^0, so that $P(x_1, \ldots, x_k) \equiv \exists y R(x_1, \ldots, x_k, y)$ with R primitive recursive, then $P = \mathrm{dom}(\psi)$ where

$$\psi(x_1, \ldots, x_k) = \mu y[R(x_1, \ldots, x_k, y)]$$

is partial recursive. Conversely, if $P = \mathrm{dom}(\psi)$ with $\psi = \varphi_e^{(k)}$, then

$$P(x_1, \ldots, x_k) \equiv \exists n(\varphi_{e,n}^{(k)}(x_1, \ldots, x_k) \downarrow)$$

so P is Σ_1^0 since the inner relation is primitive recursive as discussed above. □

Definition 1.90. Let A be an infinite subset of $\mathbb{N}$. The *principal function of A* if the one-to-one function π_A which enumerates A in increasing order, i.e. π_A is an increasing function such that $A = \mathrm{ran}(\pi_A)$.

Lemma 1.91. *Let A be an infinite subset of $\mathbb{N}$. Then A is recursive if and only the principal function π_A is recursive.*

Proof: If A is recursive, then π_A can be obtained using the recursion:

$$\begin{aligned} \pi_A(0) &= \text{ the least element of } A \\ \pi_A(n+1) &= \mu y[y > \pi_A(n) \wedge y \in A] \end{aligned}$$

Conversely, if π_A is recursive, then

$$y \in A \text{ iff } \bigvee_{x=0}^{y} \pi_A(x) = y$$

which is recursive since the class of recursive predicates is closed under bounded disjunction. □

Theorem 1.92. *For $A \subseteq \mathbb{N}$, the following are equivalent:*

1. *A is Σ_1^0.*
2. *$A = \mathrm{dom}(\psi)$ for some partial recursive function ψ.*
3. *$A = \mathrm{ran}(\psi)$ for some partial recursive function ψ.*
4. *Either $A = \emptyset$ or $A = \mathrm{ran}(\psi)$ for some total recursive function ψ.*

5. *Either A is finite or $A = \mathrm{ran}(\psi)$ for some one-to-one total recursive function ψ.*

Proof: We have already seen that (1) and (2) are equivalent, and it is clear that (5) implies (4) and that (4) implies (3). To see that (3) implies (1), note that if $A = \mathrm{ran}(\psi)$ then

$$y \in A \equiv \exists x \psi(x) = y \equiv \exists x \exists n \varphi^{(1)}_{e,n}(x) = y$$

where e is an index for ψ, so this is Σ^0_1.

To see that (1) implies (5), let R be a primitive recursive relation so that

$$x \in A \equiv \exists y R(x, y)$$

and define the set B as

$$B = \left\{ 2^x \cdot 3^y : R(x,y) \wedge \neg \bigvee_{z=0}^{x-1} R(x,z) \right\}$$

Then B is an infinite primitive recursive set, so its principal function π_B is a one-to-one recursive function. Note that for each $x \in A$ there is a unique y with $2^x \cdot 3^y \in B$. We can then define ψ to be

$$\psi(n) = (\pi_B(n))_0$$

(i.e. the power of 2 in $\pi_B(n)$, which is the x-coordinate), and we have that ψ is a one-to-one total recursive function with $A = \mathrm{ran}(\psi)$. □

Note 1.93. These sets are generally known as *recursively enumerable (r.e.)* sets.

We will now develop a better picture of the arithmetical hierarchy.

Theorem 1.94. *A k-ary relation P is in the class Δ^0_1 if and only if it is recursive.*

Proof: If P is recursive, so that χ_P is a recursive function, then $P = \mathrm{dom}(\psi)$ where

$$\psi(x) = \mu y[\chi_P(x) = 1]$$

so that P is Σ^0_1; since $\neg P$ is also recursive we have that $\neg P$ is also Σ^0_1, so that P is Π^0_1 as well, and hence Δ^0_1.

Suppose P is Δ^0_1, and let R_1 and R_2 be two primitive recursive relations so that

$$P(x_1, \ldots, x_k) \equiv \exists y R_1(x_1, \ldots, x_k, y) \equiv \forall y R_2(x_1, \ldots, x_k, y).$$

We can then define t total recursive function f by

$$f(x_1, \ldots, x_k) = \mu y[R_1(x_1, \ldots, x_k, y) \vee \neg R_2(x_1, \ldots, x_k, y)]$$

and we will have

$$P(x_1, \ldots, x_k) \equiv R_1(x_1, \ldots, x_k, f(x_1, \ldots, x_k))$$

so that P is recursive. □

Theorem 1.95 (Universal Σ^0_n relations). *For each $n \geq 1$ and $k \geq 1$ there is a relation V^n_k such that:*

1. *V^n_k is a $(k+1)$-ary relation in the class Σ^0_n.*
2. *For each k-ary relation P in Σ^0_n, there is an $e \in \mathbb{N}$ such that*

$$P(x_1, \ldots, x_k) \equiv V^n_k(e, x_1, \ldots, x_k).$$

The analogous statement holds for Π^0_n.

Proof: This is done by induction on n. Note that by the Enumeration Theorem we can set $V^1_k = \mathrm{dom}(U_k)$ where U_k is the $(k+1)$-ary universal partial recursive function. If V^n_k is universal for Σ^0_n, then $\neg V^n_k$ is universal for Π^0_n, and we can set

$$V^{n+1}_k(e, x_1, \ldots, x_k) \equiv \exists y V^n_{k+1}(e, x_1, \ldots, x_k, y).$$

□

Note 1.96. There is no analogous universal Δ^0_n relation. To see this, e.g. for $k = 1$, suppose there were a Δ^0_n binary relation D which was universal for unary Δ^0_n relations. Define the relation P as

$$P(x) \equiv \neg D(x, x)$$

Then P is Δ^0_n, so there should be some e so that $P(x) \equiv D(e, x)$ for all x, but then $D(e, e) \equiv P(e) \equiv \neg D(e, e)$, a contradiction.

Lemma 1.97. *Let $A, B \subseteq \mathbb{N}$ with $A \leq_m B$. If B is Σ^0_n, then so is A; if B is π^0_n then so is A.*

Proof: This is done by induction on n; the case of $n = 1$ is representative so we only give that. Suppose R is a primitive recursive relation so that

$$B(x) \equiv \exists y R(x, y)$$

and let f be a total recursive function so that $x \in A$ if and only if $f(x) \in B$. Then we have

$$A(x) \equiv \exists y R(f(x), y) \equiv \exists y \exists z \exists n(\varphi^{(1)}_{e,n}(x) = z \wedge R(z, y))$$

where e is an index for f, so that A is Σ^0_1 as well. □

Definition 1.98. For $n \geq 1$, we say that a set $B \subseteq \mathbb{N}$ is Σ^0_n-complete if:

1. B is in the class Σ^0_n.
2. For any Σ^0_n set A we have $A \leq_m B$.

The notion of a Π^0_n-complete set is defined similarly.

Theorem 1.99. *For each $n \geq 1$ there is a Σ_n^0-complete set and a Π_n^0-complete set. A Σ_n^0-complete set is not in Π_n^0 and a Π_n^0-complete set is not in Σ_n^0.*

Proof: Let $V_1^n(e,x)$ be the universal Σ_n^0 relation from above. Then the set

$$\{2^e \cdot 3^x : V_1^n(e,x)\}$$

is Σ_n^0, and a Σ_n^0 set A with $A(x) \equiv V_1^n(e,x)$ is reducible to this set by the function $f(x) = 2^e \cdot 3^x$. The complement of a Σ_n^0 set is easily Π_n^0-complete. To see that a Σ_n^0-complete set is not in Π_n^0 it suffices to show that there is a Σ_n^0 set which is not in Π_n^0. If we define the set $A(x) \equiv V_n^1(x,x)$, then the same sort of diagonal argument used above shows that $\neg A$ can not be Σ_n^0, hence A can not be Π_n^0. The case of Π_n^0 follows immediately. □

We can thus see that the arithmetical hierarchy is a proper hierarchy:

Corollary 1.100. *For $n \geq 1$ we have $\Delta_n^0 \subsetneq \Sigma_n^0$, $\Delta_n^0 \subsetneq \Pi_n^0$, and $\Sigma_n^0 \cup \Pi_n^0 \subsetneq \Delta_{n+1}^0$, i.e. all inclusions are proper.*

Proof: The first two inclusions are proper by the above theorem, since a Σ_n^0-complete set is not Π_n^0 and hence not Δ_n^0, and similarly for Π_n^0-complete sets. For the third inclusion, we must find a set in Δ_{n+1}^0 which is neither Σ_n^0 nor Π_n^0. Let A be a Σ_n^0-complete set (so that $\neg A$ is Π_n^0-complete), and define the set B by

$$B = \{2x : x \in A\} \cup \{2x+1 : x \notin A\}$$

Then B is a Δ_{n+1}^0 set since it is the union of a Σ_n^0 set and a Π_n^0 set. We have that A is reducible to B by the function $f(x) = 2x$, so B can not be Π_n^0, and $|negA$ is reducible to B by the function $g(x) = 2x+1$ so B can not be Σ_n^0. □

Chapter 2

Undecidability of Arithmetic

We will apply the results of the previous chapter to show that the theory of arithmetic is undecidable, that is, there is no algorithm to decide whether a sentence in the language of arithmetic is true or false.

2.1 The Language of Arithmetic

By the *theory of arithmetic* we mean the set of sentences which are true in the structure $\langle \mathbb{N}, 0, 1, +, \cdot, = \rangle$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers, 0 and 1 are names for these numbers, $+$ and $\cdot$ denote the binary functions of addition and multiplication on $\mathbb{N}$, and $=$ denotes the binary relation of equality on $\mathbb{N}$. We introduce the language we will use for this structure.

Definition 2.1 (The Language of Arithmetic). The language will contain infinitely many *variable* denoted $x_0, x_1, x_2, \ldots$ (although we will often use x, y, etc. for simplicity), and symbols for the constants 0 and 1. We next define the collection of *terms* which are built from these inductively. That is, each variable is a term, and 0 and 1 are terms, and if t_1 and t_2 are two terms, then so are $t_1 + t_2$ and $t_1 \cdot t_2$. For each natural number n we will use $\underset{\sim}{\boldsymbol{n}}$ to denote the term $\underbrace{1 + \cdots + 1}_{n \text{ times}}$, and call this the *standard term for* n; although often we will just write n to mean the standard term $\underset{\sim}{\boldsymbol{n}}$. We will similarly abbreviate $x \cdot x$ as x^2 and so forth.

We next define *formulas*. An *atomic formula* is an expression of the form $t_1 = t_2$, where t_1 and t_2 are terms. The collection of *formulas* are built inductively using the *Boolean connectives* $\neg$, $\vee$, and $\wedge$, and the *quantifiers* $\exists$ and $\forall$. That is, each atomic formula is a formula, and if F and G are formulas and x is a variable, then the following expressions are also formulas: $\neg F$, $F \vee G$, $F \wedge G$, $\exists x F$, and $\forall x F$.

Example 2.2. The expression

$$\exists z(x + z + 1 = y)$$

is a formula. Quantified variable range over the natural numbers, so the intended meaning of the above is "there is a natural number z such that $x + z + 1 = y$". Note that we can find such a z i and only if $x < y$, so we will use the expression $x < y$ to abbreviate the above formula.

Example 2.3. The formula

$$x > 1 \wedge \neg\exists y\exists z(y > 1 \wedge z > 1 \wedge x = y \cdot z)$$

expresses the property that x is a prime number (where expressions involving $<$ are again used as abbreviations).

Definition 2.4. An occurrence of a variable in a formula is said to be *bound* if it is contained within the scope of a quantifier; otherwise, the occurrence is called *free*. In the example above, the occurrences of x are all free occurrences, whereas the occurrences of y and z are all bound. Note that a variable can occur both freely and bound in the same formula.

We say that x is a *free variable* of a formula F is it has at least one free occurrence, and we often write $F(x)$ to indicate that the only free variable of F is x. A *sentence* is a formula which has no free variables. Sentences are formulas to which we can assign a truth value in the structure $\mathbb{N}$ by interpreting the functions and equality in the usual way and letting quantified variables range over $\mathbb{N}$. Note that a formula with free variables does not have a truth value unless we substitute specific values in for them. If $F(x_1, \ldots, x_k)$ is a formula with free variables $x_1, \ldots, x_k$ and $n_1, \ldots, n_k \in \mathbb{N}$, then $F(n_1, \ldots, n_k)$ is the sentence obtained by substituting the standard terms $\boldsymbol{n_1}, \ldots, \boldsymbol{n_k}$ in for the respective free occurrences of $x_1, \ldots, x_k$. This will then be either true or false.

2.2 Arithmetical Definability

We now consider which relations and functions can be defined in the language of arithmetic.

Definition 2.5 (Arithmetical Definability). A k-ary relation $R \subseteq \mathbb{N}^k$ is *arithmetically definable* if there is a formula $F(x_1, \ldots, x_j, y)$ in the language of arithmetic such that for all $n_1, \ldots, n_k \in \mathbb{N}$ the relation $R(x_1, \ldots, x_k)$ is true if and only if the sentence $F(n_1, \ldots, n_k)$ is true in $\mathbb{N}$. A k-ary partial function $f : \mathbb{N}^k \to \mathbb{N}$ is *arithmetically definable* if its graph is definable, i.e there is a formula $F(x_1, \ldots, x_k, y)$ such that for all $n_1, \ldots, n_k, m \in \mathbb{N}$, we have $f(n_1, \ldots, x_k) = m$ if and only if the sentence $F(n_1, \ldots, n_k, m)$ is true in $\mathbb{N}$.

Example 2.6. The function Quotient(y, x) introduced earlier is arithmetically definable by the formula $F(y, x, z)$:

$$\exists v(\neg(x = 0) \wedge v < x \wedge y = x \cdot z + v).$$

Note 2.7. The notion of arithmetically definable relations is similar to the arithmetical hierarchy discussed earlier. There we were allowed to build up relations using quantifiers and connectives starting from the primitive recursive relations; here the basic relations we start from are ones of the form $t_1 = t_2$ where t_1 and t_2 are terms. We will later see that these two notions coincide.

We will show that all recursive functions and relations are arithmetically definable. We will do this by induction on functions as we did when we showed that all partial recursive functions are RM-computable.

Lemma 2.8. *Each initial function is arithmetically definable.*

Proof: The zero function $Z(x) = 0$ is defined by the formula

$$F(x, y) \equiv x = x \wedge y = 0$$

(the "$x = x$" is really irrelevant, but we include it just to ensure that x is a free variable of the formula; we will omit such trivialities from now on). The successor function $S(x) = x + 1$ is defined by the formula

$$G(x, y) \equiv y = x + 1$$

and each projection function $P_i^k(x_1, \ldots, x_k) = x_i$ is defined by the formula:

$$H_i^k(x_1, \ldots, x_k, y) \equiv y = x_i.$$

□

Lemma 2.9. *The class of arithmetically definable functions is closed under composition, i.e. if $h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k)$ and $g(x_1, \ldots, x_m)$ are all arithmetically definable, then so is the function $f(x_1, \ldots, x_k)$ with*

$$f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k)).$$

Proof: Let $h_1, \ldots, h_m$ and g be defined by the formulas $H_1, \ldots, H_m$ and G, respectively. Then f is defined by the formula:

$$\exists y_1 \cdots \exists y_m (H_1(x_1, \ldots, x_k, y_1) \wedge \cdots \wedge H_m(x_1, \ldots, x_k, y_m) \wedge G(y_1, \ldots, y_m, y))$$

i.e. the variable $y_1, \ldots, y_m$ are used to stand for the intermediate values in the calculation. □

Lemma 2.10. *If the $(k+1)$-ary relation $R(x_1, \ldots, x_k)$ is arithmetically definable, then so is the k-ary partial function*

$$f(x_1, \ldots, x_k) = \mu y[R(x_1, \ldots, x_k, y)]$$

defined from R using minimization.

Proof: If R is defined by the formula $F(x_1, \ldots, x_k, y)$, then f is defined by the formula $G(x_1, \ldots, x_k, y)$:

$$F(x_1, \ldots, x_k, y) \wedge \neg existsz(z < y \wedge F(x_1, \ldots, x_k, z)).$$

□

Note 2.11. A relation is arithmetically definable if and only if its characteristic function is arithmetically definable. To see this, suppose $R(\vec{x})$ is defined by the formula $F(\vec{x})$; then χ_R is defined by the formula

$$G(\vec{x}, y) \equiv (y = 1 \wedge F(\vec{x})) \vee (y = 0 \wedge \neg F(\vec{x}))$$

and if χ_R is defined by the formula $G(\vec{x}, y)$, then R is defined by the formula

$$F(\vec{x}) \equiv G(\vec{x}, 1).$$

The case of primitive recursion will require a bit more work. Intuitively, we want to imitate the technique for RM-computability, i.e. compute $f(y, \vec{x})$ by successively computing $f(0, \vec{x})$, $f(1, \vec{x})$, $\ldots$, $f(y, \vec{x})$. In order to do this, though, we need an arithmetically definable way of coding a sequence. Although the coding we introduced earlier will turn out to be definable, this is difficult to see *a priori*, so we will develop a new manner of coding sequences which is more transparently definable. This will involve a bit of number theory. Recall that two positive integers are *relatively prime* if the have no common factor other than 1; a set of positive integers is said to be *pairwise relatively prime* if any two distinct integers in the set are relatively prime.

Lemma 2.12. *Given a positive integer k, we can find arbitrarily large integers m such that the set*

$$\{m + 1, 2m + 1, \ldots, km + 1\}$$

is pairwise relatively prime.

Proof: Let m be any positive integer which is divisible by all of the primes less than k, e.g. any multiple of $k!$. We claim that this m satisfies the conclusion. Suppose not, and let $1 \leq i < j \leq k$ be such that $im + 1$ and $jm + 1$ are not relatively prime. Let p be a prime which is a factor of both $im + 1$ and $jm + 1$. Then p can not be a factor of m. Since $p|((jm + 1) - (im + 1)) = (j - i)m$, we must have that p is a factor of $j - i$. Since $j - i < k$, we must have $p < k$. But then p is a factor of m by our choice of m, a contradiction. □

Example 2.13. If $k = 5$, then the primes less than k are 2 and 3, so we can take m to be any multiple of 6. So, taking $m = 6$ we have that the set

$$\{7, 13, 19, 25, 31\}$$

is pairwise relatively prime.

Lemma 2.14 (Chinese Remainder Theorem). *Let $\{n_1, \ldots, n_k\}$ be a set of positive integers which is pairwise relatively prime. Then given any non-negative integers $r_1, \ldots, r_k$ with $r_i < n_i$ for $1 \leq i \leq k$, we can find a non-negative integer r such that Remainder$(r, n_i) = r_i$ for all i.*

Proof: let $n = n_1 \cdot \cdots \cdot n_k$. For any r with $0 \leq r < n$, define the remainder sequence of r to be $\langle r_1, \ldots, r_k \rangle$, where $r_i = \text{Remainder}(r, n_i)$ for $1 \leq i \leq k$. We want to find an r whose remainder sequence is the sequence we are given. We claim that if $0 \leq r < s < n$, then the remainder sequences for r and s are different. Suppose that r and s have the same remainder sequence. Then $s - r$ is divisible by each of $n_1, \ldots, n_k$; since these are pairwise relatively prime we must then have that $s - r$ is divisible by their product n. But $0 < s - r < n$, a contradiction. Hence distinct r's give rise do distinct remainder sequences. There are exactly n possible remainder sequences, and n choices of r, hence each possible remainder sequence is obtained from some r. This finishes the proof. □

Example 2.15. Considering the previous example, if we choose $r_1 < 7$, ..., $r_5 < 31$, then we can find an r which is a solution to the system of modular equations

$$\begin{aligned} r &\equiv r_1 \mod 7 \\ &\vdots \quad \vdots \\ r &\equiv r_5 \mod 31 \end{aligned}$$

We can view this r (together with $m = 6$) as encoding the sequence $\langle r_1, \ldots, r_5 \rangle$.

We now introduce our coding method.

Definition 2.16 (Gödel's β-function). Given non-negative integers r, m, and i, we define the function β as

$$\beta(r, m, i) = \text{Remainder}(r, m \cdot (i + 1) + 1).$$

This is known as *Gödel's β-function*. Note that this function is arithmetically definably by the formula $B(r, m, i, y)$:

$$(0 \leq y < m \cdot (i + 1) + 1) \wedge \exists u(r = u \cdot (m \cdot (i + 1) + 1) + y).$$

Lemma 2.17. *Given a finite sequence $\langle r_0, \ldots, r_k \rangle$ of non-negative integers, we can non-negative integers r and m such that $\beta(r, m, i) = r_i$ for each $0 \leq i \leq k$.*

Proof: Using $k + 1$ in place of k above, we can find a positive integer m such that $r_i < im+1$ for each $0 \leq i \leq k$ and such that the set $\{m+1, 2m+1, \ldots, (k+1)m + 1\}$ is pairwise relatively prime. By the Chinese Remainder Theorem we can then find an r so that $r_i = \text{Remainder}(r, (i + 1)m + 1) = \beta(r, m, i)$ for all $0 \leq i \leq k$. □

Hence r and m encode the sequence $\langle r_0, \ldots, r_k \rangle$ via the β-function.

We can now prove:

Lemma 2.18. *The class of arithmetically definable functions is closed under primitive recursion, i.e. if the k-ary function g and the $(k+2)$-ary function H are arithmetically definable, then so is the $(k+1)$-ary function f defined by:*

$$f(0, x_1, \ldots, x_k) = g(x_1, \ldots, x_k)$$
$$f(y+1, x_1, \ldots, x_k) = h(y, f(y, x_1, \ldots, x_k), x_1, \ldots, x_k)$$

Proof: Let g and h be defined respectively by the formulas $G(x_1, \ldots, x_k, w)$ and $H(y, z, x_1, \ldots, x_k, w)$. We want a formula $F(y, x_1, \ldots, x_k, w)$ to say that there is a finite sequence $r_0, \ldots, r_y$ with $r_0 = g(x_1, \ldots, x_k)$ and $r_{i+1} = g(i, r_i, x_1, \ldots, x_k)$ for $0 \leq i < y$ and $r_i = w$; this would clearly define f. We will do this by saying that there are an r and m which encode this sequence via the β-function, which we know from the previous lemma. Letting $B(r, m, i, y)$ be the formula defining β, we can take our formula to be:

$$\exists r \exists m(\ \exists u(B(r, m, 0, u) \wedge G(x_1, \ldots, x_k, u)) \wedge \forall i(i \geq y \vee \exists u \exists v(B(r, m, i, u) \wedge B(r, m, i+1, v) \wedge H(i, u, x_1, \ldots, x_k, v))) \wedge B(r, m, y, w)\ )$$

This finishes the proof. □

Example 2.19. We can see that the exponential function $f(x, y) = y^x$ is arithmetically definable by the formula $F(x, y, w)$:

$$\exists r \exists m(\ B(r, m, 0, 1) \wedge \forall i(i \geq x \vee \exists u \exists v(B(r, m, i, u) \wedge B(r, m, i+1, v) \wedge v = u \cdot y)) \wedge B(r, m, x, w)\ )$$

Recalling that the class of partial recursive functions is the smallest class of partial functions containing the initial functions and closed under composition, primitive recursion, and minimization, we thus have:

Corollary 2.20. *Every partial recursive function is arithmetically definable.*

Hence we also have that every recursive relation is arithmetically definable. But more is true:

Corollary 2.21. *The set $K = \{e\varphi_e^{(1)}(e) \downarrow\}$ is arithmetically definable, so there is an arithmetically definable set which is not recursive.*

Proof: If a function $f(x_1, \ldots, x_k)$ is defined by the formula $F(x_1, \ldots, x_k, y)$, then the set $\mathrm{dom}(f)$ is defined by the formula

$$\exists y F(x_1, \ldots, x_k, y)$$

and the set $\mathrm{ran}(f)$ is defined by the formula

$$\exists x_1 \cdots \exists x_k F(x_1, \ldots, x_k, y).$$

The set K is the domain of the partial recursive function $f(x) = U_2(x, x)$; since this function is arithmetically definable, so is K. □

We in fact can characterize which sets are arithmetically definable:

Theorem 2.22. *Let P be a k-ary relation. Then the following are equivalent:*

1. *P is arithmetically definable.*
2. *P is in the arithmetical hierarchy, i.e. P is either Σ_n^0 or Π_n^0 for some n.*

Proof: The theorem above in particular shows that every primitive recursive relation is arithmetically definable, i.e. every relation in $\Sigma_0^0 = \Pi_0^0$. Since applying quantifiers to arithmetically definable relations yields arithmetically definable relations, induction on n shows that every Σ_n^0 or Π_n^0 relation is arithmetically definable. For the converse, note that

$$\Sigma_\infty^0 = \bigcup_{n \geq 0} \Sigma_n^0 = \bigcup_{n \geq 0} \Pi_n^0$$

consists precisely of relations in the arithmetical hierarchy. From the closure properties proved earlier, Σ_∞^0 is closed under the Boolean connectives and quantification and contains the primitive recursive relations. Since the class of arithmetically definable sets is obtained from the equality relations by applying these operations, every arithmetically definable relation is in Σ_∞^0. □

2.3 Gödel Numbers of Formulas

We will now develop codes for formulas in the language of arithmetic in much the same way as we did for register machines.

Definition 2.23. For each formula F in the language of arithmetic, we will assign a natural number $\ulcorner F \urcorner$, the *Gödel number* of F. We begin by inductively assigning Gödel numbers to terms:

$$\begin{aligned}
\ulcorner 0 \urcorner &= 3^0 \\
\ulcorner 1 \urcorner &= 3^1 \\
\ulcorner x_i \urcorner &= 3^2 \cdot 5^i \quad \text{for a variable } x_i \\
\ulcorner t_1 + t_2 \urcorner &= 3^3 \cdot 5^{\ulcorner t_1 \urcorner} \cdot 7^{\ulcorner t_2 \urcorner} \\
\ulcorner t_1 \cdot t_2 \urcorner &= 3^4 \cdot 5^{\ulcorner t_1 \urcorner} \cdot 7^{\ulcorner t_2 \urcorner}
\end{aligned}$$

where t_1 and t_2 are terms. We now inductively define Gödel numbers for formulas, beginning with atomic formulas:

$$\begin{aligned}
\ulcorner t_1 = t_2 \urcorner &= 2 \cdot 3^0 \cdot 5^{\ulcorner t_1 \urcorner} \cdot 7^{\ulcorner t_2 \urcorner} \\
\ulcorner F \wedge G \urcorner &= 2 \cdot 3^1 \cdot 5^{\ulcorner F \urcorner} \cdot 7^{\ulcorner G \urcorner} \\
\ulcorner F \vee G \urcorner &= 2 \cdot 3^2 \cdot 5^{\ulcorner F \urcorner} \cdot 7^{\ulcorner G \urcorner} \\
\ulcorner \neg F \urcorner &= 2 \cdot 3^3 \cdot 5^{\ulcorner F \urcorner} \\
\ulcorner \forall x_i F \urcorner &= 2 \cdot 3^4 \cdot 5^i \cdot 7^{\ulcorner F \urcorner} \\
\ulcorner \exists x_i F \urcorner &= 2 \cdot 3^5 \cdot 5^i \cdot 7^{\ulcorner F \urcorner}
\end{aligned}$$

where t_1 and t_2 are terms, F and G are formulas, and x_i is a variable.

Definition 2.24. We define the following subsets of $\mathbb{N}$:

$$\begin{aligned}\mathrm{Fml}_{\mathbb{N}} &= \{\ulcorner F \urcorner : F \text{ is a formula in the language of arithmetic}\}\\ \mathrm{Sent}_{\mathbb{N}} &= \{\ulcorner F \urcorner : F \text{ is a sentence in the language of arithmetic}\}\\ \mathrm{Truth}_{\mathbb{N}} &= \{\ulcorner F \urcorner : F \text{ is a true sentence in the language of arithmetic}\}\end{aligned}$$

Lemma 2.25. *The sets $Fml_{\mathbb{N}}$ and $Sent_{\mathbb{N}}$ are both primitive recursive.*

Proof: This is straightforward, and is handled in much the same way as the predicate $\mathrm{RM}(e)$ was. □

We will now see, though, that the set $\mathrm{Truth}_{\mathbb{N}}$ is not recursive, in fact not even arithmetically definable. This will show that the theory of arithmetic is undecidable: There is no algorithm to determine whether a given formula is true or false in $\mathbb{N}$.

Theorem 2.26. *Every arithmetically definable set $A \subseteq \mathbb{N}$ is reducible to $Truth_{\mathbb{N}}$.*

Proof: Let $F(x_1)$ be a formula with one free variable x_1 which defines A, so that

$$A = \{n \in \mathbb{N} : F(n) \text{ is truth}\}.$$

Define the function $f : \mathbb{N} \to \mathbb{N}$ as

$$f(n) = \ulcorner \exists x_1 (x_1 = \underset{\sim}{\boldsymbol{n}} \wedge F(x_1)) \urcorner$$

Then for $n \in \mathbb{N}$ we have that $n \in A$ if and only if $f(n) \in \mathrm{Truth}_{\mathbb{N}}$, so we will be done if we show that f is recursive. This follows from the facts that

$$\begin{aligned}f(n) &= 2 \cdot 3^5 \cdot 5^1 \cdot 7^{\ulcorner x_1 = \underset{\sim}{\boldsymbol{n}} \wedge F(x_1) \urcorner}\\ g(n) &= \ulcorner x_1 = \underset{\sim}{\boldsymbol{n}} \wedge F(x_1) \urcorner = 2 \cdot 3^1 \cdot 5^{\ulcorner x_1 = \underset{\sim}{\boldsymbol{n}} \urcorner} \cdot 7^{\ulcorner F(x_1) \urcorner}\\ h(n) &= \ulcorner x_1 = \underset{\sim}{\boldsymbol{n}} \urcorner = 2 \cdot 3^0 \cdot 5^{\ulcorner x_1 \urcorner} \cdot 7^{\ulcorner \underset{\sim}{\boldsymbol{n}} \urcorner}\end{aligned}$$

and the function $k(n) = \ulcorner \underset{\sim}{\boldsymbol{n}} \urcorner$ is obtained using primitive recursion:

$$\begin{aligned}k(0) &= \ulcorner 0 \urcorner = 3^0\\ k(n+1) &= \ulcorner \boldsymbol{n} \underset{\sim}{+} \mathbf{1} \urcorner = 3^3 \cdot 5^{\ulcorner \underset{\sim}{\boldsymbol{n}} \urcorner} \cdot 7^{\ulcorner 1 \urcorner} = 3^3 \cdot 5^{k(n)} \cdot 7^{3^1}\end{aligned}$$

Hence f is obtained by composing primitive recursive functions, and is thus primitive recursive so we are done. □

Corollary 2.27. *The set $Truth_{\mathbb{N}}$ is not recursive.*

Proof: The set K is arithmetically definable but not recursive; since it is thus reducible to $\mathrm{Truth}_{\mathbb{N}}$, $\mathrm{Truth}_{\mathbb{N}}$ is not recursive either. □

Theorem 2.28 (Tarski). *The set $Truth_{\mathbb{N}}$ is not arithmetically definable.*

Proof: Suppose that $\text{Truth}_{\mathbb{N}}$ were arithmetically definable, so that it was Σ^0_n for some n by our earlier theorem. Let B be a Σ^0_{n+1}-complete set, so that B is not in Σ^0_n. We can not have that B is reducible to $\text{Truth}_{\mathbb{N}}$, or B would be Σ^0_n, so we have a contradiction. □

This theorem is often quoted as "Arithmetical truth is not arithmetically definable."

Chapter 3

Decidability of The Real Numbers

We will now consider the theory of the real numbers and see that this theory, unlike the theory of arithmetic, is decidable: There is an algorithm to determine whether a given sentence in the language of the real numbers is true or false in $\mathbb{R}$. We will do this by showing that the class of definable relations over $\mathbb{R}$ is much simpler than for $\mathbb{N}$.

3.1 Quantifier Elimination

We extend our earlier definitions slightly:

Definition 3.1. Let $\mathcal{L}$ be the *language of ordered rings*, i.e.

$$\mathcal{L} = \mathcal{L}_{\text{OR}} = \{0, 1, +, \cdot, -, =, <\}$$

where $-$ is a unary function (negation) and $<$ is a binary predicate. Everything is now interpreted in $\mathbb{R}$, so that e.g. quantified variables range over $\mathbb{R}$. We define formulas and Gödel numbers as before, adding cases for the new symbols $-$ and $<$.

Definition 3.2. We say that two formulas F and G are *equivalent*, $F \equiv G$, if they have the same free variables $x_1, \ldots, x_k$ and define the same k-ary relation. This is the same as saying that the sentence

$$\forall x_1 \cdots \forall x_k (F(x_1, \ldots, x_k) \Leftrightarrow G(x_1, \ldots, x_k))$$

is true in $\mathbb{R}$, where we use $F \Leftrightarrow G$ as an abbreviation for $(F \wedge G) \vee (\neg F \wedge \neg G)$.

Example 3.3. The formulas F and G with

$$F(x, y) \equiv x < y$$
$$G(x, y) \equiv \exists z(\neg(z = 0) \wedge x + z \cdot z = y)$$

are equivalent. We could thus have omitted $<$ from the language as we did for $\mathbb{N}$; however, we will be interested below in what relations are definable without using quantifiers, and without $<$ we would have to add additional quantifiers to formulas.

Definition 3.4. We say that a formula is *quantifier-free* if it contains no quantifiers, i.e. it is a Boolean combination of atomic formulas of the form $t_1 = t_2$ or $t_1 < t_2$ for terms t_1 and t_2.

Note 3.5. The set of terms is the same as the set of polynomials with integer coefficients over the variables, since terms are formed from 0,1, and variables using $+$, $\cdot$, and $-$.

The fundamental theorem which will lead to the decidability of $\mathbb{R}$ is the following:

Theorem 3.6 (Quantifier Elimination). *For any $\mathcal{L}_{OR}$-formula F, there is a quantifier-free formula F^* such that $F \equiv F^*$.*

Example 3.7. The formula

$$\exists x(ax^2 + bx + c = 0)$$

is equivalent to the quantifier-free formula

$$(a = 0 \wedge b = 0 \wedge c = 0) \vee (a = 0 \wedge b \neq 0) \vee (a \neq 0 \wedge b^2 - 4ac \geq 0)$$

where $\neq$, $\geq$, etc. abbreviate the obvious formulas.

Note 3.8. The only properties of $\mathbb{R}$ we will need are:

1. $\mathbb{R}$ is a commutative ordered field.
2. $\mathbb{R}$ has the intermediate value property for polynomials, i.e. for any polynomial $p(x)$, if $x < y \wedge p(x) < p(y)$ then $\exists z(x < z < y \wedge p(z) = 0)$.

An ordered field with these properties is called a *real closed ordered field*; any real closed ordered field will also admit quantifier elimination.

To prove the theorem we will study a collection of functions which preserve quantifier-free formulas in an appropriate sense.

Definition 3.9. A k-ary relation $A \subseteq \mathbb{R}^k$ is *effective* if it is definable over $\mathbb{R}$ by a quantifier-free formula. Thus, a relation is effective if it is a Boolean combination of sets which are defined by polynomial equalities and inequalities. Note that relations $x = y$ can be expressed as $\neg(x > y) \wedge \neg(x < y)$, so we only need to consider Boolean combinations of inequalities.

Definition 3.10. A k-ary partial function $f : D \to \mathbb{R}$ with $\mathrm{dom}(f) = D \subseteq \mathbb{R}^k$ is *effective* if:

1. D is an effective set.

2. For every effective relation $A(y, z_1, \ldots, z_n)$, the relation B given by

$$B(x_1, \ldots, x_k, z_1, \ldots, z_n) \equiv A(f(x_1, \ldots, x_k), z_1, \ldots, z_n)$$

is effective.

Note 3.11. For the relation B above to be true we require that $(x_1, \ldots, x_k) \in \text{dom}(f)$.

Example 3.12. It can be shown that the function $f(x) = \sqrt{x}$ is effective. First, the domain is effective since it is just the effective set $\{x \in \mathbb{R} : x \geq 0\}$. The second property is established by substituting f into all effective relations; for instance, the relation $\sqrt{x} > 3$ is equivalent to $x > 9$.

Lemma 3.13. *The functions $x + y$, $x \cdot y$, $-x$, and x/y are all effective.*

Proof: The first three are immediate since these functions are all part of our language. For the fourth, note that the domain is $\{(x, y) : y \neq 0\}$ which is easily effective. Let $A(z, \vec{w})$ be an effective predicate; we must show that $B(x, y, \vec{w}) = A(x/y, \vec{w})$ is effective. We know A is a Boolean combination of relations of the form

$$a_n z^n + \cdots + a_1 z + a_0 > 0$$

where each a_i is a polynomial in $\vec{w}$. Then B is a Boolean combination of relations of the form

$$a_n \left(\frac{x}{y}\right)^n + \cdots + a_1 \left(\frac{x}{y}\right) + a_0 > 0.$$

When n is even this is equivalent to the formula

$$y \neq 0 \wedge a_n x^n + a_{n-1} x^{n-1} y + \cdots + a_1 x y^{n-1} + a_0 y^n > 0.$$

The case when n is odd can be handled in cases, or by viewing the polynomial as of degree $n + 1$ with leading coefficient 0. □

Lemma 3.14. *The composition of effective functions is effective.*

Proof: We prove this for the case $f(x) = g(h(x))$, where g and h are effective. Let D_g and D_h be the domains of g an h; by assumption these are effective. Then the domain of f, D_f, satisfies

$$D_f(x) \equiv D_h(x) \wedge D_g(h(x)).$$

Since h is effective, we have that $D_g(h(x))$ is effective, so D_f is effective. Now suppose that $A(y, \vec{z})$ is an effective relation. Then

$$A(f(x), \vec{z}) \equiv A(g(h(x)), \vec{z})$$

and we know $A(g(y), \vec{z})$ is effective since g is effective, and thus $A(g(h(x)), \vec{z})$ is effective since h is effective. □

Corollary 3.15. *Any rational function is effective.*

Lemma 3.16. *The function* $\operatorname{sgn}(x)$ *is effective, where*

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Proof: Let $A(y, \vec{z})$ be an effective relation. Then $A(\operatorname{sgn}(x), \vec{z})$ is equivalent to

$$(x > 0 \wedge A(1, \vec{z})) \vee (x = 0 \wedge A(0, \vec{z})) \vee (x < 0 \wedge A(-1, \vec{z}))$$

which is effective, so we are done. □

Lemma 3.17. *Let* $f(\vec{x})$ *be a function which takes on only finitely many values, all of which are integer. Then* f *is effective if and only if for each* $j \in \mathbb{Z}$ *the relation* $R(\vec{x}) \equiv f(\vec{x}) = j$ *is effective.*

Proof: If f is effective, then we can simply substitute f into the effective predicate $y = j$ to see that $f(\vec{x}) = j$ is effective. For the converse, let $j_1, \ldots, j_n$ be the values. Then the domain D is effective, since $D(\vec{x})$ is equivalent to

$$f(\vec{x}) = j_1 \vee \cdots \vee f(\vec{x}) = j_n.$$

Let $A(y, \vec{z})$ be an effective predicate. Then $A(f(\vec{x}), \vec{z})$ is equivalent to

$$(f(\vec{x}) = j_1 \wedge A(j_1, \vec{z})) \vee \cdots \vee (f(\vec{x}) = j_n \wedge A(j_n, \vec{z}))$$

which is effective. □

Lemma 3.18 (Definition by Cases). *Let* $f_1(\vec{x})$ *and* $f_2(\vec{x})$ *be effective functions, and let* $R(\vec{x})$ *be an effective relation. Then the function* f *is effective, where*

$$f(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{if } R(\vec{x}) \\ f_2(\vec{x}) & \text{if } \neg R(\vec{x}) \end{cases}$$

Proof: The domain D of f is effective, since it is equivalent to

$$(D_1(\vec{x}) \wedge R(\vec{x})) \vee (D_2(\vec{x}) \wedge \neg R(\vec{x}))$$

where D_1 and D_2 are the domains of f_1 and f_2. Let $A(y, \vec{z})$ be an effective relation. Then the relation $B(\vec{x}, \vec{z}) \equiv A(f(\vec{x}), \vec{z})$ is equivalent to

$$(R(\vec{x}) \wedge A(f_1(\vec{x}) m \vec{z})) \vee (\neg R(\vec{x}) \wedge A(f_2(\vec{x}), \vec{z}))$$

which is effective since both f_1 and f_2 are. □

This can easily be extended to more than two cases.

Lemma 3.19. *A function $f(\vec{x})$ is effective if and only if for every integer $d \geq 1$ and every polynomial $q(y) = a_d y^d + \cdots + a_y + a_0$ of degree d the function $\mathrm{sgn}(q(f(\vec{x})))$ is an effective function of $\vec{x}$ and $a_0, \ldots, a_d$.*

Proof: If f is effective, then $\mathrm{dom}(\mathrm{sgn}(q(f))) = \mathrm{dom}(f)$ (restricted to $\vec{x}$; the a_i's do not affect convergence) and hence is effective. We consider for example the effective predicate $A(y) \equiv y > 0$. Then

$$A(\mathrm{sgn}(q(f(\vec{x})))) \equiv \mathrm{sgn}(g(f(\vec{x}))) > 0 \equiv q(f(\vec{x})) > 0 \equiv B(f(\vec{x}))$$

where B is the effective predicate $q(y) > 0$, and hence is effective since f is.

For the converse, suppose that $A(y, \vec{z})$ is effective; we must show that the predicate $A(f(\vec{x}), \vec{z})$ is effective. We can view A as a Boolean combination of predicates of the form $p(y, \vec{z}) > 0$ for polynomials $p(y, \vec{z})$ with integer coefficients. It will thus suffice to show that each of the predicates $p(f(\vec{x}), \vec{z}) > 0$ is effective. We can consider $p(y, \vec{z})$ as a polynomial $q(y)$ whose coefficients are in $\mathbb{Z}[\vec{z}]$, i.e. polynomials in $\vec{z}$ with integer coefficients. Then

$$p(f(\vec{x}), \vec{z}) > 0 \equiv \mathrm{sgn}(q(f(\vec{x}))) = 1$$

Since $\mathrm{sgn}(q(f(\vec{x})))$ is effective as a function of $\vec{x}$ and the coefficients of q by assumption, hence since we are composing this with the polynomials determining the coefficients (which are effective functions), $\mathrm{sgn}(q(f(\vec{x})))$ is an effective function of $\vec{x}$ and $\vec{z}$ and we are done. □

The next lemma says that we can effectively find roots of polynomials.

Lemma 3.20 (Main Lemma). *Let $p(x) = a_n x^n + \cdots + a_1 x + a_0 \in \mathbb{R}[x]$ be a polynomial. Then there are $n+1$ effective functions $k(\vec{a})$ and $\xi_1(\vec{a}), \ldots, \xi_n(\vec{a})$ of the coefficients such that $k = k(\vec{a})$ is the number of roots of $p(x)$, and the roots are enumerated in increasing order by*

$$\xi_1(\vec{a}) < \cdots < \xi_k(\vec{a}).$$

Example 3.21. This is essentially a generalization of the quadratic formula. For $n = 2$ the function $k(\vec{a})$ is defined by cases depending on $\mathrm{sgn}(a_1^2 - 4a_0a_2)$ and the roots are again defined by cases using the quadratic formula. Note that we do not care about the values of ξ_i for $i > k$; we could if we like specify that these be undefined.

Proof: This is proved by induction on n, where the base case of $n = 1$ is clear. Let $p(x)$ be of degree at most n. Then the derivative

$$p'(x) = na_n x^{n-1} + \cdots + 2a_2 x + a_1$$

is of degree at most $n - 1$, so by assumption there are effective functions $m, t_1, \ldots, t_{n-1}$ of the coefficients $\vec{a}$ such that the roots of p' are $t_1(\vec{a}) < \cdots < t_m(\vec{a})$. The function is monotone on each of the intervals

$$(-\infty, t_1), (t_1, t_2), \ldots, (t_{m-1}, t_m), (t_m, \infty)$$

so each of these intervals contains at most one root of $p(x)$ (and there may also be roots at the t_i's). The number of roots of $p(x)$ is thus determined by the values of $\mathrm{sgn}(p(t_i))$ for $1 \leq i \leq m$ and $\mathrm{sgn}(p'(t_1 - 1))$ and $\mathrm{sgn}(p'(t_m + 1))$, so we can use definition by cases to define $k(\vec{a})$ as an effective function of $\vec{a}$.

To show that the ξ_i's are effective, note that we can use definition by cases to determine which interval or endpoint each ξ_i should lie in, so we just consider, for example, a root $\xi = \xi(\vec{a})$ in the interval (t_1, t_2), where we assume $p(t_1) > 0$ and $p(t_2) < 0$. By the previous lemma, it suffices to show that for each $d \geq 1$ and polynomial $q(x)$ of degree d, the function $\mathrm{sgn}(q(\xi))$ is effective as a function of $\vec{a}$ and the coefficients of q.

First, we can assume that the degree of $q(x)$ is less than n. To see this, note that by polynomial division we have

$$q(x) = p(x) \cdot f(x) + r(x)$$

where the degree of $r(x)$ is less than n, and the coefficients of r are easily seen to be effective functions of the coefficients of p and q. Since ξ is a root of p, $q(\xi) = r(\xi)$ so $\mathrm{sgn}(q(\xi)) = \mathrm{sgn}(r(\xi))$; hence we may replace q by r.

By the induction hypothesis, we can thus find the roots of $q(x)$ effectively; let these be $u_1 < \cdots < u_m$. Then $\mathrm{sgn}(q(x))$ on each of the intervals

$$(-\infty, u_1), (u_1, u_2), \ldots, (u_{m-1}, u_m), (u_m, \infty)$$

is determined by the $m + 1$ effective functions

$$\mathrm{sgn}(q(u_1 - 1)), \mathrm{sgn}\left(q\left(\frac{u_1 + u_2}{2}\right)\right), \ldots, \mathrm{sgn}\left(q\left(\frac{u_{m-1} + u_m}{2}\right)\right), \mathrm{sgn}(q(u_m + 1))$$

and the position of ξ relative to the u_i's is determined by the positions of the t_i relative to the u_i's and the effective functions $\mathrm{sgn}(p(u_i))$. We can thus use definition by cases to show that $\mathrm{sgn}(q(\xi))$ is an effective function; hence ξ is an effective function, completing the lemma. □

Lemma 3.22. *If the predicate $A(x_1, x_2, \ldots, x_k)$ is effective, then so is the predicate*

$$B(x_2, \ldots, x_k) \equiv \exists x_1 A(x_1, x_2, \ldots, x_k).$$

Proof: We may view A as a Boolean combination of inequalities of the form $p_i(x_1) > 0$, where the coefficients of the p_i's are polynomials in $x_2, \ldots, x_k$ with integer coefficients. By the Main Lemma, the roots of the p_i's are thus effective functions of $x_2, \ldots, x_k$. Let $\xi_1, \ldots, \xi_m$ be the roots of all of the p_i's. To see if there is an x_1 which satisfies the required inequalities, it thus suffices to consider the values at the ξ_i's and at arbitrary points in the intervals they determine. Hence the predicate

$$\exists x_1 A(x_1, x_2, \ldots, x_k)$$

is equivalent to a finite disjunction of the form

$$A(\eta_1, x_2, \ldots, x_k) \vee \cdots \vee A(\eta_j, x_2, \ldots, x_k)$$

where $\eta_1, \ldots, \eta_j$ enumerate all the values ξ_i, $\frac{\xi_i+\xi_j}{2}$ and $\xi_i \pm 1$. Since the η_i's are effective functions of $x_2, \ldots, x_k$, it follows that this disjunction is an effective predicate, and so B is an effective predicate as desired. □

Theorem 3.23. *Any predicate $A \subseteq \mathbb{R}^k$ which is definable over $\mathbb{R}$ is effective.*

Proof: This amounts to showing that any formula of $\mathcal{L}_{\mathrm{OR}}$ is equivalent to a quantifier-free formula. This is proved by induction on the construction of formulas. Clearly every atomic formula is quantifier-free. If $F \equiv F^*$ and $G \equiv G^*$ where F^* and G^* are quantifier-free, then $\neg F \equiv \neg F^*$, $F \wedge G \equiv F^* \wedge G^*$, and $F \vee G \equiv F^* \vee G^*$.

For a formula of the form $\exists x G(x, \vec{y})$ where G is equivalent to the quantifier free formula G^*, we have that G^* defines an effective predicate $B(x, \vec{y})$. The above lemma tells us that the predicate $A(\vec{y}) \equiv \exists x B(x, \vec{y})$ is effective, and hence is defined by a quantifier-free formula $H^*(\vec{y})$. We thus have that $\exists x G(x, \vec{y}) \equiv H^*(\vec{y})$. Finally, the formula $\forall x G(x, \vec{y})$ is equivalent to $\neg \exists x \neg G(x, \vec{y})$, so we are done. □

Corollary 3.24. *For a predicate $A \subseteq \mathbb{R}^k$, the following are equivalent:*

1. *A is effective.*
2. *A is definable over $\mathbb{R}$.*
3. *A is definable over $\mathbb{R}$ by a quantifier-free formula.*

The same is true for functions $f : d \to \mathbb{R}$ with $D \subseteq \mathbb{R}^k$.

Proof: This is immediate for predicates from the previous theorem. Note that if a function f is effective, then the predicate $y = f(\vec{x})$ is effective, so definable by a quantifier-free formula, and hence f is definable by a quantifier-free formula. If f is definable, then for any effective predicate $A(y, \vec{z})$ the predicate $A(f(\vec{x}), \vec{z})$ is equivalent to the definable predicate $\exists y(y = f(\vec{x}) \wedge A(y, \vec{z}))$, which is thus effective; hence f is an effective predicate. □

Restating this, we have the main result of this section:

Theorem 3.25 (Quantifier Elimination). *If a predicate $A \subseteq \mathbb{R}^k$ is definable over $\mathbb{R}$, then it is definable over $\mathbb{R}$ by a quantifier-free formula, hence any $\mathcal{L}_{OR}$-formula is equivalent to a quantifier-free formula.*

3.2 Decidability of the Real Numbers

We use the above results to show that the theory of the real numbers is decidable.

Theorem 3.26. *There is an algorithm which, when given a formula F of $\mathcal{L}_{OR}$, produces an equivalent quantifier-free formula F^*.*

Note 3.27. What we mean is that there is a recursive function which takes the Gödel number of F and outputs the Gödel number of F^*.

Proof: The algorithm can be considering the proof of the Quantifier Elimination Theorem and noting that all of the steps are effective. □

Theorem 3.28. *There is an algorithm to determine whether or not a given sentence S of $\mathcal{L}_{OR}$ is true in $\mathbb{R}$, i.e. the set*

$$Truth_{\mathbb{R}} = \{\ulcorner S \urcorner : S \text{ is a true sentence in } \mathcal{L}_{OR}\}$$

is recursive.

Proof: In particular, the above theorem tells us that, given a sentence S, we can effectively find a quantifier-free sentence S^* which is equivalent to S, and hence has the same truth value. But a quantifier-free sentence is just a Boolean combination of atomic formulas of the form $t_1 = t_2$ or $t_1 < t_2$, where t_1 and t_2 are variable-free terms, i.e. ones formed from 0 and 1 using $+$ and $\cdot$. The truth values of such sentences are easily determined, so we done. □

Corollary 3.29. *The theory of the ordered ring of real numbers is decidable.*

Corollary 3.30. *Euclidean Geometry is decidable.*

Proof: We using Cartesian analytic geometry, we can reduce questions of geometry to systems of equations in the real numbers, and thus interpret Euclidean Geometry in the theory of the real numbers. □

Part II

Set Theory

Chapter 4

Informal Set Theory

In this chapter we present an informal survey of set theory. We leave several fundamental concepts undefined at this point, notably the definitions of sets, cardinals, and ordinals. We present enough of their properties in order to develop the basic theory. In the next chapter we will present an axiomatic framework for set theory and give precise definitions to these concepts.

4.1 Set Operations

By a *set* we mean some collection of objects; we will not specify at this point what sort of objects these are. We will generally use capital letters X, Y, etc. to denote sets. The fundamental relation for sets is the *membership* relation. If a is an object and X is a set, we write $a \in X$ to indicate that a is a member of X (or a is an element of X).

We view sets as entirely determined by their elements, with no additional structure. This is known as *extensionality*. Formally, for two sets X and Y we have

$$X = Y \Leftrightarrow \forall a(a \in X \Leftrightarrow a \in Y)$$

This should be contrasted with an *intensional* view of sets, where sets are distinguished based on properties that define them. For instance (to use a classical example) the set of human beings and the set of featherless bipeds are intensionally different, since the properties defining them are different, but are extensionally equal since they each have the same members (as far as we know). We will take extensionality as the definition of equality between sets.

We will often use properties to define sets. If R is some relation on objects, we write $\{a : R(a)\}$ to denote the set of objects which satisfy the property R. We will see below that we will have to be somewhat more careful in defining sets this way. We will see that some collections of objects are "too big" to be sets, so we will later have to specify more carefully exactly which collections of objects are allowed to be sets.

We will usually avoid this problem by choosing objects from some given set. We say that a set Y is a *subset* of a set X, $Y \subseteq X$, if every element of Y is an element of X. Formally, $Y \subseteq X$ if and only if

$$\forall a(a \in Y \Rightarrow a \in X).$$

If X is a set and R is a relation, we write $\{a \in X : R(a)\}$ to denote the set of elements of X which satisfy the property R; this is a subset of X.

We can view sets as objects in their own right, and hence allow sets to be elements of other sets. An important case of this is the following.

Definition 4.1. If X is a set, then the *power set of* X, $\mathcal{P}(X)$, is the set consisting of all subsets of X, i.e.

$$\mathcal{P} = \{Y : Y \subseteq X\}$$

Note that the *empty set* $\emptyset$, and the set X itself are both subsets of X, so that $\emptyset \in \mathcal{P}(X)$ and $X \in \mathcal{P}(X)$. Note that if X is a finite set with n elements, then $\mathcal{P}(X)$ has 2^n elements, since for a given subset, each of the n elements can either be an element of the subset or not. Also note that there is a distinction between a set X and the set $\{X\}$ which has one element, the set X.

We will now present an example showing that not every collection of objects can be considered as a set

Theorem 4.2 (Russell's Paradox). *The collection of all sets is not a set.*

Proof: Suppose that the collection S of all sets were itself a set. We could then define the set D of all sets which are not members of themselves, i.e.

$$D = \{X \in S : X \notin X\}.$$

But now, since $D \in S$, we have that $D \in D$ if and only if $D \notin D$, a contradiction. Hence S could not have been a set. □

We can view this as saying that some collections of objects are too large to be sets. We can also view it as saying that the collection of objects satisfying a given property is not always a set, that we should limit ourselves to those objects in some starting set.

We know define some of the standard operations on sets.

Definition 4.3. Let X and Y be sets.

1. We define the *union*, *intersection*, and *difference* of X and Y as follows:

$$X \cup Y = \{a : a \in X \vee a \in Y\}$$
$$X \cap Y = \{a : a \in X \wedge a \in Y\}$$
$$X \setminus Y = \{a : a \in X \wedge a \notin Y\}$$

2. Given two objects a and b, there is an object (a, b) called the *ordered pair of a and b* with the following property:

$$(a_1, b_1) = (a_2, b_2) \Leftrightarrow (a_1 = a_2 \wedge b_1 = b_2).$$

3. We define the *Cartesian product* $X \times Y$ to be the set

$$X \times Y = \{(a, b) : a \in X \wedge b \in Y\}.$$

4. A *function* $f : Y \to X$ is a function whose domain is Y and whose range is a subset of X. We write X^Y to denote the set of functions from Y to X, i.e.

$$X^Y = \{f : \ f : Y \to X\}.$$

We call this the *exponential* of X and Y. We can identify a function $f : Y \to X$ with the set $\{(y, f(y)) : y \in Y\}$.

5. If f is a function and Z is a set, we define the *restriction of f to Z*, $f \restriction Z$, to be the unique function whose domain is $\mathrm{dom}(f) \cap Z$ which agrees with f on its domain. We write $f[Z]$ for the range of $f \restriction Z$, i.e.

$$f[z] = \{f(a) : a \in \mathrm{dom}(f) \cap Z\}.$$

6. An *indexed collection of sets with index set I* is a function f whose domain is the set I and such that $f(i)$ is a set for each $i \in I$. We denote such a collection as $\langle X_i : i \in I\rangle$ to mean $f(i) = X_i$ for each $i \in I$. Given an indexed collection, we define the *union*, *intersection*, and *product* of the collection as follows:

$$\bigcup_{i \in I} X_i = \{a : \exists i \in I(a \in X_i)\}$$
$$\bigcap_{i \in I} X_i = \{a : \forall i \in I(a \in X_i)\}$$
$$\prod_{i \in I} X_i = \{\langle a_i : i \in I\rangle : \forall i \in I(a_i \in X_i)\}$$

Thus, the product is the set of all functions whose domain is I and such that $f(i) \in X_i$ for all $I \in I$. Note that if we take $X_i = X$ for all $i \in I$, then $\prod_{i \in I} X_i = X^I$, the exponential of X and I.

An important property of products of indexed collections is the following:

Definition 4.4. The *Axiom of Choice* is the assertion that if $\langle X_i : i \in I\rangle$ is an indexed collection of sets such that $X_i \neq \emptyset$ for all $i \in I$, then the product is nonempty, $\prod_{i \in I} X_i \neq \emptyset$.

A function $f \in \prod_{i \in I} X_i$ is called a *choice function* for the collection. The Axiom of Choice says that it is possible to find a choice function for any collection, i.e. we can choose an element from each set in the collection. Although this seems intuitively obvious, the Axiom of Choice has some surprising consequences which we will consider in the next chapter. This axiom turns out to be independent of the other axioms of set theory we will present (it is neither provable nor refutable from them), and we will generally indicate when a proof requires its use.

4.2 Cardinal Numbers

One of the key properties of sets which we will consider is their size.

Definition 4.5. Two sets X and Y are said to be *equinumerous*, $X \approx Y$, if there is a bijection between them, i.e. a one-to-one and onto function $f : X \to Y$.

Lemma 4.6. *The relation $\approx$ is an equivalence relation, i.e.*

1. *$X \approx X$*
2. *$X \approx Y$ if and only if $Y \approx X$*
3. *$X \approx Y$ and $Y \approx Z$ implies $X \approx Z$*

Proof: This is straightforward. □

We can thus associate to each set X an object $\mathrm{card}(X)$, called the *cardinality* or the *cardinal number* of X. We will not specify precisely what these objects are yet. The only property we will need now is the following:

$$X \approx Y \text{ if and only if } \mathrm{card}(X) = \mathrm{card}(Y).$$

Definition 4.7. We say that κ is a *cardinal* or *cardinal number* if there is some set X such that $\kappa = \mathrm{card}(X)$.

In the case of a finite set X we could take the cardinality of X to be the natural number n, where X has n elements. The case of infinite sets will require more subtlety, as well as a precise notion of set, and will be defined in the next chapter.

Definition 4.8. For sets X and Y, we write $X \preccurlyeq Y$ if there is a subset $Y_1 \subseteq Y$ such that $X \approx Y_1$. Note that $X \preccurlyeq Y$ if and only if there is an injective function $f : X \to Y$.

Lemma 4.9. *The relation $\preccurlyeq$ has the following properties:*

1. *If $X \approx X'$ and $Y \approx Y'$, then $X \preccurlyeq Y$ if and only if $X' \preccurlyeq Y'$.*
2. *$X \preccurlyeq X$ for each set X.*

3. *If* $X \preccurlyeq Y$ *and* $Y \preccurlyeq Z$ *then* $X \preccurlyeq Z$.

4. *If* $X \preccurlyeq Y$ *and* $Y \preccurlyeq X$ *then* $X \approx Y$.

5. *For any two sets* X *and* Y, *either* $X \preccurlyeq Y$ *or* $Y \preccurlyeq X$ *(or both).*

Proof: The first three parts are straightforward. The fourth part is known as the Shröder-Bernstein Theorem; we will prove it and the fifth part later using a consequence of the Axiom of Choice known as the Well-Ordering Theorem. □

This allows us to define an ordering on cardinals.

Definition 4.10. For cardinals κ and λ, we say $\kappa \leq \lambda$ if $X \preccurlyeq Y$, where X and Y are any sets with $\kappa = \text{card}(X)$ and $\lambda = \text{card}(Y)$.

Corollary 4.11. *The relation* $\leq$ *on cardinals is a linear ordering.*

We can extend the usual arithmetic operations to the class of cardinal numbers.

Definition 4.12 (Cardinal Arithmetic). Let $\kappa = \text{card}(X)$ and $\lambda = \text{card}(Y)$ be cardinals, with $X \cap Y = \emptyset$. We define:

1. $\kappa + \lambda = \text{card}(X \cup Y)$
2. $\kappa \cdot \lambda = \text{card}(X \times Y)$
3. $\kappa^\lambda = \text{card}\left(X^Y\right)$

Theorem 4.13. *Let* κ, λ, *and* μ *be cardinals. Then:*

1. $\kappa + \lambda = \lambda + \kappa$
2. $(\kappa + \lambda) + \mu = \kappa + (\lambda + \mu)$
3. $\kappa \cdot \lambda = \lambda \cdot \kappa$
4. $(\kappa \cdot \lambda) \cdot \mu = \kappa \cdot (\lambda \cdot \mu)$
5. $\kappa \cdot (\lambda + \mu) = \kappa \cdot \lambda + \kappa \cdot \mu$
6. $\kappa^{\lambda+\mu} = \kappa^\lambda \cdot \kappa^\mu$
7. $\kappa^{\lambda \cdot \mu} = \left(\kappa^\lambda\right)^\mu$
8. $\kappa + 0 = \kappa$, $\kappa \cdot 0 = 0$, $\kappa \cdot 1 = \kappa$, *etc.*

Proof: This is straightforward; several parts are given as exercises. □

Note 4.14. We will see later that addition and multiplication of infinite cardinals is trivial, namely $\kappa + \lambda = \kappa \cdot \lambda = \max(\kappa, \lambda)$. Cardinal exponentiation will be nontrivial, though; we will see that several properties of cardinal exponentiation are independent of the standard axioms of set theory.

Example 4.15. If $\kappa = \text{card}(X)$, then $2^\kappa = \text{card}(\mathcal{P}(X))$, since each subset of X corresponds to its characteristic function, a function from X to the 2-element set $\{0, 1\}$.

Theorem 4.16 (Cantor's Theorem). *For any cardinal κ, we have $\kappa < 2^\kappa$, i.e. for any set X we have $X \preccurlyeq \mathcal{P}(X)$ but $\mathcal{P}(X) \not\preccurlyeq X$.*

Proof: We see that $X \preccurlyeq \mathcal{P}(X)$ using the function $f : X \to \mathcal{P}(X)$ with $f(a) = \{a\}$ for each $a \in X$. Suppose now that $\mathcal{P}(X) \preccurlyeq X$ and let $g : \mathcal{P}(X) \to X$ be an injective function. Define the set D by

$$D = \{a \in X : a \in \text{ran}(g) \wedge a \notin g^{-1}(a)\}.$$

Then $D \subseteq X$ so $D \in \mathcal{P}(X)$. But now we have $g(D) \in \text{ran}(g)$, so $g(D) \in D$ if and only $g(D) \notin g^{-1}(g(D)) = D$, a contradiction. □

Definition 4.17. For an indexed collection $\langle \kappa_i : i \in I \rangle$ of cardinals, we define the sum and product of the collection as

$$\sum_{i \in I} \kappa_i = \text{card}\left(\bigcup_{i \in I} X_i\right)$$

$$\prod_{i \in I} \kappa_i = \text{card}\left(\prod_{i \in I} X_i\right)$$

where $\kappa_i = card(X_i)$ for each $i \in I$ and the sets X_i are pairwise disjoint.

4.3 Ordinal Numbers

Definition 4.18. A *relational structure* is a pair (A, R) where A is a set and $R \subseteq A \times A$ is a binary relation on A. We often write aRb instead of $(a, b) \in R$ or $R(a, b)$.

Definition 4.19. Let (A, R) and (B, S) be relational structures. An *isomorphism* between (A, R) and (B, S) is a bijection $f : A \to B$ such that for all $a_1, a_2 \in A$ we have $a_1 R a_2$ if and only if $f(a_1) S f(a_2)$. We say that (A, R) and (B, S) are *isomorphic*, $(A, R) \cong (B, S)$, if there is an isomorphism between them.

Lemma 4.20. *The isomorphism relation $\cong$ is an equivalence relation on the class of relational structures.*

Proof: This is straightforward. □

We can thus associate to each relational structure (A, R) an object $\text{type}(A, R)$ called the *isomorphism type* of the structure. Again, we defer defining these objects explicitly; the key property is that we should have

$$(A, R) \cong (B, S) \text{ if and only if } \text{type}(A, R) = \text{type}(B, S)$$

for relational structures (A, R) and (B, S).

Definition 4.21. A relational structure (A, R) is said to be *well-founded* if every nonempty subset $X \subseteq A$ has an *R-minimal element*, i.e. if $\emptyset \neq X \subseteq A$ then there is an $a \in X$ for which there is no $b \in X$ with bRa.

Example 4.22. The structure $(\mathbb{N}, <)$ is well-founded since every non-empty subset has a least element. The structure $(\mathbb{Z}, <)$ is not well-founded, since if we take $X = \mathbb{Z}$ every element $a \in \mathbb{Z}$ has some $b \in \mathbb{Z}$ with $b < a$.

Lemma 4.23. *A relational structure (A, R) is well-founded if and only if there is no infinite descending R-chain (i.e. an infinite sequence $\langle a_n : n \in \mathbb{N}\rangle$ such that $a_{n+1}Ra_n$ for all $n \in \mathbb{N}$).*

Proof: If $\langle a_n : n \in \mathbb{N}\rangle$ is an infinite descending R-chain, then the set $X = \{a_n : n \in \mathbb{N}\}$ has no R-minimal element, so (A, R) is not well-founded. Conversely, if (A, R) is not well-founded, let $X \subseteq A$ be a non-empty et with no R-minimal element, i.e. $\forall a \in X \exists b \in X(bRa)$. We can then take a_0 to be any element of X, a_1 to be some element with a_1Ra_0, a_2 to be some element with a_2Ra_1, etc., to form an infinite descending R-chain.

Formally we use the Axiom of Choice to define this sequence as follows. For each $a \in X$ define the non-empty set $X_a = \{b \in X : bRa\}$; the axiom of Choice then tells us that there is a function f with domain X such that $f(a) \in X_a$ for all $a \in X$. We then choose $a_0 \in X$ arbitrarily, and let $a_{n+1} = f(a_n)$ for all $n \in \mathbb{N}$. □

Definition 4.24. A *linear ordering* is a relational structure (A, R) with the following properties:

1. If aRb and bRc then aRc.
2. For any $a, b \in A$, exactly one of the following holds: aRb, $a = b$, or bRa.

This is sometimes called a *strict* linear ordering. A *well-ordering* is a linear ordering which is well-founded.

Note 4.25. If (A, R) is a well-ordering and X is a non-empty subset of A, then X in fact has an *R-least element*, i.e. a (unique) $a \in X$ such that aRb for all $b \in X$ with $b \neq a$.

Definition 4.26. We say that α is an *ordinal* or *ordinal number* if we have $\alpha = \text{type}(A, R)$ for some well-ordering (A, R).

We generally write o.t.(A, R) (*order-type*) instead of type(A, R) when (A, R) is a well-ordering.

Note that for each finite set A with n elements there is exactly one isomorphism type of well-orderings on A, namely the order-type of the usual ordering $<$ on the set $\{0, 1, \ldots, n-1\}$. We thus let n denote the order-type of this ordering.

Definition 4.27. We let $\omega = \text{o.t.}(\mathbb{N}, <)$ be the order-type of the usual ordering on the natural numbers.

Definition 4.28 (Ordinal Arithmetic). Let α and β be ordinals with $\alpha = \text{o.t.}(A, R)$ and $\beta = \text{o.t.}(B, S)$, where $A \cap B = \emptyset$. We define:

1. $\alpha + \beta = \text{o.t.}(A \cup B, R \cup S \cup (A \times B))$
2. $\alpha \cdot \beta = \text{o.t.}(A \times B, \{((a_1, b_1), (a_2, b_2)) : b_1 S b_2 \vee (b_1 = b_2 \wedge a_1 R a_2)\})$
3. $\alpha^\beta = \text{o.t.}(C, T)$ where C is the set of all functions $f : B \to A$ which have *finite support*, i.e. for all but finitely many $b \in B$ we have $f(b) = a_0$ where a_0 is the R-least element of A, and T is the ordering on C given by $f_1 T f_2$ if and only if $f_1(b) R f_2(b)$ where b is the S-greatest $b \in B$ such that $f_1(b) \neq f_2(b)$ (we can find a greatest such b because f_1 and f_2 have finite support).

Note that the ordering of $\alpha + \beta$ looks like a copy of α followed by a copy of β, and the ordering of $\alpha \cdot \beta$ looks like β-many copies of α, i.e. a number of orderings of order-type α, themselves ordered with order-type β. This ordering is called the *anti-lexicographical ordering* on $A \times B$. Note that for exponentiation, the ordering α^2 agrees with the definition of $\alpha \cdot \alpha$.

We should check that these are all well-orderings. We check that the ordering of $\alpha \cdot \beta$ is well-founded, leaving the rest as exercises. Let $X \subseteq A \times B$ be non-empty. Define the set X_B as

$$X_B = \{b \in B : \exists a \in A((a, b) \in X)\}.$$

Then X_B is a non-empty subset of B, so it has an S-least element b_0. Define the set X_A as

$$X_A = \{a \in A : (a, b_0) \in X\}.$$

This is a non-empty subset of A and so has an R-least element a_0. Then the pair (a_0, b_0) is an element of X which is least in the ordering of $A \times B$.

Note 4.29. Unlike cardinal arithmetic, ordinal addition and multiplication are not commutative. For instance, we have that the ordinal $1 + \omega = \omega$, whereas the ordinal $\omega + 1 \neq \omega$ since $\omega + 1$ has a greatest element, whereas ω does not. Hence $1 + \omega \neq \omega + 1$. Similarly, $2 \cdot \omega = \omega$ since this consists of ω-many copies of the 2-element well-order, whereas the ordinal $\omega \cdot 2 = \omega + \omega \neq \omega$ since $\omega + \omega$ contains elements with infinitely many predecessors and ω does not. Hence $2 \cdot \omega \neq \omega \cdot 2$.

Theorem 4.30. *Let α, β, and γ be ordinals. Then:*

1. $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
2. $(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)$
3. $\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$
4. $\alpha^{\beta + \gamma} = \alpha^\beta \cdot \alpha^\gamma$
5. $\alpha^{\beta \cdot \gamma} = \left(\alpha^\beta\right)^\gamma$

6. $\alpha + 0 = 0 + \alpha = \alpha$, $\alpha \cdot 0 = 0 \cdot \alpha = 0$, $\alpha \cdot 1 = 1 \cdot \alpha = \alpha$

7. $\alpha^0 = 1$ *and* $0^\alpha = 0$ *for* $\alpha \neq 0$, $\alpha^1 = \alpha$, $1^\alpha = 1$

Proof: This is straightforward and left as an exercise. □

Definition 4.31. Let (A, R) be a linear ordering. An *initial segment* of (A, R) is a subset of A of the form $\{b : bRa\}$ for some element $a \in A$. Note that the restriction of R to an initial segment is a linear ordering, and if (A, R) is a well-ordering then the restriction of R to an initial segment is a well-ordering. We generally identify an initial segment with this restriction of R.

Theorem 4.32 (Comparability of Well-Orderings). *Let* (A, R) *and* (B, S) *be well-orderings. Then exactly one of the following holds:*

1. $(A, R) \cong (B, S)$

2. $(A, R) \cong$ *some initial segment of* (B, S)

3. $(B, S) \cong$ *some initial segment of* (A, R)

In each case, the isomorphism is unique.

Proof: First we show that an isomorphism is unique if it exists. Suppose that f_1 and f_2 are two different isomorphisms from (A, R) to (B, S) or some initial segment. Then the set

$$\{a \in A : f_1(a) \neq f_2(a)\}$$

is a non-empty subset of A, so let a_0 be the R-least element. Thus, $f_1(a_0) \neq f_2(a_0)$ but $f_1(a) = f_1(a)$ for all $a \in A$ with aRa_0. Since $f_1(a_0) \neq f_2(a_0)$ we have either $f_1(a_0)Sf_2(a_0)$ or $f_2(a_0)Sf_1(a_0)$; assume the former. We must then have $f_1(a_0) \notin \text{ran}(f_2)$ since any element mapping to it would have to be R-less than a_0. This contradicts that the range of f_2 is either all of B or an initial segment, so there could not have been two distinct isomorphisms.

Define a partial function f with $\text{dom}(f) \subseteq A$ be setting

$$\begin{aligned} f(a) = &\text{ the unique } b \in B \text{ such that} \\ &(\{a' : a'Ra\}, R \restriction \{a' : a'Ra\}) \cong (\{b' : b'Sb\}, S \restriction \{b' : b'Sb\}) \\ &\text{if such a } b \text{ exists} \end{aligned}$$

It is clear that $\text{dom}(f)$ is either all of (A, R) or an initial segment of (A, R), and that $\text{ran}(f)$ is either all of (B, S) or an initial segment of (B, S); it will suffice to check that we can not have both of them being proper initial segments. Suppose this were the case, and let a_0 be the R-least element of $A \setminus \text{dom}(f)$ and let b_0 be the S-least element of $B \setminus \text{ran}(f)$. Then we have

$$(\{a' : a'Ra_0\}, R \restriction \{a' : a'Ra_0\}) \cong (\{b' : b'Sb_0\}, S \restriction \{b' : b'Sb_0\})$$

as witnessed by f, so we would have defined $f(a_0) = b_0$, contradicting that $a_0 \notin \text{dom}(f)$. □

The above theorem allows us to define an ordering on the class of ordinal numbers.

Definition 4.33. For ordinals α and β, we set $\alpha < \beta$ if $\alpha = \text{o.t.}(A, R)$ and $\beta = \text{o.t.}(B, S)$ where (A, R) is isomorphic to an initial segment of (B, S). We set $\alpha \leq \beta$ if $\alpha < \beta$ or $\alpha = \beta$.

Corollary 4.34. *For any ordinals α and β, exactly one of the following holds: $\alpha < \beta$, $\alpha = \beta$, or $\beta < \alpha$. Also, if $\alpha < \beta$ and $\beta < \gamma$ then $\alpha < \gamma$.*

Note 4.35. Thus, the ordering $<$ on the collection of ordinals behaves like a linear ordering. We do not call it a linear ordering, though, since our definition of a linear ordering requires the relation to be defined on a *set* and we will see that the collection of ordinals is too large to be a set.

Lemma 4.36. *If (A, R) is a well-ordering and $B \subseteq A$, then the relational structure $(B, R \cap (B \times B))$ is a well-ordering, and*

$$o.t(B, R \cap (B \times B)) \leq o.t.(A, R).$$

Proof: This follows immediately from the comparability of well-orderings. □

Definition 4.37. We let Ord denote the class of all ordinals.

Lemma 4.38. *For any ordinal α, the relational structure*

$$(\{\beta \in \text{Ord} : \beta < \alpha\}, \{(\gamma, \beta) : \gamma < \beta < \alpha\})$$

is a well-ordering of order-type α.

Proof: Let (A, R) be a well-ordering of order-type α. Define $f : A \to \{\beta : \beta < \alpha\}$ by

$$f(b) = \text{o.t}(\{c \in A : cRb\}, R \restriction \{c \in A : cRb\}).$$

It is straightforward to verify that f is an isomorphism, so that $\{\beta : \beta < \alpha\}$ is a well-ordering of order-type α. □

Thus, any well-ordering is isomorphic to some initial segment of the ordinals with the relation $<$.

Lemma 4.39. *Let X be a set of ordinals. Then there is an ordinal γ, denoted $\sup X$, which is the least upper bound of X under $<$, i.e. $\alpha \leq \gamma$ for all $\alpha \in X$ and if $\beta < \gamma$ then there is an $\alpha \in X$ with $\beta < \alpha$.*

Proof: Let $A = \{\alpha : \exists \beta \in X(\beta < \alpha)\}$. Then $(A, <)$ is a well-ordering. Let $\gamma = \text{o.t.}(A, <)$; it is straightforward to verify that $A = \{\alpha : \alpha < \gamma\}$ and that $\gamma = \sup X$. □

Lemma 4.40. *If X is a non-empty set of ordinals then X has a least element under $<$.*

Proof: Let $\alpha = \sup X$; then X is a non-empty subset of $\{\beta : \beta < \alpha\}$. Since this set is well-ordered under $<$, X has a least element under $<$. □

Note 4.41. Thus, $<$ behaves like a well-ordering on the class of ordinals.

Theorem 4.42 (Burali-Forti Paradox). *The class* Ord *of all ordinals is not a set.*

Proof: If Ord were a set, then we would have that the relational structure $(\text{Ord}, <)$ was a well-ordering. Hence we could let $\alpha = \text{o.t.}(\text{Ord}, <)$, and we would have that

$$(\text{Ord}, <) \cong (\{\beta : \beta < \alpha\}, <\restriction \{\beta : \beta < \alpha\})$$

so that $(\text{Ord}, <)$ would be isomorphic to a proper initial segment of itself, contradicting the uniqueness part of the comparability of well-orderings. □

Definition 4.43. Let α be an ordinal. We say that α is a *successor ordinal* if there is an ordinal β such that $\alpha = \beta + 1$. We say that α is a *limit ordinal* if we have $\beta + 1 < \alpha$ for all β with $\beta < \alpha$.

It is straightforward to check that for each ordinal α, exactly one of the following is true: $\alpha = 0$, α is a successor ordinal, or α is a limit ordinal. The ordinal ω is a limit ordinal, whereas $\omega + 1$ is a successor ordinal.

Definition 4.44. If C is a collection of objects (not necessarily a set), then a *definable function with domain* C is a suitably definable rule F which associates to each element a of C a unique object $F(a)$.

We will not specify precisely what we mean by "definable" here. Examples of such functions which we will study include functions defined on the collection of all ordinals. The following theorem gives us a useful method for defining such functions.

Theorem 4.45 (Tranfinite Recursion). *Let $\mathcal{F}$ be the class of all functions whose domain is an initial segment of the ordinals. Let G be a function with domain $\mathcal{F}$. Then there is a unique function F whose domain is* Ord *such that, for all $\alpha \in$ Ord*

$$F(\alpha) = G(F \restriction \{\beta : \beta < \alpha\}).$$

Proof: Say that a function $f \in \mathcal{F}$ is *good* if for all $\beta \in \text{dom}(f)$ we have $f(\beta) = G(f \restriction \{\gamma : \gamma < \beta\})$. We first claim that for each ordinal α there is at most one good f with $\text{dom}(f) = \{\beta : \beta < \alpha\}$. If $f_1 \neq f_2$ were two good such functions, then let γ be the least such that $f_1(\gamma) \neq f_2(\gamma)$. We would thus have $f_1 \restriction \{\beta : \beta < \gamma\} = f_2 \restriction \{\beta : \beta < \gamma\}$, so

$$f_1(\gamma) = G(f_1 \restriction \{\beta : \beta < \gamma\}) = G(f_2 \restriction \{\beta : \beta < \gamma\}) = f_2(\gamma)$$

contradicting our choice of γ.

Thus, for each ordinal α let f_α be the unique good f with $\mathrm{dom}(f) = \{\beta : \beta < \alpha\}$, if such exists. We claim that f_α exists for each α. If not, let α be the least such that f_α does not exist. Then f_β exists for all $\beta < \alpha$, so the function

$$\{(\beta, g(f_\beta)) : \beta < \alpha\}$$

is good, contradicting the choice of α. Hence, if we set $F(\alpha) = G(f_\alpha)$ for each α this will satisfy the conclusion of the theorem. □

Note 4.46. We can think of $F \restriction \{\beta : \beta < \alpha\}$ as the sequence of values $\langle F(\beta) : \beta < \alpha\rangle$; transfinite recursion is then analogous to course-of-values recursion, i.e. G specifies a way of defining $F(\alpha)$ in terms of $<$-previous values of F.

It is often convenient to think of transfinite recursion via cases. To define a function F on the ordinals, we must specify $F(0)$, define $F(\alpha + 1)$ in terms of $F(\alpha)$ for successor ordinals, and define $F(\alpha)$ in terms of $\{F(\beta) : \beta < \alpha\}$ for a limit ordinal α.

We have the related induction principle:

Theorem 4.47 (Transfinite Induction). *Let P be a relation on ordinals which satisfies the following condition:*

- *Whenever $P(\beta)$ is true for all ordinals $\beta < \alpha$, then $P(\alpha)$ is true as well.*

Then the relation $P(\alpha)$ is true for each ordinal α.

Again, this can be expressed in terms of the three cases: 0, successor ordinal, limit ordinal.

Example 4.48. We could have used transfinite recursion to define ordinal arithmetic. For instance, ordinal addition can be defined by transfinite recursion on the second coordinate as follows:

$$\begin{aligned} \alpha + 0 &= \alpha \\ \alpha + (\beta + 1) &= (\alpha + \beta) + 1 \\ \alpha + \delta &= \sup\{\alpha + \beta : \beta < \delta\} \text{ for a limit ordinal } \delta \end{aligned}$$

We could use transfinite induction to check that this definition agrees with the one given previously for all ordinals α and β.

4.4 Initial Ordinals and Cardinals

We will use the above properties of the ordinals to give a more precise definition of cardinal numbers.

Recall the Axiom of Choice introduced earlier, and the notion of a choice function for an indexed collection of sets. We shall now introduce the notion of a choice function for a single set. We shall generally indicate which proofs require the Axiom of Choice, as this axiom is independent of the other axioms of set theory we will present in the next chapter.

Lemma 4.49 (AC). *For each set X there is a choice function for X, i.e. a function*

$$c : \mathcal{P}(X) \setminus \{\emptyset\} \to X$$

such that $c(Y) \in Y$ for each non-empty subset Y of X.

Proof: Let $I = \mathcal{P}(X) \setminus \{\emptyset\}$ and let $X_i = i$ for all $i \in I$. The Axiom of Choice says that the product $\prod_{i \in I} X_i$ is non-empty, so there is a function c with $\mathrm{dom}(c) = I$ and $c(o) \in X(i)$ for all $i \in I$, i.e. $c : \mathcal{P}(X) \setminus \{\emptyset\} \to X$ is such that $c(Y) \in Y$ for $\emptyset \neq Y \subseteq X$. □

Theorem 4.50 (Well-Ordering Theorem) (AC). *For each set X there is a relation $R \subseteq X \times X$ such that the relational structure (X, R) is a well-ordering.*

Note 4.51. Neither the relation nor even its order-type is unique. For instance, let $X = \mathbb{N}$ be the set of natural numbers. Then both the relations R_1 and R_2 are well-orderings of X, where

$$R_1 = \{(n, m) \in \mathbb{N}^2 : n < m\}$$
$$R_2 = \{(n, m) \in \mathbb{N}^2 : 0 < n < m\} \cup \{(n, 0) : 0 < n\}$$

The structure $(\mathbb{N}, R_1)$ is the usual ordering of the natural numbers and has order-type ω, whereas the structure $(\mathbb{N}, R_2)$ has order-type $\omega + 1$ with 0 being the R_2-greatest element. We have seen that $\omega \neq \omega + 1$, i.e. $(\mathbb{N}, R_1) \ncong (\mathbb{N}, R_2)$.

Proof: It will suffice to prove the following: For any set X, there is an ordinal α such that $X \approx \{\beta : \beta < \alpha\}$. The well-ordering of the second set can thus be transferred to X.

Fix a choice function c for X, and fix an object $a_0 \notin X$. We define a function F on the ordinals by transfinite recursion:

$$F(\alpha) = \begin{cases} c(X \setminus \{F(\beta) : \beta < \alpha\}) & \text{if } X \setminus \{F(\beta) : \beta < \alpha\} \neq \emptyset \\ a_0 & \text{otherwise} \end{cases}$$

We claim first that there is an ordinal α with $F(\alpha) = a_0$. If not, we have $F(\alpha) \in X$ for all ordinals α, so by the construction of F we have $F(\alpha) \neq F(\beta)$ for $\alpha \neq \beta$. Hence F is injective, so if we define the set Y as

$$Y = \mathrm{ran}(F) = \{a \in X : \exists \alpha (F(\alpha) = a)\}$$

then F has an inverse function F^{-1} defined on Y which has $\mathrm{ran}(F^{-1}) = \mathrm{Ord}$, so Ord would be a set (since the range of a function applied to a set is a set). This contradicts the Burali-Forti Paradox.

So let α be the least ordinal such that $F(\alpha) = a_0$, and note that $F(\beta) = a_0$ whenever $\alpha \leq \beta$. We then have that $F \restriction \{\beta : \beta < \alpha\}$ is one-to-one and onto X since we must have $X \setminus \{F(\beta) : \beta < \alpha\} = \emptyset$. Thus $\{\beta : \beta < \alpha\} \approx X$ as desired. □

Note 4.52. The above theorem shows that we can prove the Well-Ordering Theorem using the Axiom of Choice. The converse is true: We can prove the Axiom of Choice using the Well-Ordering Theorem. The idea is that if we have a well-ordering of a set, we can define a choice function by always choosing the least element under the well-ordering. There are other theorems equivalent to the Axiom of Choice, the most commonly used being Zorn's Lemma. We shall discuss the Axiom of Choice in greater detail later.

Since there are many possible well-orderings we can place on a given set, we would like to choose one in a more definitive fashion. We can at least specify the order-type canonically.

Definition 4.53. We say that an ordinal α is an *initial ordinal* if for all $\beta < \alpha$ we have

$$\{\gamma : \gamma < \alpha\} \not\approx \{\gamma : \gamma < \beta\}.$$

Every finite ordinal is an initial ordinal. So is the ordinal ω, since any ordinal less than ω must be some finite n, and

$$\{\gamma : \gamma < n\} \not\approx \{\gamma : \gamma < \omega\}$$

since the first set is finite whereas the second is infinite. The ordinal $\omega + 1$ is not an initial ordinal, since

$$\{\gamma : \gamma < \omega\} \approx \{\gamma : \gamma < \omega + 1\}.$$

Definition 4.54. For any set X, let $|X|$ be the least ordinal α such that $X \approx \{\beta : \beta < \alpha\}$.

We know such an α exists by the Well-Ordering Theorem. Note that $|X|$ is an initial ordinal.

Lemma 4.55. *If $X \subseteq \{\beta : \beta : \alpha\}$, then $|X| \leq \alpha$.*

Proof: This is immediate from Lemma 4.36. □

Theorem 4.56. *Let X and Y be sets. Then:*

1. *$X \approx Y$ if and only if $|X| = |Y|$.*
2. *$X \preccurlyeq Y$ if and only if $|X| \leq |Y|$.*

Proof: This is straightforward using the previous lemma. □

Hence, we have $\text{card}(X) = \text{card}(Y)$ if and only if $|X| = |Y|$. We can thus use this as our definition of cardinality (still leaving the definition of ordinals for later).

Definition 4.57. For a set X, we let $\text{card}(X) = |X|$.

We can now complete the proof of a theorem stated earlier.

Theorem 4.58. *Let X and Y be sets. Then:*

1. *Either $X \preccurlyeq Y$ or $Y \preccurlyeq X$ (or both).*
2. *If $X \preccurlyeq Y$ and $Y \preccurlyeq X$ then $X \approx Y$.*

Proof: This follows from the previous theorem, using the comparability of well-orderings. □

We can now prove some results about cardinal arithmetic. To start, we see that addition and multiplication of infinite cardinals is trivial.

Theorem 4.59. *If κ is an infinite cardinal then $\kappa \cdot \kappa = \kappa$.*

Proof: Suppose not, and let κ be the least cardinal for which this fails. Thus κ is an infinite initial ordinal and $\lambda \cdot \lambda < \kappa$ for all cardinals $\lambda < \kappa$. We will show that this implies that $\kappa \cdot \kappa = \kappa$.

Let $A = \{\alpha \in \text{Ord} : \alpha < \kappa\}$ and let $R =<\restriction A$, so that (A, R) is a well-ordering of order-type κ. Hence $|A| = \kappa$ but every initial segment I of (A, R) has $|I| < \kappa$. Let $B = A \times A$ and define a relation S on $B \times B$ by

$$\begin{aligned}(\alpha_1, \beta_1)S(\alpha_2, \beta_2) \text{ iff } &(\max(\alpha_1, \beta_1) < \max(\alpha_2, \beta_2)) \vee \\ &(\max(\alpha_1, \beta_1) = \max(\alpha_2, \beta_2) \wedge \alpha_1 < \alpha_2) \vee \\ &(\max(\alpha_1, \beta_1) = \max(\alpha_2, \beta_2) \wedge \alpha_1 = \alpha_2 \wedge \beta_1 < \beta_2)\end{aligned}$$

It is straightforward to check that S is a well-ordering of B. Note that if J is an initial segment of (B, S) then we must have $J \subseteq I \times I$ for some initial segment I of (A, R). Since then $|I| < \kappa$, we must have $|J| < \kappa$ by our assumption. This says that (A, R) can not be isomorphic to any initial segment of (B, S). We also immediately have that (B, S) can not be isomorphic to any initial segment of (A, R), so by the comparability of well-orderings we have $(A, R) \cong (B, S)$; in particular $A \approx B = A \times A$ so $\kappa \cdot \kappa = \kappa$ as desired. □

Theorem 4.60. *If κ and λ are infinite cardinals then*

$$\kappa + \lambda = \kappa \cdot \lambda = \max(\kappa, \lambda).$$

Proof: Let $\mu = \max(\kappa, \lambda)$. Then

$$\mu \leq \kappa + \lambda \leq \mu + \mu = 2 \cdot \mu \leq \mu \cdot \mu = \mu$$

and

$$\mu \leq \kappa \cdot \lambda \leq \mu \cdot \mu = \mu.$$

□

Theorem 4.61. *If λ is an infinite cardinal and $2 \leq \kappa \leq 2^\lambda$, then $\kappa^\lambda = 2^\lambda$.*

Proof: We have $2^\lambda \leq \kappa^\lambda \leq \left(2^\lambda\right)^\lambda = 2^{\lambda \cdot \lambda} = 2^\lambda$. □

We can use the ordinals to give an enumeration of the infinite cardinals.

Definition 4.62. Let β be an ordinal. We define β^+ to be the least initial ordinal κ such that $\kappa > \beta$.

Note that β^+ is a cardinal, and if β is itself a cardinal then β^+ is the least cardinal greater than β; we refer to it as the *successor* of β in this case.

Definition 4.63. We define a sequence of ordinals ω_α for $\alpha \in \text{Ord}$ by transfinite recursion:

$$\begin{aligned} \omega_0 &= \omega \\ \omega_{\alpha+1} &= \omega_\alpha^+ \\ \omega_\delta &= \sup\{\omega_\alpha : \alpha < \delta\} \text{ for limit ordinals } \delta \end{aligned}$$

This sequence then enumerates all of the infinite initial ordinals, i.e. all of the infinite cardinals. To avoid confusion, we generally use the notation $\aleph_\alpha$ to refer to the cardinal ω_α.

Although cardinal addition and multiplication is simple, cardinal exponentiation turns out to be much more interesting. An important problem is the *Continuum Problem*: What is the cardinality of the set of real numbers $\mathbb{R}$? We have seen that $|\mathbb{R}| = 2^{\aleph_0}$, and Cantor's Theorem tells us that $2^{\aleph_0} > \aleph_0$, i.e. $2^{\aleph_0} \geq \aleph_1$. The *Continuum Hypothesis*, CH, is the statement that $2^{\aleph_0} = \aleph_1$. This statement turns out to neither be provable nor refutable using the standard axioms of mathematics, and is said to be *independent* of these axioms. Thus, CH is consistent, but it is also consistent that, e.g. $2^{\aleph_0} = \aleph_2$ (Besides Cantor's Theorem, there is one additional requirement that $2^{\aleph_0}$ have uncountable cofinality, discussed below, but these turn out to be the only restrictions on the possible size of the continuum). Another formulation of CH is that every infinite subset of $\mathbb{R}$ either be countable or have the same cardinality as $\mathbb{R}$.

More generally, Cantor's Theorem again tells us that $2^{\aleph_\alpha} \geq \aleph_{\alpha+1}$ for each ordinal α. The *Generalized Continuum Hypothesis*, GCH, is the statement that $2^{\aleph_\alpha} = \aleph_{\alpha+1}$ for each ordinal α. Equivalently, GCH says that $2^\kappa = \kappa^+$ for each infinite cardinal κ. This also turns out to be independent of the standard axioms of set theory.

We say that a cardinal κ is *uncountable* if $\kappa > \aleph_0$. We consider several properties of uncountable cardinals.

Definition 4.64. Let λ be an uncountable cardinal. We say that λ is a *successor cardinal* if $\lambda = \kappa^+$ for some cardinal κ. This is equivalent to saying that $\lambda = \aleph_{\alpha+1}$ for some ordinal α. We say that λ is a limit cardinal if $\kappa^+ < \lambda$ for every cardinal $\kappa < \lambda$. This is equivalent to saying that $\lambda = \aleph_\delta$ for some limit ordinal δ.

Hence every uncountable cardinal is either a successor cardinal or a limit cardinal (but not both). All of the cardinals $\aleph_n$ for $n \geq 1$ are successor cardinals; the first limit cardinal is $\aleph_\omega$.

Definition 4.65. Let λ be an uncountable cardinal. We say that λ is a *strong limit cardinal* if $2^\kappa < \lambda$ for each cardinal $\kappa < \lambda$.

Note that every strong limit cardinal is a limit cardinal. If the GCH is true, then every limit cardinal is a strong limit cardinal, but this need not be true in general.

Definition 4.66. let κ be an infinite cardinal. The *cofinality of* κ, $\text{cof}(\kappa)$, is the least ordinal α such that $\kappa = \sum_{i<\alpha} \lambda_i$ for some indexed family of cardinals $\langle \lambda_i : i < \alpha \rangle$ with $\lambda_i < \kappa$ for all $i < \alpha$.

Note that $\text{cof}(\kappa) \leq \kappa$ is always an initial ordinal, i.e. a cardinal.

Definition 4.67. We say that an infinite cardinal κ is *regular* if $\text{cof}(\kappa) = \kappa$. If $\text{cof}(\kappa) < \kappa$ we say that κ is *singular*.

For instance $\aleph_0$ is regular since it is not the sum of finitely many finite cardinals. On the other hand, $\aleph_\omega$ is singular, since $\aleph_\omega = \sum_{n<\omega} \aleph_n$ so that $\text{cof}(\aleph_\omega) = \aleph_0$.

Theorem 4.68. *Every infinite successor cardinal is regular.*

Proof: Let $\lambda = \kappa^+$ be an infinite successor cardinal. Suppose that $\mu = \text{cof}(\lambda) < \lambda$, so that

$$\lambda = \sum_{i<\mu} \lambda_i$$

with $\lambda_i < \lambda$ for all $i < \mu$. Then $\lambda_i \leq \kappa$ and $\mu \leq \kappa$ so

$$\sum_{i<\mu} \lambda_i \leq \sum_{i<\kappa} \kappa = \kappa \cdot \kappa = \kappa < \lambda$$

a contradiction. □

In particular, each of the cardinals $\aleph_n$ for $n \geq 1$ is regular.

We can naturally ask whether the converse of the above theorem is true, i.e. whether every regular cardinal is a successor cardinal. We have that $\aleph_0$ is a regular cardinal which is not a successor; the real question is whether there are others.

Definition 4.69. We say that κ is *weakly inaccessible* if κ is an uncountable regular limit cardinal. We say that κ is *inaccessible* if κ is a regular strong limit cardinal.

Every inaccessible cardinal is weakly inaccessible, and the GCH implies that every weakly inaccessible cardinal is inaccessible, but this need not be true in general. We will see that the existence of weakly inaccessible cardinals can not be proved from the standard axioms of set theory, i.e. it is consistent that every uncountable regular cardinal is a successor cardinal.

4.5 Pure, Well-Founded Sets

We will now refine our notion of set so that the only objects we consider are sets, i.e. we will consider only sets whose elements are all sets, etc.

Definition 4.70. A set X is *transitive* if every element of X is a subset of X, i.e. for each set Y with $Y \in X$ we have $Y \subseteq X$.

Note that this is equivalent to saying that if $Z \in Y \in X$ then $Z \in X$, i.e. the $\in$-relation behaves transitively with respect to X.

Lemma 4.71. *For each set X there is a smallest transitive set $\mathrm{TC}(X)$ with $X \subseteq \mathrm{TC}(X)$, called the transitive closure of X.*

Proof: Define $U(X) = \bigcup X = \bigcup\{Y : Y \in X \text{ is a set}\}$. We then define

$$\begin{aligned} \mathrm{TC}_0(X) &= X \\ \mathrm{TC}_{n+1}(X) &= U(\mathrm{TC}_n(X)) \\ \mathrm{TC}(X) &= \bigcup_{n<\omega} \mathrm{TC}_n(X) \end{aligned}$$

Clearly $X \subseteq \mathrm{TC}(X)$ and $\mathrm{TC}(X)$ is transitive since if $Y \in \mathrm{TC}(X)$ then $Y \in \mathrm{TC}_n(X)$ for some n, so that $Y \subseteq \mathrm{TC}_{n+1}(X) \subseteq X$. It is easy to see that this is the smallest such set, since a transitive set containing X must contain $U(X)$ and so forth. □

For a set X we write $\in\restriction X$ for $\{(x, y) : x, y \in X \wedge x \in y\}$.

Definition 4.72. We say a set X is *well-founded* if the relational structure $(\mathrm{TC}(X), \in\restriction \mathrm{TC}(X))$ is well-founded.

Thus, X is well-founded if and only if there is no infinite descending $\in$-chain $X = X_0 \ni X_1 \ni X_2 \ni \cdots$. Well-founded sets are those which can be "built from the bottom".

Definition 4.73. We say a set X is *pure* if every element of $\mathrm{TC}(X)$ is a set, i.e. X is a set, every element of X is a set, every element of an element of X is a set, and so forth.

Definition 4.74. For each ordinal α define the set R_α by transfinite recursion as follows:

$$\begin{aligned} R_0 &= \emptyset \\ R_{\alpha+1} &= \mathcal{P}(R_\alpha) \\ R_\delta &= \bigcup_{\alpha<\delta} R_\alpha \text{ for limit ordinals } \delta \end{aligned}$$

By transfinite induction we see that each R_α is a transitive, pure, well-founded set, and that $R_\beta \in R_\alpha$ for $\beta < \alpha$.

Theorem 4.75. *We have $X \in \bigcup_{\alpha \in \mathrm{Ord}} R_\alpha$ if and only if X is a pure, well-founded set.*

Proof: One direction is immediate from the above remarks, noting that if $X \in R_\alpha$ then $\mathrm{TC}(X) \subseteq R_\alpha$. For the other, suppose that X is a pure, well-founded set. We claim that for each $Y \in \mathrm{TC}(\{X\})$ there is an ordinal α such that $Y \in R_\alpha$. Suppose not, and choose an $\in$-minimal $Y \in \mathrm{TC}(\{X\})$ for which no such α exists. Thus, for each $Z \in Y$ we can choose let $f(Z)$ be the least ordinal β such that $Z \in R_\beta$. Let $\gamma = \sup\{f(Z) : Z \in Y\}$. Then $Y \subseteq R_\gamma$, so $Y \in \mathcal{P}(R_\gamma) = R_{\gamma+1}$, a contradiction, so the claim is true. In particular, since $X \in \mathrm{TC}(\{X\})$ we have that $X \in R_\alpha$ for some α, as we wished. □

Definition 4.76. We let V be the class of all pure, well-founded sets, i.e.

$$V = \bigcup_{\alpha \in \mathrm{Ord}} R_\alpha.$$

V is referred to as the *universe* of pure, well-founded sets. The set R_α is sometimes denoted V_α.

Definition 4.77. If X is a pure, well-founded set, we let the *rank of* X, $\mathrm{rank}(X)$, be the least ordinal α such that $X \subseteq R_\alpha$. Equivalently,

$$\mathrm{rank}(X) = \sup\{\mathrm{rank}(Y) + 1 : Y \in X\}.$$

Clearly $\mathrm{rank}(R_\alpha) = \alpha$, and

$$R_\alpha = \{X : X \text{ is a pure, well-founded set with } \mathrm{rank}(X) < \alpha\}.$$

We shall restrict our attention to the universe of pure, well-founded sets, so that the only objects we study will be sets. As a result, we will not have any of the naive objects of standard mathematics, such as the natural numbers, real numbers, etc.; however, we can represent all of the standard mathematical objects as sets and hence use these representations in our treatment of set theory. In the rest of this section we will see how to represent several key types of objects as sets, namely: ordered pairs, functions, ordinal numbers, cardinal numbers, natural numbers, real numbers.

Definition 4.78 (ordered pair). Let $(a, b) = \{\{a\}, \{a, b\}\}$.

Thus, for sets a and b we have a representation for the ordered pair (a, b). We check that this satisfies the key property of ordered pairs.

Lemma 4.79. *We have $(a_1, b_1) = (a_2, b_2)$ if and only if $a_1 = a_2$ and $b_1 = b_2$.*

Proof: For the nontrivial direction, suppose $(a_1, b_1) = (a_2, b_2)$. If $a_1 = b_1$ then (a_1, b_1) has only one element since $\{a_1, b_1\} = \{a_1\}$ and hence (a_2, b_2) has only one element, namely a_1, so that $a_2 = b_2 = a_1 = b_1$ and we are done. If $a_1 \neq b_1$ then $\{a_1\}$ and $\{a_2\}$ are the unique singleton elements of (a_1, b_1) and (a_2, b_2), respectively, so we need $a_1 = a_2$. We then have $\{a_1, b_1\} = \{a_2, b_2\}$, from which $b_1 = b_2$. □

Definition 4.80 (function). A *function* is a set f of ordered pairs which is single-valued, i.e.

$$\forall x \forall y \forall z (((x,y) \in f \wedge (x,z) \in f) \Rightarrow y = z).$$

The *domain of* f is $\mathrm{dom}(f) = \{x : \exists y((x,y) \in f)\}$; for $x \in \mathrm{dom}(f)$ we write $f(x)$ for the unique y such that $(x,y) \in f$.

We next introduce a representation of the ordinal numbers due to von Neumann.

Definition 4.81 (von Neumann ordinal). A *von Neumann ordinal* is a transitive, pure, well-founded set A such that the relational structure $(A, \in\restriction A)$ is a well-ordering.

Note that each initial segment of a von Neumann ordinal is a von Neumann ordinal. The key property is the following:

Lemma 4.82. *For each ordinal α there is a unique von Neumann ordinal A_α such that o.t.$(A_\alpha, \in\restriction A_\alpha) = \alpha$.*

Proof: Using transfinite recursion, we can define

$$A_\alpha = \{A_\beta : \beta < \alpha\}.$$

It is straightforward to verify that each A_α is a von Neumann ordinal, and that it is the unique one of order type α using the fact that an initial segment of a von Neumann ordinal is a von Neumann ordinal. □

Note that $\mathrm{rank}(A_\alpha) = \alpha$. Also note that $A_{\alpha+1} = A_\alpha \cup \{A_\alpha\}$. We can compute the first few von Neumann ordinals as follows:

$$\begin{aligned}
A_0 &= \emptyset \\
A_1 &= \emptyset \cup \{\emptyset\} = \{\emptyset\} \\
A_2 &= \{\emptyset\} \cup \{\{\emptyset\}\} = \{\emptyset, \{\emptyset\}\} \\
A_3 &= \{\emptyset, \{\emptyset\}\} \cup \{\{\emptyset, \{\emptyset\}\}\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} \\
&\vdots \quad\quad \vdots \\
A_\omega &= \{A_n : n < \omega\}
\end{aligned}$$

We will henceforth identity the ordinal α with the von Neumann ordinal A_α. Note that we then have $\alpha = \{\beta : \beta < \alpha\}$. For any set of ordinals X we have $\sup X = \bigcup X = \bigcup_{\alpha \in X} \alpha$. For ordinals α and β we have $\alpha < \beta$ if and only if $\alpha \in \beta$, and $\alpha \leq \beta$ if and only if $\alpha \subseteq \beta$.

As before, we can identify cardinals with initial ordinals. We can also identify each natural number n with the von Neumann ordinal $n = A_n$, so that $\mathbb{N} = \omega$.

We briefly sketch representations of $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$. There are many ways to do this. For $\mathbb{Z}$ we can consider $\{0\} \cup (\mathbb{N} \setminus \{0\}) \times \{0,1\}$ where the second coordinate

determines whether we represent n or $-n$. For $\mathbb{Q}$ we can consider equivalence classes of pairs (p, q) where $p, q \in \mathbb{Z}$ with $q \neq 0$ and we identify (p_1, q_1) with (p_2, q_2) if $\frac{p_1}{q_1} = \frac{p_2}{q_2}$. For $\mathbb{R}$ we can use either Dedekind cuts, or equivalence classes of Cauchy sequences of rationals, where we identify two Cauchy sequences if they have the same limit.

Chapter 5

Axiomatic Set Theory

We shall now present an axiomatic framework for set theory in an attempt to formalize the notion of a pure, well-founded set. We will introduce a system of axioms asserting that certain sets exist and that sets have certain properties; a collection satisfying these axioms can be viewed as an interpretation of the notion of a set.

5.1 The Language of Set Theory

Definition 5.1. The language of set theory (LST, or $\mathcal{L}_\in$) consists of two binary relations, $\in$ and $=$. As with arithmetic we have variables $x, y, z, \ldots$ from which we form atomic formulas of the form $x = y$ and $x \in y$. We then form the collection of formulas using the connectives $\neg$, $\vee$, $\wedge$, $\Rightarrow$, and $\Leftrightarrow$, and the quantifiers $\exists$ and $\forall$.

Note 5.2. We did not use the connectives $\Rightarrow$ and $\Leftrightarrow$ earlier; we may view $F \Rightarrow G$ as equivalent to $\neg F \vee G$, and $F \Leftrightarrow G$ as equivalent to $(F \Rightarrow G) \wedge (G \Rightarrow F)$.

The notions of free and bound variables and sentences are the same as before. The intended interpretation is that variables range over the class of pure, well-founded sets and that the $\in$ relation represents membership. We shall discuss below other interpretations, or models of set theory.

Example 5.3. The principle of extensionality can be expressed by the following sentence of LST:

$$\forall x \forall y (x = y \Leftrightarrow \forall u (u \in x \Leftrightarrow u \in y))$$

The only non-logical symbol of the language of set theory is the $\in$ relation; however, in this language we can define all of the other notions we have introduced previously, such as subset, $\subseteq$. We will show how to express these in LST, and shall use them as abbreviations below, but the point is that everything we have discussed so far can be expressed in LST.

Definition 5.4. We introduce the following abbreviations:

1. (unordered pair) "z = {x,y}" is an abbreviation for
$$\forall u(u \in z \Leftrightarrow (u = x \vee u = y)).$$
2. (singleton) "$z = \{x\}$" is an abbreviation for $z = \{x, x\}$.
3. (ordered pair) "$z = (x, y)$" is an abbreviation for $z = \{\{x\}, \{x, y\}\}$.
4. (subset) "$x \subseteq y$" is an abbreviation for $\forall u(u \in x \Rightarrow u \in y)$.
5. (power set) "$z = \mathcal{P}(x)$" is an abbreviation for $\forall y(y \in z \Leftrightarrow y \subseteq x)$.
6. (union of a set of sets) "$z = \bigcup x$" is an abbreviation for
$$\forall u(u \in z \leftrightarrow \exists v(v \in x \wedge u \in v)).$$
7. (union of two sets) "$z = x \cup y$" is an abbreviation for $z = \bigcup\{x, y\}$.
8. (intersection of a set of sets) "$z = \bigcap x$" is an abbreviation for
$$\forall u(u \in z \Leftrightarrow \forall v(v \in x \Rightarrow u \in v)).$$
9. (intersection of two sets) "$z = x \cap y$" is an abbreviation for $z = \bigcap\{x, y\}$.
10. (empty set) "$x = \emptyset$" is an abbreviation for $\forall u(u \notin x)$ (where $u \notin x$ abbreviates $\neg(u \in x)$).

5.2 The Axioms of Set Theory

We now introduce a collection of axioms of set theory.

Definition 5.5. The *Axiom of Extensionality* is the sentence:
$$\forall x \forall y(x = y \Leftrightarrow \forall u(u \in x \Leftrightarrow u \in y)).$$
This expresses the principle that two sets are equal if and only if they have the same elements.

Definition 5.6. The following axioms assert the existence of some basic sets:

1. *Empty Set Axiom:* $\exists x(x = \emptyset)$.
2. *Pairing Axiom:* $\forall x \forall y \exists z(z = \{x, y\})$.
3. *Union Axiom:* $\forall x \exists z(z = \bigcup x)$.
4. *Power Set Axiom:* $\forall x \exists z(z = \mathcal{P}(x))$.

Definition 5.7. The *Axiom of Foundation* is the sentence:

$$\forall x(x \neq \emptyset \Rightarrow \exists u(u \in x \wedge u \cap x = \emptyset)).$$

The Axiom of Foundation expresses the idea that every set has an $\in$-minimal element, since $u \in x$ is such that $u \cap x = \emptyset$ then there is no $v \in x$ with $v \in u$.

Before presenting more axioms we introduce some additional abbreviations.

Definition 5.8. We have the following abbreviations:

1. (Cartesian product) "$z = x \times y$" is an abbreviation for

$$\forall w(w \in z \Leftrightarrow \exists u \exists v(u \in x \wedge v \in y \wedge w = (u, v))).$$

2. (function) "Function(f)" is an abbreviation for a formula which says that f is a function, i.e.

$$\forall w(w \in f \Rightarrow \exists x \exists y(w = (x, y))) \wedge \forall x \forall y \forall z(((x, y) \in f \wedge (x, z) \in f) \Rightarrow y = z)$$

3. (value of a function) "$y = f(x)$" is an abbreviation for

$$\text{Function}(f) \wedge (x, y) \in f.$$

4. (domain of a function) "$z = \text{dom}(f)$" is an abbreviation for

$$\text{Function}(f) \wedge \forall x(x \in z \Leftrightarrow \exists y(x, y) \in f).$$

5. (generalized Cartesian product) "$z = \prod f$" is an abbreviation for

$$\text{Function}(f) \wedge \forall g(g \in z \Leftrightarrow (\text{dom}(g) = \text{dom}(f) \wedge \forall x(x \in \text{dom}(f) \Rightarrow g(x) \in f(x)))).$$

We now continue with our list of axioms.

Definition 5.9. The *Axiom of Choice*, AC, is the sentence:

$$\forall f \left((\text{Function}(f) \wedge \forall x(x \in \text{dom}(f) \Rightarrow f(x) \neq \emptyset)) \Rightarrow \prod f \neq \emptyset \right).$$

This just expresses in LST our earlier definition that the product of a family of non-empty sets is non-empty.

Definition 5.10. The *Axiom of Infinity* is the sentence:

$$\exists z(\emptyset \in z \wedge \forall x(x \in z \Rightarrow x \cup \{x\} \in z)).$$

The point of the Axiom of Infinity is to ensure that there is at least one infinite set; note that the set z here must have $\omega \subseteq z$ and so be infinite.

The rest of our axioms will fall into two *schemes*, i.e. infinite collections of axioms. The reason for this is that we will want axioms saying that certain things are true for all definable properties; since our language does not allow us to quantify over these properties we must introduce a separate axiom for each property.

Definition 5.11. If F is a formula with free variables $x_1, \ldots, x_n$, then the *universal closure* of F is the sentence $\forall x_1 \cdots \forall x_n F$.

Definition 5.12. For any formula $F(u)$ with free variable u we write "$z = \{u \in x : F(u)\}$" as an abbreviation for $\forall u(u \in z \Leftrightarrow (u \in x \wedge F(u)))$.

Definition 5.13. The *Comprehension Scheme* is the set of axioms consisting of the universal closure of each formula of the form

$$\exists z \forall u(u \in z \Leftrightarrow (u \in x \wedge F(u)))$$

where $F(u)$ is a formula in which z does not occur freely.

Note 5.14. Thus, the comprehension scheme asserts that, given a set x and a definable relation we can form the subset of x consisting of those elements which satisfy the relation. The requirement that z is not a free variable of F is necessary to prevent trivial contradictions such as the set $z = \{u \in x : u \notin z\}$.

Example 5.15. Together with the other axioms we have introduced, the Comprehension Scheme can be used to show that the domain of a function exists, since we have

$$\mathrm{dom}(f) = \{x \in \bigcup\bigcup f : \exists y(x, y) \in f\}.$$

The Comprehension Scheme can be used to show that the collection of elements satisfying a given definable property is a set, provided that we have some starting set which we know contains all of the elements (in this example, the set $\bigcup\bigcup f$).

Definition 5.16. We write "$\exists! x$" to mean "there exists a unique x", i.e. "$\exists! x F(x)$" is an abbreviation for $\exists x \forall y(F(y) \Leftrightarrow x = y)$.

Definition 5.17. The *Replacement Scheme* is the set of axioms consisting of the universal closure of each formula of the form

$$\forall u(u \in x \Rightarrow \exists! v F(u, v)) \Rightarrow \exists y \forall v(v \in y \Leftrightarrow \exists u(u \in x \wedge F(u, v)))$$

where $F(u, v)$ is a formula in which y does not occur freely.

Note 5.18. The Replacement Scheme formalizes the notion that if we have any "definable procedure" applied to a set, then the range is a also a set. If we have an actual function f (i.e. a set of ordered pairs) then the existence of the range of f follows from comprehension much as did the domain above; the key point is that the definable procedure here may have a domain which is too large to be a set, as in the procedures we have used in definitions by transfinite recursion on the ordinals. Again, the requirement that y not occur freely in $F(u, v)$ is used to avoid trivial contradictions.

We now define the standard collection of axioms of set theory, treating the Axiom of Choice separately.

Definition 5.19 (Zermelo-Fraenkel Set Theory). The axioms of *Zermelo-Fraenkel Set Theory*, ZF, are the following:

1. Axiom of Extensionality
2. Empty Set Axiom
3. Pairing Axiom
4. Union Axiom
5. Power Set Axiom
6. Foundation Axiom
7. Axiom of Infinity
8. Comprehension Scheme
9. Replacement Scheme

Definition 5.20 (ZFC). The axioms of *Zermelo-Fraenkel Set Theory with Choice*, ZFC, consist of the axioms of ZF together with the Axiom of Choice.

Note 5.21. The Empty Set Axiom is redundant given the Axiom of Infinity and the Comprehension Scheme, since given any set x the set $z = \{u \in x : u \neq u\}$ is the empty set. All of the other axioms can be shown to be independent of one another, i.e. none of the other axioms is a logical consequence of the others.

The axioms of ZFC are the commonly accepted set-theoretic foundation of mathematics, and are sufficient to prove all of the theorems of standard mathematics. In particular, all of the results presented in the previous chapter can be formalized and proved within ZFC. Much of axiomatic set theory involves studying the logical consequences of this set of axioms.

5.3 Models of Set Theory

The intended interpretation of the language of set theory is the class of pure, well-founded sets; however, as this notion is necessarily somewhat vague, we wish to consider other possible interpretations. That is, we consider various possible collections of "sets" and interpretations of the $\in$-relation on these structures. We recall the following definition:

Definition 5.22. A *relational structure* is a pair (A, E) where A is a set and $E \subseteq A \times A$ is a binary relation on A.

Note 5.23. We have specified here that A is a set. It will also make sense to consider relational structures where A is a definable class; we shall discuss this later.

Definition 5.24. Let (A, E) be a relational structure, and let F be a sentence of LST. We say F *is true in* (A, E) if F is true when we interpret variables as ranging over the elements of A and interpret "$x \in y$" as meaning xEy. We also say (A, E) *models* F, $(A, E) \models F$. Otherwise, we say F *is false in* (A, E).

If $F(x_1, \ldots, x_n)$ is a formula with free variables $x_1, \ldots, x_n$ and $a_1, \ldots, a_n \in A$, then we can similarly define the truth value of $F(a_1, \ldots, a_n)$ in (A, E).

Example 5.25. We say a relational structure (A, E) is *extensional* if the Axiom of Extensionality is true in (A, E). This is equivalent to the following condition: If $a, b \in A$ and $a \neq b$, then $\{c \in A : cEa\} \neq \{c \in A : cEb\}$. For instance, any linear ordering is extensional; however, the structure

$$A = \{0, 1, 2\}$$
$$E = \{(0, 1), (0, 2)\}$$

is not extensional because $\{c \in A : cE1\} = \{0\} = \{c \in A : cE2\}$.

Definition 5.26. Let S be a set of sentences of LST. A *model of* S is a relational structure (A, E) such that each of the sentences of S is true in (A, E). We say that S is *consistent* if there is a model of S; otherwise we say that S is *inconsistent*. We say that a sentence F of LST is a *logical consequence of* S, $S \vdash F$, if F is true in every model of S.

Note that if S is inconsistent then any sentence F is a logical consequence of S. When S is consistent we may ask which sentences are logical consequences of S; this is akin to asking which conclusions follow when we take S as a set of axioms. For instance, the sentence $\forall x(x \notin x)$ is a logic consequence of the Pairing Axiom and the Axiom of Foundation, since the Pairing Axiom tells us that the set $\{x\}$ exists and the Axiom of Foundation tells us that there is an element $u \in \{x\}$ such that $u \cap \{x\} = \emptyset$; we must have $u = x$ so that $x \cap \{x\} = \emptyset$, hence $x \notin x$.

An important type of model is one in which the relation E is the actual membership relation $\in$.

Definition 5.27. An $\in$*-model* is a relational structure (A, E) such that $E = \{(a, b) \in A \times A : a \in b\}$. We write $(A, \in \restriction A)$ in this case and identify the structure with the set A.

Definition 5.28. A *model of ZF (resp. ZFC)* is a relational structure which satisfies all of the axioms of ZF (resp. ZFC). We say that a sentence F is a *theorem of ZF (resp. ZFC)* if it is a logical consequence of ZF (resp. ZFC).

Much of axiomatic set theory is the study of models of ZF and ZFC in order to determine which sentences are logical consequences of these theories. For instance, the Axiom of Choice is *independent* of the axioms of ZF, i.e. neither

AC nor its negation is a logical consequence of ZF (assuming ZF is consistent). Similarly, the Continuum Hypothesis is independent of ZFC. We will see below that AC is consistent with ZF and that CH is consistent with ZFC; showing that the negations are consistent uses the method of *forcing* which we will not cover here.

Note 5.29. We may ask whether it is possible to prove that, say, ZFC is consistent. This presents the following problem. Since ZFC is supposed to encompass the standard axioms of mathematics, the natural interpretation of the above question would amount to asking whether the axioms of ZFC can prove that ZFC is consistent. Gödel's Second Incompleteness Theorem tells us, though, that no sufficiently complex consistent theory can prove its own consistency. Hence, in order to prove that ZFC is consistent we must use additional axioms; we shall give an example of such an additional axiom below.

Definition 5.30. Let (A, E) be a relational structure. We say that a relation $R \subseteq A^k$ is *definable over* (A, E) if there is a formula $F(x_1, \ldots, x_k, y_1, \ldots, y_m)$ of LST and elements $b_1, \ldots, b_m \in A$ such that

$$R = \{(a_1, \ldots, a_k) \in A^k : (A, E) \models F(a_1, \ldots, a_k, b_1, \ldots, b_m)\}.$$

If $B \subseteq A$ we say that R is *definable over* (A, E) *with parameters from* B if we can choose $b_1, \ldots, b_m \in B$ above.

We let $\mathrm{Def}((A, E)) \subseteq \mathcal{P}(A)$ be the set of all subsets of A which are definable over (A, E).

Lemma 5.31. *Let (A, E) be a relational structure. If A is a finite set then $\mathrm{Def}((A, E)) = \mathcal{P}(A)$; if A is infinite then $|\mathrm{Def}((A, E))| = |A|$ (and hence $\mathrm{Def}((A, E))$ is a proper subset of $\mathcal{P}(A)$).*

Proof: If A is finite then any subset of $Y \subseteq A$ is defined by the formula $F(x)$:

$$x = a_1 \vee \cdots \vee x = a_n$$

where $Y = \{a_1, \ldots, a_n\}$.

If A is infinite, then any definable subset is defined using a formula F of LST and a finite set of parameters from A. Since there are only countably many formulas of LST we have

$$|\mathrm{Def}((A, E))| \leq \aleph_0 \cdot |A^{<\omega}| = \aleph_0 \cdot |A| = |A|,$$

and since every singleton $\{a\}$ with $a \in A$ is definable we have $|\mathrm{Def}((A, E))| = |A|$. □

Definition 5.32. Let (A, E) and (A', E') be relational structures. We say that (A', E') is a *substructure* of (A, E), $(A', E') \subseteq (A, E)$, if $A' \subseteq A$ and $E' = E \cap (A' \times A')$. We say that (A', E') is an *elementary substructure* of (A, E), $(A', E') \preccurlyeq (A, E)$, if $(A', E') \subseteq (A, E)$ and, for each formula $F(x_1, \ldots, x_n)$ of LST and elements $a_1, \ldots, a_n \in A'$ we have $(A', E') \models F(a_1, \ldots, a_n)$ if and only if $(A, E) \models F(a_1, \ldots, a_n)$.

Lemma 5.33. *If $(A', E') \subseteq (A, E)$ then we have $(A', E') \preccurlyeq (A, E)$ if and only if every non-empty subset of A which is definable over (A, E) using parameters from A' has non-empty intersection with A'.*

Proof: Suppose first that $(A', E') \preccurlyeq (A, E)$, and let $F(x, y_1, \ldots, y_m)$ be a formula and $a_1, \ldots, a_m \in A'$. Since the set these define over (A, E) is non-empty, the sentence $\exists x F(x, a_1, \ldots, a_m)$ is true in (A, E). Hence this sentence is true in (A', E') as well, so $\exists x(x \in A' \wedge F(x_1, \ldots, x_k, a_1, \ldots, a_m))$, i.e. the set defined by F and $a_1, \ldots, a_m$ has non-empty intersection with A'.

Conversely, suppose every such definable set has non-empty intersection with A'. We prove by induction on formulas that each sentence F is true in (A, E) if and only if it is true in (A', E'). This is immediately true for atomic formulas of the form $a = b$ or $a \in b$ where $a, b \in A'$, since $(A', E') \subseteq (A, E)$. The induction steps for connectives are immediate. If we have a sentence of the form $\exists x F(x)$ which is true in (A', E') then there is some $a \in A'$ such that $F(a)$ is true in (A', E'), hence true in (A, E) and hence $\exists x F(x)$ is true in (A, E). On the other hand, if this sentence is true in (A, E) then the set defined by F is non-empty, hence has non-empty intersection with A', hence $\exists a \in A' F(x)$, and hence the sentence is true in (A', E'). Finally, universal quantifiers are handled using the fact that $\forall x F(x)$ is logically equivalent to $\neg\exists x \neg F(x)$. □

Definition 5.34. We say a relational structure (A, E) is *countable* if A is a countable set.

Theorem 5.35 (Löwenheim-Skolem Theorem). *Every relational structure (A, E) has a countable elementary substructure $(A', E') \preccurlyeq (A, E)$.*

Proof: Let $c : \mathcal{P}(A) \setminus \{\emptyset\} \to A$ be a choice function for A. Define a sequence of sets by

$$\begin{aligned} A_0 &= \emptyset \\ A_{n+1} &= \{c(X) : X \subseteq A \wedge X \neq \emptyset \wedge X \text{ is definable} \\ &\qquad \text{over } (A, E) \text{ using parameters from } A_n\} \\ A' &= \bigcup_{n<\omega} A_n \end{aligned}$$

Then each $A_n \subseteq A_{n+1}$, and each A_n is countable since there are only countably many sets definable over (A, E) using parameters from a countable set. Hence A' is countable. If X is a non-empty subset of A definable over (A, E) using parameters from A', then there is some n such that X is definable using parameters from A_n; hence $c(X) \in X \cap A_{n+1} \subseteq X \cap A'$ and so X has non-empty intersection with A'. Hence, by the previous lemma we have $(A', E') \preccurlyeq (A, E)$. □

Applying this to a model of ZFC we have:

Corollary 5.36 (Skolem Paradox). *If ZFC is consistent, then there is a countable model of ZFC.*

This is called a paradox because the axioms of ZFC prove that there are uncountable sets, e.g. $\mathcal{P}(\omega)$. Thus a countable model of ZFC has elements which it thinks are uncountable sets, but these sets must really be countable in the real universe of sets. The resolution of this paradox is that although in the universe of sets there are bijections between these sets and the natural numbers, these bijections are not elements of the countable model and hence the model does not know that these sets are countable.

The theorem is easily generalized to the following:

Theorem 5.37. *Let κ be an infinite cardinal and let (A, E) be a relational structure such that $|A| \geq \kappa$. Let $X \subseteq A$ be such that $|X| \leq \kappa$. Then there is an elementary substructure (A', E') of (A, E) such that $X \subseteq A'$ and $|A'| = \kappa$.*

5.4 Transitive Models

We are interested in structures which model as much of ZFC as possible. We first consider which structures model the axioms of Extensionality and Foundation.

Definition 5.38. Let S be a set of sentences of LST. A *transitive model of S* is a transitive set X such that the $\in$-model $(X, \in\restriction X)$ satisfies all of the sentences of S. We identify the transitive set X with the relational structure $(X, \in\restriction X)$.

Note that every transitive model X is well-founded and extensional, since for $a \in X$ we have $a \subseteq X$. Up to isomorphism, these turn out to be all of the well-founded and extensional models.

Theorem 5.39. *Let (A, E) be a relational structure which is well-founded and extensional. Then there is a transitive, pure, well-founded set X, called the transitive collapse of (A, E), such that $(A, E) \cong (X, \in\restriction X)$. Moreover, the set X and the isomorphism are unique.*

Proof: Fix some object $a_0 \notin A$. We transfinite recursion on the rank of a pure, well-founded set x, define the function F by

$$F(x) = \begin{cases} \text{the unique } a \in A \text{ such that } & \text{if such an } a \text{ exists} \\ \qquad F[x] = \{b \in A : bEa\} & \\ a_0 & \text{otherwise} \end{cases}$$

Let $X = F^{-1}[A] = \{x : F(x) \in A\}$. When $F(x) = F(y) \in A$ we must have $x = y$, so that F is injective on X, hence the inverse function F^{-1} is well-defined on A and so X is a set. Letting $f = F^{-1} \restriction A$ it is immediate that f is an isomorphism between (A, E) and $(X, \in\restriction X)$, and we see that X is a transitive, pure, well-founded set. Verifying that X and f are unique is straightforward, using induction on the rank of elements of X. □

Corollary 5.40. *If (A, E) is a relational structure, the following are equivalent:*

1. *(A, E) is well-founded and extensional.*

2. *(A, E) is isomorphic to a transitive model X.*

We will hence restrict our attention to transitive models. We wish to characterize when transitive models are models of the various axioms of ZFC. We adapt an earlier definition:

Definition 5.41. Let A be a transitive set. We say a relation $R \subseteq A^k$ is *definable over A* if it is definable over $(A, \in\restriction A)$ using parameters from A. We let $\mathrm{Def}(A) = \mathrm{Def}((A, \in\restriction A))$.

The key idea below is that of *relativizing* a sentence to a structure A, i.e. we replace each quantifier "$\exists x$" by "$\exists x \in A$" and each "$\forall x$" by "$\forall x \in A$".

Lemma 5.42. *Let A be a transitive model.*

1. *A satisfies the Axiom of Extensionality.*
2. *A satisfies the Axiom of Foundation.*
3. *A satisfies the Pairing Axiom if and only if A is closed under pairing, i.e.*
$$\forall a \forall b((a \in A \wedge b \in A) \Rightarrow \{a, b\} \in A).$$
4. *A satisfies the union axiom if and only if A is closed under union, i.e.*
$$\forall a(a \in A \Rightarrow \bigcup a \in A).$$
5. *A satisfies the Empty Set Axiom if and only if $\emptyset \in A$.*
6. *A satisfies the Axiom of Infinity if and only if $\exists a \in A(\omega \subseteq A)$.*
7. *A satisfies the Power Set Axiom if and only if*
$$\forall a(a \in A \Rightarrow \mathcal{P}(a) \cap A \in A).$$
8. *A satisfies the Axiom of Choice if and only if every indexed family of non-empty sets in A which is in A has a choice function in A, i.e. if $\langle a_i \rangle_{i \in I} \in A$ and each a_i is non-empty, then $A \cap \prod_{i \in I} a_i \neq \emptyset$.*
9. *A satisfies the Comprehension Scheme if and only if for each $X \in \mathrm{Def}(A)$ we have*
$$\forall a(a \in A \Rightarrow a \cap X \in A).$$
10. *A satisfies the Replacement Scheme if and only if for each function $F : A \to A$ which is definable over A we have*
$$\forall a(a \in A \Rightarrow F[a] \in A).$$

Proof: This is straightforward, since we have simply relativized all of the requirements to those sets that "A knows about". □

Lemma 5.43. *Let $\delta > \omega$ be a limit ordinal. Then R_δ satisfies all of the axioms of ZFC except possibly the Replacement Scheme.*

Proof: Extensionality and Foundation hold because R_δ is transitive; Empty Set holds because $\delta > 0$ and Infinity holds because $\delta > \omega$. R_δ is always closed under taking unions or subsets, so the Union and the Comprehension Scheme are true. Since δ is a limit ordinal we have that R_δ is closed under power set, pairing and products, and hence satisfies Power Set, Pairing, and the Axiom of Choice. □

In order to determine when R_δ satisfies the Replacement Scheme we consider inaccessible cardinals.

Lemma 5.44. *An infinite cardinal λ is regular if and only if there is no set $X \subseteq \lambda$ with $|X| < \lambda$ and $\sup X = \lambda$.*

Proof: Suppose first there is $X \subseteq \lambda$ with $|X| < \lambda$ and $\sup X = \lambda$. Then $\sum_{\alpha \in X} |\alpha| = \lambda$ so the cofinality of λ is at most $|X|$, so λ is not regular. Conversely, suppose λ is not regular, and let $\kappa = \text{cof}(\lambda) < \lambda$ and $\lambda_i < \lambda$ such that $\sum_{i<\kappa} \lambda_i = \lambda$. Let $X = \{\sum_{i<\alpha} \lambda_i : \alpha < \kappa\}$. Then $X \subseteq \lambda$, $|X| = \kappa < \lambda$, and $\sup X = \sum_{i<\kappa} \lambda_i = \lambda$. □

Lemma 5.45. *If λ is an inaccessible cardinal then:*

1. $\forall x((x \subseteq R_\lambda \wedge |x| < \lambda) \Rightarrow x \in R_\lambda)$
2. $\forall x(x \in R_\lambda \Rightarrow |x| < \lambda)$
3. $|R_\lambda| = \lambda$

Proof: (1) Define $\rho : x \to \lambda$ by $\rho(u) = \text{rank}(u)$ for $u \in x$. Then $\rho[x] \subseteq \lambda$ and $|\rho[x]| \leq |x| < \lambda$, so the previous lemma implies that $\alpha = \sup \rho[x] < \lambda$ since λ is regular. Hence $x \subseteq R_\alpha$ so $x \in R_{\alpha+1} \subseteq R_\lambda$ since λ is a limit ordinal.

(2) We show that $|R_\alpha| < \lambda$ for $\alpha < \lambda$ by transfinite induction. First, $|R_0| = |\emptyset| = 0$. Then

$$|R_{\alpha+1}| = |\mathcal{P}(R_\alpha)| = 2^{|R_\alpha|} = \lambda$$

assuming $|R_\alpha| < \lambda$, since λ is a strong limit cardinal. Finally, if $\delta < \lambda$ is a limit ordinal with $|R_\alpha| < \lambda$ for all $\alpha < \delta$, then

$$|R_\delta| = \left| \bigcup_{\alpha<\delta} R_\alpha \right| \leq \sum_{\alpha<\delta} |R_\alpha| < \lambda$$

since λ is regular.

Now if $x \in R_\lambda$ we have $x \in R_\alpha$ for some $\alpha < \lambda$, so $x \subseteq R_\alpha$ and hence $|x| \leq |R_\alpha| < \lambda$.

(3) Since $\lambda \subseteq R_\lambda$ we have $|R_\lambda \geq \lambda$. Using the fact proved in part (2) we also have

$$|R_\lambda| = \left| \bigcup_{\alpha<\lambda} R_\alpha \right| \leq \sum_{\alpha<\lambda} |R_\alpha| \leq \sum_{\alpha<\lambda} \lambda = \lambda \cdot \lambda = \lambda$$

and equality holds. □

Theorem 5.46. *If λ is an inaccessible cardinal then R_λ is a transitive model of ZFC.*

Proof: Since $\lambda > \omega$ is a limit ordinal we know that R_λ satisfies all of the axioms of ZFC except possibly the Replacement Scheme, so we just need to verify that it satisfies Replacement. We need to show that R_λ is closed under definable functions; we will in fact show that it is closed under all functions. Let $f : R_\lambda \to R_\lambda$ and $a \in R_\lambda$. Then $f[a] \subseteq R_\lambda$ and $|f[a]| \leq |a| < \lambda$ so $f[a] \in R_\lambda$ by the previous lemma. □

We thus have:

Corollary 5.47. *If there is an inaccessible cardinal, then ZFC is consistent.*

Since the above corollary can be proved in ZFC, Gödel's Second Incompleteness Theorem tells us that ZFC can not prove the existence of inaccessible cardinals. We can give a direct proof of this, though.

Theorem 5.48. *If ZFC is consistent, then the existence of an inaccessible cardinal is not a logical consequence of ZFC.*

Proof: We want to show there is a model of ZFC in which there are no inaccessible cardinals. If there are no inaccessible cardinals in V then this is immediate, so suppose there is an inaccessible cardinal. Let λ be the least inaccessible cardinal. Then we have that R_λ is a model of ZFC; we claim that R_λ satisfies the sentence "there are no inaccessible cardinals". Suppose to the contrary that there is $\kappa \in R_\lambda$ such that R_λ satisfies "κ is an inaccessible cardinal". Then it is straightforward to verify that κ really is an inaccessible cardinal, and $\kappa < \lambda$ contradicting that λ is the least inaccessible cardinal. □

We also have the following:

Theorem 5.49. *If there is an inaccessible cardinal then three is a countable transitive model of ZFC.*

Proof: Let λ be an inaccessible cardinal, so that $(R_\lambda, \in \restriction R_\lambda)$ is a model of ZFC. Applying the Löwenheim-Skolem Theorem we have a countable set $A \subseteq R_\lambda$ such that $(A, \in \restriction A) \preccurlyeq (R_\lambda, \in \restriction R_\lambda)$. Then $(A, \in \restriction A)$ is extensional and well-founded, so taking the transitive collapse yields a countable transitive set X such that $(X, \in \restriction X) \cong (A, \in \restriction A)$, hence $(X, \in \restriction X)$ is a countable transitive model of ZFC. □

5.5 Constructible Sets and the Inner Model L

We shall introduce a model L of set theory, known as Gödel's universe of constructible sets. We will use this model to show that both AC and GCH are consistent with the axioms of ZF. This model is called an *inner model* because it is constructed as a submodel of some other model of set theory. Throughout this section we assume the existence of a model V of set theory which we will think of as the universe of sets, but which could be replaced by any other model of set theory.

We will assume that V is a model of the axioms of ZF, but will not assume that it is necessarily a model of AC.

Recall that for any transitive set A, the set $\mathrm{Def}(A)$ consists of all subsets of A which are definable over the structure $(A, \in\restriction A)$ using parameters from A.

Definition 5.50. We define a collection of sets L_α for $\alpha \in \mathrm{Ord}$ by transfinite recursion:

$$\begin{aligned} L_0 &= \emptyset \\ L_{\alpha+1} &= \mathrm{Def}(L_\alpha) \\ L_\delta &= \bigcup_{\alpha<\delta} L_\alpha \text{ for a limit ordinal } \delta \end{aligned}$$

We say a set X is *constructible* if $X \in L_\alpha$ for some ordinal α. The *constructible universe* is the class L of all constructible sets,

$$L = \bigcup_{\alpha\in\mathrm{Ord}} L_\alpha.$$

Lemma 5.51. *For each ordinal α, L_α is a transitive, pure, well-founded set with $L_\alpha \subseteq R_\alpha$.*

Proof: For any transitive, pure, well-founded set A we have

$$A \subseteq \mathrm{Def}(A) \subseteq \mathcal{P}(A)$$

and hence $\mathrm{Def}(A)$ is a transitive, pure well-founded set. From this the lemma follows easily using transfinite induction. □

Lemma 5.52. *For each ordinal α we have $\alpha = L_\alpha \cap \mathrm{Ord}$.*

Proof: Since $L_\alpha \subseteq R_\alpha$ we have $L_\alpha \cap \mathrm{Ord} \subseteq \alpha$. For any transitive set A we have $A \cap \mathrm{Ord} \in \mathrm{Def}(A)$ since a is an ordinal if and only if A satisfies "a is transitive and $\in$ is a linear ordering on a", which is expressible in LST. Hence $L_\alpha \cap \mathrm{Ord} \in L_{\alpha+1}$. From this the lemma follows using transfinite induction. □

We shall see that L is a model of ZFC. We begin with the easy axioms.

Lemma 5.53. *L satisfies the Extensionality, Foundation, Empty Set, Infinity, Union, and Pairing Axioms.*

Proof: L satisfies Extensionality and Foundation since it is transitive and well-founded. The set $\emptyset \subseteq L_0$ is definable, so $\emptyset \in L_1$ and hence L satisfies the Empty Set Axiom; similarly, the set $\omega \subseteq V_\omega = L_\omega$ is definable, so $\omega \in L_{\omega+1}$ and L satisfies the Infinity Axiom. The Pairing Axiom holds in L because L is closed under pairing, since if $x, y \in L_\alpha$ then $\{x, y\}$ is definable over L_α and so $\{x, y\} \in L_{\alpha+1}$. Finally, if $x \in \mathrm{Def}(A)$ for some set A, then $\bigcup x \in \mathrm{Def}(A)$ as well, so if $x \in L_\alpha$ then $\bigcup x \in L_\alpha$, so that L is closed under unions and hence satisfies the Union Axiom. □

Lemma 5.54. *L satisfies the Power Set Axiom.*

Proof: Let $X \in L$. For each $Y \in \mathcal{P}(X) \cap L$, let $\rho(Y)$ be the least β such that $Y \in L_\beta$. Let $\alpha = \sup\{\rho(Y) : Y \in \mathcal{P}(X) \cap L\}$, so $\mathcal{P}(X) \cap L \subseteq L_\alpha$. Then $\mathcal{P} \cap L$ is definable over L_α, since

$$\mathcal{P}(X) \cap L = \{Y \in L_\alpha : Y \subseteq X\}.$$

Thus $\mathcal{P}(X) \cap L \in \mathrm{Def}(L_\alpha) = L_{\alpha+1}$. Thus, for each $X \in L$ we have $\mathcal{P}(X) \cap L \in L$, so that the Power Set Axiom holds in L by our earlier lemma. □

We need some preliminary results before considering the Comprehension and Replacement Schemes.

Lemma 5.55. *Let $f_1, \ldots, f_k : \mathrm{Ord} \to \mathrm{Ord}$ be functions. Then there are arbitrarily large ordinals α such that α is closed under $f_1, \ldots, f_k$, i.e. for all $\beta < \alpha$ and all $1 \leq i \leq k$ we have $f_i(\beta) < \alpha$.*

Proof: Given an ordinal γ, define an increasing sequence of ordinals $\langle \alpha_n \rangle_{n \leq \omega}$ as follows:

$$\begin{aligned} \alpha_0 &= \gamma \\ \alpha_{n+1} &= \max(\alpha_n + 1, \sup\{f_i(\beta) : \beta < \alpha_n, 1 \leq i \leq k\}) \\ \alpha_\omega &= \sup\{\alpha_n : n < \omega\} \end{aligned}$$

Then $\alpha_\omega > \gamma$ and for each $\beta < \alpha_\omega$ and $1 \leq i \leq k$ we have $f_i(\beta) < \alpha_\omega$ as desired. □

Note 5.56. Applying this lemma to the function $f(\alpha) = 2^{|\alpha|}$ we see that there are arbitrarily large strong limit cardinals.

Lemma 5.57 (Reflection Principle). *Let $F(x_1, \ldots, x_n)$ be a formula of LST with free variables $x_1, \ldots, x_n$. Then there are arbitrarily large ordinals α such that F reflects to L_α, i.e. for all $a_1, \ldots, a_n \in L_\alpha$ we have $L \models F(a_1, \ldots, a_n)$ if and only if $L_\alpha \models F(a_1, \ldots, a_n)$.*

Proof: We can replace each occurrence of "$\forall$" in F by "$\neg\exists\neg$" and assume $\forall$ does not occur in F, i.e. F is built from atomic formulas using connectives and existential quantifiers. The idea here is similar to the proof of Lemma 5.33. Let

$\langle G_i : 1 \leq i \leq k \rangle$ enumerate all of the formulas such that a formula of the form "$\exists y G_i(y, x_{i_1}, \ldots, x_{i_{n_i}})$" occurs as a subformula of F; i.e., these are the formulas that occur in the construction of F. For $a_1, \ldots, a_{n_i} \in L$, let

$$\begin{aligned} g_i(a_1, \ldots, a_{n_i}) = \text{the least ordinal } \beta \text{ such that } & a_1, \ldots, a_{n_i} \in L_\beta \text{ and} \\ & \text{such that if } L \models \exists y G_i(y, a_1, \ldots, a_{n_i}) \text{ then} \\ & L \models G_i(b, a_1, \ldots, a_{n_i}) \text{ for some } b \in L_\beta. \end{aligned}$$

Define $f_i : \text{Ord} \to \text{Ord}$ by

$$f_i(\beta) = \sup\{g_i(a_1, \ldots, a_{n_i}) : a_1, \ldots, a_{n_i} \in L_\beta\}.$$

From the previous lemma there are arbitrarily large ordinals α such that α is closed under $f_1, \ldots, f_k$. Such an α satisfies the conclusion of the lemma, as in the proof of Lemma 5.33. □

Note 5.58. The only two properties of L we used in the proof were that $L_\alpha \subseteq L_\beta$ for $\alpha < \beta$ and that $L_\delta = \bigcup_{\alpha<\delta} L_\alpha$ for limit ordinals δ. Call a class of sets H a *cumulative hierarchy* if $H = \bigcup_{\alpha \in \text{Ord}} H_\alpha$ where the H_α's are sets such that $H_\alpha \subseteq H_\beta$ for $\alpha < \beta$ and $H_\delta = \bigcup_{\alpha<\delta}$ for limit ordinals δ. Then the Reflection Principle holds for any cumulative hierarchy, i.e. given and F we can find arbitrarily large α such that F reflects from H to H_α.

In particular, the sets R_α form a cumulative hierarchy with union V. Hence, for any sentence which is true in V there are arbitrarily large ordinals α such that F is true in R_α. This has the following consequence, which implies that we can not replace the Comprehension and Replacement Schemes by a finite set of axioms:

Corollary 5.59. *ZFC is not finitely axiomatizable.*

Proof: If there were a finite collection of sentences $S = \{F_1, \ldots, F_k\}$ such that all of the axioms of ZFC were logical consequences of S, then we could let F be the sentence $F_1 \wedge \cdots \wedge F_k$. Then F is true in V, so there is some α such that R_α is a model of S, and hence R_α is a model of ZFC. Since the above proof can be formalized in ZFC, we would have that ZFC proved that there was a model of ZFC, contradicting the Second Incompleteness Theorem. □

Lemma 5.60. *L satisfies the Comprehension and Replacement Schemes.*

Proof: We consider the Replacement Scheme. Let $f : L \to L$ be a function which is definable over L; we must show that for all $a \in L$ that $f[a] \in L$. Let $F(u, v, z_1, \ldots, z_n)$ be a formula of LST and $c_1, \ldots, c_n \in L$ parameters which define f over L, i.e. for all $u \in L$,

$$f(u) = \text{the unique } v \in L \text{ such that } L \models F(u, v, c_1, \ldots, c_n).$$

Let $a \in L$, and let $b = f[a]$; we will show $b \in L$. Let $\beta \in \text{Ord}$ be such that $a, c_1, \ldots, c_n \in L_\beta$ and $b \subseteq L_\beta$. Applying the Reflection Principle, there is an

$\alpha > \beta$ such that for all $u, v \in L_\alpha$ we have $L \models F(u, v, c_1, \ldots, c_n)$ if and only if $L_\alpha \models F(u, v, c_1, \ldots, c_n)$. We then have that b is definable over L_α, since

$$\begin{aligned} b &= \{v \in L : L \models \exists u(u \in a \wedge F(u, v, c_1, \ldots, c_n))\} \\ &= \{v \in L_\alpha : L_\alpha \models \exists u(u \in a \wedge F(u, v, c_1, \ldots, c_n))\}. \end{aligned}$$

Hence $b \in \mathrm{Def}(L_\alpha) = L_{\alpha+1}$ so $b \in L$.

This finishes the proof that L satisfies the Replacement Scheme; the proof of the Comprehension Scheme is similar. □

Corollary 5.61. $L \models ZF$

We next consider the Axiom of Choice. We need to see that the notion of a set being constructible is sufficiently absolute.

Definition 5.62. Let $\mathrm{Const}(x)$ be a formula which says that x is constructible, i.e. $\mathrm{Const}(x)$ is the formula

$$\exists \alpha \in \mathrm{Ord}\, \exists \langle L_\beta \rangle_{\beta \leq \alpha} (x \in L_\alpha \wedge \forall \beta \leq \alpha (L_\beta = \bigcup \{\mathrm{Def}(L_\gamma) : \gamma < \beta\})).$$

This can be expressed in LST. The chief difficulty is in showing that the relation "$y = \mathrm{Def}(x)$" is expressible in LST; this can be accomplished by showing that there is a finite collection of definable functions such that $\mathrm{Def}(x)$ is the closure of x under these functions.

Definition 5.63. Let "$V = L$" be the sentence which says that every pure, well-founded set is constructible, i.e. the sentence $\forall x \mathrm{Const}(x)$.

Theorem 5.64. *L satisfies the sentence $V = L$.*

Proof: The proof of this is straightforward but technical, once the definition of the predicate $\mathrm{Const}(x)$ is made precise. The key point is that for sets $x, y \in L$ we have $L \models y = \mathrm{Def}(x)$ if and only if $y = \mathrm{Def}(x)$ (in V). □

Note 5.65. There are of course models other than L which satisfy $V = L$. We will refine this below to make suitable such models sufficiently similar to L.

Lemma 5.66. *For each ordinal $\alpha \geq \omega$ we have $|L_\alpha| = |\alpha|$.*

Proof: $L_\omega = V_\omega$ since $\mathrm{Def}(A) = \mathcal{P}(A)$ for a finite set A; hence $|L_\omega| = \aleph_0$. For $\alpha \geq \omega$ we then have $|L_{\alpha+1}| = |\mathrm{Def}(L_\alpha)| = |L_\alpha|$, and for limit ordinals δ we have $|L_\delta| = \sum_{\alpha<\delta} |L_\alpha|$. The lemma then follows using transfinite induction. □

Theorem 5.67. *L satisfies the Axiom of Choice, hence $L \models ZFC$.*

Proof: In fact we will show that AC is a logical consequence of ZF and $V = L$. We will show that L satisfies the sentence "every set can be well-ordered", which implies the Axiom of Choice as previously noted (since a choice function can be explicitly defined from a well-ordering). Something stronger is in fact true, namely that there is a definable (class) well-ordering of all of L.

Note that the previously lemma says that for each α the set L_α can be well-ordered, since a bijection between L_α and α (which is well-ordered) induces a well-ordering of L_α. If we consider that proof more carefully, we can actually define by transfinite recursion a function $F : \text{Ord} \to V$ such that $F(\alpha)$ is a well-ordering of L_α for each α. Since this proof does not require the Axiom of Choice it is a theorem of ZF, and hence true in L. Thus, L satisfies the sentence "each set L_α can be well-ordered". Since we also have that L satisfies $V = L$, L satisfies the sentence "$\forall x \exists \alpha x \in L_\alpha$", which implies $\forall x \exists \alpha x \subseteq L_\alpha$. Since a subset of a well-ordered set is well-ordered, this implies that L satisfies that every set can be well-ordered.

We can define a class well-ordering of L as follows. Let $<_\alpha$ be the well-ordering given by $F(\alpha)$, and let $\alpha(x)$ be the least ordinal α such that $x \in L_\alpha$. Then we can define the ordering $<_L$ as follows:

$$x <_L y \text{ iff } (\alpha(x) < \alpha(y)) \vee (\alpha(x) = \alpha(y) \wedge x <_{\alpha(x)} y).$$

It is straightforward to check that this is a definable well-ordering of L. □

Since we only assumed that V satisfied the axioms of ZF, we have:

Corollary 5.68. *If ZF is consistent, then so is ZFC.*

We conclude by showing that L satisfies the Generalized Continuum Hypothesis.

Lemma 5.69. *There is a sentence S of LST such that for each transitive set A, A satisfies S if and only if $A = L_\alpha$ for some ordinal α.*

Proof: The sentence S is essentially the sentence $V = L$ together with Extensionality. We skip the details as they are straightforward but technical. □

Lemma 5.70 (Condensation Lemma). *If $a \in \mathcal{P}(\kappa) \cap L$ for some infinite cardinal κ, then there is an ordinal α with $|\alpha| = \kappa$ such that $a \in L_\alpha$.*

Proof: Let κ be an infinite cardinal, and let $a \subseteq \kappa$ with $a \in L$. Let $\delta > \omega$ be a limit ordinal such that $a \in L_\delta$. Applying the Generalized Löwenheim-Skolem Theorem to the set $X = \kappa \cup \{a\}$ we can find a set $A \subseteq L_\delta$ such that $X \subseteq A$, $|A| = |X| = \kappa$, and $(A, \in\restriction A) \preccurlyeq (L_\delta, \in\restriction L_\delta)$. Let T be the transitive collapse of A, so that T is a transitive set with $(T, \in\restriction T) \cong (A, \in\restriction A) \cong (L_\delta, \in\restriction L_\delta)$. Since $X \subseteq A$ is a transitive set, we have $X \subseteq T$ so that $a \in T$. Since L_δ satisfies the sentence S from the previous lemma, we have that T does as well, so that $T = L_\alpha$ for some α; since $|T| = |A| = \kappa$, we must have $|\alpha| = \kappa$. Hence $a \in L_\alpha$ with $|\alpha| = \kappa$, as we wished. □

Lemma 5.71. *For any infinite cardinal κ we have $|\mathcal{P}(\kappa) \cap L| \leq \kappa^+$.*

Proof: The Condensation Lemma tells us that $\mathcal{P}(\kappa) \cap L \subseteq L_{\kappa^+}$, and $|L_{\kappa^+}| = |\kappa^+| = \kappa^+$. □

Theorem 5.72. *L satisfies the Generalized Continuum Hypothesis.*

Proof: Since the previous lemma is a theorem of ZFC, it is true in L, and since L satisfies $V = L$, L satisfies that $\mathcal{P}(\kappa) = \mathcal{P}(\kappa) \cap L$; hence $L \models |\mathcal{P}(\kappa)| = \kappa^+$ for any infinite cardinal κ, i.e. $L \models 2^\kappa = \kappa^+$; equivalently, $L \models 2^{\aleph_\alpha} = \aleph_{\alpha+1}$. □

Theorem 5.73. *If ZF is consistent, then so is ZFC + GCH. IF ZF has a transitive model, then so does ZFC + GCH.*

Proof: Starting with a model A of ZF, we can view this model as V and form L within it to obtain L^A, the constructible sets of A. This will then be a model of ZFC + GCH. If ZF has a transitive model A we can do the same thing; in this case L^A will be a transitive model. In this case, since L^A is transitive we can show that $L^A = L_\alpha$ where $\alpha = \text{Ord} \cap A$. model, □

We give one final application of L.

Theorem 5.74. *If ZFC is consistent, then the existence of a weakly inaccessible cardinal is not a logical consequence of ZFC.*

Proof: Suppose there is a weakly inaccessible cardinal, and let κ be the least such. We can show that κ is still weakly inaccessible in L, and since $L \models GCH$ we have that $L \models \kappa$ is inaccessible. Within L, then, we see that L_κ is a model of ZFC in which there are no inaccessible cardinals, and hence no weakly inaccessible cardinals, so we have a model of ZFC in which there are no weakly inaccessible cardinals. □

Note 5.75. Gödel's construction of L shows that both AC and GCH are consistent with ZF. It is also true that ¬AC is consistent with ZF, and that ¬CH is consistent with ZFC. These results are proved using the method of *forcing* introduced by Paul Cohen; together with Gödel's results they show that AC and GCH are independent of the other axioms of set theory.