UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Decidability and Semigroups

Undecidability of the Word Problem for Semigroups

Adrian Camilo Vásconez Núñez

Matemáticas

Trabajo de fin de carrera presentado como requisito para la obtención del título de

Matemático

Quito, 16 de diciembre de 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN DE TRABAJO DE FIN DE CARRERA

Decidability and Semigroups

Undecidability of the Word Problem for Semigroups

Adrian Camilo Vásconez Núñez

Nombre del profesor:	John R. Skukalek
Título académico:	Ph. D.

Quito, 16 de diciembre de 2024

© DERECHOS DE AUTOR

Nombres y apellidos:	Adrian Camilo Vásconez Núñez
Código:	00208899
Cédula de identidad:	1720543279
Lugar y fecha:	Quito, 16 de diciembre de 2024

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en `http://bit.ly/COPETheses`.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through the Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project—in whole or in part—should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, available on `http://bit.ly/COPETheses`.

Free prophecies from a black cat.

—Clean Bandit

Resumen

En este trabajo presentamos un tratamiento accesible de la teoría de la computabilidad y de su aplicación al problema de la palabra en semigrupos. Desarrollamos los fundamentos de las máquinas de Turing y la computabilidad, culminando en la no decidibilidad del problema de la parada. Utilizando este marco teórico, demostramos la no decidibilidad del problema de la palabra en semigrupos, proporcionando un puente natural entre la teoría de la computabilidad y el álgebra abstracta. El trabajo concluye con algunas implementaciones concretas de operaciones aritméticas básicas utilizando máquinas de Turing modificadas.

Palabras clave: computabilidad, máquinas de Turing, problema de la parada, semigrupos, problema de la palabra, decidibilidad.

Abstract

In this work, we present an accessible treatment of computability theory and its application to the word problem for semigroups. We develop the fundamentals of Turing machines and computability, culminating in the undecidability of the halting problem. Using this framework, we demonstrate the undecidability of the word problem for semigroups, thus building a natural bridge between computability theory and abstract algebra. The work concludes with some concrete implementations of basic arithmetic operations using modified Turing machines.

Keywords: computability, Turing machines, halting problem, semigroups, word problem, undecidability.

Agradecimientos

Quiero agradecer a mi madre y a mi padre por su apoyo incondicional para que pueda estudiar. A mi hermana, por siempre saber empujarme para ser mejor. A José Palacios, por estar siempre conmigo, y ser el primer interlocutor para mis ideas.

A John, mi tutor, por su guía durante el proceso de escritura de este trabajo. A mis amigos, Pablo Mendieta y Pablo Padilla por escucharme. A Josué Chávez y Estefanía Coronel, por acompañarme en la pandemia, y a Milena Mora, por siempre estar dispuesta a acompañarme mientras desvarío.

Contents

Introduction

Very few results in mathematics are as thought-provoking as Gödel's Incompleteness Theorems. Their discovery in the twentieth century effectively cut short Hilbert's program of formalizing all of mathematics into a complete and decidable axiomatic system, whose most important achievements were Russell and Whitehead's *Principia Mathematica* and Zermelo–Fraenkel set theory [32].

In very few words, Gödel's Incompleteness Theorems state, respectively:

1. No axiomatizable, consistent theory that is "strong enough"[1] is complete. That is, there is some sentence A in its language for which it cannot prove A nor $\neg A$.

2. Consistent axiomatizable theories that are "strong enough" do not prove their own consistency statements.

Gödel [10] initially proved only that the formal system of the *Principia Mathematica* was incomplete. To get the full Incompleteness Theorems, a suitable development of the notion of computability was necessary. At the time, several candidates for this notion were advanced: λ-definability by Church [8], general recursiveness by Herbrand–Gödel–Kleene [11, 17, 18], computability by Turing [30], 1-definability by Post [20], and binormality by Post [21, 23]. Today, we know that all of these are equivalent in the sense that the sets of computable functions they define are the same [9], but that was anything but clear when they were formulated.

All of these concepts attempt to formalize the intuitive idea of an *algorithm* or of an *effectively calculable function*. It is worth mentioning that prominent logicians like Gödel did not initially believe that such an idea could be formalized [27]. At its core, the idea is philosophical in nature,

[1] This ends up amounting to it being able to represent all computable functions and relations, which, it turns out, is achieved by most interesting theories.

and deals with the question of how to outline human capacity to solve problems in a systematic way.

The statement that every effectively calculable function is a computable function is called *Church's thesis*, and was first stated by Church regarding λ-definability and general recursiveness. This did not convince many logicians, and was only widely accepted when Turing published his work on Turing machines, which defines his own notion of computability.

Although Gödel's Incompleteness Theorems are widely known, the underlying theory of computability remains less well appreciated by many undergraduate mathematicians. Paradoxically, this likely stems from its traditional presentation within the framework of formal logic. In this work, I present some of the fundamental ideas in the theory of computability, and apply them to a problem in abstract algebra. In this way, I hope to provide a more down-to-earth avenue into this subject.

The problem in question is the undecidability of the word problem for semigroups. The word problem for semigroups, first stated by Thue in 1914 [24], was proven undecidable by Post [22] in 1947. This problem is the first link in a chain that connects formal logic with group theory and topology [27]. In 1950, Turing [31] demonstrated the undecidability of the word problem for cancellation semigroups. Building on this work, Novikov proved the undecidability of the word problem for groups in 1955 [7], while Boone [1–6] independently reached the same conclusion between 1954 and 1957. In 1960, Markov used these results to show the undecidability of the homeomorphism problem for compact manifolds [25], and, in 1961, Higman [14] completed this line of research by showing that computability and finite generation are equivalent for groups.

In this work, I follow Davis's [9] presentation of the subject. However, I draw significantly from Miller's [19] more recent treatment, and Grillet's [13] exposition of semigroups. In Chapter 1, I present the fundamentals of Turing machines and the theory of computability, concluding with the undecidability of the halting problem. In Chapter 2, I introduce semigroups and their word problem, and show how we can use Turing machines to demonstrate that the word problem for semigroups is undecidable. Finally, in Chapter 3, I give a short informal "verification" of Church's thesis regarding Turing machines by providing a few illustrative examples.

Chapter 1

Turing machines

1.1 Machinery

1.1.1 Informal description

We can imagine that a Turing machine consists of an infinite tape of adjacent squares, which may be blank or marked with a symbol, together with a machine capable of reading the square on the tape where it is located, and of doing an action based on the mark it reads on the square and on its own internal state. The machine has a finite number of internal states, and the mark read from the square and the current internal state of the Turing machine uniquely determine its behavior. The machine will either do nothing (halt) or:

- Move to the square to the right.
- Move to the square to the left.
- Replace the mark on the current square with a symbol (also) uniquely determined by the mark read and the machine's current internal state.

After doing any one of these three actions, the machine will change its internal state to another one (also) uniquely determined by the mark originally read and its current internal state. Afterward, the process is repeated, with the machine reading the square at its current position.

Also, we shall say that there is one symbol which will be processed in the same way as if the square were blank, as to allow for the machine to "erase" a value on the tape.

This informal description of Turing machines will hopefully be helpful for understanding the following formal definition. We note that this is not the original construction of Turing machines by their namesake [30]. Our construction proposes a Turing machine with a finite but extensible tape, in the sense that whenever the machine reaches one end of the tape, it can "paste" a new square onto the tape. In this way, we avoid having to work with an infinite tape.[1]

1.1.2 Formal definition

Definition 1.1.1 A **Turing machine** Z consists of:

1. A finite set of symbols $\{S_0, S_1, \dots, S_m\}$ called the *alphabet of* Z, where S_0 is a special symbol we will call *blank*.

2. A finite set of symbols $\{q_1, q_2, \dots, q_n\}$, which we will call *internal states*, where q_1 is a special state we will refer to as the *starting state*.

3. A nonempty set of *quadruples*, which are sequences of 4 symbols from the alphabet, from the internal states, or from the set $\{R, L\}$, of one of the following forms:[2]

   - $q_i\ S_j\ S_k\ q_l$,
   - $q_i\ S_j\ R\ q_l$,
   - $q_i\ S_j\ L\ q_l$.

   This set shall be such that no two quadruples in the set have the same combination of the first two symbols.

If Z is a Turing machine, we may sometimes write Z for the set of quadruples of Z.

Definition 1.1.2 An **instantaneous description** is a finite sequence containing (exclusively) a single symbol q_i, for some $i = 1, 2, 3, \dots$, and symbols from the set $\{S_0, S_1, S_2, \dots\}$, such that the q_i is not the last element of the sequence.

Definition 1.1.3 If Z is a Turing machine and α is an instantaneous description, we say that α is an **instantaneous description of** Z if the q_i in α is an internal state of Z.

[1] This construction was first suggested by Post [22].

[2] We may sometimes omit the commas and parentheses when writing sequences $(x_1, x_2, \dots, x_n)$.

We interpret an instantaneous description of a Turing machine as describing a tape marked with the alphabet symbols of the sequence in order, where the machine's current internal state is q_i, and where the machine is located at the square associated with the symbol to the right of the q_i.

Definition 1.1.4 A **tape expression** is a sequence consisting entirely of symbols from the set

$$\{S_0, S_1, S_2, \dots\}.$$

Definition 1.1.5 If Z is a Turing machine, and α and β are two instantaneous descriptions of Z, then we write the **basic move**

$$\alpha \to \beta \quad (Z),$$

or simply $\alpha \to \beta$ when there is no ambiguity, to mean that one of the following holds:

1. There exist (possibly empty) tape expressions P and Q such that[3]

$$\alpha = Pq_iS_jQ,$$
$$\beta = Pq_lS_kQ,$$

where Z contains the quadruple

$$q_i\ S_j\ S_k\ q_l.$$

2. There exist (possibly empty) tape expressions P and Q such that

$$\alpha = Pq_iS_jS_kQ,$$
$$\beta = PS_jq_lS_kQ,$$

where Z contains the quadruple

$$q_i\ S_j\ R\ q_l.$$

[3]Here, we write Pq_iS_jQ for the *concatenation* of the sequences P, (q_i), (S_j), and Q (and similarly for the other cases). We will properly define concatenation in Chapter 2.

3. There exists a (possibly empty) tape expression P such that

$$\alpha = Pq_iS_j,$$
$$\beta = PS_jq_lS_0,$$

where Z contains the quadruple

$$q_i\ S_j\ R\ q_l.$$

4. There exist (possibly empty) tape expressions P and Q such that

$$\alpha = PS_kq_iS_jQ,$$
$$\beta = Pq_lS_kS_jQ,$$

where Z contains the quadruple

$$q_i\ S_j\ L\ q_l.$$

5. There exists a (possibly empty) tape expression Q such that

$$\alpha = q_iS_jQ,$$
$$\beta = q_lS_0S_jQ,$$

where Z contains the quadruple

$$q_i\ S_j\ L\ q_l.$$

Here, case 1 corresponds to the case where the machine changes the symbol on the tape, cases 2 and 3 to it moving to the right, and cases 4 and 5 to it moving to the left. In this way, the quadruples correspond to the instructions or programming of the machine. We need two cases for when the machine moves to account for the possibility that the machine is at the current end of the tape, where it has to "paste" a new square onto the tape.

1.1.3 Some consequences

From our definition of a basic move, and our requirement that all the quadruples of a Turing machine have different combinations of the first two symbols, we immediately get

Theorem 1.1.1 *Let Z be a Turing machine. For any instantaneous description α of Z, there is at most one instantaneous description β of Z such that there is a basic move $\alpha \to \beta$.*

Theorem 1.1.2 *If*

$$\alpha \to \beta \quad (Z) \quad \textit{and} \quad \alpha \to \gamma \quad (Z),$$

then $\beta = \gamma$.

Thus, we say that Turing machines are *deterministic*.

Theorem 1.1.3 *If Z and Z' are Turing machines such that $Z \subset Z'$, then*

$$\alpha \to \beta \quad (Z) \implies \alpha \to \beta \quad (Z').$$

Definition 1.1.6 Let Z be a Turing machine. We say that an instantaneous description α of Z is **terminal with respect to** Z if there is no instantaneous description β of Z such that

$$\alpha \to \beta \quad (Z).$$

This corresponds to when a Turing machine halts.

Definition 1.1.7 Let Z be a Turing machine. A finite sequence $(\alpha_1, \alpha_2, \ldots, \alpha_r)$ of instantaneous descriptions of Z is called a **computation of** Z if

$$\alpha_i \to \alpha_{i+1} \quad (Z)$$

for all integers i, $1 \leq i < r$, where α_r is terminal with respect to Z. Whenever that is the case, we will write

$$\alpha_r = \mathrm{Res}_Z(\alpha_1),$$

and call α_r the **resultant of** α_1 with respect to Z.

We have now defined all of the "machinery" of a Turing machine. We now turn to how we can use this machinery to define computability.

1.2 Computability

We will be working a lot with the set of nonnegative integers, so we adopt the following notation.

Definition 1.2.1 We write Ω to mean the set of nonnegative integers.

We expect to use Turing machines as computers. But for that, we need to be able to give them inputs and interpret their outputs.

Definition 1.2.2 We write S_i^n to mean the tape expression

$$\underbrace{S_i S_i \cdots S_i}_{n \text{ times}}.$$

Furthermore, we shall write 1 for the alphabet symbol S_1, and 0 for the alphabet symbol S_0.

Definition 1.2.3 Let x be in Ω. We write

$$\overline{x} = 1^{x+1} = \underbrace{11\cdots1}_{x+1 \text{ times}}.$$

Furthermore, if $(x_1, x_2, \ldots, x_n)$ is in Ω^n, we write

$$\overline{(x_1, x_2, \ldots, x_n)} = \overline{x_1}0\overline{x_2}0\cdots0\overline{x_n}.$$

Definition 1.2.4 Let α be any instantaneous description. Then, we write $\langle\alpha\rangle$ for the number of occurrences of the symbol 1 in α.

Notice that $\langle\overline{x-1}\rangle = x$ for any positive integer x.

Definition 1.2.5 Let Z be a Turing machine. For each positive integer n, we define the function $\Psi_Z^{(n)} : D \to \Omega$, $D \subset \Omega^n$, as follows. For $(x_1, x_2, \ldots, x_n)$ in Ω^n, let $\alpha_1 = q_1\overline{(x_1, x_2, \ldots, x_n)}$. We have two cases:

- If there exists a computation $(\alpha_1, \alpha_2, \ldots, \alpha_p)$ of Z, then we define

$$\Psi_Z^{(n)}(x_1, x_2, \ldots, x_n) = \langle \alpha_p \rangle = \langle \mathrm{Res}_Z(\alpha_1) \rangle.$$

- If there exists no such computation, i.e., $\mathrm{Res}_Z(\alpha_1)$ is undefined, we leave

$$\Psi_Z^{(n)}(x_1, x_2, \ldots, x_n)$$

  undefined as well.

Finally, we write $\Psi_Z(x)$ for $\Psi_Z^{(1)}(x)$.

In this way, we can use Turing machines to define functions whose domain is a subset of Ω^n (it should be clear that, if it exists, the output of a Turing machine is unique). We defer the discussion of how to translate concrete functions, like

$$f(x, y) = x + y \quad \text{and} \quad g(x, y) = (x + 1)(y + 1),$$

into the appropriate Turing machines to Chapter 3.

We may now define computability.

Definition 1.2.6 A function $f\colon D \to \Omega$, $D \subset \Omega^n$, is called **partially computable** if there exists a Turing machine Z such that, for every $(x_1, x_2, \ldots, x_n)$ in D,

$$f(x_1, x_2, \ldots, x_n) = \Psi_Z^{(n)}(x_1, x_2, \ldots, x_n).$$

We say that f is **computable** if $D = \Omega^n$.

We have only defined computability for a very narrow class of functions. However, the conventional constructions of the negative integers, and rational, real, and complex numbers from the set of nonnegative integers (see, for example, Spivak [26]) should convince the reader that the concept of computability can be extended relatively easily to a very big class of functions. In fact, if we are to accept Church's thesis, our definition of computability should be extensible to

any effectively calculable function—any function whose values can be determined in a systematic way.

We also define computable *sets* and *predicates* as follows.

Definition 1.2.7 Let A be a set. We define the **characteristic function of** A as

$$C_A(x) = \begin{cases} 0, & \text{for } x \text{ not in } A, \\ 1, & \text{for } x \text{ in } A. \end{cases}$$

Definition 1.2.8 A set $A \subset \Omega^n$ is **computable** if its characteristic function C_A is computable.

Definition 1.2.9 Let $(x_1, x_2, \dots, x_n)$ be in Ω^n. A predicate $P(x_1, x_2, \dots, x_n)$ is **computable** if the set

$$\{(x_1, x_2, \dots, x_n) \mid P(x_1, x_2, \dots, x_n)\}$$

is computable.

1.3 Undecidable problems

In this section, we develop more the theory of computability. We will use this theory to show that there exist undecidable problems.

1.3.1 Arithmetization of Turing machines

An important part of the theory of Turing machines is that we can use the natural numbers a kind of cypher for them. We will use the technique of Gödel numbers to create this cypher.

Definition 1.3.1 We define the set

$$\mathbf{U} = \{R, L\} \cup \{S_0, S_1, S_2, \dots\} \cup \{q_1, q_2, q_3, \dots\}.$$

Then, $\mathbf{U}$ is the set of basic symbols we used to define the machinery of Turing machines. Now, to each symbol in $\mathbf{U}$ we assign an odd integer greater than or equal to 3 as follows.

Definition 1.3.2 We define the function $c : \mathbf{U} \to \Omega$ such that

$$R \mapsto 3,$$

$$L \mapsto 5,$$

and, for $i = 0, 1, 2, \ldots$,

$$S_i \mapsto 4i + 7,$$

$$q_{i+1} \mapsto 4i + 9.$$

Notice that c is clearly a one-to-one function.

Definition 1.3.3 For $j = 1, 2, 3, \ldots$, we define p_j as the j-th prime number[4] in increasing order of magnitude.

For example, $p_1 = 2$, $p_2 = 3$, and $p_{10} = 29$. Now, we may define the Gödel number of a sequence of symbols as follows.

Definition 1.3.4 Let $M = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ be a finite sequence of symbols in $\mathbf{U}$. Let $c_j = c(\gamma_j)$ for $j = 1, 2, \ldots, n$. We define the **Gödel number of** M as the integer

$$\mathrm{gn}(M) = \prod_{j=1}^{n} p_j^{c_j}.$$

If M is an empty sequence, we define $\mathrm{gn}(M) = 1$.

The Fundamental Theorem of Arithmetic immediately yields

Theorem 1.3.1 *Let M and N be finite sequences of symbols in* $\mathbf{U}$. *If* $\mathrm{gn}(M) = \mathrm{gn}(N)$, *then* $M = N$.

We may also define the Gödel number of a sequence of sequences of symbols, like for a sequence of quadruples or for a computation, as follows.

[4] See Tattersall [29] for a definition of a prime number, properties, and the Fundamental Theorem of Arithmetic.

Definition 1.3.5 Let $\mathscr{M} = (M_1, M_2, \ldots, M_n)$ be a finite sequence of finite sequences of symbols in **U**. We define the **Gödel number of** $\mathscr{M}$ as the integer

$$\prod_{j=1}^{n} p_j^{\mathrm{gn}(M_j)}.$$

Again, from the Fundamental Theorem of Arithmetic, we get

Theorem 1.3.2 *Let $\mathscr{M}$ and $\mathscr{N}$ be finite sequences of finite sequences of symbols in* **U**. *If their Gödel numbers are the same, then $\mathscr{M} = \mathscr{N}$.*

Theorem 1.3.3 *If M is a finite sequence of symbols, and $\mathscr{M}$ is a finite sequence of finite sequences of symbols, then their Gödel numbers are different.*

Proof: If M is a finite sequence of symbols, then we can write

$$\mathrm{gn}(M) = 2^n m,$$

where n and m are odd integers. If $\mathscr{M}$ is a finite sequence of finite sequences of symbols, then its Gödel number has the form

$$2^{n'} m',$$

where n' is an even integer (because it itself is the Gödel number of a finite sequence of symbols), and m' is an odd integer. By the Fundamental Theorem of Arithmetic, the Gödel numbers of M and $\mathscr{M}$ are different. ■

For example, the Gödel number of the sequence of quadruples

$$(q_1\ 1\ 0\ q_2,\ q_2\ 0\ R\ q_2)$$

is $2^{2^9 3^{11} 5^7 7^{13}} 3^{2^{13} 3^7 5^3 7^{13}}$ or

$$2^{686\,543\,901\,062\,753\,160\,000\,000} 3^{216\,981\,776\,138\,351\,616\,000},$$

which is a rather big number. However, the usefulness of Gödel numbers does not come from

direct operations with them. Instead, we use them because they allow us to enumerate certain interesting sets, as we will see later on.

We may also define Gödel numbers of Turing machines as follows.

Definition 1.3.6 Let Z be a Turing machine. Let $(M_1, M_2, \dots, M_n)$ be any ordering without repetitions of the set of quadruples of Z. Then, the Gödel number of the sequence $(M_1, M_2, \dots, M_n)$ is a **Gödel number of** Z.

Thus, a Turing machine Z with n unique quadruples has $n!$ distinct Gödel numbers.

Definition 1.3.7 Let $(z, x_1, x_2, \dots, x_n, y)$ be in Ω^{n+2}, and n be a positive integer. We define the predicate

$$T_n(z, x_1, x_2, \dots, x_n, y)$$

to mean that there exists a Turing machine Z such that:

- z is a Gödel number of Z.
- y is the Gödel number of a computation $\left(\alpha_1, \alpha_2, \dots, \alpha_p\right)$ with respect to Z.
- α_1 is the instantaneous description

$$q_1\overline{(x_1, x_2, \dots, x_n)}.$$

We state the following theorem without proof. See Davis [9] for a proof using recursive functions.

Theorem 1.3.4 *Let $(z, x_1, x_2, \dots, x_n, y)$ be in Ω^{n+2}. Then, the predicate $T_n(z, x_1, x_2, \dots, x_n, y)$ is computable.*

Definition 1.3.8 Let $(y, x_1, x_2, \dots, x_n)$ be in Ω^{n+1}. The operation of **minimalization** associates with each predicate $P(y, x_1, x_2, \dots, x_n)$ a function $h\colon D \to \Omega$, $D \subset \Omega^n$, such that:

- If there exists y such that $P(y, x_1, x_2, \dots, x_n)$ is true, then $h(x_1, x_2, \dots, x_n)$ is the least such y.

- If not, then $h(x_1, x_2, \ldots, x_n)$ is left undefined.

We write

$$h(x_1, x_2, \ldots, x_n) = \min_y [P(y, x_1, x_2, \ldots, x_n)].$$

Theorem 1.3.5 *Let $(y, x_1, x_2, \ldots, x_n)$ be in Ω^{n+1}. If the predicate $P(y, x_1, x_2, \ldots, x_n)$ is computable, then the function*

$$h(x_1, x_2, \ldots, x_n) = \min_y [P(y, x_1, x_2, \ldots, x_n)].$$

is partially computable. Furthermore, if the domain of h is Ω^n, then h is computable.

We will only provide a sketch of the proof of this theorem. For a full proof, see Davis [9]. In essence, we want to construct a Turing machine E that implements a linear search starting from $y = 0$, and incrementing y by 1, until we find the first y that makes $P(y, x_1, x_2, \ldots, x_n)$ true.

If P is computable, then there is some Turing machine Z_P such that, for any $(y, x_1, x_2, \ldots, x_n)$ in Ω_{n+1},

$$\Psi_{Z_P}^{(n+1)}(y, x_1, x_2, \ldots, x_n) = \begin{cases} 0, & \text{for } P(y, x_1, x_2, \ldots, x_n) \text{ false,} \\ 1, & \text{for } P(y, x_1, x_2, \ldots, x_n) \text{ true.} \end{cases}$$

Now, we may construct a Turing machine A which converts an input $q_1\overline{(x_1, x_2, \ldots, x_n)}$ into $q_3\overline{(0, x_1, x_2, \ldots, x_n)}$. Also, we may edit Z_P to get a Turing machine Z_P' such that it takes inputs of the form $q_3\overline{(k, x_1, x_2, \ldots, x_n)}$, and transforms them into

$$q_m\overline{\left(\Psi_{Z_P}^{(n+1)}(k, x_1, x_2, \ldots, x_n), k, x_1, x_2, \ldots, x_n\right)}. \tag{1.1}$$

We see at once that it is only a matter of constructing a machine B such that it takes an input of the form (1.1), and then:

- If $\Psi_{Z_P}^{(n+1)}(k, x_1, x_2, \ldots, x_n) = 1$, then B transforms the input into $q_r\overline{(k-1)}$, and halts.
- If not, then B transforms the input into $q_3\overline{(k+1, x_1, x_2, \ldots, x_n)}$.

We may now hook A, B, and Z'_P together, and we get a Turing machine E such that

$$h(x_1, x_2, \ldots, x_n) = \Psi_E^{(n)}(x_1, x_2, \ldots, x_n).$$

The second part of the theorem follows by definition. This completes our sketch.

Definition 1.3.9 Let $(M_1, M_2, \ldots, M_n)$ be a finite sequence of finite sequences of symbols. If y is its Gödel number, then we define

$$U(y) = \langle M_n \rangle.$$

We state the following theorem without proof. For a detailed proof, see Davis [9].

Theorem 1.3.6 *Let Z_0 be a Turing machine, and let z_0 be a Gödel number of Z_0. Then:*

- *The domain of the function*

$$\Psi_{Z_0}^{(n)}(x_1, x_2, \ldots, x_n)$$

  is the same as the domain of

$$\min_y [T_n(z_0, x_1, x_2, \ldots, x_n, y)].$$

- *We have*

$$\Psi_{Z_0}^{(n)}(x_1, x_2, \ldots, x_n) = U\big(\min_y [T_n(z_0, x_1, x_2, \ldots, x_n, y)]\big).$$

- *If $T_n(z_0, x_1, x_2, \ldots, x_n, y_0)$ is true for some fixed $(x_1, x_2, \ldots, x_n)$ in Ω^n, then*

$$y_0 = \min_y [T_n(z_0, x_1, x_2, \ldots, x_n, y)].$$

This theorem effectively characterizes certain enumeration properties of Turing machines.

1.3.2 Semicomputable predicates

To better understand the concept of computability and its relationship with the arithmetization of Turing machines, we need to understand the concept of semicomputability.

In this subsection, we shall assume that addition, (proper) subtraction, and composition of computable functions are computable. We will explicitly show this in Chapter 3 for addition, but proofs in terms of Turing machines can be found in Davis [9].

Definition 1.3.10 Let $(x_1, x_2, \ldots, x_n)$ be in Ω^n. We say that the predicate $P(x_1, x_2, \ldots, x_n)$ is **semicomputable** if there exists a partially computable function whose domain is the set

$$\{(x_1, x_2, \ldots, x_n) \mid P(x_1, x_2, \ldots, x_n)\}.$$

Theorem 1.3.7 *Let $(x_1, x_2, \ldots, x_n)$ be in Ω^n. If the predicate $P(x_1, x_2, \ldots, x_n)$ is computable, then it is semicomputable.*

Proof: Let the function $C'_P \colon \Omega^n \to \Omega$ be defined such that

$$C'_P(x_1, x_2, \ldots, x_n) = 1 - C_P(x_1, x_2, \ldots, x_n).$$

If P is computable, then so is C'_P. Therefore, the function

$$\min_y \left[C'_P + y = 0\right]$$

is computable, and its domain is

$$\{(x_1, x_2, \ldots, x_n) \mid P(x_1, x_2, \ldots, x_n)\}. \qquad \blacksquare$$

It turns out that semicomputable predicates are precisely those obtained by prefixing an existential quantifier to a computable predicate. We have

Theorem 1.3.8 *Let $(x_1, x_2, \ldots, x_n)$ be in Ω^n. Consider the predicate $R(x_1, x_2, \ldots, x_n)$ such that*

$$R(x_1, x_2, \ldots, x_n) \iff (\exists y \in \Omega)\,(P(y, x_1, x_2, \ldots, x_n))\,,$$

where $P(y, x_1, x_2, \ldots, x_n)$ is a computable predicate. Then, $R(x_1, x_2, \ldots, x_n)$ is semicomputable.

Proof: The set

$$\{(x_1, x_2, \ldots, x_n) \mid R(x_1, x_2, \ldots, x_n)\}$$

is the domain of the computable function

$$\min_y [C'_R(y, x_1, x_2, \ldots, x_n) = 0], \tag{1.2}$$

where C'_R was defined in the proof of Theorem 1.3.7. ■

Theorem 1.3.9 *Let $(y, x_1, x_2, \ldots, x_n)$ be in Ω^{n+1}. Let $R(x_1, x_2, \ldots, x_n)$ be a semicomputable predicate. Then, there exists some computable predicate $P(y, x_1, x_2, \ldots, x_n)$ such that*

$$R(x_1, x_2, \ldots, x_n) \iff (\exists y \in \Omega)\,(P(y, x_1, x_2, \ldots, x_n))\,.$$

Proof: By definition, there exists a computable function $f(x_1, x_2, \ldots, x_n)$ whose domain is the set

$$\{(x_1, x_2, \ldots, x_n) \mid R(x_1, x_2, \ldots, x_n)\}.$$

Now, by Theorem 1.3.6, there is some z_0 in Ω such that

$$f(x_1, x_2, \ldots, x_n) = U\big(\min_y [T_n(z_0, x_1, x_2, \ldots, x_n, y)]\big),$$

and, therefore, the domain of $f(x_1, x_2, \ldots, x_n)$ is the set

$$\{(x_1, x_2, \ldots, x_n) \mid (\exists y \in \Omega)(T_n(z_0, x_1, x_2, \ldots, x_n, y))\}.$$

We conclude that

$$R(x_1, x_2, \ldots, x_n) \iff (\exists y \in \Omega)(T_n(z_0, x_1, x_2, \ldots, x_n, y)).$$

Take $P(y, x_1, x_2, \ldots, x_n) = T_n(z_0, x_1, x_2, \ldots, x_n, y)$. The result follows from Theorem 1.3.4. ■

We may now establish the relationship between computability and semicomputability more clearly.

Theorem 1.3.10 *Let $(x_1, x_2, \ldots, x_n)$ be in Ω^n. Then, the predicate $R(x_1, x_2, \ldots, x_n)$ is computable if and only if $R(x_1, x_2, \ldots, x_n)$ and $\neg R(x_1, x_2, \ldots, x_n)$ are semicomputable.*

Proof: If $R(x_1, x_2, \ldots, x_n)$ is computable, then so is $\neg R(x_1, x_2, \ldots, x_n)$ by considering their characteristic functions. Therefore, both

$$R(x_1, x_2, \ldots, x_n) \quad \text{and} \quad \neg R(x_1, x_2, \ldots, x_n)$$

are semicomputable by Theorem 1.3.7.

Now, suppose that

$$R(x_1, x_2, \ldots, x_n) \quad \text{and} \quad \neg R(x_1, x_2, \ldots, x_n)$$

are semicomputable. Then, for y in Ω, there exist computable predicates

$$P(y, x_1, x_2, \ldots, x_n) \quad \text{and} \quad Q(y, x_1, x_2, \ldots, x_n)$$

such that

$$\begin{aligned} R(x_1, x_2, \ldots, x_n) &\iff (\exists y \in \Omega)\,(P(y, x_1, x_2, \ldots, x_n))\,, \\ \neg R(x_1, x_2, \ldots, x_n) &\iff (\exists y \in \Omega)\,(Q(y, x_1, x_2, \ldots, x_n))\,. \end{aligned}$$

For any given $(x_1, x_2, \ldots, x_n)$, either

$$R(x_1, x_2, \ldots, x_n) \quad \text{or} \quad \neg R(x_1, x_2, \ldots, x_n)$$

must be true. Therefore, the domain of the function

$$h(x_1, x_2, \ldots, x_n) = \min_y \left[P(y, x_1, x_2, \ldots, x_n) \vee Q(y, x_1, x_2, \ldots, x_n)\right]$$

is Ω^n. As the predicate

$$P(y, x_1, x_2, \ldots, x_n) \vee Q(y, x_1, x_2, \ldots, x_n)$$

is computable by considering its characteristic function, we conclude that

$$h(x_1, x_2, \ldots, x_n)$$

is computable as well by Theorem 1.3.5.

Now, notice that

$$R(x_1, x_2, \ldots, x_n) \iff P(h(x_1, x_2, \ldots, x_n), x_1, x_2, \ldots, x_n).$$

Therefore, $R(x_1, x_2, \ldots, x_n)$ is computable. ■

We will now find a semicomputable predicate that is not computable.

Theorem 1.3.11 *Let x be in Ω. The predicate*

$$\mathfrak{D}(x) \iff (\exists y \in \Omega)(T(x, x, y))$$

is semicomputable, but not computable.

Proof: $\mathfrak{D}(x)$ is semicomputable by Theorems 1.3.4 and 1.3.8. Let us suppose that $\neg\mathfrak{D}(x)$ is semicomputable. Then, there exists some z_0 in Ω such that

$$\neg(\exists y \in \Omega)(T(x, x, y)) \iff (\exists y \in \Omega)(T(z_0, x, y)).$$

Let $x = z_0$. We get

$$\neg(\exists y \in \Omega)(T(z_0, z_0, y)) \iff (\exists y \in \Omega)(T(z_0, z_0, y)).$$

This is a contradiction. We conclude that $\neg\mathfrak{D}(x)$ is not semicomputable, so that $\mathfrak{D}(x)$ is not computable. ■

The reader will notice that this proof employed a variation of Cantor's diagonal argument.

1.3.3 The halting problem

We may now show that there exist undecidable problems.

Definition 1.3.11 Let $(x_1, x_2, \ldots, x_n)$ be in Ω^n. The **decision problem** associated with the predicate $P(x_1, x_2, \ldots, x_n)$ is the problem of determining whether $P(x_1, x_2, \ldots, x_n)$ is true.

Definition 1.3.12 We say that a decision problem is **undecidable** if the associated predicate is not computable.

Using this language, we say that the decision problem for the predicate $\mathfrak{D}(x)$ is undecidable. We will use this result to show that the halting problem is undecidable.

Definition 1.3.13 Let Z be a Turing machine. The **halting problem** for Z is the decision problem of determining, for a given instantaneous description α_1 of Z, whether there exists a computation $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ of Z. We write the associated predicate $\mathfrak{H}_Z(x)$ to mean that x is the Gödel number of such an α_1.

Theorem 1.3.12 *There exists a Turing machine whose halting problem is undecidable.*

Proof: Let Z_0 be a Turing machine such that

$$\Psi_{Z_0}(x) = \min_y [T(x, x, y)].$$

Then, x belongs to the domain of $\Psi_{Z_0}(x)$ if and only if

$$(\exists y \in \Omega)(T(x, x, y)). \tag{1.3}$$

However, we also have that x belongs to the domain of $\Psi_{Z_0}(x)$ if and only if

$$\mathfrak{H}_{Z_0}(\mathrm{gn}\,(q_1\bar{x})).$$

Now, the function gn is computable,[5] so that if $\mathfrak{H}_{Z_0}(x)$ were computable, so would be (1.3). However, this is not the case by Theorem 1.3.11, so that $\mathfrak{H}_{Z_0}(x)$ must be not computable. ■

The undecidability of the halting problem is an important result for computer scientists and programmers. It implies that there cannot be a general algorithm to determine if a computer program will do what it is supposed to do.

We now turn to the theory of semigroups.

[5] See Davis [9].

Chapter 2

Semigroups

2.1 Semigroup theory

Semigroups are very simple to describe, and are sometimes considered a generalization of groups [12, 15]. However, they are so numerous when compared to groups that it is often better to think of them as completely independent algebraic objects [13].

Definition 2.1.1 We say that (S, μ) is a **semigroup** if S is a set on which an associative binary operation μ is defined.

That is, we have a function $\mu : S \times S \to S$ such that, for all x, y, and z in S, we have

$$\mu(x, \mu(y, z)) = \mu(\mu(x, y), z).$$

Whenever no confusion with arithmetic multiplication may arise, we shall write $x \cdot y$ or xy for $\mu(x, y)$, and speak in terms of products. Also, we may write S to refer to the semigroup (S, μ) if the associated binary operation is clear.

Definition 2.1.2 Let S be a semigroup. If S contains an identity element, we define $S^1 = S$. If S does not contain such an element, we define $S^1 = S \cup \{1\}$, where 1 is not in S, and 1 acts as the identity element, i.e., for any x and y in S,

$$1x = x1 = x,$$

and xy is the same in S^1 as in S.

Clearly, if S is a semigroup, then S^1 is a semigroup with identity, which is also known as a *monoid*. Thus every semigroup can be extended to a monoid in a natural way.

2.1.1 Homomorphisms and congruences

Definition 2.1.3 Let S and T be semigroups. We say a function $\varphi : S \to T$ is a **(semigroup) homomorphism** if

$$\varphi(xy) = \varphi(x)\varphi(y)$$

for all x and y in S.

Semigroup homomorphisms share a number of properties with homomorphisms of groups. Analogously to subgroups, we define *subsemigroups*.

Definition 2.1.4 Let T be a semigroup and S be a subset of T as a set. Then, we call S a **subsemigroup** of T if S is closed under the semigroup operation, i.e., if x and y are elements of S, then so is xy.

Clearly, we have

Theorem 2.1.1 *Let T be a semigroup. Let S and V be subsemigroups of T. Then, $S \cap V$ is also a subsemigroup of T.*

If we are given a semigroup, we may ask which is its smallest subsemigroup such that it contains a given set. Thus, we define the subsemigroup *generated* by a set as follows.

Definition 2.1.5 Let S be a semigroup and X be a set such that $X \subset S$. Then, we define the **subsemigroup X^* generated by** X as the intersection of all subsemigroups of S that contain X.

Theorem 2.1.2 *Let S be a semigroup and X be a set such that $X \subset S$. Then, X^* is the set of all products of one or more elements of X.*

Proof: Let V be the set of all products of one or more elements of X. Then, V is clearly closed under the semigroup operation, so that V is a subsemigroup of S which contains X. Therefore, we have $X^* \subset V$.

Now, let T be any subsemigroup of S containing X. By definition, T is closed under the semigroup operation, so that by induction T must contain all products of one or more elements of X. Therefore, we have that $T \supset V$, and this implies that $X^* \supset V$.

We conclude that $X^* = V$. ■

Definition 2.1.6 Let S be a semigroup and X be a set such that $X \subset S$. If $X^* = S$, we say that X **generates** S and call the elements of X the **generators** of S.

We will be working with *equivalence relations* and *equivalence classes*[1] within semigroups. We introduce the following notation.

Definition 2.1.7 Let $\mathscr{C}$ be an equivalence relation on a set X. We define the **quotient set of** X **by** $\mathscr{C}$ as the set of all equivalence classes of elements of X with respect to $\mathscr{C}$. We write this set as

$$X/\mathscr{C}.$$

For a given x in X, we write

$$[x]_{\mathscr{C}},$$

or simply $[x]$ when there is no ambiguity, for the equivalence class of x with respect to $\mathscr{C}$.

The following theorem is taken from Grillet [13], and we state it without proof.

Theorem 2.1.3 *Let $\mathscr{C}$ be an equivalence relation[2] on a semigroup S. Then, the following are equivalent:*

1. *There is a semigroup operation on $S/\mathscr{C}$ such that the projection map $p\colon S \to S/\mathscr{C}$ defined by $p(s) = [s]_{\mathscr{C}}$ is a homomorphism.*

2. *$\mathscr{C}$ is compatible with the semigroup operation, i.e., if $a\ \mathscr{C}\ b$ and $c\ \mathscr{C}\ d$, then $ac\ \mathscr{C}\ bd$ for any a, b, c, and d in S.*

[1] See Judson [16] for the definitions.
[2] Here we are thinking of $\mathscr{C}$ as defined on S as a set.

3. $\mathscr{C}$ *admits "multiplication from the left," i.e.,*

$$a \mathscr{C} b \implies xa \mathscr{C} xb,$$

and "multiplication from the right," i.e.,

$$a \mathscr{C} b \implies ax \mathscr{C} bx,$$

for any a, b, and x in S.

In 1, the operation on $S/\mathscr{C}$ is unique, and we have that $[a][b] = [ab]$ for any a and b in S. Furthermore, $[ab]$ is the only class in $S/\mathscr{C}$ such that it contains the product of the classes $[a]$ and $[b]$ as subsets of S.

Definition 2.1.8 Let $\mathscr{C}$ be an equivalence relation on a semigroup S. We say that $\mathscr{C}$ is a **congruence** in S if it satisfies any of the equivalent conditions in Theorem 2.1.3. We call the resulting semigroup $S/\mathscr{C}$ the **quotient (semigroup) of S by $\mathscr{C}$**.

Now, clearly we have that

Theorem 2.1.4 *The intersection of two congruences in S is also a congruence in S.*

We may wonder whether we can extend any binary relation to a congruence, and which is the smallest such congruence. We have

Definition 2.1.9 Let S be a semigroup. Let $\mathscr{A}$ be any subset of $S \times S$, i.e., a binary relation on S. We define the **congruence generated by** $\mathscr{A}$ as the intersection $\mathscr{T}$ of all the congruences in S that contain $\mathscr{A}$.

We can construct $\mathscr{T}$ in Definition 2.1.9 explicitly: we first extend $\mathscr{A}$ to a symmetric relation $\mathscr{B}$ such that it admits the semigroup operation, and then we construct its *transitive closure*.

Definition 2.1.10 Let $\mathscr{B}$ be a binary relation on a set X. The **transitive closure of $\mathscr{B}$** is the binary relation $\mathscr{T}$ defined such that, for any a and b in X,

$$a \mathscr{T} b$$

if and only if there exists a sequence $(x_1, x_2, \ldots, x_r) \in X^r$ such that $r > 0$, $a = x_1$, $b = x_r$, and

$$x_i \mathrel{\mathscr{B}} x_{i+1}$$

for all integers i, $1 \leq i < r$.

Theorem 2.1.5 *Let S be a semigroup and $\mathscr{A}$ be any binary relation on S. The congruence $\mathscr{T}$ generated by $\mathscr{A}$ is the transitive closure of the binary relation $\mathscr{B}$ defined such that, for any a and b in S,*

$$a \mathrel{\mathscr{B}} b$$

if and only if

$$a = uxv \quad \text{and} \quad b = uyv,$$

where u and v are in S^1, x and y are in S, and

$$x \mathrel{\mathscr{A}} y \quad \text{or} \quad y \mathrel{\mathscr{A}} x.$$

Proof: First, notice that $\mathscr{B}$ contains $\mathscr{A}$ by letting $u = v = 1$. Second, notice that $\mathscr{B}$ is symmetric, and admits both multiplication on the left and on the right. Therefore, $\mathscr{T}$ inherits symmetry and also admits multiplication on the left and on the right.

Now, letting $r = 2$, we get that $\mathscr{T}$ contains $\mathscr{B}$, and letting $r = 1$, we get that $\mathscr{T}$ is reflexive. Also, $\mathscr{T}$ is clearly transitive, so we conclude that $\mathscr{T}$ is a congruence which contains $\mathscr{A}$.

Conversely, let $\mathscr{C}$ be any congruence that contains $\mathscr{A}$. Let x and y be in S. If

$$x \mathrel{\mathscr{A}} y \quad \text{or} \quad y \mathrel{\mathscr{A}} x,$$

then $x \mathrel{\mathscr{C}} y$, and, for any u and v in S^1, we have

$$uxv \mathrel{\mathscr{C}} uyv.$$

This shows $\mathscr{C}$ contains $\mathscr{B}$. Now, since $\mathscr{C}$ is reflexive and transitive, it contains $\mathscr{T}$, and, therefore,

$\mathscr{T}$ is the smallest congruence that contains $\mathscr{A}$. ■

2.1.2 Free semigroups

We now introduce the concept of a *free semigroup* which will be essential to our statement of the word problem for semigroups.

Let X be any set. Then, we may construct a semigroup F_X such that $X \subset F_X$, and where any element of F_X can be written uniquely as a product of elements of X. All elements of F_X are finite sequences $(x_1, x_2, \dots, x_n)$ of elements of X, and the binary operation associated to F_X is *concatenation*.

Definition 2.1.11 Let $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_m)$ be two finite sequences. The binary operation of **concatenation** of $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_m)$ is defined such that

$$(x_1, x_2, \dots, x_n)(y_1, y_2, \dots, y_m) = (x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m).$$

Clearly, concatenation is associative. Furthermore, notice that the empty sequence $()$ is the identity element for the operation of concatenation.

Now, every sequence can be written uniquely as a concatenation of one-element sequences:

$$(x_1, x_2, \dots, x_n) = (x_1)(x_2)\cdots(x_n).$$

We identify every element x in X with the one-element sequence (x), so that $X \subset F_X$ and every element of F_X can be written uniquely as a concatenation of elements of X. We will usually write sequences in F_X as *words* or *strings*, like $x_1x_2\cdots x_n$ for the sequence $(x_1, x_2, \dots, x_n)$. Then, we will call X the *alphabet* of those words.

Definition 2.1.12 The **free semigroup** F_X on the set X is the semigroup of all finite nonempty sequences of elements of X together with the operation of concatenation, as above.

We also define

Definition 2.1.13 The **free monoid** F_X^1 on the set X is the semigroup of all finite (possibly empty) sequences of elements of X together with the operation of concatenation, as above.

This is compatible with our notation above.

Theorem 2.1.6 *Let X be any set and S be a semigroup. Then, any function $f\colon X \to S$ extends uniquely to a homomorphism $\varphi\colon F_X \to S$. Furthermore, the image of φ is the subsemigroup of S generated by*

$$f(X) = \{f(x) \mid x \in X\}.$$

If S is generated by $f(X)$, then φ is onto.

Proof: Notice that if $\varphi\colon F_X \to S$ is a homomorphism that extends f, then we necessarily have that

$$\varphi(x_1x_2\cdots x_n) = \varphi(x_1)\varphi(x_2)\cdots\varphi(x_n) = f(x_1)f(x_2)\cdots f(x_n),$$

for any $x_1x_2\cdots x_n$ in F_X. Conversely, the function $\varphi\colon F_X \to S$ such that

$$\varphi(x_1x_2\cdots x_n) = f(x_1)f(x_2)\cdots f(x_n)$$

for any $x_1x_2\cdots x_n$ in F_X is a semigroup homomorphism that extends f.

Finally, by Theorem 2.1.2, the image of φ is the semigroup of S generated by $f(X)$. ■

This shows

Theorem 2.1.7 *Every semigroup S is the image of a homomorphism $\varphi\colon F_X \to S$ for some set X.*

Theorem 2.1.7 above is called the *universal mapping property* of free semigroups.

2.1.3 Presentation of a semigroup

Congruences and free semigroups allow us to describe semigroups in terms of *presentations* as follows.

Definition 2.1.14 Let X be any set. We define a **relation between the elements of** X (or simply, a **relation**) as an ordered pair of elements of F_X. We will usually write a relation (u, v) as $u \doteq v$.

Definition 2.1.15 Let X be any set and S be a semigroup. Consider a function $f\colon X \to S$. Let the function $\varphi\colon F_X \to S$ be the unique homomorphism that extends f. We say that $u \doteq v$ **holds in** S **(via** f**)** if $\varphi(u) = \varphi(v)$ is true in S.

If X is a subset of a semigroup S, and f is the inclusion mapping defined by $f(x) = x$, then φ in Definition 2.1.15 takes a sequence of elements of X from F_X to the product of those same elements in S. Therefore, a relation $u \doteq v$ holds in S if and only if the corresponding products of elements of X are equal in S.

Definition 2.1.16 Let X be a set and R be a set of relations between the elements of X, i.e., R is a binary relation on F_X. We write

$$\langle X \mid R \rangle$$

to mean the quotient semigroup $F_X/\mathscr{C}$, where $\mathscr{C}$ is the congruence generated by R. We say that $\langle X \mid R \rangle$ is the **semigroup generated by** X **subject to** R, and call the elements of R the **defining relations** of $\langle X \mid R \rangle$.

For any $\langle X \mid R \rangle$ as in Definition 2.1.16, we can define a function $\imath \colon X \to \langle X \mid R \rangle$ such that[3] $\imath(x) = [(x)]_{\mathscr{C}}$. The homomorphism this function extends to is precisely the projection map $p \colon F_X \to \langle X \mid R \rangle$ such that

$$\begin{aligned} p(x_1, x_2, \ldots, x_n) &= \left[(x_1, x_2, \ldots, x_n)\right]_{\mathscr{C}} \\ &= \left[(x_1)\right]_{\mathscr{C}} \left[(x_2)\right]_{\mathscr{C}} \cdots \left[(x_n)\right]_{\mathscr{C}} \\ &= \imath(x_1)\imath(x_2)\cdots\imath(x_n). \end{aligned}$$

This shows

Theorem 2.1.8 *Let X be a set and R be a set of relations between the elements of X. Then, $\langle X \mid R \rangle$ is generated by*

$$\imath(X) = \{\imath(x) \mid x \in X\},$$

where $\imath$ is as above, and every relation $u \doteq v$ in R holds in $\langle X \mid R \rangle$ via $\imath$.

It should be noted that X need not be a subset of $\langle X \mid R \rangle$ as $\imath$ need not be one-to-one. Furthermore, $\langle X \mid R \rangle$ is not the only semigroup generated by $\imath(X)$ where every relation in R holds. However, it can be shown that there is a sense in which $\langle X \mid R \rangle$ is the largest such semigroup.[4]

[3] Here, we write (x) to refer explicitly to x as an element of F_X.

[4] See Grillet [13].

Definition 2.1.17 Let S be a semigroup. A **presentation** of S consists of a set X, a set R of relations between the elements of X, and bijective homomorphism (or *isomorphism*) between S and $\langle X \mid R \rangle$. If X is finite, we say that S is **finitely generated**. If R is finite, we say that S is **finitely related**. If both X and R are finite, we say that S is **finitely presented**.

Now, by Theorem 2.1.7, every semigroup S has a presentation, in which X can be any subset of S which generates S, and R be any binary relation such that it generates the congruence induced[5] by $\varphi : F_X \to S$.

2.1.4 Semi-Thue and Thue systems

We may also describe finitely presented semigroups in terms of *string rewriting*.

Consider the finite sets

$$X = \{a_1, a_2, \dots, a_n\},$$

and

$$R = \{(u_i, v_i) \mid i = 1, 2, \dots, m\},$$

where R is a set of relations between the elements of X. We may think of the elements of R as *rewrite rules* as follows.

Definition 2.1.18 A **rewrite rule** is any element (u, v) of $F_X \times F_X$ such that it can be applied to words on X of the form $z_1 u z_2$, where z_1 and z_2 are in F_X^1. The result of applying the rule (u, v) to

$$z_1 u z_2$$

is

$$z_1 v z_2,$$

and we write

$$z_1 u z_2 \to z_1 v z_2.$$

[5] See Grillet [13].

Here, we repurpose our notation for basic moves from Definition 1.1.5. This is not a coincidence, as we will see later.

Definition 2.1.19 A system S, consisting of F_X together with a finite set of rewrite rules R as above, is called a **semi-Thue system**. If the rewrite rules are symmetric, we call it a **Thue system**.

Definition 2.1.20 A string rewrite of the form

$$z_1 u_i z_2 \to z_1 v_i z_2,$$

where z_1 and z_2 are in F_X^1, is called a **forward application** of the rewrite rule (u_i, v_i) in R. Similarly, a string rewrite of the form

$$z_1 v_i z_2 \to z_1 u_i z_2,$$

is called a **backward application** of the rewrite rule (u_i, v_i) in R.

Definition 2.1.21 Let S be a semi-Thue system associated with the sets F_X and R, as above. We define a **proof** in S as a sequence of string rewrites of the form

$$w_1 \to w_2 \to \cdots \to w_r,$$

where each string rewrite is a single (backward or forward) application of a rewrite rule in R. A proof is said to be **reversal-free** or **without reversals** if it does not contain two successive string rewrites of the forms

$$z_1 u_i z_2 \to z_1 v_i z_2 \to z_1 u_i z_2 \quad \text{or} \quad z_1 v_i z_2 \to z_1 u_i z_2 \to z_1 v_i z_2,$$

the second of which undoes the first, where z_1 and z_2 are in F_X^1, and (u_i, v_i) is in R.

We define two words to be *congruent* in a Thue system S if and only if there exists a proof in S beginning with one and ending with the other. Then, we may identify S, consisting of F_X

and R as above, with the semigroup $\langle X \mid R \rangle$. To do this, consider the congruence $\approx$ generated by R. Recall that $\approx$ is first of all an equivalence relation. In the proof of Theorem 2.1.5, we used Theorem 2.1.3 to show that $\approx$ was a congruence in addition to being an equivalence relation. We now prove directly that $\approx$ is compatible with concatenation, and thus explicitly construct $\langle X \mid R \rangle$.

Definition 2.1.22 Let A and B be words on X, i.e., elements of F_X. Then, we write $A \sim B$ if there exists a rewrite rule in R such that $A \to B$ or $B \to A$.

Here, $\sim$ takes the role of $\mathcal{B}$ in Theorem 2.1.5. That is, $\approx$ is the transitive closure of $\sim$. Furthermore, notice that $\approx$ is identical to congruence in terms of proofs.

Theorem 2.1.9 *Let A, B, C, and D be words on X. If $A \sim B$ and $C \sim D$, then $AC \approx BD$.*

Proof: Suppose (u_i, v_i) and $\left(u_j, v_j\right)$ are in R. Without loss of generality, we can write

$$A = z_1 u_i z_2,$$

$$B = z_1 v_i z_2,$$

$$C = z_3 u_j z_4,$$

$$\text{and} \quad D = z_3 v_j z_4,$$

for some z_1, z_2, z_3, and z_4 on F_X^1. Now, we have $AC \sim AD \sim BD$. ∎

Theorem 2.1.10 *Let A, B, C, and D be words on X. If $A \approx B$ and $C \sim D$, then $AC \approx BD$.*

Proof: Let $A = A_1 \sim A_2 \sim \cdots \sim A_r = B$. Then, by Theorem 2.1.9,

$$AC = A_1 C \approx A_2 D \sim A_3 D \sim \cdots \sim A_r D = BD,$$

so that $AC \approx BD$. ∎

Theorem 2.1.11 *Let A, B, C, and D be words on X. If $A \approx B$ and $C \approx D$, then $AC \approx BD$.*

Proof: Let $C = C_1 \sim C_2 \sim \cdots \sim C_s = D$. Then, by Theorem 2.1.10,

$$AC = AC_1 \approx BC_2 \sim BC_3 \sim \cdots \sim BC_s = BD,$$

so that $AC \approx BD$. ■

Definition 2.1.23 Let A and B be words on X. Then, we shall write $[A][B]$ for the class $[AB]$ (with respect to the equivalence relation $\approx$), and speak of **multiplication on equivalence classes**.

Theorem 2.1.12 *Multiplication on equivalence classes is a well-defined binary operation.*

Proof: This follows at once from Theorem 2.1.11. We have that if $[A] = [B]$ and $[C] = [D]$, then $[AC] = [BD]$. ■

This shows that $\approx$ is compatible with concatenation, and, therefore, that it is a congruence. Furthermore, we have

Theorem 2.1.13 *Multiplication on equivalence classes is an associative binary operation.*

Proof: We have

$$\begin{aligned}
[A]([B][C]) &= [A][BC] \\
&= [A(BC)] \\
&= [(AB)C] \\
&= [AB][C] \\
&= ([A][B])[C].
\end{aligned}$$

■

Thus, we have explicitly described the semigroup

$$\langle X \mid R \rangle = F_X/\approx,$$

whose elements are the equivalence classes $[A]_\approx$ for A in F_X, and whose semigroup operation is multiplication on equivalence classes.

From this point onward, we will use both defining relations and rewrite rules interchangeably when discussing finitely generated semigroups.

It is important to understand that even when given the sets X and R, we may still be none the wiser about the structure of the semigroup $\langle X \mid R \rangle$, or even whether it has a finite or infinite number of elements.

2.2 The word problem for semigroups

2.2.1 Statement

Now, we may finally pose the *word problem for semigroups*.

Definition 2.2.1 If we are given finite sets X and R, where R is a set of relations on X, then the **word problem for the (finitely presented) semigroup** $\langle X \mid R \rangle$ is to determine, of two words A and B on F_X, whether or not $A \approx B$, where $\approx$ is the congruence generated by R.

2.2.2 Method

The following discussion is based on Tarski [28].

Two general methods have emerged for proving that decision problems are undecidable since the discovery of Gödel's Incompleteness Theorems.

The first method, which follows Gödel's approach (outlined in Chapter 1), requires extensive work to develop a sufficient number-theoretical framework within the theory being studied.

The second method is indirect, proving undecidability by reducing known undecidable problems to our target problem. This reduction can be accomplished in two ways. First, we can show that our target theory can be obtained by removing finitely many axioms from a known undecidable theory. Second, we can show that a known undecidable theory can be interpreted within our target theory.

We will use the interpretation technique to prove the word problem for semigroups is undecidable by showing that Turing machines can be "simulated" within the theory of finitely presented semigroups. In this way, we will find that the word problem can be reduced to the halting problem.

2.2.3 Simulating Turing machines

Definition 2.2.2 Let Z be a Turing machine with alphabet

$$\{S_0, S_1, \ldots, S_m\}$$

and internal states

$$\{q_1, q_2, \dots, q_n\}.$$

We define the finitely presented semigroup

$$\rho(Z) = \langle \{h, q, q_0\} \cup \{S_0, S_1, \dots, S_m\} \cup \{q_1, q_2, \dots, q_n\} \mid R_Z \rangle,$$

such that:

1. For each quadruple of Z of the form

   $$q_i \; S_j \; S_k \; q_l,$$

   R_Z contains

   $$q_i S_j \doteq q_l S_k.$$

2. For each quadruple of Z of the form

   $$q_i \; S_j \; R \; q_l,$$

   and each S_k in the alphabet of Z, R_Z contains

   $$q_i S_j S_k \doteq S_j q_l S_k \quad \text{and} \quad q_i S_j h \doteq S_j q_l S_0 h.$$

3. For each quadruple of Z of the form

   $$q_i \; S_j \; L \; q_l,$$

   and each S_k in the alphabet of Z, R_Z contains

   $$S_k q_i S_j \doteq q_l S_k S_j \quad \text{and} \quad h q_i S_j \doteq h q_l S_0 S_j.$$

4. For each internal configuration q_i of Z, and each S_j in the alphabet of Z for which no

quadruple of Z begins with the combination

$$q_i\ S_j,$$

R_Z contains

$$q_i S_j \doteq q_0 S_j.$$

5. For each S_i in the alphabet, R_Z contains

$$q_0 S_i \doteq q_0,$$
$$S_i q_0 h \doteq q_0 h,$$
$$\text{and} \quad h q_0 h \doteq q.$$

Finally, we write $\approx$ for the congruence generated by R_Z.

We have introduced new symbols in addition to those used to describe Turing machines in Chapter 1.

The first one is the symbol h. Intuitively, this symbol marks the ends of the extendable tape of the Turing machine and allows us to translate the "pasting" of a new square onto the tape to the language of defining relations. With this consideration, 1, 2, and 3 above correspond to the normal functioning of the Turing machine Z.

The second symbol is q_0, which we will call the *halting state*. Recall that a Turing machine Z halts whenever no more basic moves are possible. This is equivalent to saying that an instantaneous description α is terminal with respect to Z if and only if there exists no quadruple in Z such that it begins with $q_i\ S_j$, where q_i is the internal state in α, and S_j is the symbol to its right in α. Then, 4 above corresponds to adding new quadruples

$$q_i\ S_j\ S_j\ q_0$$

to Z for all the possible "missing" quadruples. This new machine Z' has the property that the

functions

$$\Psi_Z^{(n)}(x_1, x_2, \ldots, x_n) \quad \text{and} \quad \Psi_{Z'}^{(n)}(x_1, x_2, \ldots, x_n)$$

have the same domain. That is, for any $(x_1, x_2, \ldots, x_n)$ in Ω^n, there is a computation in Z starting with $q_1\overline{(x_1, x_2, \ldots, x_n)}$ if and only if there is a computation in Z' starting with $q_1\overline{(x_1, x_2, \ldots, x_n)}$.

The last symbol is q. In terms of rewrite rules, forward applications of the defining relations in 5 correspond to taking words of the form huq_0vh, where u and v are words on the alphabet of Z, and transforming them in the following way:

- First, we delete all symbols in the alphabet of Z to the right of q_0, obtaining huq_0h.
- Second, we delete all symbols in the alphabet of Z to the left of q_0, obtaining hq_0h.
- Third, we replace hq_0h with q.

Notice that, whenever

$$\mathrm{Res}_{Z'}\left(q_1\overline{(x_1, x_2, \ldots, x_n)}\right)$$

is defined for $(x_1, x_2, \ldots, x_n)$ in Ω^n, it is of the form huq_0vh. In this sense, 5 intuitively corresponds to "clearing" the tape of Z' at the end of a successful computation.

We summarize this discussion in the following theorem.

Theorem 2.2.1 *Let Z be a Turing machine. If there is a computation in Z starting with*

$$q_1\overline{(x_1, x_2, \ldots, x_n)},$$

then there exists a proof in $\rho(Z)$ of the form

$$hq_1\overline{(x_1, x_2, \ldots, x_n)}h \to \cdots \to hq_0h \to q,$$

where each step is a forward application of a defining relation. Thus, in $\rho(Z)$,

$$hq_1\overline{(x_1, x_2, \ldots, x_n)}h \approx q.$$

To show the word problem for semigroups is reducible to the halting problem, we must show the converse of Theorem 2.2.1.

Definition 2.2.3 Let Z be a Turing machine. We say a word[6] of $\rho(Z)$ is h**-special** if it has the form

$$huq'vh,$$

where u and v are (possibly empty) words on the alphabet of Z and

$$q' \in \{q\} \cup \{q_0, q_1, \ldots, q_n\}.$$

Theorem 2.2.2 *Let w_1 and w_2 be words of $\rho(Z)$ such that neither of them is the word q. If*

$$w_1 \to w_2$$

is an application of a relation in R_Z, then w_1 is h-special if and only if w_2 is h-special.

Proof: Notice that only applications of the relation $hq_0h \doteq q$ create or destroy h. The result follows. ■

Theorem 2.2.3 *Let*

$$w = huq'vh$$

be an h-special word of $\rho(Z)$ as in Definition 2.2.3. Then, at most one of the relations of R_Z has a forward application to w.

Proof: First, if $q' = q$, then clearly there is no relation in R_Z such that it has a forward application to w. Second, if $q' \neq q$ and $q' \neq q_0$, then the result follows from Theorem 1.1.1 applied to Z'. Finally, suppose that $q' = q_0$. We have:

- If u and v are empty, then clearly the only relation in R_Z that has a forward application to w is $hq_0h \doteq q$.

[6]That is, an element of an equivalence class in the quotient semigroup $\rho(Z)$.

- If u is nonempty and v is empty, then only one of the relations $S_i q_0 h \doteq q_0 h$ in R_Z has a forward application to w.
- If v is nonempty, then only one of the relations $q_0 S_i \doteq q_0$ in R_Z has a forward application to w.

This completes the proof. ■

Theorem 2.2.4 *Let*

$$w = huq'vh$$

be an h-special word of $\rho(Z)$ as in Definition 2.2.3. If there exists a proof

$$w = w_1 \to w_2 \to \cdots \to w_n = q$$

in $\rho(Z)$, then either this proof is not reversal-free or all the applications of relations in the proof are forward. In particular, the shortest such proof consists entirely of forward applications.

Proof: First, notice that $w_{n-1} \to w_n = q$ must be a forward application of $hq_0h \doteq q$, so that

$$w_{n-1} = hq_0h.$$

Let us suppose that not all of the applications of relations in the proof are forward. Then, let

$$w_{j-1} \to w_j$$

be the last backward application. Notice that $j \neq n$. We have that

$$w_j \to w_{j+1} \to \cdots \to w_n$$

consists entirely of forward applications of relations.

Now, we have that $w_{n-1} = hq_0h$ is h-special and that none of the $w_j, w_{j+1}, \ldots, w_{n-1}$ can be q, as only a backward application applies to q. Therefore, by Theorem 2.2.2, all of the $w_j, w_{j+1}, \ldots, w_{n-1}$ are h-special as well.

Consider

$$w_{j-1} \to w_j \to w_{j+1}. \tag{2.1}$$

We have that if $w_{j-1} \to w_j$ is a backward application, then $w_j \to w_{j-1}$ is a forward application. Therefore,

$$w_j \to w_{j-1} \quad \text{and} \quad w_j \to w_{j+1}$$

are both forward applications of a relation. By Theorem 2.2.3,

$$w_{j-1} = w_{j+1},$$

and (2.1) is a reversal, so that the proof

$$w = w_1 \to w_2 \to \cdots \to w_n = q$$

is not reversal-free.

As we can always reduce the length of a proof by removing reversals, the shortest such proof must be reversal-free. This completes the proof. ■

Now, since forward applications of relations in $\rho(Z)$ correspond exactly to the basic moves of Z' or to tape-clearing moves at the end, we conclude the following.

Theorem 2.2.5 *Let Z be a Turing machine. There is a computation of Z starting with*

$$q_1\overline{(x_1, x_2, \ldots, x_n)}$$

if and only if

$$hq_1\overline{(x_1, x_2, \ldots, x_n)}h \approx q$$

in $\rho(Z)$.

Finally, by Theorem 1.3.12, we obtain

Theorem 2.2.6 *There exists a finitely presented semigroup whose word problem is undecidable.*

Chapter 3

Some computable functions

In Chapter 2, we assumed that basic operations like addition, subtraction, and function composition are computable. Under Church's thesis, this assumption is natural since we can describe algorithms for these operations. Even without Church's thesis, we can prove their computability by constructing the appropriate Turing machines; explicit constructions for these operations can be found in Davis [9]. Indeed, for Church's thesis to be plausible, we should be able to construct Turing machines for all common mathematical operations, like division, taking square roots, exponentiation, etc.

However, constructing these machines would not constitute a proof of Church's thesis, which would necessarily lie outside the domain of mathematics. In this section, we have a simpler goal: representing two basic functions using Turing machines.

We have implemented a Python program[1] to simulate the behavior of Turing machines as defined in Chapter 1. This program was used to create the computations found in this chapter.

3.1 An adding machine

Let the function $f\colon \Omega^2 \to \Omega$ be defined such that

$$f(x, y) = x + y.$$

[1] See the code at `https://github.com/adricamilo/Turing.git`.

We build a Turing machine Z_1 such that

$$\Psi^{(2)}_{Z_1}(x, y) = f(x, y)$$

for all $(x, y) \in \Omega^2$.

Let Z_1 consist of the following quadruples:

$$q_1\ 1\ 0\ q_1,$$
$$q_1\ 0\ R\ q_2,$$
$$q_2\ 1\ R\ q_2,$$
$$q_2\ 0\ R\ q_3,$$
$$q_3\ 1\ 0\ q_3.$$

The purpose of Z_1 is to delete two symbols 1 from the tape. Consider an instantaneous description $\alpha_1 = q_1\overline{(x, y)} = q_1\overline{x}0\overline{y}$. With respect to Z_1, we have

$$\begin{aligned}
\alpha_1 &= q_1 11^x 011^y \\
&\to q_1 01^x 011^y \\
&\to 0q_2 1^x 011^y \\
&\to \cdots \\
&\to 01^x q_2 011^y \\
&\to 01^x 0q_3 11^y \\
&\to 01^x 0q_3 01^y,
\end{aligned}$$

which is terminal. Therefore, for (x, y) in Ω_2,

$$\begin{aligned}
\Psi^{(2)}_{Z_1}(x, y) &= \mathrm{Res}_{Z_1}\left(q_1\overline{(x, y)}\right) \\
&= \left\langle 01^x 0q_3 01^y \right\rangle \\
&= x + y.
\end{aligned}$$

We have proved

Theorem 3.1.1 *The function* $f\colon \Omega^2 \to \Omega$ *such that*

$$f(x, y) = x + y$$

is computable.

3.1.1 An example computation

We initialized an object `adding_machine` of the class `TuringMachine` with the quadruples above to produce the following computations.

```
>>> adding_machine.compute([4,5])
[(q₁111110111111, None),
 (q₁011110111111, [q₁ 1 0 q₁]),
 (0q₂11110111111, [q₁ 0 R q₂]),
 (01q₂1110111111, [q₂ 1 R q₂]),
 (011q₂110111111, [q₂ 1 R q₂]),
 (0111q₂10111111, [q₂ 1 R q₂]),
 (01111q₂0111111, [q₂ 1 R q₂]),
 (011110q₃111111, [q₂ 0 R q₃]),
 (011110q₃011111, [q₃ 1 0 q₃])]
```

On the left-hand side, we find the successive instantaneous descriptions of Z_1, starting with $q_1\overline{(4,5)}$. On the right-hand side, we find the quadruples used in each step. The last row contains the resultant of the computation.

```
>>> adding_machine.resultant([4,5])
011110q₃011111

>>> adding_machine.resultant([4,5]).count_ones()
9
```

As we see, this agrees with our results above.

3.2 A multiplier

Let the function $g\colon \Omega^2 \to \Omega$ be defined such that

$$g(x, y) = (x + 1)(y + 1).$$

Notice that, for (x, y) in Ω^2, we have

$$(x + 1)(y + 1) = \underbrace{(x + 1) + (x + 1) + \cdots + (x + 1)}_{y+1 \text{ times}}.$$

We construct a machine that generates $y + 1$ copies of the tape expression 1^{x+1}.

Let Z_2 consist of the quadruples in Appendix A. S_2 and S_3 work in an analogous way to counters in a conventional computer program. We claim that

$$\Psi^{(2)}_{Z_2}(x, y) = g(x, y)$$

for all $(x, y) \in \Omega^2$. See Davis [9] for a proof. We have

Theorem 3.2.1 *The function $g\colon \Omega^2 \to \Omega$ such that*

$$g(x, y) = (x + 1)(y + 1)$$

is computable.

3.2.1 An example computation

We initialized an object `multiplier` of the class `TuringMachine` with the quadruples of Z_2 to produce the following computation.

```
>>> multiplier.compute([1,1])
[(q₁11011, None),
 (q₁01011, [q₁ 1 0 q₁]),
 (0q₂1011, [q₁ 0 R q₂]),
 (0q₃S₂011, [q₂ 1 S₂ q₃]),
 (0S₂q₃011, [q₃ S₂ R q₃]),
 (0S₂0q₄11, [q₃ 0 R q₄]),
 (0S₂01q₃1, [q₄ 1 R q₃]),
```

```
(0S₂011q₃0, [q₃ 1 R q₃]),          (0S₂0S₃1q₇01, [q₇ 1 R q₇]),
(0S₂0110q₄0, [q₃ 0 R q₄]),         (0S₂0S₃10q₈1, [q₇ 0 R q₈]),
(0S₂011q₅00, [q₄ 0 L q₅]),         (0S₂0S₃101q₈0, [q₈ 1 R q₈]),
(0S₂01q₆100, [q₅ 0 L q₆]),         (0S₂0S₃101q₉1, [q₈ 0 1 q₉]),
(0S₂01q₆S₃00, [q₆ 1 S₃ q₆]),       (0S₂0S₃10q₉11, [q₉ 1 L q₉]),
(0S₂01S₃q₇00, [q₆ S₃ R q₇]),       (0S₂0S₃1q₉011, [q₉ 1 L q₉]),
(0S₂01S₃0q₈0, [q₇ 0 R q₈]),        (0S₂0S₃q₉1011, [q₉ 0 L q₉]),
(0S₂01S₃0q₉1, [q₈ 0 1 q₉]),        (0S₂0q₉S₃1011, [q₉ 1 L q₉]),
(0S₂01S₃q₉01, [q₉ 1 L q₉]),        (0S₂0q₅11011, [q₉ S₃ 1 q₅]),
(0S₂01q₉S₃01, [q₉ 0 L q₉]),        (0S₂q₆011011, [q₅ 1 L q₆]),
(0S₂01q₅101, [q₉ S₃ 1 q₅]),        (0S₂q₁₀011011, [q₆ 0 0 q₁₀]),
(0S₂0q₆1101, [q₅ 1 L q₆]),         (0q₁₀S₂011011, [q₁₀ 0 L q₁₀]),
(0S₂0q₆S₃101, [q₆ 1 S₃ q₆]),       (0q₁0011011, [q₁₀ S₂ 0 q₁]),
(0S₂0S₃q₇101, [q₆ S₃ R q₇]),       (00q₂011011, [q₁ 0 R q₂])]
```

```
>>> multiplier.resultant([1,1]).count_ones()
4
```

As we can see, the computations with respect to Z_2 are much longer than those with respect to Z_1. However, we do obtain $(1 + 1)(1 + 1) = 4$, which is the expected result.

Final Remarks

The theory of computability cuts deep into the philosophical underpinnings of mathematics itself. While this work has focused on a specific application—the word problem for semigroups—the broader implications deserve careful consideration.

Gödel's Incompleteness Theorems and Church's thesis should be viewed neither in isolation from other mathematical domains nor as philosophical "escape hatches" to conclude that mathematical pursuit is futile. Similarly, while the current trend toward formalism offers valuable tools and frames of reference, we should also resist the temptation of reducing mathematics to a mere manipulation of symbols.

Mathematicians have a duty to champion analytical thought. However, we must also recognize the intrinsic boundaries of our theoretical frameworks, which must be continually tested and problematized. This is particularly important in a moment in history dominated by conversations about artificial intelligence and its role in society.

The relationship between computability theory and other mathematical disciplines points toward a more nuanced epistemological—and indeed ontological—approach to the foundations of mathematics and science. This approach should combine technical rigor with philosophical insight, fostering a perspective that acknowledges both the power and limitations of mathematical thinking in advancing human knowledge.

Bibliography

[1] William W. Boone. “Certain Simple, Unsolvable Problems of Group Theory. I.” In: *Indagationes Mathematicae (Proceedings)* 57 (1954), pp. 231–237. ISSN: 1385-7258. DOI: 10.1016/s1385-7258(54)50033-8.

[2] William W. Boone. “Certain Simple, Unsolvable Problems of Group Theory. II.” In: *Indagationes Mathematicae (Proceedings)* 57 (1954), pp. 492–497. ISSN: 1385-7258. DOI: 10.1016/s1385-7258(54)50061-2.

[3] William W. Boone. “Certain Simple, Unsolvable Problems of Group Theory. III.” In: *Indagationes Mathematicae (Proceedings)* 58 (1955), pp. 252–256. ISSN: 1385-7258. DOI: 10.1016/S1385-7258(55)50032-1.

[4] William W. Boone. “Certain Simple, Unsolvable Problems of Group Theory. IV.” In: *Indagationes Mathematicae (Proceedings)* 58 (1955), pp. 571–577. ISSN: 1385-7258. DOI: 10.1016/S1385-7258(55)50079-5.

[5] William W. Boone. “Certain Simple, Unsolvable Problems of Group Theory. V.” In: *Indagationes Mathematicae (Proceedings)* 60 (1957), pp. 22–27. ISSN: 1385-7258. DOI: 10.1016/S1385-7258(57)50003-6.

[6] William W. Boone. “Certain Simple, Unsolvable Problems of Group Theory. VI.” In: *Indagationes Mathematicae (Proceedings)* 60 (1957), pp. 227–232. ISSN: 1385-7258. DOI: 10.1016/S1385-7258(57)50030-9.

[7] John L. Britton. “P. S. Novikov. Ob algoritmičéskoj nérazréšimosti problémy toždéstva slov v téorii grupp (Algorithmic Unsolvability of the Word Problem in Group Theory).”

In: *Journal of Symbolic Logic* 23.1 (Mar. 1958), pp. 50–52. ISSN: 00224812. DOI: 10.2307/2964487. URL: https://www.jstor.org/stable/2964487.

[8] Alonzo Church. "An Unsolvable Problem of Elementary Number Theory." In: *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Ed. by Martin Davis. Orig. publ.: Hewlett, NY, Raven Press Books, Ltd., 1965. Mineola, NY: Dover Publications, Inc., 2004, pp. 89–107. ISBN: 9780486432281.

[9] Martin Davis. *Computability & Unsolvability*. Orig. publ.: New York, McGraw-Hill, 1958. New York, NY: Dover Publications, Inc., 1982. 248 pp. ISBN: 9780486614717.

[10] Kurt Gödel. "On Formally Undecidable Propositions of Principia Mathematica and Related Systems. I." In: *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Ed. by Martin Davis. Orig. publ.: Hewlett, NY, Raven Press Books, Ltd., 1965. Mineola, NY: Dover Publications, Inc., 2004, pp. 4–38. ISBN: 9780486432281.

[11] Kurt Gödel. "On Undecidable Propositions of Formal Mathematical Systems." In: *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Ed. by Martin Davis. Orig. publ.: Hewlett, NY, Raven Press Books, Ltd., 1965. Mineola, NY: Dover Publications, Inc., 2004, pp. 39–74. ISBN: 9780486432281.

[12] Pierre A. Grillet. *Abstract Algebra*. 2nd ed. Graduate Texts in Mathematics. New York, NY: Springer, 2007. 686 pp. ISBN: 9780387715674. DOI: 10.1007/978-0-387-71568-1.

[13] Pierre A. Grillet. *Semigroups. An Introduction to the Structure Theory*. Monographs and Textbooks in Pure and Applied Mathematics 193. New York, NY: Marcel Dekker, Inc., 1995. 398 pp. ISBN: 0824796624.

[14] Graham Higman. "Subgroups of Finitely Presented Groups." In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 262.1311 (1961), pp. 455–475. DOI: 10.1098/rspa.1961.0132.

[15] John M. Howie. *An Introduction to Semigroup Theory*. Ed. by P. M. Cohn and G. E. H. Reuter. L.M.S. Monographs 7. London, UK: Academic Press Inc., 1976. 272 pp. ISBN: 9780123569509.

[16] Thomas W. Judson. *Abstract Algebra. Theory and Applications*. July 28, 2022. URL: http://abstract.pugetsound.edu (visited on 2024-11-12).

[17] Stephen C. Kleene. "General Recursive Functions of Natural Numbers." In: *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Ed. by Martin Davis. Orig. publ.: Hewlett, NY, Raven Press Books, Ltd., 1965. Mineola, NY: Dover Publications, Inc., 2004, pp. 236–253. ISBN: 9780486432281.

[18] Stephen C. Kleene. "Recursive Predicates and Quantifiers." In: *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Ed. by Martin Davis. Orig. publ.: Hewlett, NY, Raven Press Books, Ltd., 1965. Mineola, NY: Dover Publications, Inc., 2004, pp. 254–287. ISBN: 9780486432281.

[19] Charles F. Miller III. "Turing Machines to Word Problems." In: *Turing's Legacy. Developments from Turing's Ideas in Logic*. Cambridge University Press, May 2014, pp. 329–385. DOI: 10.1017/CBO9781107338579.010.

[20] Emil L. Post. "Finite Combinatory Processes. Formulation 1." In: *Journal of Symbolic Logic* 1.3 (1936), pp. 103–105. ISSN: 00224812. DOI: 10.2307/2269031. URL: https://www.jstor.org/stable/2269031.

[21] Emil L. Post. "Formal Reductions of the General Combinatorial Decision Problem." In: *American Journal of Mathematics* 65.2 (Apr. 1943), pp. 197–215. ISSN: 0002-9327. DOI: 10.2307/2371809. URL: https://www.jstor.org/stable/2371809.

[22] Emil L. Post. "Recursive Unsolvability of a Problem of Thue." In: *Journal of Symbolic Logic* 12.1 (Mar. 1947), pp. 1–11. ISSN: 00224812. DOI: 10.2307/2267170. URL: https://www.jstor.org/stable/2267170.

[23] Emil L. Post. "Recursively Enumerable Sets of Positive Integers and Their Decision Problems." In: *Bulletin of the American Mathematical Society* 50.5 (1944), pp. 284–316.

[24] James F. Power. *Thue's 1914 Paper: A Translation*. 2013. arXiv: 1308.5858 [cs.FL]. URL: https://arxiv.org/abs/1308.5858.

[25] Dana S. Scott. "A. A. Markov. Nérazrěšimosť problémy goméomorfii (Insolubility of the Problem of Homeomorphy)." In: *Journal of Symbolic Logic* 27.1 (Mar. 1962), p. 99. ISSN: 00224812. DOI: 10.2307/2963718. URL: https://www.jstor.org/stable/2963718.

[26] Michael Spivak. *Calculus*. 4th ed. Houston, Texas: Publish or Perish, Inc., 2008. 680 pp. ISBN: 9780914098911.

[27] John Stillwell. "Emil Post and his Anticipation of Gödel and Turing." In: *Mathematics Magazine* 77.1 (Feb. 2004), pp. 3–14. ISSN: 0025-570X. DOI: 10.2307/3219226.

[28] Alfred Tarski. "A General Method in Proofs of Undecidability." In: Alfred Tarski, Andrzej Mostowski, and Raphael M. Robinson. *Undecidable Theories. Studies in Logic and the Foundation of Mathematics*. Orig. publ.: Amsterdam, North-Holland Publishing Co., 1953. Garden City, NY: Dover Publications, Inc., 2010, pp. 1–35. ISBN: 9780486477039.

[29] James J. Tattersall. *Elementary Number Theory in Nine Chapters*. 2nd ed. Cambridge, UK: Cambridge University Press, 2005. 430 pp. ISBN: 9780521615242.

[30] Alan M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem." In: *Proceedings of the London Mathematical Society* s2-42.1 (1937), pp. 230–265. ISSN: 0024-6115. DOI: 10.1112/plms/s2-42.1.230.

[31] Alan M. Turing. "The Word Problem in Semi-Groups with Cancellation." In: *Annals of Mathematics* 52.2 (Sept. 1950), pp. 491–505. ISSN: 0003-486X. DOI: 10.2307/1969481.

[32] Richard Zach. *Incompleteness and Computability. An Open Introduction to Gödel's Theorems*. 2021. ISBN: 9781077323391. URL: https://ic.openlogicproject.org (visited on 2024-10-31).

Appendix A

Multiplier quadruples

The quadruples of Z_2 in Chapter 3 are:

$q_1\ 1\ 0\ q_1$,

$q_1\ 0\ R\ q_2$,

$q_2\ 1\ S_2\ q_3$,

$q_3\ S_2\ R\ q_3$,

$q_3\ 1\ R\ q_3$,

$q_3\ 0\ R\ q_4$,

$q_4\ 1\ R\ q_3$,

$q_4\ 0\ L\ q_5$,

$q_5\ 1\ L\ q_6$,

$q_5\ 0\ L\ q_6$,

$q_6\ 1\ S_3\ q_6$,

$q_6\ S_3\ R\ q_7$,

$q_6\ 0\ 0\ q_{10}$,

$q_7\ 1\ R\ q_7$,

$q_7\ 0\ R\ q_8$,

$q_8\ 1\ R\ q_8$,

$q_8\ 0\ 1\ q_9$,

$q_9\ 1\ L\ q_9$,

$q_9\ 0\ L\ q_9$,

$q_9\ S_3\ 1\ q_5$,

$q_{10}\ 1\ L\ q_{10}$,

$q_{10}\ 0\ L\ q_{10}$,

$q_{10}\ S_2\ 0\ q_1$.