4.2 The Halting Problem

- The technique of diagonalization was discovered in 1873 by Georg Cantor who was concerned with the problem of measuring the sizes of infinite sets.

- For finite sets we can simply count the elements.

- Do infinite sets $\mathbb{N} = \{0, 1, 2, \ldots\}$ and $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$ have the same size?

  - $\mathbb{N}$ is larger because it contains the extra element 0 and all other elements of $\mathbb{Z}^+$.
  - The sets have the same size because each element of $\mathbb{Z}^+$ can be mapped to an unique element of $\mathbb{N}$ by $f(z) = z-1$.

- We have already used this comparison of sizes:
  - $|A| \leq |B|$ iff there exists a one-to-one function $f: A \rightarrow B$.
  - One-to-one (injection): $a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2)$.

- We have also examined the equal size of two sets $|A| = |B|$ through a bijective $f: A \rightarrow B$ mapping:
  - Correspondence (bijection) = one-to-one + onto.
  - Onto (surjection): $f(A) = B$ or $\forall b \in B \exists a \in A: b = f(a)$.
  - A bijection uniquely pairs the elements of the sets $A$ and $B$.

  - $|\mathbb{N}| = |\mathbb{Z}|$.
  - An infinite set that has the same size as $\mathbb{N}$ is countable.
The set of rational numbers 
\[ Q = \{ \frac{m}{n} \mid m, n \in \mathbb{Z} \land n \neq 0 \} \]

- In between any two integers there is an infinite number of rational numbers
- Nevertheless, \( |Q| = |N| \)

- Mapping \( f : Q \rightarrow \mathbb{Z}^2, f(m/n) = (m, n) \)
  - Integers \( m \) and \( n \) have no common factors!
  - Mapping \( f \) is a one-to-one. Hence, \( |Q| \leq |\mathbb{Z}^2| = |N| \)

- On the other hand, \( N \subseteq Q \Rightarrow |N| \leq |Q| \)

\[ \therefore |Q| = |N| \]
Let us assume that the interval $[0, 1[$ is countable and apply Cantor’s diagonalization to the numbers $x_1, x_2, x_3, \ldots$ in $[0, 1[$.

Let the decimal representations of the numbers within the interval be (excluding infinite sequences of 9s)

$$x_i = \sum_{j=1}^{\infty} d_{ij} \cdot 10^{-j}$$

Let us construct a new real number $x = \sum_{j=1}^{\infty} d_j \cdot 10^{-j}$ such that

$$d_j = \begin{cases} 0, & \text{if } d_j > 0 \\ 1, & \text{if } d_j = 0 \end{cases}$$

If, for example

$$x_1 = 0,23246...$$
$$x_2 = 0,30589...$$
$$x_3 = 0,21254...$$
$$x_4 = 0,05424...$$
$$x_5 = 0,99548...$$

then $x = 0,01000...$

Hence, $x \neq x_i$ for all $i$.

The assumption about the countability of the numbers within the interval $[0, 1[$ is false

$|\mathbb{R}| = |\mathbb{R} [0, 1[^{|} \neq |\mathbb{N}|$
Universal Turing Machines

- The universal language $U$ over the alphabet $\{0, 1\}$ is
  \[ U = \{ \langle M, w \rangle \mid w \in L(M) \}. \]

- The language $U$ contains information on all Turing-recognizable language over $\{0, 1\}$:
  - Let $A \subseteq \{0, 1\}^*$ be some Turing-recognizable language and $M$ a standard TM recognizing $A$. Then
    \[ A = \{ w \in \{0, 1\}^* \mid \langle M, w \rangle \in U \}. \]

- Also $U$ is Turing-recognizable.
- Turing machines recognizing $U$ are called universal Turing machines.

**Theorem E** Language $U$ is Turing-recognizable.

**Proof.** The following three-tape TM $M_U$ recognizes $U$

1. First $M_U$ checks that the input $cw$ in tape 1 contains a legal encoding $c$ of a Turing machine. If not, $M_U$ rejects the input
2. Otherwise $w = a_1a_2...a_k \in \{0, 1\}^*$ is copied to tape 2 in the form
   \[ 00010^{a_1}10^{a_2}10^{a_3}...10^{a_k}10000 \]
3. Now $M_U$ has to find out whether the TM $M (c = \langle M \rangle)$ would accept $w$. Tape 1 contains the description $c$ of $M$, tape 2 simulates the tape of $M$, and tape 3 keeps track of the state of the TM $M$:
   \[ q_i \sim 0^{i+1} \]
4. $M_U$ works in phases, simulating one transition of $M$ at each step

1. First $M_U$ searches the position of the encoding of $M$ (tape 1) that corresponds to the simulated state (tape 3) of $M$ and the symbol in tape 2 at the position of the tape head

2. Let the chosen sequence of encoding be
   
   $0^{i+1}10^{r+1}10^{s+1}10^{t+1}$,
   
   which corresponds transition function $\delta$ rule
   
   $\delta(q_i, a_j) = (q_r, a_s, \delta_t)$.

   tape 3: $0^{i+1} \to 0^{r+1}$
   
   tape 2: $0^{t+1} \to 0^{s+1}$

   In addition the head of tape 2 is moved to the left so that the code of one symbol is passed, if $t = 0$, and to the right otherwise

3. When tape 1 does not contain any code for the simulated state $q_i$, $M$ has reached a final state. Now $i = k + 1$ or $i = k + 2$, where $q_k$ is the last encoded state. The TM $M_U$ transitions to final state accept or reject.

   Clearly the TM $M_U$ accepts the binary string $\langle M, w \rangle$ if and only if $w \in L(M)$.

   $\square$
Theorem F Language \( U \) is not decidable.

Proof. Let us assume that \( U \) has a total recognizer \( M_{U^T} \).

Then we could construct a recognizer \( M_D \) for the "diagonal language" \( D \), which is not Turing-recognizable (Lemma D), based on \( M_{U^T} \) and the following total Turing machines:

- \( M_{OK} \) tests whether the input binary string is a valid encoding of a Turing machine
- \( M_{DUP} \) duplicates the input string \( c \) to the tape: \( cc \)

Combining the machines as shown in the next picture, we get the Turing machine \( M_D \), which is total whenever \( M_{U^T} \) is. Moreover,

\[
\begin{align*}
e & \in L(M_D) \\
\Leftrightarrow & \ c \notin L(M_{OK}) \lor cc \notin L(M_{U^T}) \\
\Leftrightarrow & \ c \notin L(M_U) \\
\Leftrightarrow & \ c \in D = \{ c | c \notin L(M) \} \\
\end{align*}
\]

By Lemma D the language \( D \) is not decidable. Hence, we have a contradiction and the assumption must be false. Therefore, there cannot exist a total recognizer \( M_{U^T} \) for the language \( U \). \( \Box \)
Corollary G \( \mathcal{D} = \{ \langle M, w \rangle \mid w \notin L(M) \} \) is not Turing-recognizable.

Proof. \( \mathcal{D} = \mathcal{D} \cup \text{ERR} \), where \( \text{ERR} \) is the easy to decide language:

\[
\text{ERR} = \{ x \in \{0, 1\}^* \mid x \text{ does not have a prefix that is a valid code for a Turing machine} \}.
\]

Counter-assumption: \( \mathcal{D} \) is Turing-recognizable

- Then by Theorem B, \( \mathcal{D} \cup \text{ERR} = \mathcal{D} \) is Turing-recognizable.
- \( U \) is known to be Turing-recognizable (Th. E) and now also \( \mathcal{D} \) is Turing-recognizable. Hence, by Theorem 4.22, \( U \) is decidable.

This is a contradiction with Theorem F and the counter-assumption does not hold. I.e., \( \mathcal{D} \) is not Turing-recognizable. \( \square \)
The Halting Problem is Undecidable

- Analogously to the acceptance problem of DFAs, we can pose the halting problem of Turing machines:
  
  Does the given Turing machine \( M \) halt on input \( w \) ?

- This is an undecidable problem. If it could be decided, we could easily decide also the universal language

**Theorem 5.1**

\[ \text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and halts on input } w \} \]

is Turing-recognizable, but not decidable.

**Proof.** \( \text{HALT}_{TM} \) is Turing-recognizable: The universal Turing machine \( M_U \) of Theorem E is easy to convert into a TM that simulates the computation of \( M \) on input \( w \) and accepts if and only if the computation being simulated halts.

\( \text{HALT}_{TM} \) is not decidable: counter-assumption: \( \text{HALT}_{TM} = L(M\text{HALT}) \) for the total Turing machine \( M_{\text{HALT}} \).

Now we can compose a decider for the language \( U \) by combining machines \( M_U \) and \( M_{\text{HALT}} \) as shown in the next figure.

The existence of such a TM is a contradiction with Theorem F. Hence, the counter-assumption cannot hold and \( \text{HALT}_{TM} \) is not decidable.

**Corollary H** \( \bar{U} = \{ \langle M, w \rangle \mid M \text{ is a TM and does not halt on input } w \} \) is not Turing-recognizable.

**Proof.** Like in corollary G.
A total TM for the universal language $U$

Chomsky hierarchy

- A formal language $L$ can be recognized with a Turing machine if and only if it can be generated by an unrestricted grammar.

- Hence, the languages generated by unrestricted grammars are Turing-recognizable languages.

- They constitute type 0 languages of Chomsky hierarchy.

- Chomsky’s type 1 languages are the context-sensitive ones. It can be shown that they are all decidable.

- On the other hand, there exists decidable languages, which cannot be generated by context-sensitive grammars.
Halting Problem in Programming Language

The correspondence between Turing machines and programming languages:

All TMs ~ programming language
One TM ~ program
The code of a TM ~ representation of a program in machine code
Universal TM ~ interpreter for machine language

The interpretation of the undecidability of the halting problem in programming languages:

```
There does not exist a Java method, which could decide whether any given Java method M halts on input w`.
```
Let us assume that there exists a total Java method $h$ that returns $true$ if the method represented by string $m$ halts on input $w$ and $false$ otherwise:

$$boolean \ h(String \ m, \ String \ w)$$

Now we can program the method $hHat$

```java
boolean hHat(String m) {
    if (h(m, m))
        while (true);
}
```

Let $H$ be the string representation of $hHat$. $hHat$ works as follows:

$hHat(H)$ halts $\Leftrightarrow h(H, H) = false$ $\Leftrightarrow hHat(H)$ does not halt

---

5. Reducibility

- The proof of unsolvability of the halting problem is an example of a reduction:
  - a way of converting problem $A$ to problem $B$ in such a way that a solution to problem $B$ can be used to solve problem $A$
  - If the halting problem were decidable, then the universal language would also be decidable
  - Reducibility says nothing about solving either of the problems alone; they just have this connection
  - We know from other sources that the universal language is not decidable
  - When problem $A$ is reducible to problem $B$, solving $A$ cannot be harder than solving $B$ because a solution to $B$ gives one to $A$
  - If an unsolvable problem is reducible to another problem, the latter also must be unsolvable
Non-emptiness Testing for TMs

(Observe that the book deals with $E_{TM}$.)

The following decision problem is undecidable:

"Does the given Turing machine accept any inputs?"

$NE_{TM} = \{ \langle M \rangle | M \text{ is a Turing machine and } L(M) \neq \emptyset \}$

**Theorem (5.2)** $NE_{TM}$ is Turing-recognizable, but not decidable

**Proof.** The fact that $NE_{TM}$ is Turing-recognizable will be shown in the exercises.

- Let us assume that $NE_{TM}$ has a decider $MT_{NE}$
- Using it we can construct a total Turing machine for the language $U$
- Let $M$ be an arbitrary Turing machine, whose operation on input $w$ is under scrutiny

Let $M^w$ be a Turing machine that replaces its actual input with the string $w = a_1a_2...a_k$ and then works as $M$

- Operation of $M^w$ does not depend in any way about the actual input.
  - The TM either accepts or rejects all inputs:

$$L(M^w) = \begin{cases} \{0,1\}^*, & \text{if } w \in L(M) \\ 0, & \text{if } w \not\in L(M) \end{cases}$$
Let $M_{\text{ENC}}$ be a TM, which
- Inputs the concatenation of the code $\langle M \rangle$ for a Turing machine $M$ and a binary string $w$, $\langle M, w \rangle$, and
- Leaves to the tape the code $\langle M^w \rangle$ of the TM $M^w$

By combining $M_{\text{ENC}}$ and the decider $M_{\text{TNE}}^T$ for the language $\text{NE}_{\text{TM}}$, we are now able to construct the following Turing machine $M_{\text{UT}}$. 

The Turing machine $M^w$
A decider $M_U$ for the universal language $U$

- $M_U$ is total whenever $M_{NE}$ is, and $L(M_U) = U$ because

$$\langle M, w \rangle \in L(M_U)$$

- However, by Theorem F $U$ is not decidable, and the existence of the TM $M_U$ is a contradiction
- Hence, the language $NE_{TM}$ cannot have a total recognizer $M_{NE}$ and we have, thus, proved that the language $NE_{TM}$ is not decidable.

$\Box$
TMs Recognizing Regular Languages

Similarly, we can show that recognizing those Turing machines that accept a regular language is undecidable by reducing the decidability of the universal language into this problem.

The decision problem is:

"Does the given Turing machine accept a regular language?"

\[ \text{REG}_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) \text{ is a regular language} \} \]

Theorem 5.3 \text{REG}_{TM} is undecidable.

Proof.

- Let us assume that \text{REG}_{TM} has a decider \( M^{\text{REG}} \)

- Using \( M^{\text{REG}} \) we could construct a decider for the universal language \( U \)

- Let \( M \) be an arbitrary Turing machine, whose operation on input \( w \) we are interested in

- The language corresponding to balanced pairs of parenthesis \( \{ 0^n 1^n | n \geq 0 \} \) is not regular, but easy to decide using a TM

- Let \( M_{\text{parenth}} \) be a decider for the language

- Now, let \( M_{\text{encode}} \) be a TM, which on input \( \langle M, w \rangle \) composes an encoding for a TM \( M^e \), which on input \( x \)
  - First works as \( M_{\text{parenth}} \) on input \( x \).
  - If \( M_{\text{parenth}} \) rejects \( x \), \( M^e \) operates as \( M \) on input \( w \).
  - Otherwise \( M^e \) accepts \( x \)
Deciding a regular language: the TM $M^w$

Thus, $M^w$ either accepts the regular language $\{0, 1\}^*$ or non-regular $\{0^n1^n \mid n \geq 0\}$.

Accepting/rejecting the string $w$ on $M$ reduces to the question of the regularity of the language of the TM $M^w$.

$L(M^w) = \begin{cases} \{0, 1\}^* & \text{if } w \in L(M) \\ \{0^n1^n \mid n \geq 0\} & \text{if } w \not\in L(M) \end{cases}$

Let $M_{ENC}$ be a TM, which
- inputs the concatenation of the code $\langle M \rangle$ for a Turing machine $M$ and a binary string $w$, $\langle M, w \rangle$, and
- leaves to the tape the code $\langle M^w \rangle$ of the TM $M^w$.

Now by combining $M_{ENC}$ and $M_{REG}$ would yield the following Turing machine $M_T^w$. 

![Diagram of Turing Machine](image)
A decider $M_T^T$ for the universal language $U$.

- $M_T^T$ is total whenever $M'_\text{REG}$ is and $L(M_T^T) = U$, because

$$\langle M, w \rangle \in L(M_T^T) \iff \langle M' \rangle \in L(M'_\text{REG}) = \text{REG}_T$$

- By Theorem F, $U$ is not decidable, and the existence of the TM $M_T^T$ is a contradiction.
- Hence, language $\text{REG}_T$ cannot have a decider $M'_\text{REG}$.
- Thus, we have shown that the language $\text{REG}_T$ is not decidable.
Rice’s Theorem

- Any property that only depends on the language recognized by a TM, not on its syntactic details, is called a semantic property of the Turing machine.
- E.g.
  - "M accepts the empty string",
  - "M accepts some string" (NE),
  - "M accepts infinitely many strings",
  - "The language of M is regular" (REG) etc.

- If two Turing machines $M_1$ and $M_2$ have $L(M_1) = L(M_2)$, then they have exactly the same semantic properties.

More abstractly: a semantic property $S$ is any collection of Turing-recognizable languages over the alphabet $\{0, 1\}$.

- Turing machine $M$ has property $S$ if $L(M) \in S$.
- Trivial properties are $S = \emptyset$ and $S = \text{TR}$.
- Property $S$ is solvable, if language $\text{codes}(S) = \{ \langle M \rangle \mid L(M) \in S \}$ is decidable.

Rice’s theorem. All non-trivial semantic properties of Turing machines are unsolvable.