Undecidability

Last Updated: December 9th, 2025

1 Introduction

In this lecture we look at several problems which cannot be decided by a Turing machine. We call these problems *undecidable*. The theory of undecidability has immense applications to computer science and mathematics. For example, mathematicians had long dreamed of the day when an algorithm would be developed which, given any multivariate polynomial with integer cofficients, could determine the integral roots of the polynomial. This is known as “Hilbert’s Tenth Problem”. In 1970, Yuri Matiyasevich proved that no such algorithm exists by proving that a related problem is undecidable.

Definition 1.1. A language L over some alphabet Σ is said to be **undecidable** iff it is not Turing decidable, meaning that there is no Turing machine M that halts on all inputs, and for which $L = L(M)$.

Encoding a Turing Machine as a Binary Word

Let

$$M = (Q \subset \{0,1,2,\ldots\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\sqcup\}, \delta, q_0, q_a = 0)$$

be a Turing machine whose state set Q is a finite set of nonnegative integers, whose input alphabet is binary, and whose accepting state is 0. We may write M as a word over the alphabet

$$\Sigma_{\mathrm{TM}} = \{0,1,\ldots,9,\sqcup,L,R,\#,\$\},$$

where \$ is used to delimit tuples of the δ-transition table, and # is used to delimit components within a given tuple.

Example 1.2. The following word over Σ_{TM} describes a Turing machine that accepts all binary inputs having an even number of zeros.

Universal TM: one that can simulate any TM on any input

$$\$2\#0\#3\#0\#R\$2\#1\#2\#1\#R\$2\#\sqcup\#\mathbf{0}\#\sqcup\#L\$3\#0\#2\#0\#R\$3\#1\#3\#1\#R\$.$$

State read symbol next state

$q_0 = 2$ $q_a = 0$

Here we use the convention that i) the initial state $q_0 = 2$ equals the first state of the first tuple, ii) the accepting state $q_a = 0$, and q_r is undefined, in that the machine rejects in case it reaches a state q with the head reading a symbol s, and for which $\delta(q,s)$ is undefined.

In the remainder of this lecture we use the notation $\langle M \rangle$ to denote the encoding of M as a word over Σ_{TM}. □

1.1 Lexicographical ordering words over an alphabet

Given an alphabet $\Sigma = \{a_1, a_2, \ldots, a_m\}$, assume an order of the letters so that

$$a_1 < a_2 < \cdots < a_m.$$

Then we may use this ordering to order all words over Σ^*. Let $u, v \in \Sigma^*$ be two distinct words over Σ. Then the following rules determine which word precedes the other in what is called the **lecicographical ordering** of the words.

1. Shorter words precede longer ones.
2. If $|u| = |v|$, then let i be the first letter (from left to right) for which $u_i \neq v_i$. If $u_i < v_i$, then u precedes v. Otherwise v precedes u.

$$2^0 + 2^1 + 2^2 + \cdots + 2^{n-1} = 2^n - 1$$

Example 1.3. Assuming $0 < 1$, the first twelve words in the lexicographical ordering of $\{0,1\}^*$ are

$$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100. \quad \square \quad 101$$

$$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \qquad 12$$

Definition 1.4. Let Σ be an alphabet whose letters are ordered. Then for any word $w \in \Sigma^*$, $\phi(w)$ denotes its lexicographical order.

Example 1.5. For the lexicographical ordering of binary words, we have $\phi(\varepsilon) = 0$, $\phi(10) = 5$, and $\phi(100) = 11$. □

The following proposition is left as an exercise.

Proposition 1.6. Given an alphabet $\Sigma = \{\underline{a_1}, \underline{a_2}, \ldots, \underline{a_m}\}$ and a word $w = a_{i_1}a_{i_2}\cdots a_{i_n}$, where

words of length $< |w|$

$$1 \le i_1, \ldots, i_n \le m,$$

$$\phi(w) = \boxed{\frac{m^n - 1}{m - 1}} + \underline{(i_1 - 1)}m^{n-1} + (i_2 - 1)m^{n-2} + \cdots + (i_{n-1} - 1)m + (i_n - 1).$$

Example 1.7. Use the above formula confirm that i) $\phi(101) = 11$ and ii) determine $\phi(11001)$ (assuming a binary alphabet).

Solution. $m = 2$ $\qquad \dfrac{2^n - 1}{2 - 1} = 2^n - 1 \qquad \underline{0}\ \underline{\{0,1\}}\underline{\{0,1\}}$

$|101| = 3 \qquad 2^3 - 1 + 2^2 + \underline{1} = 12 \Rightarrow$

$\phi(101) = 12$

101

100

$\phi(11001) = (2^5 - 1) + 2^4 + 2^3 + 1 =$

$\boxed{56}$

$0\ \underline{2} \times \underline{2} \times \underline{2} \times \underline{2}$

$1\ 0\ \underline{2} \times \underline{2} \times \underline{2}$

$1\ 1\ 0\ 0\ \underline{0}$

Definition 1.8. Let

$$M = (Q \subset \{0,1,2,\ldots\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\sqcup\}, \delta, q_0, q_a = 0)$$

be a Turing machine. Then the **Gödel number** of M is defined as $\underline{\phi(\langle M \rangle)}$ with respect to the lexicographical ordering of words over Σ_{TM}, where the ordering of Σ_{TM} is

$$0 < 1 < \cdots < 9 < \sqcup < L < R < \# < \$\}.$$

We see that associated with every Turing machine M is the Gödel number $\phi(\langle M \rangle)$. However, not every natural number is the Gödel number for some Turing machine. In fact, the fraction of numbers that are Gödel numbers is vanishingly small, meaning that, if we let $\varepsilon(n)$ denote the number of natural numbers that are Gödel numbers and that do not exceed n, then $\varepsilon(n)/n \to 0$ as n becomes increasingly large. However, we wish to associate with *every* natural number i some Turing machine M_i. Let $M_\emptyset$ denote the Turing machine whose δ-transition table is empty, meaning that $L(M_\emptyset) = \emptyset$. Then the following procedure can be used to define the i th Turing machine M_i, for $i = 0, 1, \ldots$.

```
Input: natural number $i \geq 0$.
Output: Turing machine $M_i$.
If there is a TM $M$ for which $\underline{\phi(\langle M \rangle) = i}$ then return $M$.
Return $M_\emptyset$.
```

The Diagonalization Method

The diagonalization method was first invented by the Russian-German mathematician George Cantor. Cantor used it to prove that the real numbers are not countably infinite, i.e. cannot be enumerated in a list. The proof is by contradiction and assumes that the real numbers between zero and one can be enumerated in a list as $x_0, x_1, x_2, \ldots x_k, \ldots$. Moreover, letting x_{ij} denote the j th digit of the i th real number, we get an infinite matrix (in both rows and columns) of digits. The key idea is to use the diagonal of this matrix to define a real number y whose i th digit y_i is defined as $(x_{ii}+1)$ mod 10. By defining y is this way, we see that $y \neq x_i$ for all $i \geq 0$. This is because digit i of y diagrees with digit i of x_i:

$$y_i = (x_{ii} + 1) \bmod 10 \neq x_{ii}.$$

This contradicts the assumption that all real numbers are in the list. Therefore, the set of real numbers between 0 and 1 (and hence the set of all real numbers) cannot be enumerated in a list. □

The following table demonstrates the diagonalization method with respect to numbers between 0 and 1.

$x_0 = 0.117\ldots4\ldots$

index\nth digit	0	1	2	⋯	k	⋯	**Observation**
x_0	$1 \to 2$	1	7	⋯	4	⋯	$y_0 = 2 \neq x_{00} = 1$
x_1	4	$5 \to 6$	3	⋯	7	⋯	$y_1 = 6 \neq x_{11} = 5$
x_2	8	7	$9 \to 0$	⋯	6	⋯	$y_2 = 0 \neq x_{22} = 9$
⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
x_k	4	1	9	⋯	$0 \to 1$	⋯	$y_k = 1 \neq x_{kk} = 0$
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮

$x_1 = 0.453\ldots7\ldots$

$y = 0.260\ldots6\ldots \neq x_i$

for every $i \geq 0$

Our First Undecidable Problem

Let M be a Turing machine whose input words are binary, and let $w \in \{0,1\}^*$ be an input to M, then we may encode both M and w together as one word, denoted $\langle M, w\rangle$. One way of doing this is to append w to $\langle M\rangle$, i.e.

$$\langle M, w\rangle = \langle M\rangle \circ w.$$

Using this encoding we may now define the binary language

$$\texttt{Accept}_{\mathrm{TM}} = \{\langle M, w\rangle \,|M \text{ accepts } w\}.$$

Theorem 1.9. $\texttt{Accept}_{\mathrm{TM}}$ is undecidable.

$D(\langle M, w\rangle) = 1$ iff $M(w) = 1$

Proof. The proof uses the method of diagonalization. With the goal of showing a contradiction, suppose $\texttt{Accept}_{\mathrm{TM}}$ is decidable via Turing machine D (D stands for Decide). Consider the Turing machine C (C stands for Contradict) that implements the following program.

Program for Turing Machine C

Input: $\langle M\rangle$, the encoding of an arbitrary Turing machine M.

Output: accept or rejecting, depending on the computation of D on input $\langle M, \langle M\rangle\rangle$.

Simulate D on input $\langle M, \langle M\rangle\rangle$.

If D accepts, then reject.

Else accept.

Notice that, since D halts on all inputs, so does C. Now, what happens when $\langle C\rangle$ serves as input to C? In other words, what happens when C computes with its own encoding as input?

Case 1. C accepts $\langle C\rangle$. By definition of C, this can only happen if D rejects $\langle C, \langle C\rangle\rangle$ which, by definition of D, means that C rejects $\langle C\rangle$, a contradiction.

Case 2. C rejects $\langle C\rangle$. By definition of C, this can only happen if D accepts $\langle C, \langle C\rangle\rangle$ which, by definition of D, means that C accepts $\langle C\rangle$, a contradiction.

Therefore, our assumption that $\texttt{Accept}_{\mathrm{TM}}$ is decidable via D must be false, and A_{TM} is undecdiable. □

The following table suggests that the above proof can be understood as another diagonalization argument. Indeed, the way in which C is defined makes it behave differently than every other Turing machine M_i, on at least one input: namely, $\langle M_i \rangle$. But this is impossible, since C is also a Turing machine, and we are assuming that the list of machines includes all possible Turing machines (e.g. M_i is the Turing machine with Gödel number i). Then, on input $\langle C \rangle$, C must behave in the opposite way in which C would behave on input $\langle C \rangle$, which is impossible! In other words,

$$C(\langle C \rangle) = 0 \Leftrightarrow C(\langle C \rangle) = 1,$$

a contradiction. Therefore, we must assume that C is not well defined. But the only part of C's definition that is questionable is whether or not TM D exists, which in turn relies on whether or not $\texttt{Accept}_{\mathrm{TM}}$ is decidable. Threfore, $\texttt{Accept}_{\mathrm{TM}}$ is undecidable.

TM\input n	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\cdots$	$\langle C \rangle$	$\cdots$	**self accepting?**
M_0	$1 \rightarrow 0$	1	0	$\cdots$	1	$\cdots$	yes
M_1	$\uparrow$	$\uparrow \rightarrow 1$	0	$\cdots$	$\uparrow$	$\cdots$	no
M_2	0	1	$1 \rightarrow 0$	$\cdots$	1	$\cdots$	yes
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
C	0	1	0	$\cdots$	$0/1 \rightarrow 1/0$	$\cdots$	no/yes
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$

Note: in the table $\uparrow$ means that the machine does not halt on that input.

$\langle M_1, \langle M_1 \rangle \rangle$

Now suppose that we have a decision problem L_1 that is undecidable, we may use it prove that another decision problem L_2 is undecidable as follows.

1. Assume L_2 is decidable.
2. Let D be a Turing machine that decides L_2.
3. Use D to design another Turing machine C that decides L_1.
4. Since L_1 is undecidable, the previous step leads to a contradiction.
5. Therefore, L_2 is undecidable. □

The Halting Problem

An instance of decision problem Halt is a Turing machine M and an input w to M. The problem is to decide if M halts on input w. More formally,

$$\texttt{Halt} = \{\langle M, w\rangle \,|M \text{ halts on input } w\}.$$

Theorem 1.10. Halt is undecidable.

Proof. Assume Turing machine D decides Halt. The following program for Turing machine C decides $\texttt{Accept}_{\text{TM}}$.

```
//C Program
Input: TM $M$ and word $w$ defined over alphabet $\Sigma$, the input alphabet for $M$.

Output: accept iff $M$ accepts $w$.

Simulate $D$ on input $\langle M, w\rangle$.

If $D$ accepts $\langle M, w\rangle$,

    Simulate $M$ on input $w$.
    If $M$ accepts $w$, then accept.
    Else reject.

Else reject.
```

If D rejects $\langle M, w\rangle$, then M does not halt on input w, and thus cannot accept w, so C rejects. On the other hand, if D accepts $\langle M, w\rangle$, then M halts on input w. Hence, we may simulate M on input w and be guaranteed that the simulation will terminate. C then accepts iff the simulation of M on w is accepting.

Therefore, C decides $\texttt{Accept}_{\text{TM}}$, which contradicts the fact that $\texttt{Accept}_{\text{TM}}$ is undecidable. □

Theorem 1.11. The language

$$\texttt{Empty}_{\text{TM}} = \{\langle M\rangle\,|M \text{ is a Turing machine and } L(M) = \emptyset\}$$

is undecidable.

Proof.

Assume Turing machine D decides $\texttt{Empty}_{\text{TM}}$. The following program for Turing machine C makes use of D to decided the `Halt` problem.

> **Input:** Turing machine M and word w over the input alphabet Σ of M.
>
> **Output:** accept iff M halts on input w.
>
> Construct a Turing machine E that works in the following way. On input v, E erases v and simulates M on input w, if M halts on input w, then E accepts v. Otherwise E runs forever.
>
> Simulate D on input $\langle E\rangle$.
>
> If D accepts $\langle E\rangle$, then reject.
>
> Else accept.

Case 1: assume $\langle M, w\rangle$ is a positive instance of `Halt`. Then, when E runs on any input v, E accepts v since M halts on input w. Thus, $L(E) = \{0, 1\}^* \neq \emptyset$, which implies that D rejects E, which in turn implies that C accepts $\langle M, w\rangle$.

Case 2. assume $\langle M, w\rangle$ is a negative instance of `Halt`. Then when E runs on any input v, E (implicitly) rejects v since E will run forever on input v. This is because M runs forever on input w. Thus, $L(E) = \emptyset$, which implies that D accepts E, which in turn implies that C rejects $\langle M, w\rangle$.

Therefore, C decides `Halt`, which contradicts the fact that `Halt` is undecidable. □

Theorem 1.12. The language

$$\texttt{Equal}_{\text{TM}} = \{\langle M, N\rangle \,|L(M) = L(N)\}$$

in undecidable.

Proof. Suppose Turing machine D decides $\texttt{Equal}_{\text{TM}}$. This would imply that $\texttt{Empty}_{\text{TM}}$ is decidable. Why? Suppose we want to know of some machine M accepts the empty lanugage. Construct a machine M' for which $L(M') = \emptyset$ (simply make M' immediately enter an infinite loop). Then

$$L(M) = \emptyset \Leftrightarrow L(M) = L(M'),$$

where the latter property can be decided by D. Therefore, since was arbitrary, $\texttt{Enpty}_{\text{TM}}$ is decidable which contradicts Theorem 1.11. □