Introduction to the Theory of Computation

XU Ming

School of Software Engineering, East China Normal University

June 11, 2023

OUTLINE

- Undecidable Problem
- The Turing Machine

Undecidable Problem

Problems That Computer Cannot Solve

Example

C Programs that print “hello, world”.

```
main()
{
  printf("hello, world\n");
}
```

This program prints **hello, world** and terminates. There are other programs that also print **hello, world**; yet the fact that they do so is far from obvious.

```
main()
{
  int n, total, x, y, z;
  scanf("%d", &n);
  total=3;
  while (1) {
    for (x=1; x<=total-2; x++)
      for (y=1; y<=total-x-1; y++)
      {
        z=total-x-y;
        if (power(x,n)+power(y,n)==power(z,n))
          printf("hello, world\n");
      }
    total++;
  }
}
```

This program searches every triple of positive integers (x, y, z) in some order, and tests to see if $x^n + y^n = z^n$. If so, the program prints **hello, world**, and if not, it prints nothing.

Fermat's Last Theorem

If $n > 2$, there are no integer solutions to the equation $x^n + y^n = z^n$.

Fermat's last theorem was made by Fermat in 1637, but no proof was found until 1995.

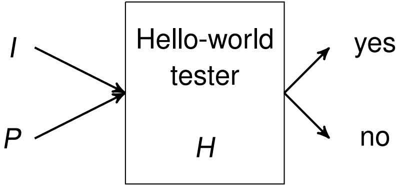
Hello-world Problem

Determine whether a given C program, with a given input, prints **hello, world** *(as the first 12 characters that it prints)*.

It seems likely that, if it takes mathematicians 350 years to resolve a question about a single program, the general problem must be hard indeed.

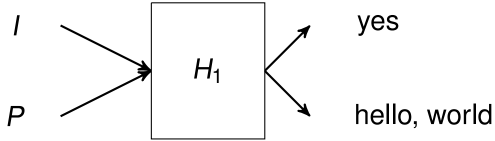
We shall prove that no program or algorithm exists to resolve “Hello-world Problem”. That means, computer cannot solve this problem.

Assume there is a program H that takes as input a program P and an input I, and tells whether P with input I prints **hello, world**.



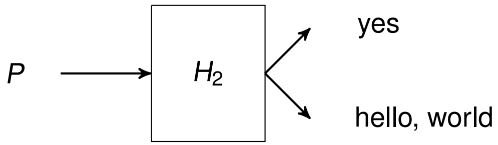
We will prove that H doesn't exist by contradiction.

First, we make a slightly modification to H. Change the output **no** of H to **hello, world**. The new program is called H_1.



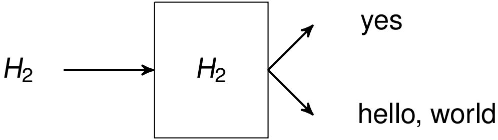
H_1 behaves like H except it prints **hello, world** exactly when H prints **no**.

The next modification we perform on H_1 to produce the program H_2, whose input is the program P with its own code as its input.



H_2 behaves like H_1, but uses its input P as both P and I.

Now we prove that H_2 cannot exist. Thus, H_1 does not exist, and likewise, H does not exist. The heart of the argument is: *what H_2 does when given itself as input*.



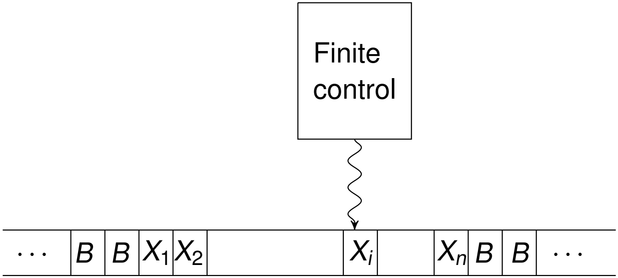
The situation is paradoxical, and we conclude that H_2 cannot exist.

A problem that cannot be solved by computer is called undecidable.

The Turing Machine

Notation for the Turing Machine

We need tools that will allow us to prove problem undecidable or intractable. The theory of undecidability / intractability are both based on a very simple model of a computer, called the Turing machine.



The Turing machine consists of a finite control, a tape divided into cells, and a tape head positioned at one of the tape cells.

- Initially, the input (finite-length string of symbols) is placed on the tape.
- All other tape cells, extending infinitely to the left and right, initially hold blank symbol.
- In one move, the Turing machine changes state, writes a tape symbol in the cell scanned, and moves the tape head left or right.

A Turing machine (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, in which

- Q is a finite set of states of the finite control,
- Σ is a finite set of input symbols,
- Γ is a complete set of tape symbols, $\Sigma \subset \Gamma$,
- δ is a transition function from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$,
- $q_0 \in Q$ is a start state,
- B is blank symbol being in Γ but not in Σ, and
- $F \subseteq Q$ is a set of final or accepting states.

The arguments of transition function δ are a state q and a tape symbol X. The value of $\delta(q, X)$, if it is defined, is a triple (p, Y, D), where:

1. $p \in Q$ is the next state,
2. $Y \in \Gamma$ is the tape symbol, written in the cell being scanned, replacing whatever symbol was there, and
3. D is a direction, either L or R, standing for “left” or “right”, respectively, and telling us the direction in which the head moves.

$\delta(q, X)$ may be undefined for some $q \in Q$ and $X \in \Gamma$.

Example

Let $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_2\})$, where δ is defined by

$$\delta(q_0, 0) = (q_0, 0, R), \qquad \delta(q_0, 1) = (q_1, 1, R),$$
$$\delta(q_1, 0) = (q_1, 0, R), \qquad \delta(q_1, B) = (q_2, B, R).$$

Turing machine M accepts the (0,1)-strings including one and only one 1.

The transition function can also be given by a table

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	–
q_1	$(q_1, 0, R)$	–	(q_2, B, R)
q_2	–	–	–

Instantaneous Descriptions for the Turing Machine

We use the string

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n$$

to represent an instantaneous description (ID) in which

- q is the state of the Turing machine,
- the tape head is scanning the ith symbol from the left, and
- $X_1 X_2 \cdots X_n$ is the portion of the tape between the leftmost to the rightmost nonblank.

Now we describe moves of a TM by the $\vdash$ notation.

Suppose $\delta(q, X_i) = (p, Y, L)$, then

$$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \underset{M}{\vdash} X_1X_2\cdots X_{i-2}pX_{i-1}YX_{i+1}\cdots X_n$$

There are two important exceptions:

1. If $i = 1$, then $qX_1X_2\cdots X_n \underset{M}{\vdash} pBYX_2\cdots X_n$.
2. If $i = n$ and $Y = B$, then $X_1X_2\cdots X_{n-1}qX_n \underset{M}{\vdash} X_1X_2\cdots X_{n-2}pX_{n-1}$.

Now suppose $\delta(q, X_i) = (p, Y, R)$, then

$$X_1X_2 \cdots X_{i-1}qX_iX_{i+1} \cdots X_n \underset{M}{\vdash} X_1X_2 \cdots X_{i-1}YpX_{i+1} \cdots X_n$$

Again, there are two important exceptions:

1. If $i = n$, then $X_1X_2 \cdots X_{n-1}qX_n \underset{M}{\vdash} X_1X_2 \cdots X_{n-1}YpB$.
2. If $i = 1$ and $Y = B$, then $qX_1X_2 \cdots X_n \underset{M}{\vdash} pX_2 \cdots X_n$.

As usual, $\overset{*}{\underset{M}{\vdash}}$, or just $\overset{*}{\vdash}$ when the TM M is understood, will be used to indicate zero, one, or more moves of the TM M.

Example

Consider $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_2\})$, where δ is defined by

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	–
q_1	$(q_1, 0, R)$	–	(q_2, B, R)
q_2	–	–	–

Let's see the moves of M on input 00100

$$q_0 00100 \vdash 0q_0 0100 \vdash 00q_0 100 \vdash 001q_1 00 \vdash 0010q_1 0$$
$$\vdash 00100q_1 B \vdash 00100Bq_2 B$$

We find that M accepts this string.

Here is the case of non-accepting computations by M.

- The ID sequence of moves of M on 00000

$$q_0 00000 \vdash 0q_0 0000 \vdash 00q_0 000 \vdash 000q_0 00 \vdash 0000q_0 0 \vdash 00000q_0 B \quad \text{dies}$$

- The ID sequence of moves of M on 00101

$$q_0 00101 \vdash 0q_0 0101 \vdash 00q_0 101 \vdash 001q_1 01 \vdash 0010q_1 1 \quad \text{dies}$$

Example

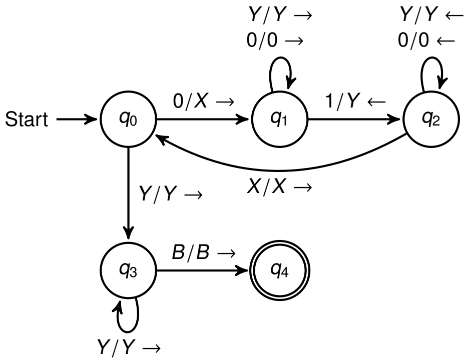
Design a Turing machine M accepting the language $\{0^n 1^n \mid n \geq 1\}$.

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

where δ is given by

δ	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

We can represent the transition of this TM by the following figure, much as we did for the PDA.



Here is an example of an accepting computation by M. Its input is 0011.

$$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1 \vdash q_2 X0Y1 \vdash Xq_0 0Y1 \vdash XXq_1 Y1$$
$$\vdash XXYq_1 1 \vdash XXq_2 YY \vdash Xq_2 XYY \vdash XXq_0 YY \vdash XXYq_3 Y$$
$$\vdash XXYYq_3 B \vdash XXYYBq_4 B$$

For another example, consider what M does on the input 0010.

$$q_0 0010 \vdash Xq_1 010 \vdash X0q_1 10 \vdash Xq_2 0Y0 \vdash q_2 X0Y0 \vdash Xq_0 0Y0 \vdash XXq_1 Y0$$
$$\vdash XXYq_1 0 \vdash XXY0q_1 B$$

M dies and does not accept its input.

The Language of a Turing Machine

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a Turing machine. The language of M is defined by

$$L(M) = \{w \mid w \in \Sigma^*, q_0 w \overset{*}{\vdash} \alpha p \beta \quad \text{for some } p \in F \text{ and } \alpha, \beta \in \Gamma^*\}$$

$L(M)$ is often called the recursively enumerable languages or RE languages.

Note that there is an important difference between TM and FA/PDA. The TM can decide whether it accepts the input before scanned all symbols in the input string!

Example

Let $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_3\})$ where δ is given by

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	–
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	–
q_2	$(q_2, 0, R)$	$(q_3, 1, R)$	–
q_3	–	–	–

Analyzing the moves of M, we can see

$$L(M) = \{w \mid w \in \{0, 1\}^* \text{that has at least three } 1's\}$$

For instance, $q_0 100110010010 \overset{*}{\vdash} 10011 q_3 0010010$

Turing Machine as a Computer of Functions

Turing machines could be not only as recognizers of languages, but also as computers of integer-valued functions.

In Turing's scheme, integers are represented in unary. We use 0^n represent any nonnegative integer n.

For an integer-valued function $f(n_1, n_2, \ldots, n_k)$, we use string $0^{n_1}10^{n_2}1\cdots10^{n_k}$ representing the values of its variables $n_1, n_2, \ldots, n_k$.

Turing machine might compute the function $f(n_1, n_2, \cdots, n_k)$. It will start with a tape consisting of $0^{n_1}10^{n_2}1\cdots10^{n_k}$ surrounded by blanks. If $f(n_1, n_2, \ldots, n_k) = m$, Turing machine halts with 0^m on its tape, surrounded by blanks.

An integer-valued function is called Turing computable, if there is a Turing machine M which can compute as this function.

The arithmetic operations such as addition, proper-subtraction, and multiplication on integers are all Turing computable.

Example

Design a Turing machine M, compute $n + m$ for any nonnegative integers n and m.

The input of M is 0^n10^m, the output should be 0^{n+m} when M halts.

- M scans symbols in the input string. After finding 1, replaces 1 by 0, then searches right until meets blank. M then return left, replaces the last 0 by a blank.
- There is an exception. When $n = 0$, what M does is just change 1 to blank

Now we give a formal description for desired Turing machine.

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_3\})$$

where δ is given by

δ	0	1	B
q_0	$(q_1, 0, R)$	(q_3, B, R)	–
q_1	$(q_1, 0, R)$	$(q_1, 0, R)$	(q_2, B, L)
q_2	(q_3, B, R)	–	–
q_3	–	–	–

e.g. $q_0000100 \vdash 0q_100100 \vdash 00q_10100 \vdash 000q_1100 \vdash 0000q_100 \vdash 00000q_10 \vdash 000000q_1B \vdash 00000q_20B \vdash 00000Bq_3B$.

Example

Design a Turing machine M, compute $m \dot{-} n = \max(m - n, 0)$ for any nonnegative integers m and n, e.g. $5 \dot{-} 3 = \max(5 - 3, 0) = 2$ and $3 \dot{-} 5 = \max(3 - 5, 0) = 0$.

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_6\})$$

where δ is given by

δ	0	1	B
q_0	(q_1, B, R)	(q_5, B, R)	–
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	–
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	(q_4, B, L)	$(q_6, 0, R)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
q_6	–	–	–

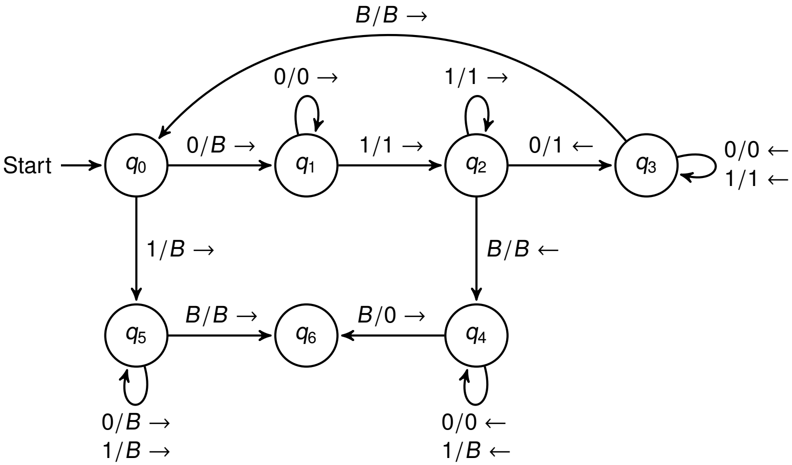
$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_6\})$$

M repeatedly finds its leftmost remaining 0 and replaces it by a blank. It then searches right, looking for a 1. After finding a 1, it continues right, until it comes to a 0, which it replaces by a 1.

M then returns left, seeking the leftmost 0, which it identifiers when ID first meets a blank and then moves one cell to the right.

The repetition ends if either:

- Searching right for a 0, M encounters a blank. Then the n 0's in 0^m10^n have all been changed to 1's, and $n+1$ of the m 0's have been changed to B. M replaces the $n+1$ 1's by $n+1$ B's, and moves to left, replaces first B by one 0, leaving $m-n$ 0's on the tape. Since $m \geq n$ in the case, $m-n = m \dot{-} n$.
- Beginning the cycle, M cannot find a 0 to change to a blank, because the first m 0's already have been changed to B. Then $m \leq n$, so $m \dot{-} n = 0$. M replaces all remaining 1's and 0's by B and ends with a completely blank tape.


$B/B \rightarrow$
$0/0 \rightarrow$
$1/1 \rightarrow$
Start
q_0
$0/B \rightarrow$
q_1
$1/1 \rightarrow$
q_2
$0/1 \leftarrow$
q_3
$0/0 \leftarrow$
$1/1 \leftarrow$
$1/B \rightarrow$
$B/B \leftarrow$
q_5
$B/B \rightarrow$
q_6
$B/0 \rightarrow$
q_4
$0/B \rightarrow$
$1/B \rightarrow$
$0/0 \leftarrow$
$1/B \leftarrow$

Homework

Exercises 8.2.1(b), 8.2.5(b)