Recall

- A language $L$ is recursively enumerable iff exists a Turing machine $M$ such that $L = L(M)$.

Definition

A language $L$ is undecidable iff $L$ is not recursive.
Undecidability

Recall

- A language $L$ is recursively enumerable iff exists a Turing machine $M$ such that $L = L(M)$.
- A language $L$ is recursive iff exists a Turing machine $M$ such that
  1) $L = L(M)$ and
  2) $M$ always halt (even if it does not accept).
Recall

- A language $L$ is recursively enumerable iff exists a Turing machine $M$ such that $L = L(M)$.
- A language $L$ is recursive iff exists a Turing machine $M$ such that
  i) $L = L(M)$ and
  ii) $M$ always halt (even if it does not accept).

Definition

A language $L$ is **undecidable** iff $L$ is not recursive.
Why “Recursive”? 

- The term “recursive” as synonym for “decidable” comes from Mathematics (prior to computers).
Why “Recursive”?

- The term “recursive” as synonym for “decidable” comes from Mathematics (prior to computers).
- Equivalent formalization to Turing-machine computability based on recursive functions.
Why “Recursive”?  

- The term “recursive” as synonym for “decidable” comes from Mathematics (prior to computers).
- Equivalent formalization to Turing-machine computability based on recursive functions.
- A function is recursive if only if it is Turing-machine computable (see, for example, [Boolos, Burges and Jeffrey 2007], [Hermes 1969] or [Kleene 1952]).
Why “Recursive”? 

- The term “recursive” as synonym for “decidable” comes from Mathematics (prior to computers).
- Equivalent formalization to Turing-machine computability based on recursive functions.
- A function is recursive if only if it is Turing-machine computable (see, for example, [Boolos, Burges and Jeffrey 2007], [Hermes 1969] or [Kleene 1952]).
- Recursive problem: “it is sufficiently simple that I can write a recursive function to solve it, and the function always finishes.” [Hopcroft, Motwani and Ullman 2007, p. 385]
Codification of Turing Machines

Convention

The Turing machine \( M \) is of the form:

\[
M = (\{q_1, \ldots, q_n\}, \{0, 1\}, \{X_1, X_2, X_3, \ldots, X_m\}, \delta, q_1, B, \{q_2\}),
\]

where \( X_1 = 0 \), \( X_2 = 1 \) and \( X_3 = B \). Moreover, \( D_1 = L \) and \( D_2 = R \).
Codification of Turing Machines

Convention

The Turing machine $M$ is of the form:

$$M = (\{q_1, \ldots, q_n\}, \{0, 1\}, \{X_1, X_2, X_3, \ldots, X_m\}, \delta, q_1, B, \{q_2\}),$$

where $X_1 = 0$, $X_2 = 1$ and $X_3 = B$. Moreover, $D_1 = L$ and $D_2 = R$.

Codification of an instruction

The instruction $\delta(q_i, X_j) = (q_k, X_l, D_m)$ is codified by

$$0^i10^j10^k10^l10^m.$$
Codification of Turing Machines

Codification of a Turing machine

Let $C_1, C_2, \ldots, C_p$ be the codifications of the instructions of a Turing machine $M$. The codification of $M$ is

$$
\overrightarrow{M} = C_1 11C_2 11 \ldots 11C_p.
$$
Codification of Turing Machines

Codification of a Turing machine

Let $C_1, C_2, \ldots, C_p$ be the codifications of the instructions of a Turing machine $M$. The codification of $M$ is

$$\overrightarrow{M} = C_111C_211 \ldots 11C_p.$$ 

Remark

Note that there are other possible codes for $M$. 
Codification of Turing Machines

A enumeration for the binary strings

We ordered the binary strings by [length-]lexicographical order (strings are ordered by length, and strings of equal length are ordered lexicographically).

If $w$ is a binary string, we call $w$ the $i$-th string where $1w$ is the binary integer $i$. We refer to the $i$-th string as $w_i$. 

Given a Turing machine $M$ with code $w_i$, we can now associate a natural number to it: $M$ is the $i$-th Turing machine, referred to as $M_i$.

Convention

If $w_i$ is not a valid Turing machine code, we shall take $M_i$ to be the Turing machine with one state and no transitions. Hence $L(M_i) = \emptyset$ if $w_i$ is not a valid Turing machine code.
Codification of Turing Machines

**A enumeration for the binary strings**

We ordered the binary strings by [length-]lexicographical order (strings are ordered by length, and strings of equal length are ordered lexicographically).

If $w$ is a binary string, we call $w$ the $i$-th string where $1w$ is the binary integer $i$. We refer to the $i$-th string as $w_i$.

**$i$-th Turing machine**

Given a Turing machine $M$ with code $w_i$, we can now associate a natural number to it: $M$ is the $i$-th Turing machine, referred to as $M_i$. 
A enumeration for the binary strings
We ordered the binary strings by [length-]lexicographical order (strings are ordered by length, and strings of equal length are ordered lexicographically).
If \( w \) is a binary string, we call \( w \) the \( i \)-th string where \( 1w \) is the binary integer \( i \). We refer to the \( i \)-th string as \( w_i \).

\( i \)-th Turing machine
Given a Turing machine \( M \) with code \( w_i \), we can now associate a natural number to it: \( M \) is the \( i \)-th Turing machine, referred to as \( M_i \).

Convention
If \( w_i \) is not a valid Turing machine code, we shall take \( M_i \) to be the Turing machine with one state and no transitions. Hence \( L(M_i) = \emptyset \) if \( w_i \) is not a valid Turing machine code.
The Diagonalization Language

The diagonalization language

A language that is not recursively enumerable.

\[ L_d = \{ w_i \mid w_i \notin L(M_i) \}. \]
## The Diagonalization Language

### The diagonalization language

A language that is not recursively enumerable.

\[ L_d = \{ w_i \mid w_i \notin L(M_i) \}. \]

<table>
<thead>
<tr>
<th>( M_i )</th>
<th>( w_j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

\[ a_{ij} = \begin{cases} 1, & \text{if } w_j \in L(M_i); \\ 0, & \text{if } w_j \notin L(M_i). \end{cases} \]

Vector for the language \( L(M_i) \): \( i \)-th row

\( L_d \): Complement of the diagonal

Is it possible that \( L_d \) be in a row?
The Diagonalization Language

Theorem (9.2)

$L_d$ is not recursively enumerable.

Proof

Whiteboard.
References


