CM0081 Formal Languages and Automata
§ 9.1 A Language That Is Not Recursively Enumerable

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2024-1

Preliminaries

Conventions

- The number and page numbers assigned to chapters, examples, exercises, figures, quotes, sections and theorems on these slides correspond to the numbers assigned in the textbook [Hopcroft, Motwani and Ullman (1979) 2007].
- The natural numbers include the zero, that is, $\mathbb{N} = \{0, 1, 2, \dots\}$.
- The power set of a set A, that is, the set of its subsets, is denoted by $\mathcal{P}A$.

Undecidability

Recall

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

Undecidability

Recall

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

Undecidability

Recall

- A language L is recursively enumerable iff exists a Turing machine M such that $L = \mathrm{L}(M)$.
- A language L is recursive iff exists a Turing machine M such that
  (i) $L = \mathrm{L}(M)$ 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, e.g. [Boolos, Burges and Jeffrey (1974) 2007], [Hermes (1961) 1969] or [Kleene (1952) 1974]).

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, e.g. [Boolos, Burges and Jeffrey (1974) 2007], [Hermes (1961) 1969] or [Kleene (1952) 1974]).
- Recursive problem: *‘it is sufficiently simple that I can write a recursive function to solve it, and the function always finishes.’ [p. 385]*

Codification of Turing Machines

Convention

The Turing machine M is of the form:

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

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

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m.$$

Codification of Turing Machines

Codification of a Turing machine

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

$$\overrightarrow{M} := C_1 11 C_2 11 \dots 11 C_p.$$

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 defined by

$$\overrightarrow{M} := C_1 11 C_2 11 \ldots 11 C_p.$$

Observation

Note that there are other possible codes for M.

Codification of Turing Machines

Enumeration of 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).

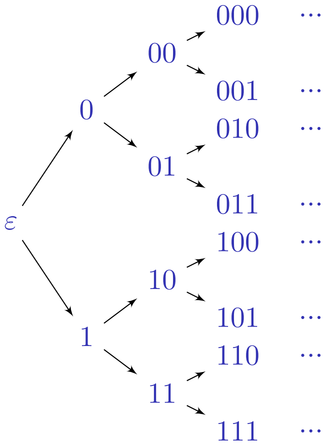
(continued on next slide)

Codification of Turing Machines

Enumeration of the binary strings (continuation)

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.

$$
\begin{aligned}
\varepsilon &\to 1_b &&\to 1,\\
0 &\to 10_b &&\to 2,\\
1 &\to 11_b &&\to 3,\\
00 &\to 100_b &&\to 4,\\
01 &\to 101_b &&\to 5,\\
10 &\to 110_b &&\to 6,\\
&\ \ \vdots
\end{aligned}
$$



Codification of Turing Machines

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.

Codification of Turing Machines

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, that is,

$$\mathrm{L}(M_i) = \emptyset.$$

Cantor's Diagonalisation Proof

Theorem

The open interval $(0, 1)$ is an uncountable (non-enumerable) set.

(continued on next slide)

Cantor's Diagonalisation Proof

Proof.

Let's suppose $(0,1)$ is (infinite) countable.

$$
\begin{aligned}
r_1 &= 0.d_{11}d_{12}d_{13}d_{14}\ldots \\
r_2 &= 0.d_{21}d_{22}d_{23}d_{24}\ldots \\
r_3 &= 0.d_{31}d_{32}d_{33}d_{34}\ldots \\
&\vdots
\end{aligned}
$$

Let $r = 0.d_1d_2d_3\ldots \in (0,1)$, where

$$
d_i = \begin{cases} 4, & \text{if } d_{ii} \neq 4; \\ 5, & \text{if } d_{ii} = 4. \end{cases}
$$

The number r does not belong to the above enumeration. Therefore the interval $(0,1)$ is an uncountable set. ∎

The Diagonalization Language

Definition

Let $\Sigma = \{0,1\}$. The **diagonalization language** is defined by

$$\mathrm{L_d} := \{\, w_i \in \Sigma^* \mid w_i \notin \mathrm{L}(M_i) \,\}.$$

The Diagonalization Language

Definition

Let $\Sigma = \{0, 1\}$. The **diagonalization language** is defined by

$$\mathrm{L_d} := \{\, w_i \in \Sigma^* \mid w_i \notin \mathrm{L}(M_i) \,\}.$$

		w_j 1	2	3	4	...
M_i	1	0	1	1	0	...
	2	1	1	0	1	...
	3	0	1	1	0	...
	4	1	1	0	0	...
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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

Language $\mathrm{L}(M_i)$'s vector: i-th row

$\mathrm{L_d}$: Complement of the diagonal

Is it possible that $\mathrm{L_d}$ be in a row?

The Diagonalization Language

Theorem 9.2

The language $\mathrm{L_d}$ is not recursively enumerable.

Proof by contradiction (proof of negation)

Whiteboard.

References

- Boolos, G. S., Burges, J. P. and Jeffrey, R. C. [1974] (2007). Computability and Logic. 5th ed. Cambridge University Press (cit. on pp. 6–9).
- Hermes, H. [1961] (1969). Enumerability · Decidability · Computability. Second revised edition. Translated G. T. Hermann and O. Plassmann. Springer-Verlag (cit. on pp. 6–9).
- Hopcroft, J. E., Motwani, R. and Ullman, J. D. [1979] (2007). Introduction to Automata Theory, Languages, and Computation. 3rd ed. Pearson Education (cit. on p. 2).
- Kleene, S. C. [1952] (1974). Introduction to Metamathematics. Seventh reprint. North-Holland (cit. on pp. 6–9).