CM0081 Automata and Formal Languages
§ 9.2 An Undecidable Problem That Is 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 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 $\mathcal{P}\,A$.

Theorems About Recursive Languages

Theorem 9.3

If L is a recursive language, then $\overline{L}$ is also a recursive language.

Proof

Whiteboard.

Theorems About Recursive Languages

Theorem 9.3
If L is a recursive language, then $\overline{L}$ is also a recursive language.

Proof
Whiteboard.

Theorem 9.4
If both L and $\overline{L}$ are recursively enumerable languages, then L is recursive (and $\overline{L}$ is recursive as well by Theorem 9.3).

Proof
Whiteboard.

Theorems About Recursive Languages

Possible relations between a language L and its complement $\overline{L}$

(i) Both L and $\overline{L}$ are recursive.

Theorems About Recursive Languages

Possible relations between a language L and its complement $\overline{L}$

(i) Both L and $\overline{L}$ are recursive.

(ii) Neither L nor $\overline{L}$ are recursively enumerable.

Theorems About Recursive Languages

Possible relations between a language L and its complement $\overline{L}$

(i) Both L and $\overline{L}$ are recursive.

(ii) Neither L nor $\overline{L}$ are recursively enumerable.

(iii) L is recursively enumerable but not recursive and $\overline{L}$ is not recursively enumerable.

Theorems About Recursive Languages

Possible relations between a language L and its complement $\overline{L}$

(i) Both L and $\overline{L}$ are recursive.

(ii) Neither L nor $\overline{L}$ are recursively enumerable.

(iii) L is recursively enumerable but not recursive and $\overline{L}$ is not recursively enumerable.

(iv) $\overline{L}$ is recursively enumerable but not recursive and L is not recursively enumerable.

Theorems About Recursive Languages

Exercise 9.2.5

Let L be recursively enumerable and let $\overline{L}$ be non recursively enumerable. Consider the language

$$L' = \{\, 0w \mid w \text{ is in } L \,\} \cup \{\, 1w \mid w \text{ is not in } L \,\}.$$

Can you say for certain whether L' is recursive, recursively enumerable, or non recursively enumerable? Justify your answer.

Solution (from Hopcroft, Motwani and Ullman [2007])

Suppose L' were recursively enumerable. Then we could design a Turing machine M for $\overline{L}$ as follows. Given input w, M changes its input to $1w$ and simulates the hypothetical Turing machine for L'. If that Turing machine accepts, then w is in $\overline{L}$, so M should accept. If the Turing machine for L' never accepts, then neither does M. Thus, M would accept exactly $\overline{L}$, which contradicts the fact that $\overline{L}$ is not recursively enumerable. We conclude that L' is not recursively enumerable.

The Universal Language

Conventions

1. (M, w): Represents the Turing machine M with input w.
2. w is a string of 0's and 1's.

Codification of a Turing machine with an input

Let w_i be the codification of a Turing machine M. The codification of (M, w) is defined by

$$\overrightarrow{(M, w)} := w_i 111 w.$$

The Universal Language

Definition

Let $\Sigma = \{0, 1\}$. The **universal language**, denoted $\mathrm{L_u}$, is the set of binary strings that encode a pair (M, w) such that $w \in \mathrm{L}(M)$, that is,

$$\mathrm{L_u} := \left\{\, \overrightarrow{(M, w)} \in \Sigma^* \;\middle|\; w \in \mathrm{L}(M) \,\right\}.$$

The Universal Language

Theorem

The language $\mathrm{L_u}$ is recursively enumerable.

Idea of the proof

There exists a Turing machine U such that $\mathrm{L_u} = \mathrm{L(U)}$. The machine U is called a **universal Turing machine**.

The Universal Language

Theorem 9.6

The language $\mathrm{L_u}$ is recursively enumerable but not recursive.

Proof of $\mathrm{L_u}$ is not recursive (by contradiction (proof of negation))

Suppose $\mathrm{L_u}$ is recursive

$\Rightarrow$ $\overline{\mathrm{L_u}}$ is recursive

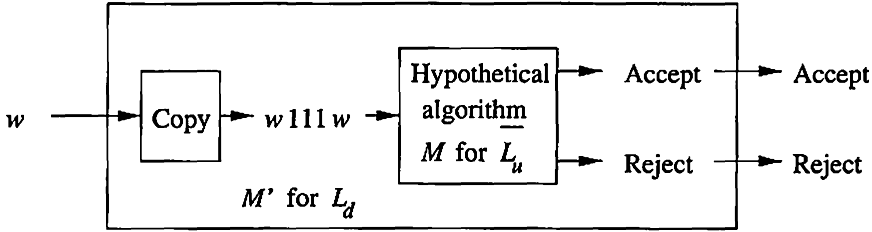
$\Rightarrow$ $\mathrm{L_d}$ is recursive (see next slide)

$\Rightarrow$ Contradiction because $\mathrm{L_d}$ is non recursively enumerable $\blacksquare$

The Universal Language

From the recursiveness of $\overline{\mathrm{L_u}}$ to the recursiveness of $\mathrm{L_d}$

Given a terminating Turing machine for accepting $\overline{\mathrm{L_u}}$ we could use this machine for building a terminating Turing machine for accepting $\mathrm{L_d}$.†



Since $\mathrm{L_d}$ is not recursive (because it is not recursive enumerable) then $\overline{\mathrm{L_u}}$ is not recursive.

† Figure from Hopcroft, Motwani and Ullman [2007, Fig. 9.6].

Code for a Universal Turing Machine

Code for U

Since U is a Turing machine exists i (1654 digits) such that $\mathrm{U} = M_i$ given by (using a different codification) [Penrose 1991, pp. 56-57]:

7244855335339317577198395039615711237952360672556559631108144796606505
0594042410903104836136323593656444434583822268832787676265561446928141
1771501784255170755408565768975334635694247848859704693472573998858228
3827795294683460521061169835945938791885546326440925525505820555989451
8907165374148960330967530204315536250349845298323206515830476641421307
0881932971723415105698026273468642992183817215733348282307345371342147
5059740345184372359593090640024321077342178851492760797597634415123079
5863963544922691594796546147113457001450481673375621725734645227310544
8298078496512698878896456976090663420447798902191443793283001949357096
3921703904833270882596201301773727202718625919914428275437422

Code for a Universal Turing Machine

235135567513408422229988937441053430547104436869587640517812801943753
081387063994277282315642528923751456544389905278079324114482614235728
619311833261065612275553181020751108533763380603108236167504563585216
421486954234718742643754442879006248582709124042207653875426445413345
174856629157429990950262300973373813772416217274772361020678685400289
356608569682262014198248621698902609130940298570600174300670086896759
034473417412787425581201549366393899690581773859165405535670409282133
222163141097871081459978669599704509681841906299443656015145490488092
208448003482249207730403043188429899393135266882349662101947161910701
461968523192847482034495897709553561107027581748733327296678998798473
284098190764851272631001740166787363477605857245036964434897992034489
997455662402937487668839751404451665707750060513883991668814072545544
665222050724262392379211525318162512536305093172863142200406457130527
5802307665183351995689139748137504926429605010013651980186945639498

Turing's Universal Turing Machine

- Based on M-functions (subroutines with parameters) [Sicard 1997; Copeland 2004b].
- The machine is composed by 12 symbols and 4.000 instructions, approximately [Sicard Ramírez 1998].

Small Universal Turing Machines

Notation

Let $\mathrm{UTM}(m, n)$ be the class of universal Turing machines with m states and n symbols.

Small Universal Turing Machines

Notation

Let $\mathrm{UTM}(m, n)$ be the class of universal Turing machines with m states and n symbols.

Theorem

If $\mathrm{UTM}(m, n) \neq \emptyset$ then [Shannon 1956]:

(i) $\mathrm{UTM}(2, n') \neq \emptyset$, where n' is at most $4mn + n$ and

(ii) $\mathrm{UTM}(m', 2) \neq \emptyset$, where $m' = (2^l - 1)m$ and l is the smaller integer such that $m \leq 2^l$.

Small Universal Turing Machines

Theorem

There exists universal Turing machines in the following classes [Rogozhin 1996; Neary and Woods 2012]:

$\mathrm{UTM}(m, n)$	Author(s)
$(24, 2)$	Rogozhin [1996]
$(19, 2)$	Baiocchi [2001]
$(18, 2)$	Neary and Woods [2007]
$(15, 2)$	Neary and Woods [2009]

(continued on next slide)

Small Universal Turing Machines

Theorem (continuation)

$\text{UTM}(m, n)$	Author(s)
$(10, 3)$	Rogozhin [1996]
$(9, 3)$	Neary and Woods [2009]
$(7, 4)$	Rogozhin [1996]
$(6, 4)$	Neary and Woods [2009]
$(5, 5)$	Rogozhin [1996]
$(4, 6)$	Rogozhin [1996]
$(3, 10)$	Rogozhin [1996]
$(2, 18)$	Rogozhin [1996]

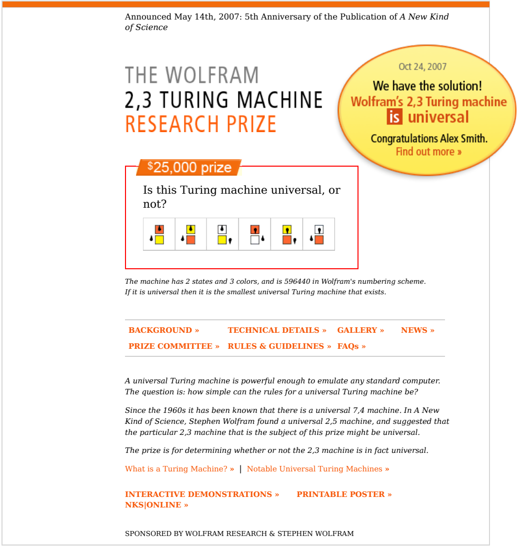
Small Universal Turing Machines

Theorem

The following classes are empty [Rogozhin 1996; Neary and Woods 2012]:

$\mathrm{UTM}(m, n)$	Author(s)
$(m, 1)$	trivial
$(3, 2)$	Rogozhin [1996]
$(2, 3)$	Rogozhin [1996]
$(2, 2)$	Rogozhin [1996]
$(1, n)$	Herman [1968]

Wolfram Turing Machine



Announced May 14th, 2007: 5th Anniversary of the Publication of *A New Kind of Science*

THE WOLFRAM
2,3 TURING MACHINE
RESEARCH PRIZE

Oct 24, 2007
We have the solution!
Wolfram's 2,3 Turing machine is universal
Congratulations Alex Smith.
Find out more »

\$25,000 prize

Is this Turing machine universal, or not?

The machine has 2 states and 3 colors, and is 596440 in Wolfram's numbering scheme.
If it is universal then it is the smallest universal Turing machine that exists.

BACKGROUND » TECHNICAL DETAILS » GALLERY » NEWS »
PRIZE COMMITTEE » RULES & GUIDELINES » FAQs »

A universal Turing machine is powerful enough to emulate any standard computer. The question is: how simple can the rules for a universal Turing machine be?

Since the 1960s it has been known that there is a universal 7,4 machine. In A New Kind of Science, Stephen Wolfram found a universal 2,5 machine, and suggested that the particular 2,3 machine that is the subject of this prize might be universal.

The prize is for determining whether or not the 2,3 machine is in fact universal.

What is a Turing Machine? » | Notable Universal Turing Machines »

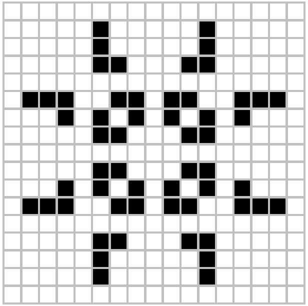
INTERACTIVE DEMONSTRATIONS » PRINTABLE POSTER »
NKS|ONLINE »

SPONSORED BY WOLFRAM RESEARCH & STEPHEN WOLFRAM

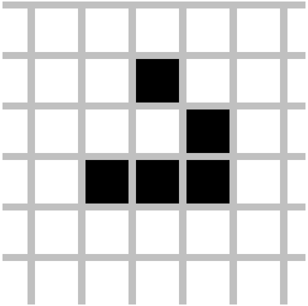
Conway's Game of Life

Examples

(From Wikipedia)



Pulsar (Oscillator)
Author: Jokey Smurf



Glider (Spaceship)
Author: Rodrigo Silveira Camargo

Conway's Game of Life

Rules

(i) Any live cell with fewer than two live neighbours dies, as if caused by under-population.

Conway's Game of Life

Rules

(i) Any live cell with fewer than two live neighbours dies, as if caused by under-population.

(ii) Any live cell with two or three live neighbours lives on to the next generation.

Conway's Game of Life

Rules

(i) Any live cell with fewer than two live neighbours dies, as if caused by under-population.

(ii) Any live cell with two or three live neighbours lives on to the next generation.

(iii) Any live cell with more than three live neighbours dies, as if by overcrowding.

Conway's Game of Life

Rules

(i) Any live cell with fewer than two live neighbours dies, as if caused by under-population.
(ii) Any live cell with two or three live neighbours lives on to the next generation.
(iii) Any live cell with more than three live neighbours dies, as if by overcrowding.
(iv) Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Conway's Game of Life

Rules

(i) Any live cell with fewer than two live neighbours dies, as if caused by under-population.
(ii) Any live cell with two or three live neighbours lives on to the next generation.
(iii) Any live cell with more than three live neighbours dies, as if by overcrowding.
(iv) Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Theorem

It is possible codified a universal Turing machine in Conway's Game of Life [Rendell 2011].

The Halting Problem

The language of the halting problem

Let $\Sigma = \{0, 1\}$. The original Turing machine accepted by halting, no by final state.

$$\mathrm{H}(M) := \{\, w \in \Sigma^* \mid M \text{ halts given the input } w \,\}.$$

The language of the halting problem

Let $\Sigma = \{0, 1\}$. The original Turing machine accepted by halting, no by final state.

$$\mathrm{H}(M) := \{\, w \in \Sigma^* \mid M \text{ halts given the input } w \,\}.$$

We define the language of the halting problem by

$$\mathrm{L}_{\mathrm{hp}} := \left\{\, \overrightarrow{(M, w)} \in \Sigma^* \;\middle|\; w \in \mathrm{H}(M) \,\right\}.$$

The Halting Problem

The language of the halting problem

Let $\Sigma = \{0, 1\}$. The original Turing machine accepted by halting, no by final state.

$$\mathrm{H}(M) := \{\, w \in \Sigma^* \mid M \text{ halts given the input } w \,\}.$$

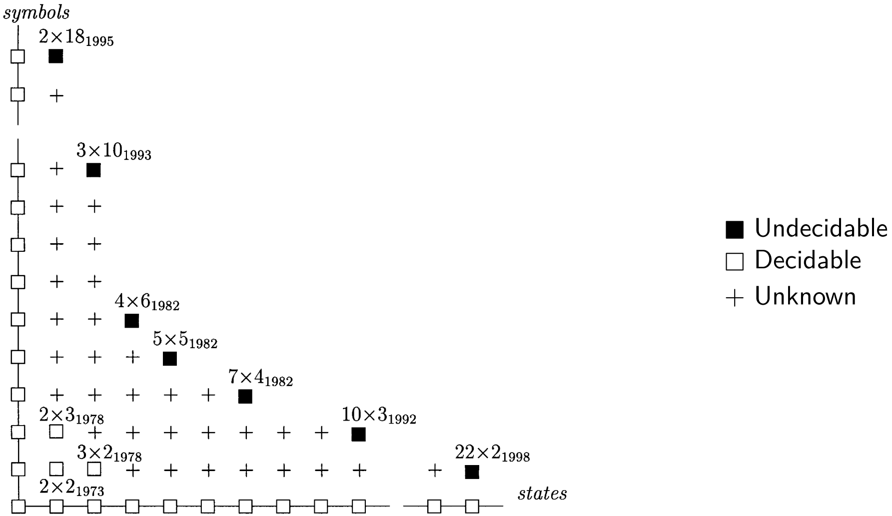
We define the language of the halting problem by

$$\mathrm{L}_{\mathrm{hp}} := \left\{\, \overrightarrow{(M, w)} \in \Sigma^* \;\middle|\; w \in \mathrm{H}(M) \,\right\}.$$

Exercise 9.2.1

Show that L_{hp} is recursively enumerable but not recursive.

The Halting Problem: State of Art[†]



[†] Figure from [Margenstern 2000].

The Halting Problem

Observation

The halting problem was introduced and named by Davis [1958, p. 70] not by Turing himself, contrary to popular belief [Copeland 2004a, p. 40].

The Halting Problem

Observation

The halting problem was introduced and named by Davis [1958, p. 70] not by Turing himself, contrary to popular belief [Copeland 2004a, p. 40].

Practical approach

'In contrast to popular belief, proving termination is not always impossible.' [Cook, Podelski and Rybalchenko 2011, p. 1]

References

Baiocchi, C. (2001). Three Small Universal Turing Machines. In: Machines, Computations, and Universality (MCU 2001). Ed. by Margenstern, M. and Rogozhin, Y. Vol. 2055. Lecture Notes in Computer Science. Springer, pp. 1–10. DOI: 10.1007/3-540-45132-3_1 (cit. on p. 20).

Cook, B., Podelski, A. and Rybalchenko, A. (2011). Proving Program Termination. Communications of the ACM 54.5, pp. 88–98. DOI: 10.1145/1941487.1941509 (cit. on pp. 34, 35).

Copeland, B. J. (2004a). Hypercomputation: Philosophical Issues. Theoretical Computer Science 317.1–3, pp. 251–267. DOI: 10.1016/j.tcs.2003.12.014 (cit. on pp. 34, 35).

Davis, M. (1958). Computability and Unsolvability. McGraw-Hill (cit. on pp. 34, 35).

Herman, G. T. (1968). The Uniform Halting Problem for Generalized One State Turing Machines. In: 9th Annual Symposium on Switching and Automata Theory (SWAT 1968). IEEE, pp. 368–372. DOI: 10.1109/SWAT.1968.36 (cit. on p. 22).

Hopcroft, J. E., Motwani, R. and Ullman, J. D. [1979] (2007). Introduction to Automata Theory, Languages, and Computation. 3rd ed. Pearson Education (cit. on pp. 2, 9, 14).

References

Margenstern, M. (2000). Frontier Between Decidability and Undecidability: A Survey. Theoretical Computer Science 231.2, pp. 217–251. DOI: 10.1016/S0304-3975(99)00102-4 (cit. on p. 33).

Neary, T. and Woods, D. (2007). Four Small Turing Machines. In: Machines, Computations, and Universality (MCU 2007). Ed. by Durand-Lose, J. and Margenstern, M. Vol. 4664. Lecture Notes in Computer Science. Springer, pp. 242–254. DOI: 10.1007/978-3-540-74593-8_21 (cit. on p. 20).

— (2009). Four Small Turing Machines. Fundamenta Informaticae 91.1, pp. 123–144. DOI: 10.3233/FI-2009-0036 (cit. on pp. 20, 21).

— (2012). The Complexity of Small Universal Turing Machines: A Survey. In: Theory and Practice of Computer Science (SOFSEM 2012). Ed. by Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S. and Turán, G. Vol. 7147. Lecture Notes in Computer Science. Springer, pp. 385–405. DOI: 10.1007/978-3-642-27660-6_32 (cit. on pp. 20, 22).

Penrose, R. (1991). The Emperor's New Mind. Penguin Books (cit. on p. 15).

References

Rendell, P. (2011). A Universal Turing Machine in Conway's Game of Life. In: Proceedings of the 2011 International Conference on High Performance Computing & Simulation (HPCS 2011). Ed. by Smari, W. W. IEEE, pp. 764–772. DOI: 10.1109/HPCSim.2011.5999906 (cit. on pp. 25–29).

Rogozhin, Y. (1996). Small Universal Turing Machines. Theoretical Computer Science 168, pp. 215–240. DOI: 10.1016/S0304-3975(96)00077-1 (cit. on pp. 20–22).

Shannon, C. E. (1956). A Universal Turing Machine with Two Internal States. In: Automata Studies. Ed. by Shannon, C. E. and McCarthy, J. Vol. 34. Annals of Mathematics Studies. Princenton University Press, pp. 157–165 (cit. on pp. 18, 19).

Sicard, A. (1997). Máquina Universal de Turing: Algunas Indicaciones para su Construcción. Revista Universidad EAFIT 33.108, pp. 61–106 (cit. on p. 17).

Sicard, A. and Copeland, B. J. (2004b). Appendix: Subroutines and M-functions. In: The Essential Turing. Vol. 317. 1–3, pp. 54–57. DOI: 10.1016/j.tcs.2003.12.014 (cit. on p. 17).

References

Sicard Ramírez, A. (1998). Máquinas de Turing Dinámicas: Historia y Desarrollo de una Idea. MA thesis. Departamento de Informática y Sistemas. Universidad EAFIT (cit. on p. 17).