CM0081 Automata and Formal Languages
§ 8.1 Problems That Computers Cannot Solve

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 $\mathcal{P}\,A$.

Undecidable Problems

Recall

Given $L \subseteq \Sigma^*$ and a word $w \in \Sigma^*$, to recall that to decide whether or not $w \in L$ is a (decision) problem on L.

Undecidable Problems

Recall

Given $L \subseteq \Sigma^*$ and a word $w \in \Sigma^*$, to recall that to decide whether or not $w \in L$ is a (decision) problem on L.

Definition (informally)

A problem is **undecidable** iff no algorithm can solve it.

Question

Are there undecidable problems?

Undecidable Problems

Theorem

There are undecidable problems.

Proof (by cardinality)

Let Σ be an alphabet and let $\overline{\overline{S}}$ be the cardinality of the set S.

(i) Since Σ is an alphabet, then $\overline{\overline{\Sigma}} = n$, with $n \in \mathbb{Z}^+$, that is, the alphabet Σ is an enumerable set.

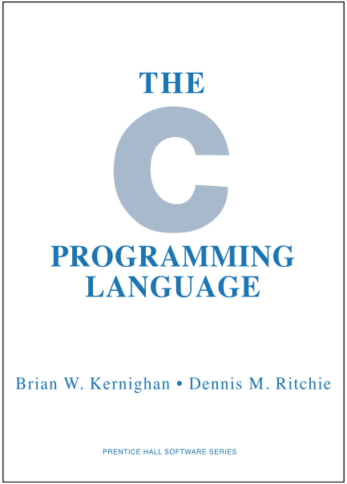
(ii) The set of all the strings over Σ, that is Σ^*, is an enumerable set (lexicographical order).

(iii) The set of all the languages over Σ, that is, the set $\{\, L \mid L \subseteq \Sigma^* \,\}$ is infinite not enumerable set.

(iv) The set $\{\, P \mid P$ is a program in a general programming language $\}$ is an enumerable set (lexicographical order).

Hence, there are more languages over Σ than programs. In consequence, there must be languages whose decision problems are undecidable. ■

The `hello-world` Problem: An Undecidable Problem



'The first program to write is the same for all languages: Print the words hello, world.' [1978, §1.1]

The `hello-world` Problem: An Undecidable Problem

Problem description

To determine whether a given program, with a given input, prints hello, world as the first 12 characters that it prints.

The `hello-world` Problem: An Undecidable Problem

Problem description

To determine whether a given program, with a given input, prints hello, world as the first 12 characters that it prints.

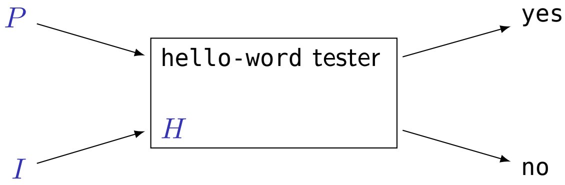
Theorem

The `hello-world` problem is an undecidable problem.

The hello-world Problem: An Undecidable Problem

Informal proof

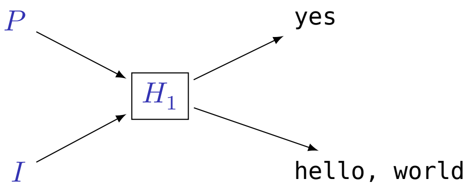
1. We assume that the following program H exists:



(continued on next slide)

The `hello-world` Problem: An Undecidable Problem

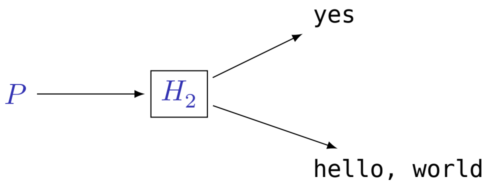
2. From the program H build the program H_1 with the following behaviour:



(continued on next slide)

The `hello-world` Problem: An Undecidable Problem

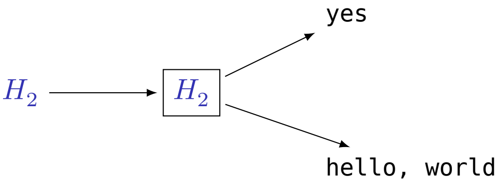
3. To build the program H_2 by restricting H_1 to take only the input P (i.e. P is also the input of the program P):



(continued on next slide)

The hello-world Problem: An Undecidable Problem

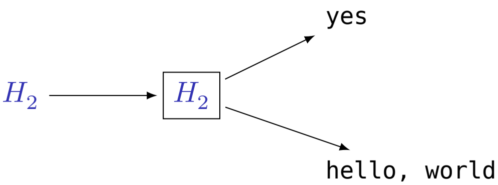
4. What does the program H_2 do when given itself as input?



Analysis on the whiteboard.

The `hello-world` Problem: An Undecidable Problem

4. What does the program H_2 do when given itself as input?



   Analysis on the whiteboard.

   Therefore the program H_2 can not exist.

5. Therefore the program H can not exists, i.e. the `hello-world` problem is undecidable. ■

Computability: Historical Remarks

1900 David Hilbert's problems
Wir müssen wissen — wir werden wissen!
(We must know — we will know!)

Computability: Historical Remarks

1900 David Hilbert's problems
Wir müssen wissen — wir werden wissen!
(We must know — we will know!)

1931 Kurt Gödel's incompleteness theorems

Computability: Historical Remarks

1900 David Hilbert's problems
Wir müssen wissen — wir werden wissen!
(We must know — we will know!)

1931 Kurt Gödel's incompleteness theorems

1934-1937 Confluence of ideas

- Derivability from a system of equations (Kurt Gödel, Jacques Herbrand)
- Lambda calculus (Alonzo Church, Stephen C. Kleene, J. Barkley Rosser)
- Recursive functions (Kurt Gödel, Stephen C. Kleene)
- Post machines (Emil L. Post)
- Turing machines (Alan M. Turing)

(continued on next slide)

Computability: Historical Remarks

1936–40 The Church-Turing-Kleene thesis: A (number-theoretic) function is effectively calculable if and only if there is a Turing machine which computes it.

Computability: Historical Remarks

1936–40 The Church-Turing-Kleene thesis: A (number-theoretic) function is effectively calculable if and only if there is a Turing machine which computes it.

1985 Deutsch's quantum Turing machines

Computability: Historical Remarks

- **1936–40** The Church-Turing-Kleene thesis: A (number-theoretic) function is effectively calculable if and only if there is a Turing machine which computes it.
- **1985** Deutsch's quantum Turing machines
- **1940–today** Many equivalents models of computation

Problems Reduction

How to prove that a problem is undecidable?

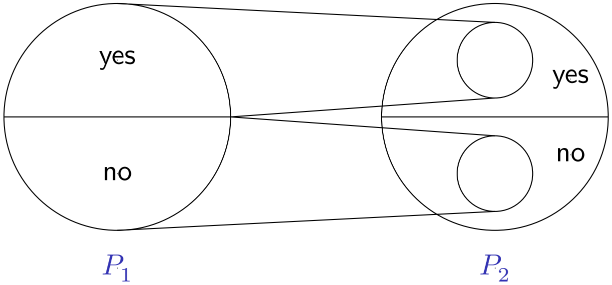
- Proof by contradiction (or constructively speaking, proof of negation [Bauer 2017])

Problems Reduction

How to prove that a problem is undecidable?

- Proof by contradiction (or constructively speaking, proof of negation [Bauer 2017])
- Problem reduction

  Reduction from a problem P_1 to a problem P_2:



Problems Reduction

Theorem

If P_1 is undecidable and there is a reduction of P_1 to P_2, then P_2 is undecidable too.

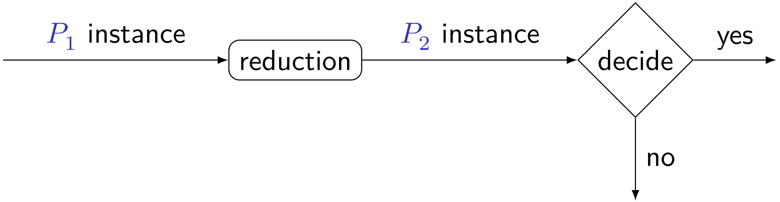
Problems Reduction

Theorem

If P_1 is undecidable and there is a reduction of P_1 to P_2, then P_2 is undecidable too.

Proof by contradiction

Let's suppose P_2 decidable. The reduction from P_1 to P_2 implies that P_1 is decidable which is a contradiction. Therefore, P_2 is undecidable.



■

Problems Reduction

Example (the `calls-foo` problem)

Problem description: Does a program Q with an input z calls the function `foo`?

Problems Reduction

Example (the `calls-foo` problem)

Problem description: Does a program Q with an input z calls the function `foo`?

Prove that the `calls-foo` problem is undecidable.

(continued on next slide)

Problems Reduction

Proof

Idea: Reduction of the `hello-world` problem to the `calls-foo` problem.

Problems Reduction

Proof

Idea: Reduction of the `hello-world` problem to the `calls-foo` problem.

That is, a program P with input y prints `hello, world` iff a program Q with input z calls the function `foo`.

Problems Reduction

Proof

Idea: Reduction of the `hello-world` problem to the `calls-foo` problem.

That is, a program P with input y prints `hello, world` iff a program Q with input z calls the function `foo`.

Reduction:

1. Program P_1: Rename `foo` in the program P.
2. Program P_2: Add the function `foo` to the program P_1.
3. Program P_3: Save the first 12 characters that prints the program P_2.
4. Program P_4: When the program P_3 executes an output statement if output is `hello, world` then calls the function `foo`.
5. $Q = P_4$ and $y = z$. ∎

Problems Reduction

Exercise 8.1.1.a (the halting problem)

Problem description: Given a program and an input, does the program eventually halt; i.e. does the program not loop forever on the input?

Problems Reduction

Exercise 8.1.1.a (the halting problem)

Problem description: Given a program and an input, does the program eventually halt; i.e. does the program not loop forever on the input?

Prove that the halting problem is undecidable.

(continued on next slide)

Problems Reduction

Proof (based on the solution in Hopcroft, Motwani and Ullman [2007])

Idea: Reduction of the `hello-world` problem to the halting problem.

Problems Reduction

Proof (based on the solution in Hopcroft, Motwani and Ullman [2007])

Idea: Reduction of the `hello-world` problem to the halting problem.

That is, a program P with input y prints `hello, world` iff a program Q with input z halts.

Problems Reduction

Proof (based on the solution in Hopcroft, Motwani and Ullman [2007])

Idea: Reduction of the `hello-world` problem to the halting problem.

That is, a program P with input y prints `hello, world` iff a program Q with input z halts.

Reduction:

1. Program P_1: Add an infinite loop (e.g. `while(1);`) to the end of `main()`.
2. Program P_2: Save the first 12 characters that prints the program P_1.
3. Program P_3: When the program P_2 executes an output statement if output is `hello, world` then the program P_3 halts by going to the end of `main`.
4. $Q = P_3$ and $y = z$. ∎

Problems Reduction

Theorem

If P_1 is undecidable and there is a reduction of P_1 to P_2, then P_2 is undecidable too.

Be careful

In the above theorem the reduction is from P_1 to P_2, it is not from P_2 to P_1:

Problems Reduction

Theorem

If P_1 is undecidable and there is a reduction of P_1 to P_2, then P_2 is undecidable too.

Be careful

In the above theorem the reduction is from P_1 to P_2, it is not from P_2 to P_1:

- From a reduction of P_1 to P_2:

$$P_2 \text{ decidable} \Rightarrow P_1 \text{ decidable}$$
$$P_1 \text{ undecidable} \Rightarrow P_2 \text{ undecidable}$$

Problems Reduction

Theorem

If P_1 is undecidable and there is a reduction of P_1 to P_2, then P_2 is undecidable too.

Be careful

In the above theorem the reduction is from P_1 to P_2, it is not from P_2 to P_1:

- From a reduction of P_1 to P_2:

$$P_2 \text{ decidable} \Rightarrow P_1 \text{ decidable}$$
$$P_1 \text{ undecidable} \Rightarrow P_2 \text{ undecidable}$$

- From a reduction of P_2 to P_1:

$$P_1 \text{ decidable} \Rightarrow P_2 \text{ decidable} \qquad \text{(hypothesis false)}$$
$$P_2 \text{ undecidable} \Rightarrow P_1 \text{ undecidable} \qquad \text{(hypothesis unknown)}$$

References

Bauer, A. (2017). Five States of Accepting Constructive Mathematics. Bulletin of the American Mathematical Society 54.3, pp. 481–498. DOI: 10.1090/bull/1556 (cit. on pp. 20, 21).

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, 31–33).