

# Theory of Computer Science

## D8. Rice's Theorem and Other Undecidable Problems

Malte Helmert

University of Basel

May 15, 2017

# Theory of Computer Science

## May 15, 2017 — D8. Rice's Theorem and Other Undecidable Problems

D8.1 Other Halting Problem Variants

D8.2 Rice's Theorem

D8.3 Outlook: Further Undecidable Problems

D8.4 Summary

## Overview: Computability Theory

### Computability Theory

- ▶ imperative models of computation:
  - D1. Turing-Computability
  - D2. LOOP- and WHILE-Computability
  - D3. GOTO-Computability
- ▶ functional models of computation:
  - D4. Primitive Recursion and  $\mu$ -Recursion
  - D5. Primitive/ $\mu$ -Recursion vs. LOOP-/WHILE-Computability
- ▶ undecidable problems:
  - D6. Decidability and Semi-Decidability
  - D7. Halting Problem and Reductions
  - D8. **Rice's Theorem and Other Undecidable Problems**
    - ~~Post's Correspondence Problem~~
    - ~~Undecidable Grammar Problems~~
    - ~~Gödel's Theorem and Diophantine Equations~~

## Further Reading (German)

### Literature for this Chapter (German)

Theoretische Informatik – kurz gefasst  
by Uwe Schöning (5th edition)

- ▶ **Chapter 2.6**
- ▶ Outlook: **Chapters 2.7–2.9**



## Further Reading (English)

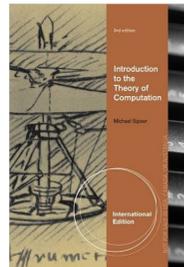
### Literature for this Chapter (English)

Introduction to the Theory of Computation  
by Michael Sipser (3rd edition)

- ▶ Chapters 5.1 and 5.3
- ▶ Outlook: Chapters 5.2 and 6.2

Notes:

- ▶ Sipser's definitions differ from ours.



## D8.1 Other Halting Problem Variants

## General Halting Problem (1)

### Definition (General Halting Problem)

The **general halting problem** or **halting problem** is the language

$$H = \{w\#x \in \{0, 1, \#\}^* \mid w, x \in \{0, 1\}^*, \\ M_w \text{ started on } x \text{ terminates}\}$$

**German:** allgemeines Halteproblem, Halteproblem

**Note:**  $H$  is semi-decidable. (Why?)

### Theorem (Undecidability of General Halting Problem)

*The general halting problem is undecidable.*

**Intuition:** if the special case  $K$  is not decidable,  
then the more general problem  $H$  definitely cannot be decidable.

## General Halting Problem (2)

**Proof.**

We show  $K \leq H$  (Reminder:  $K$  is the special halting problem).

We define  $f : \{0, 1\}^* \rightarrow \{0, 1, \#\}^*$  as  $f(w) := w\#w$ .

$f$  is clearly total and computable, and

$$\begin{aligned} w \in K \\ \text{iff } M_w \text{ started on } w \text{ terminates} \\ \text{iff } w\#w \in H \\ \text{iff } f(w) \in H. \end{aligned}$$

Therefore  $f$  is a reduction from  $K$  to  $H$ .

Because  $K$  is undecidable,  $H$  is also undecidable.  $\square$

## Halting Problem on Empty Tape (1)

### Definition (Halting Problem on the Empty Tape)

The **halting problem on the empty tape** is the language

$$H_0 = \{w \in \{0, 1\}^* \mid M_w \text{ started on } \varepsilon \text{ terminates}\}.$$

German: Halteproblem auf leerem Band

Note:  $H_0$  is semi-decidable. (Why?)

### Theorem (Undecidability of Halting Problem on Empty Tape)

*The halting problem on the empty tape is undecidable.*

## Halting Problem on Empty Tape (2)

### Proof.

We show  $H \leq H_0$ .

Consider the function  $f : \{0, 1, \#\}^* \rightarrow \{0, 1\}^*$  that computes the word  $f(z)$  for a given  $z \in \{0, 1, \#\}^*$  as follows:

- ▶ Test if  $z$  has the form  $w\#x$  with  $w, x \in \{0, 1\}^*$ .
- ▶ If not, return any word that is not in  $H_0$  (e. g., encoding of a TM that instantly starts an endless loop).
- ▶ If yes, split  $z$  into  $w$  and  $x$ .
- ▶ Decode  $w$  to a TM  $M_2$ .

...

## Halting Problem on Empty Tape (3)

### Proof (continued).

- ▶ Construct a TM  $M_1$  that behaves as follows:
  - ▶ If the input is empty: write  $x$  onto the tape and move the head to the first symbol of  $x$  (if  $x \neq \varepsilon$ ); then stop
  - ▶ otherwise, stop immediately
- ▶ Construct TM  $M$  that first runs  $M_1$  and then  $M_2$ .
- ▶ Return the encoding of  $M$ .

$f$  is total and (with some effort) computable. Also:

$$\begin{aligned} z \in H &\text{ iff } z = w\#x \text{ and } M_w \text{ run on } x \text{ terminates} \\ &\text{ iff } M_{f(z)} \text{ started on empty tape terminates} \\ &\text{ iff } f(z) \in H_0 \end{aligned}$$

$$\rightsquigarrow H \leq H_0 \rightsquigarrow H_0 \text{ undecidable} \quad \square$$

## D8.2 Rice's Theorem

## Rice's Theorem (1)

- ▶ We have shown that a number of (related) problems are undecidable:
  - ▶ special halting problem  $K$
  - ▶ general halting problem  $H$
  - ▶ halting problem on empty tape  $H_0$
- ▶ Many more results of this type could be shown.
- ▶ Instead, we prove a much more general result, **Rice's theorem**, which shows that a very large class of different problems are undecidable.
- ▶ Rice's theorem can be summarized informally as: **every** question about **what** a given Turing machine computes is undecidable.

## Rice's Theorem (2)

### Theorem (Rice's Theorem)

Let  $\mathcal{R}$  be the class of all computable functions.

Let  $\mathcal{S}$  be an **arbitrary** subset of  $\mathcal{R}$  except  $\mathcal{S} = \emptyset$  or  $\mathcal{S} = \mathcal{R}$ .

Then the language

$$C(\mathcal{S}) = \{w \in \{0, 1\}^* \mid \text{the function computed by } M_w \text{ is in } \mathcal{S}\}$$

is undecidable.

German: Satz von Rice

Question: why the restriction to  $\mathcal{S} \neq \emptyset$  and  $\mathcal{S} \neq \mathcal{R}$ ?

Extension (without proof): in most cases neither  $C(\mathcal{S})$  nor  $\overline{C(\mathcal{S})}$  is semi-decidable. (But there are sets  $\mathcal{S}$  for which one of the two languages is semi-decidable.)

## Rice's Theorem (3)

Proof.

Let  $\Omega$  be the function that is undefined everywhere.

Case distinction:

Case 1:  $\Omega \in \mathcal{S}$

Let  $q \in \mathcal{R} \setminus \mathcal{S}$  be an arbitrary computable function outside of  $\mathcal{S}$  (exists because  $\mathcal{S} \subseteq \mathcal{R}$  and  $\mathcal{S} \neq \mathcal{R}$ ).

Let  $Q$  be a Turing machine that computes  $q$ .

...

## Rice's Theorem (4)

Proof (continued).

Consider function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ , where  $f(w)$  is defined as follows:

- ▶ Construct TM  $M$  that first behaves on input  $y$  like  $M_w$  on the empty tape (independently of what  $y$  is).
- ▶ Afterwards (if that computation terminates!)  $M$  clears the tape, creates the start configuration of  $Q$  for input  $y$  and then simulates  $Q$ .
- ▶  $f(w)$  is the encoding of this TM  $M$

$f$  is total and computable.

...

## Rice's Theorem (5)

Proof (continued).

Which function is computed by the TM encoded by  $f(w)$ ?

$$M_{f(w)} \text{ computes } \begin{cases} \Omega & \text{if } M_w \text{ does not terminate on } \varepsilon \\ q & \text{otherwise} \end{cases}$$

For all words  $w \in \{0, 1\}^*$ :

$$\begin{aligned} w \in H_0 &\implies M_w \text{ terminates on } \varepsilon \\ &\implies M_{f(w)} \text{ computes the function } q \\ &\implies \text{the function computed by } M_{f(w)} \text{ is not in } \mathcal{S} \\ &\implies f(w) \notin C(\mathcal{S}) \end{aligned}$$

...

## Rice's Theorem (6)

Proof (continued).

Further:

$$\begin{aligned} w \notin H_0 &\implies M_w \text{ does not terminate on } \varepsilon \\ &\implies M_{f(w)} \text{ computes the function } \Omega \\ &\implies \text{the function computed by } M_{f(w)} \text{ is in } \mathcal{S} \\ &\implies f(w) \in C(\mathcal{S}) \end{aligned}$$

Together this means:  $w \notin H_0$  iff  $f(w) \in C(\mathcal{S})$ ,  
thus  $w \in \bar{H}_0$  iff  $f(w) \in C(\mathcal{S})$ .

Therefore,  $f$  is a reduction of  $\bar{H}_0$  to  $C(\mathcal{S})$ .

Since  $H_0$  is undecidable,  $\bar{H}_0$  is also undecidable.

We can conclude that  $C(\mathcal{S})$  is undecidable.

...

## Rice's Theorem (7)

Proof (continued).

Case 2:  $\Omega \notin \mathcal{S}$

Analogous to Case 1 but this time choose  $q \in \mathcal{S}$ .

The corresponding function  $f$  then reduces  $H_0$  to  $C(\mathcal{S})$ .

Thus, it also follows in this case that  $C(\mathcal{S})$  is undecidable.  $\square$

## Rice's Theorem: Consequences

Was it worth it?

We can now know conclude immediately that (for example) the following informally specified problems are all undecidable:

- ▶ Does a given TM compute a constant function?
- ▶ Does a given TM compute a total function (i. e. will it always terminate, and in particular terminate in a "correct" configuration)?
- ▶ Is the output of a given TM always longer than its input?
- ▶ Does a given TM compute the identity function?
- ▶ Does a given TM compute the computable function  $f$ ?
- ▶ ...

## Rice's Theorem: Practical Applications

Practical problems that are undecidable due to Rice's theorem:

- ▶ **automated debugging:**
  - ▶ Can a given variable ever receive a null value?
  - ▶ Can a given assertion in a program ever trigger?
  - ▶ Can a given buffer ever overflow?
- ▶ **virus scanners and other software security analysis:**
  - ▶ Can this code do something harmful?
  - ▶ Is this program vulnerable to SQL injections?
  - ▶ Can this program lead to a privilege escalation?
- ▶ **optimizing compilers:**
  - ▶ Is this dead code?
  - ▶ Is this a constant expression?
  - ▶ Can pointer aliasing happen here?
  - ▶ Is it safe to parallelize this code path?
- ▶ **parallel program analysis:**
  - ▶ Is a deadlock possible here?
  - ▶ Can a race condition happen here?

## D8.3 Outlook: Further Undecidable Problems

## And What Else?

- ▶ Here we conclude our discussion of undecidable problems.
- ▶ Many more undecidable problems exist.
- ▶ In this section, we briefly discuss some further classical results.

## Post's Correspondence Problem

### Post's Correspondence Problem (PCP)

Given: a set of "domino" types

$$A: \begin{array}{|c|} \hline 1 \\ \hline 101 \\ \hline \end{array} \quad B: \begin{array}{|c|} \hline 10 \\ \hline 00 \\ \hline \end{array} \quad C: \begin{array}{|c|} \hline 011 \\ \hline 11 \\ \hline \end{array}$$

Question: Can we lay a (non-empty) sequence of dominoes such that the top and bottom word match?  
(We may use each type as often as we want.)

$$\begin{array}{|c|} \hline 1 \\ \hline 101 \\ \hline \end{array} \quad \begin{array}{|c|} \hline 011 \\ \hline 11 \\ \hline \end{array} \quad \begin{array}{|c|} \hline 10 \\ \hline 00 \\ \hline \end{array} \quad \begin{array}{|c|} \hline 011 \\ \hline 11 \\ \hline \end{array}$$

A      C      B      C

↔ undecidable by reduction from Halting problem  
(see Schöning, Chapter 2.7)

German: Postsches Korrespondenzproblem

## Undecidable Grammar Problems

### Some Grammar Problems

Given context-free grammars  $G_1$  and  $G_2, \dots$

- ▶ ... is  $\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset$ ?
- ▶ ... is  $|\mathcal{L}(G_1) \cap \mathcal{L}(G_2)| = \infty$ ?
- ▶ ... is  $\mathcal{L}(G_1) \cap \mathcal{L}(G_2)$  context-free?
- ▶ ... is  $\mathcal{L}(G_1) \subseteq \mathcal{L}(G_2)$ ?
- ▶ ... is  $\mathcal{L}(G_1) = \mathcal{L}(G_2)$ ?

Given a context-sensitive grammar  $G, \dots$

- ▶ ... is  $\mathcal{L}(G) = \emptyset$ ?
- ▶ ... is  $|\mathcal{L}(G)| = \infty$ ?

↔ all undecidable by reduction from PCP  
(see Schöning, Chapter 2.8)

## Gödel's First Incompleteness Theorem (1)

### Definition (Arithmetic Formula)

An **arithmetic formula** is a closed predicate logic formula using

- ▶ constant symbols 0 and 1,
- ▶ function symbols + and ·, and
- ▶ no relation symbols.

It is called **true** if it is true under the usual interpretation of 0, 1, + and · over  $\mathbb{N}_0$ .

German: arithmetische Formel

## Gödel's First Incompleteness Theorem (2)

### Gödel's First Incompleteness Theorem

The problem of **deciding if a given arithmetic formula is true** is undecidable.

Moreover, neither it nor its complement are semi-decidable.

As a consequence, there exists no sound and complete proof system for arithmetic formulas.

↔ proof idea: relate arithmetic formulas to halting problem for WHILE programs (see Schöning, Chapter 2.9)

German: erster Gödelscher Unvollständigkeitssatz

## D8.4 Summary

## Summary

undecidable but semi-decidable problems:

- ▶ **special halting problem** a.k.a. self-application problem (from previous chapter)
- ▶ **general halting problem**
- ▶ **halting problem on empty tape**

Rice's theorem:

- ▶ “In general one cannot determine algorithmically what a given program (or Turing machine) computes.”

further undecidable problems:

- ▶ Post's Correspondence Problem
- ▶ many grammar problems
- ▶ proving or refuting the truth of an arithmetic formula