Theory of Computer Science

D8. Halting Problem Variants & Rice's Theorem

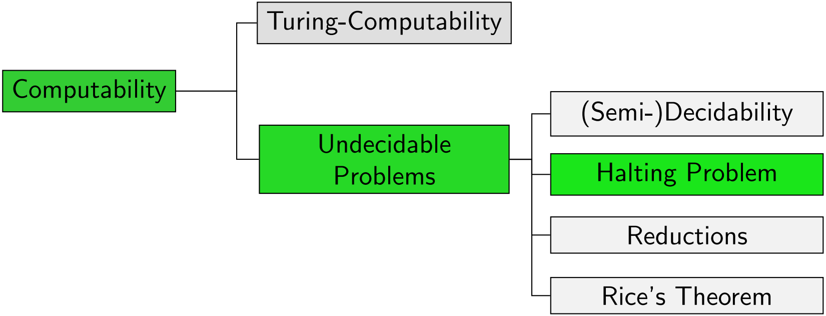
Gabriele Röger

University of Basel

April 17, 2019

Other Halting Problem Variants

Overview: Computability Theory



Reminder: Special Halting Problem

Definition (Special Halting Problem)

The special halting problem or self-application problem is the language

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

German: spezielles Halteproblem, Selbstanwendbarkeitsproblem

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

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?)

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 $K \leq H$.

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

f is clearly total and computable, and

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

General Halting Problem (2)

Proof.

We show $K \leq H$.

We define $f : \{0,1\}^* \to \{0,1,\#\}^*$ 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

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?)

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 $H \leq H_0$.

Halting Problem on Empty Tape (2)

Proof.

We show $H \leq H_0$.

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

Halting Problem on Empty Tape (2)

Proof.

We show $H \leq H_0$.

Consider the function $f : \{0, 1, \#\}^* \to \{0, 1\}^*$
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.

Halting Problem on Empty Tape (2)

Proof.

We show $H \leq H_0$.

Consider the function $f : \{0, 1, \#\}^* \to \{0, 1\}^*$
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

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.

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.

Halting Problem on Empty Tape (3)

Proof (continued).

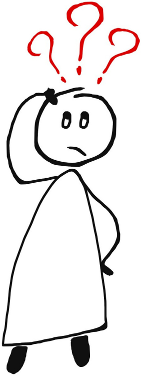
- Construct a TM M_1 that behaves as follows:
  - If the input is empty: write x onto the tape and move the head to the first symbol of x (if $x \neq \varepsilon$); then stop
  - otherwise, stop immediately
- Construct TM M that first runs M_1 and then M_2.
- Return the encoding of M.

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

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

$\rightsquigarrow H \leq H_0 \rightsquigarrow H_0$ undecidable $\square$

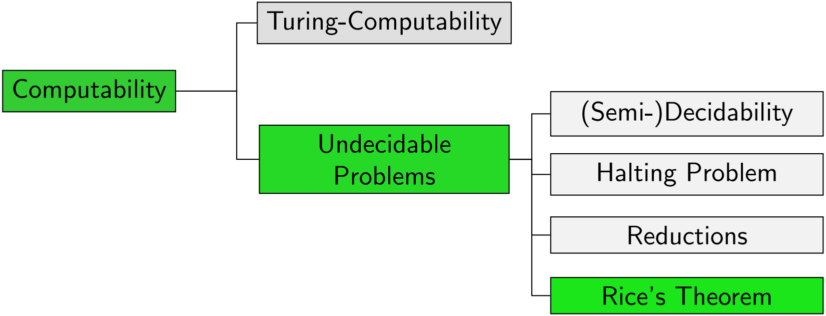
Questions



Questions?

Rice's Theorem

Overview: Computability Theory



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 non-trivial 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 Ω be the function that is undefined everywhere.

Rice's Theorem (3)

Proof.

Let Ω be the function that is undefined everywhere.

Case distinction:

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

Rice's Theorem (3)

Proof.

Let Ω 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 $\mathcal{S} \neq \mathcal{R}$).

Rice's Theorem (3)

Proof.

Let Ω 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).

We show that $\bar{H}_0 \leq C(\mathcal{S})$.

Consider function $f : \{0,1\}^* \to \{0,1\}^*$,
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

Rice's Theorem (4)

Proof (continued).

We show that $\bar{H}_0 \leq C(\mathcal{S})$.

Consider function $f : \{0,1\}^* \to \{0,1\}^*$,
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 $f(w)$?

Rice's Theorem (5)

Proof (continued).

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

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

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

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

...

Rice's Theorem (6)

Proof (continued).

Further:

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

Rice's Theorem (6)

Proof (continued).

Further:

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

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

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 $C(\mathcal{S})$.

Rice's Theorem (6)

Proof (continued).

Further:

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

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.

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. □

Rice's Theorem: Consequences

Was it worth it?

We can now 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: Examples

- Does a given TM compute a constant function?
  $\mathcal{S} = \{f \mid f$ is total and computable and for all x, y in the domain of $f : f(x) = f(y)\}$
- Does a given TM compute a total function?
  $\mathcal{S} = \{f \mid f$ is total and computable$\}$
- Does a given TM compute the identity function?
  $\mathcal{S} = \{f \mid f(x) = x$ for all $x\}$
- Does a given TM add two natural numbers?
  $\mathcal{S} = \{f : \mathbb{N}_0^2 \to \mathbb{N}_0 \mid f(x, y) = x + y\}$
- Does a given TM compute the computable function f?
  $\mathcal{S} = \{f\}$
  (full automization of software verification is impossible)

Rice's Theorem: Pitfalls

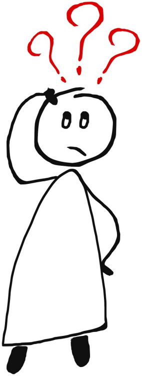
- $\mathcal{S} = \{f \mid f$ can be computed by a DTM with an even number of states$\}$
  Rice's theorem not applicable because $\mathcal{S} = \mathcal{R}$
- $\mathcal{S} = \{f : \{0,1\}^* \to_p \{0,1\} \mid f(w) = 1$ iff M_w does not terminate on $\epsilon\}$?
  Rice's theorem not applicable because $\mathcal{S} \not\subseteq \mathcal{R}$
- Show that $\{w \mid M_w$ traverses all states on every input$\}$ is undecidable.
  Rice's theorem not directly applicable because not a semantic property (the function computed by M_w can also be computed by a TM that does not traverse all states)

Rice's Theorem: Practical Applications

Undecidable due to Rice's theorem + a small reduction:

- 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?

Questions



Questions?

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."

What's Next?

contents of this course:

A. background ✓
  - ▷ mathematical foundations and proof techniques

B. logic ✓
  - ▷ How can knowledge be represented?
    How can reasoning be automated?

C. automata theory and formal languages ✓
  - ▷ What is a computation?

D. Turing computability
  - ▷ What can be computed at all?

E. complexity theory
  - ▷ What can be computed efficiently?

F. more computability theory
  - ▷ Other models of computability

What's Next?

contents of this course:

A. background ✓
   - ▷ mathematical foundations and proof techniques

B. logic ✓
   - ▷ How can knowledge be represented?
     How can reasoning be automated?

C. automata theory and formal languages ✓
   - ▷ What is a computation?

D. Turing computability ✓
   - ▷ What can be computed at all?

E. complexity theory
   - ▷ What can be computed efficiently?

F. more computability theory
   - ▷ Other models of computability

Quiz

kahoot.it