Theory of Computer Science

C5. Rice's Theorem

Gabriele Röger

University of Basel

May 9, 2022

C5.1 Rice's Theorem

Rice's Theorem (1)

- We have shown that the following problems are undecidable:
  - 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 partial 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 Turing-recognizable. (But there are sets $\mathcal{S}$ for which one of the two languages is Turing-recognizable.)

Rice's Theorem (3)

Proof.

Let Ω be the partial 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 partial 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

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

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

...

Rice's Theorem (6)

Proof (continued).
Further:

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

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?
  $\mathcal{S} = \{f \mid f \text{ is total and computable and for all } x, y \text{ in the domain of } f : f(x) = f(y)\}$
- Does a given TM compute a total function?
  $\mathcal{S} = \{f \mid f \text{ is total and computable}\}$
- Does a given TM compute the identity function?
  $\mathcal{S} = \{f \mid f(x) = x \text{ for all } x\}$
- 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)

Exercise

This was an exam question in 2019.

Is the following informally described problem decidable? Give a brief justification.

Given a deterministic Turing machine M, is the language recognized by M regular?

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?

C5.2 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 Correspondence Problem: Example

Example (Post Correspondence Problem)

Given: different kinds of "'dominos"'

	1:	2:	3:
upper	1	10	011
lower	101	00	11

(an infinite number of each kind)

Question: Is there a sequence of dominos such that the upper and lower row match (= are equal)

1	011	10	011
101	11	00	11
1	3	2	3

Post Correspondence Problem: Definition

Definition (Post Correspondence Problem PCP)

Given: Finite sequence of pairs of words $(t_1, b_1), (t_2, b_2), \dots, (t_k, b_k)$, where $t_i, b_i \in \Sigma^+$ (for an arbitrary alphabet Σ)

Question: Is there a sequence $i_1, i_2, \dots, i_n \in \{1, \dots, k\}$, $n \geq 1$, with $t_{i_1} t_{i_2} \dots t_{i_n} = b_{i_1} b_{i_2} \dots b_{i_n}$?

Theorem (Undecidability of PCP)

PCP *is undecidable*.

Undecidable Grammar Problems

Some Grammar Problems

Given context-free grammars G_1 and G_2, ...

- ... 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, ...

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

$\rightsquigarrow$ 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 $\cdot$, and
- equality ($=$) as the only relation symbols.

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

German: arithmetische Formel

Beispiel: $\forall x \exists y \forall z (((x \cdot y) = z) \wedge ((1 + x) = (x \cdot y)))$

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 Turing-recognizable.

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

German: erster Gödelscher Unvollständigkeitssatz

Summary

Rice's theorem:

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

How to Prove Undecidability?

- statements on the computed function of a TM/an algorithm
  $\rightarrow$ easiest with Rice' theorem
- other problems
  - directly with the definition of undecidability
    $\rightarrow$ usually quite complicated
  - reduction from an undecidable problem, e.g.
    $\rightarrow$ halting problem (H)
    $\rightarrow$ Post correspondence problem (PCP)

What's Next?

contents of this course:

A. background ✓
  ▷ mathematical foundations and proof techniques
B. automata theory and formal languages ✓
  ▷ What is a computation?
C. Turing computability ✓
  ▷ What can be computed at all?
D. complexity theory
  ▷ What can be computed efficiently?
E. more computability theory
  ▷ Other models of computability