Rice's Theorem [*]

Hans Hüttel

Autumn 2014

Contents

1 Introduction

In this note I will show a theorem which states that a very large family of problems are all undecidable. Our result, known as *Rice's theorem* after Henry Rice who proved the result in 1953 [2], states that if $\mathcal{S}$ is a non-trivial property of Turing-recognizable languages, then the problem

Given a TM M, does $L(M)$ have the property $\mathcal{S}$?

is undecidable.

An example of a property of Turing-recognizable languages could be, say, the property of being a regular language. If we had an algorithm for determining, given a TM M, if $L(M)$ was regular we would have a tool that we could use to determine whether we could replace M by an equivalent finite-state automaton. (In fact, we already know that this problem is undecidable – this is the $REGULAR_{\mathsf{TM}}$ problem of Section 5.1 in ITTTOC.)

It would also be nice if we could determine if a Turing machine M accepted all strings that contained some specific sequence of characters. Then we would have a tool for determining e.g. if all data that we assumed to be correct were indeed correctly handled.

Unfortunately this is also not possible. Algorithms that check properties of input languages cannot exist, as we shall now see.

[*]Denne note findes også på dansk som [1]

2 Properties

First,we give a mathematical definition of the notion of property. In our setting, a property is simply a class of Turing-recognizable languages.

Definition 1. A class of languages $\mathcal{S}$ is called a *property of Turing-recognizable languages* if the members of $\mathcal{S}$ are Turing-recognizable languages, i.e. for every $L \in \mathcal{S}$ there exists a TM that recognizes L.

Example 1. The class of languages

$$\mathcal{S}_{\mathsf{REGULAR}} = \{L \mid L \text{ is regular}\}$$

is a property of Turing-recognizable languages, since all its members are Turing-recognizable languages. The class of languages

$$\mathcal{S}_2 = \{EQ_{\mathsf{TM}}, A_{\mathsf{TM}}\}$$

is *not* a property of Turing-recognizable languages, since EQ_{TM} is a language in $\mathcal{S}_2$ which is not Turing-recognizable.

A property of Turing-recognizable languages is non-trivial if some recognizable languages have the property and others do not. We formalize this as follows:

Definition 2. Let $\mathcal{S}$ be a property of Turing-recognizable languages. We say that $\mathcal{S}$ is *non-trivial* if there exist Turing-recognizable languages L_1 and L_2 such that $L_1 \in \mathcal{S}$ but $L_2 \notin \mathcal{S}$. If a property of Turing-recognizable languages is not non-trivial, we call it *trivial.*

Example 2. The class of languages

$$\mathcal{S}_{\mathsf{REGULAR}} = \{L \mid L \text{ is regular}\}$$

is a non-trivial property of Turing-recognizable languages, since the language $L_1 = \{a^n \mid n \geq 0\}$ is a member of $\mathcal{S}_{\mathsf{REGULAR}}$ (and therefore also recognizable, as $\mathcal{S}_{\mathsf{REGULAR}}$ is a property), whereas the language $L_2 = \{a^n b^n \mid n \geq 0\}$ is also recognizable, but not a member of $\mathcal{S}_{\mathsf{REGULAR}}$, as it is a non-regular language.

Example 3. There are only two ways of violation the conditions for being non-trivial. The properties

$$\begin{aligned} \mathcal{S}_{\mathsf{all}} &= \{L \mid L \text{ is Turing-recognizable}\} \quad (1) \\ \mathcal{S}_{\mathsf{none}} &= \emptyset \quad (2) \end{aligned}$$

are both trivial, and they are the only such properties. $\mathcal{S}_{\mathsf{all}}$ is trivial, since there does not exist a Turing-recognizable L_2 such that $L_2 \notin \mathcal{S}_{\mathsf{all}}$. $\mathcal{S}_{\mathsf{none}}$ is trivial, since there does not exist a Turing-recognizable L_1 such that $L_1 \in \mathcal{S}_{\mathsf{none}}$.

Example 4. The following are examples of non-trivial properties of Turing-recognizable languages:

- The class $\{\emptyset\}$ whose only member is the empty language (not to be confused with the empty property $\mathcal{S}_{\mathsf{none}}$ from Example 3!)

- The class $\{\Sigma^*\}$ whose only member is the language of all strings over alphabet Σ.
- The class of languages that have the empty string ε as an element.
- The class of context-free languages
- The class of Turing-decidable languages

Problem 1. Express the properties in Example 4 using set notation, if they were not already expressed in this way, and prove that they are non-trivial properties of Turing-recognizable languages.

3 The undecidability result

Definition 3. Let $\mathcal{S}$ be a property. Then the language $\mathcal{S}_{\mathsf{TM}}$ is defined as

$$\mathcal{S}_{\mathsf{TM}} = \{\langle M\rangle \mid L(M) \in \mathcal{S}\}$$

The result that we shall now state and prove, states that $\mathcal{S}_{\mathsf{TM}}$ is undecidable whenever $\mathcal{S}$ is non-trivial.

Example 5. Consider the property $\mathcal{S}_{\mathsf{REGULAR}}$defined in Example 1. Then

$$\mathcal{S}_{\mathsf{TM}} = \{\langle M\rangle \mid L(M) \text{ is a regular language}\}$$

Note that this is the language $\mathit{REGULAR}_{\mathsf{TM}}$ of Section 5.1 of ITTTOC.

Theorem 2. *If $\mathcal{S}$ is a non-trivial property of Turing-recognizable languages, then $\mathcal{S}_{\mathsf{TM}}$ is undecidable.*

Proof idea: We show that if $\mathcal{S}$ is a non-trivial property of Turing-recognizable languages, then we can reduce A_{TM} to $\mathcal{S}_{\mathsf{TM}}$. In other words, we show that, given $\langle M, w\rangle$ we can construct the description of a machine a $\langle M'\rangle$ such that M accepts w if and only if $L(M') \in \mathcal{S}$. □

Proof. We know that $\mathcal{S}$ is non-trivial, so there is some $L_1 \in \mathcal{S}$ and some other $L_2 \notin \mathcal{S}$. In our proof we pick some L_1 and assume that the empty language $\emptyset$ is our L_2, i.e. that $\emptyset \notin \mathcal{S}$. We then construct, given $\langle M, w\rangle$ a Turing machine M' such that if M accepts w then $L(M') = L_1$ and if M does not accept w, then $L(M') = \emptyset$.

It is no loss of generality to assume that $\emptyset \notin \mathcal{S}$. For if we wanted to show that $\mathcal{S}$ was undecidable and $\emptyset \in \mathcal{S}$, we could simply consider the complementary property $\overline{\mathcal{S}}$. Clearly $\overline{\mathcal{S}}_{\mathsf{TM}}$ is decidable if and only if $\mathcal{S}_{\mathsf{TM}}$ is.

Since $\mathcal{S}_{\mathsf{TM}}$ is a non-trivial property of Turing-recognizable languages, by Definition 2 there must be some $L_1 \in \mathcal{S}$. Since L_1 is Turing-recognizable, there exists a Turing machine recognizing L_1. We denote the recognizer for L_1 by M_L.

We can now present the reduction from A_{TM} to $\mathcal{S}_{\mathsf{TM}}$. From $\langle M, w\rangle$ and using M_L we construct the description of a new machine M':

```
$M'$ = "On input $x$
        1. Simulate $M$ on $w$
```

2. If M accepts w, then simulate M_L on x
3. If M_L accepted x, then *accept*"

Now we show that our reduction is faithful, i.e. that $\langle M, w \rangle \in A_{\mathsf{TM}}$ if and only if $\langle M' \rangle \in \mathcal{S}_{\mathsf{TM}}$.

First, suppose that $\langle M, w \rangle \in A_{\mathsf{TM}}$, that is, M accepts w. Then M' will accept its input x exactly when M_L accepts x. So here $L(M') = L_1$, and $L_1 \in \mathcal{S}$ by assumption.

Now suppose that $\langle M, w \rangle \notin A_{\mathsf{TM}}$, that is, M does not accept w. Then M' will never reach the simulation of M_L, and consequently M' accepts no string. So here $L(M') = \emptyset$, and $\emptyset \notin \mathcal{S}$ by assumption.

We can now conclude that $\mathcal{S}_{\mathsf{TM}}$ cannot be decidable. For if $\mathcal{S}_{\mathsf{TM}}$ were decidable, we would have a decider for it, R, but then we could also build the following decider for A_{TM}:

$H =$ "On input $\langle M, w \rangle$

1. Construct $\langle M' \rangle$ from $\langle M, w \rangle$
2. Feed $\langle M' \rangle$ to R
3. If R accepts,then *accept.* If R rejects, then *reject.*

□

4 Consequences of Rice's theorem

Theorem 2 has a whole bunch of consequences, some of which I will first present. I will then mention some of the frequent pitfalls that should be avoided.

5 Lots of undecidability results for free!

The following results are immediate consequences of Theorem 2 and Example 4:

Corollary 3. *The following problems are all undecidable. Given Turing machine M,*

1. *– does M accept the empty string?*
2. *– does M accept no inputs at all?*
3. *– does M accept all inputs?*
4. *– is $L(M)$ regular ? Context-free ? Turing-decidable?*

Example 6. Consider the decision problem

Given TM M, does M accept all strings that are palindromes?

If we want to use Rice's theorem, we need to reformulate this problem so that it talks directly about a class of languages. First, we can express our decision problem as the language

$$PALINDROMES_{\mathsf{TM}} = \{\langle M\rangle \mid L(M) \text{ contains all palindromes}\}$$

We can now see that the property we are really looking at is

$$\mathcal{S}_{PALINDROMES} = \{L \mid L \text{ is Turing-recognizable and contains all palindromes}\}$$

This class is by definition a class of Turing-recognizable languages. It is non-trivial, since the Turing-recognizable language of all strings Σ^* is a member of the class, whereas the empty language $\emptyset$ (which is of course recognizable) is not a member of the class.

By Rice's theorem we now get that $PALINDROMES_{\mathsf{TM}}$ is undecidable.

We can also use Rice's theorem to show that automatic complexity analysis is not possible.

Example 7. Consider the following class of decidable languages:

$$P = \{L \mid L \text{ can be decided by a Turing-machine with polynomial-time complexity}\}$$

We know that P must be a non-trivial property of recognizable languages (why?). Therefore the problem

> Given a TM M, is it the case that M can optimized to have polynomial-time complexity?

is undecidable.

6 Some frequent pitfalls

Sometimes people try to apply Rice's theorem in situations where it cannot be applied or cannot be easily applied. As a result, the 'proofs' that they come up with are either very imprecise or completely incorrect. One should of course strive to avoid such situations. Here are three examples.

Example 8. Consider the decision problem

> Given TM M, is $L(M)$ Turing-recognizable?

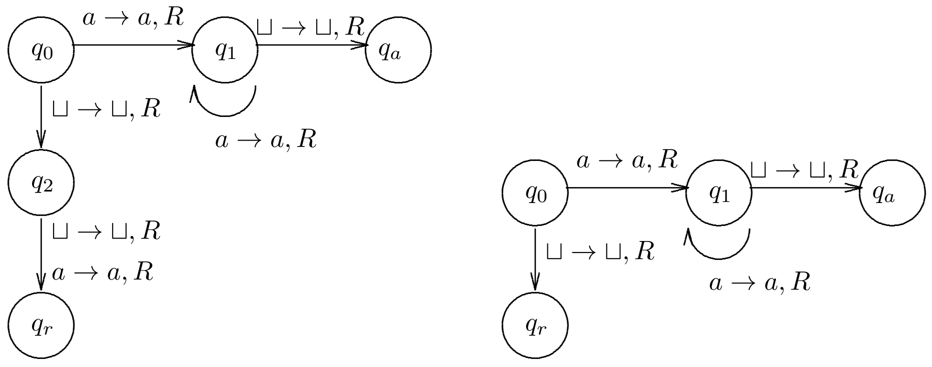
Sometimes, people will say:

> The property 'L is Turing-recognizable' is non-trivial, and by Rice's theorem we know that our problem is undecidable.

However, this is false. The property is *trivial*, as we saw in Example 3. In fact, our problem is decidable with the following decider:

> R = "On input $\langle M\rangle$ *accept*"

Example 9. Let us say that a state of a TM is *ordinary* if it is not the accept state or the reject state. Consider the decision problem



(a) M_1 (b) M_2

Figure 1: Two machines where $L(M_1) = L(M_2)$ but $\langle M_1\rangle \notin TOUR_{\mathsf{TM}}$ and $\langle M_2\rangle \in TOUR_{\mathsf{TM}}$

> Given TM M, is there any input w such that M visits every ordinary state during its computation?

Sometimes, people will say:

> The property 'For some input w M visits every ordinary state during its computation' is non-trivial, and by Rice's theorem we know that our problem is undecidable.

However, this does not constitute a proof. The concept 'property' is used in an incorrect way.

The decision problem can be expressed as the language

$$TOUR_{\mathsf{TM}} = \{\langle M\rangle \mid \text{For some input } w\ M \text{ visits every ordinary state during its computation}\}$$

$TOUR_{\mathsf{TM}}$ is indeed undecidable but one cannot apply Rice's theorem to prove it. The condition 'M visits every ordinary state during its computation for some input w' directly mentions the ordinary states of the Turing machine, i.e. we are not talking about a property corresponding to a class of languages but about a property of a machine.

We can prove that we cannot find a property that would apply; we do this by finding two machines M_1 and M_2 such that $L(M_1) = L(M_2)$, but where $\langle M_1\rangle \in TOUR_{\mathsf{TM}}$ and $\langle M_2\rangle \notin TOUR_{\mathsf{TM}}$. A simple example can be seen in figure 1. Both machines recognize the language $\{a^n \mid n > 0\}$, but M_1 has an additional state which is only used when the machine does not accept its input. Consequently $\langle M_1\rangle \notin TOUR_{\mathsf{TM}}$.

Problem 4. Prove that $TOUR_{\mathsf{TM}}$ is undecidable by reduction from A_{TM}.

Example 10. Consider the problem EQ_{TM} (ITTTOC, section 5.1). Sometimes, people will say:

> The property '$L(M_1) = L(M_2)$' is non-trivial, and by Rice's theorem we know that our problem is undecidable.

This is not just imprecise, it is also meaningless, since the terminology is used incorrectly. Rice's theorem cannot be applied in the form that we have seen. For the problem stated involves two languages, not one. We will need a generalization of Rice's theorem to pairs of languages if we want to prove that EQ_{TM} is undecidable in this way. See problem 5 below.

Problem 5. A property of *pairs* of recognizable languages is any set $\mathcal{S}$ of pairs that are all of the form (L_1, L_2), where L_1 and L_2 are Turing-recognizable languages. We say that a property $\mathcal{S}$ of pairs of recognizable languages is *non-trivial* if there exist pairs of recognizable languages (L_1, L_2) and (L_3, L_4) such that $(L_1, L_2) \notin \mathcal{S}$ and $(L_3, L_4) \in \mathcal{S}$.

> **Theorem 6.** *If a property $\mathcal{S}$ of pairs of recognizable languages is non-trivial, the language*
>
> $$L_{\mathcal{S}} = \{\langle M, M' \rangle \mid (L(M), L(M')) \in \mathcal{S}\}$$
>
> *is undecidable.*

1. Prove this theorem. *Hint:* Reduction from normal property checking.
2. Use the theorem to show that EQ_{TM} is undecidable.

References

[1] Hüttel, H. *Om Rice's sætning*, Aalborg Universitet 2013.

[2] Rice, H.G. Classes of recursively enumerable sets and their decision problems, *Trans. AMS* **89**:25–59.

[3] Sipser, M. Introduction to the Theory of Computation, First edition, PWS Publishing 1997.