Rice's Theorem about Undecidability

Language Properties: Fix some alphabet Σ. There are many possible languages A over alphabet Σ; that is, many different possible $A \subseteq \Sigma^*$. Some of these languages are finite, some are regular, some of them are context-free, some of them are decidable, etc. Each of "finite","regular","context-free", "decidable", "Turing-recognizable" is an example of what we call a *language property*. A language property $\mathcal{P}$ is simply a collection of some of the possible languages over Σ. In keeping with how we use "finite","regular","context-free", "decidable", and "Turing-recognizable", we will say that a language A *has property* $\mathcal{P}$ iff A is one of the languages in the collection $\mathcal{P}$.

So far, all of the examples of properties we have mentioned have an infinite number of different languages in the associated collection. However, the collection $\mathcal{P}$ might be very small. For example the collection $\mathcal{P}$ might be empty (no language satisfies this property) or have just a single language in it; for example, $\mathcal{P}$ might be the "emptiness" property (only the empty language $\varnothing$ has this property).

Given any language property $\mathcal{P}$, we can define its complement, $\overline{\mathcal{P}}$, to be the collection of all languages that do *not* have property $\mathcal{P}$.

For any language property $\mathcal{P}$, we can define an associated language of Turing machine descriptions

$$\mathcal{P}_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ has property } \mathcal{P}\}.$$

We have seen a couple of languages of this sort that we have already proven to be undecidable:

$$REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\} \text{ and}$$
$$E_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \varnothing\}.$$

The question that Rice's Theorem answers is: for which properties $\mathcal{P}$ is $\mathcal{P}_{TM}$ decidable?

There are two easy cases:

1. If *no* Turing-recognizable language has property $\mathcal{P}$ then, since $L(M)$ is always a Turing-recognizable language, $L(M)$ never has property $\mathcal{P}$, so $\mathcal{P}_{TM}$ is empty and hence is decidable (always reject).
2. If *every* Turing-recognizable language has property $\mathcal{P}$ then, since $L(M)$ is always a Turing-recognizable language, the answer must always be that $L(M)$ has property $\mathcal{P}$, so $\mathcal{P}_{TM} = \{\langle M\rangle \mid M \text{ is a TM}\}$ and this is decidable because all we need to check is that the input is a well-formed code of a TM.

If neither of these cases holds, then there is some TM M_1 such that $L(M_1)$ has property $\mathcal{P}$ and some other TM M_0 such that $L(M_0)$ does not have property $\mathcal{P}$; we say that the language property $\mathcal{P}$ is *non-trivial* if both M_0 and M_1 exist.

Rice's Theorem: $\mathcal{P}_{TM}$ is undecidable for every non-trivial language property $\mathcal{P}$.

Proof. We will actually need to split this into two different cases depending on the property $\mathcal{P}$.

CASE 1: $\varnothing$ does *not* have property $\mathcal{P}$.

In this case, to show that $\mathcal{P}_{TM}$ is undecidable, we will prove that $A_{TM} \leq_m \mathcal{P}_{TM}$. The mapping reduction f will use the TM M_1 such that $L(M_1)$ has property $\mathcal{P}$ that we know exists because $\mathcal{P}$ is non-trivial.

The description of the function f is:
On input $\langle M, w\rangle$ produce $\langle M'_w\rangle$ where M'_w is the following TM:
"On input x, run M on input w; if M rejects w then reject. If M accepts w then run M_1 on input x and accepts iff M_1 accepts."

Clearly, f is a computable function. We just need to check that it has the right properties:

If M accepts w then M'_w accepts its input x iff $x \in L(M_1)$; that is, $L(M'_w) = L(M_1)$. Therefore $L(M'_w)$ has property $\mathcal{P}$ and so $\langle M'_w\rangle \in \mathcal{P}_{TM}$.

If M does not accept w then M'_w will not accept any input x, so $L(M'_w) = \varnothing$. Since $\varnothing$ does not have property $\mathcal{P}$, this means that $\langle M'_w\rangle \notin \mathcal{P}_{TM}$.

Therefore we have $\langle M, w\rangle \in A_{TM} \Leftrightarrow f(\langle M, w\rangle) \in \mathcal{P}_{TM}$ and hence $A_{TM} \leq_m \mathcal{P}_{TM}$.

CASE 2: $\varnothing$ has property $\mathcal{P}$.

In this case, we will prove undecidability by showing that $\overline{A_{TM}} \leq_m \mathcal{P}_{TM}$ (which also shows that $\mathcal{P}_{TM}$ is not even Turing-recognizable.)

The first observation in this case is that the complement property $\overline{\mathcal{P}}$ satisfies Case 1. Therefore, as we have shown, $A_{TM} \leq_m \overline{\mathcal{P}}_{TM}$.

Since $A \leq_m \overline{B}$ is equivalent to $\overline{A} \leq_m B$, this is essentially enough, except for a minor point: $\overline{\mathcal{P}}_{TM} \neq \overline{\mathcal{P}_{TM}}$; the reason is that strings in $\overline{\mathcal{P}}_{TM}$ are all codes of Turing machines, but $\overline{\mathcal{P}_{TM}}$ is the complement of $\mathcal{P}_{TM}$ and hence includes all strings that are not codes of Turing machines.

However, we can easily see that $\overline{\mathcal{P}}_{TM} \leq_m \overline{\mathcal{P}_{TM}}$ since we can simply have the reduction function f check that its input string is a correctly formed code of a TM: If it is correctly formed, just pass it on as output; otherwise, we are supposed to reject it so we map it to $\langle M_\varnothing\rangle$ where $M_\varnothing$ rejects all its inputs. Since $L(M_\varnothing)$ has property $\mathcal{P}_{TM}$, we know that $\langle M_\varnothing\rangle \notin \overline{\mathcal{P}_{TM}}$.

Combining the reductions we get $A_{TM} \leq_m \overline{\mathcal{P}_{TM}}$ and hence $\overline{A_{TM}} \leq_m \mathcal{P}_{TM}$ as required. □

Note: This is not the only way we could have split up the cases to produce the proof. The key thing we used for Case 1 is that we had two languages, a smaller one ($\varnothing$) that didn't have the property and a larger one $L(M_1)$ that did. We then used running M on input w and checking if it accepted to determine which of the two languages was produced. An alternative Case 1 could have been: Σ^* has property $\mathcal{P}$. In that case we would use M_0 instead of M_1 and the reduction would produce $\langle M''_w\rangle$ where M''_w does the following:
"On input x, in parallel run M_0 on input x and run M on input w. If either accepts then accept."
In this case, $L(M''_w)$ is either M_0 or Σ^* depending on whether or not M accepts w. If we know that M_0 is a decider then in M''_w we could instead run M_0 on input x *before* we run M on input w, but in general we don't know that. The example of the proof for the undecidability of $REGULAR_{TM}$ given in Sipser's text is like this version of the argument in which M_0 is known to be a decider. (For the alternative Case 2, we would again use the complement of the property and reduce it to the alternative Case 1.)

Rice's Theorem is kind of an analogue of the well-known saying "you can't tell a book by its cover". Instead it says, in general, "You can't tell what a TM/program does just by looking at its code".

Rice's Theorem is very general but it is important to be careful about what it does and does not say: It only applies when the input is supposed to be the code of a TM and only says that one cannot decide anything non-trivial about its input/output behavior (where the output behavior is just about which inputs are accepted).