Notes on Rice's Theorem

Consider any kind of software testing problem. Its description will typically start as "For a given program decide whether the function it computes is ...". In the setting of Turing machines, we often encounter natural problems of the form "Decide if the language recognized by a given Turing machine $\langle M\rangle$ ... ." Rice's theorem proves in one clean sweep that *all* these problems are undecidable. That is, whenever we have a decision problem in which we are given a Turing machine and we are asked to determine a property of the language recognized by the machine, that decision problem is always undecidable. The only exceptions will be the trivial properties that are always true or always false.

We use the following notation. If M is a Turing machine with input alphabet Σ, then $L(M) \subseteq \Sigma^*$ is the language recognized by M, that is, the set of strings that are accepted by M.

Theorem 1 *Let $\mathcal{C}$ be a set of languages. Consider the language $L_{\mathcal{C}}$ defined as follows*

$$L_{\mathcal{C}} = \{\langle M\rangle \mid L(M) \in \mathcal{C}\} \ .$$

Then either $L_{\mathcal{C}}$ is empty, or it contains the descriptions of all Turing machines, or it is undecidable.

To make sense of the statement of the theorem, think of a property of languages that you would like to test. For example the property of being regular. Then define $\mathcal{C}$ to be the set of all languages with that property. (In the example, $\mathcal{C}$ would be the set of regular languages.) Now, $L_{\mathcal{C}}$ is the language of (representations of) Turing machines that recognize languages having the property. (In the example, $L_{\mathcal{C}}$ would be the language of Turing machines that recognize regular languages.) The theorem says that unless every Turing machine recognizes a language with the property (not true for regular languages) and unless no Turing machine recognizes a language with the property (not true for regular languages), then $L_{\mathcal{C}}$ is undecidable. (So it is undecidable to tell whether a given Turing machine recognizes a regular language or not.)

Think of all the corollaries that you can infer from Rice's Theorem. It is undecidable to determine whether a given Turing machine accepts a finite or infinite number of inputs. It is undecidable to determine whether a given Turing machine accepts only (representations of) prime numbers, and so on.

Proof: Suppose towards a contradiction that for same class $\mathcal{C}$ the language $L_{\mathcal{C}}$ is not empty, it does not contain the descriptions of all Turing machines, and it is decidable. Then $L_{\bar{\mathcal{C}}}$ is also not empty, not containing all Turing machines, and decidable.

Suppose that $\emptyset \notin \mathcal{C}$, otherwise apply the argument below to $\bar{\mathcal{C}}$ instead of $\mathcal{C}$.

Let M_{in} be a machine such that $\langle M_{in}\rangle$ is in $L_{\mathcal{C}}$.

We will show that the Acceptance problem is decidable, and so we will reach a contradiction.

Given an input $(\langle M\rangle, w)$ for the Acceptance problem, we constract a new Turing machine M_w that does the following: on input x, M_w first simulates the behaviour of M on input w and

- If M on input w loops, then so does M_w;
- If M on input w rejects, then so does M_w;

- If M on input w accepts, then M_w continues with a simulation of M_{in} on input x.

In summary:

- If M accepts w, then M_w behaves like M_{in}, and M_w accepts an input x if and only if M_{in} does. In other words, $L(M_w) = L(M_{in}) \in \mathcal{C}$ and so $\langle M_w \rangle \in L_{\mathcal{C}}$;
- if M does not accept w, then M_w does not accept any input, and $L(M_w) = \emptyset \notin \mathcal{C}$, which implies $\langle M_w \rangle \notin L_{\mathcal{C}}$.

We have proved that $(\langle M \rangle, w) \in A$ if and only if $\langle M_w \rangle \in L_{\mathcal{C}}$, and so A would be decidable if $L_{\mathcal{C}}$ were decidable. We have reached a contradiction. $\square$