Note on Rice’s Theorem

The purpose of this note is to give some details of Rice’s Theorem and its proof. This theorem is a useful tool in determining undecidability.

Recall that the set of all languages, \( \mathcal{P}(\Sigma^*) \), is uncountable.

Let \( T = \{ \langle M \rangle \mid M \text{ is a Turing machine} \} \) be the set of all Turing machines. This set is countable. Let \( R = \{ L(M) \mid \langle M \rangle \in T \} \) be the set of all Turing-recognizable languages. This set is also countable. Rice’s Theorem helps identify languages which are not decidable. Specifically, it helps identify undecidable languages which are subsets of \( T \). These languages will all be sets of Turing machine descriptions.

**Definition 1.** A set \( P \subset T \) is a property if, whenever \( L(M_1) = L(M_2) \), we have either that

- both \( \langle M_1 \rangle, \langle M_2 \rangle \in P \), or
- both \( \langle M_1 \rangle, \langle M_2 \rangle \notin P \).

For example, the following are properties:

<table>
<thead>
<tr>
<th>property ( P \subseteq T )</th>
<th>corresponding set of languages ( { L(M) \mid \langle M \rangle \in P } \subseteq R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ \langle M \rangle \mid M \text{ recognizes the language } \emptyset }</td>
<td>{ \emptyset }</td>
</tr>
<tr>
<td>{ \langle M \rangle \mid \text{ on any input, } M \text{ never halts and accepts} }</td>
<td>{ \emptyset }</td>
</tr>
<tr>
<td>{ \langle M \rangle \mid M \text{ halts on only a finite number of input strings} }</td>
<td>{ L \mid L \subseteq \Sigma^* \text{ is finite} }</td>
</tr>
<tr>
<td>{ \langle M \rangle \mid M \text{ halts and rejects the input string } \varepsilon }</td>
<td>{ L \mid L \not\ni \varepsilon }</td>
</tr>
</tbody>
</table>

Examples of sets that are not properties:

- \{ \langle M \rangle \mid M \text{ has more that 3 states} \}
- \{ \langle M \rangle \mid M \text{ accepts some string in } \leq 100 \text{ steps of computation} \}
- \{ \langle M \rangle \mid M \text{ uses } \$ \text{ in its tape alphabet} \}

So a property of a Turing machine is something that is true of the language it recognizes (and not just a trivial feature of the machine). We will want to use the term “property” interchangeably to refer to both a particular set \( P \) of Turing machines and to the set \( \{ L(M) \mid \langle M \rangle \in P \} \) of languages recognized by those machines.

**Definition 2.** A property \( P \) is trivial if \( P = \emptyset \) or \( P = T \).

**Rice’s Theorem.** If \( P \) is any nontrivial property, then \( P \) is undecidable.

**Proof.** For the sake of contradiction, assume that \( P \) is decidable and let \( M_P \) be the Turing machine that decides \( P \): \( M_P \) accepts \( \langle M \rangle \) if \( \langle M \rangle \in P \), and \( M_P \) rejects \( \langle M \rangle \) if \( \langle M \rangle \notin P \). We will use \( M_P \) to build a Turing machine that decides \( \text{HALT}_{TM} \).
Case 1: Suppose there is some $\langle M \rangle \in P$ such that $L(M) = \emptyset$.

Let $M_{\text{nope}}$ be some Turing machine which does not have property $P$: $\langle M_{\text{nope}} \rangle \notin P$. We know that $M_{\text{nope}}$ exists because $P$ is not a trivial property, so there has to be some Turing machine not in $P$.

We design a Turing machine $H$ to decide $\text{HALT}_{TM}$ using $M_P$ as a subroutine. This will be the contradiction we aim for.

$$H = \begin{cases} \text{on input } \langle M, w \rangle \text{ where } M \text{ is a Turing machine and } w \text{ is a string:} \\ (1) \text{ Build a Turing machine } J \text{ as follows:} \\ \quad J = \begin{cases} \text{on input } w: \\ \quad \text{(a) Simulate } M \text{ on } w. \\ \quad \text{(b) Then simulate } M_{\text{nope}} \text{ on } w. \text{ If it accepts, accept. If it rejects, reject.} \end{cases} \\ (2) \text{ Use } M_P \text{ to decide if } \langle J \rangle \notin P. \text{ If it accepts, reject. If it rejects, accept.} \end{cases}$$

Notice that this construction means that either $L(J) = \emptyset$ or $L(J) = L(M_{\text{nope}})$. This Turing machine $H$ was designed with the goal that $H$ should accept $\langle M, w \rangle$ if and only if $\langle J \rangle \notin P$. The idea is that if $P$ is decidable, then machine $H$ can decide the halting problem.

**Claim.** $H$ accepts $\langle M, w \rangle$ if and only if $M$ halts on $w$.

**Proof.** If $M$ halts on $w$ then when $J$ runs, it will finish step (a) and get to run step (b). This means $L(J) = L(M_{\text{nope}})$, so $J$ will not have property $P$ (by the definition of a property). Thus on line (2), the decider $M_P$ will reject $\langle J \rangle$, so $H$ will accept.

If $M$ does not halt on $w$, then $J$ will loop on (a) forever, so it will never accept any string. Thus $L(J) = \emptyset$, so we know that $J$ does have property $P$ (by the definition of a property), so on line (2), the decider $M_P$ will accept, so $H$ will reject.

Observe that $H$ is a decider, since step (1) is simply constructing a Turing machine, and step (2) is running a decider.

Thus $H$ is a decider for $\text{HALT}_{TM}$. Contradiction! $\text{HALT}_{TM}$ is undecidable! \[\Rightarrow \Leftarrow\]

Case 2: Suppose there is no $\langle M \rangle \in P$ such that $L(M) = \emptyset$.

Then a decider for $P$ can be turned into a decider for $\overline{P}$ as follows:

$$Q = \begin{cases} \text{on input } \langle M \rangle \text{ where } M \text{ is a Turing machine:} \\ (1) \text{ Run the decider for } P \text{ on } \langle M \rangle. \\ (2) \text{ If it accepted, reject. If it rejected, accept.} \end{cases}$$

Now we’ve reduced this case to case 1 (we have a decider for a property which contains some $M$ such that $L(M) = \emptyset$), so switch to case 1.