Rice’s Theorem

Recall that a language is a set of strings (over a given alphabet).

**Definition:** A *property of languages* is a set of languages (over a given alphabet).

**Examples:**
- The set of all finite languages
- The set of all infinite languages
- The set of all context-free (or regular, or recursive, or recursively enumerable) languages
- The set of all languages containing at least three strings.

**Definition:** A nontrivial property of languages (over an alphabet $\Sigma$) is a property that neither is empty nor contains every recursively enumerable language.

(I.e., the set contains some languages but excludes others.)

**Counterexample:** The set of all languages containing 0 or more strings is a trivial property.

**Note:** these language-based definitions also apply to sets of numbers, since strings can encode numbers.

**Example:** One property of sets of numbers, with alphabet $\Sigma = \{1\}$: The set of all finite sets of (strings representing) numbers
**Definition:** A property of languages is **decidable** if there is an algorithm (thus, a TM) which recognizes that property.

(That is, there is a TM which always halts, returning 1 if a given language has the property and 0 if the language does not have the property.)

**Definition:** A property of languages is **undecidable** if there is no algorithm (thus, no TM) which recognizes that property.

(That is, there is no TM which always halts, returning 1 if a given language has the property and 0 if the language does not have the property.)
Rice’s Theorem: Any nontrivial property $P$ of the recursively enumerable languages is undecidable.

Proof (by contradiction):

Idea of proof: Show that if we had a machine to decide membership in $P$, we’d have a machine to decide halting.

Without loss of generality, assume $\emptyset \notin P$.

(No generality is lost: if $\emptyset \in P$, then repeat the proof below, replacing $P$ by the complement of $P$ throughout.)

Since $P$ is nontrivial, there exists a language $L$ with property $P$.

Let $M_L$ be a TM accepting $L$.

($M_L$ exists, because $L$ is recursively enumerable.)

Suppose that $P$ is decidable.

Then there is a TM $M_P$ which recognizes $P$.

$M_P$ can be used to determine whether an arbitrary TM $M$ halts on an arbitrary input string $w$:

Starting with $M$, $w$, and $M_L$, construct a TM $M'$:
M′ runs M on w.
If M halts on w, then
   M′ reads an input string x and runs M_L on x.
   If M_L returns 1, M′ returns 1;
   Otherwise (if M doesn’t halt on w or if M_L does not return 1),
   M′ computes forever.

We’ve just constructed one of two machines:
   a machine to accept L (if M halts on w)

or
   a machine to accept ∅ (if M doesn’t halt on w)
   (because if M doesn’t halt on w, then M′ doesn’t halt on
   anything)

Recall that L ∈ P and ∅ ∉ P.
   Thus, L(M′) ∈ P iff M halts on w.

Recall also that we’ve assumed there’s a machine M_P that decides
membership in P.

Thus, we have an algorithm to determine whether M halts on w:
   Just run M_P on L(M′).

But there is no algorithm to determine whether M halts on w — a
contradiction.

Thus, M_P must not exist, and P is not decidable.

□
Applying Rice’s theorem

None of the following properties is decidable; equivalently, none of the following sets is recursive (that is, there is no algorithm for determining, in general, whether or not a language has one of these properties):

- The set of all finite languages
- The set of all infinite languages
- The set of all context-free [or regular, or recursive, or recursively enumerable] languages
- The set of all languages containing at least three strings

Some properties of languages are not even r.e. (that is, there is no algorithm which can answer yes if the language has the property).

Recall that if S and $S^C$ are both r.e., then S is recursive.

This means that if a set S is not recursive, then either S or $S^C$ (or maybe both) is not r.e.

If you can show that a property of languages is r.e., then you know that the complement of that property is not r.e.
Example:

\[ P_1 = \text{“nonemptiness” (the set of nonempty r.e. languages)} \]
\[ P_2 = \text{“emptiness” (the set of empty r.e. languages)} \]

**Property \( P_1 \) is r.e.:**

Suppose we have a machine \( M_L \) which accepts some language \( L \). We want to find out whether \( L \) contains any strings. Using dovetailing, simulate \( M_L \) on every string over the alphabet until \( M_L \) halts and returns a 1 for some string. At that point, halt and return a 1; if that never happens, compute forever.

(Alternatively, we could start with a machine \( M_G \) which generates \( L \); simulate \( M_G \), and return a 1 if \( M_G \) ever generates a string.)

**Because \( P_1 \) is r.e., \( P_2 \) is not r.e.:**

There is no general algorithm that answers yes when its input language is empty.
Extended application of Rice’s theorem

Rice’s theorem addresses properties of a single language.

Properties of pairs (triples, etc.) of languages are not directly addressed by Rice’s theorem, but properties of single languages can sometimes be used to show the undecidability of other problems.

**Example:** “Given TMs $M_1$ and $M_2$, is $L(M_1) = L(M_2)$?” is undecidable.

**Proof:**

Suppose there is a machine $E$ which decides whether $L(M_1) = L(M_2)$, for arbitrary $M_1$ and $M_2$.

Build a machine $N$ for $\emptyset$:
(No matter what the input, erase all characters and write 0.)

Call $E$ with $M$ and $N$ as input.

- If $L(M) = \emptyset$, the machine will halt and return 1
- If $L(M) \neq \emptyset$, the machine will halt and return 0

Thus we have an algorithm to decide “$L(M) = \emptyset$”, for arbitrary $M$.

But there is no such algorithm, by Rice’s theorem.

Therefore, “Is $L(M_1) = L(M_2)$?” is also not decidable.  

Copyright © 2010 Nancy Lynn Tinkham
A note on Rice’s theorem

Rice’s theorem applies to languages. It applies to machines only when all machines for a given language must have the property (or not) together.

Examples:

“Is the language recognized by M finite?”
This is a property of languages, because all TMs that recognize this language must have the same I/O behavior.

“Does M have exactly 5 states?”
This is not a property of languages: different TMs for M’s language might have different numbers of states.
(This question is actually decidable.)
Bibliographic note:

This explanation is based on the lectures and books of

Arnold Rosenberg, professor at Duke University


Thomas Sudkamp, Languages and Machines, 1996.

It differs from the presentation in Taylor, in that it focuses on languages rather than on sets of numbers.