

Lecture 9: Undecidability and Rice's Theorem

David Dill

Department of Computer Science

Outline

- A language that is RE but not recursive
- Rice's theorem

A Language that is RE but not Recursive

$$L_u = \{(\langle M \rangle, w) \mid M \text{ accepts } w\}.$$

Theorem: L_u is RE.

L_u is accepted by the “Universal Turing Machine” M_U .

Idea: To check $(\langle M \rangle, w)$, simulate M on w using one tape to hold $\langle M \rangle$ and a second tape to simulate the tape of M (so the second tape starts off with w and the first tape starts off with $\langle M \rangle$).

Proofs by Reduction

This is one of the most important proof techniques in computer science.

To prove that problem P_2 is hard, show that there is an “easy” reduction from a *known* hard problem P_1 to P_2 .

Then P_2 is at least as hard as P_1 .

Example: (Suppose it is well-known that \langle a student \rangle cannot lift a car.)

Thm \langle a student \rangle cannot lift a loaded truck (P_2).

Proof By reduction from the car-lifting problem.

Suppose \langle a student \rangle could lift a loaded truck.

Then, \langle a student \rangle could solve the car-lifting problem by putting the car on the truck and lifting the truck.

But, it is known that \langle a student \rangle cannot lift a car.

IMPORTANT: Make sure you are doing the reduction in the right direction: *hard problem* \rightarrow *new problem*.

A wrong proof

Thm: \langle a student \rangle cannot lift an anvil.

proof: By reduction to the car-lifting problem.

We can reduce anvil lifting to car-lifting by putting the anvil in a car.

It is known that \langle a student \rangle cannot lift a car.

Therefore, \langle a student \rangle cannot lift an anvil.

It may be true, but this is not a proof because the reduction goes the wrong way. (Substitute “feather” for anvil.)

Theorem: $\overline{L_u}$ is not RE,

Proof sketch:

We can reduce the problem of recognizing $x \in L_d$ to $x \in \overline{L_u}$ as follows:

A TM, M , accepting $\overline{L_u}$ would input $(\langle M_i \rangle, w)$ and accept iff M_i does not accept w .

We could construct a TM M' to accept L_d , as follows:

Given input string $\langle M_i \rangle$, M' first constructs the string $(\langle M_i \rangle, \langle M_i \rangle)$ and then runs M on it.

$L(M') = L_d$, since M' accepts $\langle M_i \rangle$ iff M_i does *not* accept $\langle M_i \rangle$.

But L_d is not RE, so $\overline{L_u}$ is not RE.

L_u is not Recursive

Theorem: L_u is not recursive.

Proof sketch:

$\overline{L_u}$ is not recursive, so L_u must not be recursive (since the complement of a recursive language must be recursive).

Other Properties of Turing Machines

Problem: Is $L(M_i)$ empty?

$$L_e = \{M_i \mid L(M_i) = \emptyset\}.$$

$$L_{ne} = \{M_i \mid L(M_i) \neq \emptyset\}.$$

Theorem L_{ne} is RE.

Proof Use an NTM.

Use nondeterministic choice to *guess* a string w

- Repeatedly guess a symbol or “stop guessing input”
- After “stop guessing,” simulate M_i on w , and accept if M_i accepts

If $L(M_i) \neq \emptyset$, there will be some w that accepts, so our NTM will have an accepting computation.

(Of course, we could do this on a deterministic TM as well by simulating M_i with a DTM.)

L_{ne} is not recursive

Theorem L_{ne} is not recursive.

Proof sketch:

We prove by reduction from L_u .

Suppose we had a machine M_{ne} that decided L_{ne} .

Approach: Build a TM M_u that

1. Takes as input $(\langle M \rangle, w)$
2. Manufactures (see below) a TM $\langle M' \rangle$ s.t. $L(M') \neq \emptyset$ iff M accepts w .
3. Uses $\langle M_{ne} \rangle$ to check whether $L(M') \neq \emptyset$.

So $L(M_u) = \{(\langle M \rangle, w) \mid M \text{ accepts } w\} = L_u$.

M' takes an arbitrary string x as input.

1. M' ignores x
2. M' simulates M on w .
3. If M accepts w , M' accepts

$L(M') = \Sigma^*$ if M accepts w , and $L(M') = \emptyset$ if M does not accept w .

L_e is not RE

By a previous theorem, if L and \bar{L} are both RE, L is recursive.

Since L_{ne} is RE and not recursive, L_e must not be RE.

Rice's Theorem

Rice's theorem is a “weapon of mass destruction” for decidability problems.

To a first approximation, it says “no interesting property of a Turing Machine language is decidable.”

The proof is a generalization of the recent proof we did of the undecidability of L_{ne} .

Statement of Rice's theorem

Thm: Every non-trivial property P of the R.E. languages is undecidable.

(a). *Property:* A set of languages (in this case, a set of Turing Machines).

(b). *Trivial:* Either every language has it or no language has it.

Since elements of a property P are R.E. languages, that are represented by strings, we can consider P to be the set of encodings of these TMs.

But it's not an arbitrary set, because if $L(M_1) = L(M_2)$, then $\langle M_1 \rangle \in P \iff \langle M_2 \rangle \in P$.

Then P is *just* a set of strings.

Q: What is difference between properties \emptyset and $\{\emptyset\}$?

Proof of Rice's theorem

PROOF:

Let P be a non-trivial property of the R.E. languages, i.e., a set of Turing Machines such that $P \neq \emptyset$ and $\overline{P} \neq \emptyset$.

We prove by reduction from L_u , which is known to be non-recursive.

Case (i): $\emptyset \notin P$.

Suppose there were a TM M_P that decided P . We could then construct a TM that decides L_u .

As we show below, given $(\langle M \rangle, w)$, it is possible to construct another TM M' such that $M' \in P$ iff $(\langle M \rangle, w) \in L_u$ (i.e., M accepts w). Given M_P , we can then construct a TM that decides L_u : the Turing machine transforms $(\langle M \rangle, w)$ to M' , then runs M_P on M' , and accepts iff M_P does.

Now, the details of the construction of M' : $P \neq \emptyset$, so it contains some strings. Let L be the language L , and $L \neq \emptyset$ because $\emptyset \notin P$.

Let M_L be a TM accepting L .

M' takes an input x . It first simulates M on w . If M accepts w then M' simulates M_L on x . If M_L accepts w , then M' accepts. Otherwise,

Proof of Rice's theorem (2)

There are two possibilities for M' .

(1) If M accepts w , then M' will eventually halt the simulation of $(\langle M \rangle, w)$ and then simulate M_L . In this case, the language of M' is the language L , which is in property P .

(2) If M does not accept w , then M' rejects every input (without even simulating it!). Hence, its language is \emptyset , which is not in property P by assumption.

Consequently, deciding whether M' is in the property P or not would decide whether M accepts w , i.e. whether $(\langle M \rangle, w) \in L_u$.

So, if $\emptyset \notin P$, then P is undecidable.

Proof of Rice's theorem (3)

case (ii): $\emptyset \in P$. Consider the property \overline{P} .

This property contains exactly those RE languages that P does not.

Since P is non-trivial, so is \overline{P} . $\emptyset \in P$ because $\emptyset \notin \overline{P}$, so case (i) is not recursive.

Since recursive languages are closed under complement, P cannot be recursive and hence is undecidable.

Applications of Rice's theorem

Whether a language of a TM is empty is undecidable.

Whether a language of a TM is non-empty undecidable.

Whether a language of a TM is regular is undecidable.

Whether a language of a TM is context-free is undecidable.

WRONG Applications of Rice's theorem

Rice's theorem cannot be used for these:

- Whether a TM has less than 7 states,
- Whether a TM has a final state,
- Whether a TM has a start state.
- Whether the language is RE.
- Whether the language is L_d .

(Note how these properties are not properties of languages, but properties of TMs.)