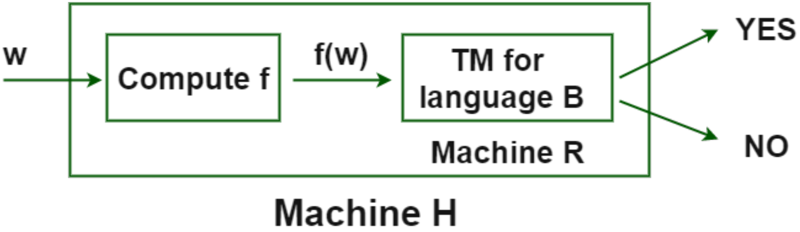
Reducibility

Textbook: Chapter 5



Recall: A_{TM}

- A_{TM} is the set of all TM-input pairs $\langle M, w \rangle$ such that M **A**ccepts w
- Church and Turing proved A_{TM} is undecidable by contradiction

It's harder to find contradictions in other undecidable languages:
We need another way!

Informal definition of reducibility

- Some problems can be “reduced” to each other

Def: Reducibility. For two problems A and B, if A **reduces** to B, we can use a solution to B to solve A.

- Ex: “Getting to Jim’s house” reduces to “knowing where Jim lives” because you can use that knowledge to derive directions

Def: Reducibility in TMs. If, for languages A and B, a decider for B can be used to decide A, A is said to be **reducible** to B.

Corollary: If A is reducible to B (a solution to B will solve A): 1. If B is decidable, so is A 2. If B is undecidable, so is A

Ex: The halting problem

Def: The halting problem. Does a TM halt given its input? Formally:

$$HALT_{TM} = \{\langle M, w\rangle : M \texttt{ is a TM which halts on input } w\}$$

This is similar to A_{TM} (the set of all TM-input pairs that accept), but also allows rejecting states.

Thm. $HALT_{TM}$ is undecidable.

Halting problem undecidability proof

Pf. By contradiction. We will assume $HALT_{TM}$ is decidable and show that its decider can be used to decide A_{TM} (a known contradiction). This means A_{TM} is reducible to $HALT_{TM}$.

Assume we have a TM R that decides $HALT_{TM}$. We construct a new TM S to decide A_{TM} using the output of R (recall that this is legal for TMs).

$S =$ "On input $\langle M, w\rangle$ where M is a TM and w is its input:

1. Run R on input $\langle M, w\rangle$
2. If R rejects, the input never halts. **Reject.**
3. Otherwise, simulate M on w until it halts
4. If M accepted, **accept**. Otherwise, **reject.**"

Thus, the existence of R implies that A_{TM} can be decided: a contradiction. Thus, R cannot exist. End of proof.

E_{TM}: The emptiness problem

$$E_{TM} = \{\langle M, w\rangle : M \texttt{ is a TM and } \mathcal{L}(M) = \emptyset\}$$

Thm: E_{TM} is undecidable. **Pf:** By reduction. Let R be a TM deciding E_{TM}.

Let M_1 be a TM operating on x **given some** w **and** M **(not input)** such that:

1. If $x \neq w$, **reject**
2. If $x = w$, run M on input w and **accept** if M does

Let S be a TM operating on w such that:

1. Use M and w to construct the TM M_1
2. Run R on $\langle M_1\rangle$, accepting iff M_1's language is empty
3. If R accepts (M_1 is nonempty, meaning M rejects w), **reject.** If R rejects, **accept.**

Thus, A_{TM} is reducible to E_{TM} and thus E_{TM} is undecidable.

$REGULAR_{TM}$: Whether or not a TM has an equivalent NFA

$$REGULAR_{TM} = \{\langle M \rangle : M \texttt{ is a TM and } \mathcal{L}(M) \texttt{ is regular}\}$$

$S =$ "On input $\langle M, w \rangle$ where M is a TM and w is its input:

1. Construct the following TM M_1.

   $M_1 =$ "On input string x:

   1.1 If x is of the form $0^i 1^i$, **accept.**
   1.2 Else, **accept** if M accepts w."

   Note that M_1 recognizes the nonregular language $0^i 1^i$ if M does not accept w and the regular language Σ^* if M accepts w.

2. Run R on input $\langle M_1 \rangle$

3. If R accepts, **accept**. If R rejects, **reject.**"

EQ_{TM}: TM equivalence

$$EQ_{TM} = \{\langle M_1, M_2\rangle : M_1, M_2 \texttt{ are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$$

We show that E_{TM} (the emptiness problem) reduces to EQ_{TM} (the equivalence problem) and thus the later is undecidable.

Let R be a TM deciding EQ_{TM}. We will use it to construct S deciding E_{TM}.

$S =$ "On input $\langle M\rangle$, where M is a TM:

1. Let $M_\emptyset$ be the TM rejecting all inputs.
2. Run R on $\langle M, M_\emptyset\rangle$.
3. If R accepted, **accept.** If R rejected, **reject.**"

Since S decides E_{TM}, R cannot exist.

Computation histories

- Recall: An automaton has some number of **configurations** at any given time
  - A DFA configuration is its state $q \in Q$
  - A TM configuration has its state, position, and tape contents, formatted like: uqv for string contents uv, state q, where the R/W head is on the first character of v

Def: A computation history for some automaton is a series of its configurations.

Def: An **accepting computation history** is a computation history which represents a valid series of state transitions ending in an accept state.

Reductions via computation histories

- Computation histories encode the entire operation of an automaton
- Assumptions about them generalize to their machines
- Can be used to prove properties
- We will come back to this after talking about LBA in section 3

Mapping reducibility and computable functions

Def: Computable functions. A function $f : \Sigma^* \to \Sigma^*$ is **computable** if some TM M, for every input w, **halts** with $f(w)$ on its tape.

Def: Mapping reducibility. Language A is **mapping reducible** to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$ where, for every w,

$$w \in A \iff f(w) \in B$$

The function f is called the **reduction** of A to B.

This formalizes the intuitive notion of reduction.

Rice's theorem

Thm: Rice's theorem. Any property P of a TM's language such that the following two conditions hold is **undecidable**.

1. $P \neq \emptyset$ and $\overline{P} \neq \emptyset$ (P is nontrivial)
2. $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ implies that $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$ (P is a semantic property).

Pf: By reduction. Assume R is a TM deciding some arbitrary nontrivial property P. We will use R to build a decider for $HALT_{TM}$.

Let M_P be some TM such that $\langle M_P \rangle \in P$ (guaranteed by condition 1). We construct some S using M_P and R to decide $HALT_{TM}$.

Rice's theorem proof

$S =$ "On input $\langle M, w \rangle$, where M is a TM:

1. Construct M_1 to be a TM that simulates M on w. As soon as M halts, M_1 **simulates** M_P. Therefore, M_1 is $\in P$ iff M halts on w.
2. Run our decider R on M_1. If R accepts, M must halt on w: **accept**. If not, M_1 must not be $\in P$ and therefore M must not halt: **reject**."

Since R allows us to decide the halting problem, its language must be undecidable. End of proof.

Next time: Post's Correspondence Problem