

Read Section 12.1.

Computability A function f with domain D is *computable* if there exists some TM M such that M computes f for all values in its domain.

Decidability A problem is *decidable* if there exists a TM that can answer yes or no to every statement in the domain of the problem.

The Halting Problem

Domain: set of all TMs and all strings w .

Question: Given coding of M and w , does M halt on w ? (yes or no)

Theorem The halting problem is undecidable.

Proof: (by contradiction)

- Assume there is a TM H (or algorithm) that solves this problem.

TM H has 2 final states, q_y represents yes and q_n represents no.

TM H has input the coding of TM M (denoted w_M) and input string w and ends in state q_y (yes) if M halts on w and ends in state q_n (no) if M doesn't halt on w .

$$H(w_M, w) = \begin{cases} \text{(yes) halts in } q_y & \text{if } M \text{ halts on } w \\ \text{(no) halts in } q_n & \text{if } M \text{ doesn't halt on } w \end{cases}$$

TM H always halts in a final state.

Construct TM H' from H such that H' halts if H ends in state q_n and H' doesn't halt if H ends in state q_y .

$$H'(w_M, w) = \begin{cases} \text{halts} & \text{if } M \text{ doesn't halt on } w \\ \text{doesn't halt} & \text{if } M \text{ halts on } w \end{cases}$$

Construct TM \hat{H} from H' such that \hat{H} makes a copy of w_M and then behaves like H' . (simulates TM M on the input string that is the encoding of TM M , applies M_w to M_w).

So $\hat{H}(w_M)$ runs $H'(w_M, w_M)$

$$\hat{H}(w_M) = \begin{cases} \text{halts} & \text{if } M \text{ doesn't halt on } w_M \\ \text{doesn't halt} & \text{if } M \text{ halts on } w_M \end{cases}$$

Note that \hat{H} is a TM.

There is some encoding of it, say $\hat{w}_{\hat{H}}$.

What happens if we run \hat{H} with input $\hat{w}_{\hat{H}}$?

Theorem If the halting problem were decidable, then every recursively enumerable language would be recursive. Thus, the halting problem is undecidable.

- **Proof:** Let L be an RE language over Σ .

Let M be the TM such that $L=L(M)$.

Let H be the TM that solves the halting problem.

A problem A is *reduced* to problem B if the decidability of B follows from the decidability of A. Then if we know B is undecidable, then A must be undecidable.

State-entry problem Given TM $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$, state $q \in Q$, and string $w \in \Sigma^*$, is state q ever entered when M is applied to w ?

This is an undecidable problem!

- **Proof:** We will reduce this problem to the halting problem.

Suppose we have a TM E to solve the state-entry problem.

TM E takes as input the coding of a TM M (denoted by w_M), a string w and a state q . TM E answers *yes* if state q is entered and *no* if state q is not entered.

Construct TM E' which does the following. On input w_M and w E' first examines the transition functions of M. Whenever δ is not defined for some state q_i and symbol a add the transition $\delta(q_i, a) = (q, a, R)$. Let this new state q be the only final state. Let M' be the modified TM. Next, simulate TM E on input $w_{M'}$, w and q .

$$E'(w_M, w) = \begin{cases} \text{M halts on } w & \text{if M' enters state } q \\ \text{M doesn't halt on } w & \text{if M' doesn't enter state } q \end{cases}$$

TM E' determines if M halts on w . If M halts on w then TM E' will enter state q in M' and answer *yes*. If M doesn't halt on w then TM E' will not enter state q , so it will answer *no*. Since the state-entry problem is decidable, E always gives an answer yes or no.

But the halting problem is undecidable. Contradiction! Thus, the state-entry problem must be undecidable. QED.

There are some more examples of undecidability in section 12.1.