CSC 320 - Lecture 14

#decider #decidable #existence #undecidable #undecidability #halting-problem #reduction #decidable-languages #undecidable-languages #countability #languages

---

We Know

A language L is Turing-Recognizable
if and only if
some (Single-Tape) Turing Machine recognizes L
if and only if
some Multitape Turing Machine recognizes L
if and only if
some Enumerator outputs L
if and only if
some Nondeterministic Turing Machine outputs L

Decidable Problems / Languages

Show. $A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ is decidable.

TM M that decides A_{DFA}. Use high-level description that describes M.

- M = "On input $\langle B, w\rangle$, where B is a DFA and w is a string:
  - Simulate B on input w
    - If simulation of B ends in accept state: accept
    - If simulation of B ends in non-accepting state: reject."

Why does M work / how can M be realized?

- Input for M: $Q, \Sigma, \delta, q_0, F, w$
  - Verify input for DFA and w
- At any step of simulation: M keeps track of B's current state & position in w...
  - M writes information on tape

Note. TM M is a decider.

TM M always halts since the simulation of DFA B *halts*. And because it is a well defined DFA and we expect it to be proper (otherwise we would have problems).

Show. $A_{NFA} = \{\langle B, w\rangle \mid B \text{ is a NFA that accepts input string } w\}$ is decidable.

TM N that decides A_{NFA}. Use high-level description that describes N.

- N = "On input $\langle B, w\rangle$, where B is a NFA and w is a string:
  - Convert NFA B to an equivalent DFA C
  - Run TM M on input $\langle C, w\rangle$
    - If M accepts, accept; otherwise, reject."

Note. N is a decider.

Note. This is not cumbersome because we are using the theorem and algorithms previously seen in this course.

Are All Languages Decidable?

Recall. How large is the set of all languages?

The set of all languages over Σ equals $\mathcal{P}(\Sigma^*)$. Σ^* is countable infinite and therefore $\mathcal{P}(\Sigma^*)$ is uncountably infinite. (The set of all languages in uncountable infinite).

Question. How many Turing Machines are there?

The Set of All Turing Machines is Countable

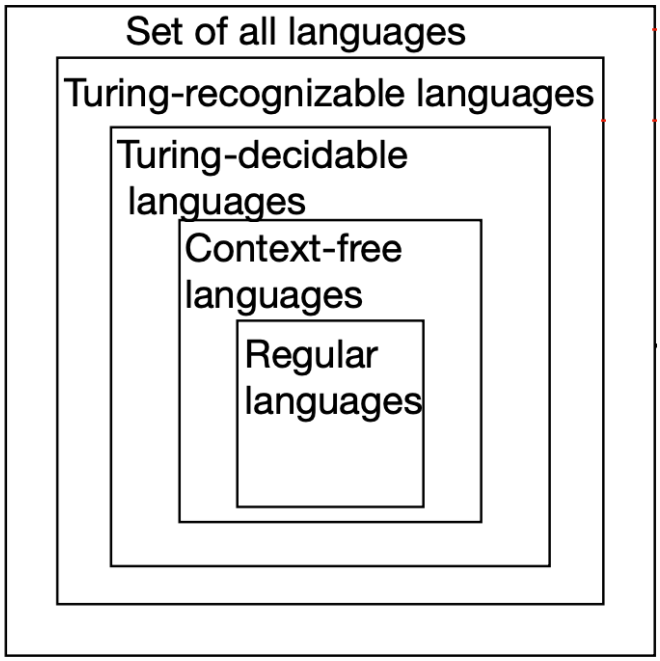
Turing Machines can be encoded over some alphabet Σ. Each TM M has encoding $\langle M\rangle$. Enumerate all strings in Σ^*, omitting strings that don't encode any TM.

There Exist Languages That Cannot Be Decided

Because there are more languages (the set of all languages is uncountable infinite) than Turing Machines (the set of Turing Machines is countable).

Thus, there are more languages than Turing Machines.

Languages



The set of all languages is uncountable, and everything within and including Turing-Recognizable is countable.

Note. There exist languages that are not Turing-Recognizable. Thus, there are also languages that are not Turing-Decidable, etc.

The Halting Problem

Does there exist a decider / an algorithm that can check any code / program with any input and determine if it halts?

$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$.

Note. It would be cool if there was a universal checking mechanism, but that doesn't exist.

Methods to Prove Undecidability (for Halting Problem)

- Diagonalisation (seen in Lecture 02).
- Reduction

$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a } TM \text{ and accepts input string } w\}$

We will use A_{TM} to help us prove undecidability of Halting problem.

Question. Is A_{TM} decidable? NO!

An Undecidable Language

$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a } TM \text{ and accepts input string } w\}$

Show. A_{TM} is Turing-Recognizable and is undecidable.

A_{TM} is Turing Recognizable

- $U =$ "On input $\langle M, w\rangle$, where M is TM and w is string:
  - Simulate M on input w...
    - If M enters accept state, accept
    - If M ever enters reject state, reject."

Note. U is called universal TM. U is no decider. U loops whenever M loops.

A_{TM} is Undecidable

Proof by contradiction, using diagonalisation: Assume A_{TM} is decidable.

Assume H is a decider for A_{TM}. On input $\langle M, w\rangle$, where M is TM and w is string:

- H halts and accepts if M accepts w
- H halts and rejects if M fails to accept w

That is H is a TM with...

$$H(\langle M, w\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

Construct TM D with H as a subroutine...

- $D =$ "On input $\langle M\rangle$, where M is TM
  - Run H on input $\langle M, \langle M\rangle\rangle$
  - If H accepts: reject
  - If H rejects: accept"

$$D(\langle M\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accepts } \langle M\rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M\rangle \end{cases}$$

$$D(\langle D\rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accepts } \langle D\rangle \\ \text{reject} & \text{if } D \text{ accepts } \langle D\rangle \end{cases}$$

Which is a contradiction! So, A_{TM} is undecidable.

A_{TM} is Undecidable (Reconsidered)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle D \rangle$
M_1	accept	*reject*	accept	*reject*	
M_2	accept	*reject*	*reject*	*reject*	
M_3	*reject*	*reject*	accept	*reject*	
M_4	*reject*	*reject*	*reject*	*reject*	
D	*reject*	*accept*	*reject*	*accept*	??

Note. We go in a diagonal to get D. And we can't do the last diagonalisation. So we obtain a contradiction.

- On input $\langle M, w \rangle$ for TM M...
- On input $\langle D, \langle D \rangle \rangle$: ($D$ on input $\langle D \rangle$)
    - and string w
        - H accepts if D accepts $\langle D \rangle$
        - H accepts if M accepts w
        - H rejects if D does not accept $\langle D \rangle$
        - H rejects if M does not accept w

D does the opposite of H, that is:

- $D(\langle D \rangle) =$ accept if $H(\langle D, \langle D \rangle \rangle) =$ reject
- $D(\langle D \rangle) =$ reject if $H(\langle D, \langle D \rangle \rangle) =$ accept

That is, D accepts when given as input its own encoding **if and only if** D rejects when given as inputs its own encoding. Which is a contradiction!

What Happened?!? We Have Shown...

Assuming that A_{TM} is decidable

- We built decider D that uses H, the decider for A_{TM}, as subroutine
- D on input $\langle M \rangle$ simply does the opposite of H on input $\langle M, \langle M \rangle \rangle$
- But: D on input $\langle D \rangle$ cannot return any of accept or reject
- Therefore decider D does not exists, which means that
- Decider H does not exists which meant that

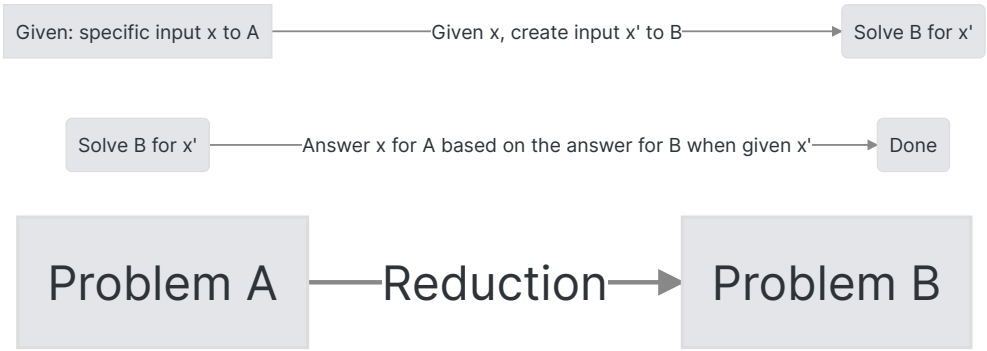
A_{TM} is undecidable.

The Halting Problem (Again)

$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$.

We show that $HALT_{TM}$ is undecidable via the method of **reduction** (and the knowledge that A_{TM} is undecidable).

Reduction



Note. B is used as a subroutine to solve A.

Questions. How can I solve A? Maybe what we know about B can help? Just figure out how to solve A using B!

Definition. Reduction is a way of converting a problem, A, to another problem, B, such that a solution to B can be used to solve A.

- If A **reduces** to B (or A is **reducible** to B), we can use a solution to B to solve A.

Note. That reducibility says nothing about solving A or B alone, but only about solvability of A in **presence** of a solution to B.

- Therefore, If A is reducible to B then...
  - solving **A cannot be harder than solving B**
  - a solution to B yields a solution to A.

Proving Undecidability

If A is reducible to B and B is decidable then A is decidable.

$$\begin{array}{c} A \longrightarrow B \\ \text{Decidable} \Leftarrow \text{Decidable} \end{array}$$

If A is undecidable and A is reducible to B then B is undecidable.

$$A \longrightarrow B$$
$$\text{Undecidable} \Rightarrow \text{Undecidable}$$

A method for proving that a problem is undecidable is to show that some other problem, which is already known to be undecidable, reduces to it.

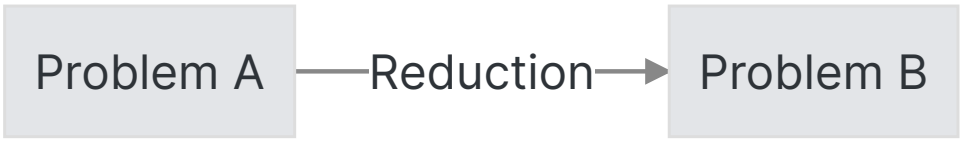
Ex.

$$A_{TM} \longrightarrow HALTH_{TM}$$
$$\text{Decidable} \Leftarrow \text{Decidable}$$

BUT. This is a contradiction because we know A_{TM} is undecidable.

Reduction to Prove Undecidability

Problem A is known to be undecidable...



Reduction comes in form of a decider / algorithm.

Questions. How can I show B is undecidable? Maybe we use that A is undecidable... Yes! Show that if B is decidable then so is A! Use deciding B as subroutine (ex. function) to decide A. This shows that B is at least as difficult as A. And therefore B must be undecidable also!

$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ is Undecidable

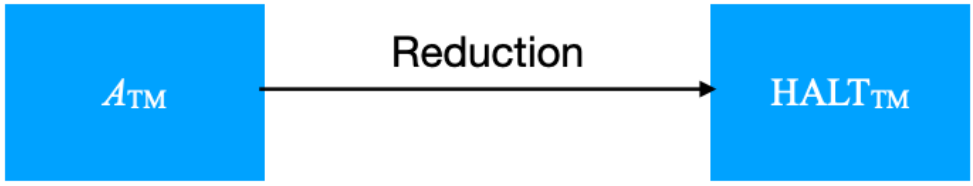
Show. A_{TM} is reducible to $HALT_{TM}$.

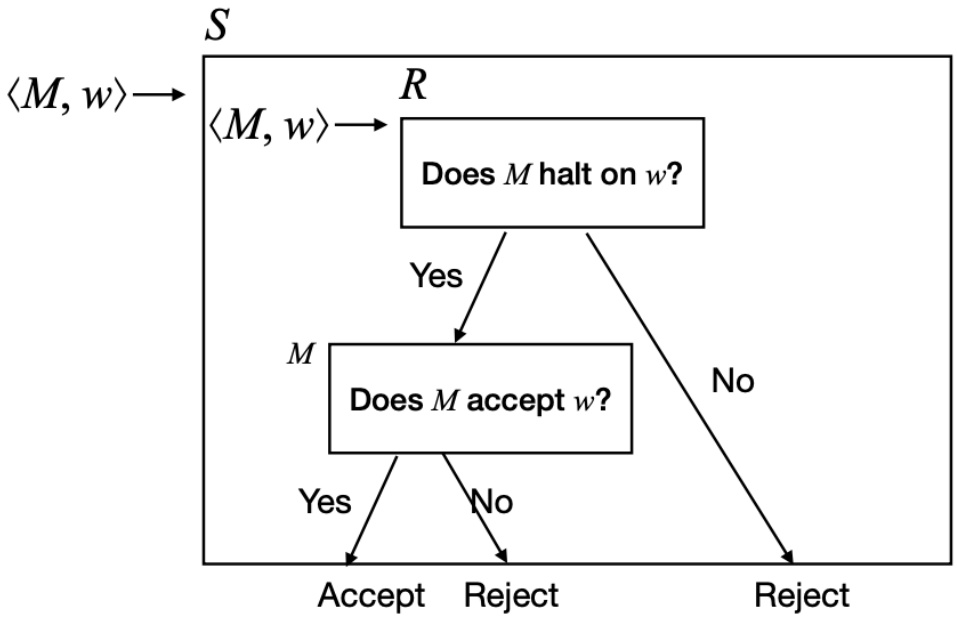
Assume. $HALT_{TM}$ is decidable.

Goal. Show then A_{TM} is decidable also (which we know is not true). (Always use a contradiction when we do this kind of reduction).

Since $HALT_{TM}$ is decidable there is a decider R for $HALT_{TM}$.

The Reduction.





- We construct decider S for A_{TM}. Given input $\langle M, w \rangle$...
  - S must accept if M accepts w and
  - S must reject if M loops for w or rejects w.

Show $HALT_{TM}$ is Undecidable

- We can simulate M on w...
  - If M accepts or rejects, S does the same.
  - **BUT**. What if M is stuck in a loop?

Idea. Use decider R for $HALT_{TM}$ to test whether or not M halts on w.

- If R reports that M does not halt on w then $\langle M, w \rangle \notin A_{TM}$ and therefore S must reject.
- If R reports that M halts on w, simulate M and if M accepts or rejects, S does the same.

Therefore, If decider R for $HALT_{TM}$ exists, then we also can decide A_{TM}.

- But A_{TM} is undecidable...
  - Contradiction
  - Therefore R does not exist

- Therefore $HALT_{TM}$ is undecidable

Undecidable Language

- A_{TM}
- $HALT_{TM}$

Previous Lecture

Lecture13

Next Lecture

Lecture15