Undecidability and Reductions

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Fall 2020

Chinese University of Hong Kong

Undecidability

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid \text{Turing machine } M \text{ accepts input } w\}$$

Turing's Theorem

The language A_{TM} is undecidable

Note: a Turing machine M may take as input its own description $\langle M\rangle$

Turing's Theorem: Proof sketch (in Python)

Suppose function $H(M)$ correctly decides whether program M halts, given its source code $\langle M \rangle$

```
>>> M = "x = 1"
>>> print(H(M))
True
```

```
>>> M = """
while True:
  continue
"""
>>> print(H(M))
False
```

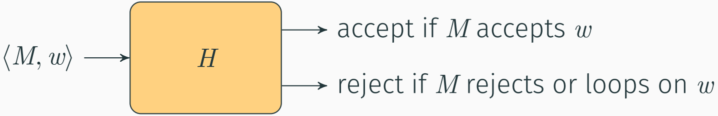
D checks whether itself halts using H and does the opposite

```
def D():
  if H(D):
    loop_forever()
```

Does D halt?

Proof by contradiction:

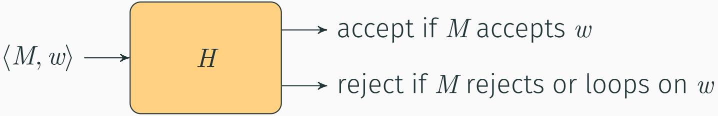
Suppose A_{TM} is decidable, then some TM H decides A_{TM}:



Formal proof of Turing's Theorem

Proof by contradiction:

Suppose A_{TM} is decidable, then some TM H decides A_{TM}:

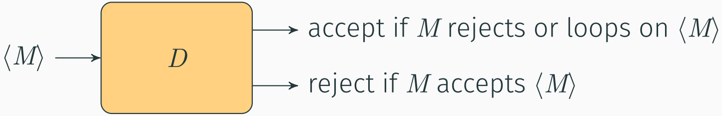


Construct a new TM D (that uses H as a subroutine):

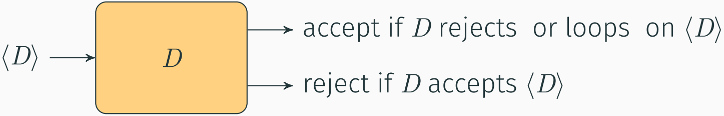
Turing machine D: On input $\langle M \rangle$

1. Run H on input $\langle M, \langle M \rangle \rangle$
2. Output the opposite of H: If H accepts, reject; if H rejects, accept

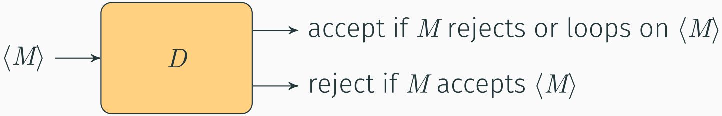
Formal proof of Turing's Theorem



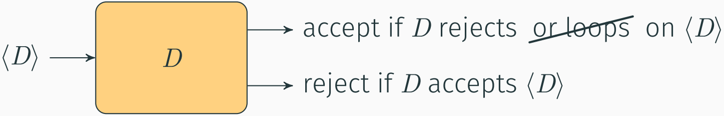
What happens when $M = D$?



Formal proof of Turing's Theorem



What happens when $M = D$?



H never loops indefinitely, neither does D

If D rejects $\langle D \rangle$, then D accepts $\langle D \rangle$

If D accepts $\langle D \rangle$, then D rejects $\langle D \rangle$

Contradiction! D cannot exist! H cannot exist!

Proof of Turing's theorem: conclusion

Proof by contradiction

Assume A_{TM} is decidable

Then there are TM H, H' and D

But D cannot exist!

Conclusion

The language A_{TM} is undecidable

Diagonalization

		all possible inputs w				
		ε	`0`	`1`	`00`	…
all possible Turing machines	M_1	acc	rej	rej	acc	
	M_2	rej	acc	loop	rej	…
	M_3	rej	loop	rej	rej	
	M_4	acc	rej	acc	loop	
			⋮			

Write an infinite table for the pairs (M, w)

(Entries in this table are all made up for illustration)

Diagonalization

		inputs w				
		$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
all possible Turing machines	M_1	acc	loop	rej	rej	
	M_2	rej	rej	acc	rej	...
	M_3	loop	acc	acc	acc	
	M_4	acc	acc	loop	acc	
			$\vdots$			

Only look at those w that describe Turing machines

Diagonalization

		inputs w				
		$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
all possible Turing machines	M_1	acc	loop	rej	rej	
	M_2	rej	rej	acc	rej	...
	M_3	loop	acc	acc	acc	
	$\vdots$		$\vdots$			
	D	rej	acc	rej	rej	
	$\vdots$		$\vdots$			

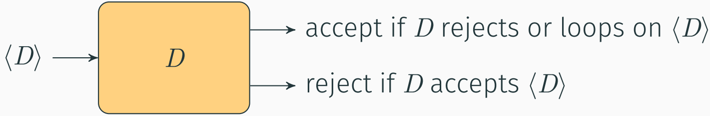
If A_{TM} is decidable, then TM D is in the table

Diagonalization

all possible Turing machines	inputs w: $\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\dots$
M_1	$\boxed{\text{acc}}$	loop	rej	rej	
M_2	rej	$\boxed{\text{rej}}$	acc	rej	$\dots$
M_3	loop	acc	$\boxed{\text{acc}}$	acc	
$\vdots$		$\vdots$			
D	rej	acc	rej	rej	
$\vdots$		$\vdots$			

D does the opposite of the diagonal entries

D on $\langle M_i \rangle$ = opposite of M_i on $\langle M_i \rangle$



Diagonalization

		inputs w					
		$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
all possible Turing machines	M_1	$\boxed{\text{acc}}$	loop	rej	rej		loop
	M_2	rej	$\boxed{\text{rej}}$	acc	rej	...	acc
	M_3	loop	acc	$\boxed{\text{acc}}$	acc		rej
	$\vdots$		$\vdots$				
	D	rej	acc	rej	rej		?
	$\vdots$		$\vdots$				

We run into trouble when we look at $(D, \langle D \rangle)$

Unrecognizable languages

The language A_{TM} is recognizable but not decidable

How about languages that are not recognizable?

$$\begin{aligned}\overline{A_{\text{TM}}} &= \{\langle M, w\rangle \mid M \text{ is a TM that does not accept } w\} \\ &= \{\langle M, w\rangle \mid M \text{ rejects or loops on input } w\}\end{aligned}$$

Claim

The language $\overline{A_{\text{TM}}}$ is not recognizable

Unrecognizable languages

Theorem

If L and $\overline{L}$ are both recognizable, then L is decidable

Proof of Claim from Theorem:

We know A_{TM} is recognizable

if $\overline{A_{\mathsf{TM}}}$ were also, then A_{TM} would be decidable

But Turing's Theorem says A_{TM} is not decidable

Unrecognizable languages

Theorem

If L and $\overline{L}$ are both recognizable, then L is decidable

Proof idea (flawed):

Let $M =$ TM recognizing L, $\quad M' =$ TM recognizing $\overline{L}$

The following Turing machine N decides L:

Turing machine N: On input w

1. Simulate M on input w. If M accepts, accept
2. Simulate M' on input w. If M' accepts, reject

Theorem

If L and $\overline{L}$ are both recognizable, then L is decidable

Proof idea (flawed):

Let $M =$ TM recognizing L, $M' =$ TM recognizing $\overline{L}$

The following Turing machine N decides L:

Turing machine N: On input w

1. Simulate M on input w. If M accepts, accept
2. Simulate M' on input w. If M' accepts, reject

Problem: If M loops on w, we will never go to step 2

Unrecognizable languages

Theorem

If L and $\overline{L}$ are both recognizable, then L is decidable

Proof idea (2nd attempt):

Let $M =$ TM recognizing L, $\qquad M' =$ TM recognizing $\overline{L}$

The following Turing machine N decides L:

Turing machine N: On input w

For $t = 0, 1, 2, 3, \ldots$
- Simulate first t transitions of M on input w.
- If M accepts, accept
- Simulate first t transitions of M' on input w.
- If M' accepts, reject

Reductions

Reductions

Reducing B to A

Transform program R that solves A into program S that solves B

If you can reduce B to A

Then you can solve problem B using subroutine R as a blackbox

Example from Lecture 16:

$A_{\text{DFA}} = \{\langle D, w\rangle \mid D \text{ is a DFA that accepts input } w\}$

$A_{\text{NFA}} = \{\langle N, w\rangle \mid N \text{ is an NFA that accepts input } w\}$

A_{NFA} reduces to A_{DFA} (by converting NFA into DFA)

Reductions in this course

If language B reduces to language A, and B is undecidable then A is also undecidable

Steps for showing a language A to be undecidable:

1. If some TM R decides A
2. Using R, build another TM S that decides $B = A_{\mathsf{TM}}$

But by Turing's theorem, A_{TM} is not decidable

Another undecidable language

$\text{HALT}_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on input } w\}$

We'll show:

HALT_{TM} is an undecidable language

We will argue that

If HALT_{TM} is decidable, then so is A_{TM}

Undecidability of halting

If $\mathsf{HALT}_{\mathsf{TM}}$ can be decided, so can A_{TM}

$$\mathsf{HALT}_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on input } w\}$$
$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input } w\}$$

Suppose $\mathsf{HALT}_{\mathsf{TM}}$ is decidable by a Turing machine H

Then the following TM S decides A_{TM}

Turing machine S: On input $\langle M, w\rangle$

Run H on input $\langle M, w\rangle$
If H rejects, reject
If H accepts, run the universal TM U on input $\langle M, w\rangle$
  If U accepts, accept; else reject

Example 1

$$A'_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts input } \varepsilon\}$$

Is A'_{TM} decidable? Why?

Example 1

$$A'_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM that accepts input } \varepsilon\}$$

Is A'_{TM} decidable? Why?

Undecidable!

Intuitive reason:

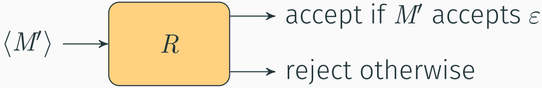
To know whether M accepts ε seems to require simulating M

But then we need to know whether M halts

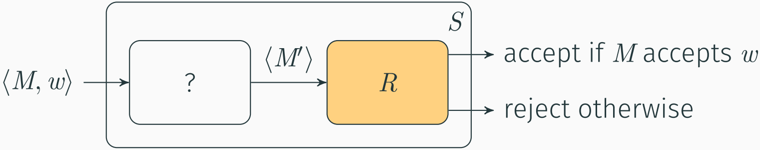
Let's justify this intuition

Example 1: Figuring out the reduction

Suppose A'_{TM} can be decided by a TM R



We want to build a TM S



M' should be a Turing machine such that

outcome of M' on input ε $=$ outcome of M on input w

Example 1: Implementing the reduction

$$\langle M, w\rangle \longrightarrow \boxed{\quad ? \quad} \longrightarrow \langle M'\rangle$$

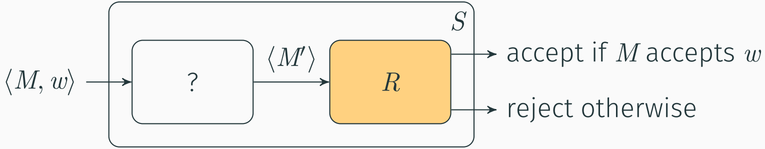
M' should be a Turing machine such that

M' on input $\varepsilon = M$ on input w

Turing machine M': On input z

1. Simulate M on input w
2. If M accepts w, accept
3. If M rejects w, reject

- If M accepts w, M' accepts ε
- If M rejects w, M' rejects ε
- If M loops on w, M' loops on ε



Turing machine S: On input $\langle M, w \rangle$ where M is a TM

1. Construct the following TM M':

   M' = a TM such that on input z,

   Simulate M on input w and accept/reject according to M

2. Run R on input $\langle M' \rangle$ and accept/reject according to R

Example 1: The formal proof

$$A'_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts input } \varepsilon\}$$

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

Suppose A'_{TM} is decidable by a TM R
Consider the TM S:

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

1. Construct the following TM M':

   M' = a TM such that on input z,

   Simulate M on input w and accept/reject according to M
2. Run R on input $\langle M'\rangle$ and accept/reject according to R

Then S accepts $\langle M, w\rangle$ if and only if M accepts w
So S decides A_{TM}, which is impossible

Example 2

$$A''_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM that accepts some input strings}\}$$

Is A''_{TM} decidable? Why?

Undecidable!

Intuitive reason:

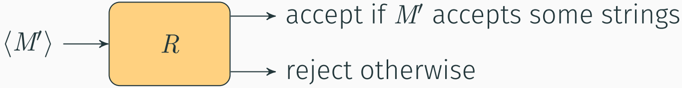
To know whether M accepts some strings seems to require simulating M

But then we need to know whether M halts

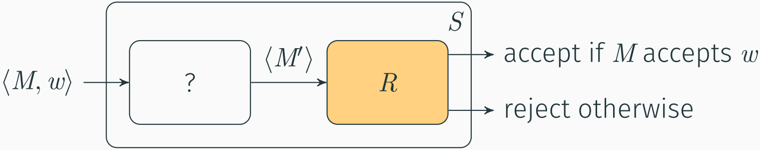
Let's justify this intuition

Eample 2: Figuring out the reduction

Suppose A''_{TM} can be decided by a TM R



We want to build a TM S



M' should be a Turing machine such that

M' accepts some strings if and only if M accepts input w

Implementing the reduction

Task: Given $\langle M, w\rangle$, construct M' so that

If M accepts w, then M' accepts some input

If M does not accept w, then M' accepts no inputs

TM M': On input z

1. Simulate M on input w
2. If M accepts, accept
3. Otherwise, reject

Example 2: The formal proof

$$A''_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM that accepts some input}\}$$
$$A_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$$

Suppose A''_{TM} is decidable by a TM R

Consider the TM S:

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

1. Construct the following TM M':

   M' = a TM such that on input z,

   Simulate M on input w and accept/reject according to M
2. Run R on input $\langle M' \rangle$ and accept/reject according to R

Then S accepts $\langle M, w \rangle$ if and only if M accepts w

So S decides A_{TM}, which is impossible

Example 3

$E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts no input}\}$

Is E_{TM} decidable?

Undecidable! We will show:

If E_{TM} can be decided by some TM R

Then A''_{TM} can be decided by another TM S

$A''_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts some input strings}\}$

Example 3

$$E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts no input}\}$$
$$A''_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts some input}\}$$

Then $E_{\text{TM}} = \overline{A''_{\text{TM}}}$ (except ill-formatted strings, which we will ignore)

Suppose E_{TM} can be decided by some TM R

Consider the following Turing machine S:

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

1. Run R on input $\langle M\rangle$
2. If R accepts, reject
3. If R rejects, accept

Then S decides A''_{TM}, a contradiction

Example 4

$$\mathsf{EQ}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs such that } L(M_1) = L(M_2)\}$$

Is $\mathsf{EQ}_{\mathsf{TM}}$ decidable?

Undecidable!

We will show that $\mathsf{EQ}_{\mathsf{TM}}$ can be decided by some TM R

then E_{TM} can be decided by another TM S

Example 4: Setting up the reduction

$$\mathrm{EQ}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs such that } L(M_1) = L(M_2)\}$$
$$E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts no input}\}$$

Given $\langle M\rangle$, we need to construct $\langle M_1, M_2\rangle$ so that

- If M accepts no input, then M_1 and M_2 accept the same set of inputs
- If M accepts some input, then M_1 and M_2 do not accept the same set of inputs

Idea: Make $M_1 = M$

Make M_2 accept nothing

Example 4: The formal proof

$$\mathsf{EQ}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs such that } L(M_1) = L(M_2)\}$$
$$E_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM that accepts no input}\}$$

Suppose $\mathsf{EQ}_{\mathsf{TM}}$ is decidable and R decides it

Consider the following Turing machine S:

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

1. Construct a TM M_2 that rejects every input z
2. Run R on input $\langle M, M_2\rangle$ and accept/reject according to R

Then S accepts $\langle M\rangle$ if and only if M accepts no input

So S decides E_{TM} which is impossible