Acceptance Problem

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is TM}, w \text{ string}, w \in L(M) \}$

Monday May 16

Theorem: A_{TM} is not Turing-decidable.

There is no decider whose language is A_{TM}.

Proof: Suppose **towards a contradiction** that there is a Turing machine that decides A_{TM}. We call this presumed machine M_{ATM}.

Turing machine, M_{ATM} halts for every input

By assumption, for every Turing machine M and every string w

string TM

$\langle M,w\rangle \in A_{TM}$ so $\langle M,w\rangle \in L(M_{ATM})$

- If $w \in L(M)$, then the computation of M_{ATM} on $\langle M, w\rangle$ halts and accepts.
- If $w \notin L(M)$, then the computation of M_{ATM} on $\langle M, w\rangle$ halts and rejects.

$\langle M,w\rangle \notin A_{TM}$ so $\langle M,w\rangle \notin L(M_{ATM})$

Define a **new** Turing machine using the high-level description:

> D =" On input $\langle M\rangle$, where M is a Turing machine:
> 1. Run M_{ATM} on $\langle M, \langle M\rangle\rangle$.
> 2. If M_{ATM} accepts, reject; if M_{ATM} rejects, accept."

Type check (halts)

subroutine calling M_{ATM} halts because M_{ATM} is decider.

single step conditional.

Is D a Turing machine? Yes, high level description that uses a type check and calls another TM as a subroutine gives a TM.

Is D a decider? Yes: input x is a string

① $x \neq \langle M\rangle$ for any TM M

② $x = \langle M\rangle$ for some TM M where $\langle M\rangle \in L(M)$

③ $x = \langle M\rangle$ for some TM M where $\langle M\rangle \notin L(M)$

What is the result of the computation of D on $\langle D\rangle$? type check? Yes.

Case ① D accepts $\langle D\rangle$
i.e. $\langle D\rangle \in L(D)$
i.e. $\langle D, \langle D\rangle\rangle \in A_{TM}$.
Trace computation of D on $\langle D\rangle$
Step 1: Run M_{ATM} on $\langle D, \langle D\rangle\rangle$ because $\langle D\rangle \in L(D)$ and since $L(M_{ATM}) = A_{TM}$ M_{ATM} accepts $\langle D, \langle D\rangle\rangle$
Step 2: Because M_{ATM} accepts, D rejects!
CONTRADICTION with case assumption that D accepts $\langle D\rangle$.

Case ② D rejects $\langle D\rangle$
i.e. $\langle D\rangle \notin L(D)$
i.e. $\langle D, \langle D\rangle\rangle \notin A_{TM}$.
Trace computation of D on $\langle D\rangle$
Step 1: Run M_{ATM} on $\langle D, \langle D\rangle\rangle$ reject by case assumption that $\langle D\rangle \notin L(D)$.
Step 2: Because M_{ATM} rejects $\langle D, \langle D\rangle\rangle$, D accepts $\langle D\rangle$.
CONTRADICTION with case assumption!

Yes No

Theorem (Sipser Theorem 4.22): A language is Turing-decidable if and only if both it and its complement are Turing-recognizable.

Proof, first direction: Suppose language L is Turing-decidable. WTS that both it and its complement are Turing-recognizable.

Let M be a decider with $L = L(M)$.
Since a decider is a TM, M also witnesses that L is Turing-recognizable. To show $\overline{L}$ is recognizable, consider the TM M_{opp} = "On input w,
1. Run M on w.
2. If accepts, reject; if rejects, accept"

Since M is a decider, so is M_{opp} (details for extra practice)
Moreover, $L(M_{opp}) = \overline{L(M)} = \overline{L}$, as required. (details for extra practice)

Proof, second direction: Suppose language L is Turing-recognizable, and so is its complement. WTS that L is Turing-decidable.

Let M_1 be a TM with $L = L(M_1)$ and let M_2 be a TM with $\overline{L} = L(M_2)$. We define the new TM

M = "On input w
1. Run computations of M_1 and M_2 on w, one step at a time (alternating between the steps of M_1 and M_2).
2. If M_1 accepts, accept; if M_1 rejects, reject; if M_2 accepts, reject; if M_2 accepts, reject"

Claim ① $L(M) = L$ (details for extra practice)

Claim ② M is a decider (details for extra practice: idea is that since each string is either in L or not in L, either M_1 or M_2 must accept in finite time

Give an example of a **decidable** set: A_{DFA}, A_{PDA}, EQ_{REX}, E_{CFG}
$\emptyset$, Σ^*, $L(0^*1(011)^*)$ $\{0^n1^n \mid n \geq 0\}$.

Give an example of a **recognizable undecidable** set:
(Eri)
A_{TM} recognizable

Give an example of an **unrecognizable** set:
$\overline{A_{TM}}$ if recognizable then Thm would say A_{TM} decidable.

True or ~~**False**~~: The class of Turing-decidable languages is closed under complementation?

WTS for every Turing-decidable language, its complement is also Turing-decidable.

Let L be arbitrary Turing-dec language.

By Thm 4.22. L is recognizable ** and $\overline{L}$ is recognizable. *

WTS $\overline{L}$ is decidable.

Since $\overline{L}$ is recognizable (*) and $\overline{\overline{L}} = L$ (by double complementation) is recognizable by (**), Thm 4.22 gives $\overline{L}$ is decidable

Note: additional proof is to build M_{opp} deciding $\overline{L}$ given M deciding L. Left as extra practice

~~True~~/False The class of Turing recognizable languages is closed under complementation.

Counterexample: A_{TM}

Definition: A language L over an alphabet Σ is called **co-recognizable** if its complement, defined as $\Sigma^* \setminus L = \{x \in \Sigma^* \mid x \notin L\}$, is Turing-recognizable.

Notation: The complement of a set X is denoted with a superscript c, X^c, or an overline, $\overline{X}$.

Review: Week 8 Monday

Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

Please complete the review quiz questions on Gradescope about undecidability.

Wednesday May 18

Mapping reduction

Motivation: Proving that A_{TM} is undecidable was hard. How can we leverage that work? Can we relate the decidability / undecidability of one problem to another?

If problem X is **no harder than** problem Y
. . . and if Y is easy,
. . . then X must be easy too.

Y

X

If problem X is **no harder than** problem Y
. . . and if X is hard,
. . . then Y must be hard too.

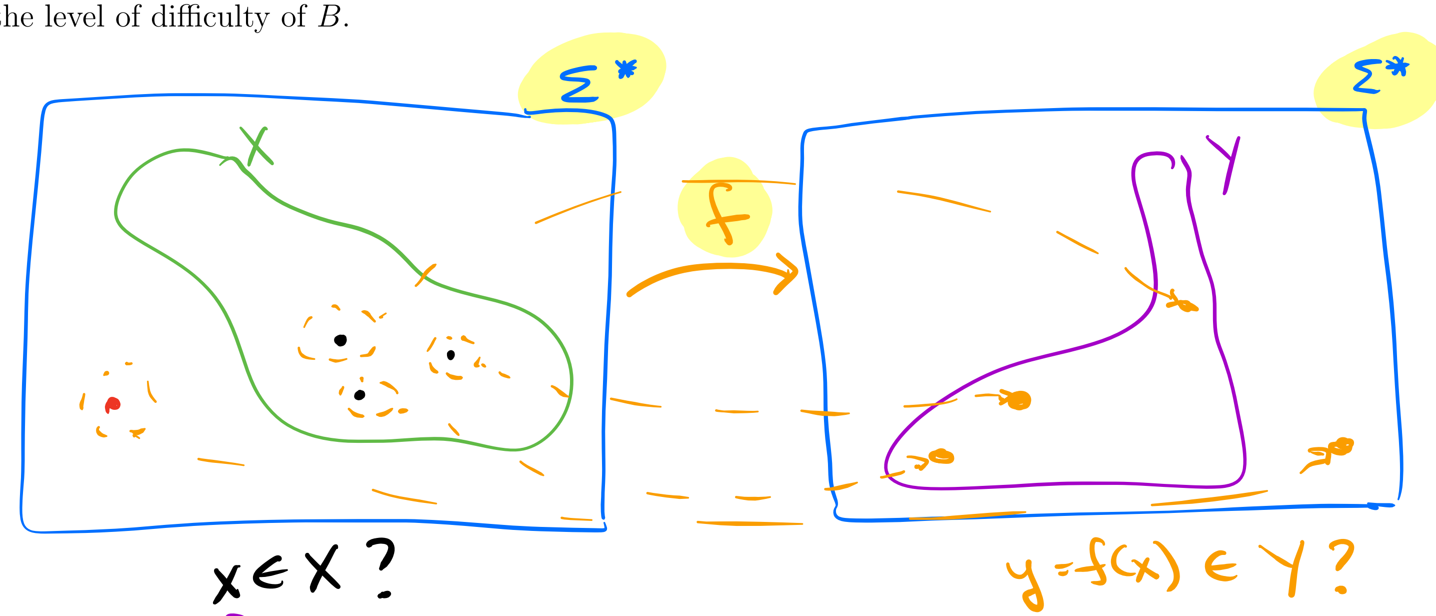
"Problem X is no harder than problem Y" means "Can answer questions about membership in X by converting them to questions about membership in Y".

Definition: A is **mapping reducible to** B means there is a computable function $f : \Sigma^* \to \Sigma^*$ such that *for all* strings x in Σ^*,

$$x \in A \qquad \text{if and only if} \qquad f(x) \in B.$$

Notation: when A is mapping reducible to B, we write $A \leq_m B$.

Intuition: $A \leq_m B$ means A is no harder than B, i.e. that the level of difficulty of A is less than or equal the level of difficulty of B.



computable function . . . .
language is decidable
language is undecidable
language is recognizable
language is unrecognizable
language is context-free

Computable functions

Definition: A function $f : \Sigma^* \to \Sigma^*$ is a **computable function** means there is some Turing machine such that, for each x, on input x the Turing machine halts with exactly $f(x)$ followed by all blanks on the tape

Examples of computable functions:

$\langle n \rangle$ = bitstring

The function that maps a string to a string which is one character longer and whose value, when interpreted as a fixed-width binary representation of a nonnegative integer is twice the value of the input string (when interpreted as a fixed-width binary representation of a non-negative integer)

$$f_1 : \Sigma^* \to \Sigma^* \qquad f_1(x) = x0$$

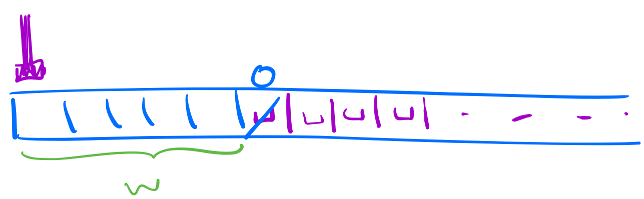
$f_1(\varepsilon) = 0$ $\qquad$ $f_1(00) = 000$ $\qquad$ $f_1(10) = 100$

To prove f_1 is computable function, we define a Turing machine computing it.

High-level description

"On input w

1. Append 0 to w.
2. Halt."

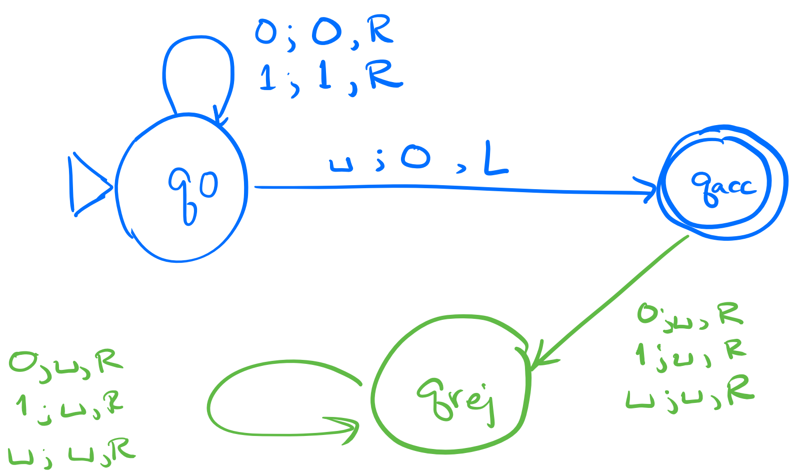


Implementation-level description

"On input w

1. Sweep read-write head to the right until find first blank cell.
2. Write 0.
3. Halt."

Formal definition $(\{q0, qacc, qrej\}, \{0, 1\}, \{0, 1, \text{␣}\}, \delta, q0, qacc, qrej)$ where δ is specified by the state diagram:



Conventions:
omit drawing qreject
any transitions missing from drawing assumed to have output (qrej, ␣, R)

The function that maps a string to the result of repeating the string twice.

$$f_2 : \Sigma^* \to \Sigma^* \qquad f_2(x) = xx$$

extra practice.

The function that maps strings that are not the codes of Turing machines to the empty string and that maps strings that code Turing machines to the code of the related Turing machine that acts like the Turing machine coded by the input, except that if this Turing machine coded by the input tries to reject, the new machine will go into a loop.

$$f_3 : \Sigma^* \to \Sigma^* \qquad f_3(x) = \begin{cases} \varepsilon & \text{if } x \text{ is not the code of a TM} \\ \langle (Q \cup \{q_{trap}\}, \Sigma, \Gamma, \delta', q_0, q_{acc}, q_{rej}) \rangle & \text{if } x = \langle (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}) \rangle \end{cases}$$

related TM

where $q_{trap} \notin Q$ and

$$\delta'((q, x)) = \begin{cases} (r, y, d) & \text{if } q \in Q,\ x \in \Gamma,\ \delta((q, x)) = (r, y, d), \text{ and } r \neq q_{rej} \\ (q_{trap}, \sqcup, R) & \text{otherwise} \end{cases}$$

Define TM computing f_3 as

"On input x

1. If $x \neq \langle M \rangle$ for any TM M, output ε (and halt)

using subroutine for type checking string encodings.

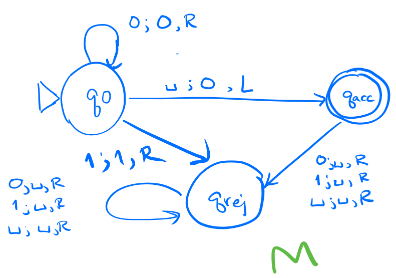
2. Else: $x = \langle M \rangle$ for $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

build $M' = (Q \cup \{q_{trap}\}, \Sigma, \Gamma, \delta', q_0, q_{acc}, q_{rej})$

with δ' as above

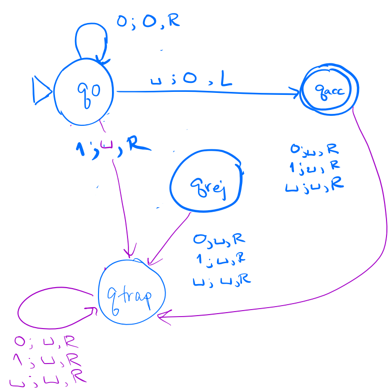
3. Output $\langle M' \rangle$ "

using subroutine for encoding decoding TMs as strings

$x = \langle$  $\rangle$

$f_3(x) = \langle M' \rangle$

where M' is:



Some observations relating M and M':

$0 \in L(M)$

$01 \notin L(M)$

$\varepsilon \in L(M)$.

M halts & accepts strings with no 1s.

M halts & rejects strings with any 1s.

$L(M') = L(M)$

M' halts & accept strings with no 1s.

M' loops on strings that have any 1s.

The function that maps strings that are not the codes of CFGs to the empty string and that maps strings that code CFGs to the code of a PDA that recognizes the language generated by the CFG.

extra practice

Other examples?

Review: Week 8 Wednesday

Please complete the review quiz questions on Gradescope about mapping reductions.

Pre class reading for next time: Theorem 5.21 (page 236)

Friday May 20

Recall definition: A is **mapping reducible to** B means there is a computable function $f : \Sigma^* \to \Sigma^*$ such that *for all* strings x in Σ^*,

$$x \in A \qquad \text{if and only if} \qquad f(x) \in B.$$

Notation: when A is mapping reducible to B, we write $A \leq_m B$.

Intuition: $A \leq_m B$ means A is no harder than B, i.e. that the level of difficulty of A is less than or equal the level of difficulty of B.

Example: $A_{TM} \leq_m A_{TM}$

Example: $A_{DFA} \leq_m \{ww \mid w \in \{0,1\}^*\}$

Example: $\{0^i 1^j \mid i \geq 0, j \geq 0\} \leq_m A_{TM}$

Theorem (Sipser 5.22): If $A \leq_m B$ and B is decidable, then A is decidable.

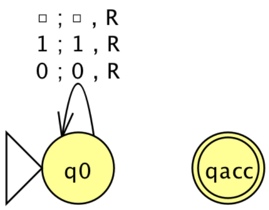
Theorem (Sipser 5.23): If $A \leq_m B$ and A is undecidable, then B is undecidable.

Halting problem

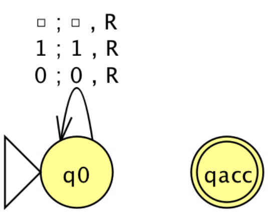
$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string, and } M \text{ halts on } w\}$$

Define $F : \Sigma^* \to \Sigma^*$ by

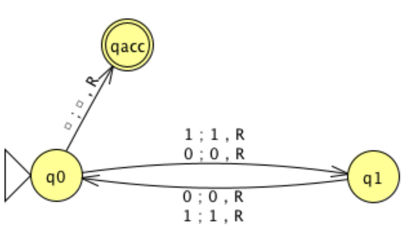
$$F(x) = \begin{cases} const_{out} & \text{if } x \neq \langle M, w\rangle \text{ for any Turing machine } M \text{ and string } w \text{ over the alphabet of } M \\ \langle M', w\rangle & \text{if } x = \langle M, w\rangle \text{ for some Turing machine } M \text{ and string } w \text{ over the alphabet of } M. \end{cases}$$

where $const_{out} = \langle$ 

$, \varepsilon\rangle$ and M' is a Turing machine that computes like M except, if the computation ever were to go to a reject state, M' loops instead.

$F(\langle$ 

$, 001\rangle) =$

$F(\langle$ 

$, 1\rangle) =$

To use this function to prove that $A_{TM} \leq_m HALT_{TM}$, we need two claims:

Claim (1): F is computable

Claim (2): for every x, $x \in A_{TM}$ iff $F(x) \in HALT_{TM}$.

Review: Week 8 Friday

Please complete the review quiz questions on Gradescope about the relationship between A_{TM} and $HALT_{TM}$

Pre class reading for next time: Example 5.30.