Monday

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

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

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

- If $w \in L(M)$, then the computation of M_{ATM} on $\langle M, w \rangle$ _____
- If $w \notin L(M)$, then the computation of M_{ATM} on $\langle M, w \rangle$ _____

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."

Is D a Turing machine?

Is D a decider?

What is the result of the computation of D on $\langle D \rangle$?

CC BY-NC-SA 2.0 Version January 22, 2024 (1)

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.

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

Give an example of a **decidable** set:

Give an example of a **recognizable undecidable** set:

Give an example of an **unrecognizable** set:

CC BY-NC-SA 2.0 Version January 22, 2024 (2)

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

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, $\overline{X}$.

CC BY-NC-SA 2.0 Version January 22, 2024 (3)

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.

CC BY-NC-SA 2.0 Version January 22, 2024 (4)

Wednesday

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.

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.

CC BY-NC-SA 2.0 Version January 22, 2024 (5)

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:

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)

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

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

CC BY-NC-SA 2.0 Version January 22, 2024 (6)

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

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

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.

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

where $q_{trap} \notin Q$ and

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

CC BY-NC-SA 2.0 Version January 22, 2024 (7)

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.

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)

CC BY-NC-SA 2.0 Version January 22, 2024 (8)

Friday

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.

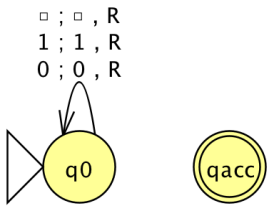
CC BY-NC-SA 2.0 Version January 22, 2024 (9)

Halting problem

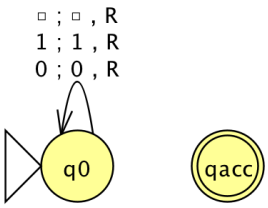
$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string, and } M \text{ halts on } w\}$$

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

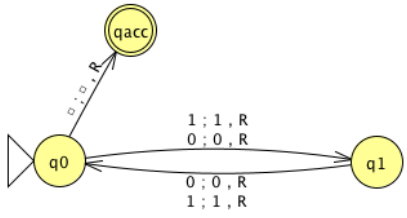
$$F(x) = \begin{cases} const_{out} & \text{if } x \neq \langle M, w\rangle \text{ for any Turing machine } M \text{ and string } w \text{ over the alphabet of } M \\ \langle M', w\rangle & \text{if } x = \langle M, w\rangle \text{ for some Turing machine } M \text{ and string } w \text{ over the alphabet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

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.

$F(\langle$ 

$, 001\rangle) =$

$F(\langle$ 

$, 1\rangle) =$

CC BY-NC-SA 2.0 Version January 22, 2024 (10)

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 $F(x) \in HALT_{TM}$.

CC BY-NC-SA 2.0 Version January 22, 2024 (11)

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.

CC BY-NC-SA 2.0 Version January 22, 2024 (12)