Homework 6

CSE 105, Fall 2025
Due: Monday November 24, 11:59pm

Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning, where applicable. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

Instructions:

- Read each question carefully.
- Selection of questions
  - You are required to answer <u>one question from each topic</u> in this homework.
  - You may choose any one question from the set of questions within each topic.
- Each question is worth 5 points (20 points total).
- Submission guidelines:
  - Upload a single file to Gradescope for each group.
  - All group members' names and PIDs should be on each page of the submission.
  - Include the text of each question you are answering, followed by your solution.

Topics:

1. Language Classes and Intuition
2. Recognizability vs. Decidability
3. Decidable Problems for Automata and Grammars
4. Turing Machine Variants

Problem 1:

Give the formal 7-tuple definition of a Turing machine. Clearly describe what each component of the tuple represents (e.g., the sets and the transition function).

Problem 2:

Provide a high-level description of a Turing machine that decides the language L = $\{0^n 1^n$ | $n \geq 0\}$. (Note: This language is context-free, but you must use a TM). Describe the steps the machine would take, including how it uses its tape to check the string.

Problem 3:

Assume you have a TM for the language L = $\{0^n 1^n$ | $n \geq 0\}$. Trace the machine's operation on the input string 0011. List the first 6 configurations of the machine, starting from the initial configuration q0 0011. You may use the shorthand *u q v* to represent a configuration.

Problem 4:

A Turing machine's computation is defined by its sequence of configurations. Describe the three special types of configurations that start and stop a computation. (a) What is the start configuration for a TM on a given input string w? (b) What defines an accepting configuration? (c) What defines a rejecting configuration?

Problem 5:

Explain the difference between a Turing machine that is a decider and one that is only a recognizer. What specific condition must a decider meet for all possible inputs?

Problem 6:

Prove that a language L is decidable **if and only if** both L and its complement L^c are Turing-recognizable.

Problem 7:

We know that:

A_{TM} = { ⟨M,w⟩ | M is a TM and M accepts w} is Turing-recognizable but **not** decidable (we will prove this in class).

Using your result from Question 6, what can you conclude about the language $A_{TM}{}^c$? Is it Turing-recognizable? Justify your answer.

Problem 8:

A language is **co-Turing-recognizable** if its complement is Turing-recognizable.
Let:

- RE = the class of Turing-recognizable languages
- co-RE = the class of co-Turing-recognizable languages

Answer the following:

A. What is the set RE ∩ co-RE? (Hint: Use theorem from Question 6)
B. Draw a Venn diagram illustrating the relationship between:
   a. the set of all languages,
   b. RE,
   c. co-RE,
   d. the class of decidable languages.
C. Place A_{TM} and $A_{TM}{}^c$ in their correct regions of this diagram.

Problem 9:

Prove that the class of Turing-recognizable languages is closed under intersection. That is, given two recognizable languages *L1* and *L2*, describe how to construct a TM M_{int} that recognizes *L1* ∩ *L2*.

Problem 10:

Prove that the class of Turing-recognizable languages is closed under union. Given Turing Machines *M1* and *M2*, describe a *TM* M_{union} that recognizes *L(M1)* ∪ *L(M2)*

In this section: Give a formal proof for the question that includes Construction, and Proof of correctness (both directions).

Problem 11:

Prove that:

A_{NFA} = { ⟨N,w⟩ | N is an NFA and N accepts w } is a decidable language.

Problem 12:

Prove that

A_{CFG} = { ⟨G,w⟩ | G is a CFG and G generates w } is a decidable language.

Problem 13:

Prove that:

MIN_{DFA} = { ⟨D⟩ | D is a minimal DFA} is a decidable language.

For the purpose of this problem, a DFA:
D = (Q, Σ, δ, q0, F) is considered **minimal** if it satisfies both of the following:

1. **No Unreachable States:** Every state q ∈ Q is reachable from the start state.
2. **No Equivalent States:** No two distinct states p, q ∈ Q (with p ≠ q) are indistinguishable.

Notes:

1. **Reachable State:**
   A state *q* is reachable if there exists a string w ∈ Σ* such that **δ*(q0, w) = q**
   Any state not satisfying this is unreachable.
2. **Equivalent States:**
   Two states *p* and *q* are **equivalent** (written p≈q) if, for all strings w ∈ Σ*:
   δ*(p, w) ∈ **F** ⇔ **δ*(q, w)** ∈ **F**
   In other words, no string can distinguish between starting at *p* versus starting at *q*; they behave identically with respect to acceptance.

Problem 14:

Prove that:

$INFINITE_{DFA}$ = { ⟨D⟩ | D is a DFA and L(D) is an infinite language} is a decidable language.
Hint: How can you use the graph of the DFA's states to check for this property?

Problem 15:

Prove that:

ALL_{DFA} = { ⟨D⟩ | D is a DFA and L(D) = Σ* } is a decidable language.
Hint: Consider the complement language $L(D)^c$. How can you construct a *DFA D'* that recognizes this complement? And how does this relate to the decidability of E_{DFA}?

Problem 16:

Provide an implementation-level description of how a **2-stack machine** (with a read-only input tape) can simulate a standard single-tape TM *M*. Your description must explain:

- How the two stacks (S_{left} and S_{right}) represent the TM's tape and head position.
- The procedure for simulating *M's* "Move Right" transition.
- The procedure for simulating *M's* "Move Left" transition.

Problem 17:

Provide an implementation-level description of how a **1-queue machine** (with a read-only input tape) can simulate a standard single-tape TM *M*. (Hint: Store the entire tape contents in the queue, with a special marker for the head position. How do you simulate a "Move Left"?)

Problem 18:

Provide an implementation-level description for how a **standard (*semi-infinite* - infinite to the right) TM** *M'* can simulate a **doubly-infinite tape TM** *M*. Your description must detail:

- How *M'* 's single tape is formatted (e.g., using a special "center" marker) to represent M's entire doubly-infinite tape.
- The procedure for M' to simulate a "Move Right" by M.
- The procedure for M' to simulate a "Move Left" by M, especially when M's head is at the "center" marker.

Problem 19:

Suppose you have a Turing Machine *M* with tape alphabet Γ = {⊔, 0, 1, a}.
You need to construct an equivalent Turing Machine M′ with tape alphabet Γ′ = {⊔, 0, 1}.

(a) Define a binary encoding for each symbol in Γ.

(b) Describe in detail how M′ would simulate a single transition of M, such as:

$$\delta(q1,a) = (q2,1,L)$$

Explain precisely how M′ reads the encoded symbol, writes the encoded output symbol, and performs the left move, including any intermediate states needed.

Problem 20:

Prove that if a language *L* is enumerated by an enumerator *E*, then *L* is **Turing-recognizable**. (That is, show how to use *E* as a component to build a TM *M* that recognizes *L*).