HW4CSE105W25: Homework assignment 4

CSE105W25

Due: February 20th at 5pm, via Gradescope

In this assignment,

You will work with context-free languages and their representations. You will also practice analyzing, designing, and working with Turing machines. You will explore recognizable and decidable languages.

Resources: To review the topics for this assignment, see the class material from Weeks 5 and 6. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Chapters 2 and 3. Chapter 2 exercises 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.9, 2.11, 2.12, 2.13, 2.16, 2.17. Chapter 3 exercises 3.1, 3.2, 3.5, 3.8.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. On the "graded for correctness" questions, you may only collaborate with CSE 105 students in your group; if your group has questions about a problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza. On the "graded for completeness" questions, you may collaborate with all other CSE 105 students this quarter, and you may make public posts about these questions on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of

machines, you can (1) use the LaTex tikzpicture environment (see templates in the class notes), or (2) use the software tools Flap.js or JFLAP described in the class syllabus (and include a screenshot in your PDF), or (3) you can carefully and clearly hand-draw the diagram and take a picture and include it in your PDF. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while getting to know and learn from your classmates.

- On the "graded for correctness" questions, you may only collaborate with CSE 105 students in your group. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.

- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw4CSE105W25".

Assigned questions

1. **Regular and nonregular languages and context-free grammars (CFG)** (6 points): On page 7 of the week 4 notes, we have the following list of languages over the alphabet $\{a, b\}$

$$\{a^n b^n \mid 0 \leq n \leq 5\} \quad \{b^n a^n \mid n \geq 2\} \quad \{a^m b^n \mid 0 \leq m \leq n\}$$
$$\{a^m b^n \mid m \geq n + 3, n \geq 0\} \quad \{b^m a^n \mid m \geq 1, n \geq 3\}$$
$$\{w \in \{a, b\}^* \mid w = w^{\mathcal{R}}\} \quad \{ww^{\mathcal{R}} \mid w \in \{a, b\}^*\}$$

(a) (*Graded for completeness*) [1] Pick one of the regular languages and design a context-free grammar that generates it. Briefly justify your grammar by describing the role of each of the rules and connecting it to the intended language and referencing relevant definitions.

---

[1]This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we expect you to include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

(b) (*Graded for completeness*) Pick one of the nonregular languages and design a context-free grammar that generates it. Briefly justify your grammar by describing the role of each of the rules and connecting it to the intended language and referencing relevant definitions.

2. **General constructions for context-free languages** (15 points):

In class in week 5, we described several general constructions with PDAs and CFGs, leaving their details to homework. In this question, we'll fill in these details. The first constructions help us prove that the class of regular languages is a subset of the class of context-free languages. The other construction allows us to make simplifying assumptions about PDAs recognizing languages.

(a) (*Graded for completeness*) When we first introduced PDAs we observed that any NFA can be transformed to a PDA by not using the stack of the PDA at all. Suppose a friend gives you the following construction to formalize this transformation:

> Given a NFA $N = (Q, \Sigma, \delta_N, q_0, F)$ we define a PDA M with $L(M) = L(N)$ by letting $M = (Q, \Sigma, \Sigma, \delta, q_0, F)$ where $\delta(\ (q, a, b)\ ) = \delta_N(\ (q, a)\ )$ for each $q \in Q$, $a \in \Sigma_\varepsilon$ and $b \in \Sigma_\varepsilon$.

For each of the six defining parameters for the PDA, explain whether it's defined correctly or not. If it is not defined correctly, explain why not and give a new definition for this parameter that corrects the mistake.

(b) (*Graded for correctness*) [2] In the book on page 107, the top paragraph describes a procedure for converting DFAs to CFGs:

> You can convert any DFA into an equivalent CFG as follows. Make a variable R_i for each state q_i of the DFA. Add the rule $R_i \to aR_j$ to the CFG if $\delta(q_i, a) = q_j$ is a transition in the DFA. Add the rule $R_i \to \varepsilon$ if q_i is an accept state of the DFA. Make R_0 the start variable ofthe grammar, where q_0 is the start state of the machine. Verify on your own that the resulting CFG generates the same language that the DFA recognizes.

Use this construction to get a context-free grammar generating the language

$$\{w \in \{a, b\}^* \mid w \text{ has at least one } a \text{ and does not end in } bb\}$$

by (1) designing a DFA that recognizes this language and then (2) applying the construction from the book to convert the DFA to an equivalent CFG. A complete and correct submission will include the state diagram of the DFA, a brief justification of why it recognizes the language, and then the complete and precise definition of the CFG that results from applying the construction from the book to this DFA. *Ungraded bonus: take a sample string in the language and see how the computation of the DFA on this string translates to a derivation in your grammar.*

---

[2]This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. 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.

(c) Let $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, F_1)$ be a PDA and let $q_{new}, r_{new}, s_{new}$ be three fresh state labels (i.e. $Q_1 \cap \{q_{new}, r_{new}, s_{new}\} = \emptyset$) and let $\#$ be a fresh stack symbol (i.e. $\# \notin \Gamma_1$). We define the PDA M_2 as

$$(Q_2, \Sigma, \Gamma_2, \delta_2, q_{new}, \{s_{new}\})$$

with $Q_2 = Q_1 \cup \{q_{new}, r_{new}, s_{new}\}$ and $\Gamma_2 = \Gamma_1 \cup \{\#\}$ and $\delta_2 : Q_2 \times \Sigma_\varepsilon \times \Gamma_{2\varepsilon} \to \mathcal{P}(Q_2 \times \Gamma_{2\varepsilon})$ given by

$$\delta_2(\ (q, a, b)\ ) = \begin{cases} \{(q_1, \#)\} & \text{if } q = q_{new}, a = \varepsilon, b = \varepsilon \\ \delta_1(\ (q, a, b)\ ) & \text{if } q \in Q_1 \setminus F_1, a \in \Sigma_\varepsilon, b \in \Gamma_{1\varepsilon} \\ \delta_1(\ (q, a, b)\ ) & \text{if } q \in F_1, a \in \Sigma, b \in \Gamma_{1\varepsilon} \\ \delta_1(\ (q, a, b)\ ) & \text{if } q \in F_1, a = \varepsilon, b \in \Gamma_1 \\ \delta_1(\ (q, a, b)\ ) \cup \{(r_{new}, \varepsilon)\} & \text{if } q \in F_1, a = \varepsilon, b = \varepsilon \\ \{(r_{new}, \varepsilon)\} & \text{if } q = r_{new}, a = \varepsilon, b \in \Gamma_1 \\ \{(s_{new}, \varepsilon)\} & \text{if } q = r_{new}, a = \varepsilon, b = \# \\ \emptyset & \text{otherwise} \end{cases}$$

for each $q \in Q_2$, $a \in \Sigma_\varepsilon$, and $b \in \Gamma_{2\varepsilon}$.

In this question, we'll apply this construction for a specific PDA and use this example to extrapolate the effect of this construction.

i. (*Graded for correctness*) Consider the PDA M_1 with input alphabet $\{0, 1\}$ and stack alphabet $\{0, 1, \$\}$ whose state diagram is
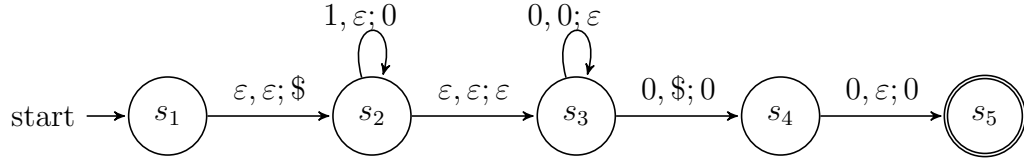
Draw the state diagram for the PDA M_2 that results from applying the construction to M_1. Also, give an example string of length 4 that is accepted by both M_1 and M_2 and justify your choice by describing an accepting computation for each of the PDAs on your input string.

ii. (*Graded for completeness*) Compare $L(M_1)$ and $L(M_2)$. Are these sets equal? Does your answer depend on the specific choice of M_1? Why or why not?

iii. (*Graded for completeness*) Consider the PDA N with input alphabet $\{0, 1\}$ and stack alphabet $\{0, 1\}$ whose state diagram is
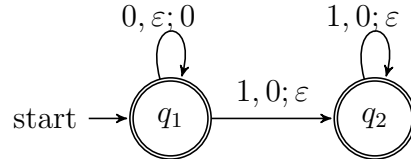
Remember that the definition of set-wise concatenation is: for languages L_1, L_2 over the alphabet Σ, we have the associated set of strings

$$L_1 \circ L_2 = \{w \in \Sigma^* \mid w = uv \text{ for some strings } u \in L_1 \text{ and } v \in L_2\}$$

In class, we discussed how extrapolating the construction that we used to prove that the class of regular languages is closed under set-wise concatenation by drawing spontaneous transitions from the accepting states in the first machine to the start state of the second machine doesn't work. Use the example of M_1 and N to prove this by showing that

$$L(M_1) \circ L(N)$$

is **not** the language recognized by the machine results from taking the two machines M_1 and N, setting the start state of M_1 to be the start state of the new machine, setting the set of accepting states of N to be the set of accepting states of the new machine, and drawing spontaneous arrows from the accepting states of M_1 to the start state of N. Then, describe the language recognized by the machine that results from taking the two machines M_2 and N, setting the start state of M_2 to be the start state of the new machine, setting the set of accepting states of N to be the set of accepting states of the new machine, and drawing spontaneous arrows from the accepting states of M_2 to the start state of N. Use this description to explain why we used the construction of M_2 from M_1 and how this construction could be used in a proof of the closure of the class of context-free languages under set-wise concatenation.

A complete response will give an example string that witnesses that $L(M_1) \circ L(N)$ is not equal to the language recognized by the PDA resulting from the wrong construction (described above) **and** the state diagram of the PDA that results from applying that construction to M_2 and N (instead of M_1), with a brief justification about why that approach works.

3. **Turing machines** (9 points):

Consider the Turing machine T over the input alphabet $\Sigma = \{0, 1\}$ with the state diagram below (the tape alphabet is $\Gamma = \{0, 1, X, \square\}$). Convention: we do not include the node for the reject state $qrej$ and any missing transitions in the state diagram have value $(qrej, \square, R)$

(a) (*Graded for correctness*) Specify an example string w_1 of length 4 over Σ that is **accepted** by this Turing machine, or explain why there is no such example. A complete solution will

include either (1) a precise and clear description of your example string and a precise and clear description of the accepting computation of the Turing machine on this string or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

To describe a computation of a Turing machine, include the contents of the tape, the state of the machine, and the location of the read/write head at each step in the computation.

Hint: In class we've drawn pictures to represent the configuration of the machine at each step in a computation. You may do so or you may choose to describe these configurations in words.

(b) (*Graded for correctness*) Specify an example string w_2 of length 3 over Σ that is **rejected** by this Turing machine or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example string and a precise and clear description of the rejecting computation of the Turing machine on this string or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

(c) (*Graded for correctness*) Specify an example string w_3 of length 3 over Σ on which the computation of this Turing machine is **never halts** or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example string and a precise and clear description of the looping (non-halting) computation of the Turing machine on this string or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

4. **Implementation-level descriptions of deciders and recognizers** (12 points): Consider the language
$$\{a^i b^j \mid i \geq 0, j > 1\}$$
over the alphabet $\{a, b\}$.

(a) (*Graded for correctness*) Give an example of a Turing machine that **decides** this language. A complete solution will include **both** a state diagram and an implementation-level description of this Turing machine, along with a brief explanation of why it recognizes this language, and why it is a decider.

(b) (*Graded for correctness*) Give an example of a Turing machine that **recognizes but does not decide** this language. A complete solution will include **both** a state diagram and an implementation-level description of this Turing machine, along with a brief explanation of why it recognizes this language, and why it is not a decider.

5. **Classifying languages** (8 points): Our first example of a more complicated Turing machine was of a Turing machine that recognized the language $\{w\#w \mid w \in \{0,1\}^*\}$ (Figure 3.10 in the textbook), which we know is not context-free. Let's call that Turing machine M_0. The language

$$L = \{ww \mid w \in \{0,1\}^*\}$$

is also not context-free.

(a) (*Graded for correctness*) Choose an example string of length 2 in L that is in **not** in $\{w\#w \mid w \in \{0,1\}^*\}$ and describe the computation of the Turing machine M_0 on your example string. Include the contents of the tape, the state of the machine, and the location of the read/write head at each step in the computation.

(b) (*Graded for completeness*) Explain why the Turing machine from the textbook and class that recognized $\{w\#w \mid w \in \{0,1\}^*\}$ does not recognize $\{ww \mid w \in \{0,1\}^*\}$. Use your example to explain why M_0 doesn't recognize L.

(c) (*Graded for completeness*) Explain how you would change M_0 to get a new Turing machine that does recognize L. Describe this new Turing machine using both an implementation-level definition and a state diagram of the Turing machine. You may use all our usual conventions for state diagrams of Turing machines (we do not include the node for the reject state $qrej$ and any missing transitions in the state diagram have value $(qrej, \square, R)$; $b \to R$ label means $b \to b, R$).