HW3CSE105W24: Homework assignment 3

CSE105W24

Due: February 15th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment,

You will practice with the definition of pushdown automata and context-free grammars and reason about regular and context-free languages. You will also practice analyzing and designing Turing machines.

Resources: To review the topics for this assignment, see the class material from Weeks 4-6. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Chapter 2, Section 3.1 Chapter 2 exercises 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.9, 2.10, 2.11, 2.12, 2.13, 2.16, 2.17. Chapter 3 exercises 3.1, 3.2.

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. For "graded for correctness" questions: collaboration is allowed only 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. For "graded for completeness" questions: collaboration is allowed with any CSE 105 students this quarter; if your group has questions about a problem, you may ask in drop-in help hours or post a public post 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, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. 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.
- You may not collaborate on homework questions graded for correctness with anyone other than your group members. 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.

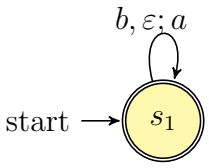
You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw3CSE105W24".

Assigned questions

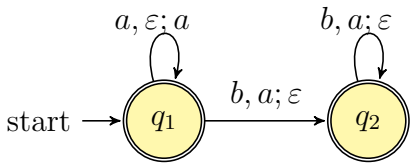
1. **Constructions** (18 points):

Consider the push-down automata M_1 and M_2 over $\{a, b\}$ with stack alphabet $\{a, b\}$ whose state diagrams are

State diagram for M_1



State diagram for M_2



(a) (*Graded for completeness*) [1] What is the language A_1 recognized by M_1 and what is the language A_2 recognized by M_2? Include a sample string that is accepted and one that is

[1] This means you will get full credit so long as your submission demonstrates honest effort to answer the

rejected for each of these PDA. Justify these examples with sample accepting computations or with an explanation why there is no accepting computation.

(b) (*Graded for correctness*) [2] Design CFGs G_1 and G_2 over $\{a, b\}$ so that $L(G_1) = A_1$ and $L(G_2) = A_2$. A complete solution will include precise definitions for each of the parameters required to specify a CFG, as well as a brief explanation about why each string in A_i can be derived in G_i and each string not in A_i cannot be derived in G_i (for $i = 1, 2$).

(c) (*Graded for completeness*) 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 (and in the review quiz) we learned that the class of context-free languages is closed under set-wise concatenation. The proof of this closure claim using CFGs uses the construction: given $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ with $V_1 \cap V_2 = \emptyset$ and $S_{new} \notin V_1 \cup V_2$, define a new CFG

$$G = (V_1 \cup V_2 \cup \{S_{new}\}, \Sigma, R_1 \cup R_2 \cup \{S_{new} \to S_1 S_2\}, S_{new})$$

Apply this construction to your grammars from part (b) and give a sample derivation of a string in $A_1 \circ A_2$ in this resulting grammar.

(d) (*Graded for correctness*) If we try to extrapolate the construction that we used to prove that the class of regular languages is closed under set-wise concategation, we would get the following construction for PDAs: Given $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, F_2)$ with $Q_1 \cap Q_2 = \emptyset$, define $Q = Q_1 \cup Q_2$, $\Gamma = \Gamma_1 \cup \Gamma_2$, and

$$M = (Q, \Sigma, \Gamma, \delta, q_1, F_2)$$

with $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ given by

$$\delta(q, a, b) = \begin{cases} \delta_1(q, a, b) & \text{if } q \in Q_1,\ a \in \Sigma_\varepsilon,\ b \in \Gamma_{1_\varepsilon} \\ \delta_2(q, a, b) & \text{if } q \in Q_2,\ a \in \Sigma_\varepsilon,\ b \in \Gamma_{2_\varepsilon} \\ \delta_1(q, a, b) \cup \{(q_2, \varepsilon)\} & \text{if } q \in F_1,\ a = \varepsilon,\ b = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$

Apply this construction to the machines M_1 and M_2 from part (a), and then use the resulting PDA to prove that *this* construction cannot be used to prove that the class of context-free

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.

[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.

languages is closed under set-wise concatenation. A complete solution will include (1) the state diagram of the machine M that results from applying this construction to M_1 and M_2, (2) an example of a string that is accepted by this PDA M but that is **not** in the language $A_1 \circ A_2$ with a description of the computation that witnesses that this string is accepted by M and an explanation of why this string is not in $A_1 \circ A_2$ by referring back to the definitions of A_1, A_2, and set-wise concatenation.

2. **Regular languages are context-free** (10 points):

Informally, we think of regular languages as potentially simpler than context-free languages. In this question, you'll make this precise by showing that every regular language is context-free, in two ways.

(a) (*Graded for correctness*) When we first introduced PDAs we saw that any NFA can be transformed to a PDA by not using the stack of the PDA at all. Make this precise by completing the following construction: Given a NFA $N = (Q, \Sigma, \delta_N, q_0, F)$ we define a PDA M with $L(M) = L(N)$ by choosing ... A complete solution will have precise, correct definitions for each of the defining paramaters of M: the set of states, the input alphabet, the stack alphabet, the transition function, the start state, and the set of accept states. Be careful to use notation that matches the types of the objects involved.

(b) (*Graded for correctness*) In the book on page 107, the top paragraph describes a procedure for converting DFA 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

$$\{w \in \{0,1\}^* \mid w \text{ does not have 11 as a substring}\}$$

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 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.*

3. **Classifying languages** (12 points): On page 4 of the week 4 notes, we have the following list of languages over the alphabet $\{a, b\}$

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

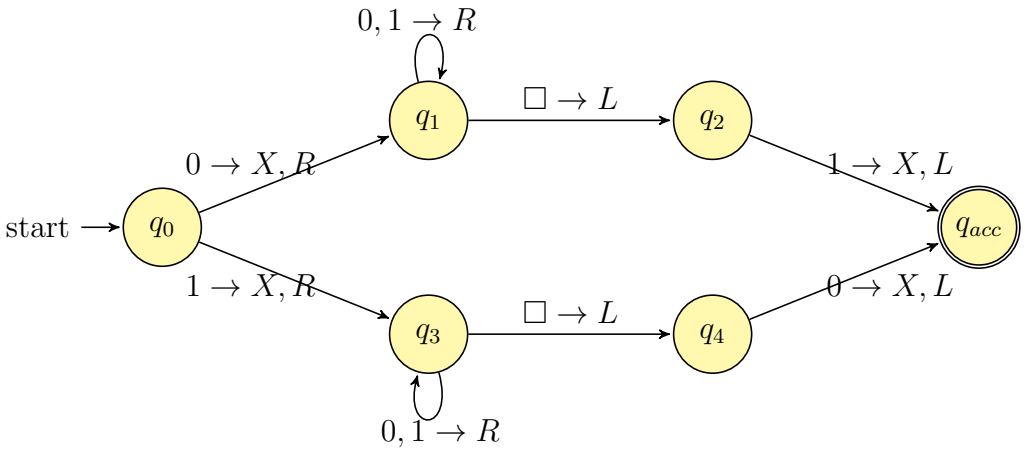
(a) (*Graded for completeness*) Pick one of the regular languages and design a regular expression that describes it. Briefly justify your regular expression by connecting the subexpressions of it to the intended language and referencing relevant definitions.

(b) (*Graded for completeness*) Pick another one of the regular languages and design a DFA that recognizes it. Draw the state diagram of your DFA. Briefly justify your design by explaining the role each state plays in the machine, as well as a brief justification about how the strings accepted and rejected by the machine connect to the specified language.

(c) (*Graded for completeness*) Pick one of the nonregular languages and design a PDA that recognizes it. Draw the state diagram of your PDA. Briefly justify your design by explaining the role each state plays in the machine, as well as a brief justification about how the strings accepted and rejected by the machine connect to the specified language.

(d) (*Graded for completeness*) Pick one of the nonregular languages and write a CFG that generates it. Briefly justify your design by by demonstrating how derivations in the grammar relate to the intended language.

4. **Turing machines** (10 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 2 over Σ on which the computation of this Turing machine **loops** 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.

(d) (*Graded for completeness*) Write an implementation level description of the Turing machine T.