Midterm Exam Study Guide

CSE 30151 Spring 2025

Exam date: 2023/03/04 at 2pm

The cover page of the exam will look like this.

Name:

NetID:

- This exam has six questions, worth 20 points each, for a total of 120 points (16% of your grade).
- You have the whole class period of 75 minutes to write your solutions.
- You may use your textbook and paper notes, but computers, smartphones, and tablets are **not** allowed.
- You may use the textbook, lectures, and lecture notes for this course without citation. However, you may **not** copy or quote from any other materials in your notes that you are not the author of.
- On this page, please write your name and NetID, but please don't write any solutions. On the remaining pages, front and back, please write your solutions, but please don't write your name.

Problems

The exam covers everything about regular languages, non-regular languages, and context-free languages. It does *not* cover non-context-free languages. The six questions will be of the following types. (Exercise/problem numbers are from Sipser; an $^{\mathsf{I}}$ means 3rd international edition and a $^{\mathsf{U}}$ means 3rd US edition.)

- Multiple choice: One question with several parts.
  - For each language below, write whether the language is (F) finite, (R) regular but not finite, (C) context-free but not regular.
    (a) $\{\mathtt{a}^i\mathtt{b}^j \mid i \geq j \geq 100\}$
    (b) $\{\mathtt{a}^i\mathtt{b}^j \mid i \leq j \leq 100\}$
    (c) $\{\mathtt{a}^i\mathtt{b}^j \mid i \geq 100, j \geq 100\}$
    (d) $\{\mathtt{a}^i\mathtt{b}^j \mid i \leq 100, j \leq 100\}$
    (e) $\{ww^{\mathrm{R}} \mid w \in \{\mathtt{a}\}^*\}$
    (f) $\{w\#w^{\mathrm{R}} \mid w \in \{\mathtt{a}\}^*\}$
    (g) $\{ww^{\mathrm{R}} \mid w \in \{\mathtt{a},\mathtt{b}\}^*\}$
    (h) $\{w\#w^{\mathrm{R}} \mid w \in \{\mathtt{a},\mathtt{b}\}^*\}$
- Design: Two questions.
  - Show that a language is regular, by writing a DFA, NFA, or regular expression for it (Exercise 1.4–7, 1.18).
  - Show that a language is context-free, by writing a CFG or PDA for it (Exercise 2.4ad, 2.6ac, 2.7ac but actually write the PDA).
- Constructions: One of the following types of questions.
  - Convert a NFA to a DFA (Exercise 1.16).
  - Convert a regular expression to a NFA (Exercise 1.28).
  - Convert a NFA to a regular expression (Exercise 1.21).
  - Convert a CFG to a PDA (Exercise 2.11–12).
  - Convert a PDA to a CFG. You will not be asked to write out all the rules of the form $A_{pq} \to A_{pr}\,A_{rq}$. For a sample question, convert the PDA in Example 2.18 and see below for the solution.
- Proofs: Two of the following types of questions.
  - Prove that regular languages are closed under some operation (Exercise 1.31$^{\mathsf{U}}$/1.36$^{\mathsf{I}}$, Problem 1.43$^{\mathsf{U}}$/1.33$^{\mathsf{I}}$, 1.40a$^{\mathsf{U}}$/1.45a$^{\mathsf{I}}$, 1.66a$^{\mathsf{U}}$/1.60a$^{\mathsf{I}}$).
  - Prove that two variations on finite automata or regular expressions are equivalent (Exercise 1.11).

- Prove that a language is non-regular (Problem 1.29ac, 1.46b$^{\mathsf{U}}$/1.51b$^{\mathsf{I}}$).
- Prove that CFLs are closed under some operation (Exercise 2.16, Problem 2.25$^{\mathsf{U}}$/2.37$^{\mathsf{I}}$).
- Prove that two variations on CFGs or PDAs are equivalent. Sample question: Define an *e-PDA* to be a PDA that does not have accept states; instead, it accepts a string if and only if it can reach the end of the string with an empty stack. Prove that PDAs and e-PDAs recognize the same languages.

Solutions to Selected Exercises

Classifying languages as (F) finite, (R) regular but not finite, or (C) context-free but not regular:

(a) C

(b) F

(c) R

(d) F

(e) R

(f) C

(g) C

(h) C

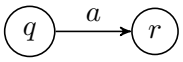
Exercise 1.18

1. $\mathtt{1}\Sigma^*\mathtt{0}$
2. $\Sigma^*\mathtt{1}\Sigma^*\mathtt{1}\Sigma^*\mathtt{1}\Sigma^*$
3. $\Sigma^*\mathtt{0101}\Sigma^*$
4. $\Sigma\Sigma\mathtt{0}\Sigma^*$
5. $(\mathtt{0} \cup \mathtt{1}\Sigma)(\Sigma\Sigma)^*$
6. $\mathtt{0}^*(\mathtt{100}^*)^*\mathtt{1}^*$
7. $(\Sigma \cup \varepsilon)(\Sigma \cup \varepsilon)(\Sigma \cup \varepsilon)(\Sigma \cup \varepsilon)(\Sigma \cup \varepsilon)$
8. $\Sigma^*\mathtt{0}\Sigma^* \cup \varepsilon \cup \mathtt{1} \cup \mathtt{11111}^*$
9. $(\mathtt{1}\Sigma)^*(\mathtt{1} \cup \varepsilon)$
10. $\mathtt{0}^*(\mathtt{00} \cup \mathtt{100} \cup \mathtt{010} \cup \mathtt{001})\mathtt{0}^*$

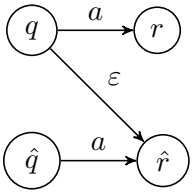
11. $\varepsilon \cup \mathtt{0}$
12. $\mathtt{1}^*(\mathtt{01}^*\mathtt{01}^*)^* \cup \mathtt{0}^*\mathtt{10}^*\mathtt{10}^*$
13. $\emptyset$
14. $\Sigma\Sigma^*$

Problem 1.43$^{\mathsf{U}}$/1.33$^{\mathsf{I}}$ For any regular language L, let $M = (Q, \Sigma, \delta, s, F)$ be a DFA that recognizes L. Construct a new NFA N:

- For every state $q \in Q$, create states q and $\hat{q}$. (Assume that Q does not contain any states whose names have hats, like $\hat{q}$.)
- The start state is s.
- The accept states are $\{\hat{q} \mid q \in F\}$.
- For every transition in M that looks like this:

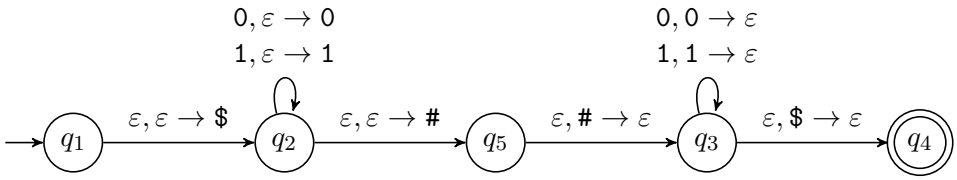


create three transitions in N:



The new NFA N reads the first part of the string (x) simulating M using the states without hats, i.e., q, r. Then, at some point, it follows a transition from q to $\hat{r}$, reading nothing but simulating the reading of symbol a. Then it reads the rest of the string (z) simulating M using the states with hats, i.e., $\hat{q}$, $\hat{r}$.

Example 2.18 The $\varepsilon, \varepsilon \to \varepsilon$ transition first has to be broken up into two transitions:



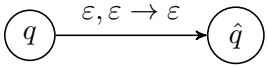
The start symbol is $A_{q_1 q_4}$, and the rules are:

$$\begin{aligned}
A_{q_1 q_4} &\to A_{q_2 q_3} \\
A_{q_2 q_3} &\to \mathtt{0} A_{q_2 q_3} \mathtt{0} \\
A_{q_2 q_3} &\to \mathtt{1} A_{q_2 q_3} \mathtt{1} \\
A_{q_2 q_3} &\to A_{q_5 q_5} \\
A_{q_1 q_1} &\to \varepsilon \\
A_{q_2 q_2} &\to \varepsilon \\
A_{q_3 q_3} &\to \varepsilon \\
A_{q_4 q_4} &\to \varepsilon \\
A_{q_5 q_5} &\to \varepsilon
\end{aligned}$$

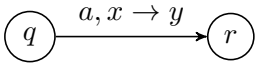
as well as all rules of the form $A_{pq} \to A_{pr} A_{rq}$ for $p, q, r \in \{q_1, q_2, q_3, q_4, q_5\}$.

Problem 2.25$^{\mathsf{U}}$/2.37$^{\mathsf{I}}$ For any context-free language L, let $P = (Q, \Sigma, \Gamma, \delta, s, F)$ be a PDA that recognizes L. Construct a new PDA P':

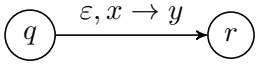
- For every state $q \in Q$, create states q and $\hat{q}$. (Assume that Q does not contain any states whose names have hats, like $\hat{q}$.)
- The start state is s.
- The accept states are $\{\hat{q} \mid q \in F\}$.
- For every state $q \in Q$, create the transition

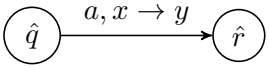


- For every transition in P that looks like this:



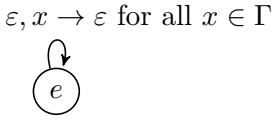
  create two transitions in P':



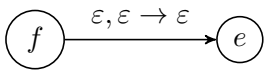


Initially, P' simulates P using the states without hats, i.e., q, r, reading nothing but simulating the reading of some string u. Then, at some point, it follows a transition from q to $\hat{q}$ and continues to simulate P using the states with hats, i.e., $\hat{q}$, $\hat{r}$, and actually reading the string (v).

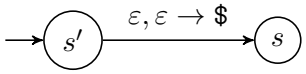
Equivalence of PDAs and e-PDAs Given a PDA P, we can convert it to an equivalent e-PDA as follows. Create the following state and transitions, whose job is to empty the stack completely:



Then for every accept state f, add a transition



Given an e-PDA P with start state s, we can convert it to an equivalent PDA as follows. Create a new start state s' and transition:



where we assume $ is a new stack symbol not used by P. Then create a new accept state f and, for every state q, create a transition that lets the PDA go to the accept state if the stack is empty save for $: