University of Houston

Midterm Exam Solutions

COSC 3340
Introduction to Automata & Computability

Instructor: Rathish Das
TA: Aayush Gupta

Instructions:

- Answer all the questions. Each question carries 10 marks.
- Scanning the Exam:
  - Use a scanning app on your smartphone.
  - Ensure all pages are scanned in the correct order.
  - Check the scans for clarity and legibility.
  - Save the scanned document in a PDF format.
  - Naming the File: Name the file clearly, e.g., "YourName_PSID_Exam.pdf".
- Uploading to Canva:
  - Open your web browser and go to the Canva website.
  - Log in with your credentials.
  - Navigate to the appropriate class or course page where you need to upload the exam.
  - Look for an 'Upload' or 'Submit Assignment' button.
  - Click on the button, and a dialog box will open.
  - Choose the scanned exam file from your computer.
  - Once selected, click 'Open' or 'Upload'.
  - Wait for the file to upload completely.
  - After uploading, you might see a preview or a confirmation message, depending on the system.
- Final Confirmation:
  - Ensure that the file has been uploaded correctly.
  - If there's an option to leave a comment or message, you may mention that you have uploaded the exam.
  - If you receive a confirmation email or notification, keep it for your records.
- Troubleshooting:
  - If you encounter any issues during the upload, try refreshing the page and uploading again.

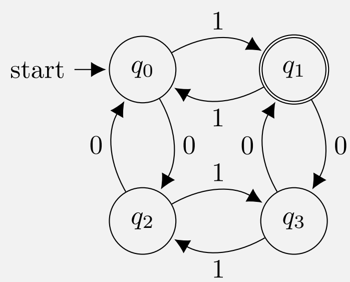
1 Regular Languages

Exercise 1

Design a DFA to accept the language

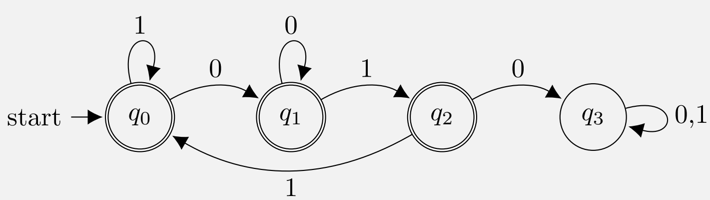
$$L = \{w|w \text{ has an } \textbf{even} \text{ number of 0's and an } \textbf{odd} \text{ number of 1's}\}$$

Solution.



Exercise 2

Find a DFA that accepts all the strings on $\{0, 1\}$, except those containing the substring 010.
Solution.



Exercise 3

Show that the language

$$L = \{a^p b^q : p = q\}$$

is not regular **using pumping lemma**.

Solution. We aim to prove that the language

$$L = \{a^p b^q : p = q\}$$

is not regular using the Pumping Lemma.

Proof

1. Assume that L is regular. According to the Pumping Lemma, there exists a pumping length n such that any string w in L with $|w| \geq n$ can be divided into three parts, $w = xyz$, satisfying the following conditions:
   - $|xy| \leq n$
   - $|y| > 0$
   - For all $i \geq 0$, the string $xy^i z$ is also in L.
2. Choose a string $w = a^n b^n$ from L such that $|w| = 2n \geq n$.
3. Divide w into xyz where $x = a^k$, $y = a^l$ with $l > 0$ and $k + l \leq n$, and $z = a^{n-k-l} b^n$.

4. Consider pumping y. The string xy^2z becomes $a^k a^{2l} a^{n-k-l} b^n = a^{n+l} b^n$. Since $l > 0$, the number of a's in xy^2z is greater than the number of b's, and hence $xy^2z \notin L$.

5. This result contradicts the requirement of the Pumping Lemma. Therefore, our initial assumption that L is regular must be false.

Hence, the language $L = \{a^p b^q : p = q\}$ is not regular.

Exercise 4

Using the fact that the language

$$L = \{a^p b^q : p = q\}$$

is not regular, prove that the following language

$$L^{'} = \{a^p b^q : p \neq q\}$$

is not regular **without using pumping lemma**.

Solution. We aim to prove that the language

$$L' = \{a^p b^q : p \neq q\}$$

is not regular, using the fact that the language

$$L = \{a^p b^q : p = q\}$$

is not regular.

Proof

1. Consider the closure property of regular languages. If a language L is regular, then its complement $\overline{L}$ is also regular.

2. We know that the language $L = \{a^p b^q : p = q\}$ is not regular.

3. The complement of L, denoted as $\overline{L}$, consists of all strings over the alphabet $\{a, b\}$ that are not in L. Formally,

$$\overline{L} = \{w \in \{a, b\}^* : w \text{ does not have the form } a^p b^q \text{ where } p = q\}$$

.

4. The language $L' = \{a^p b^q : p \neq q\}$ is a subset of $\overline{L}$, as it includes all strings where the number of a's is not equal to the number of b's but follows the pattern $a^p b^q$.

5. If L' were regular, then its union with all strings that do not follow the pattern $a^p b^q$ (which is a regular set) would yield $\overline{L}$. This would imply that $\overline{L}$ is regular, contradicting our earlier conclusion that $\overline{L}$ is not regular.

6. Therefore, L' cannot be regular.

Hence, we conclude that the language $L' = \{a^p b^q : p \neq q\}$ is not regular.

2 Context-free Languages

Exercise 5

Write down the context-free grammer of the language

$$L = \{a^p b^q : p > q\}.$$

Solution. Consider the language defined as

$$L = \{a^p b^q : p > q\}.$$

We aim to construct a context-free grammar (CFG) for this language.

Context-Free Grammar

Let S be the start variable. The CFG for the language L is given by the following production rules:

$$\begin{aligned} S &\rightarrow aSb \mid aA \\ A &\rightarrow aA \mid a \end{aligned}$$

In these rules:

- The production $S \rightarrow aSb$ ensures that for every b produced, there is already an a present.
- The production $S \rightarrow aA$ and the productions for A add additional a's, ensuring that the number of a's is always greater than the number of b's.

Thus, this CFG generates all and only the strings in L.

Exercise 6

What is a leftmost and rightmost derivation of the string $abbbb$, considering the grammar with productions

$$\begin{aligned} S &\rightarrow aAB, \\ A &\rightarrow bBb, \\ B &\rightarrow A|\lambda. \end{aligned}$$

Solution. Consider the grammar with the following productions:

$$\begin{aligned} S &\rightarrow aAB, \\ A &\rightarrow bBb, \\ B &\rightarrow A|\lambda. \end{aligned}$$

We will find the leftmost and rightmost derivations for the string $abbbb$.

Leftmost Derivation

The leftmost derivation of the string $abbbb$ is:

$$
\begin{aligned}
S &\Rightarrow aAB \quad (\text{using } S \to aAB) \\
&\Rightarrow abBbB \quad (\text{using } A \to bBb) \\
&\Rightarrow abbBbb \quad (\text{using } B \to A \text{ and then } A \to bBb) \\
&\Rightarrow abbbb \quad (\text{using } B \to \lambda)
\end{aligned}
$$

Rightmost Derivation

The rightmost derivation of the string $abbbb$ is:

$$
\begin{aligned}
S &\Rightarrow aAB \quad (\text{using } S \to aAB) \\
&\Rightarrow aAbB \quad (\text{using } B \to A) \\
&\Rightarrow abBbB \quad (\text{using } A \to bBb) \\
&\Rightarrow abbbB \quad (\text{using } B \to \lambda) \\
&\Rightarrow abbbb \quad (\text{using } A \to bBb)
\end{aligned}
$$

Both derivations result in the string $abbbb$, demonstrating that it is part of the language generated by the given grammar.

Exercise 7

Convert the grammar with productions

$$
\begin{aligned}
S &\to ABc, \\
A &\to aab, \\
B &\to Aa.
\end{aligned}
$$

to Chomsky normal form.

Solution. Given a grammar with the following productions:

$$
\begin{aligned}
S &\to ABc, \\
A &\to aab, \\
B &\to Aa.
\end{aligned}
$$

We aim to convert this grammar to Chomsky Normal Form (CNF).

Conversion Process

To convert the grammar to CNF, we introduce new non-terminal symbols for each terminal and modify the production rules accordingly.
Let X, Y, and Z represent the terminals a, b, and c respectively. The modified grammar in CNF is:

$$
\begin{aligned}
S &\to AXZ, \\
A &\to XY, \\
B &\to AX, \\
X &\to a, \\
Y &\to b, \\
Z &\to c.
\end{aligned}
$$

In this form, each production rule either produces two non-terminals or a single terminal, complying with the requirements of CNF.

Exercise 8

Give a Pushdown Automata recognizing the language

$$\{ww^{\mathcal{R}}|w \in \{a,b\}^*\},$$

where $w^{\mathcal{R}}$ is the reverse string of w.
Solution. We describe a Pushdown Automaton (PDA) for the language

$$\{ww^{\mathcal{R}}|w \in \{a,b\}^*\},$$

where $w^{\mathcal{R}}$ is the reverse string of w.

PDA Description

The PDA is defined as follows:

- States: q_0, q_1, q_2
- Input alphabet: $\Sigma = \{a, b\}$
- Stack alphabet: $\Gamma = \{a, b, Z_0\}$ where Z_0 is the initial stack symbol
- Transition function: δ
- Initial state: q_0
- Initial stack symbol: Z_0
- Accepting state: q_2

Transitions

$$\begin{aligned}
\delta(q_0, a, Z_0) &= \{(q_0, aZ_0), (q_1, Z_0)\} \\
\delta(q_0, b, Z_0) &= \{(q_0, bZ_0), (q_1, Z_0)\} \\
\delta(q_0, a, a) &= \{(q_0, aa), (q_1, a)\} \\
\delta(q_0, b, b) &= \{(q_0, bb), (q_1, b)\} \\
\delta(q_1, a, a) &= \{(q_1, \lambda)\} \\
\delta(q_1, b, b) &= \{(q_1, \lambda)\} \\
\delta(q_1, \lambda, Z_0) &= \{(q_2, Z_0)\}
\end{aligned}$$

This PDA works by non-deterministically guessing the midpoint of the input string and then matching the second half with the reversed first half stored in the stack.