CS/ECE-374-B: Algorithms and Models of Computation, Spring 2024
Midterm exam 1 – February 15, 2024

• You can do hard things! Grades do matter, but not as much as you may think, but then life is uncertain anyway, so what.

• **Don't cheat.** The consequence for cheating is far greater than the reward. Just try your best and you'll be fine.

• **Please read the entire exam before writing anything.** There are 8 problems and most have multiple parts.

• This is a closed-book exam. At the end of the exam, you'll find a multi-page cheat sheet. *Do not tear out the cheat sheet!* No outside material is allowed on this exam.

• You should write your answers legibly and in the space given for the question. Overly verbose answers will be penalized.

• Scratch paper is available on the back of the exam. *Do not tear out the scratch paper!* It messes with the auto-scanner.

• **You have 75 minutes (1.25 hours) for the exam.** Manage your time well. *Do not spend too much time on questions you do not understand and focus on answering as much as you can!*

Name: ________________________________

NetID: ________________________________

Date: _________________________________
Problem 1 [20 points]

a. Write the recursive definition for the following language. 

\[ L_a = \{ w \mid w \in \{0, 1\}^*, |w| = 2n \text{ for some } n \geq 0, \text{ and } w = w^R \text{ where } w^R \text{ is the reverse of } w. \} \]

**Solution:** \( L_a \) represents the language of all even-length binary palindromes. The recursive definition is given as follows.

- \( w = \epsilon \), or
- \( w = axa \) for some \( a \in \{0, 1\} \) and \( x \in L_a \).

b. Write regular expressions for the following languages. 

i. \( L_{bi} = \{ w \mid w \in \{0, 1\}^*, w \text{ does not contain the subsequence } 00. \} \)

**Solution:** \( L_{bi} \) represents the language of all binary strings with number of zeros less than or equal to 1. The regular expression is given as follows.

\[ \epsilon + 1^* + 1^*01^* \]

ii. \( L_{bii} = \{ w \mid w \in \{0, 1\}^*, w \text{ contains } 00 \text{ and } 11 \text{ as subsequences.} \} \)

**Solution:** Let \( S \) be the set of all strings of length 4 that have the desired property. That is, we have the following.

\[ S = \{ 0011, 0101, 0110, 1001, 1100, 1010 \} \].

There are \( \binom{4}{2} = 6 \), such strings, and they all belong to our language. For a string \( s \), let \( f(s) \) be the regular expression of inserting \( (0 + 1)^* \) between any two characters of \( s \), and also in the beginning of \( s \) and the end of \( s \). For example, we have \( f(01) = (0 + 1)^*0(0 + 1)^*1(0 + 1)^* \). The desired expression is

\[ \sum_{s \in S} f(s). \]
Problem 2 [10 points]

Consider the state diagram given in Figure 1.

![State Diagram](image)

Figure 1.

(a) Write the corresponding nondeterministic finite automaton in a formal manner. [5]

**Solution:** For the given state diagram, the corresponding finite automaton in a formal manner is as follows.

- $Q = \{q_1, q_2, q_3, q_4\}$.
- $\Sigma = \{0, 1\}$.
- $\delta$ is given as follows.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>${q_1}$</td>
<td>${q_1, q_2}$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>${q_3}$</td>
<td>$\phi$</td>
<td>${q_3}$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\phi$</td>
<td>${q_4}$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>${q_4}$</td>
<td>${q_4}$</td>
<td>$\phi$</td>
</tr>
</tbody>
</table>

- $q_0 = q_1$.
- $F = \{q_4\}$.

(b) What ALL sequences of states does the above machine go through on inputs 010 and 010110? Note that there may be multiple sequences of states for the same input. Does the machine accept these inputs? **Hint.** Draw the computation tree for each input. [5]

**Solution:** The required sequences of states for 010 and 010110 are given in the computation trees in Figures 2 and 3, respectively. Furthermore, based on those trees, we conclude that the machine does not accept 010 and accepts 010110.
Figure 2. Computation tree for 010.
Figure 3. Computation tree for 010110.
Problem 3 [20 points]

a. Convert the NFA given in Figure 4 to an equivalent DFA. [10]

\[\begin{array}{c}
\text{start} \rightarrow q_0 \xrightarrow{1} q_1 \xrightarrow{0,1} q_2
\end{array}\]

\textbf{Figure 4.}

\textbf{Solution:} The formal representation of the above NFA upon conversion (using the incremental method) to a DFA is as follows.

(a) \(Q = \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\} \).

(b) \(\Sigma = \{0, 1\}\).

(c) \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function given as follows.

\[
\begin{array}{c|cc}
\delta & 0 & 1 \\
\hline
\{q_0\} & \{q_0\} & \{q_0, q_1\} \\
\{q_0, q_1\} & \{q_0, q_2\} & \{q_0, q_1, q_2\} \\
\{q_0, q_2\} & \{q_0\} & \{q_0, q_1\} \\
\{q_0, q_1, q_2\} & \{q_0, q_2\} & \{q_0, q_1, q_2\}
\end{array}
\]

(d) \(q_0 = \{q_0\}\).

(e) \(F = \{\{q_0, q_2\}, \{q_0, q_1, q_2\}\}\).

b. For \(\Sigma = \{a, b\}\), convert the regular expression \((a + b)^{*}aba\) to an equivalent NFA. [10]

\textbf{Solution:} The step-by-step construction of an equivalent NFA is shown in Figure 5. Interpret \(\cup\) as +.
Figure 5. Regular expression to NFA.
Problem 4 [10 points]

Given a regular language $L$ over $\{0, 1\}^*$, prove that the language $L' := \{xy \mid x1y \in L\}$ is regular.

**Solution:** Intuitively, $L'$ is the set of all strings that can be obtained from strings in $L$ by deleting exactly one 1. For example, if $L = \{101101, 00, \epsilon\}$, then $L' = \{01101, 10101, 10110\}$.

Let $M = (Q, s, A, \delta)$ be a DFA that accepts $L$. We construct an NFA $M' = (Q', s', A', \delta')$ with $\epsilon$-transitions that accepts $L'$ as follows.

Intuitively, $M'$ simulates $M$, but inserts a single 1 into $M$’s input string at a nondeterministically chosen location.

- The state $(q, \text{before})$ means (the simulation of) $M$ is in state $q$ and $M'$ has not yet inserted a 1.

- The state $(q, \text{after})$ means (the simulation of) $M$ is in state $q$ and $M'$ has already inserted a 1.

\[
Q' := Q \times \{\text{before, after}\} \\
S' := (s, \text{before}) \\
A' := \{(q, \text{after}) \mid q \in A\} \\
\delta'((q, \text{before}), \epsilon) = \{(\delta(q, 1), \text{after})\} \\
\delta'((q, \text{after}), \epsilon) = \emptyset \\
\delta'((q, \text{before}), a) = \{(\delta(q, a), \text{before})\} \\
\delta'((q, \text{after}), a) = \{(\delta(q, a), \text{after})\}
\]

$\square$
Problem 5 [15 points]

Consider the PDA, P given in Figure 6.

![PDA Diagram]

**Figure 6.**

a. Does the above PDA accept the following strings? [4]

i. 1010
ii. 0101
iii. 1001
iv. 1011

**Solution:** The solution is given as follows.

i. No.
ii. No.
iii. Yes.
iv. Yes.

b. Describe $L(P)$ in one sentence. [3]

**Solution:** All binary strings of length greater than 1 that start with 1 and end with 1.
c. Give a context free grammar that generates $L(P)$. [4]

**Solution:** The context free $G = (V, T, P, S)$ that generates $L(P)$ is given as follows.

- $V = \{S, A\}$
- $T = \{0, 1\}$
- $P : S \rightarrow 1A1, A \rightarrow \epsilon \mid 0A \mid 1A$
- $S = S$

---

d. Which one of the following statements is true? [4]

i. $L(P)$ is context sensitive but not regular.

ii. $L(P)$ is not context sensitive but regular.

iii. $L(P)$ is both context sensitive and regular.

iv. $L(P)$ is neither context sensitive nor regular.

Prove the statement you chose.

**Solution:** The true statement is iii, i.e., $L(P)$ is both context sensitive and regular. The regular expression $1(0 + 1)^*1$ generates $L(P)$. Hence, $L(P)$ is regular, and hence, context-sensitive.
Problem 6 [10 points]

Consider a context free grammar \( G = (V, T, P, S) \), where

- \( V = \{B, E\} \)
- \( T = \{a, b\} \)
- \( P = \{B \rightarrow a \mid aEb; E \rightarrow \epsilon \mid Ea\} \)
- \( S = B \)

Construct a PDA that recognizes \( L(G) \).

**Solution:** The PDA that recognizes \( L(G) \) is given in Figure 7.
Problem 7 [5 points]

Let $L_1, \ldots, L_n$ be some regular languages and $L_k$ be a non-regular language such that

$$L_k = L_1 \oplus L_2 \oplus \cdots \oplus L_n \oplus L_u, \quad \oplus \in \{\cup, \cap, \cdot\},$$

for some $L_u$ then (formally) prove that $L_u$ is non-regular.

**Solution:** By way of contradiction, assume that $L_u$ is regular. Hence, $L_k = L_1 \oplus L_2 \oplus \cdots \oplus L_n \oplus L_u, \quad \oplus \in \{\cup, \cap, \cdot\}$ is regular as regular languages are closed under union, intersection, and concatenation. But, $L_k$ is given to be non-regular. Hence, a contradiction. This implies that $L_u$ is regular. ■
Problem 8 [10 points]

For each of the following languages defined over $\Sigma = \{a, b, c\}$, prove if it is regular or not. Furthermore, prove if it is context free or not.

a. $L_1 = \{a^n b^n | 0 \leq n \leq 3\}$

Solution: The solution is given as follows.

- $L_1$ is a finite language. As every finite language is regular, $L_1$ is regular.
- As every regular language is also context free, $L_1$ is context free.

b. $L_2 = \{a^m b^n c^n | m, n \geq 0\}$

Solution: The solution is given as follows.

- Consider the fooling set: $\{a^i : i \geq 0\}$. Note that, for $i \neq j$, $m \geq 0$, $a^i b^m c^i \in L_2$ but $a^j b^m c^i \notin L_2$. Hence, $L_2$ is non-regular.
- Consider the following context free grammar $G = (V, T, P, S)$, where
  - $V = \{S, X\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSc \mid X \mid \epsilon; X \rightarrow bX \mid \epsilon\}$
  - $S = S$

  $G$ generates $L_2$. Hence, $L_2$ is context free.

 c. $L_3 = \{a^n b^m | n > m \text{ or } m > n\}$

Solution: The solution is given as follows.

- Consider the fooling set: $\{a^i : i \geq 0\}$. Note that, for $i \neq j$, $a^i b^j \notin L_3$ but $a^i b^i \in L_3$. Hence, $L_3$ is non-regular.
- Consider the following context free grammar $G = (V, T, P, S)$, where
  - $V = \{S, X, Y\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSb \mid X \mid Y; X \rightarrow aX \mid a; Y \rightarrow bY \mid b\}$
  - $S = S$

  $G$ generates $L_3$. Hence, $L_3$ is context free.
This page is for additional scratch work!
1 Languages and strings

**Languages**

- An alphabet $\Sigma$ is a finite set of symbols.
- A string in $\Sigma^*$ is a finite sequence of symbols in $\Sigma$.
- A language is $L$ is a set of strings over some alphabet.

All languages represent mathematical problems. For example, multiplication of two integers:

$$L_{\text{MULT}} = \{ 1 \times 1, 1 \times 2, 1 \times 3, \ldots, 2 \times 1, 2 \times 2, 2 \times 3, \ldots \} \quad (1)$$

- For languages $A, B$ the concatenation of $A, B$ is $AB = \{ xy \mid x \in A, y \in B \}$.
- For languages $A, B$, their union is $A \cup B$, intersection is $A \cap B$, and difference is $A \setminus B$ (also written as $A - B$).
- For language $A \subseteq \Sigma^*$, the complement of $A$ is $\bar{A} = \Sigma^* \setminus A$.
- $\Sigma^n$ is the set of all strings of length $n$.
- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ is the set of all strings over $\Sigma$.
- $\Sigma^+ = \cup_{n \geq 1} \Sigma^n$ is the set of non-empty strings over $\Sigma$.

**Operations**

- $\gamma \rightarrow \alpha$ (no constraints)
- $\alpha A \beta \rightarrow \alpha \gamma \beta$ (with $\gamma$ replacing $A$)
- $A \rightarrow \alpha$ (with $\alpha$ replacing $A$)
- $A \rightarrow aB$ (with $a$ replacing $A$)

**Strings**

- The length of a string $w$ (denoted by $|w|$) is the number of symbols in $w$.
- For integer $n > 0$, $\Sigma^n$ is set of all strings over $\Sigma$ of length $n$.
- $\Sigma^*$ is the set of all strings over $\Sigma$.
- $\epsilon$ is a string containing no symbols.
- $\emptyset$ is the empty set. It contains no strings.

- If $x$ and $y$ are strings then $xy$ denotes their concatenation. Recursively:
  - $xy = y$ if $x = \epsilon$
  - $xy = a(wy)$ if $x = a$
- $v$ is substring of $w$ if there exist strings $x, y$ such that $w = xy$.
  - If $x = \epsilon$ then $v$ is a prefix of $w$.
  - If $y = \epsilon$ then $v$ is a suffix of $w$.
- A subsequence of a string $w = w_1 w_2 \ldots w_n$ is either a subsequence of $w_2 \ldots w_n$ or $w_1$ followed by a subsequence of $w_2 \ldots w_n$.
- If $w$ is a string then $w^n$ is defined inductively as follows:
  - $\epsilon^n = \epsilon$ if $n = 0$ or $w^n = w w^{n-1}$ if $n > 0$

2 Overview of language complexity

**Overview**

- **recursively enumerable**
- **context-sensitive**
- **context-free**
- **regular**

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Languages</th>
<th>Production Rules</th>
<th>Automaton</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type-0</td>
<td>recursively enumerable</td>
<td>$\gamma \rightarrow \alpha$ (no constraints)</td>
<td>Turing machine</td>
<td>$L = { w</td>
</tr>
<tr>
<td>Type-1</td>
<td>context-sensitive</td>
<td>$\alpha A \beta \rightarrow \alpha \gamma \beta$</td>
<td>linear bounded nondeterministic Turing machine</td>
<td>$L = { a^n b^n c^n</td>
</tr>
<tr>
<td>Type-2</td>
<td>context-free</td>
<td>$A \rightarrow \alpha$</td>
<td>nondeterministic pushdown automata</td>
<td>$L = { a^n b^n</td>
</tr>
<tr>
<td>Type-3</td>
<td>regular</td>
<td>$A \rightarrow aB$</td>
<td>finite state machine</td>
<td>$L = { a^n</td>
</tr>
</tbody>
</table>

Meaning of symbols:
- $\alpha$ - terminal
- $A, B$ - variables
- $\alpha, \beta, \gamma$ - strings in $(\alpha \cup A)^*$ where $\alpha, \beta$ are maybe empty, $\gamma$ is never empty

*Table borrowed from Wikipedia: https://en.wikipedia.org/wiki/Chomsky_hierarchy*
Regular language - overview

A language is regular if and only if it can be obtained from finite languages by applying:
- union,
- concatenation or
- Kleene star

finnitly many times. All regular languages are representable by regular grammars, DFAs, NFAs and regular expressions.

Regular expressions

Useful shorthand to denotes a language.
A regular expression $r$ over an alphabet $\Sigma$ is one of the following:

Base cases:
- $\emptyset$ denotes the language $\emptyset$
- $\varepsilon$ denotes the language $\{\varepsilon\}$
- $a$ denotes the language $\{a\}$

Inductive cases: If $r_1$ and $r_2$ are regular expressions denoting languages $L_1$ and $L_2$ respectively (i.e. $L(r_1) = L_1$ and $L(r_2) = L_2$) then,
- $r_1 \cup r_2$ denotes the language $L_1 \cup L_2$
- $r_1 \cdot r_2$ denotes the language $L_1 L_2$
- $r_1^*$ denotes the language $L_1^*$

Examples:
- $0^*$ - the set of all strings of 0s, including the empty string
- $(000000)^*$ - set of all strings of 0s with length a multiple of 5
- $(0 + 1)^*$ - set of all binary strings

Deterministic finite automata

DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.
The language accepted (or recognized) by a DFA $M$ is denoted by $L(M)$ and defined as $L(M) = \{w \mid M \text{ accepts } w\}$.
A deterministic finite automaton (DFA) $M = (Q, \Sigma, s_0, A, \delta)$ is a five tuple where:
- $Q$ is a finite set whose elements are called states
- $\Sigma$ is a finite set called the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- $s_0 \in Q$ is the start state
- $A \subseteq Q$ is the set of accepting/final states

Example:

\begin{align*}
Q &= \{q_0, q_1\} \\
\Sigma &= \{0, 1\} \\
\delta &= \begin{cases}
\varepsilon : q_0 \rightarrow q_0 \\
0 : q_0 \rightarrow q_1 \\
1 : q_0 \rightarrow q_2 \\
\varepsilon : q_1 \rightarrow q_1 \\
\varepsilon : q_2 \rightarrow q_3 \\
\end{cases} \\
s &= q_0 \\
A &= \{q_1\}
\end{align*}

Every string has a unique walk along a DFA. We define the extended transition function as $\delta^* : Q \times \Sigma^* \rightarrow Q$ defined inductively as follows:
- $\delta^*(q, \varepsilon) = q$ if $w = \varepsilon$
- $\delta^*(q, w) = \delta^*(\delta(q, x), a)$ if $w = ax$

Can create a larger DFA from multiple smaller DFAs. Suppose:
- $L(M_0) = \{w \mid N \text{ accepts } w\}$ pictured above and
- $L(M_1) = \{w \mid N \text{ accepts } w\}$

For DFA $N = (Q, \Sigma, A, s_0, A)$ and $q \in Q$, the $\varepsilon$-reach($q$) is the set of all states that $q$ can reach using only $\varepsilon$-transitions.

Inductive definition of $\delta^* : Q \times \Sigma^* \rightarrow Q$:
- if $w = \varepsilon$, $\delta^*(q, w) = \varepsilon$-reach($q$)
- if $w = a$ for $a \in \Sigma$, $\delta^*(q, a) = \varepsilon$-reach($\bigcup_{p \in \varepsilon$-reach($q$)} \delta(p, a)$)
- if $w = ax$ for $a \in \Sigma, x \in \Sigma^*$, $\delta^*(q, w) = \varepsilon$-reach($\bigcup_{p \in \varepsilon$-reach($q$)} \bigcup_{r \in \delta^*(p, a)} \delta^*(r, x)$)

Regular language equivalences

A regular language can be represented by a regular expression, regular grammar, DFA and NFA.

Fooling sets

Some languages are not regular (Ex. $L = \{0^n1^n \mid n \geq 0\}$).
Two states $p, q \in Q$ are distinguishable if there exists a string $w \in \Sigma^*$ such that $\delta^*(p, w) \in A$ and $\delta^*(q, w) \notin A$.
Two states $p, q \in Q$ are equivalent if for all strings $w \in \Sigma^*$, we have that
- $\delta^*(p, w) \in A \iff \delta^*(q, w) \in A$.

For a language $L$ over a set of strings $F$ (could be infinite) is a fooling set or distinguishing set for $L$ if every two distinct strings $x, y \in F$ are distinguishable.

Regular closure

Regular languages are closed under union, intersection, complement, difference, reversal, Kleene star, concatenation, etc.
Context-free languages

A language is context-free if it can be generated by a context-free grammar. A context-free grammar is a quadruple \( G = (V, T, P, S) \)

\begin{itemize}
  \item \( V \) is a finite set of nonterminal (variable) symbols
  \item \( T \) is a finite set of terminal symbols (alphabet)
  \item \( P \) is a finite set of productions, each of the form \( A \rightarrow \alpha \) where \( A \in V \) and \( \alpha \) is a string in \( (V \cup T)^* \). Formally, \( P \subseteq V \times (V \cup T)^* \).
  \item \( S \in V \) is the start symbol
\end{itemize}

Example: \( L = \{ww \mid w \in \{0, 1\}^*\} \) is described by \( G = (V, T, P, S) \) where \( V, T, P \) and \( S \) are defined as follows:

\begin{itemize}
  \item \( V = \{S\} \)
  \item \( T = \{0, 1\} \)
  \item \( P = \{S \rightarrow \varepsilon, S \rightarrow 0S0, S \rightarrow 1S1\} \)
  \item \( S = S \)
\end{itemize}

Pushdown automata

A pushdown automaton is an NFA with a stack. The language \( L = \{0^n1^n \mid n \geq 0\} \) is recognized by the pushdown automaton:

\begin{itemize}
  \item \( Q \) is a finite set whose elements are called states
  \item \( \Sigma \) is a finite set called the input alphabet
  \item \( \Gamma \) is a finite set called the stack alphabet
  \item \( \delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow P(Q \times (\Gamma \cup \{\varepsilon\})) \) is the transition function
  \item \( s \) is the start state
  \item \( A \) is the set of accepting states
\end{itemize}

In the graphical representation of a PDA, transitions are typically written as \((\text{input read}), (\text{stack pop}) \rightarrow (\text{stack push})\).

A CFG can be converted to a pushdown automaton.

Context-free closure

Context-free languages are closed under union, concatenation, and Kleene star. They are not closed under intersection or complement.