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. [6]

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

b. Write regular expressions for the following languages. [7+7]

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

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

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]
Problem 3 [20 points]

a. Convert the NFA given in Figure 2 to an equivalent DFA. [10]

b. For $\Sigma = \{a, b\}$, convert the regular expression $(a + b)^*aba$ to an equivalent NFA. [10]
Problem 4 [10 points]

Given a regular language $L$ over $\{0, 1\}^*$, prove that the language $L' := \{xy \mid x1y \in L\}$ is regular. [10]
Problem 5  [15 points]

Consider the PDA, $P$ given in Figure 3.

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

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

b. Describe $L(P)$ in one sentence. [3]
c. Give a context free grammar that generates $L(P)$. [4]

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.
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 $L(G)$. [10]
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.
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. $L_1 = \{a^n b^n \mid 0 \leq n \leq 3\}$

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

c. $L_3 = \{a^n b^m \mid n > m \text{ or } m > n\}$
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. Example: multiplication of two integers:

$$L_{MULT_2} = \left\{ 1 \times 1, 1 \times 2, 1 \times 3, \ldots, 2 \times 1, 2 \times 2, 2 \times 3, \ldots, n \times 1, n \times 2, n \times 3, \ldots \right\}$$

- For languages $A$, $B$ the concatenation of $A$, $B$ is $AB = \{ xy | x \in A, y \in B \}$.
- 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 $A^c = \Sigma^* \setminus A$.
- $\Sigma^n$ 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 $\Sigma$.

Strings

- The length of a string $w$ (denoted by $|w|$) is the number of symbols in $w$.
- For integer $n \geq 0$, $\Sigma^n$ is set of all strings over $\Sigma$ of length $n$.
- $\Sigma^*$ is the set of all strings over $\Sigma$.

Definitions

- A string $\varepsilon$ 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 $x = \varepsilon$
  - $xy = a(wy)$ if $x = a$.
- $v$ is a substring of $w$ if there exist strings $x, y$ such that $w = xvy$.
- If $x = \varepsilon$ then $v$ is a prefix of $w$.
- If $y = \varepsilon$ then $v$ is a suffix of $w$.
- A subsequence of a string $w = w_1w_2\ldots w_n$ is either a subsequence of $w_2 \ldots w_n$ or $w_2$ followed by a subsequence of $w_3 \ldots w_n$.
- If $w$ is a string then $w^n$ is defined inductively as follows: $w^0 = \varepsilon$ if $n = 0$ or $w^n = w\,w^{n-1}$ if $n > 0$.

Language operations

- Type-0: recursively enumerable
  - $\gamma$ $\rightarrow$ $\alpha$ (no constraints)
  - Grammar is $\gamma$.
  - Languages is $L = \{ w | w$ is a TM which halts $\}$.

- Type-1: context-sensitive
  - $\alpha A\beta$ $\rightarrow$ $\alpha \gamma \beta$
  - Grammar is $\alpha A \beta$.
  - Languages is $L = \{ a^n b^n \varepsilon^n | n > 0 \}$.

- Type-2: context-free
  - $A$ $\rightarrow$ $\alpha$
  - Grammar is $A$.
  - Languages is $L = \{ a^n b^n | n > 0 \}$.

- Type-3: regular
  - $A$ $\rightarrow$ $aB$
  - Grammar is $A = aB$.
  - Languages is $L = \{ a^n | n > 0 \}$.

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 \varepsilon^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 languages

A language is regular if and only if it can be obtained from finite languages by applying

- union,
- concatenation or
- Kleene star

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

Regular language - overview

Useful shorthand to denote 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,

- $r_1 + r_2$ 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

Regular expressions

Non-deterministic finite automata

NFAs are similar to DFAs, but may have more than one transition destination for a given state/character pair.

An NFA $N$ accepts a string $w$ if some accepting state is reached by $N$ from the start state on input $w$.

The language accepted by an NFA $N$ is denoted $L(N)$ and defined as $L(N) = \{ w \mid N \text{ accepts } w \}$.

A non-deterministic finite automaton (NFA) $N = (Q, \Sigma, \delta, q_0, F)$ is a five tuple where:

- $Q$ is a finite set whose elements are called states
- $\Sigma$ is a finite set called the input alphabet
- $\delta : Q \times \Sigma \rightarrow P(Q)$ is the transition function (where $P(Q)$ is the power set of $Q$)
- $q_0$ and $\Sigma$ are the same as in DFAs
- $F \subseteq Q$ is the set of accepting states

Example:

$$
\begin{array}{c|cccc}
\delta & 0 & 1 & \varepsilon & 2 \\
\hline
q_0 & \{q_0\} & \{q_0\} & \{q_0, q_1\} & \{q_0\} \\
q_1 & \{q_2\} & \{q_3\} & \{q_3\} & \{q_3\} \\
q_2 & \{q_3\} & \{q_3\} & \{q_3\} & \{q_3\} \\
q_3 & \{q_3\} & \{q_3\} & \{q_3\} & \{q_3\} \\
\end{array}
$$

For NFA $N = (Q, \Sigma, \delta, q_0, F)$ 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 $\varepsilon$-reach($q$) is the set of all states that $q$ can reach using only $\varepsilon$-transitions:

- if $q = \varepsilon$, $\varepsilon$-reach($q$) = $\{q\}$
- if $q = aA$ for $a \in \Sigma$, $\varepsilon$-reach($aA$) = $\bigcup_{p \in \varepsilon$-reach($q$)} \delta(p, a)$
- if $q = ax$ for $a \in \Sigma, x \in \varepsilon^*$, $\varepsilon$-reach($ax$) = $\bigcup_{p \in \varepsilon$-reach($q$)} \delta\varepsilon(p, a)$

Regular closure

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

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) $M = (Q, \Sigma, s, A, \delta)$ 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
- $s \in Q$ is the start state
- $A \subseteq Q$ is the set of accepting/final states

Example:

$$
\begin{array}{c|cccc}
\delta & 0 & 1 & \varepsilon & 2 \\
\hline
q_0 & q_1 & q_0 & q_0 & q_0 \\
q_1 & q_0 & q_1 & q_1 & q_1 \\
q_2 & q_0 & q_1 & q_1 & q_1 \\
q_3 & q_0 & q_1 & q_1 & q_1 \\
\end{array}
$$

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 $w \in \varepsilon^*$ such that $\delta^*(p, w) \in A$ and $\delta^*(q, w) \notin A$ or $\delta^*(p, w) \notin A$ and $\delta^*(q, w) \in A$.

For a language $L$ over $\Sigma$ 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.
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 $G = (V, T, P, S)$

- $V$ is a finite set of nonterminal (variable) symbols
- $T$ is a finite set of terminal symbols (alphabet)
- $P$ 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, $P \subseteq V \times (V \cup T)^*$.
- $S \in V$ is the start symbol

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

- $V = \{S\}$
- $T = \{0, 1\}$
- $P = \{S \rightarrow \varepsilon, S \rightarrow 0S0, S \rightarrow 1S1\}$ (abbreviation for $S \rightarrow \varepsilon, S \rightarrow 0S0, S \rightarrow 1S1$)
- $S = S$

Pushdown automata

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

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

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.