# 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. [6]

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

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

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

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

## Problem 2 [10 points]

Consider the state diagram given in Figure 1 .

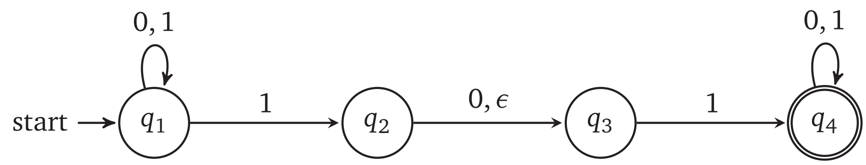



**Figure 1.**

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]

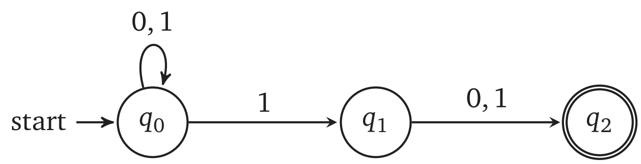



**Figure 2.**

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

## Problem 4 [10 points]

Given a regular language $L$ over $\{0,1\}^*$, prove that the language $L' := \{xy \mid x\mathbf{1}y \in L\}$ is regular. [10]

# Problem 5 [15 points]

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

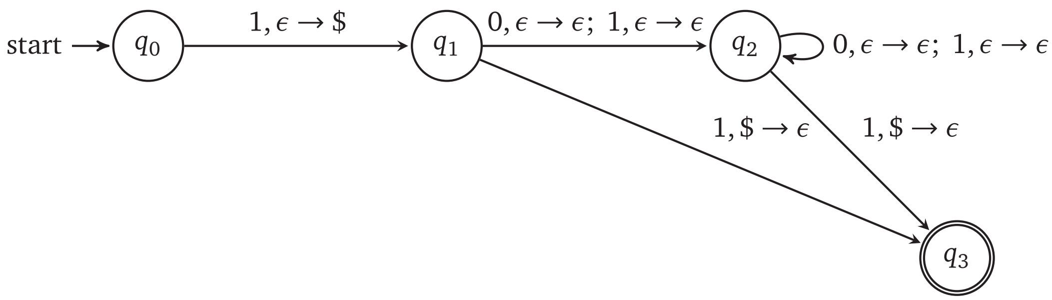



**Figure 3.**

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

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

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

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

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 $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, prove if it is regular or not. Furthermore, prove if it is context free or not.

a. $L_1 = \{\mathtt{a}^n\mathtt{b}^n \mid 0 \leq n \leq 3\}$ [3]

b. $L_2 = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n \mid m, n \geq 0\}$ [3]

c. $L_1 = \{\mathtt{a}^n\mathtt{b}^m \mid n > m \text{ or } m > n\}$ [3]

*This page is for additional scratch work!*

# ECE 374 B Language Theory: Cheatsheet

## 1 Languages and strings

### Languages

**Definitions**

- 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:

$$L_{MULT2} = \left\{ \begin{matrix} 1 \times 1|1, & 1 \times 2|2, & 1 \times 3|3, \dots \\ 2 \times 1|2, & 2 \times 2|4, & 2 \times 3|6, \dots \\ \vdots & \vdots & \vdots \\ n \times 1|n, & n \times 2|2n, & n \times 3|3n, \dots \end{matrix} \right\} \tag{1}$$

**Language operations**

- 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 $\Sigma$.

### Strings

**Definitions**

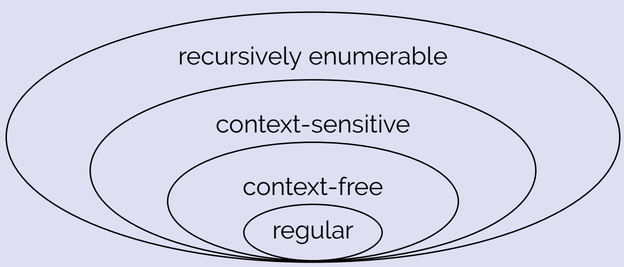
- The *length* of a string $w$ (denoted by $|w|$) is the number of symbols in $w$.
- For integer $n \geq 0$, $\Sigma^n$ is set of all strings over $\Sigma$ of length $n$. $\Sigma^*$ is the set of all strings over $\Sigma$.
- $\Sigma^*$ is the set of all strings of all lengths including empty string.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

**String operations**

- If $x$ and $y$ are strings then $xy$ denotes their concatenation. Recursively:
  - $xy = y$ if $x = \varepsilon$
  - $xy = a(wy)$ if $x = aw$
- $v$ is *substring* of $w \iff$ 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 $w = w_1 w_2 \dots w_n$ is either a subsequence of $w_2 \dots w_n$ or $w_1$ followed by a subsequence of $w_2 \dots w_n$.
- If $w$ is a string then $w^n$ is defined inductively as follows: $w^n = \varepsilon$ if $n = 0$ or $w^n = ww^{n-1}$ if $n > 0$

## 2 Overview of language complexity

### Overview




| Grammar | Languages | Production Rules | Automaton | Examples |
|---|---|---|---|---|
| Type-0 | recursively enumerable | $\gamma \to \alpha$ (no constraints) | Turing machine | $L = \{w \mid w$ is a *TM* which halts$\}$ |
| Type-1 | context-sensitive | $\alpha A \beta \to \alpha \gamma \beta$ | linear bounded nondeterministic Turing machine | $L = \{a^n b^n c^n \mid n > 0\}$ |
| Type-2 | context-free | $A \to \alpha$ | nondeterministic pushdown automata | $L = \{a^n b^n \mid n > 0\}$ |
| Type-3 | regular | $A \to aB$ | finite state machine | $L = \{a^n \mid n > 0\}$ |

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

[a]

[a] Table borrowed from Wikipedia: `https://en.wikipedia.org/wiki/Chomsky_hierarchy`

# 3 Regular languages

## 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

finitely 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* $\mathbf{r}$ over an alphabet $\Sigma$ is one of the following:
**Base cases:**

- $\varnothing$ the language $\varnothing$
- $\varepsilon$ denotes the language $\{\varepsilon\}$
- $a$ denote the language $\{a\}$

**Inductive cases:** If $\mathbf{r_1}$ and $\mathbf{r_2}$ are regular expressions denoting languages $L_1$ and $L_2$ respectively (i.e.,$L(\mathbf{r_1}) = L_1$ and $L(\mathbf{r_2}) = L_2$) then,

- $\mathbf{r_1} + \mathbf{r_2}$ denotes the language $L_1 \cup L_2$
- $\mathbf{r_1} \bullet \mathbf{r_2}$ denotes the language $L_1 L_2$
- $\mathbf{r_1^*}$ denotes the language $L_1^*$

**Examples:**

- $0^*$ - the set of all strings of 0s, including the empty string
- $(00000)^*$ - set of all strings of 0s with length a multiple of 5
- $(0+1)^*$ - set of all binary strings

## Nondeterministic finite automata

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

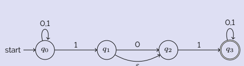
An NFA $N$ *accepts a string* $w$ iff some accepting state is reached by $N$ from the start state on input $w$.

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

A *nondeterministic finite automaton (NFA)* $N = (Q, \Sigma, s, A, \delta)$ is a five tuple where

- $Q$ is a finite set whose elements are called *states*
- $\Sigma$ is a finite set called the *input alphabet*
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the *transition function* (here $\mathcal{P}(Q)$ is the power set of $Q$)
- $s$ and $\Sigma$ are the same as in DFAs

Example:




- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta$ :

| | $\varepsilon$ | 0 | 1 |
|---|---|---|---|
| $q_0$ | $\{q_0\}$ | $\{q_0\}$ | $\{q_0, q_1\}$ |
| $q_1$ | $\{q_1, q_2\}$ | $\{q_2\}$ | $\varnothing$ |
| $q_2$ | $\{q_2\}$ | $\varnothing$ | $\{q_3\}$ |
| $q_3$ | $\{q_3\}$ | $\{q_3\}$ | $\{q_3\}$ |

- $s = q_0$
- $A = \{q_3\}$

For NFA $N = (Q, \Sigma, \delta, s, A)$ 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 $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$:

- if $w = \varepsilon$, $\delta^*(q, w) = \varepsilon\text{-reach}(q)$
- if $w = a$ for $a \in \Sigma$, $\quad \delta^*(q, a) = \varepsilon\text{reach}\left(\bigcup_{p \in \varepsilon\text{-reach}(q)} \delta(p, a)\right)$
- if $w = ax$ for $a \in \Sigma, x \in \Sigma^*$: $\delta^*(q, w) = \varepsilon\text{reach}\left(\bigcup_{p \in \varepsilon\text{-reach}(q)} \left(\bigcup_{r \in \delta^*(p,a)} \delta^*(r, x)\right)\right)$

## Regular closure

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

## Deterministic finite automata

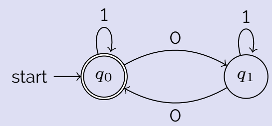
DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.

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*
- $\delta : Q \times \Sigma \to Q$ is the *transition function*
- $s \in Q$ is the *start state*
- $A \subseteq Q$ is the set of *accepting/final* states

Example:




- $Q = \{q_0, q_1\}$
- $\Sigma = \{0, 1\}$
- $\delta$ :

| | 0 | 1 |
|---|---|---|
| $q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_0$ | $q_1$ |

- $s = q_0$
- $A = \{q_0\}$

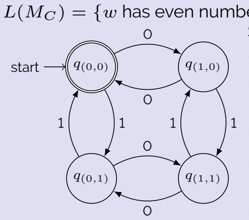
Every string has a unique walk along a DFA. We define the extended transition function as $\delta^* : Q \times \Sigma^* \to Q$ defined inductively as follows:

- $\delta^*(q, w) = q$ if $w = \varepsilon$
- $\delta^*(q, w) = \delta^*(\delta(q, a), x)$ if $w = ax$.

Can create a larger DFA from multiple smaller DFAs. Suppose

- $L(M_0) = \{w \text{ has an even number of 0s}\}$ (pictured above) and
- $L(M_1) = \{w \text{ has an even number of 1s}\}$.

$L(M_C) = \{w \text{ has even number of 0s and 1s}\}$

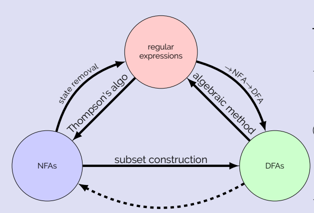



Suppose $M_0 = (Q_0, \Sigma, s_0, A_0, \delta_0)$ and $M_1 = (Q_1, \Sigma, s_1, A_1, \delta_1)$. Then

- $Q = Q_0 \times Q_1 = \{(q_0, q_1) \mid q_0 \in Q_0, q_1 \in Q_1\}$
- $s = (s_0, s_1)$
- $\delta : Q \times \Sigma \to Q$, where $\delta((q_0, q_1), a) = (\delta_0(q_0, a), \delta_1(q_1, a))$
- $A = \{(q_0, q_1) \mid q_0 \in A_0 \text{ and } q_1 \in A_1\}$

## Regular language equivalences

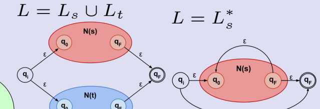
A regular language can be represented by a regular expression, regular grammar, DFA and NFA.



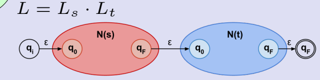

Thompson's algorithm:

$L = L_s \cup L_t$

$L = L_s^*$




$L = L_s \cdot L_t$




**Arden's rule:** If $R = Q + RP$ then $R = QP^*$.

## Fooling sets

Some languages are not regular (Ex. $L = \{0^n 1^n \mid n \geq 0\}$).

Two states $p, q \in Q$ are *distinguishable* if there exists a string $w \in \Sigma^*$, such that

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

or

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

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 $\Sigma$ 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.

# 4 Context-free languages

## 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 $A \to \alpha$ 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:

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

## Pushdown automata

A pushdown automaton is an NFA with a stack.

The language $L = \{0^n 1^n \mid n \geq 0\}$ is recognized by the pushdown automaton:

A *nondeterministic pushdown automaton (PDA)* $P = (Q, \Sigma, \Gamma, \delta, s, A)$ is a **six** tuple where

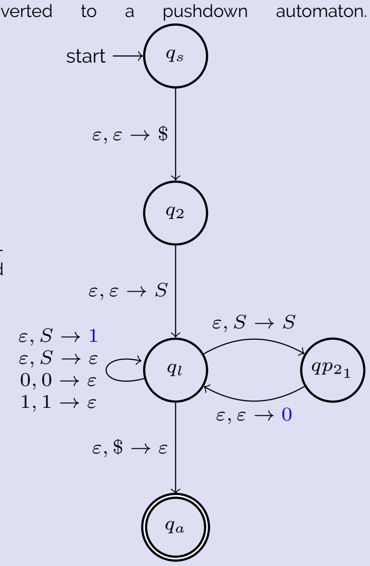
- $Q$ is a finite set whose elements are called *states*
- $\Sigma$ is a finite set called the *input alphabet*
- $\Gamma$ is a finite set called the *stack alphabet*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$ 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 $\langle$input read$\rangle$, $\langle$stack pop$\rangle$ $\to$ $\langle$stack push$\rangle$.

A CFG can be converted to a pushdown automaton.

The PDA to the right recognizes the language described by the following grammar:

$$S \to 0S|1|\varepsilon$$




## Context-free closure

Context-free languages are closed under union, concatenation, and Kleene star.

They are **not** closed under intersection or complement.