ECE-374-B: Algorithms and Models of Computation, Fall 2023

Midterm 1 – September 21, 2023

- 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 6 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 cheatsheet!* 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!*
- Proofs are required only if we specifically ask for them. Even then, none of the questions require long inductive proofs. You are only required to give a short explanation of why your answer is correct.

Name: ______________________________

NetID: ______________________________

Date: ______________________________

1 Short Answer (Regular) (2 parts) - 20 points

Unless the question asks for it, no explanation is required for your answers for full credit. Keep any explanations of your answers to 2 sentences maximum.

a. Write the recursive definition for the following language ($\Sigma = \{\texttt{0}, \texttt{1}\}$):

$$L_{1a} = \{w | w \in \Sigma^*, w \text{ has alternating } \texttt{0}\text{'s and } \texttt{1}\text{'s } \}$$ [1]

b. Write the regular expression for the following languages ($\Sigma = \{\texttt{0}, \texttt{1}\}$):

i $L_{1bi} = \{w | w \in \Sigma^*, w \text{ does not contain the subsequence } \texttt{00}\}$

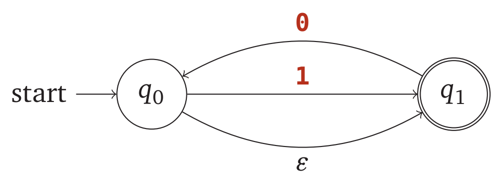
ii $L_{1bii} = \{w | w \in \Sigma^*, w \text{ has alternating } \texttt{0}\text{'s and } \texttt{1}\text{'s } \}$ [2]

[1] ε, “0", and “1" are a part of this language.
[2] See previous footnote. Also note that recursive definitions and regular expressions are separate things.

2 Short Answer II (2 parts) - 20 points

Unless the question asks for it, no explanation is required for your answers for full credit. Keep any explanations of your answers to 2 sentences maximum.

a. Consider the NFA ($\Sigma = \{\mathbf{0}, \mathbf{1}\}$):



Draw the equivalent DFA:

b. Consider the language ($\Sigma = \{\mathbf{0}, \mathbf{1}\}$) represented by the regular expression $r_{2b} = \mathbf{1}^*$. Show that a DFA that represents this language ($L(r_{2b})$) must have at least two states (Hint: think about fooling sets).

3 Language Transformation - 15 points

Assume L is a regular language and $\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Assume zero-indexing (first bit is at position "[0]").

Prove that the language $\textit{FlipEveryThirdChar}(L) := \{\textit{flipThirdChars}(w) \mid w \in L\}$ **is regular.**

$$\textit{flipThirdChars} = \left(\underline{\mathtt{0}}\mathtt{00}\underline{\mathtt{1}}\mathtt{11}\underline{\mathtt{1}}\mathtt{00}\underline{\mathtt{0}}\mathtt{11}\right) = \underline{\mathtt{1}}\mathtt{00}\underline{\mathtt{0}}\mathtt{11}\underline{\mathtt{0}}\mathtt{00}\underline{\mathtt{1}}\mathtt{11}$$

Intuitively, *flipThirdChars* changes every third character in the input alphabet to its bit-wise complement.

4 Language classification I (2 parts) - 15 points

Let $\Sigma_4 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$ and each row of the string represent a binary number.

$$L_4 = \{w \in \Sigma^* | \text{ the top row of } w \text{ is a larger number than is the bottom row.}\}.$$

For the sake of simplicity, you may assume a binary number may (but does not have to) begin with a **0**. As an example, the string "$\begin{bmatrix} 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$" is in the language but the string "$\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$" is not.

a. Is L_4 regular? Indicate whether or not by circling one of the choices below. Either way, prove it.

regular   not regular

b. Is L_4 context-free? Indicate whether or not by circling one of the choices below. Either way, prove it.

context-free   not context-free

5 Language classification II (2 parts) - 15 points

Let $\Sigma_5 = \{\mathtt{0}, \mathtt{1}\}$ and

$$L_5 = \left\{ x y^R | x, y \in \Sigma_5^* \text{ , } |x| = |y| \text{ but } x \neq y \right\}$$

a. Is L_5 regular? Indicate whether or not by circling one of the choices below. Either way, prove it.

regular   not regular

b. Is L_5 context-free? Indicate whether or not by circling one of the choices below. Either way, prove it.

context-free   not context-free

6 Language classification III (2 parts) - 15 points

Let $\Sigma_6 = \{\mathbf{1}\}$ and

$$L_6 = \{xy | x, y \text{ are unary numbers and } x > y\}$$ [3]

a. Is L_6 regular? Indicate whether or not by circling one of the choices below. Either way, prove it.

regular   not regular

b. Is L_6 context-free? Indicate whether or not by circling one of the choices below. Either way, prove it.

context-free   not context-free

[3] Remember from Lab 0, unary numbers (also known as tick marks) are base 1 where a number n is represented by $\mathbf{1}^n$. So numbers $[0, 1, 2, 3, \ldots]_{10} = [\text{“”}, \text{“}\mathbf{1}\text{”}, \text{“}\mathbf{11}\text{”}, \text{“}\mathbf{111}\text{”}, \ldots]_1$ respectively.

This page is for additional scratch work!

ECE 374 B Language Theory: Cheatsheet

1 Languages and strings

Languages

Definitions

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

All languages represent mathematical problems.
Example: multiplication of two integers:

$$L_{MULT2} = \begin{Bmatrix} 1 \times 1|1, & 1 \times 2|2, & 1 \times 3|3, \ldots \\ 2 \times 1|2, & 2 \times 2|4, & 2 \times 3|6, \ldots \\ \vdots & \vdots & \vdots \\ n \times 1|n, & n \times 2|2n, & n \times 3|3n, \ldots \end{Bmatrix} \quad (1)$$

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 $\bar{A} = \Sigma^* \setminus A$.
- Σ^n is the set of all strings of length n.
- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ is the set of all strings over Σ.
- $\Sigma^+ = \cup_{n \geq 1} \Sigma^n$ is the set of non-empty strings over Σ.

Strings

Definitions

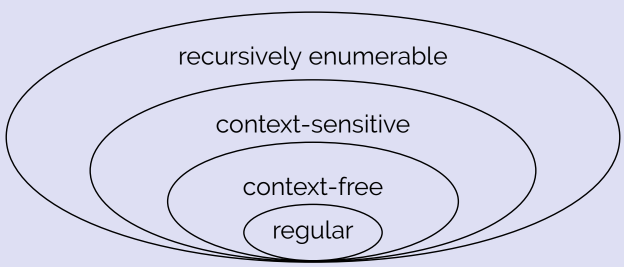
- The *length* of a string w (denoted by $|w|$) is the number of symbols in w.
- For integer $n \geq 0$, Σ^n is set of all strings over Σ of length n. Σ^* is the set of all strings over Σ.
- Σ^* is the set of all strings of all lengths including empty string.
- ε is a *string* containing no symbols.
- $\varnothing$ 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 $w = w_1 w_2 \ldots w_n$ 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: $w^n = \varepsilon$ 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
- α, β, γ - strings in $\{a \cup A\}^*$ where α, β are maybe empty, γ 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 Σ is one of the following:
Base cases:

- $\varnothing$ the language $\varnothing$
- ε 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 $L_1 \cup L_2$
- $\mathbf{r_1} \cdot \mathbf{r_2}$ 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.

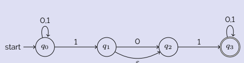
An NFA N *accepts a string* w iff some accepting state is reached by N from the start state on input w.

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*
- Σ 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 Σ are the same as in DFAs

Example:



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

	ε	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 ε-reach(q) is the set of all states that q can reach using only ε-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

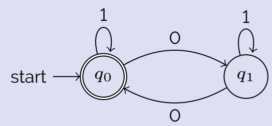
DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.

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*
- Σ 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:



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

	0	1
q_0	q_1	q_0
q_1	q_0	q_1

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

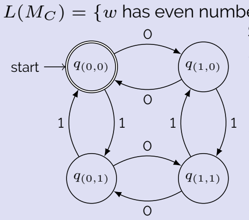
Every string has a unique walk along a DFA. We define the extended transition function as $\delta^* : Q \times \Sigma^* \to Q$ defined inductively as follows:

- $\delta^*(q, w) = q$ 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
- $L(M_1) = \{w \text{ has an even number of 1s}\}$.

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

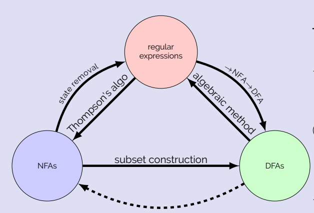


Suppose $M_0 = (Q_0, \Sigma, s_0, A_0, \delta_0)$ and $M_1 = (Q_1, \Sigma, s_1, A_1, \delta_1)$. Then

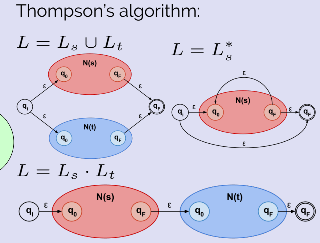
- $Q = Q_0 \times Q_1 = \{(q_0, q_1) \mid q_0 \in Q_0, q_1 \in Q_1\}$
- $s = (s_0, s_1)$
- $\delta : Q \times \Sigma \to Q$, where $\delta((q_0, q_1), a) = (\delta_0(q_0, a), \delta_1(q_1, a))$
- $A = \{(q_0, q_1) \mid q_0 \in A_0 \text{ and } q_1 \in A_1\}$

Regular language equivalences

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



Thompson's algorithm:



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

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

or

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

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 Σ 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 α 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

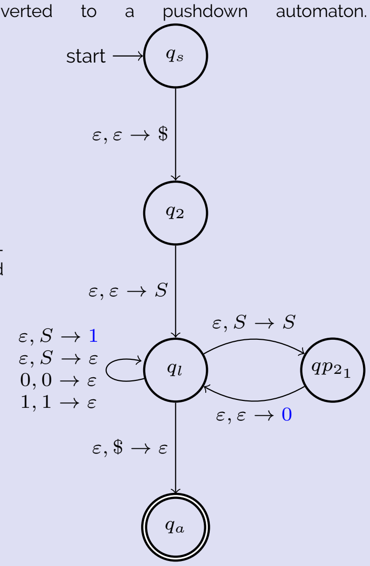
- Q is a finite set whose elements are called *states*
- Σ is a finite set called the *input alphabet*
- Γ is a finite set called the *stack alphabet*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$ is the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

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.