CSC236 UTM **Fall 2024**
Assignment 3 **Due Nov 25**

You are not allowed to post the assignment questions anywhere; however, you are allowed to search the internet (just cite your resources if you find any). You are also allowed to bounce ideas off classmates and TAs, but at the end, you must write your own solutions.

If you use AI tools, please mention the name of the tool and the prompts you used.

Q1. (18 pts)

Show with proof which of the following languages is regular and which is not. If you show that a language is regular, you are expected to provide a DFA that accepts the language.

Let $\Sigma = \{0, 1\}$.

a) **(3 pts)** Σ^*

Proof. From the definition of a regular language, $\{0\}$ and $\{1\}$ are regular languages. Then $\{0\} \cup \{1\} = \Sigma$ is a regular language. Finally, it is true that Σ^* is a regular language.

For this regular language, define the DFA $\mathcal{D} = (\{q_0\}, \Sigma, \delta, q_0, \{q_0\})$, where $\delta : \{q_0\} \times \Sigma \to \{q_0\}$ is a function that always maps to q_0.

□

b) **(3 pts)** $\Sigma^* \backslash \{K\}, K = \{01, 101, 010\}$

c) **(3 pts)** $\{w \mid w \text{ is a palindrome}\}$

d) **(3 pts)** $\{ww \mid w \in \Sigma^*\}$

e) **(3 pts)** $\{w \mid ww \in \Sigma^*\}$

f) **(3 pts)** $\{w \mid w \text{ is a binary representation of a multiple of 3}\}$

Q2. (10 pts)

Implementations of regular expressions often allow you to also have a complement operation. For example, in python [ˆa] means a string that does not have the character 'a' in it.

More formally, the *complement* of L is the language $\overline{L} = \{x \in \Sigma^* \mid x \notin L\}$.

Then in regular expressions, if r is a regular expression matching $\mathcal{L}(r)$, then $\overline{r}$ is a regular expression, with $\mathcal{L}(\overline{r}) = \overline{\mathcal{L}(r)}$.

Prove that regular expressions that also have access to complement can still only express the same class of languages (i.e., the class of regular languages) as regular expressions without the complement operation.

Q3. (15 pts)

Counter-free languages are a subset of regular languages that satisfy the condition: $\exists n \in \mathbb{N}, \forall x, y, z \in \Sigma^*, \forall m \geq n, [xy^m z \in L \iff xy^n z \in L]$

A known result in formal language theory is that counter-free languages are equivalent to the languages that can be expressed by **star-free regular expressions**. **Star-free regular expressions** are regular expressions without the Kleene star, but with complementation.

a) **(5 pts)** Prove that $(ab)^*$ can be matched with a star-free regular expression, $\Sigma = \{a, b\}$

Proof. Define the regex $r = \overline{(b\overline{\varnothing}) + (\overline{\varnothing}bb\overline{\varnothing}) + (\overline{\varnothing}aa\overline{\varnothing})}$. Then

$$\mathcal{L}(r) = \mathcal{L}\left(\overline{(b\overline{\varnothing}) + (\overline{\varnothing}bb\overline{\varnothing}) + (\overline{\varnothing}aa\overline{\varnothing})}\right) = \overline{\mathcal{L}((b\overline{\varnothing}) + (\overline{\varnothing}bb\overline{\varnothing}) + (\overline{\varnothing}aa\overline{\varnothing}))}$$

$$= \overline{\mathcal{L}(b\overline{\varnothing}) \cup \mathcal{L}(\overline{\varnothing}bb\overline{\varnothing}) \cup \mathcal{L}(\overline{\varnothing}aa\overline{\varnothing})} = \overline{\mathcal{L}(b)\overline{\mathcal{L}(\varnothing)} \cup \overline{\mathcal{L}(\varnothing)}\mathcal{L}(bb)\overline{\mathcal{L}(\varnothing)} \cup \overline{\mathcal{L}(\varnothing)}\mathcal{L}(aa)\overline{\mathcal{L}(\varnothing)}}$$
$$= \overline{\{b\}\overline{\varnothing} \cup \overline{\varnothing}\{bb\}\overline{\varnothing} \cup \overline{\varnothing}\{aa\}\overline{\varnothing}} = \overline{\{b\}\Sigma^* \cup \Sigma^*\{bb\}\Sigma^* \cup \Sigma^*\{aa\}\Sigma^*}$$

□

b) **(5 pts)** Prove that $(ab)^*$ is a counter-free language, $\Sigma = \{a, b\}$.

c) **(5 pts)** Prove that $(aa)^*$ is not a counter-free language, $\Sigma = \{a\}$

(Continued on the next page)

Q4. (10 pts + 3 bonus)

The textbook introduces non-determinism on page 78 by highlighting the difference in DFAs and NFAs for language $L = \{w \mid \text{ the third last character of } w \text{ is } 1\}$.

Prove that for any $k \in \mathbb{N}$, the language $L = \{w \mid \text{the } k\text{th to last character of } w \text{ is } 1\}$ has the following attributes:

a) **(5 pts)** A DFA that accepts L has to have at least 2^k number of states.

b) **(5 pts)** The smallest NFA that accepts L has to have exactly k number of states.

c) **(3 pts)** *Bonus*: The smallest DFA that accepts L has to have exactly $2^{k+1} - 1$ number of states.

Q5. (10 pts) Solve question 4 on page 83 in the textbook.

Prove by induction that every finite language can be represented by a regular expression.

Proof. Let $\mathcal{L}$ be a finite language, and let $n \in \mathbb{N}$ represent the number of strings in $\mathcal{L}$. The claim will be proven using simple induction on n.

Base Case. Let $n = 0$. Then $\mathcal{L}$ is the empty set $\varnothing$, which is matched by the regex $\varnothing$.

Let $n = 1$. Then $\mathcal{L}$ contains only one string w. It can be shown using induction on $k = |w|$ that languages with one string can be expressed using some regular expression.

Base Case. Let $k = 0$. Then $w = \varepsilon$, which is matched by the regex ε.

Induction Hypothesis. Suppose that all strings of length m can be represented by a regex. The goal is to show that any string of length $m + 1$ can also be represented by a regex.

Induction Step. Assume that $|w| = m + 1$. Suppose that the last symbol of w is b, for some $b \in \Sigma$. Notice that $w = xb$, where x is a string with m characters. By the induction hypothesis, x is represented by some regex r. As well, b can be represented by the regex b.

Consider the regex rb. It follows that

$$\mathcal{L}(rb) = \mathcal{L}(r)\mathcal{L}(b)$$

which matches only $xb = w$.

By the principle of induction, it is true that all strings can be represented by a regex.

Thus finite languages of size 0 and 1 can be represented by a regex.

Induction Hypothesis. Let $l \in \mathbb{N}$. Suppose that any finite language of size l can be represented by a regex.

Induction Step. Let $\mathcal{L}$ be a language such that $|\mathcal{L}| = l + 1$. Since $l + 1 > 0$, $\mathcal{L}$ is non-empty, so there exists some string $w \in \mathcal{L}$. Consider the language $\mathcal{L} \setminus \{w\}$. The number of elements in this set is l, so by the induction hypothesis, it can be represented as a regex r_1. As well, it was proved in the base case that any language with one string can be represented by a regex. In this case, let r_2 be the regex that represents the language $\{w\}$. It can be verified that the regex $r_1 + r_2$ represents $\mathcal{L}$, since

$$\mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2) = \mathcal{L} \setminus \{w\} \cup \{w\} = \mathcal{L}$$

Thus by the principle of induction all finite languages can be represented as a regex.

$\square$

End of questions