CSC236 Homework Assignment #3

Language Regularity, Regular Expressions, and DFA/NFA Complexity

Alexander He Meng

Prepared for November 25, 2024

Question #1

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

(a):

Claim: Σ^* is a regular language.

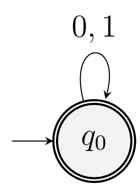
Proof.
Let $L_1 = \{0\}$ and $L_2 = \{1\}$ be regular languages of Σ.
Define $L_3 = L_1 \cup L_2 = \{0, 1\}$ as the regular language obtained by the union of L_1 and L_2.

By definition, ${L_3}^*$ is a regular language. Since $L_3 = \Sigma$, Σ^* is also a regular language.

While this proof is complete, the assignment encourages DFA proofs to show that languages are regular.

So, as an alternative to *Part (a)*, and for all relevant subsequent parts, adopt a DFA proof.

Denote the transition function δ by the following diagram:



Old State	**Symbol**	**New State**
q_0	0	q_0
q_0	1	q_0

Table 1: State Transition Table

Using δ, define the deterministic finite automaton $\mathcal{D} = (\mathcal{Q}, \Sigma, \delta, s, F)$, where

$$Q = \{q_0\} \text{ is the set of states in } \mathcal{D}$$

$$\Sigma = \{0, 1\} \text{ is the alphabet of symbols used by } \mathcal{D}$$

$$\delta : Q \times \Sigma \to Q \text{ is the transition function defined by Table 1}$$

$$s = q_0 \text{ is the initial state of } \mathcal{D}$$

$$F = \{q_0\} \subseteq Q \text{ is the set of accepting states of } \mathcal{D}.$$

For each state q_i, $i \in \{0\}$, in $\mathcal{D}$, define a state invariant $P_q(w)$:

$$P_{q_0}(w) : w \text{ is } \epsilon \text{ or consists of 0s and 1s.}$$

As the only state, q_0 is trivially mutually exclusive. As well, every string in $\Sigma^* = \{0, 1\}^*$ is either ϵ or consists of some number of 0s and 1s, clearly satisfying q_0; exhaustivity is satisfied.

Let $q \in Q = \{q_0\}$ and $w \in \Sigma^* = \{0, 1\}^*$ both be arbitrary (here, $q = q_0$ always).
Denote the predicate:

$$P_{\delta(q_0,w)}(w) \coloneqq P_q(w) \text{ is the state invariant.}$$

Perform structural induction as follows:

<u>Base Case:</u>
Let $q = q_0$ and $w = \epsilon$.
$P_q(w) = P_{q_0}(\epsilon)$ is true as $w = \epsilon$.

<u>Induction Hypothesis:</u>
Assume that $P_q(w)$ is true for all $q \in \{q_0\}$ and some $w \in \{0, 1\}^*$.
This means w is either empty or consists of 0s and 1s.

<u>Induction Step:</u>
Demonstrate that all state invariants hold when processing strings from the $\Sigma = \{0, 1\}$.
This can be achieved by showing that $P_{\delta(q,z)}(wz)$ holds for all $q \in \{q_0\}$ and $z \in \{0, 1\}$.

Let $w \in \{0, 1\}^*$ be arbitrary.

Let $q = q_0$, $z \in \{0, 1\}$, and assume $P_{q_0}(w)$ is true.

Notice that $\delta(q, z) = \delta(q_0, 0) = q_0$. Conveniently, P_{q_0} is already true by assumption.

There are no other state invariants in $\mathcal{D}$. By the principle of structural induction, $P_{\delta(q_0,w)}$ is true for all $q \in Q = \{q_0\}$ and $w \in \Sigma^* = \{0, 1\}^*$.

Finally, demonstrate that $\mathcal{D}$ accepts exactly the language $L = \Sigma^*$ over $\Sigma = \{0, 1\}$.
This is achievable by showing that if $w \in \Sigma^* = \{0, 1\}^*$ is arbitrary, w is a member of L if and only if there exists an accepting state $q \in F$ such that $P_q(w)$ holds.

By the definition of $P_{q_0}(w) : w$ is ϵ or consists of 0s and 1s.
If $w \in L$, then it is either an empty string or consists of 0s and 1s. Clearly, $P_{q_0}(w)$ must be true.
On the other hand, choose the accepting state $q_0 \in F$ and assume $P_{q_0}(w)$ is true. Clearly, $w \in L$, due to matching definitions.

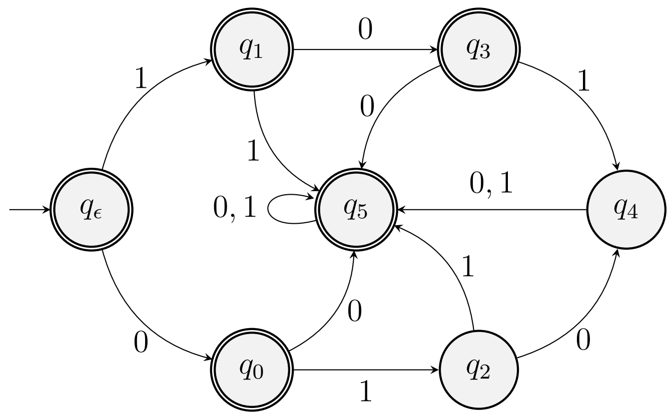
Therefore, while this DFA proof is redundant in nature, it is clear that $\mathcal{D}$ accepts $L = \Sigma^*$ over $\Sigma = \{0, 1\}$.
Again, $L = \Sigma^*$ is demonstrated to be a regular language.

$\square$

(b):
<u>Claim:</u> $\Sigma^* \setminus K, K = \{01, 101, 010\}$ is a regular language.

Proof.
This proof aims to show that the language $\Sigma^* \setminus K$ is a regular language by constructing a DFA that accepts all strings except the literal strings in $K = \{01, 101, 010\}$.

Denote the transition function δ by the following diagram:



Old State	**Symbol**	**New State**
q_ϵ	0	q_0
q_ϵ	1	q_1
q_0	0	q_5
q_0	1	q_2
q_1	0	q_3
q_1	1	q_5
q_2	0	q_4
q_2	1	q_5
q_3	0	q_5
q_3	1	q_4
q_4	0	q_5
q_4	1	q_5
q_5	0	q_5
q_5	1	q_5

Table 2: State Transition Table

Using δ, define the DFA $\mathcal{D} = (\mathcal{Q}, \Sigma, \delta, s, F)$, where

$$\mathcal{Q} = \{q_0\} \text{ is the set of states in } \mathcal{D}$$

$$\Sigma = \{0, 1\} \text{ is the alphabet of symbols used by } \mathcal{D}$$

$$\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q} \text{ is the transition function define by Table 2}$$

$$s = q_0 \text{ is the initial state of } \mathcal{D}$$

$$F = \{q_\epsilon, q_0, q_1, q_3, q_5\} \subseteq \mathcal{Q} \text{ is the set of accepting states of } \mathcal{D}.$$

For each state q_i, $i \in \{\epsilon, 0, 1, 2, 3, 4, 5\}$, in $\mathcal{D}$, define a state invariant $P_q(w)$:

$$P_{q_\epsilon}(w) : w \text{ is the empty string, } \epsilon$$
$$P_{q_0}(w) : w \text{ is } 0$$
$$P_{q_1}(w) : w \text{ is } 1$$
$$P_{q_2}(w) : w \text{ is } 01$$
$$P_{q_3}(w) : w \text{ is } 10$$
$$P_{q_4}(w) : w \text{ is either } 101 \text{ or } 010$$
$$P_{q_5}(w) : w \text{ consists of 0s and 1s but is neither of 01, 010, nor 101}$$

what's below needs to be changed to fit this question As the only state, q_0 is trivially mutually exclusive. As well, every string in $\Sigma^* = \{0,1\}^*$ is either ϵ or consists of some number of 0s and 1s, clearly satisfying q_0; exhaustivity is satisfied.

Let $q \in Q = \{q_0\}$ and $w \in \Sigma^* = \{0,1\}^*$ both be arbitrary (here, $q = q_0$ always).
Denote the predicate:

$$P_{\delta(q_0,w)}(w) \coloneqq P_q(w) \text{ is the state invariant.}$$

Perform structural induction as follows:

<u>Base Case:</u>
Let $q = q_0$ and $w = \epsilon$.
$P_q(w) = P_{q_0}(\epsilon)$ is true as $w = \epsilon$.

<u>Induction Hypothesis:</u>
Assume that $P_q(w)$ is true for all $q \in \{q_0\}$ and some $w \in \{0,1\}^*$.
This means w is either empty or consists of 0s and 1s.

<u>Induction Step:</u>
Demonstrate that all state invariants hold when processing strings from the $\Sigma = \{0,1\}$.
This can be achieved by showing that $P_{\delta(q,z)}(wz)$ holds for all $q \in \{q_0\}$ and $z \in \{0,1\}$.

Let $w \in \{0,1\}^*$ be arbitrary.

Let $q = q_0$, $z \in \{0,1\}$, and assume $P_{q_0}(w)$ is true.
Notice that $\delta(q,z) = \delta(q_0, 0) = q_0$. Conveniently, P_{q_0} is already true by assumption.

There are no other state invariants in $\mathcal{D}$. By the principle of structural induction, $P_{\delta(q_0,w)}$ is true for all $q \in Q = \{q_0\}$ and $w \in \Sigma^* = \{0,1\}^*$.

Finally, demonstrate that $\mathcal{D}$ accepts exactly the language $L = \Sigma^*$ over $\Sigma = \{0,1\}$.
This is achievable by showing that if $w \in \Sigma^* = \{0,1\}^*$ is arbitrary, w is a member of L if and only if there exists an accepting state $q \in F$ such that $P_q(w)$ holds.

By the definition of $P_{q_0}(w) : w$ is ϵ or consists of 0s and 1s.
If $w \in L$, then it is either an empty string or consists of 0s and 1s. Clearly, $P_{q_0}(w)$ must be true.
On the other hand, choose the accepting state $q_0 \in F$ and assume $P_{q_0}(w)$ is true. Clearly, $w \in L$, due to matching definitions.

Therefore, while this DFA proof is redundant in nature, it is clear that $\mathcal{D}$ accepts $L = \Sigma^*$ over $\Sigma = \{0,1\}$.
Again, $L = \Sigma^*$ is demonstrated to be a regular language.

$\square$

(c):
Claim: $\{w|w$ is a palindrome$\}$ is NOT a regular language.

Proof.
proofgoeshere

$\square$

(d):

<u>Claim:</u> $\{ww|w \in \Sigma*\}$ is NOT a regular language.

Proof.
proofgoeshere

□

(e):
<u>Claim:</u> $\{w|ww \in \Sigma*\}$ is a regular language.

Proof.
proofgoeshere

□

(f):
<u>Claim:</u> $\{w|w$ is a binary representation of a multiple of $3\}$ is a regular language.

Proof.
proofgoeshere

□

Question #2

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

Proof.
Suppose r is an arbitrary regular expression with alphabet Σ so that $L = \mathcal{L}(r)$ is a regular language.
Assume r has access to the complement operation.

Let $\overline{L} = \{x \in \Sigma^* \mid x \notin L\}$ be the regular language representing the *complement* of L.
Assume $\overline{r}$ is a regular expression, with $\mathcal{L}(\overline{r}) = \overline{\mathcal{L}(r)}$.

By definition, there exists a DFA $\mathcal{M}$ that accepts L.

Construct $\mathcal{M}$:
$\mathcal{M} = (Q, \Sigma, \delta, s, F)$, where

- Q is a set of finite states;
- Σ is the alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function;
- s is the start state;
- $F \subseteq Q$ is the set of accepting states.

Next, define $\overline{L} = \Sigma^* \setminus L$ as the regular language containing everything obtainable from the alphabet Σ except members of L.

Construct $\overline{M}$ to be identical to M, except for its accepting states:

- Q is a set of finite states;

- Σ is the alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function;
- s is the start state;
- $Q \setminus F \subseteq Q$ is the set of accepting states.

Notice that $\overline{M}$ share the same states, alphabet, transition function, and start state as those of M. The difference is the set of accepting states in $\overline{M}$, which is designed to be mutually exclusive from that of M.

Self-Note: now add the proof that $\overline{M}$ accepts $\overline{L}$, and connect it back to regexes. Next, let $\overline{r}$ be some regular expression without the complement operation.

$\square$

Question #3

Counter-free languages are a subset of languages that satisfy the condition:

$$(\exists n \in \mathbb{N})(\forall x, y, z \in \Sigma^*)(\forall m \geq n)(xy^m z \in L \iff xy^n z \in L).$$

Star-free regular expressions are regular expressions without the Kleene star, but with complementation.

It is known in formal language theory that counter-free languages are equivalent to the languages that can be expressed as **star-free regular expressions**.

(a):

<u>Claim:</u> $(ab)^*$ can be matched with a star-free regular expression, where $\Sigma = \{a, b\}$.

Proof.
The expression $(ab)^*$ represents strings in the set $\{\epsilon, ab, abab, ababab, \dots\}$.

This means matching strings are strings where every occurrence of a is immediately followed by a b.

Due to the definition of $\Sigma = \{a, b\}$, an equivalent star-free regular expression can be written in terms of the complement. The complement definition is highlighted below:

- The string starts with b if it is not empty;
- The string contains an a not immediately followed by a b.

As a regular expression, this is ?.

$\square$

(b):

<u>Claim:</u> $(ab)^*$ is a counter-free language, where $\Sigma = \{a, b\}$.

Proof.
By definition, if $(ab)^*$ is a counter-free language, there exists natural n for all $x, y, z \in \{a, b\}^*$ and for all $m \geq n$ such that $xy^m z \in (ab)^* \iff xy^n z \in (ab)^*$.

Let $n \in \mathbb{N}$. Let $x, y, z \in \{a, b\}^*$ and $m \geq n$ both be arbitrary.

<u>Show that $xy^m z \in (ab)^* \implies xy^n z \in (ab)^*$:</u>
Suppose $xy^m z \in (ab)^*$.

Since $m \geq n$ is arbitrary,

a

<u>Show that $xy^m z \in (ab)^* \impliedby xy^n z \in (ab)^*$:</u>

□

(c):
<u>Claim:</u> $(aa)^*$ is NOT a counter-free language, where $\Sigma = \{a\}$.

Proof.
proofgoeshere

□

Question #4

Let $k \in \mathbb{N}$ be arbitrary. Let $w \in \Sigma^*$, where $|\Sigma| \geq 2$ and has 1 as one of its symbols.

Consider the language $L = \{w | \text{the } k^{\text{th}} \text{ to last character of } w \text{ is } 1\}$.

(a):

Claim: A DFA that accepts L has to have at least 2^k number of states.

Proof.

ACTUALLY, SHOW THIS BY CONTRADICTION: Suppose whatever... less than 2^k states.

A DFA is determinstic and requires states to remember the "history" of the input. For the language $L = \{w | \text{the } k^{\text{th}} \text{ to last character of } w \text{ is } 1\}$, the DFA must track the last k characters of the input string.

Notice that there are 2^k possible combinations of k-length binary substrings, and each of these combinations must map to a unique state in the DFA for accurate processing.

Any DFA with fewer than 2^k states cannot differentiate between all possible k-length suffixes, causing the automaton to classify strings incorrectly.

Any DFA with more than 2^k states either accepts L with a larger alphabet with more than 2 symbols, or is introducing repetitive and redundant states, but still works.

□

(b):

Claim: The smallest NFA that accepts L has to have exactly k number of states.

Proof.

In an NFA, non-determinism allows the automaton to "guess" when it is k-steps away from the end of the string.

The NFA for L needs only k states because: - The start state (initial state) represents the starting position; - The NFA transitions through $k-1$ intermediate states to track progress.

□

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

Proof.
proofgoeshere

□

Question #5

<u>Claim:</u> Every finite language can be represented by a regular expression (meaning all finite languages are regular).

Proof.
Let Σ be an arbitrary alphabet. Let L be an arbitrary finite language over Σ.

Let n be an arbitrary natural number.
Denote the predicate:

$$P(n) := |L_n| = n \implies L_n \text{ can be represented as a regular expression.}$$

This proof uses the principle of simple induction to show $P(n)$ for all $n \in \mathbb{N}$.

<u>Base Cases:</u>
Let $n = 0$.
This means $|L_n| = 0$, so $L_n = \emptyset$. By definition, the empty set is a regular expression.
Thus, $P(0)$.

Let $n = 1$.
Then $|L_n| = 1$, so $L_n = \{w\}$ for some string $w \in \Sigma^*$. By definition, any single string over an alphabet is a regular expression.
Thus, $P(1)$.

<u>Induction Hypothesis</u>
Assume that $P(k)$ holds for some natural k.

This means if L_k has k strings, then L_k can be represented as a regular expression.

<u>Induction Step:</u>
Let $L_{k+1} = \{w_1, w_2, \ldots, w_k, w_{k+1}\}$, where $w_i \in \Sigma^*$ for $i \in [1, k+1] \cap \mathbb{N}$.

By the Induction Hypothesis, $L_k = L_{k+1} \setminus \{w_{k+1}\} = \{w_1, w_2, \ldots, w_k, w_{k+1}\} \setminus \{w_{k+1}\} = \{w_1, w_2, \ldots, w_k\}$ has language has regular expression r_k such that $L_k = \mathcal{L}(r_k)$.

Notice that:

- The regex r_k represents the language L_k;
- The regex w_{k+1} represents the language $\{w_{k+1}\}$.

Then, L_{k+1} can be constructed as a regex as follows:

$$L_{k+1} = L_k \cup \{w_{k+1}\}$$

By definition, the union of two regexes is a regex. Construct r_{k+1}:

$$r_{k+1} = r_k + w_{k+1}$$

As desired, the regex r_{k+1} represents the language L_{k+1}.

<u>Conclusion:</u>
By the principle of simple induction, $P(n)$ holds for all $n \in \mathbb{N}$. It follows that all finite languages must be regular.

$\square$