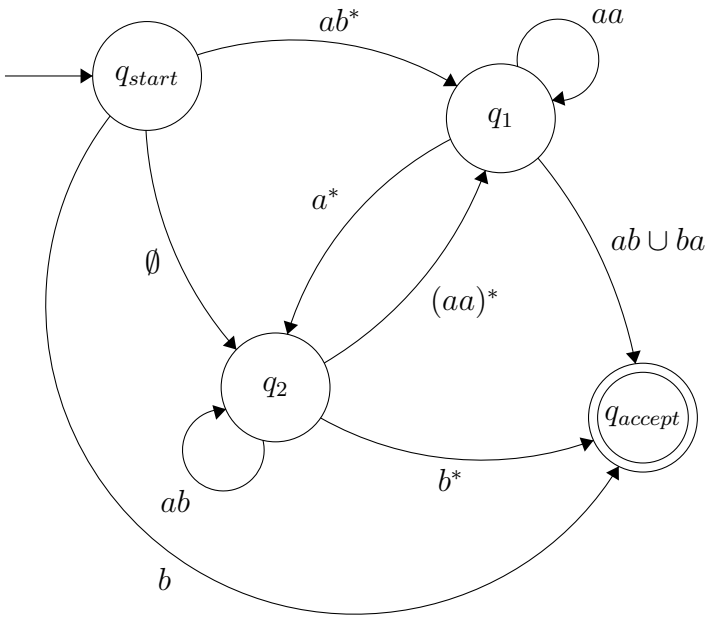
| **CS 181: Formal Languages and Automata Theory** | **Spring 2021** |
|---|---|
| Homework 4 Solution | |
| *Assigned: Tuesday, 20 April* | *Due: Monday, 26 April 9:00pm PDT* |

Note Submission deadline is now 9:00pm.

# Problem 1

Refer to the GNFA diagram in Figure 1.61 on page 70 of the Sipser textbook, presented here for convenience with names given to all four of the states to make it easier to write your answers:




Show two different ways that the GNFA can accept the string "aaab". For each way, list the sequence of states and transitions and show the portion of the input string matched for each transition.

**Solution:**
There are many solutions, here are some examples:
$q_{start} \rightarrow_a q_1 \rightarrow_{aa} q_1 \rightarrow_\epsilon q_2 \rightarrow_b q_{accept}$
$q_{start} \rightarrow_a q_1 \rightarrow_{aa} q_1 \rightarrow_\epsilon q_2 \rightarrow_\epsilon q_1 \rightarrow_\epsilon q_2 \rightarrow_b q_{accept}$
$q_{start} \rightarrow_a q_1 \rightarrow_a q_2 \rightarrow_\epsilon q_1 \rightarrow_{ab} q_{accept}$

# Problem 2

Consider the following Context Free Grammar (CFG), $G_2 = (V, \Sigma, R, S)$, with rule set R given by:

$$S \longrightarrow TT$$
$$T \longrightarrow 0T \mid T0 \mid \#$$

where $S$ is the start variable; the variable set, $V$, is $\{S, T\}$; and the terminal set is $\Sigma = \{0, \#\}$. Describe the language represented by this grammar in clear and simple terms.

**Solution:**

This is the language of 0's and #'s containing exactly 2 #. Or we could also express it as: $L = \{\omega \mid \omega = 0^i\#0^j\#0^k, i, j, k \geq 0\}$. Note that we begin with two of the non-terminal, $T$, and the only way to complete a derivation and get rid of the $T$'s is to replace each one with a #.

# Problem 3

Recall: For any string $\omega = \omega_1\omega_2...\omega_n$, the **reverse** of $\omega$, written as $\omega^R$, is the string $\omega$ in reverse order, $\omega_n...\omega_2\omega_1$; and $\epsilon^R = \epsilon$. For any language, $A$, we have defined $A^R = \{\omega^R | \omega \in A\}$. Show that if $A$ is a FSL (aka regular language), then so is $A^R$.
You may use any approach we have discussed in lecture to show that $A^R$ will always be a FSL. Except of course you may not use the fact that the professor said the family of FSLs was closed under Reversal as your answer! If you use a model, such as a DFA or NFA, you do not need to formally prove that the model is correct. However, as always in this class, your response must include a brief explanation of how your answer correctly represents the solution.

**Two Valid Solutions:**

**Via Constucting a Regular Expression from a Regular Expression:**
Since $A$ is an FSL, there must be a regular expression over $\Sigma$ that represents it. Recall the recursive definition of regular expressions from class. We will recursively define the reverse of each component, according to the recursive definition. This will show us how to take any regular expression and construct one representing the reverse language. Suppose $R$ is a regular expression, then we construct a new regular expression to represent the reverse $R^R$ as follows. If $R$ is:

1. $a$, then $R^R = a$
2. $\epsilon$ then $R^R = \epsilon$
3. $\emptyset$ then $R^R = \emptyset$

4. $R_1 \cup R_2$ then $R^R = (R_1^R \cup R_2^R)$ for $R_1, R_2$ regular expressions

5. $R_1 \circ R_2$ then $R^R = (R_2^R \circ R_1^R)$ for $R_1, R_2$ regular expressions

6. $R_1^*$ then $R_R = (R_1^R)^*$ for $R_1$ regular expression.

Brief Justification: We can see that 1 and 2 are clearly true by simply reversing a single symbol or empty string and seeing that it stays the same. 3 is true since there is nothing to reverse in an empty set. To get the reverse of a union of regular expressions, we must match one of the regular expressions (but in reverse), so 4 holds as well. We proved a very similar result to 5 on Homework 1. Finally, since Kleene star is an arbitrary number of concatenations of the same regular expression (and hence we don't care about the order of concatenation), 6 holds as well.
Thus given a regular expression for a language, we can construct a regular expression for the reverse of the language. This means that FSLs are closed under reversal.

**Via Constructing an NFA from a DFA:**
If a language $L$ is regular, then, there must a DFA $M$ that accepts the language $L$. Next, we can construct a Finite Automaton $M^R$, based on the machine $M$, that accepts the reversed language $L^R$.

We can construct the $M^R$ by modifying $M$

1. Reverse the direction of all the transitions in $M$

2. Change the original accepting states in $M$ to non-accepting states, and change the start state in $M$ to be the only accepting state. (Note: This means that if the initial state of $M$ was accepting, then it will still be accepting in $M^R$).

3. Add a new start state, and add $\epsilon$-transitions from the new start state to the original accepting states in $M$. There are no transitions into the new start state.

Brief Justification: If a string is accepted by M, then there is a path from the start state to an accepting state; so in $M^R$ there will be a corresponding path in the reverse direction leading to the one accepting state. And if there is no accepting path in $M$, then this construction will not create a corresponding erroneously accepting path in $M^R$. Thereby, we can construct a NFA $M^R$ that accepts the reversed language $L^R$, if the original language could be recognized by a DFA $M$. This implies that for all FSLs, $L$, the reversed language $L^R$ is also a FSL.

# Problem 4

Let $\Sigma = \{a, b, \#\}$. Consider following language:

$$L_5 = \{\ w = x\#y\#z \mid x, y, z \in \{a, b\}^* \text{ and } z = x^R \text{ or } |y| = 2|x|\ \}$$

Design a Context Free Grammar that generates this language. Be sure to clearly indicate your variable set, start variable, and rule set. And as always, include a *brief* description of

how your grammar is designed to correctly represent the language.

**Soution:**
Let's first work on the condition $z = x^R$

$$\begin{aligned} S_1 &\longrightarrow A \mid B \\ A &\longrightarrow aAa \mid B \mid C \\ B &\longrightarrow bAb \mid A \mid C \\ C &\longrightarrow \#R\# \\ R &\longrightarrow aR \mid bR \mid \epsilon \end{aligned}$$

Then we work on the condition $|y| = 2|x|$

$$\begin{aligned} S_2 &\longrightarrow D\#R \\ D &\longrightarrow aDaa \mid aDab \mid aDba \mid aDbb \\ &\quad \mid bDaa \mid bDab \mid bDba \mid bDbb \mid \# \end{aligned}$$

Finally, we drop the $S_1$ and $S_2$ rules (which are no longer needed), and we combine the two sets of remaining rules together by adding a new start variable, $S$ and three new rules:

$$S \longrightarrow A \mid B \mid D\#R$$

Brief justification: Since, the definition of the language is the "or" of the two conditions, the language is the union of two languages, one for each condition.
The first rule set uses the rules for variables $A$ and $B$ in the recursion design pattern to generate matching pairs of $a$'s and $b$'s, respectively. Then it generates an arbitrary string between two #'s.
The second rule set uses the sequencing design pattern to generate strings matching the second condition followed by a single # followed by an arbitrary string. The variable, $D$, generates two of any symbol on the right for any single symbol on the left, thereby ensuring the second condition.

# Problem 5

Use the Pumping Lemma for the FSLs (and possibly other results) to show that the following language over alphabet $\Sigma = \{0, 1\}$ is not a FSL:

$$L_5 = \{\ w \mid w = xy,\ x, y \in \Sigma^*,\ |x| = |y|, \text{ and } \#(0, x) = \#(0, y)\ \}$$

**Solution:**

**Proof:**
Proof by contradiction, suppose $L_5$ is regular, and there exists a pumping length $p > 0$.

Take $w = 0^p1^p0^p1^p$. Then we know $w \in L_5$ and $|w| > p$.

Then, from the pumping lemma, we know that there exists $x, y, z \in \Sigma^*$ such that $w = xyz$, $|xy| \leq p$, $|y| \geq 1$, and $\forall i\ xy^iz \in L_5$

Since the first $p$ symbols of w are 0 and $|xy| \leq p$, we can write $x = 0^m$, $y = 0^n$ where $m + n \leq p$ and $n \geq 1$.

By pumping lemma, we know that $\forall i\ xy^iz \in L_5$. Here, we take $i = 2$, then we have

$$xy^2z = 0^{p+n}1^p0^p1^p$$

Since $n \leq m + n \leq p$, we know that $p + n \leq 2p$. Thereby, the number of 0's in the first half of $xy^2z$ is $p + n$, whereas the number of 0's in the second half of $xy^2z$ is $p$. Thereby, $xy^2z \notin L_5$. Here comes to a contradiction, and thus $L_5$ is not a regular language.
■