CS 181: Formal Languages and Automata Theory	**Spring 2021**
Homework 3 Solution	
Assigned: Tue 4/13	*Due: Mon 4/19, 9:00pm PDT*

Note new time: submission deadline has been extended from 6:00 to 9:00pm PDT.

Problem 1

Carry-over from last week:
Let $\Sigma = \{a, b\}$. Use the Pumping Lemma for the FSLs to show that the following language over Σ is not a FSL:

$$L = \{a^{(2n)}b^n \mid n \geq 0\}$$

Solution:
Assume L is regular, then let $\omega = a^{2p}b^p$, and note that ω is in the language. Per the lemma, we can let $\omega = xyz$,
From pumping lemma:
$|xy| \leq p$ (condition 3)
$|y| \geq 1$ (condition 2)
Then y contains at least 1 a and only a's, so $y = a^t, t \geq 1$
Then we "pump down" the substring y by choosing $i = 0$ per the lemma.
Let $\omega' = xy^0z = a^{2p-t}b^p$.
To satisfy condition 1, we will need $\omega' = a^{2p-t}b^p \in L$. However, we have reduced the number of a's by t without changing the number of b's. So the number of a's will be fewer than 2 times of the number of b's, thus ω' is not in the language.
Therefore, condition 1 is not satisfied. So L is not a regular language.

Problem 2

Let $\Sigma = \{a, b, c\}$. Show a regular expression which recognizes the following language over Σ:

$$L_2 = \left\{ w \in \Sigma^+ \;\middle|\; |w| > 1, \text{ and the first and last symbols of } w \text{ are different} \right\}$$

E.g.,

- L_2 contains: $ab, ca, abcb$.
- L_2 does not contain: $\epsilon, c, bb, abcba$.

Briefly describe how your regular expression is designed to correctly represent the language.

Solution:

$$(a\Sigma^*(b \cup c)) \cup (b\Sigma^*(a \cup c)) \cup (c\Sigma^*(a \cup b))$$

Brief Explanation: We consider the 3 cases separately: a string starting with either a, b, or c. For each of them, there can be any number of symbols of any sort (hence Σ^*), followed by a single differing character.

Problem 3

Let $\Sigma = \{a, b, c\}$. Show an NFA which recognizes the following language over Σ:

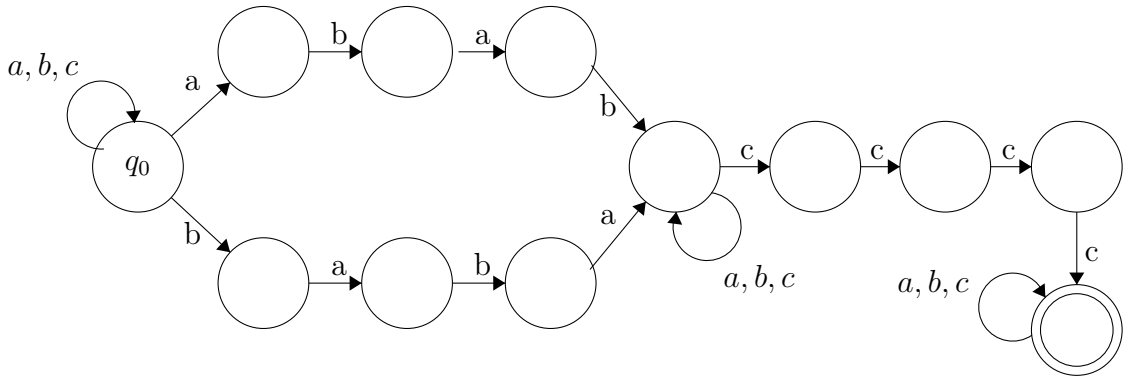
$$L_3 = \{w \in \Sigma^* \mid w \text{ contains } abab \text{ or } baba \text{ (or both) and after one of those substrings there is substring } cccc\}$$

Example: abbababbbccccabaa

Specify the NFA as a state diagram. It does not have to be fully specified; but if you use the blocking convention, you must use it correctly. *Be sure to clearly indicate your initial state and accepting state(s).* Part of your score will be based on demonstrating effective use of nondeterminism in the NFA model. You do not need to find an NFA with the minimal number of states, but part of your score will be based on avoiding unnecessary complexity in your NFA.

Briefly describe how your design works. Part of your score will also be based on the clarity of your description and how well the state diagram and the description match each other.

Solution:



Brief explanation: Using a guess and check approach to non-determinism, we begin with two branches, one for finding abab, and one for baba. Once this is completed, we again have a guess and check self-loop, and check for a string of 4 straight c's.

Problem 4

Let $\Sigma = \{v, +, -, (,)\}$. Let $e \in \Sigma^*$ be the string $v - v - v$. Let G_4 be the $CFG(V, \Sigma, R, E)$, with variable set $V = \{E, O\}$ and rule set R given by:

$$\begin{aligned} E &\longrightarrow v \mid EOE \mid (E) \\ O &\longrightarrow + \mid - \end{aligned}$$

a. Show a parse tree for e in G_4.

b. Show the corresponding left-most derivation in G_4.

c. Show a *different* parse tree for e in G_4.

d. Show the corresponding left-most derivation in G_4.

Solution:

a. Figure 1 shows a parse tree for e in G_4.

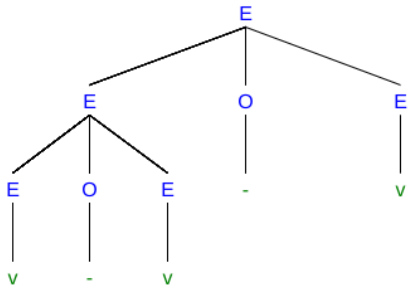


Figure 1: Problem 4(a) parse tree for e in G_4

b. The corresponding left-most derivation of Figure 1 is as follows:

$$\underline{E} \Rightarrow \underline{E}OE \Rightarrow \underline{E}OEOE \Rightarrow v\underline{O}EOE \Rightarrow v - \underline{E}OE \Rightarrow v - v\underline{O}E \Rightarrow v - v - \underline{E} \Rightarrow v - v - v$$

c. Figure 2 shows a parse tree for e in G_4.

d. The corresponding left-most derivation of Figure 2 is as follows:

$$\underline{E} \Rightarrow \underline{E}OE \Rightarrow v\underline{O}E \Rightarrow v - \underline{E} \Rightarrow v - \underline{E}OE \Rightarrow v - v\underline{O}E \Rightarrow v - v - \underline{E} \Rightarrow v - v - v$$

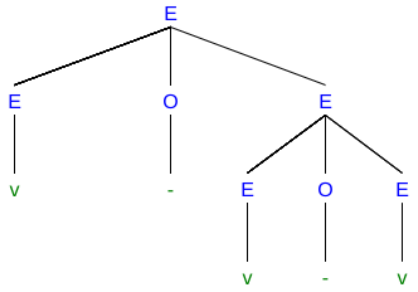


Figure 2: Problem 4(c) a different parse tree for e in G_4

Problem 5

For this problem, it may be helpful to think of the symbols "`b`", "`e`", and "`s`" as representing "begin", "end", and "statement", respectively.

Let $\Sigma = \{\texttt{b}, \texttt{e}, \texttt{s}, \texttt{;}\}$. Show a context free grammar (CFG) over Σ for the language, L_5, of "begin-end blocks with the semicolon used as a statement terminator". I.e., every "statement" must be followed by a "`;`", including the outer-most "`b e`" pair. E.g.:

- L_5 contains: `bs;e;` & `bs;s;e;` & `bs;bs;e;e;` & `bbs;e;e;`
- L_5 does not contain: `s;` & `bs;e` & `bs;se;` & `bs;` & `bs;;` & `bs;s;e;e;` & `bs;bs;e;` & `be;` & `be` & `b;e` & `es;b;` & `bs;e;bs;e;`

Solution:
$G = (V, \Sigma, R, P)$, with $R =$ the rule set below.
$V = \{P, B, L, I\}$

$$P \longrightarrow B;$$
$$B \longrightarrow bLe$$
$$L \longrightarrow I|LI$$
$$I \longrightarrow s;|B;$$

Brief explanation: The variable names P, B, L, and I are intended to represent Program, Block, List, and Item, respectively. The B rule generates a be block, and each block is a List surrounded by b and e. The L rules generate lists of Items. An Item will be an s or a be block. Each Item comes with its own ";" to ensure that the ";" works as a terminator here. The one P rule helps ensure that we do not generate incorrect string like "bs;e;bs;e;" with two adjacent b-e pairs at the top level when only one is allowed.

Problem 6

For the this problem, it may be helpful to think of the symbols "`b`", "`e`", and "`s`" as representing "begin", "end", and "statement", respectively.

Let $\Sigma = \{\texttt{b}, \texttt{e}, \texttt{s}, \texttt{,}\}$. Show a context free grammar (CFG) over Σ for the language, L_6, of "begin-end blocks with the comma used as a statement separator". I.e., two consecutive "statements" in a "`b e`" pair must have a "`,`" between them. E.g.:

- L_6 contains: `bse` & `bs,se` & `bs,bs,see` & `bbs,see`
- L_6 does not contain: `s` & `bs,s,e` & `bs,s` & `b,se` & `be` & `bs,se,` & `bs,see` & `bs,bse` & `b,e` & `es,sb` & `bse,bse`

Solution
$G = (V, \Sigma, R, P)$, with $R =$ the rule set below.
$V = \{P, B, L, I\}$

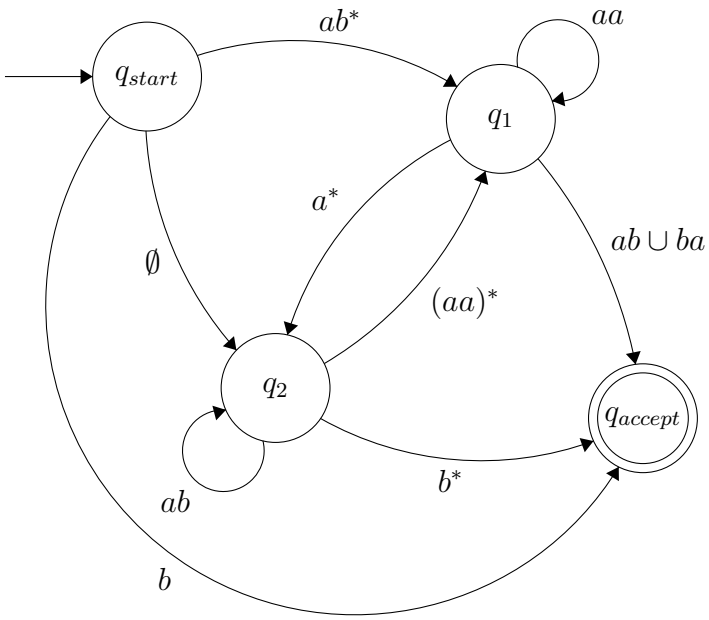
$$
\begin{aligned}
P &\longrightarrow B \\
B &\longrightarrow bLe \\
L &\longrightarrow I \mid L, I \\
I &\longrightarrow s \mid B
\end{aligned}
$$

Brief explanation: The variable names P, B, L, and I are as above. The L rules generate lists of Items with a "," between consecutive items. Note the difference in the L rules compared to the solution for the previous problem. By changing the way we generate "," vs. ";", we are able to ensure that the "," works as a separator here instead of as a terminator. The one P rule is not really needed here, but we left it in to show the similarities and differences between the two solutions.

Problem 7

Postponed to Week 4

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.