CS164 Programming Languages and Compilers **Spring 2024**

Written Assignment 1 Solutions

Assigned: January 25 **Due:** February 8 at 3:00 pm

1. Consider the following languages of binary numbers over the alphabet $\Sigma = \{0, 1\}$.

   - L_1: *All binary numbers where the last digit is a 1* (e.g. $1, 011, 111, 1011 \in L_1$)
   - L_2: *All binary numbers divisible by 4*
   - L_3: *All binary numbers divisible by 3*
   - L_4: *All binary numbers that contain exactly 2 0's or no 1's*

   Give a deterministic finite automaton (DFA) for all the languages above. (Note: Empty strings are not binary numbers.)

   Solution:

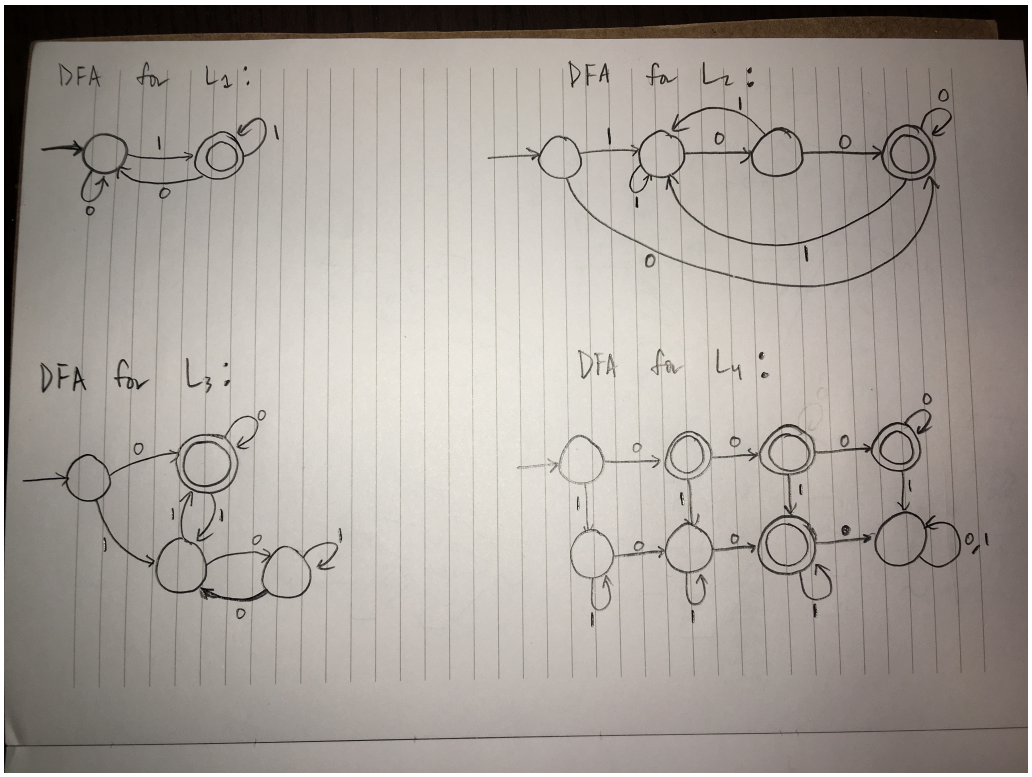


Figure 1: NFA

2. Consider the regular expression $R = (ab)^* \mid (bb \mid aba)^*$, note that language $L(R)$ is over the alphabet $\Sigma = \{a, b\}$

   - Construct an ϵ-NFA for the language $L(R)$.
   - Convert the above NFA to DFA.

   (**Hint:** use approach describe in the lecture : ϵ-NFA $\rightarrow$ NFA (label states) $\rightarrow$ DFA).

   Solution:

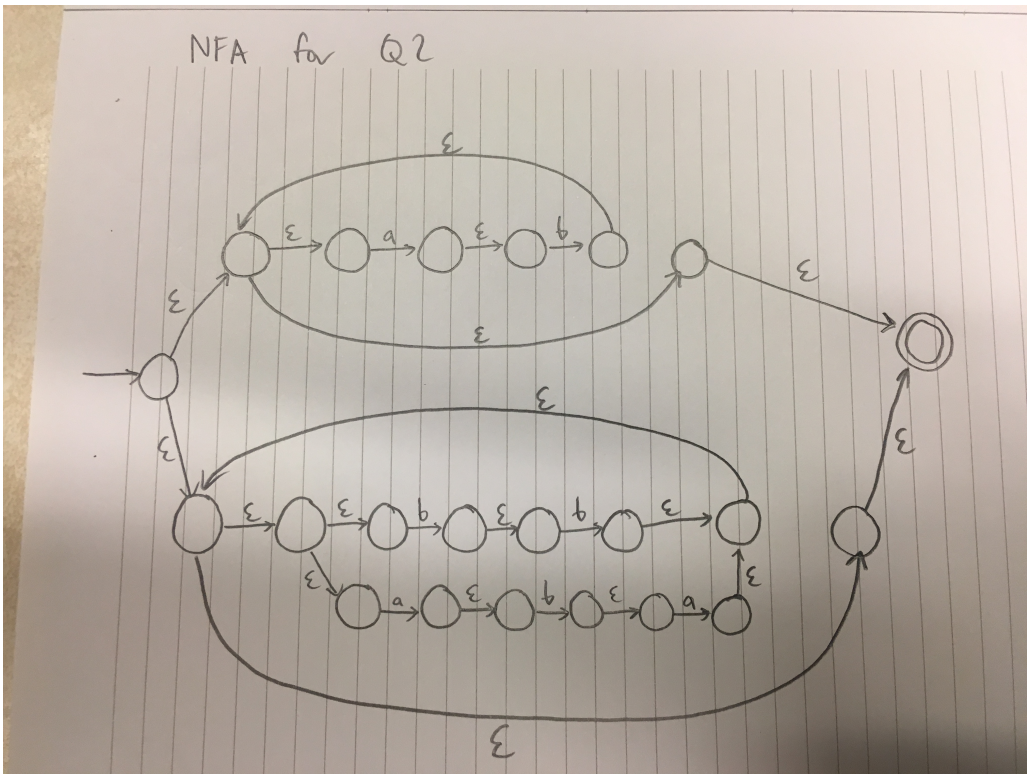


Figure 2: NFA

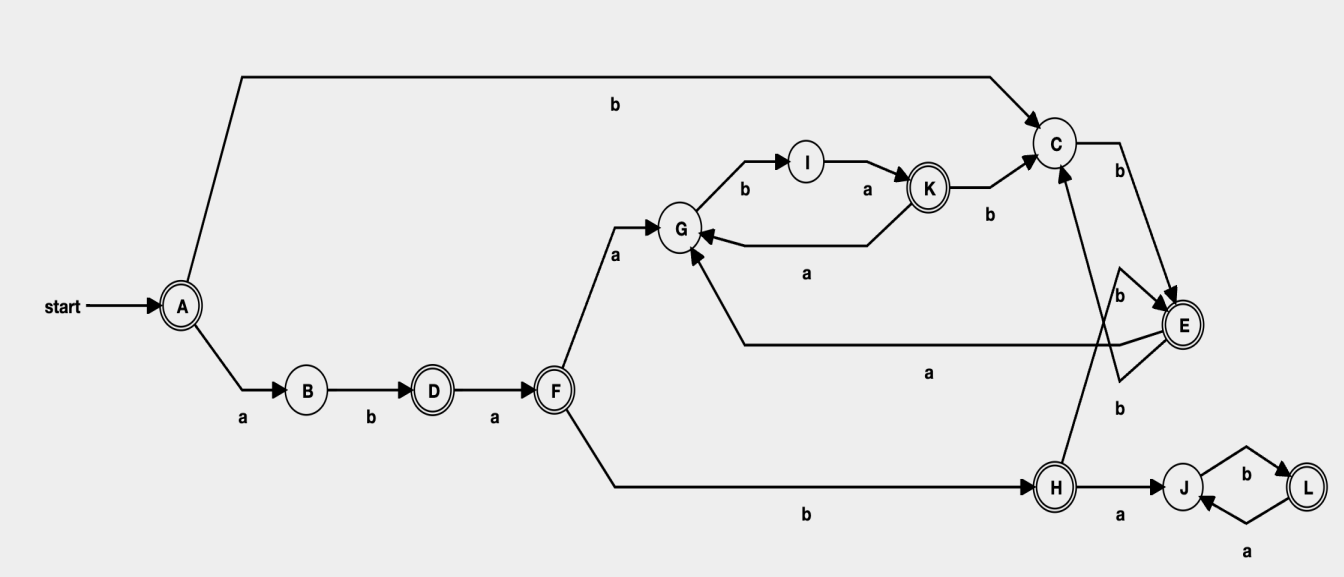


Figure 3: DFA

3. Let $\Sigma_m = \{a_1, \ldots, a_m\}$ be an alphabet containing m elements, for some integer $m \geq 1$. Let L_m be the following language that includes all strings in which at least one of the characters occurs an even number of times, i.e.

   All strings in which a_i occurs an even number of times for some i, where $1 \leq i \leq m$

   Construct a DFA for the language L_3. Also construct an NFA for the language L_4.

   Solution:

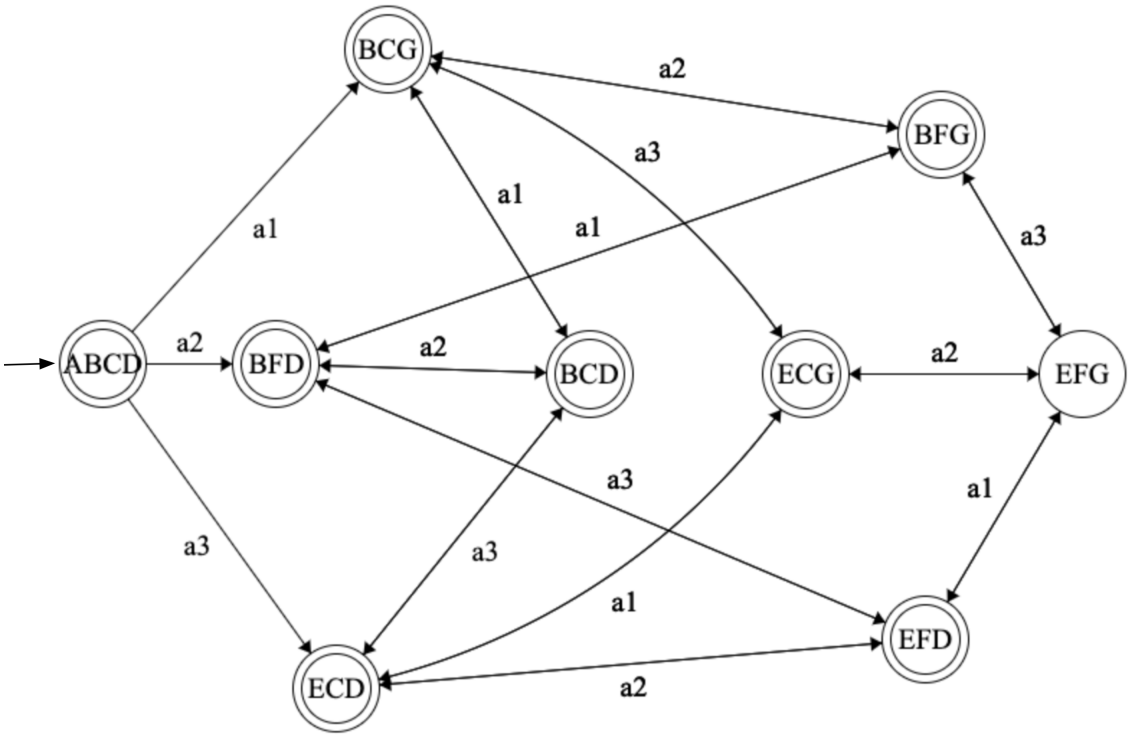


Figure 4: L_3 DFA

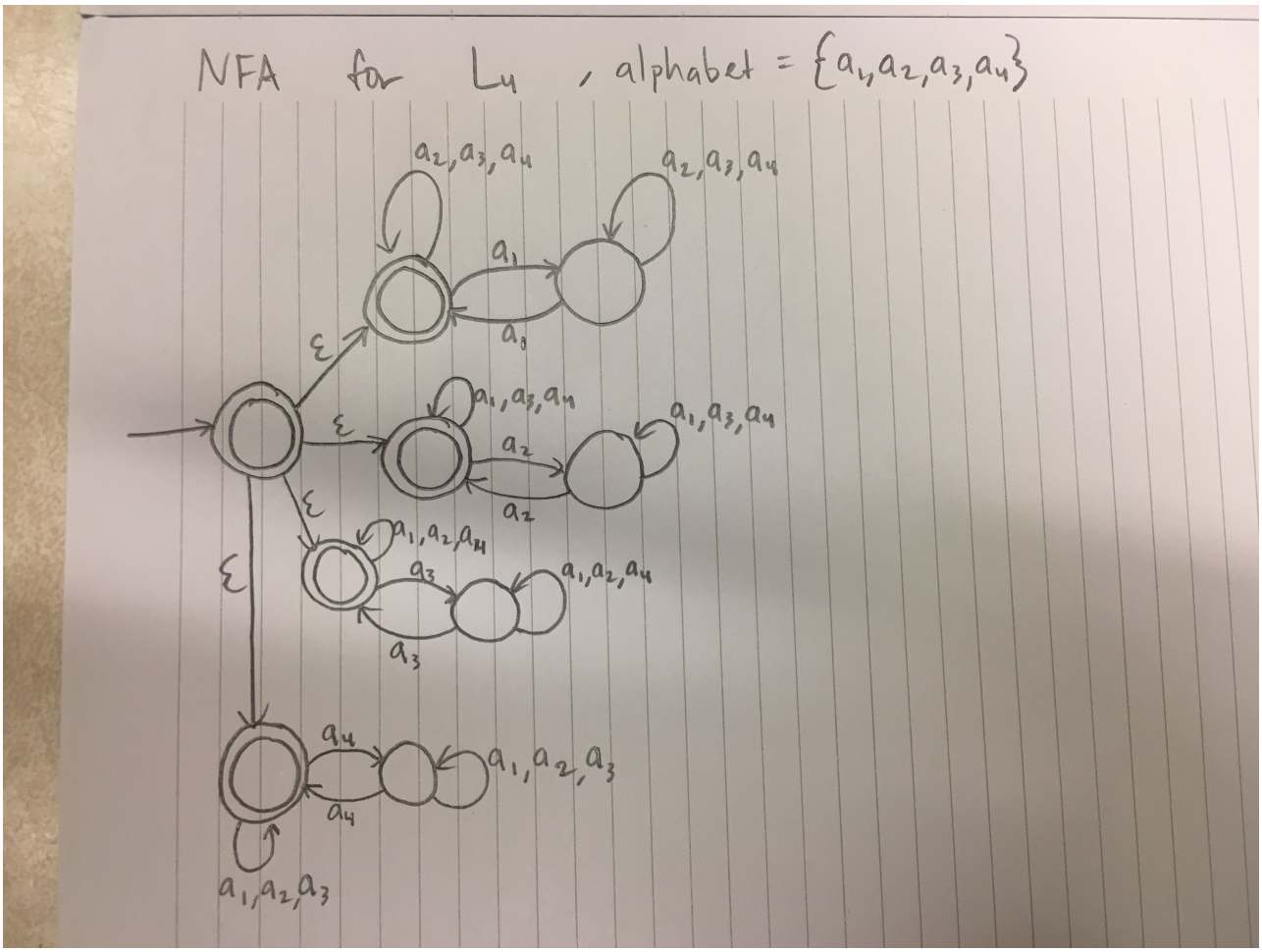


Figure 5: L_4 NFA

4. Determine whether or not the following languages are regular. Explain why in one or two sentences.

   - L_1: *All strings over the alphabet* $\{a, b\}$ *where there are at least as many* a*'s as there are* b*'s.*
   - L_2: *All strings over the alphabet* $\{a, b\}$ *that are palindromes (same string when reversed).*
   - L_3: *All words in the Oxford English dictionary.* (**Hint:** assume dictionary has finite number of words).

   Solution:

   - L_1: It is not regular since for any string the DFA needs to record the the number of a's and b's that it has seen so far. Every DFA has a finite number of states; hence for every DFA, there will exist large enough strings for which the DFA cannot keep track of the number of a's and b's.
   - L_2: It is not regular since for any string the DFA needs to be able to record part of the string to see if the later parts repeat it in a reversed form. This cannot be done with a finite number of states because one cannot know how long the sequence that may be reversed later is.
   - L_3: It is regular since it contains finitely many strings.

5. Let $\Sigma = \{a, b\}$ be the alphabet for the language $L = \{waw^R \mid w, w^R \in \{a, b\}^*, \text{ and } w \text{ has even length}\}$, where w^R is the reverse of w.

   Write a context free grammar for the language L.

   Solution: $S \to a \mid aaSaa \mid abSba \mid baSab \mid bbSbb$.

6. Consider the following grammar:

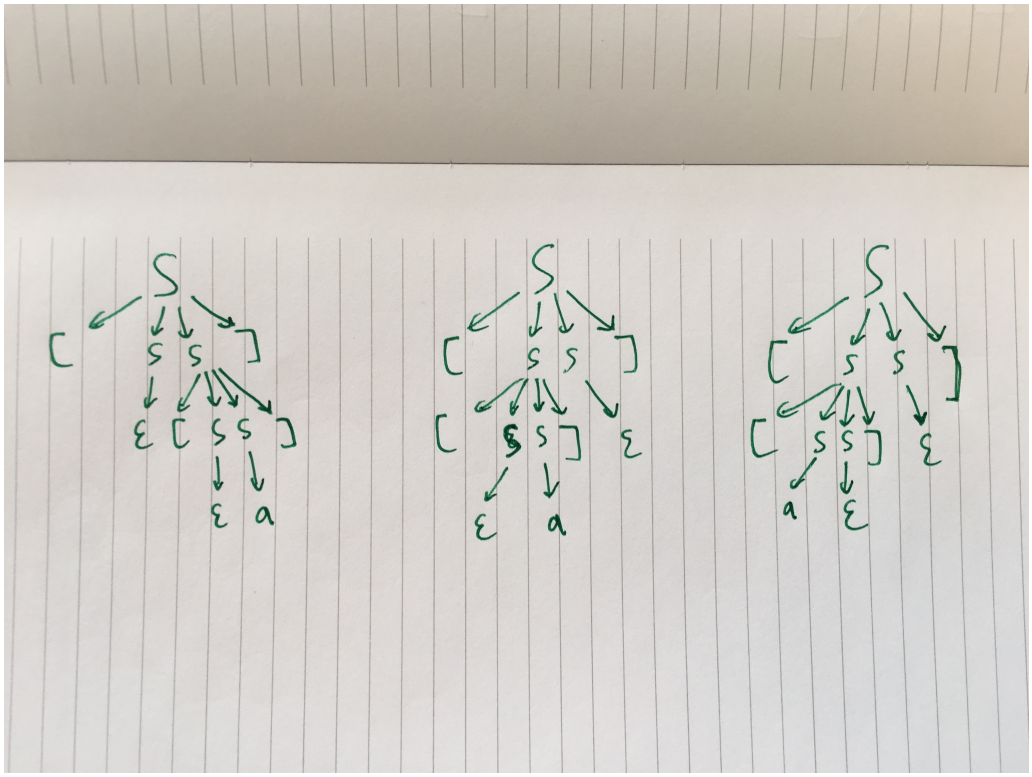
$$S \to [\ S\ S\ ]$$
$$S \to a$$
$$S \to \varepsilon$$

Show that this grammar is ambiguous by finding a string that can be parsed in at least three different ways. Draw three different parse trees for this string, and write down the left-most derivation for each of the three trees.

Solution: One option of an ambiguously parsed string is $[[a]]$. Three different left-most derivations are:

$$S \to [SS] \to [\varepsilon S] = [S] \to [[SS]] \to [[\varepsilon S]] = [[S]] \to [[a]]$$
$$S \to [SS] \to [[SS]S] \to [[\varepsilon S]S] = [[S]S] \to [[a]S] \to [[a]\varepsilon] = [[a]]$$
$$S \to [SS] \to [[SS]S] \to [[aS]S] \to [[a\varepsilon]S] = [[a]S] \to [[a]\varepsilon] = [[a]]$$

Here are the corresponding parse trees:



7. Give context free grammars for the following languages. Your grammars should not be unnecessarily complex. For each grammar, briefly explain why your grammar accepts precisely the specified language.

   (a) $L = \{x^i y^j : 0 \leq j \leq i\}$, where for example $x^5y^2 = xxxxxyy$.
   Solution:

$$S \to \epsilon \mid xS \mid xSy$$

   We start with the empty string, then repeatedly prepend an x or prepend an x and append a y, so that there are never more y's added than x's.

   (b) $L = \{a^i b^j c^k : i \geq 0,\ j \geq 0,\ \text{and } i + j \leq k\}$.
   Solution:

$$S \to \epsilon \mid aSc \mid B$$
$$B \to \epsilon \mid Bc \mid bBc \mid \epsilon$$

   There is only one place in the grammar where we can prepend a to the string, and notice that it always appends c to the string as well. Additionally, there is only one place where b can be added to the string, and again c is always appended here as well. However, B also has the rule Bc, which allows for additional c's to be appended. Therefore, $i + j \leq k$.

   (c) $L = \{w^R \# w \mid w \in \{0,1\}^*$ and w as a binary number is divisible by $3\}$. For this problem, if w is empty then its value as a binary number is 0.
   Solution:

$$S \to S_0$$
$$S_0 \to \# \mid 0S_00 \mid 1S_11$$
$$S_1 \to 0S_20 \mid 1S_01$$
$$S_2 \to 0S_10 \mid 1S_21$$

   Here S_i matches any $w^R \# w$ where w is a (maybe even empty) binary string equivalent to i mod 3. For a production $S_i \to nS_jn$, we just require that $i \equiv 2j + n \pmod 3$.