Computer Science 181, Homework 4

Michael Wu
UID: 404751542

April 30th, 2018

Postponed Problem 5

Assume for contradiction that this language

$$L = \{0^i 1^j 0^k \mid i, j, k \geq 0 \wedge k = |i - j|\}$$

is finite state. Then by the pumping lemma if a language is finite state then there exists some p such that for any string s in L with $|s| > p$, s can be split into three strings x, y, and z where $s = xyz$, $|y| \geq 1$, $|xy| \leq p$, and $xy^*z \subseteq L$. Take the string $0^p 1^p \in L$, and note that xy must be a string made entirely of 0's, since $|xy| \leq p$. Additionally, y must be in 0^+, since $|y| \geq 1$. Let $x = 0^a$ and $y = 0^b$ for some constants $a \geq 0$ and $b > 0$. Then z must be $0^{p-a-b}1^p$. By the pumping lemma $xy^*z \subseteq L$, so

$$\{0^a(0^b)^* 0^{p-a-b} 1^p \mid a \geq 0 \wedge b > 0\} \subseteq L$$

This can be stated equivalently as

$$\{0^{p-b} 0^{xb} 1^p \mid \forall x \in \{0, 1, 2, \ldots\} \wedge b > 0\} \subseteq L$$

When $x = 2$, this means that the string $0^{p+b}1^p \in L$ for some constant $b > 0$. But in the definition of L, this string has the parameters $i = p + b$, $j = p$, and $k = 0$. Thus $k \neq |i - j|$ and $0^{p+b}1^p \notin L$. This is a contradiction, so L cannot be finite state.

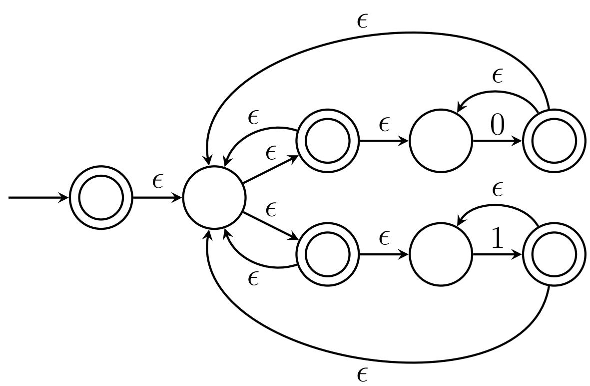
Problem 1

a) Always. The intersection of a finite language and any other language must be a finite language, and thus $L_{\text{f}} \cap L_{\text{nfs}}$ is finite state.

b) Sometimes. $\{\} \cup L_=$ is not finite state but $\Sigma^* \cup L_=$ is finite state.

c) Never. Non finite state languages are infinite languages, so there are an infinite number of strings in L_{nfs} that cannot be expressed through a finite state machine. Because L_{f} is finite, there aren't enough strings in L_{f} to simplify L_{nfs} into a finite state language. For example, consider the finite language L_x consisting of all the strings with length smaller than x. $L_x \cup L_{\text{nfs}}$ would still be a non finite state language, as there would be no finite state machine to express the strings in $L_x \cup L_{\text{nfs}}$ that have a length greater than x, since these are strings that are exclusively from L_{nfs}. For any finite language L_{f} a similar argument could apply, as there must be a maximum length string in L_{f}. After this maximum length, all the strings in $L_x \cup L_{\text{nfs}}$ would come from L_{nfs} and could not be expressed through a finite state machine.

Problem 2



Problem 3

We can write a regular expression for this language L_3, which shows that it is finite state. Let x be a string in the set

$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

and R be some constant. Then the regular expression

$$E_x = (\Sigma^* x \Sigma^* x^R \Sigma^*) \cup (\Sigma^* x^R \Sigma^* x \Sigma^*)$$

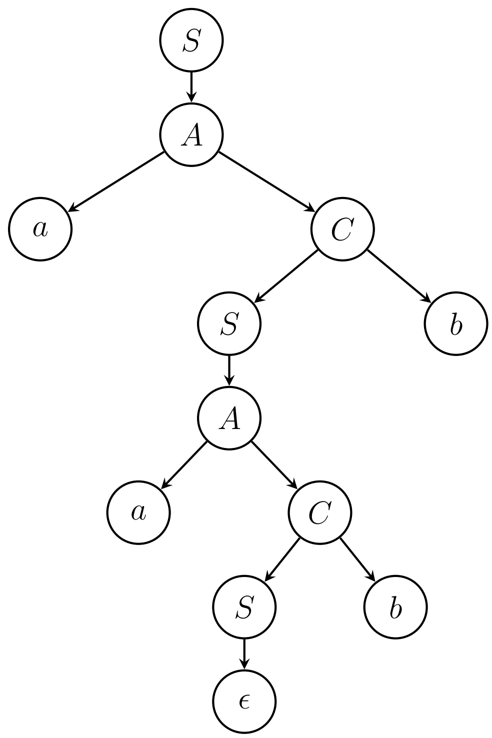
denotes a string in L_3 that contains a given x and x^R. Then we have that

$$L_3 = E_{aaa} \cup E_{aab} \cup E_{aba} \cup E_{abb} \cup E_{baa} \cup E_{bab} \cup E_{bba} \cup E_{bbb}$$

which is a regular expression that proves that L_3 is a finite state language.

Problem 4

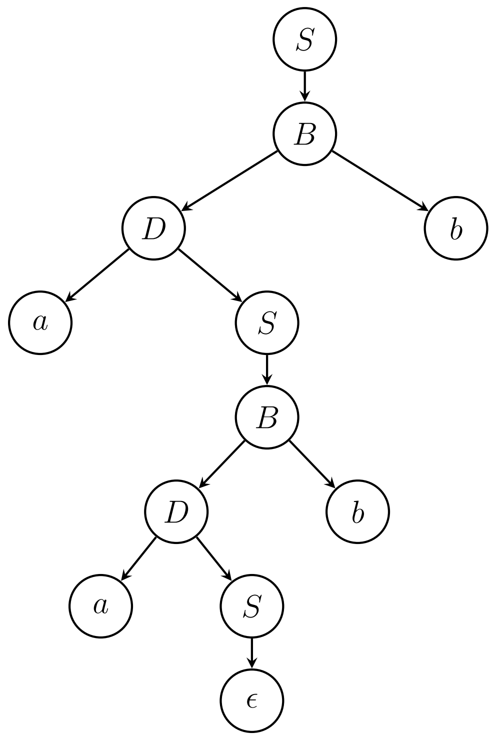
a)



b) The leftmost derivation yields the following sequence.

$$S, A, aC, aSb, aAb, aaCb, aaSbb, aabb$$

c)



d) The leftmost derivation yields the following sequence.

$$S, B, Db, aSb, aBb, aDbb, aaSbb, aabb$$

Problem 5

Given a starting state S the following context free grammar describes the language.

$$S \to LR$$
$$L \to aLb \mid \epsilon$$
$$R \to bRc \mid \epsilon$$