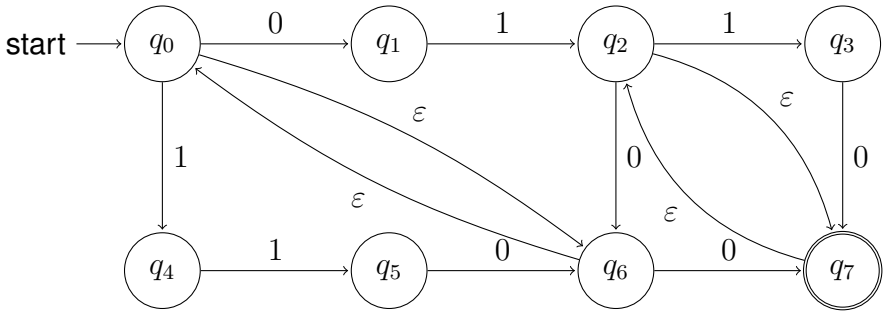
Assignment 2

Due Friday February 17th, 2017

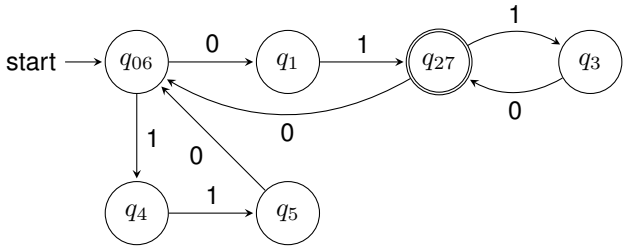
Submission Instructions: Submit solutions in a single PDF via OWL. Assignments are due at 11:55 pm (Eastern Time) on the date listed above. Assignments submitted more than 48 hours late will not be accepted, and a mark of zero (0) will be recorded. See the course outline for details.

1. [5 marks] DFAs, NFAs, and ε-NFAs

(a) [5 marks] The following NFA is an ε-NFA. That is, it contains transitions that occur on no input (i.e., the empty string). Convert this ε-NFA into an equivalent non ε-NFA, i.e., one that does not contain ε-transitions.



SOLUTION:



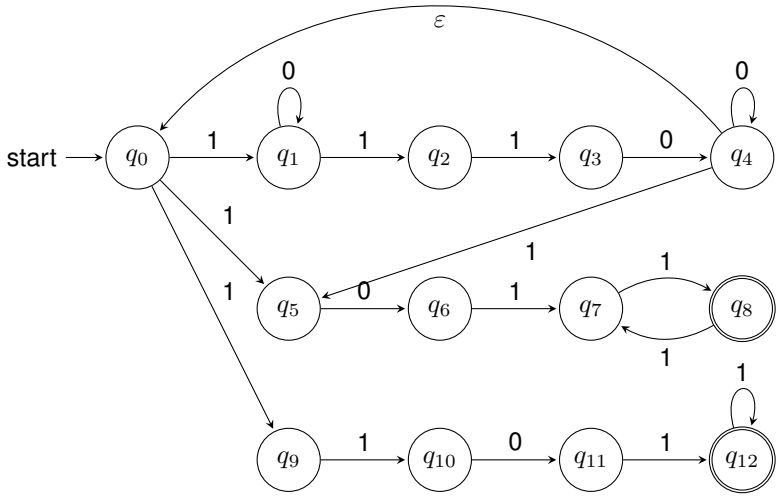
2. [9 marks] Regular Expressions

(a) [3 marks] Construct an ε-NFA that represents the language defined by the following regular expression:

$$R = (10^*110^+)^*10(11)^+ + 1101^+$$

Note the '$+$' notation has two meaning depending on context. a^+ means "one or more copies of a", while $a + b$ means a or b.

SOLUTION:



(b) [3 marks] Let R_1 and R_2 be the following regular expressions:

$$R_1 = (00 + 11)^+$$

$$R_2 = (11)^*$$

Using set builder notation, describe the following language:

$$L = \{L_1/L_2\},$$

where L_1 and L_2 are the languages represented by regular expressions R_1 and R_2, and where A/B denotes the right quotient of languages A and B.

SOLUTION:

$$L = \{w : w \in L_1 \cup \{\varepsilon\}\}$$

(c) [3 marks] On the alphabet $\Sigma = \{0, 1\}$, if we assume that 0 and 1 are binary digits and the strings in our language represent numeric values, describe, in English, the language that the following regular expression represents:

$$R = (1 + 0)^*1^+ + (01 + 001)^*$$

SOLUTION:

Any string of binary letters that end with at least one 1, or is a null string (empty set)

3. [18 marks] Pumping Lemma

(a) [6 marks] Use the pumping lemma to prove language L_a is not regular:

$$L_a = \{0^n 1^n 2^n : n > 0\}$$

SOLUTION:

First, assume L_a is regular. Let p be the pumping length of L_a. Toward a contradiction we will pick a candidate string s, where $s \in L_a$ and $|s| \geq p$, and we will proceed to show there is some way to divide s into 3 parts xyz such that y cannot be pumped.

Suppose we pick our candidate string to be $s = 0^p 1^p 2^p$. It is clear in this example that $s \in L_a$ and $|s| = 3p$ and thus $|s| \geq p$. As per the pumping lemma if L_a is regular, there must be SOME way to divide s into 3 parts xyz such that $xy^i z \in L_a$ for all $i \geq 0$.

***Case 1:** y is entirely within 0^p :*

Recall the pumping lemma defines $|xy| \leq p$, and that $|y| > 0$, (i.e., y is not the empty string). So if $|xy|$ is p characters or fewer in length, and the first p characters of s are 0's, then we know xy must contain only 0's, and therefore y must contain only 0's, i.e., there are no other cases of $s = xyz$ to consider.

We must now show there is some $i \geq 0$ for which $xy^i z \notin L_a$. For this language we can either pump up or down. Suppose we pump 'down,' i.e., let $i = 0$. Then $s_0 = xy^0 z = xz$. Since y contains some number of 0's, pumping down results in a string s_0 that has fewer 0's than 1's or 2's. Therefore $s_0 \notin L_a$. But according to the pumping lemma there must always be a way to divide s into 3 parts xyz such that $s_i = xy^i z \in L_a$ for all $i \geq 0$. We have arrived at a contradiction. This means L_a cannot be regular.

(b) [6 marks] Use the pumping lemma to prove language L_b is not regular:

$$L_b = \{10^i 10^{i+1} 10^{i+2} 1 ... 0^{i+n} 1 : i, n > 1\}$$

SOLUTION: First, assume L_b is regular. Let p be the pumping length of L_b. Let us examine $s = 10^p 10^{p+1} 10^{p+2} 1 ... 0^{p+n} 1$. In this example $s \in L_b$. If $i = 1$ then $|s| = p + 2$ which is greater than p, and s is longer for any other value of i.

We must now consider all possible ways to divide s into 3 parts xyz.

Case 1: y contains just the initial 1: In the case we have $x = \varepsilon$ and $y = 1$. Suppose we pump down, i.e., $s_0 = x$. Then s_0 will be missing the leading 1 and will not be in the language.

Case 2: y contains the initial 1 and some number of 0's: As in case 1 if we pumped down, the resulting string s_0 would be missing the leading 1 and would not be in the language.

Case 3: y contains only 0's: This case is a little trickier. Suppose this time we pump up, i.e., we examine $s_2 = xy^2z$. Since y contained some non-zero number of 0's, and $s = 10^p1p^{p+1}\ldots$, then repeating y twice means s_2 must have more 0's, i.e., $s_2 = 10^{p+k}10^{p+1}$ for some $k > 0$. That means the first run of 0's will contain more than it should relative to the second run of 0's, and therefore $s_2 \notin L_a$/

Are there any more cases to consider? Recall by our definition of s, we have a 1 followed by p 0's. Again as per the pumping lemma $|xy| \leq p$, meaning that y cannot extend past the initial run of p 0's at the beginning of the string. Therefore there are no other cases to consider. And all of the cases we did consider yielded counter examples where you could not pump the string (and have the result also be in the language). This is a contradiction as per the pumping lemma. Therefore L_b is not regular.

(c) [6 marks] Use the pumping lemma to prove language L_c is not regular:

$$L_c = \{0^n : n = m^3 \text{ for } m \geq 2 \text{ , i.e., } n = 8, 27, 64...\}$$

SOLUTION:

Again assume L_c is regular. Let p be the pumping length of L_c. Let us choose $s = 0^{p^3}$ as our candidate string. When we decompose the string, y must to be some non-zero number of 0's. In fact it's the only possibility. Let $y = 0^k$ of some unknown non-zero length k.

If we pump up, e.g., consider $s_2 = xy^2z$, we are adding k 0's to the string, giving us $s_2 = 0^{p^3+k}$. To demonstrate that this string cannot be in the language, we must prove the following inequality bounding consecutive cubes around our string length: $p^3 < p^3 + k < (p+1)^3$. That is, we must show that s_2 cannot result in a length which is a power of 3, and thus cannot be in the language.

First recall (as per the pumping lemma) that y is non-empty, i.e., $|y| > 0$ and thus $k > 0$. Therefore the left side of the inequality (i.e., $p^3 < p^3 + k$) is satisfied.

Now let's look at the right side, i.e., $p^3 + k < (p+1)^3$. Expanding these terms we have $p^3 + k < p^3 + 3p^2 + 3p + 1$. Now recall by the pumping lemma we have $|xy| \leq p$, meaning $k \leq p$. Thus the inequity holds for all possible k. Therefore $s_2 \notin L_c$. There are no other cases to consider (since y can only contain 0's). Therefore we conclude L_c cannot be regular.

4. [8 marks] Context-free Grammars

For each of the following languages, give the associated context-free grammar:

(a) [2 marks]

$$L_a = \{ww^R \ : \ w \in \{a, b, c\}^*\}$$

Note: w^R denotes the *reverse* of a string w.

SOLUTION: $S \to aSa \,|\, bSb \,|\, cSc|\varepsilon$

(b) [2 marks]

$$L_c = ((01^* + 0^*) + 11)^*$$

SOLUTION:

$S \to SS \,|\, \varepsilon \,|\, A \,|\, 11$

$A \to 0B \,|\, C$

$B \to \varepsilon \,|\, B1$

$C \to \varepsilon \,|\, 0C$

(c) [2 marks]

$$L_d = \{0^i 1^{i+j} 2^j \ : \ i, j > 1\}$$

SOLUTION:

$S \to 0A11B2$

$A \to 0A1 \,|\, 01$

$B \to 1B2 \,|\, 12$

Let CFG *G* be the following grammar:

$$S \to \mathsf{a}S\mathsf{b} \,|\, \mathsf{b}Y \,|\, Y\mathsf{a}$$
$$Y \to \mathsf{b}Y \,|\, \mathsf{a}Y \,|\, \epsilon$$

[2 marks] Give a description of *L(G)* in English
SOLUTION:

This language describes any string that is at least one character long excluding ab

5. [10 marks] Regex in Practice

For this question you will perform string matching on a large text file. Download the Complete Works of Shakespeare, a text file containing the entire collection of Shakespearean works, located here:

http://www.gutenberg.org/cache/epub/100/pg100.txt

Use Unix command-line utilities such as grep, sed, and awk. If you don't have Linux, there are many distros to choose from. I recommend Ubuntu run in Virtual Box.

For each question, give the answer in a .txt file and list of command(s) used. Find:

(a) All words that start with a 'th' and end in a consonant

SOLUTION: `grep -iwo 'th\w*[bcdfghjklmnpqrstvwxz]' pg100.txt | awk {print $0, "\r\n"}' | sort | uniq > Answers5a.txt`

(b) All words that are 12 characters long

SOLUTION: `grep -E -iwo '\w{12}' pg100.txt | awk '{print $0, "\r\n}"' pg 100.txt | sort | uniq > Answers5b.txt`

(c) All questions asked by Cleopatra

SOLUTION: `grep -o "CLEOPATRA.*\?" Shakespeare.txt > Answers5c.txt`

(d) The longest word whose only vowels are a's

SOLUTION:

```
grep -o -E -ignore-case "[abcdfghjklmnpqrstvwxz]+" Shakespeare.txt | awk
'if (length($0) > length(longest)) longest = $0  END  print longest ' >
Answers5d.txt
```

(e) The shortest word with two r's

SOLUTION: `grep -o "[a-zA-Z]*r[a-zA-Z]*r[a-zA-Z]*" Shakespeare.txt | awk '(NR==1 || length($0) < length(shortest)) {shortest = $0} END { print shortest }' > Answers5e.txt`