Alex Jia

CS 205
157000846

Homework 6

1. For each of the following strings, explain precisely why (why not) they belong to the language denoted by the regular expression 8 (9 | Λ) 8*9:

a. 8989 DOES belong to the language:
A=8, B=9 (from 9 | Λ), C=8 (from 8*), D=9. Each of these exist in the language denoted by 8 (9 | Λ) 8*9.

b. 889 DOES belong to the language:
A=8, B=Λ (null string, so there is nothing there), C=8 (from 8*), D=9. Each of these exist in the language denoted by 8 (9 | Λ) 8*9.

c. 989 DOES NOT belong to the language:
A=null, but it SHOULD be an 8 (because the language is denoted by 8 (9 | Λ) 8*9, and 8 is the FIRST character in the string, NOT 9 (or null, if you prefer that perspective).

d. 89 DOES belong to the language:

A=8, B=Λ (null string, so there is nothing there), C=null(from 8*, which means 0 to infinite repeating '8's. Therefore, 0 '8's is valid as well.), D=9.

2.In each of the cases below, write the shortest/clearest regular expression over the alphabet Σ = {b, c}, whose language contains exactly the sentences described. Explain why your answer is correct (i.e., why it has all the required strings, and no extra ones.)
a. c(cc)*
This will give us either 1 ‘c’, or 1 + (2n) ‘c’s, where n is a number between 0 and infinity.
b. (bb)*
This will give us (2n) ‘b’s, where n is a number between 0 and infinity.
c. c*(bcc)*
The specifications of the question ask for exactly *two ‘c’s after every ‘b’*. Thus, (bcc)* fulfills this n times, where n refers to a number between 0 and infinity. The c* in front of the (bcc)* fulfills the unspoken requisite; the case where no ‘b’ exists. Here, the user may input as many ‘c’s as desired; thus, the entire expression fulfills every possible outcome, including the null/empty strings.

3.Dealing with ∅. (Recall that L(∅) is the empty set {}.)
a. 5*
Since we know that the given alphabet is equivalent to ∅ | (5(∅)*), and that ∅ refers to the empty set, we can simplify the expression to be either the empty set, OR a single 5 followed by 0 to infinity empty sets.
c. L(erase8(R)) = { character *w*, where if (w == ‘8’), then w = ‘/\’, else ignore)

4. Given arbitrary regular expression R, prove formally in detail, or disprove with the smallest counterexample, each of the following 3 hypotheses:
(a) R*S* == (RS)*
Let R = 8, and S = 9.
For the LHS, let R* = one ‘8’, and S* = two ‘9’s. Thus, LHS = 899. No matter what you do with the RHS, based off the above definitions, RHS = any combination of ‘89’s. LHS != RHS

(b) R* == R*R*

Given arbitrary string w, let us define R as 8. L(LHS) = L(R*) = L(8*). This is equivalent to L(8*8*) = L(R*R*) = L(RHS). Similarly, if we take a look at L(RHS) with the given definition of R = 9, L(9*9*) = L(9*)=L(R*)=L(LHS). Thus, the two statements are equivalent.

(c) (RS*)* == Λ | R(R | S)*

Let R = 8, and S = 9

For the LHS, let (RS*)* be equivalent to R. For the RHS, let /\ | R(R|S)* be equivalent to RR. LHS != RHS. While LHS and RHS can be equivalent in many cases, upon closer closer inspection of either statement, the RHS can either have an empty set, or at least one R, followed by an infinite number of Rs and Ses. However, the LHS can either have an empty set, or an infinite number of Rs followed by another equally infinite-yet-still-distinct number of Ses.