CSE 340 FALL 2018

Homework 1 Solution

Problem 1. Consider the following regular expressions

R_0 = 1 | 2 | 3

R_1 = 1 | 2 | 4 | 8

R_2 = (a | b) (a* | b*) (a | b)

R_3 = (a* | b*) R_1* (ab)*

R_4 = ab R_3* (a | b)*

R_5 = R_3* aaa R_2*

In the expressions, the dot operator is omitted and some parentheses are omitted, in which case the Kleene star operator (*) has the highest precedence, followed by the dot operator (.), followed by the or operator (|).

Let `getToken()` be a function that returns the next token in the input. If we call it repeatedly it will return one token after another. When all the input is consumed, `getToken()` returns EOF (end of file). Assume that longest prefix-matching rule is used by `getToken()` and ties are broken in favor of the regular expression listed first.

1. Give an example of input for which calling getToken() twice returns R0 first then EOF

Answer: 1

calling getToken() on input consisting only of 1 will return R0. Even though it also matches R1, R0 is listed first so, R0 will be returned. Since there is no input after 1, a second call to getToken() will return EOF.

2 and 3 are also acceptable answers.

2. Give an example of input for which calling getToken() twice returns R1 first then EOF

Answer: 4

Calling getToken() on input consisting only of 4 will return R1. Even though 4 also matches R3, R1 is listed first. Also 4 does not match R0. Since there is no input after 4, a second call to getToken() will return EOF.

8 is also an acceptable answer.

3. Give an example of input for which calling getToken() twice returns R2 first then EOF

Answer: ab

ab is a lexeme for R2. It is not a lexeme for R0 or a lexeme for R1. Even though ab is also a lexeme for R3 and R4, calling getToken() on ab will return R2 because R2 is listed before R3 and R4. Since there is no input after 4, a second call to getToken() will return EOF.

4. Give an example of input for which calling getToken() twice returns R3 first then EOF

Answer: 8ab

8ab is a lexeme for R3 and is not a lexeme for any other token. 8 is a lexeme for R1, but R1 is not returned because 8 is a shorter match (longest matching prefix rule). Since there is no input after 8ab, a second call to getToken() will return EOF.

5. Give an example of input for which calling getToken() twice returns R4 first then EOF

Answer: ab1

ab1 is a lexeme for R4 and is not a lexeme for any other token. ab is a lexeme for R2, but since it is shorter match R1 is not returned. Since there is no input after ab1, a second call to getToken() will return EOF.

6. Give an example of input for which calling getToken() twice returns R5 first then EOF

Answer: 8aaa

8aaa is a lexeme for R5 and is not a lexeme for any other token. 8 is a lexeme for R1 but R1 is not returned because 8 is shorter than 8aaa.

7. If getToken() is called repeatedly on the following input, what is the sequence of tokens and lexemes returned? In your work, show step by step the Matched, Potential (Viable), and Maximal tokens.

aaa1ba1ba1daa1ab

Answer: Here is a step by step lexical analysis of the input

Token	Matched	Viable	Maximal
aaa1ba1ba1daa1ab . ^	R3	R2,R3,R4,R5	-
aaa1ba1ba1daa1ab . ^	R2,R3	R2,R3,R5	-
aaa1ba1ba1daa1ab . ^	R2,R3, R5	R2,R3,R5	-
aaa1ba1ba1daa1ab . ^	R3	R3,R5	R2 (aaa)
aaa1ba1ba1daa1ab . ^		R5	R3 (aaa1)
aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^		R5	

aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^	-	-	Return R3 “aaa1”
aaa1ba1ba1daa1ab . ^	R3	R2,R3,R5	
aaa1ba1ba1daa1ab . ^	R2	R2,R3,R5	R3 “b”
aaa1ba1ba1daa1ab . ^	-	R5	R2 “ba”
aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^	-	-	Return R2 “ba”
aaa1ba1ba1daa1ab . ^	R0,R1,R3	R3,R5	
aaa1ba1ba1daa1ab . ^		R5	R0 “1”
aaa1ba1ba1daa1ab . ^		R5	

aaa1ba1ba1daa1ab . ^		R5	
aaa1ba1ba1daa1ab . ^		-	Return R0 “1”
aaa1ba1ba1daa1ab . ^	R3	R2,R3,R5	
aaa1ba1ba1daa1ab . ^	R2	R2,R3,R5	R3 “b”
aaa1ba1ba1daa1ab . ^		R5	R2 “ba”
aaa1ba1ba1daa1ab . ^			Return R2 “ba”
aaa1ba1ba1daa1ab . ^	R0,R1,R3	R3,R5	
aaa1ba1ba1daa1ab . ^	-	-	Return R0 “1”
aaa1ba1ba1daa1ab . ^	-	-	Return Error

The token and lexemes that are returned are :

R3 “aaa1” , R2 “ba” , R0 “1” , R2 “ba” , R0 “1” , and Error

Problem 2. Let R1 and R2 be two regular expressions over the alphabet {a , b}

1. Is it always the case that L(((R1).(R2))*) = L(((R1) | (R2))*)

Answer: False. We show that this is not always satisfied by giving a counter example. Let R1 = a and R2 = b. L1 = L(((R1).(R2))*) = L((ab)*) and L2 = L(((R1) | (R2))*) = L((a | b)*). The string ba belongs to L2 but not L1.

2. Is it always the case that L(((R1)*.(R2)*)*) = L(((R1) | (R2))*)?

Answer: True.

A string in L1 consists the concatenation of zero or more string sequence from L((R1)*.(R2)*). A string in L((R1)*.(R2)*) consists of zero or more concatenation of strings from L(R1) followed by zero or more concatenations of strings from L(R2). In particular L((R1)*.(R2)*) contains all strings in L(R1) and all string in L(R2) which means that L((R1)*.(R2)*) contains L(R1 | R2). It follows that L1 contains L2. Similarly we can show that L2 contains L1.

3. Is it always the case that L((R1 *)*) = L((R1)*)?

Answer: True

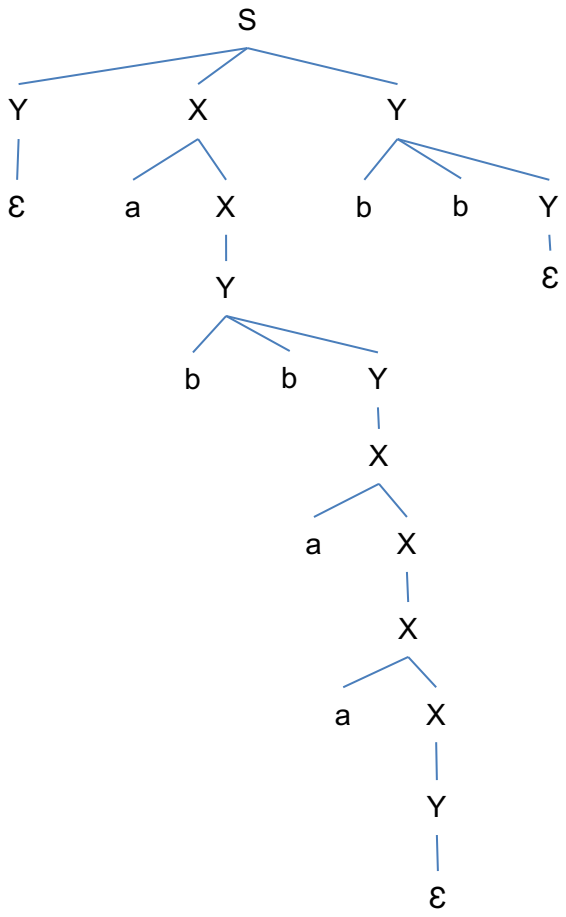
The argument is similar to the one made for part 2 above. **L((R1 *)*)** consists of concatenation of zero or more strings in **L(R1 *)** which contains all strings of **L((R1)*)**. Also, every string in **L((R1 *)*)** consists of zero or more concatenation of strings s_1, s_2, s_k from and each string s_i ($1 \le i \le k$) consists of zero or more concatenations of strings from **L(R1)** . So every string in **L((R1 *)*)** is also a string in **L(R1 *)**.

Problem 3. Consider the grammar

$S \rightarrow Y\ X\ Y$
$X \rightarrow a\ X \mid Y$
$Y \rightarrow b\ b\ Y \mid X \mid \varepsilon$

Draw a parse tree for input string abbaabb

Answer:



Problem 4. Consider the grammar

$S \rightarrow A$
$S \rightarrow a$
$A \rightarrow a\ b\ S \mid a\ b\ S\ d\ S$

1. What are the non-terminals?

Answer Non-terminal : S, A

2. What is the start symbol?

Answer Start Symbol: S

3. What are the terminals?

Answer Terminal: a, b, d

4. Show that this grammar is ambiguous (this is not an easy question and would require some effort to find a input that has two parse trees)

Answer For string: ababada there are two parse trees.