Coursework 1 (Strand 1)

This coursework is worth 4% and is due on 25 October at 16:00. You are asked to implement a regular expression matcher and submit a document containing the answers for the questions below. You can do the implementation in any programming language you like, but you need to submit the source code with which you answered the questions, otherwise a mark of 0% will be awarded. You can submit your answers in a txt-file or pdf. Code send as code.

Disclaimer

It should be understood that the work you submit represents your own effort. You have not copied from anyone else. An exception is the Scala code I showed during the lectures or uploaded to KEATS, which you can freely use.

Task

The task is to implement a regular expression matcher based on derivatives of regular expressions. The implementation should be able to deal with the usual (basic) regular expressions

$$\mathbf{0},\ \mathbf{1},\ c,\ r_1 + r_2,\ r_1 \cdot r_2,\ r^*$$

but also with the following extended regular expressions:

$[c_1 c_2 \ldots c_n]$	a range of characters
r^+	one or more times r
$r^?$	optional r
$r^{\{n,m\}}$	at least n-times r but no more than m-times
$\sim r$	not-regular expression of r

In the case of $r^{\{n,m\}}$ you can assume the convention that $0 \leq n \leq m$. The meanings of the extended regular expressions are

$$\begin{array}{rcl} L([c_1 c_2 \ldots c_n]) & \stackrel{\text{def}}{=} & \{[c_1], [c_2], \ldots, [c_n]\} \\ L(r^+) & \stackrel{\text{def}}{=} & \bigcup_{1 \leq i} . L(r)^i \\ L(r^?) & \stackrel{\text{def}}{=} & L(r) \cup \{[]\} \\ L(r^{\{n,m\}}) & \stackrel{\text{def}}{=} & \bigcup_{n \leq i \leq m} . L(r)^i \\ L(\sim r) & \stackrel{\text{def}}{=} & \Sigma^* - L(r) \end{array}$$

whereby in the last clause the set Σ^* stands for the set of *all* strings over the alphabet Σ (in the implementation the alphabet can be just what is represented by, say, the type `Char`). So $\sim r$ means 'all the strings that r cannot match'.

Be careful that your implementation of *nullable* and *der* satisfies for every r the following two properties (see also Question 2):

- $nullable(r)$ if and only if $[] \in L(r)$
- $L(der\, c\, r) = Der\, c\, (L(r))$

Important! Your implementation should have explicit cases for the basic regular expressions, but also explicit cases for the extended regular expressions. That means do not treat the extended regular expressions by just translating them into the basic ones. See also Question 2, where you are asked to explicitly give the rules for *nullable* and *der* for the extended regular expressions.

Question 1

What is your King's email address (you will need it in Question 3)?

Question 2

From the lectures you have seen the definitions for the functions *nullable* and *der* for the basic regular expressions. Implement the rules for the extended regular expressions:

$$
\begin{array}{ll}
nullable([c_1 c_2 \ldots c_n]) & \stackrel{\text{def}}{=} ? \\
nullable(r^+) & \stackrel{\text{def}}{=} ? \\
nullable(r^?) & \stackrel{\text{def}}{=} ? \\
nullable(r^{\{n,m\}}) & \stackrel{\text{def}}{=} ? \\
nullable(\sim r) & \stackrel{\text{def}}{=} ? \\
\\
der\, c\, ([c_1 c_2 \ldots c_n]) & \stackrel{\text{def}}{=} ? \\
der\, c\, (r^+) & \stackrel{\text{def}}{=} ? \\
der\, c\, (r^?) & \stackrel{\text{def}}{=} ? \\
der\, c\, (r^{\{n,m\}}) & \stackrel{\text{def}}{=} ? \\
der\, c\, (\sim r) & \stackrel{\text{def}}{=} ?
\end{array}
$$

Remember your definitions have to satisfy the two properties

- $nullable(r)$ if and only if $[] \in L(r)$
- $L(der\, c\, r)) = Der\, c\, (L(r))$

Give the implementation and the text-version of the clauses above.

Question 3

Implement the following regular expression for email addresses

$$([a\text{-}z0\text{-}9\_.\!-]^+) \cdot @ \cdot ([a\text{-}z0\text{-}9.\!-]^+) \cdot . \cdot ([a\text{-}z.]^{\{2,6\}})$$

and calculate the derivative according to your email address. When calculating the derivative, simplify all regular expressions as much as possible by applying the following 7 simplification rules:

$$\begin{array}{lcl} r \cdot \mathbf{0} & \mapsto & \mathbf{0} \\ \mathbf{0} \cdot r & \mapsto & \mathbf{0} \\ r \cdot \mathbf{1} & \mapsto & r \\ \mathbf{1} \cdot r & \mapsto & r \\ r + \mathbf{0} & \mapsto & r \\ \mathbf{0} + r & \mapsto & r \\ r + r & \mapsto & r \end{array}$$

Write down your simplified derivative in a readable notation using parentheses where necessary. That means you should use the infix notation $+$, $\cdot$, * and so on, instead of code.

Question 4

Suppose *[a-z]* stands for the range regular expression $[a, b, c, \ldots, z]$. Consider the regular expression $/ \cdot * \cdot (\sim ([a\text{-}z]^* \cdot * \cdot / \cdot [a\text{-}z]^*)) \cdot * \cdot /$ and decide wether the following four strings are matched by this regular expression. Answer yes or no.

1. `"/**/"`
2. `"/*foobar*/"`
3. `"/*test*/test*/"`
4. `"/*test/*test*/"`

Also test your regular expression matcher with the regular expressions $a^{\{3,5\}}$ and $(a^?)^{\{3,5\}}$. Test whether the strings

5. `aa`
6. `aaa`
7. `aaaaa`
8. `aaaaaa`

are matched or not. Does your matcher produce the expected results?

Question 5

Let r_1 be the regular expression $a \cdot a \cdot a$ and r_2 be $(a^{\{19,19\}}) \cdot (a^?)$. Decide whether the following three strings consisting of as only can be matched by $(r_1^+)^+$. Similarly test them with $(r_2^+)^+$. Again answer in all six cases with yes or no.

These are strings are meant to be entirely made up of as. Be careful when copy-and-pasting the strings so as to not forgetting any a and to not introducing any other character.

1. "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

2. "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

3. "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"