Homework 1

Please submit your solutions via email. Please submit only PDFs! Every solution should be preceeded by the corresponding question text, like:

Qn:	**...a difficult question from me...**
A:	**...an answer from you ...**
Q$n+1$	**...another difficult question...**
A:	**...another brilliant answer from you...**

Solutions will only be accepted until 20th December! Please send only one homework per email.

1. **(Optional)** If you want to run the code presented in the lectures, install the Scala programming language available (for free) from

   http://www.scala-lang.org

   If you want to follow the code I present during the lectures, it might be useful to install VS Code or Codium. I will be using Scala Version 3.5, which has the `scala-cli` REPL used in PEP already built in.

2. **(Optional)** Have a look at the catastrophic backtracking programs uploaded on KEATS. Convince yourself that they really require a lot of computation time. If you have similar examples in your own favourite programming language, I am happy to hear about it.

3. Read the handout of the first lecture and the handout about notation. Make sure you understand the concepts of strings and languages. In the context of the CFL-course, what is meant by the term *language*?

   A language - in this context - is just a set of strings. Some of these sets can actually not be described by regular expressions. Only regular languages can. This is something for lecture 3.

4. Give the definition for regular expressions—this is an inductive datatype. What is the meaning of a regular expression? (Hint: The meaning is defined recursively.)

   Here I would also expect the grammar for basic regular expressions and the definition of the recursive L-function. Discuss differences between $r_1 + r_2$ and r^+. Discuss differences between "real-life regexes" and regexes in this module.

5. Assume the concatenation operation of two strings is written as $s_1 @ s_2$. Define the operation of *concatenating* two sets of strings. This operation is also written as _@_. According to this definition, what is $A @ \{\}$ equal to? Is in general $A @ B$ equal to $B @ A$?

What is $A @ []$? Are there special cases where $A @ B = B @ A$? Obviously when $A = B$ the stament is true. But there are also cases when $A \neq B$, for example $A = \{a\}$ and $B = \{aaa\}$.

6. Assume a set A contains 4 strings and a set B contains 7 strings. None of the strings is the empty string. How many strings are in $A @ B$?

   Everyone will probably answer with 28, but there are corner cases where there are fewer than 28 elements. Can students think of such corner cases? For example $A = \{a, ab, \ldots\}$, $B = \{bc, c, \ldots\}$

7. How is the power of a language defined? (Hint: There are two rules, one for $\_^0$ and one for $\_^{n+1}$.)

   Two rules: 0-case and n+1 case.

8. Let $A = \{[a], [b], [c], [d]\}$. (1) How many strings are in A^4? (2) Consider also the case of A^4 where one of the strings in A is the empty string, for example $A = \{[a], [b], [c], []\}$.

   121 is correct. But make sure you understand why it is 121 in cases you do not have a computer at your fingertips.

9. (1) How many basic regular expressions are there to match **only** the string *abcd*? (2) How many if they cannot include $\mathbf{1}$ and $\mathbf{0}$? (3) How many if they are also not allowed to contain stars? (4) How many if they are also not allowed to contain $\_ + \_$?

   1-3 are infinite (tell the idea why and give examples); 4 is five - remember regexes are trees (that is the main point of the question.

10. When are two regular expressions equivalent? Can you think of instances where two regular expressions match the same strings, but it is not so obvious that they do? For example $a + b$ and $b + a$ do not count…they obviously match the same strings, namely $[a]$ and $[b]$.

    for example $r^* = 1 + r \cdot r^*$ for any regular expression r. Can students think about why this is the case? - this would need a proof.

11. What is meant by the notions *evil regular expressions* and by *catastrophic backtracking*?

    catastrophic backtracking also applies to other regexes, not just $(a^*)^*b$. Maybe https://www.trevorlasn.com/blog/when-regex-goes-wrong/ is of help - even the CrowdStrike issue had an underlying problem with a regex, though this one was not due to catastrophic backtracking.

12. Given the regular expression $(a+b)^* \cdot b \cdot (a+b)^*$, which of the following regular expressions are equivalent

$$\begin{array}{ll} 1) & (ab+bb)^* \cdot (a+b)^* \\ 2) & (a+b)^* \cdot (ba+bb+b) \cdot (a+b)^* \\ 3) & (a+b)^* \cdot (a+b) \cdot (a+b)^* \end{array}$$

    no, yes (why?), no.

13. Given the extended regular expression `[b-d]a?e+`, what does the equivalent basic regular expression look like?

    $(b+c+d) \cdot (a+\mathbf{1}) \cdot (e \cdot e^*)$

14. **(Optional)** This question is for you to provide regular feedback to me: for example what were the most interesting, least interesting, or confusing parts in this lecture? Any problems with my Scala code? Please feel free to share any other questions or concerns. Also, all my material is ~~crap~~ imperfect. If you have any suggestions for improvement, I am very grateful to hear.

    If *you* want to share anything (code, videos, links), you are encouraged to do so. Just drop me an email or send a message to the Forum.