Homework 4

1. Given the regular expressions

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

   there are several values for how these regular expressions can recognise the strings (for 1) ab and (for 2) aaa. Give in each case *all* the values and indicate which one is the POSIX value.

2. If a regular expression r does not contain any occurrence of $\mathbf{0}$, is it possible for $L(r)$ to be empty? Explain why, or give a proof.

3. Define the tokens and regular expressions for a language consisting of numbers, left-parenthesis $($, right-parenthesis $)$, identifiers and the operations $+$, $-$ and $*$. Can the following strings in this language be lexed?

   - $(a + 3) * b$
   - $)() + + - 33$
   - $(a/3) * 3$

   In case they can, can you give the corresponding token sequences.

4. Assume r is nullable. Show that

$$1 + r + r \cdot r \;\equiv\; r \cdot r$$

   holds.

5. Construct a regular expression that can validate passwords. A password should be at least 8 characters long and consist of upper- and lower-case letters and digits. It should contain at least a single lower-case letter, at least a single upper-case letter and at least a single digit. If possible use the intersection regular expression from CW1, written _&_, and the bounded regular expressions; you can also assume a regular expression written `ALL` that can match any character.

6. Assume the delimiters for comments are `/*` and `*/`. Give a regular expression that can recognise comments of the form

```
/* ... */
```

   where the three dots stand for arbitrary characters, but not comment delimiters. (Hint: You can assume you are already given a regular expression written `ALL`, that can recognise any character, and a regular expression `NOT` that recognises the complement of a regular expression.)

7. Simplify the regular expression

$$(\mathbf{0} \cdot (b \cdot c)) + ((\mathbf{0} \cdot c) + \mathbf{1})$$

   Does simplification always preserve the meaning of a regular expression?

8. The Sulzmann & Lu algorithm contains the function *mkeps* which answers how a regular expression can match the empty string. What is the answer of *mkeps* for the regular expressions:

$$(\mathbf{0} \cdot (b \cdot c)) + ((\mathbf{0} \cdot c) + \mathbf{1})$$
$$(a + \mathbf{1}) \cdot (\mathbf{1} + \mathbf{1})$$
$$a^*$$

9. What is the purpose of the record regular expression in the Sulzmann & Lu algorithm?

10. Recall the functions *nullable* and *zeroable*. Define recursive functions *atmostempty* (for regular expressions that match no string or only the empty string), *somechars* (for regular expressions that match some non-empty string), *infinitestrings* (for regular expressions that can match infinitely many strings).

11. There are two kinds of automata that are generated for regular expression matching—DFAs and NFAs. (1) Regular expression engines like the one in Python generate NFAs. Explain what is the problem with such NFAs and what is the reason why they use NFAs. (2) Regular expression engines like the one in Rust generate DFAs. Explain what is the problem with these regex engines and also what is the problem with $a^{\{1000\}}$ in these engines.

12. **(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.