

Homework 3

1. The regular expression matchers in Java, Python and Ruby can be very slow with some (basic) regular expressions. What is the main reason for this inefficient computation?
2. What is a regular language? Are there alternative ways to define this notion? If yes, give an explanation why they define the same notion.
3. Why is every finite set of strings a regular language?
4. Assume you have an alphabet consisting of the letters a, b and c only. (1) Find a regular expression that recognises the two strings ab and ac . (2) Find a regular expression that matches all strings *except* these two strings. Note, you can only use regular expressions of the form

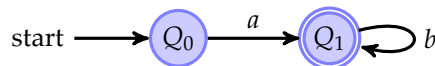
$$r ::= \emptyset \mid 1 \mid c \mid r_1 + r_2 \mid r_1 \cdot r_2 \mid r^*$$

5. Given the alphabet $\{a, b\}$. Draw the automaton that has two states, say Q_0 and Q_1 . The starting state is Q_0 and the final state is Q_1 . The transition function is given by

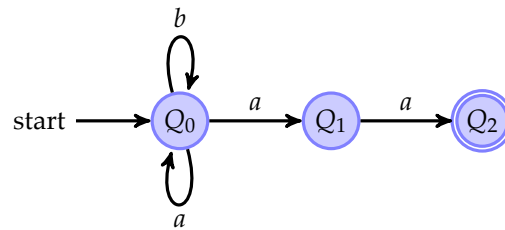
$$\begin{aligned}(Q_0, a) &\rightarrow Q_0 \\ (Q_0, b) &\rightarrow Q_1 \\ (Q_1, b) &\rightarrow Q_1\end{aligned}$$

What is the language recognised by this automaton?

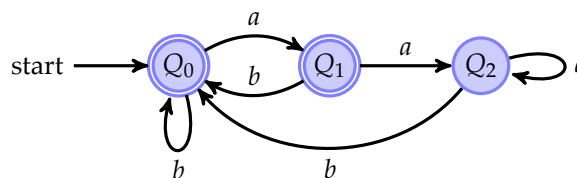
6. Give a non-deterministic finite automaton that can recognise the language $L(a \cdot (a + b)^* \cdot c)$.
7. Given a deterministic finite automaton $A(\Sigma, Q, Q_0, F, \delta)$, define which language is recognised by this automaton. Can you define also the language defined by a non-deterministic automaton?
8. Given the following deterministic finite automaton over the alphabet $\{a, b\}$, find an automaton that recognises the complement language. (Hint: Recall that for the algorithm from the lectures, the automaton needs to be in completed form, that is have a transition for every letter from the alphabet.)



9. Given the following non-deterministic finite automaton over the alphabet $\{a, b\}$, find a deterministic finite automaton that recognises the same language:



10. Given the following finite deterministic automaton over the alphabet $\{a, b\}$:



Give a regular expression that can recognise the same language as this automaton. (Hint: If you use Brzozowski's method, you can assume Arden's lemma which states that an equation of the form $q = q \cdot r + s$ has the unique solution $q = s \cdot r^*$.)

11. If a non-deterministic finite automaton (NFA) has n states. How many states does a deterministic automaton (DFA) that can recognise the same language as the NFA maximal need?
12. Rust implements a non-backtracking regular expression matcher based on the classic idea of DFAs. Still, some regular expressions take a surprising amount of time for matching problems. Explain the problem?
13. On Mentimeter there was a question: "Why does the [regex] $(a^*)^*b$ takes much longer for strings of length 28 compared to say 25?"

For this consider a lake with $1000m^2$ surface and an invasive plant that tries to cover the lake with leaves, think of the famous water lily that produces leaves on which you can stand. This plant starts out with a seedling covering just $0.001m^2$ of the lake, but doubles every day the surface that is covers. So on day two it would cover $0.002m^2$, on day three $0.004m^2$ and so on. How many days does the plant need to cover the entire lake? How many days is the lake still 90% uncovered?

14. Prove that for all regular expressions r we have

$$\text{nullable}(r) \quad \text{if and only if} \quad \epsilon \in L(r)$$

Write down clearly in each case what you need to prove and what are the assumptions.

15. **(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 ~~erap~~ 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.