Q1 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?

Because the regex engine of these languages uses the NFA structure, it needs to search in the form of DFS, which will lead to catastrophic backtracking.

Q2 What is a regular language? Are there alternative ways to define this no- tion? If yes, give an explanation why they define the same notion.

(1) A language is a set of strings. A regular expression specifies a language. A language is regular iff there exists a regular expression that recognises all its strings.

(2) Yes.

(3) A language is regular iff there exists a deterministic finite automaton that recognises all its strings. According to Kleene's theorem, we know that DFA, NFA, and regular expressions can be translated to each other, so they can all be used to describe regular language.

Q3 Why is every finite set of strings a regular language?

(1) Regular language is the language that can be recognized by regular expression.

(2) A regular language can be recognised by a regular expression. Therefore, for every finite set of strings/finite language, we can definitely construct a regular expression that is the union of all strings in this language. For example, assume we can recognise respectively, we can build r1 + r2 +... + rn to recognise.

(3) A language is regular iff there exists a regular expression that recognises all its strings

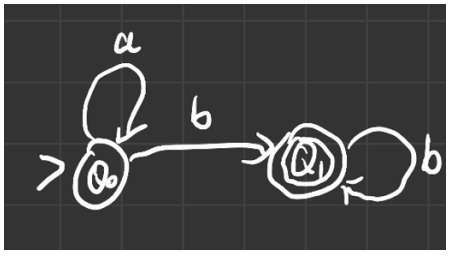
(4) A language is regular iff there exists an automaton that recognises all its strings.

Q4 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::=0|1|c|r1+r2 |r1·r2 |r∗

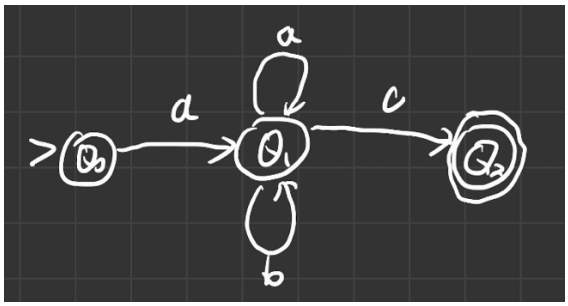
(1) ac + ab
(2) (a+b+c+ab+bc+ba+bb+ca+cb+cc) · (1+(a+b+c)*)

Q5 What is the language recognised by this automaton?

a*bb*



Q6 Give a non-deterministic finite automaton that can recognise the language L(a·(a+b)∗ ·c).

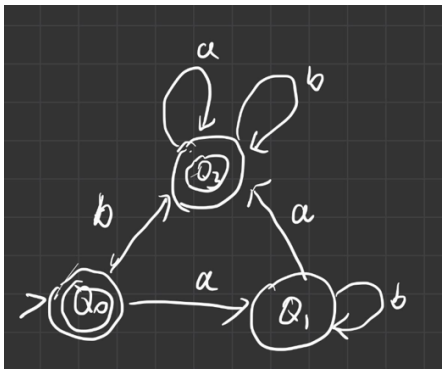


Q7 Given a deterministic finite automaton A(Σ,Q,Q0,F,δ), define which language is recognised by this automaton. Can you define also the language defined by a non-deterministic automaton?

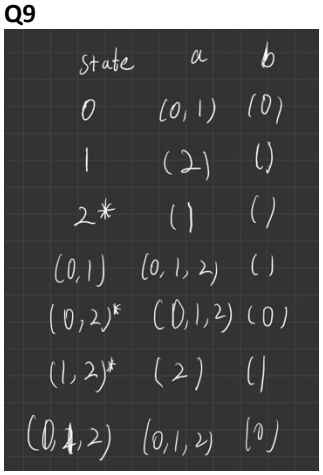
A *deterministic finite automaton* (DFA), say A, is given by a five-tuple written $\mathcal{A}(\Sigma, Qs, Q_0, F, \delta)$ where

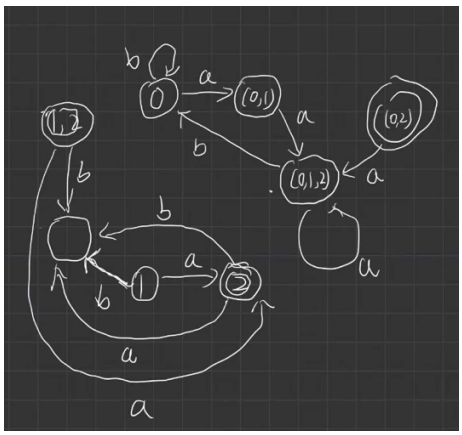
- Σ is an alphabet,
- Qs is a finite set of states,
- $Q_0 \in Qs$ is the start state,
- $F \subseteq Qs$ are the accepting states, and
- δ is the transition function.

Q8

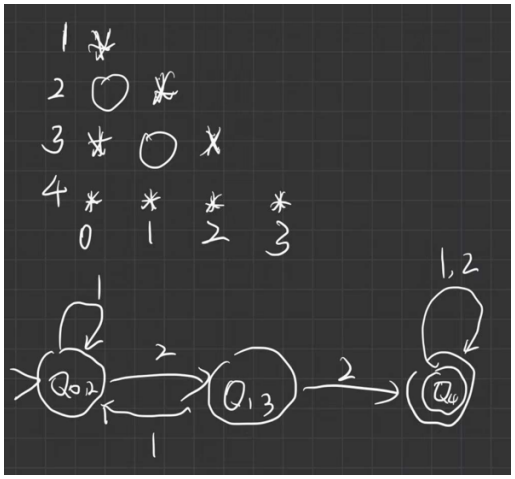


Q9





Q10



Q11

$$
\begin{array}{lcll}
Q_0 & = & \mathbf{1} + Q_0\,b + Q_1\,b + Q_2\,b & (3)\\
Q_1 & = & Q_0\,a & (4)\\
Q_2 & = & Q_1\,a + Q_2\,a & (5)
\end{array}
$$

$$
\begin{array}{lcll}
Q_0 & = & \mathbf{1} + Q_0\,b + Q_0\,a\,b + Q_2\,b & (6)\\
Q_2 & = & Q_0\,a\,a + Q_2\,a & (7)
\end{array}
$$

$$
\begin{array}{lcll}
Q_0 & = & \mathbf{1} + Q_0\,(b + a\,b) + Q_2\,b & (8)\\
Q_2 & = & Q_0\,a\,a + Q_2\,a & (9)
\end{array}
$$

$$
\begin{array}{lcll}
Q_0 & = & \mathbf{1} + Q_0\,(b + a\,b) + Q_2\,b & (10)\\
Q_2 & = & Q_0\,a\,a\,(a^*) & (11)
\end{array}
$$

$$Q_0 = \mathbf{1} + Q_0\,(b + a\,b) + Q_0\,a\,a\,(a^*)\,b \quad (12)$$

$$Q_0 = \mathbf{1} + Q_0\,(b + a\,b + a\,a\,(a^*)\,b) \quad (13)$$

$$Q_0 = \mathbf{1}\,(b + a\,b + a\,a\,(a^*)\,b)^* \quad (14)$$

$$Q_0 = (b + a\,b + a\,a\,(a^*)\,b)^* \quad (15)$$

$$
\begin{array}{lcll}
Q_0 & = & (b + a\,b + a\,a\,(a^*)\,b)^* & (16)\\
Q_1 & = & (b + a\,b + a\,a\,(a^*)\,b)^*\,a & (17)\\
Q_2 & = & (b + a\,b + a\,a\,(a^*)\,b)^*\,a\,a\,(a)^* & (18)
\end{array}
$$

Q12

$NFA = n$

$DFA = 2^n$

Q13

$$
\begin{array}{lcl}
nullable(\mathbf{0}) & \stackrel{\text{def}}{=} & false\\
nullable(\mathbf{1}) & \stackrel{\text{def}}{=} & true\\
nullable(c) & \stackrel{\text{def}}{=} & false\\
nullable(r_1 + r_2) & \stackrel{\text{def}}{=} & nullable(r_1) \vee nullable(r_2)\\
nullable(r_1 \cdot r_2) & \stackrel{\text{def}}{=} & nullable(r_1) \wedge nullable(r_2)\\
nullable(r^*) & \stackrel{\text{def}}{=} & true
\end{array}
$$