Assignment G2

Morten Fausing & Jens Kristian Refsgaard Nielsen & Thomas Vinther

14. februar 2019

1

Given a number of regular expressions $E_1, \ldots, E_n$ we wish to compute the intersection as a minimal DFA

$$\bigcap_{i=1}^{n} E_i$$

To do this we apply De Morgans law on complements of sets, where $A^c := A^{'} = X \setminus A = \{x \in X \mid x \notin A\}$

$$\begin{aligned}(A \cap B)^c &\overset{\text{De Morgan}}{=} A^c \cup B^c \Rightarrow \\ A \cap B = ((A \cap B)^c)^c &= (A^c \cup B^c)^c \Rightarrow \\ \left(\bigcup_{i=1}^{n} E_i^c\right)^c &= \bigcap_{i=1}^{n} (E_i^c)^c = \bigcap_{i=1}^{n} E_i\end{aligned}$$

With this result in hand we construct the following procedure:

$$E_1, \ldots, E_n \xrightarrow{\text{Kleene, (JM) Theorem 3.25}} N_1, \ldots, N_n \in \text{NFA} \tag{1.1}$$

$$N_1, \ldots, N_n \xrightarrow{\text{Subset construction, (JM) Theorem 3.18}} D_1, \ldots, D_n \in \text{DFA} \tag{1.2}$$

$$D_1, \ldots, D_n \xrightarrow{\text{Complement}} D_1^c, \ldots D_n^c \in \text{DFA} \tag{1.3}$$

$$D_1^c, \ldots D_n^c \xrightarrow{\Lambda \text{ Union}} \bigcup_{i=1}^{n} D_i^c = N \in \text{NFA} \tag{1.4}$$

$$N \xrightarrow{\text{Subset construction, (JM) Theorem 3.18}} D \in \text{DFA} \tag{1.5}$$

$$D \xrightarrow{\text{Complement}} D^c \in \text{DFA} \tag{1.6}$$

$$D^c \xrightarrow{\text{Minimisation, (JM) algorithm 2.40}} M \in \text{DFA} \tag{1.7}$$

Explanation of procedure In step (1.1) we use the known operation, of composition from a regular expression to an NFA by using Kleenes theorem. In (1.2) and (1.5) we use the constructive Theorem 3.18, which states that for any NFA there exsists a DFA, accepting the same language. In (1.3) and (1.6) we wish to compute the DFA that accepts complement language, this is simply achieved by

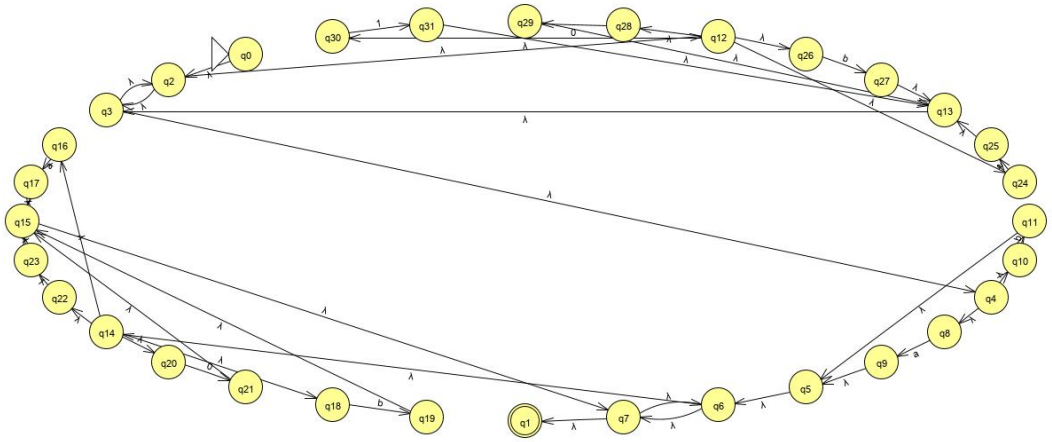
$$\begin{aligned}M &= (Q, \Sigma, q_0, A, \delta) \Rightarrow \\ M^c &= (Q, \Sigma, q_0, Q \setminus A, \delta)\end{aligned}$$

Note however that this procedure fails on incomplete DFA, and as such these steps should be interpreted as fixing incompleteness before finding the complement. In (1.4) we union all the DFA's using a Λ-step, as seen in lecture 2 slide 27, thereby not using the product automaton, which we cannot do in JFLAP and get a NFA. In (1.7) we apply algorithm (JM) 2.40 to achieve the minimized DFA we seek.

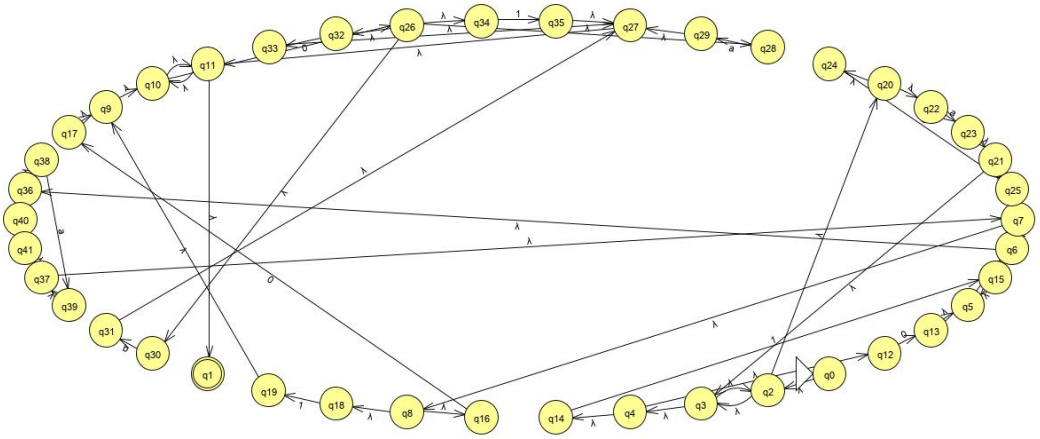
2

Step (1.1)

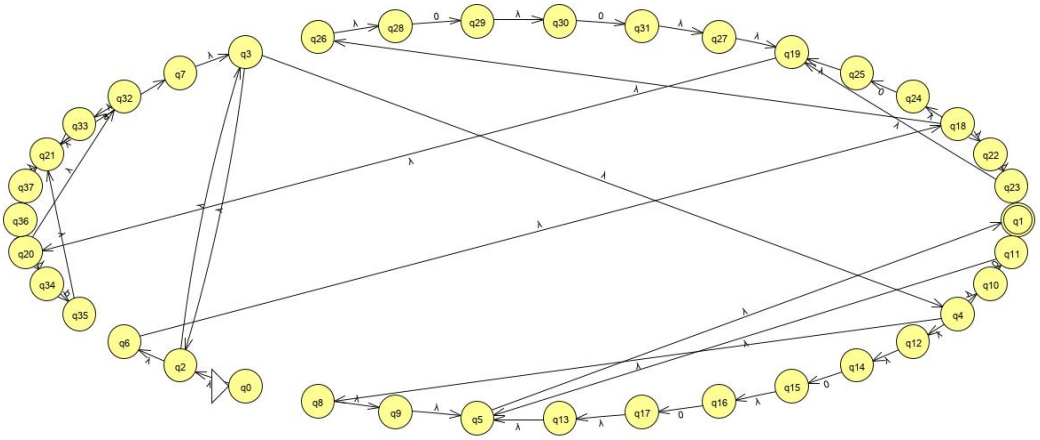
Expression 1: At least one letter: $(a+b+0+1)^*(a+b)(a+b+0+1)^*$ as a NFA



Expression 2:At least two digits $(a+b)^*(0+1)(a+b)^*(0+1)(a+b+0+1)^*$ as a NFA

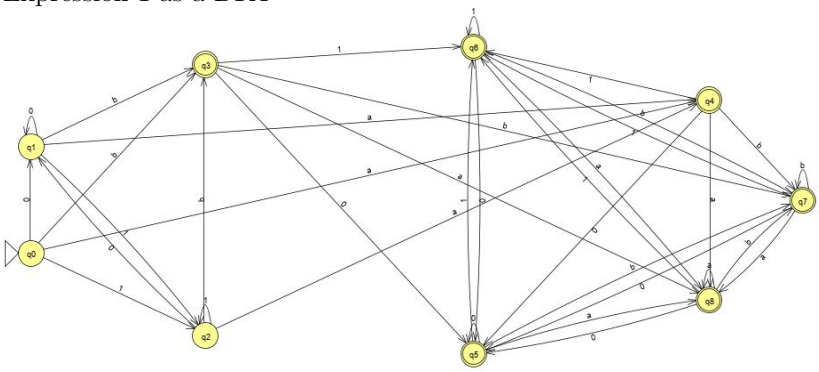


Expression 3: No three 0's in a row: $((\Lambda+0+00)(a+b+1))^*(\Lambda+0+00)$ as a NFA

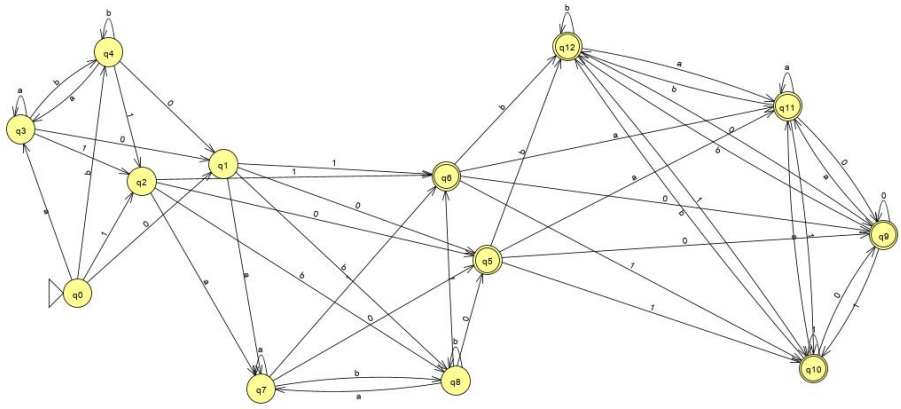


Step (1.2)

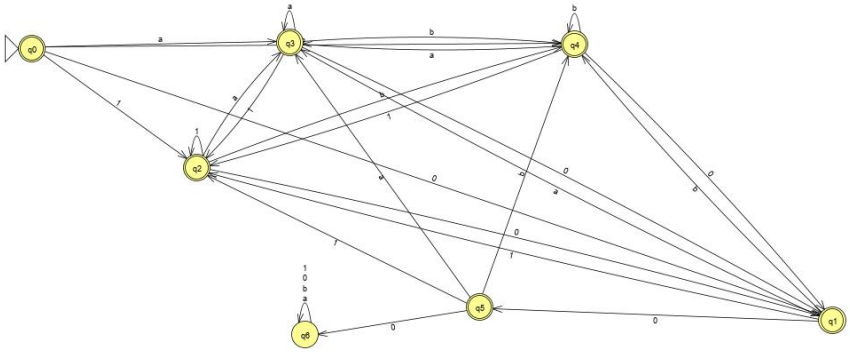
Expression 1 as a DFA



Expression 2 as a DFA

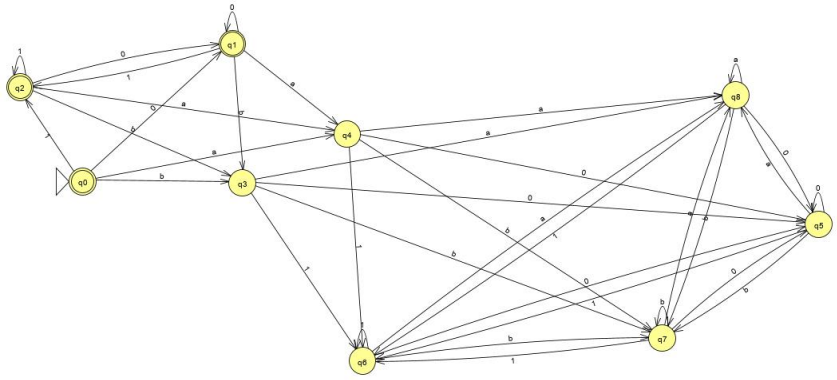


Expression 3 as a DFA, with trap state

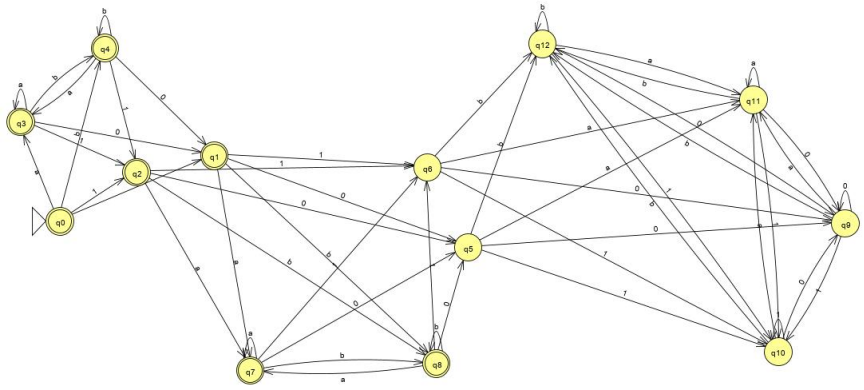


Step (1.3)

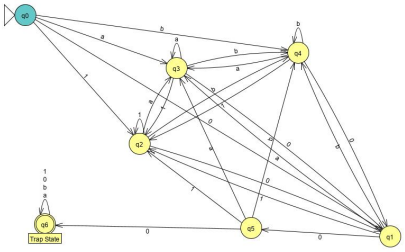
Expression 1 DFA complement



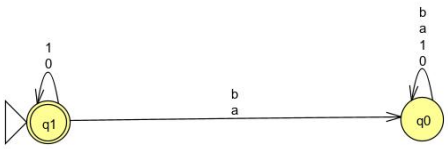
Expression 2 DFA complement



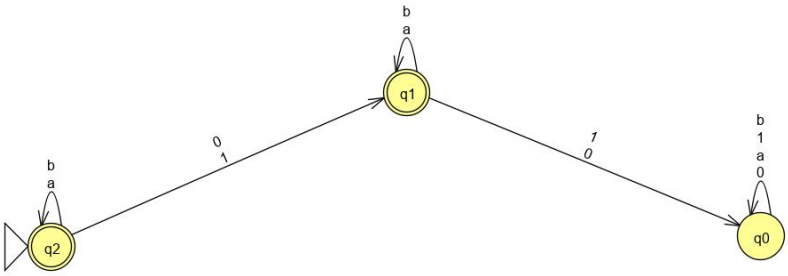
Expression 3 DFA complement



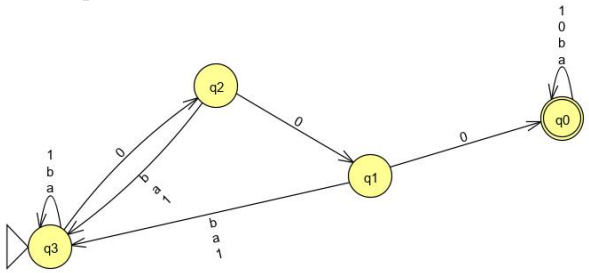
It is unneccesary to minimize at this stage, but for the sake of readability we have chosen to do so.
Expression 1 DFA complement minimized



Expression 2 DFA complement minimized

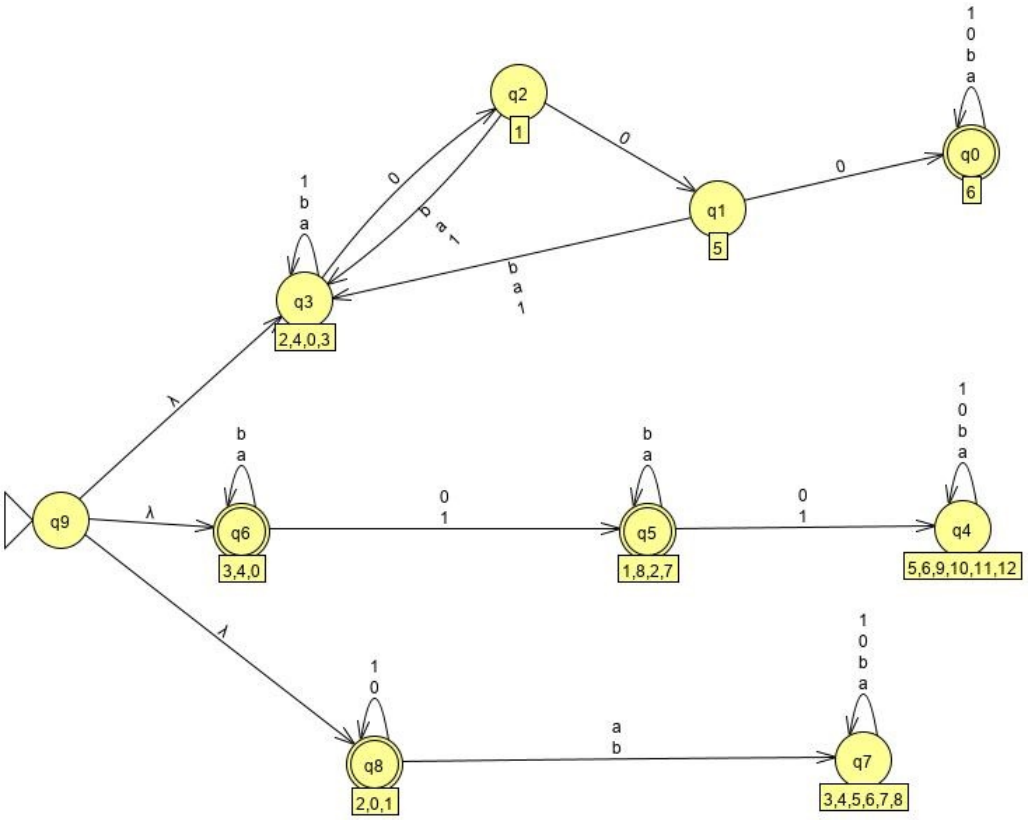


Expression 3 DFA complement minimized



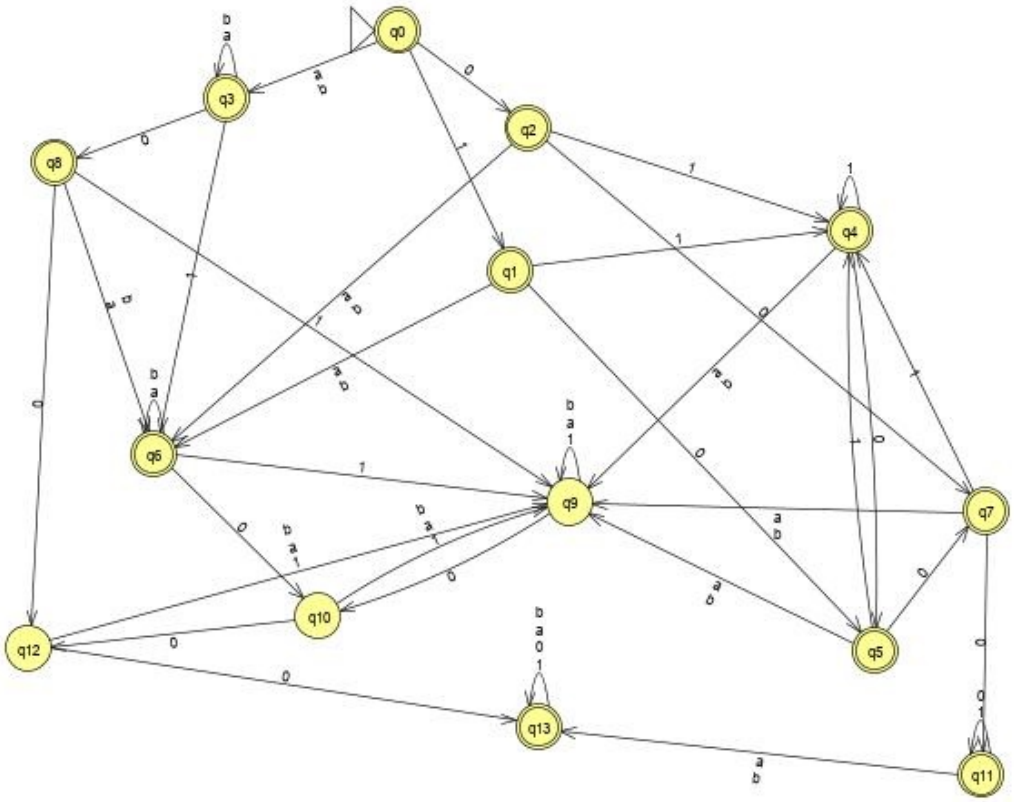
Step (1.4)

j Combination of the minimized DFA's via Λ step, new state 9 added.



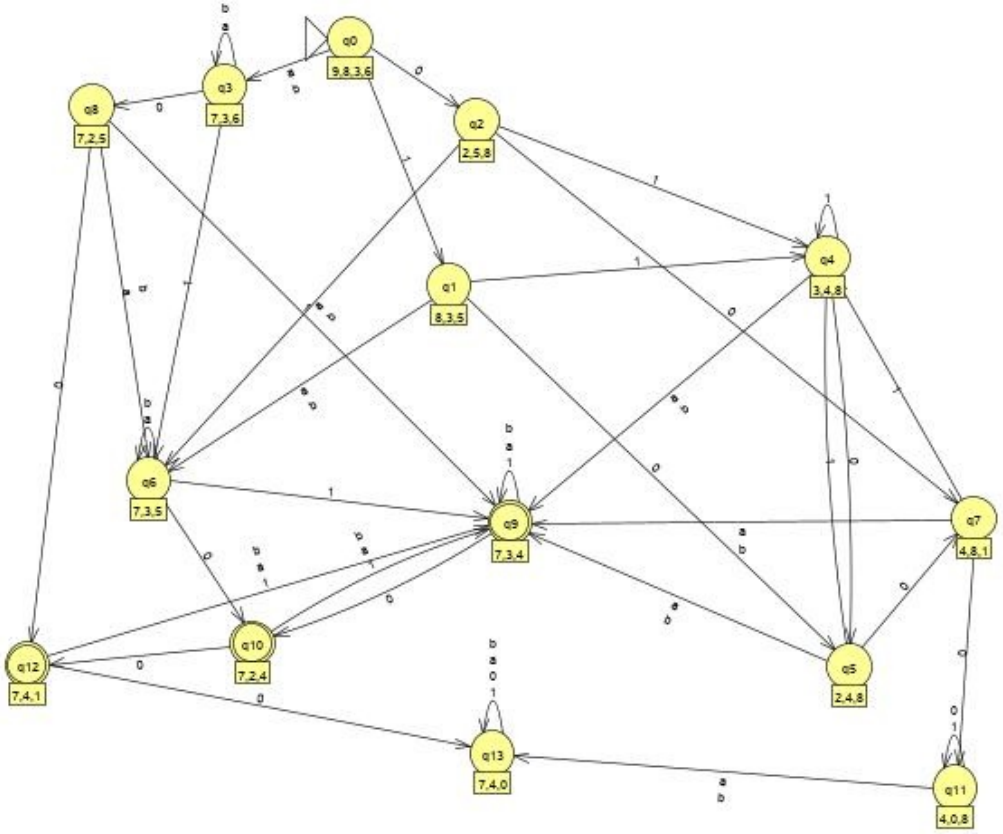
Step (1.5)

After converting the NFA of step (1.4) to a DFA



Step (1.6)

After taking the complement of the DFA:



Step (1.7) the final result

Minimizing the DFA, note that it was already minimal