Homework 3 Solutions

CS 321 - Fall 2006

3.1.4

Find a regular expression for $L = \{a^n b^m : n \geq 3, m \text{ is even}\}$.

$$r = aaaa^*(bb)^*$$

3.1.5

Find a regular expression for $L = \{a^n b^m : (n + m) \text{ is even}\}$.

Use the fact that if $n + m$ is even then either: 1) both n and m are even, or 2) both n and m are odd.

$$r = (aa)^*(bb)^* + (aa)^*ab(bb)^*$$

3.1.8

The description in English is "strings with exactly one b and an even number of a's". We can describe it in set notation as follows:

$$L = \{a^m b a^n : (m + n) \text{ is even}\}$$

3.1.15

Find a regular expression for

$$L = \{w : w \in \{0, 1\}^*, w \text{ has exactly one pair of consecutive zeros}\}$$

Note that a sequence of three zeros is not allowed since that counts as two consecutive pairs of zeros. The following expression describes L.

$$r = (1 + 01)^*00(1 + 10)^*$$

To understand it note that $r_1 = (1+01)^*$ describes the language of strings without any consecutive zeros and that end with a one. Likewise $r_2(1+10)^*$ describes strings without consecutive zeros and that begin with a one. By putting r_1 before 00 we are ensured that the string before does not end with a zero and hence will not lead to a triple of zeros. For this same reason we are guaranteed that 00 followed by r_2 will not generate a triple of zeros.

3.1.17

a) $r = (0+1)^*01$
b) $r = (0+1)^*(11+10+00)+0+1$
d) $r = (0+1)^*00(0+1)^*00(0+1)^* + (0+1)^*000(0+1)^*$

e) Let r_{00} be the expression from exercise 3.1.15 that represents the language with exactly one consecutive pair of zeros. We will construct a new regular expression by considering four cases: 1) there are no consecutive zeros, 2) there is exactly one pair of consecutive zeros, 3) there are two disjoint pairs of consecutive zeros, 4) there is a triple of consecutive zeros and no other consecutive pairs.

- The expression for the first case is: $r_{\overline{00}} = (0+\lambda)(1+10)^*$
- The expression for the second case is given by r_{00}.
- The expression for the third case is given by $r_{00,00} = r_{00}1r_{00}$.
- The expression for the fourth case is given by $r_{000} = (1+01)^*000(1+10)^*$.

Putting everything together we get the final expression to be:

$$r = r_{\overline{00}} + r_{00} + r_{00,00} + r_{000}$$

3.1.18b

$L = \{w : w \in \{a,b\}^*, n_a(w) \bmod 3 = 0\}$

To construct an expression for this language notice that any such string can be broken up into substrings where each substring has exactly 3 a's. The expression for a string with exactly three a's is $r_{3a} = b^*ab^*ab^*ab^*$ from which we can construct a expression for L:

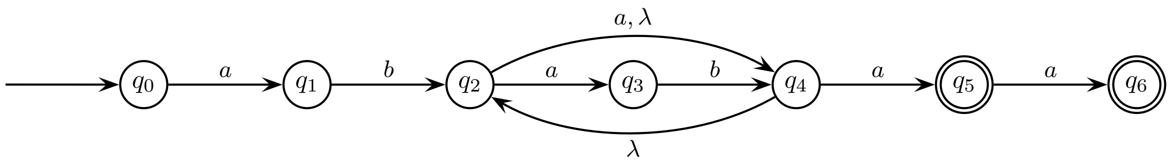
$$r = r_{3a}^* = (b^*ab^*ab^*ab^*)^*$$

3.2.4b

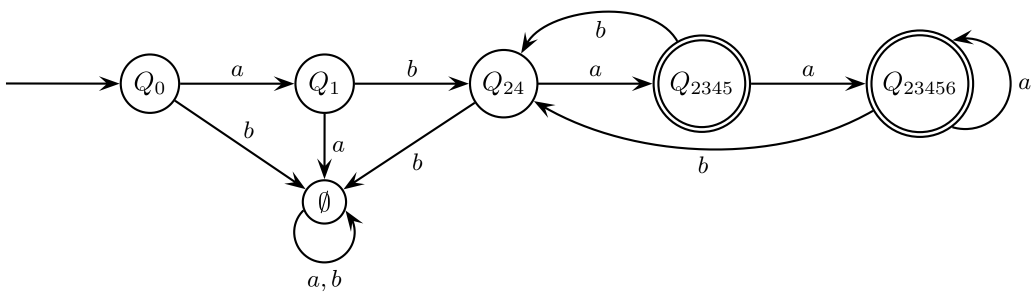
Construct a DFA for $L(ab(a+ab)^*(a+aa))$.

We will do this by first constructing an NFA for the language shown below, and then converting it to an equivalent DFA.

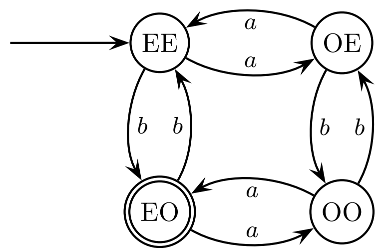
DFA:



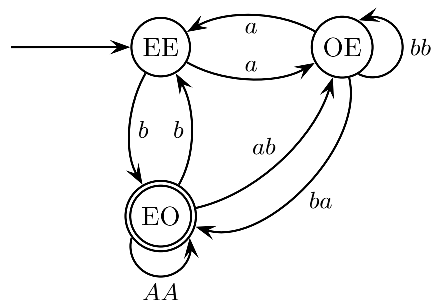
Equivalent NFA:



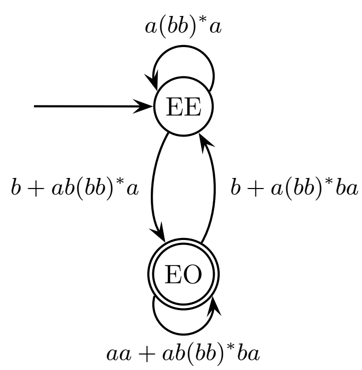
3.2.11



Remove node OO.



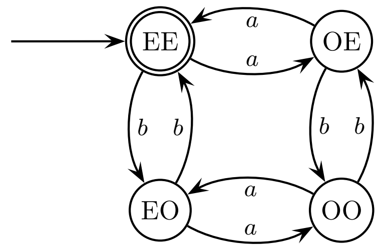
Remove node OE.



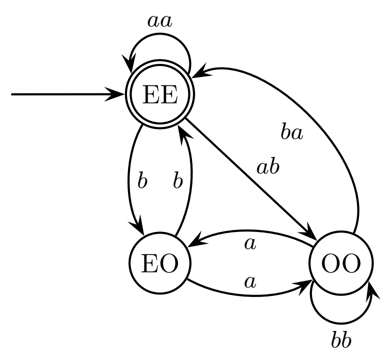
The final regular expression is $r_1^* r_2 r_3^* (r_4 r_1^* r_2 r_3^*)^*$, where

$$
\begin{aligned}
r1 &= a(bb)^*a \\
r2 &= b + a(bb)^*ba \\
r3 &= aa + ab(bb)^*ba \\
r4 &= b + ab(bb)^*a
\end{aligned}
$$

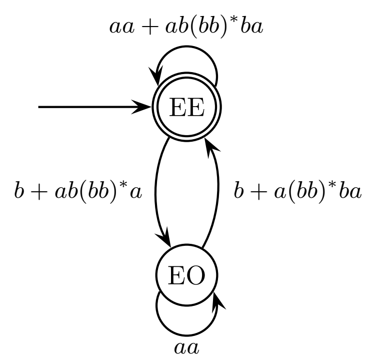
3.2.13a



Remove node OE.



Remove OO.



The final regular expression is $(r_1 + r_2 r_3^* r_4)^*$, where

$$\begin{aligned} r1 &= aa + ab(bb)^*ba \\ r2 &= b + ab(bb)^*a \\ r3 &= aa \\ r4 &= b + a(bb)^*ba \end{aligned}$$

18) $Drop(L) = \{uv : uav \in L \text{ for } a \in \Sigma\}$

Let M be a DFA for L.
We will construct an NFA N for $Drop(L)$.

N will contain two copies of M which we will call M_1 and M_2.

The initial state of N is the initial state of M_1.

For every state q of M_1 and $a \in \Sigma$ add a λ-transition from q to the state in M_2 corresponding to $\delta(q, a)$ in M.

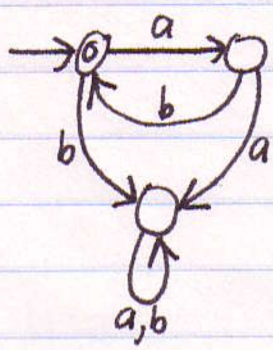
The final states of N are the final states of M_2.

The idea here is that we effectively ~~skip an input~~ avoid the need to read an input symbol by transitioning to M_2.

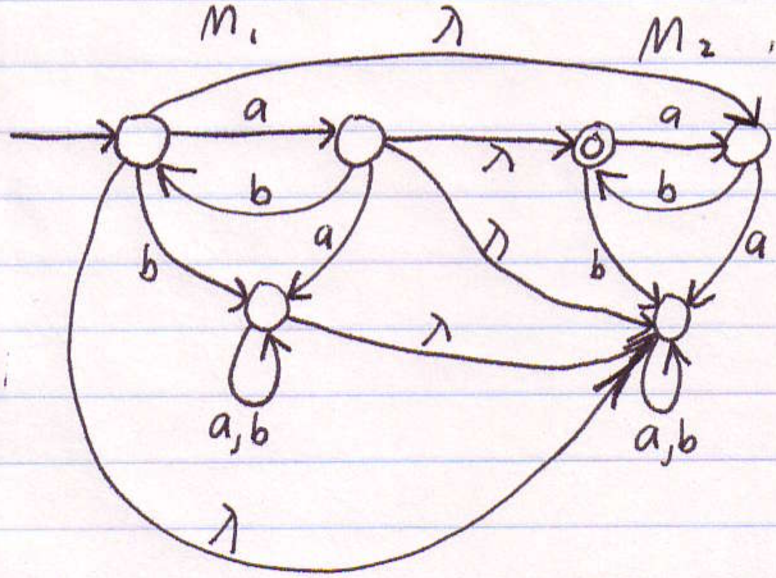
Mead

Here is a simple example for $L = (ab)^*$.

M is



N is



Notice that this NFA will accept abb or aab, ect.