**German University in Cairo**
**Department of Computer Science**
**Dr. Haythem O. Ismail**

**Theory of Computation**, Winter Term 2013
**Assignment5**
Discussion: 2.11.13 - 9.11.13

### Exercise 5-1
**Reading**

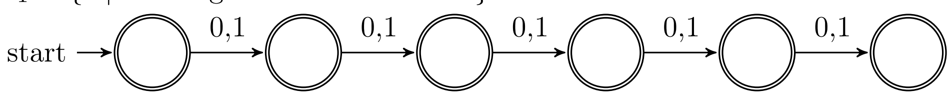
- Read pages 60 through 76 of the text.

### Exercise 5-2
**Exercises from Textbook**

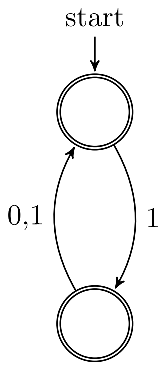
**Sipser** (*pp* 85 – 90): Solve exercises 1.9, 1.10, 1.12 1.18[1], 1.20[2], 1.21.

**Solution:**

1.9 a) $L_1 = \{w|$ the length of $w$ is at most $5\}$

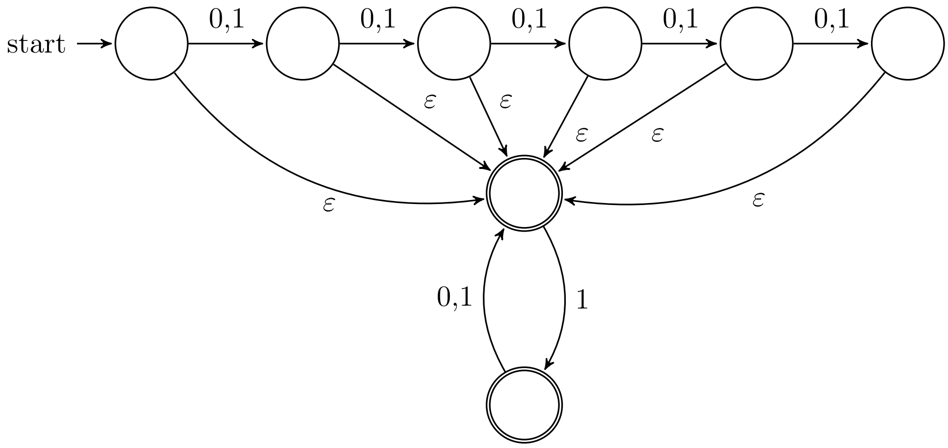



$L_2 = \{w|$ every odd position of $w$ is a $\mathtt{1}\}$

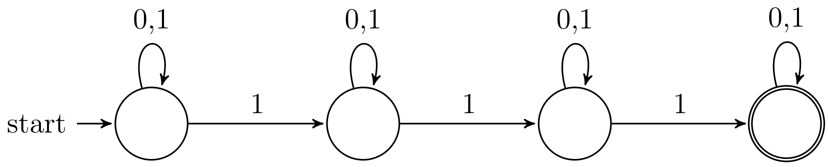



$L = L_1 \circ L_2$

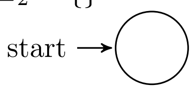
[1] 1.19 in US edition
[2] 1.18 in US edition



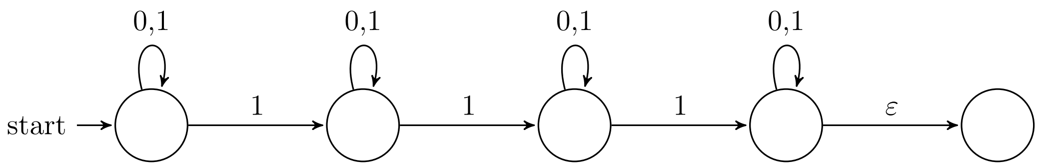

b) $L_1 = \{w | w \text{ contains at least three } \texttt{1}\text{s}\}$



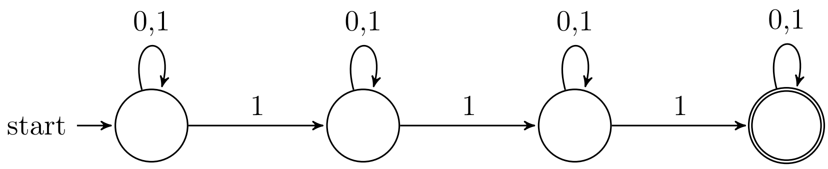

$L_2 = \{\}$



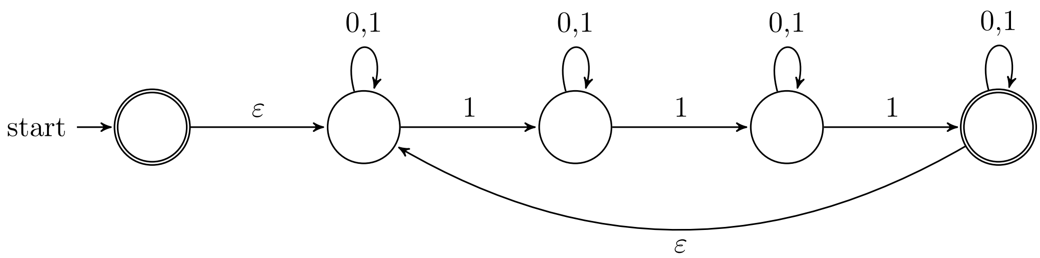

$L = L_1 \circ L_2$



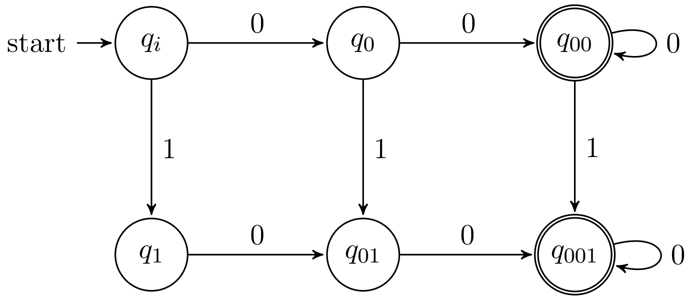

1.10 a) $L = \{w | w \text{ contains at least three } \texttt{1}\text{s}\}$



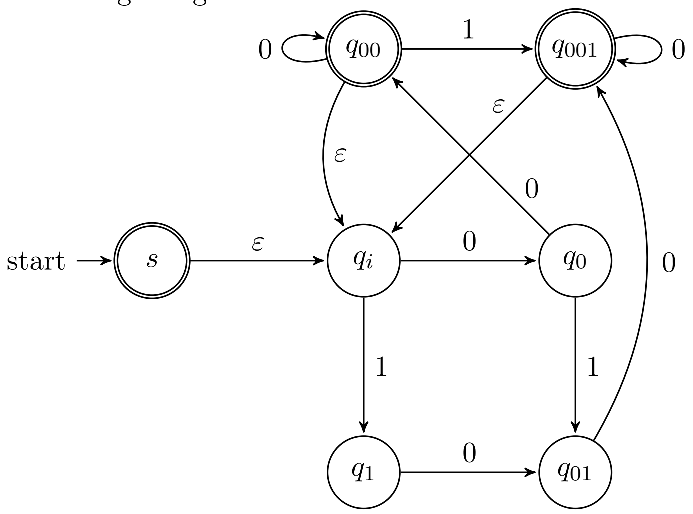

NFA recognizing $L^*$



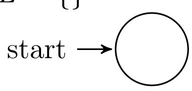

b) $L = \{w | \ w \text{ contains at least two } \texttt{0}\text{s and at most one } \texttt{1}\}$



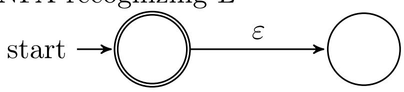

NFA recognizing $L^*$



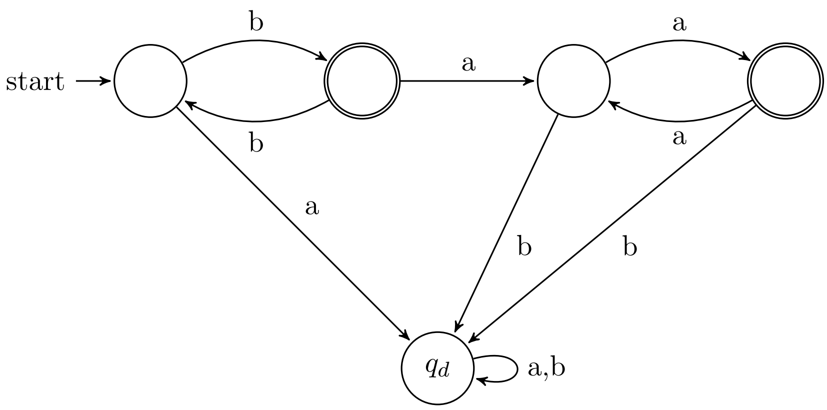

c) $L = \{\}$




NFA recognizing $L^*$




1.12 We can try to put this in a simpler formulation. $L = \{w|w$ contains an odd number of `b`s followed by an even number of `a`s$\}$.

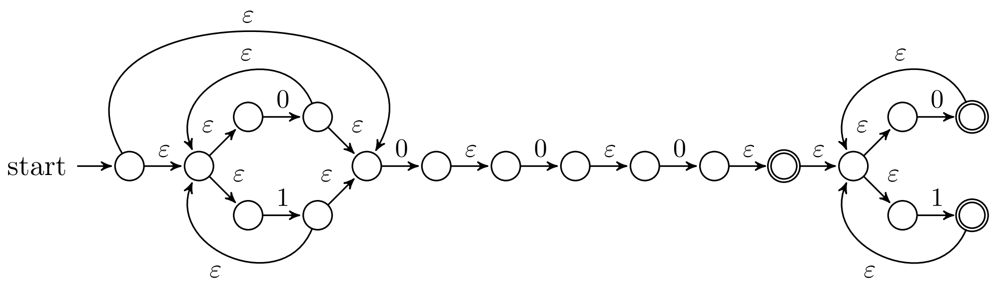



This language, $L$, can be considered as teh concationation of two simpler languages $L = L_1 \circ L_2$, where $L_1$ contains all the strings of `b`s only with odd length and $L_2$ is the set of all strings of `a`s of even length. The regular expression describing $L$ is the result of the concatination of the regular expression of $L_1$ and $L_2$.

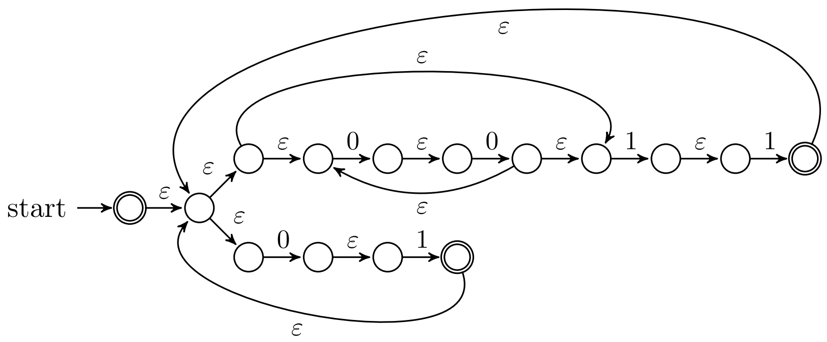
$R_{L_1} = b(bb)^*$

$R_{L_2} = (aa)^*$
$R_L = R_{L_1} \circ R_{L_2} = b(bb)^*(aa)^*$

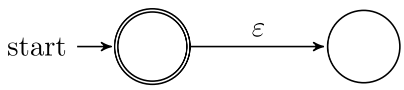
1.18 a)




b)




c)




1.20 a) $1(0 \cup 1)^*0 = 1\Sigma^*0$

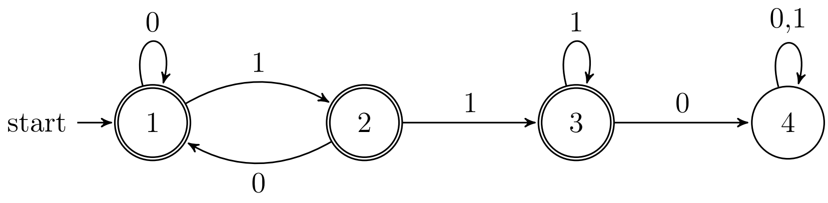
b) $0^*10^*10^*1(0 \cup 1)^* = 0^*10^*10^*1\Sigma^*$

c) $\Sigma^*0101\Sigma^*$

d) $\Sigma\Sigma0\Sigma^*$

e) $(0(\Sigma\Sigma)^*) \cup (1\Sigma(\Sigma\Sigma)^*)$

f) $(0 \cup 10)^*1^*$
For this regular expression, keep in mind the following DFA:




g) $\varepsilon \cup \Sigma \cup \Sigma\Sigma \cup \Sigma\Sigma\Sigma \cup \Sigma\Sigma\Sigma\Sigma \cup \Sigma\Sigma\Sigma\Sigma\Sigma$

h) $0^* \cup 1 \cup 1111^+ \cup \Sigma^*0\Sigma^*$

i) $(1\Sigma)^* \cup (1\Sigma)^*1 \equiv (1\Sigma)^*(\varepsilon \cup 1)$

j) $00^+ \cup 00^+10^* \cup 0^*100^+ \cup 0^+10^+$

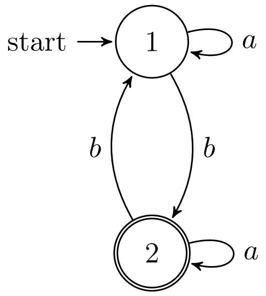
k) $\varepsilon \cup 0$

l) $(00)^* \cup 0^*10^*10^*$

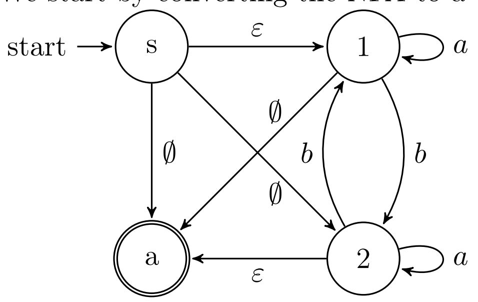
m) $\emptyset$

n) $(0 \cup 1)^+ \equiv \Sigma^+$

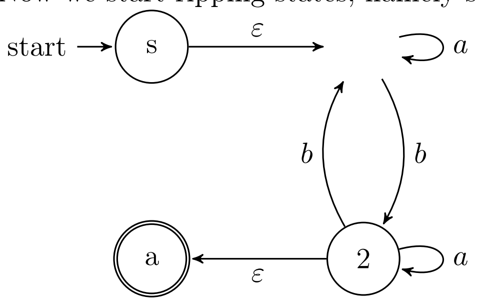
1.21 a)

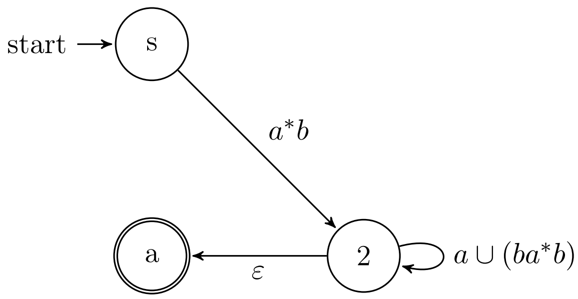



We start by converting the NFA to a GNFA:

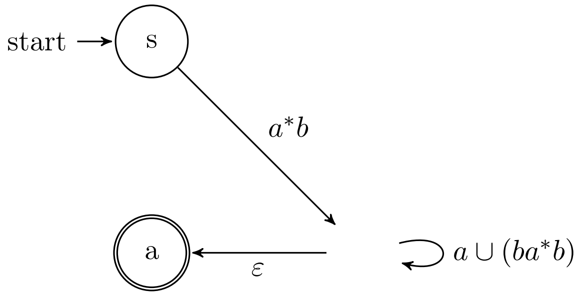


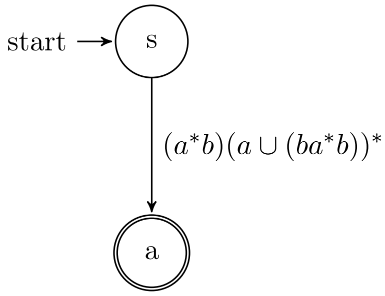

Now we start ripping states, namely starting with state 1:



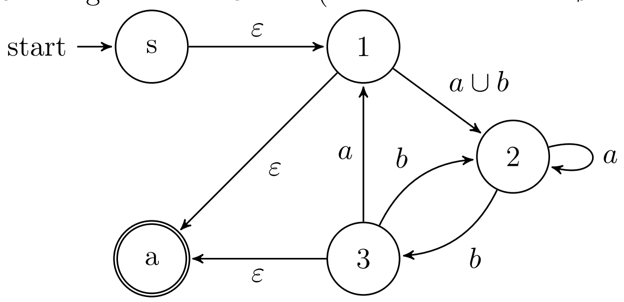

Now we can rip state 2:



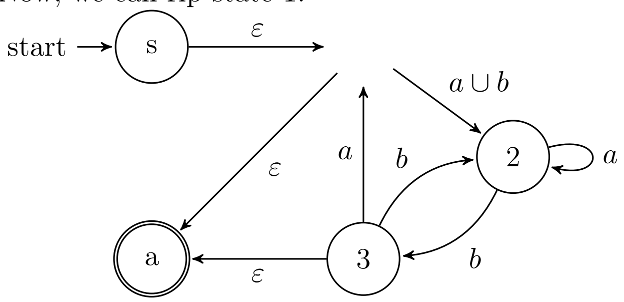



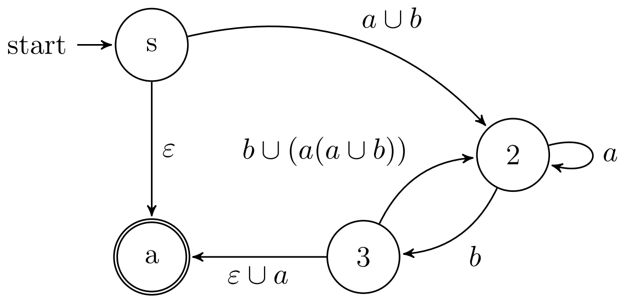

Thus, the regular expression given by this DFA is $(a^*b)(a \cup (ba^*b))^*$

b) Starting with the GNFA (we will remove the $\emptyset$-transitions for clarity):

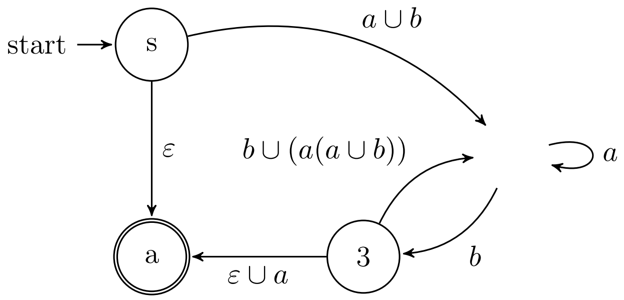


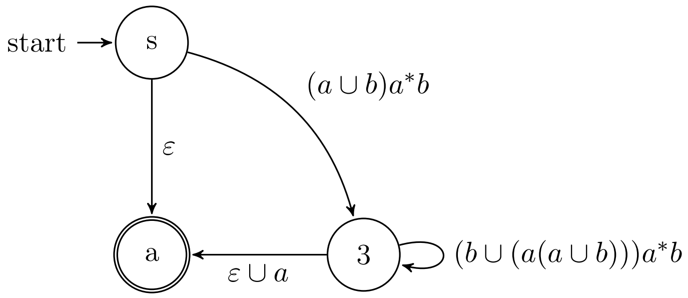

Now, we can rip state 1:



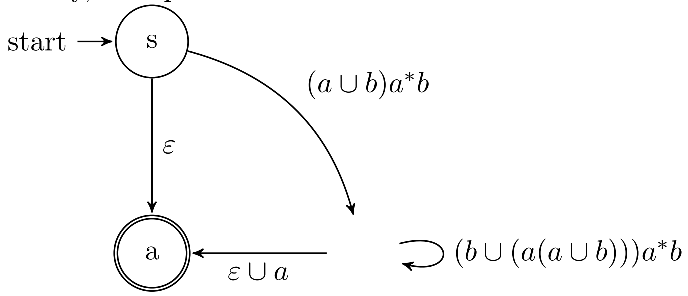



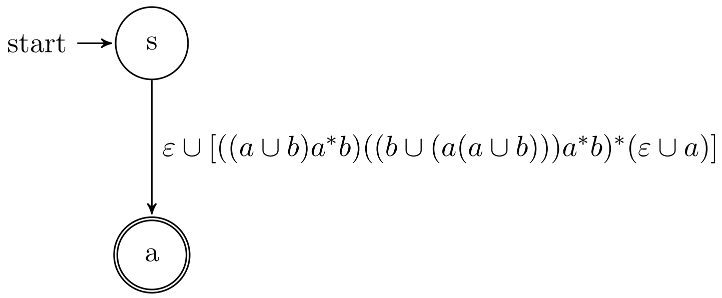

The following state to be rupped will be state 2:




Finally, we rip state 3:




The regular expression given by this DFA is: $\varepsilon \cup [((a \cup b)a^*b)((b \cup (a(a \cup b)))a^*b)^*(\varepsilon \cup a)]$

**Exercise 5-3**

**Programming**

Using your favorite programming language, write a method/function/clause that, given a DFA $M$, constructs an equivalent regular expression. Note that you will need to implement a GNFA abstract data type.