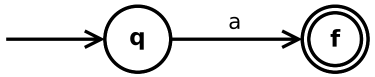
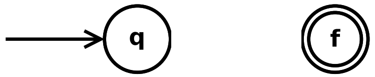
INF1a-CL

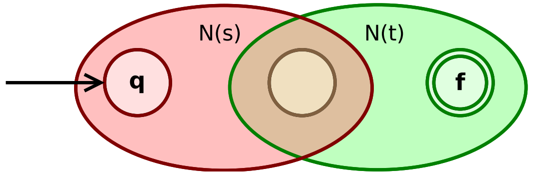
NFA DFA regex

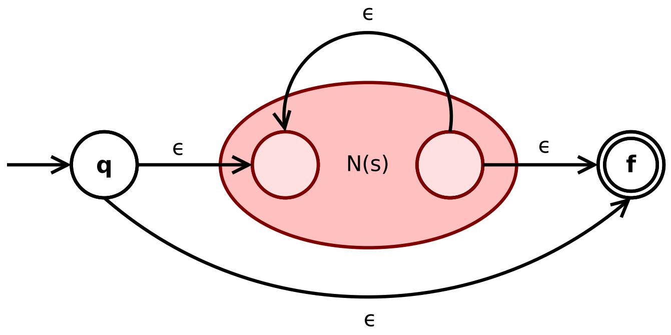
-Tseytin
-Syllogisms
-DPLL
-The arrow rule
-Haskell coding in CL
-Sequent calculus
-Satisfiability and CNF
-Operations on machine languages
-Resolution
-Logic
-Karnaugh maps

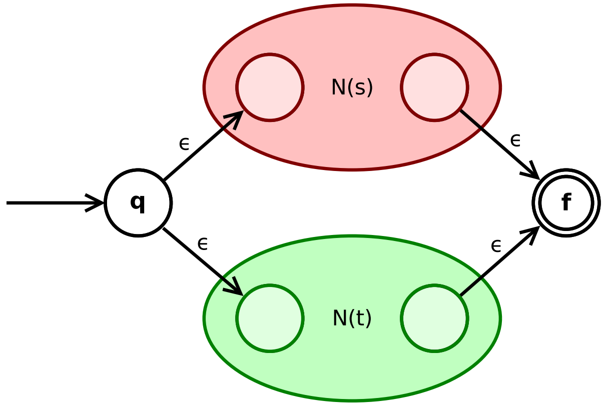
Today regex NFA DFA
Monday Syllogisms Arrow Rule KM
Thursday Sequent Calculus CNF Tseytin
Friday DPLL Satisfiability

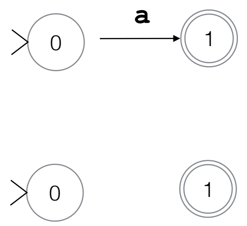

q
a
f

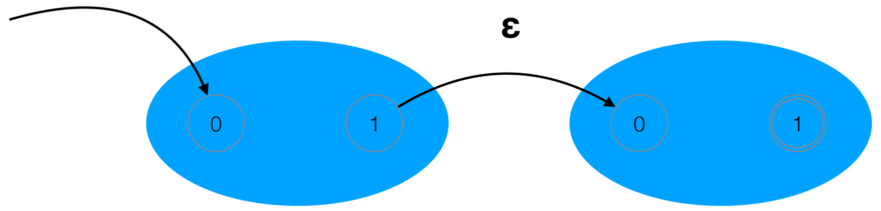

q
f

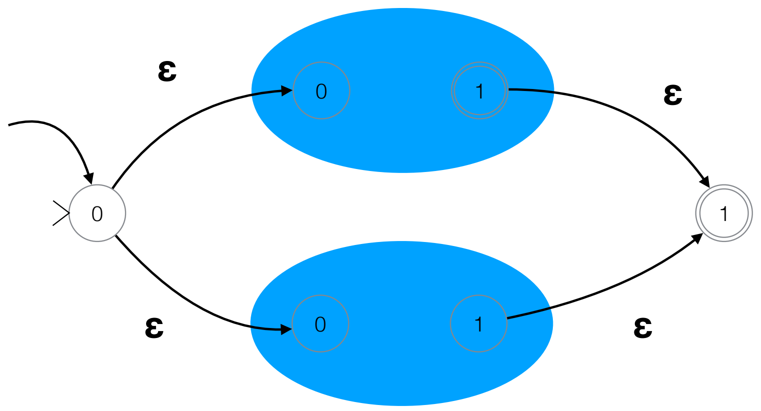

N(s)
N(t)
q
f

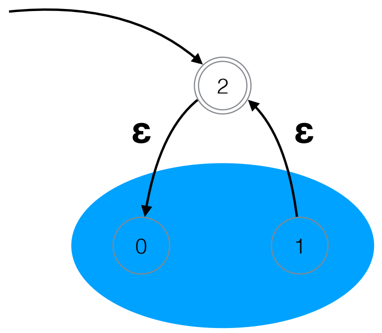

ϵ
q
ϵ
N(s)
ϵ
f
ϵ


N(s)
ϵ
ϵ
q
f
ϵ
ϵ
N(t)


0
a
1
0
1


0
1
ε
0
1


ε
0
1
ε
0
1
ε
0
1
ε

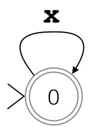

2
ε
ε
0
1

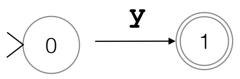
(d) For each of the following regular expressions, draw a non-deterministic finite state machine that accepts the language described by the regular expression.

i. x^*y

ii. $(x^*|y)$

iii. $(x^*y)^*$



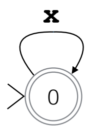


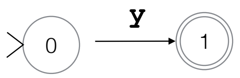
(d) For each of the following regular expressions, draw a non-deterministic finite state machine that accepts the language described by the regular expression.

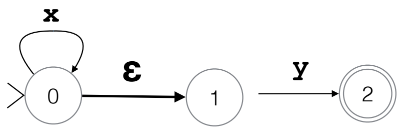
i. x^*y

ii. $(x^*|y)$

iii. $(x^*y)^*$





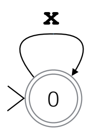


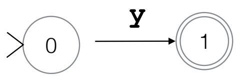
(d) For each of the following regular expressions, draw a non-deterministic finite state machine that accepts the language described by the regular expression.

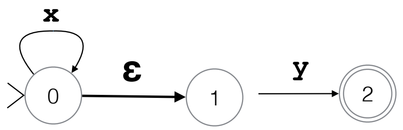
i. x^*y

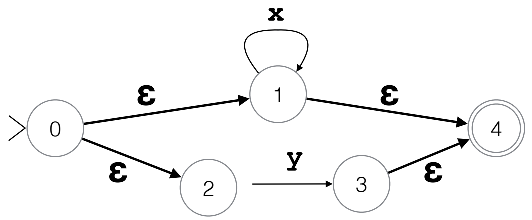
ii. $(x^*|y)$

iii. $(x^*y)^*$







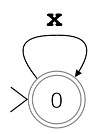


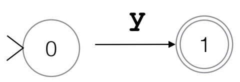
(d) For each of the following regular expressions, draw a non-deterministic finite state machine that accepts the language described by the regular expression.

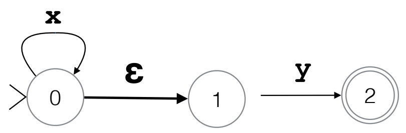
i. x^*y

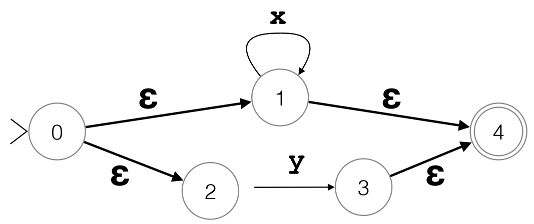
ii. $(x^*|y)$

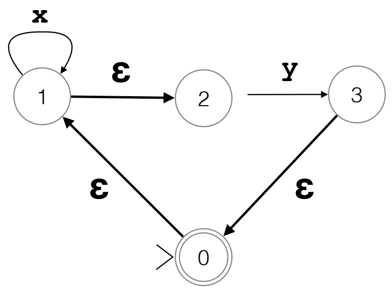
iii. $(x^*y)^*$









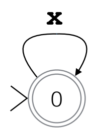


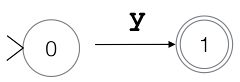
(d) For each of the following regular expressions, draw a non-deterministic finite state machine that accepts the language described by the regular expression.

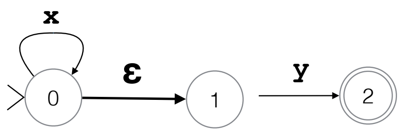
i. x^*y

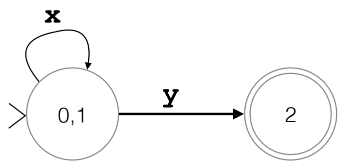
ii. $(x^*|y)$

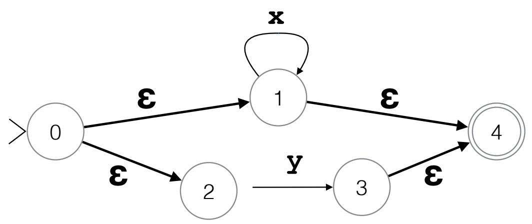
iii. $(x^*y)^*$

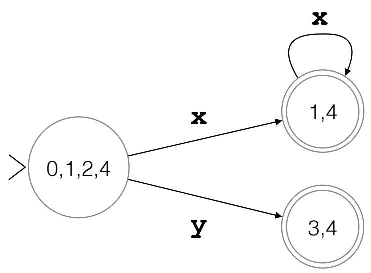


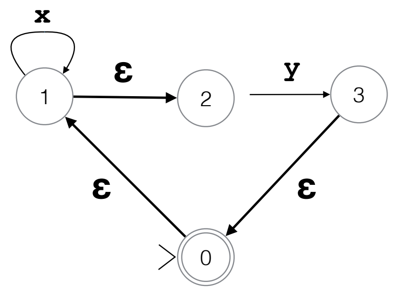


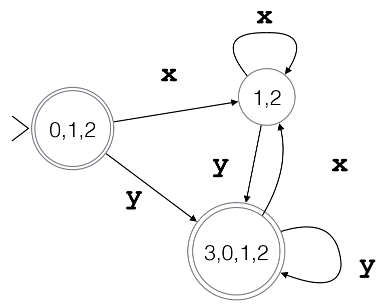




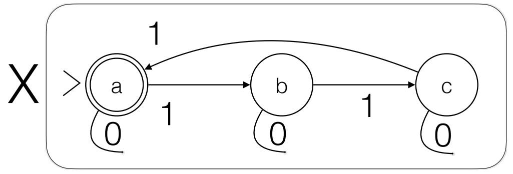


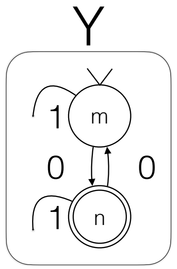






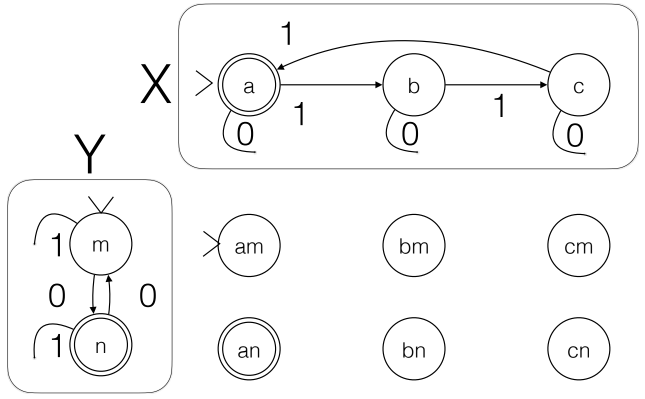
DFA





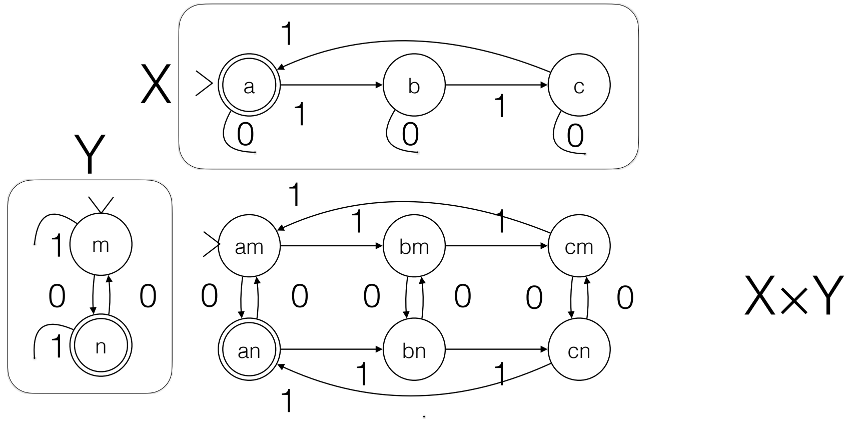
X×Y

DFA



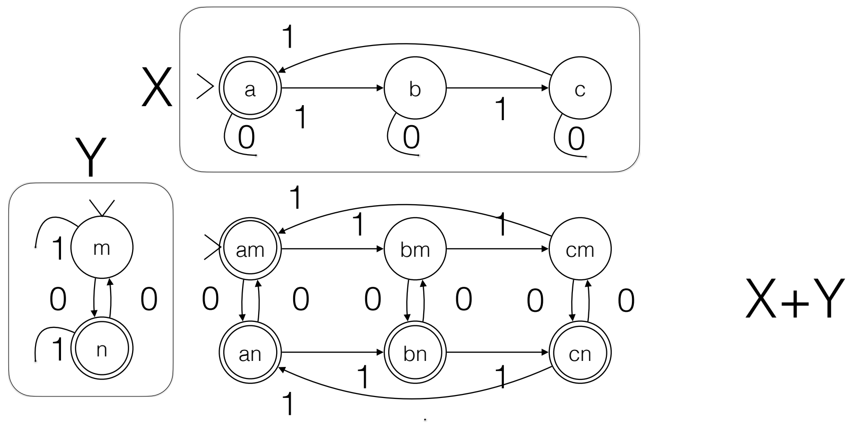
X×Y

DFA

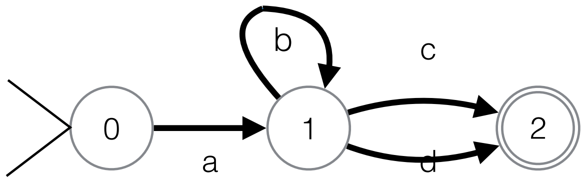


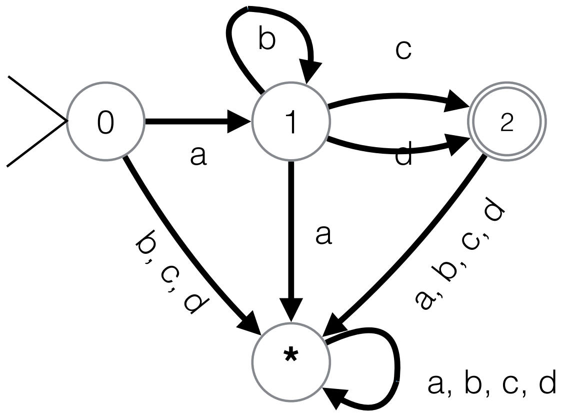
For the product construction we can ignore black hole states in either component

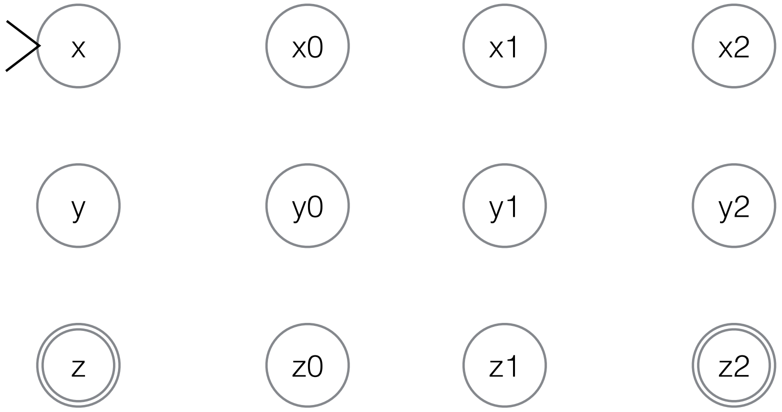
DFA

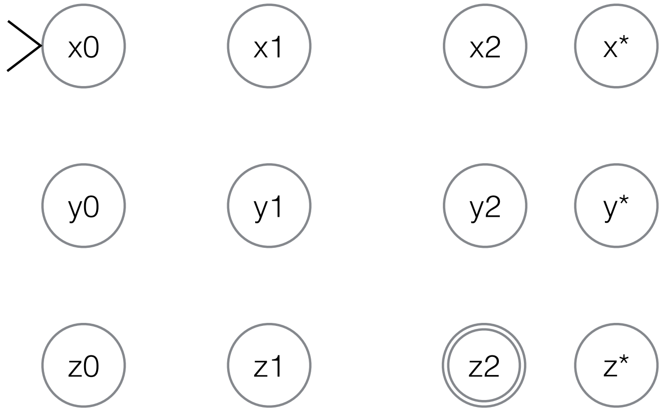


For the sum construction we must include any black hole state in each component

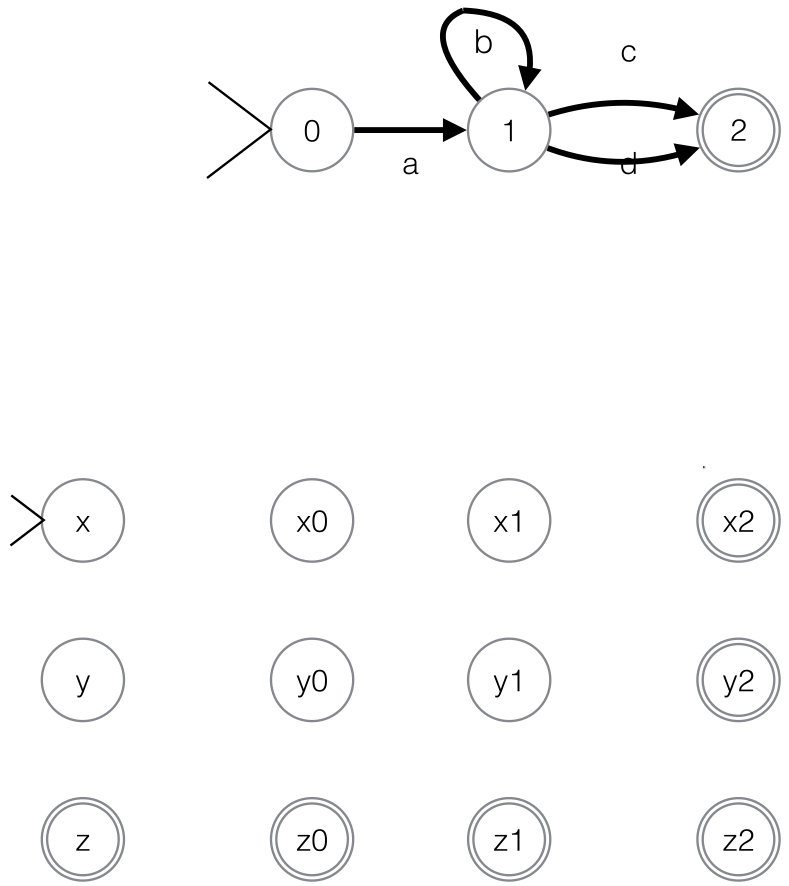


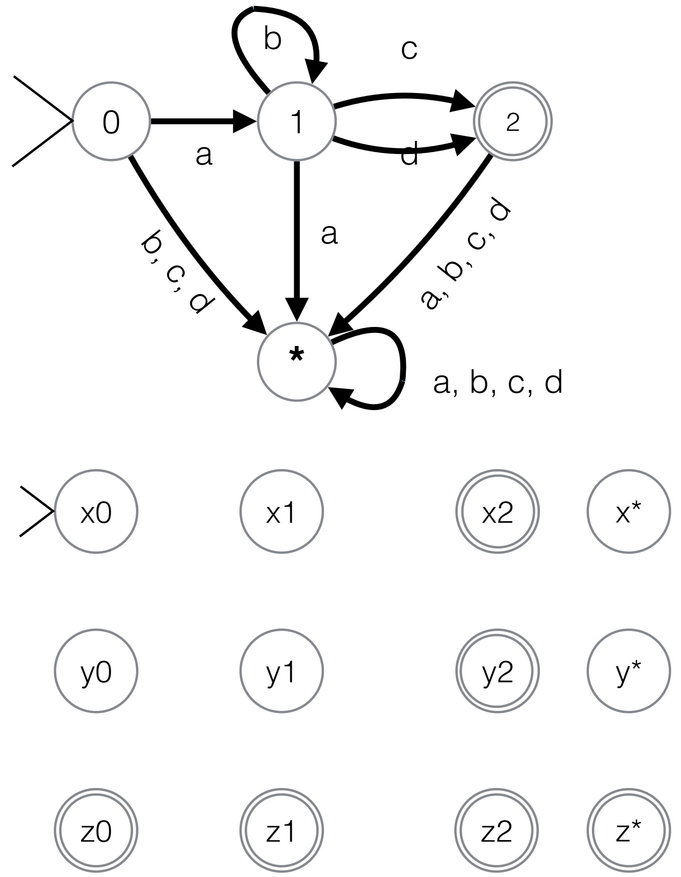






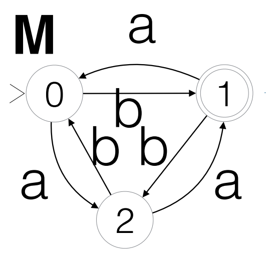
Product : OK to ignore black hole



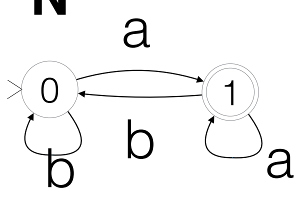


sum : must include black hole

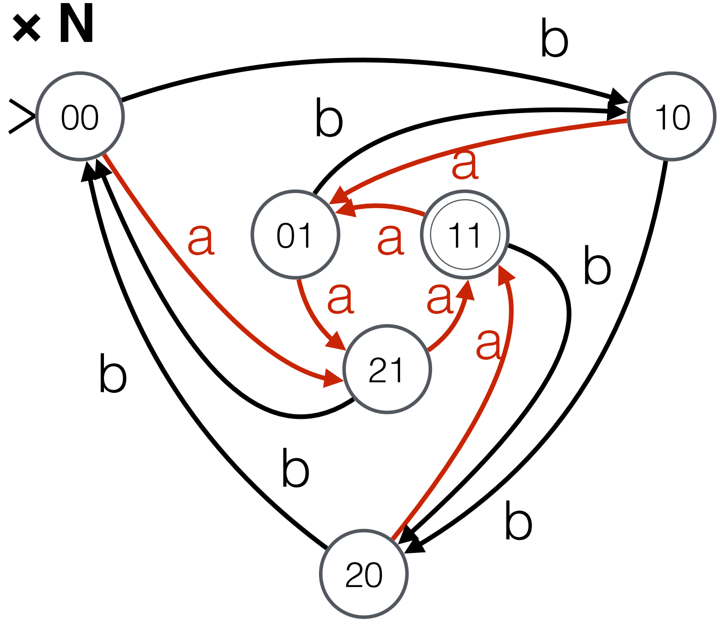
M



N



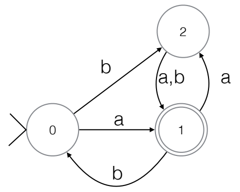
M × N



5. Each diagram shows an FSM. In each case give a regular expression for the language accepted by the FSM, make a mark in the check box against each string that it accepts (and no mark against those strings it does not accept), make a mark in the DFA check box if it is deterministic, and draw an equivalent DFA if it is not.

[4 marks]

(a)

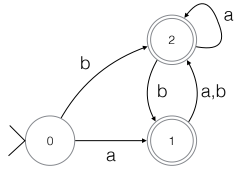


aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐

regex:

[4 marks]

(b)

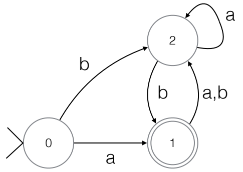


aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐

regex:

[4 marks]

(c)

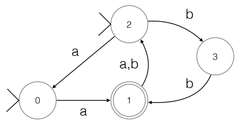


aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐

regex:

[4 marks]

(d)

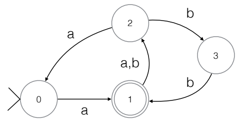


aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐

regex:

[4 marks]

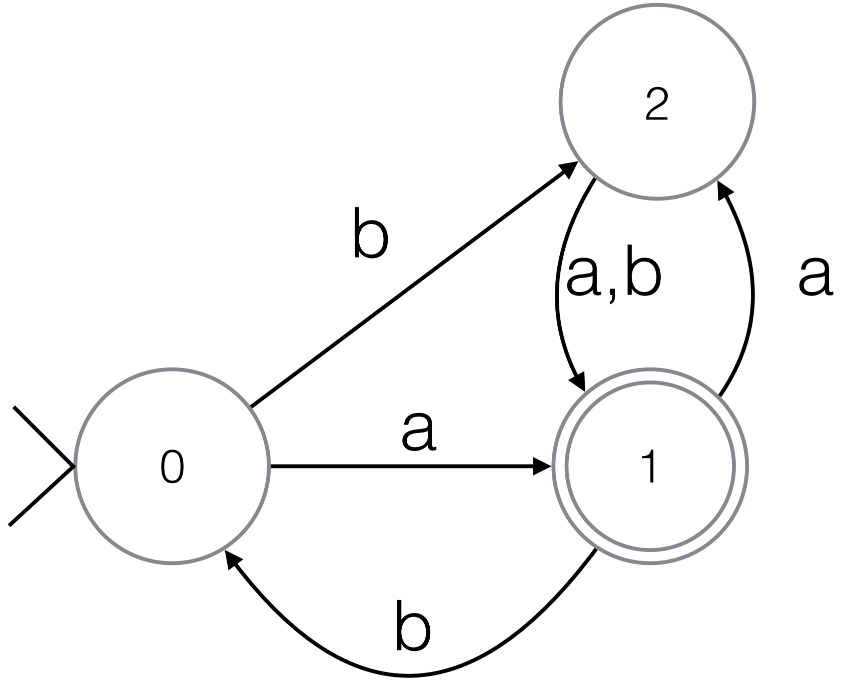
(e)



aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐

regex:

(a)



aab ☐

aba ☐

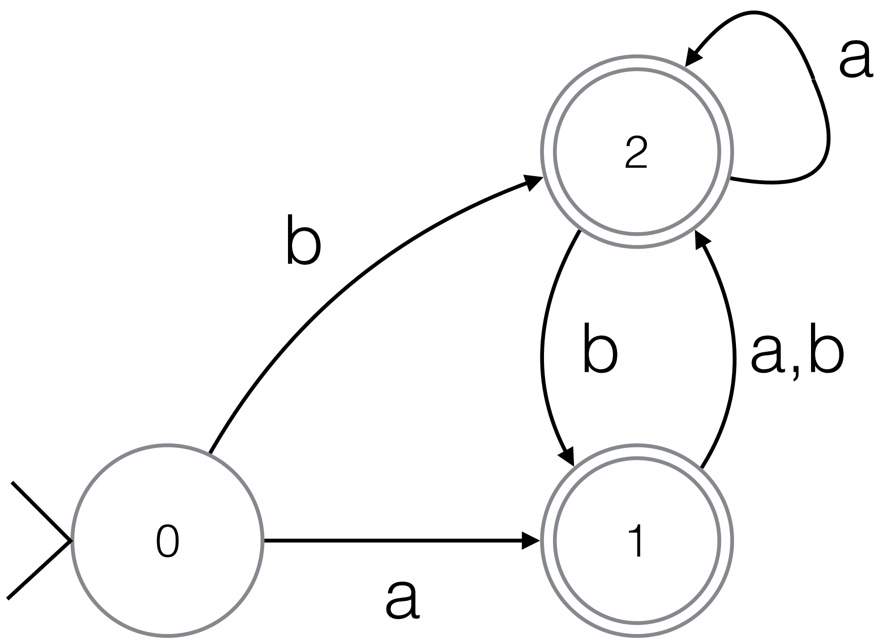
bab ☐

aaa ☐

bbb ☐

DFA ☐

(b)



aab ☐

aba ☐

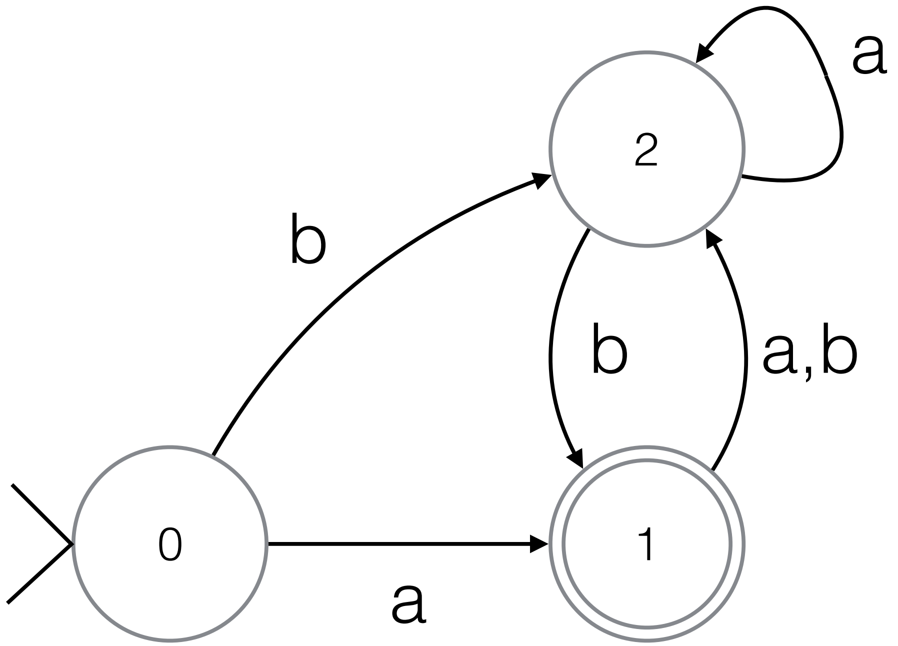
bab ☐

aaa ☐

bbb ☐

DFA ☐

(c)



aab ☐

aba ☐

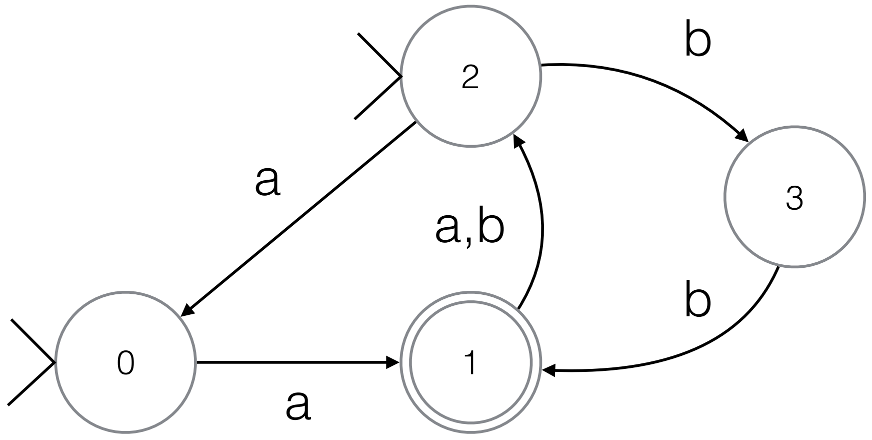
bab ☐

aaa ☐

bbb ☐

DFA ☐

(d)



aab ☐

aba ☐

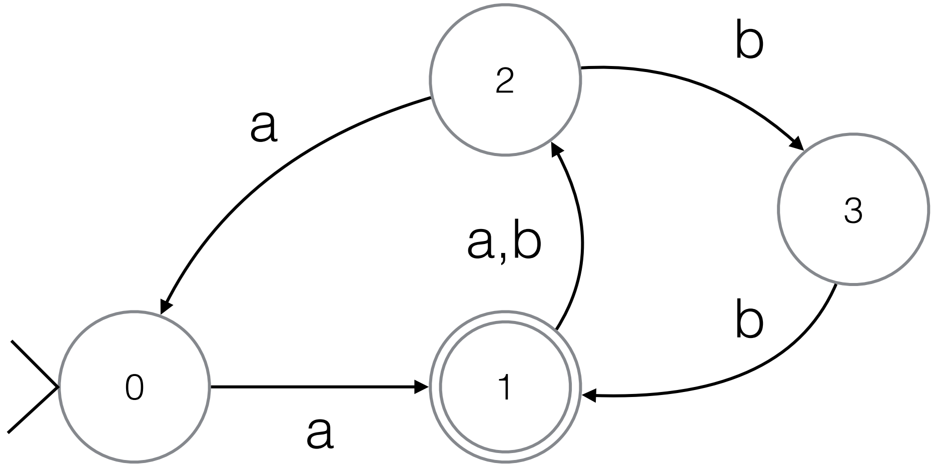
bab ☐

aaa ☐

bbb ☐

DFA ☐

(e)



aab ☐

aba ☐

bab ☐

aaa ☐

bbb ☐

DFA ☐

a) `(a|b(a|b))(a(a|b)|b(a|b(a|b)))*`

b) `(a|ba*b)((a|b)a*b)*|(b|a(a|b))(a|b(a|b))*`

c) `(a|ba*b)((a|b)a*b)*`

d) `a((a|b)(aa|bb))*|(aa|bb)((a|b)(aa|bb))*`

e) `a((a|b)(aa|bb))*`