1. Regular Languages
   1. One is to draw a DFA, and the other one is to find a regular expression.
   2. 0|1(0|1)\*

ϵ

ϵ

ϵ

1

0

1

ϵ

ϵ

11

ϵ

ϵ

ϵ

ϵ

ϵ

4

0

start

* 1. Initial: ϵ-closure(0) = {1, 3}≡A  
     moveto(A, 0) = ϵ-closure({4}) = {4}≡B  
     moveto(A, 1) = ϵ-closure({2}) = {2, 5, 6, 8, 11}≡C  
     moveto(B, 0) = ϵ-closure({}) = {}  
     moveto(B, 1) = ϵ-closure({}) = {}  
     moveto(C, 0) = ϵ-closure(7) = {5, 6, 7, 8, 10, 11}≡D  
     moveto(C, 1) = ϵ-closure(9) = {5, 6, 8, 9, 10, 11}≡E  
     moveto(D, 0) = ϵ-closure(7) = D   
     moveto(D, 1) = ϵ-closure(9) = E   
     moveto(E, 0) = ϵ-closure(7) = D   
     moveto(E, 1) = ϵ-closure(9) = E  
     Transition Table:

C

1

0

D

0

0

B

E

1

1

0

1

|  |  |  |
| --- | --- | --- |
| State | 0 | 1  C  1  0  B  0  1 |
| A | B | C |
| B | -- | -- |
| C | D | E |
| D | D | E |
| E | D | E |

* 1. Split into two groups:  
     {A}≡G0 (Nonaccepting state); {B, C, D, E}≡G1 (Accepting states)  
     B is the final state without outgoing arrow 🡺 {A} {B} {C, D, E}≡G2
  2. (C, 0) 🡪 DG2(C, 1) 🡪 EG2(D, 0) 🡪 DG2(D, 1) 🡪 EG2(E, 0) 🡪 DG2(E, 1) 🡪 EG2

∴ C, D, and E are equivalent.

* 1. L = {0n1n | n > 0}
  2. See L4-page 9 and 10.

By the description, we can draw PDA:

q1

a, z🡪z

z

b, z🡪z

a, z🡪z

b, z🡪z

Each step we pop z and push z, so there is always only one z in the stack.

Therefore, we can just ignore what the top of stack is, and we can ignore the stack.

The DFA is like:

q1

a

b

a

b

1. Out of range