LEX6: NFA to DFA

Lexical Analysis

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

Building a Lexical Analyzer

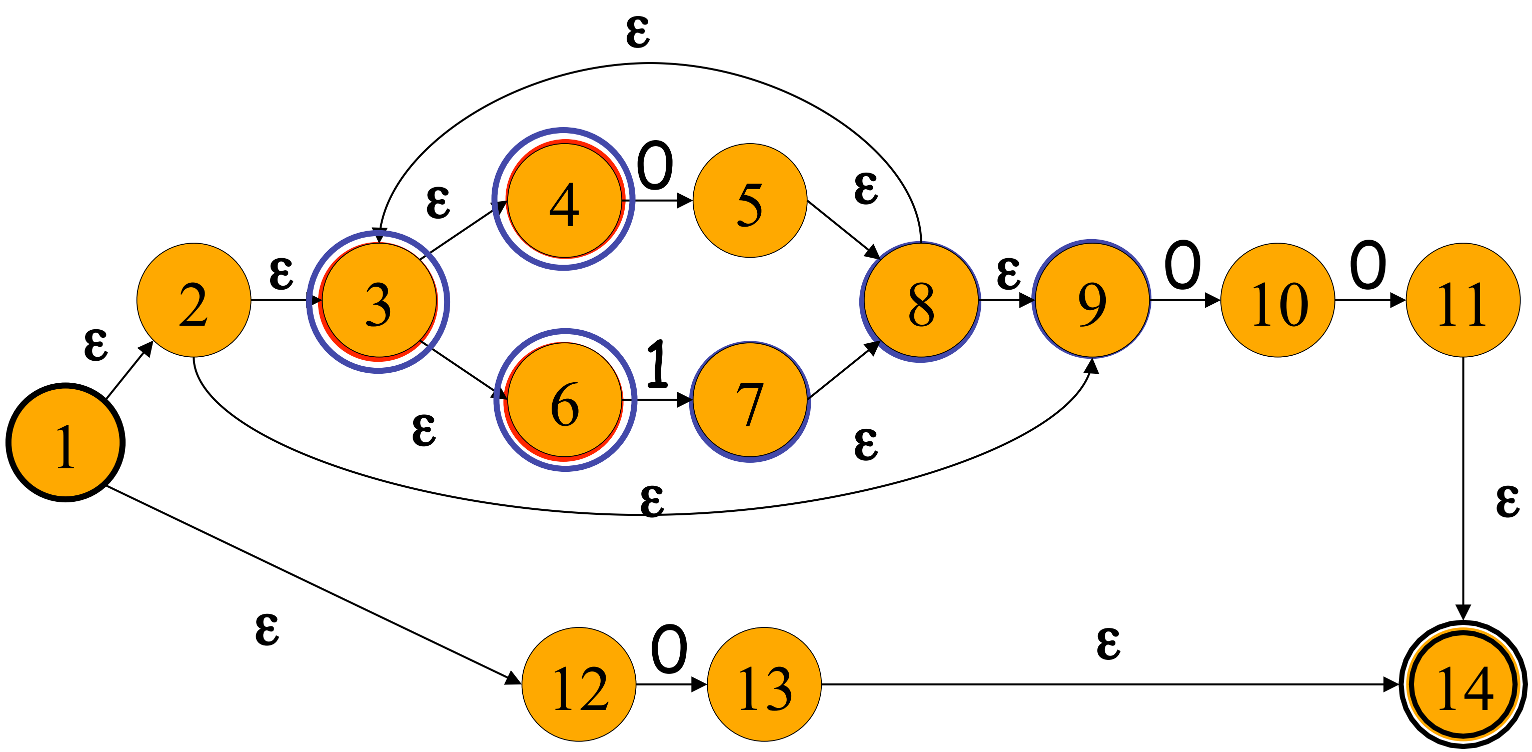
- Token $\Rightarrow$ Pattern
- Pattern $\Rightarrow$ Regular Expression
- Regular Expression $\Rightarrow$ NFA
- NFA $\Rightarrow$ DFA
- DFA $\Rightarrow$ Table-driven implementation of DFA

ε-closure

ε-closure(s)= all states reached by following only ε-transitions

ε-closure(3) = {3,4,6}

ε-closure(7) = {7,8,9,3,4,6}



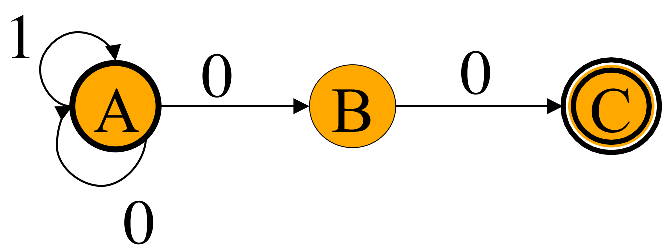
((0|1)*00)|0

ε-Closure (T: set of states)

push all states in T onto *stack*
initialize ε-*closure*(T) to T
while *stack* is not empty **do begin**
    pop t off *stack*
    for each state u with u ∈ **move**(t, **ε**) **do**
        if u ∉ **ε**-*closure*(T) **do begin**
            add u to **ε**-*closure*(T)
            push u onto *stack*
        end
end

Simulating NFAs

- An NFA may be in many states at any time



state	input
{A}	100
{A}	100
{A,B}	100
{A,B,C}	100

- How many different states?

s

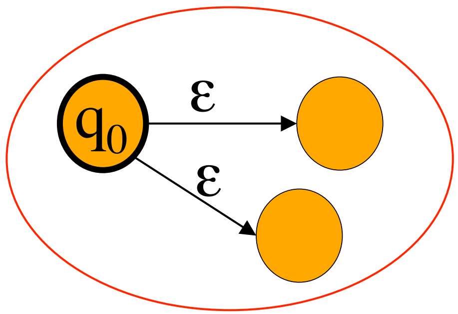
|S|=N	No. of states
|s| ≤ N	possible states in each step

$2\uparrow N - 1$

Non-empty subsets

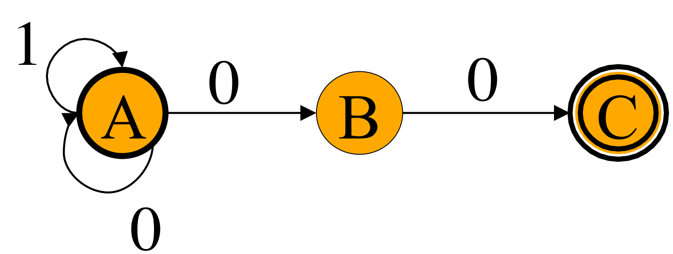
NFA to DFA Conversion

	NFA	DFA
• states	S	$X \subseteq S$
• start	q_0	ε-closure(q_0)
• final	$F \subseteq S$	
• transition	$\delta(x,a) = Y$	



NFA to DFA Conversion

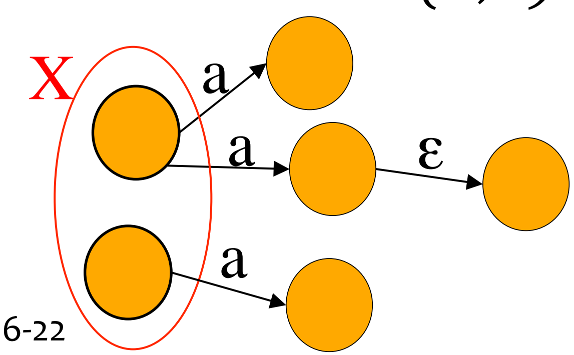
	NFA	**DFA**
• states	S	$X \subseteq S$
• start	q_0	$\varepsilon\text{-closure}(q_0)$
• final	$F \subseteq S$	$\{X \mid X \cap F \neq \emptyset\}$
• transition	$\delta(x,a) = Y$	



state	input
{A}	100
{A}	100
{A,B}	100
{A,B,C}	100

NFA to DFA Conversion

	NFA	**DFA**
• states	S	$X \subseteq S$
• start	q_0	ε-closure(q_0)
• final	$F \subseteq S$	$\{X \mid X \cap F \neq \emptyset\}$
• transition	$\boldsymbol{\delta(x,a) = Y}$	$\boldsymbol{\delta(X,a)} = \bigcup_{x \in X} \delta(x,a)$



ε-closure($\boldsymbol{\delta(X,a)}$)

DFAedge(X,a)=$\varepsilon\text{-}closure\left(\bigcup_{x \in X} \delta(x,a)\right)$

DFA construction

Dstates = {}, Dtrans = []

add ε-*closure*(q_0) to *Dstates* unmarked

while ∃ unmarked T ∈ *Dstates* **do**

mark T;

for each symbol c **do**

U := DFAedge(T,c);

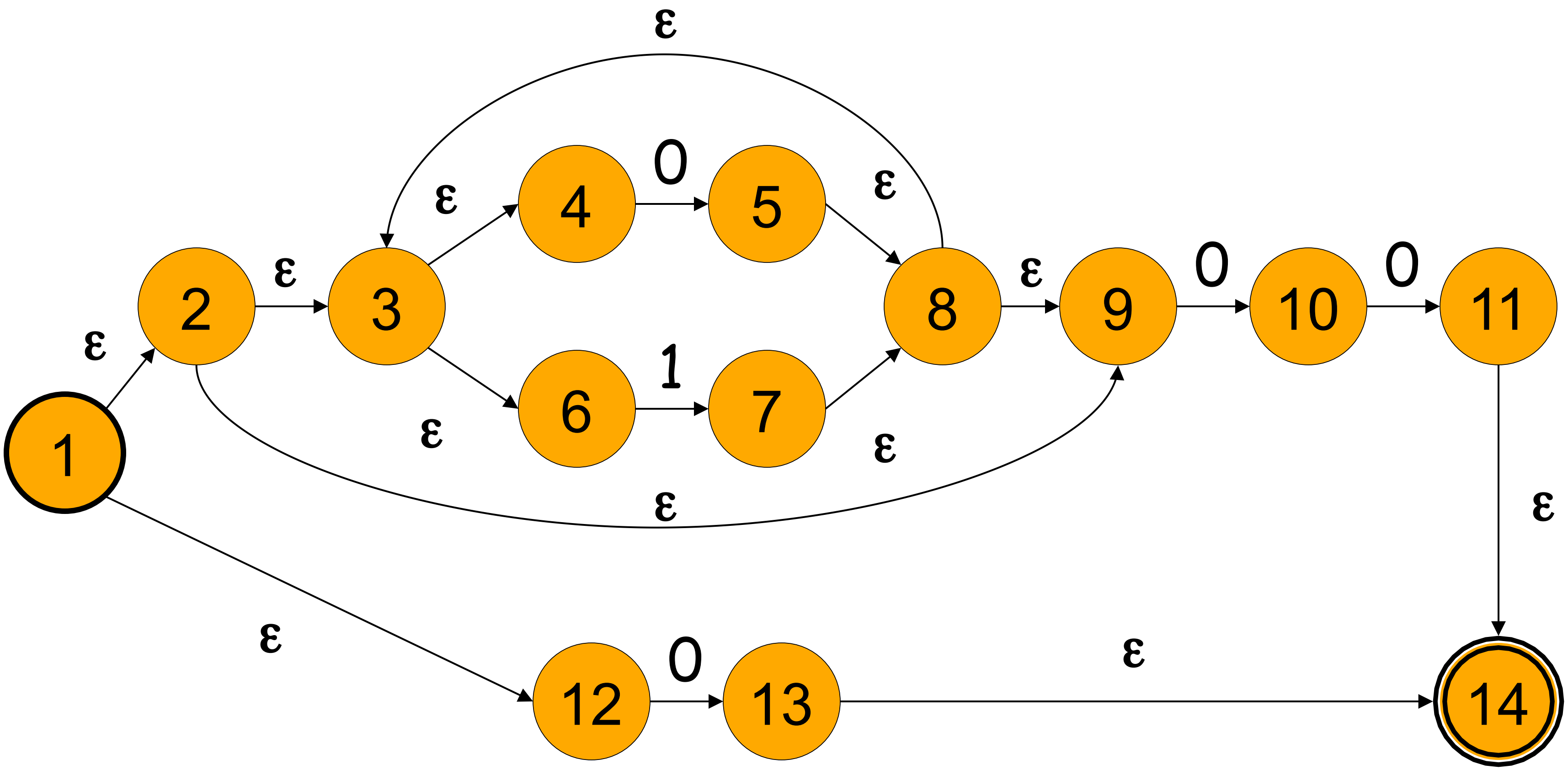
if U ∉ *Dstates* **then**

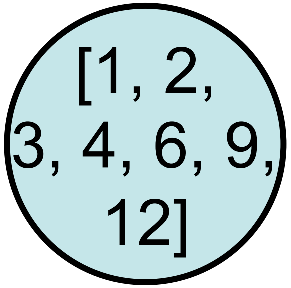
add U to *Dstates* unmarked

Dtrans[T, c] := U;

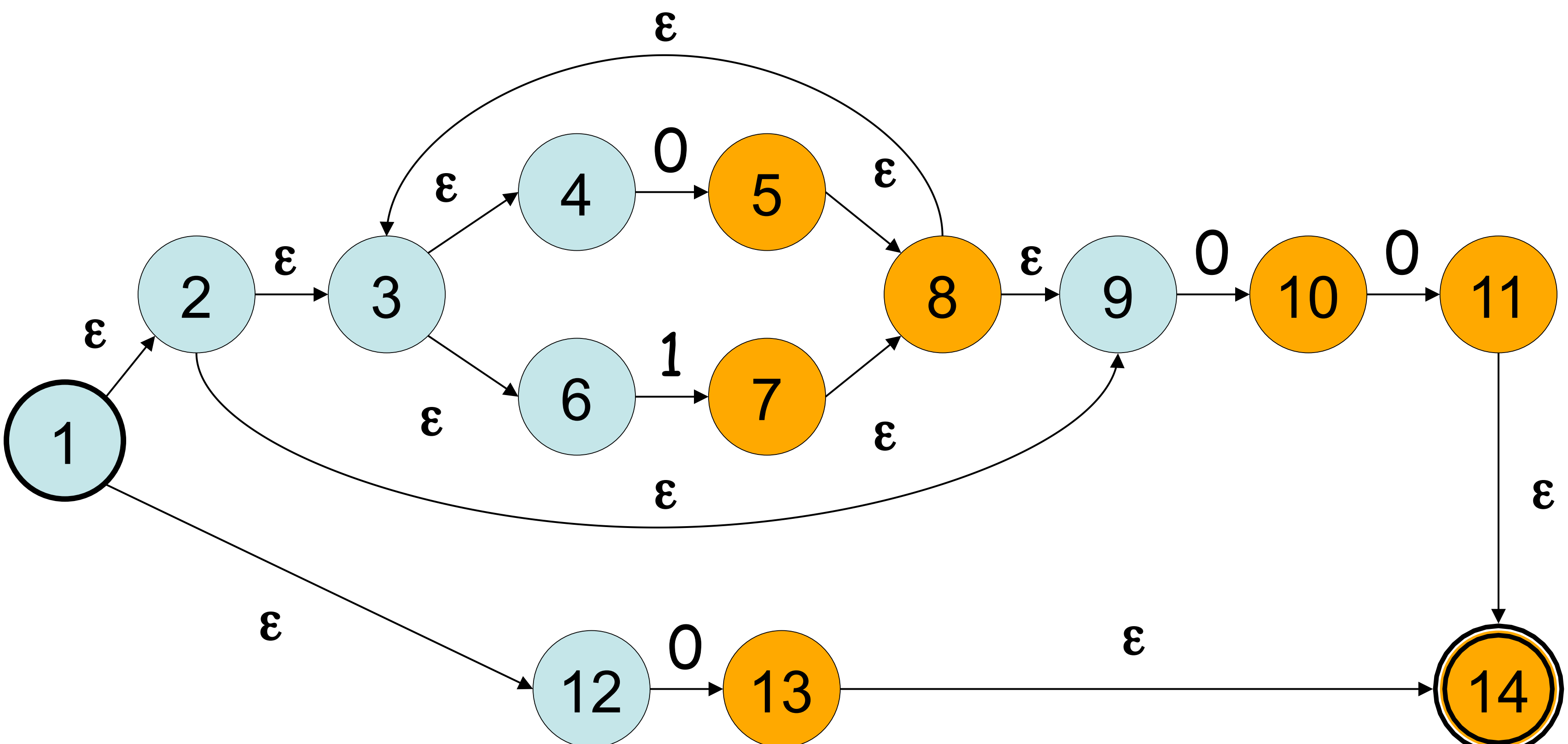
$$\mathbf{DFAedge}(\boldsymbol{T},\boldsymbol{c}) = \varepsilon - closure\left(\bigcup_{t\in T} \delta(t,c)\right)$$

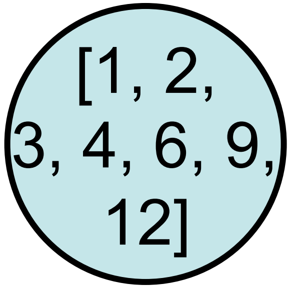
NFA to DFA



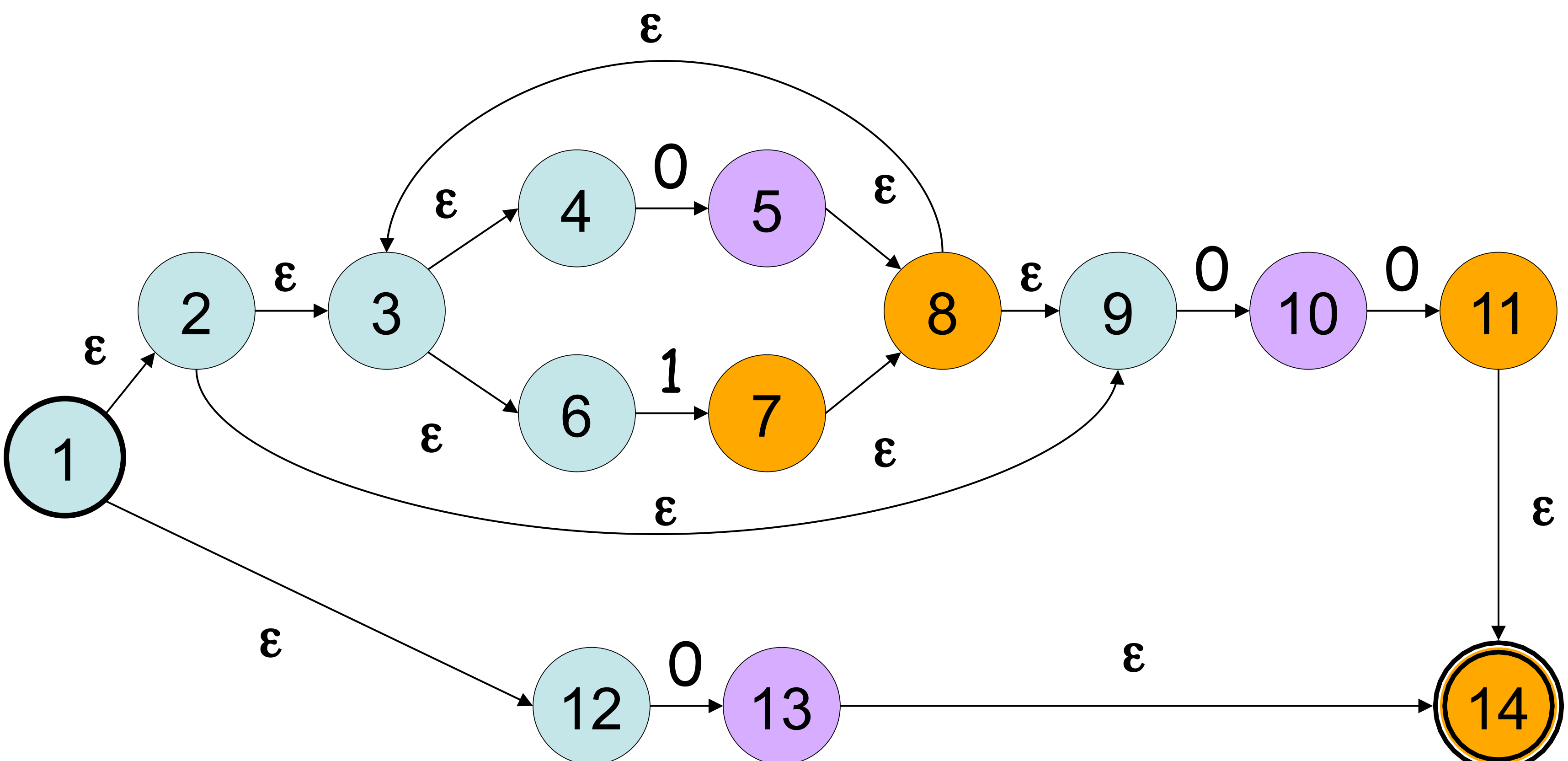


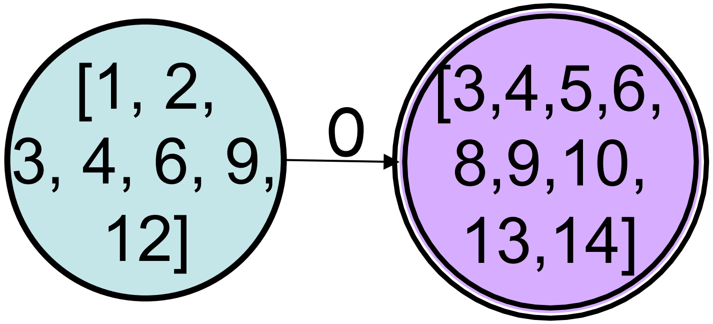
ε-*closure*(q_0)



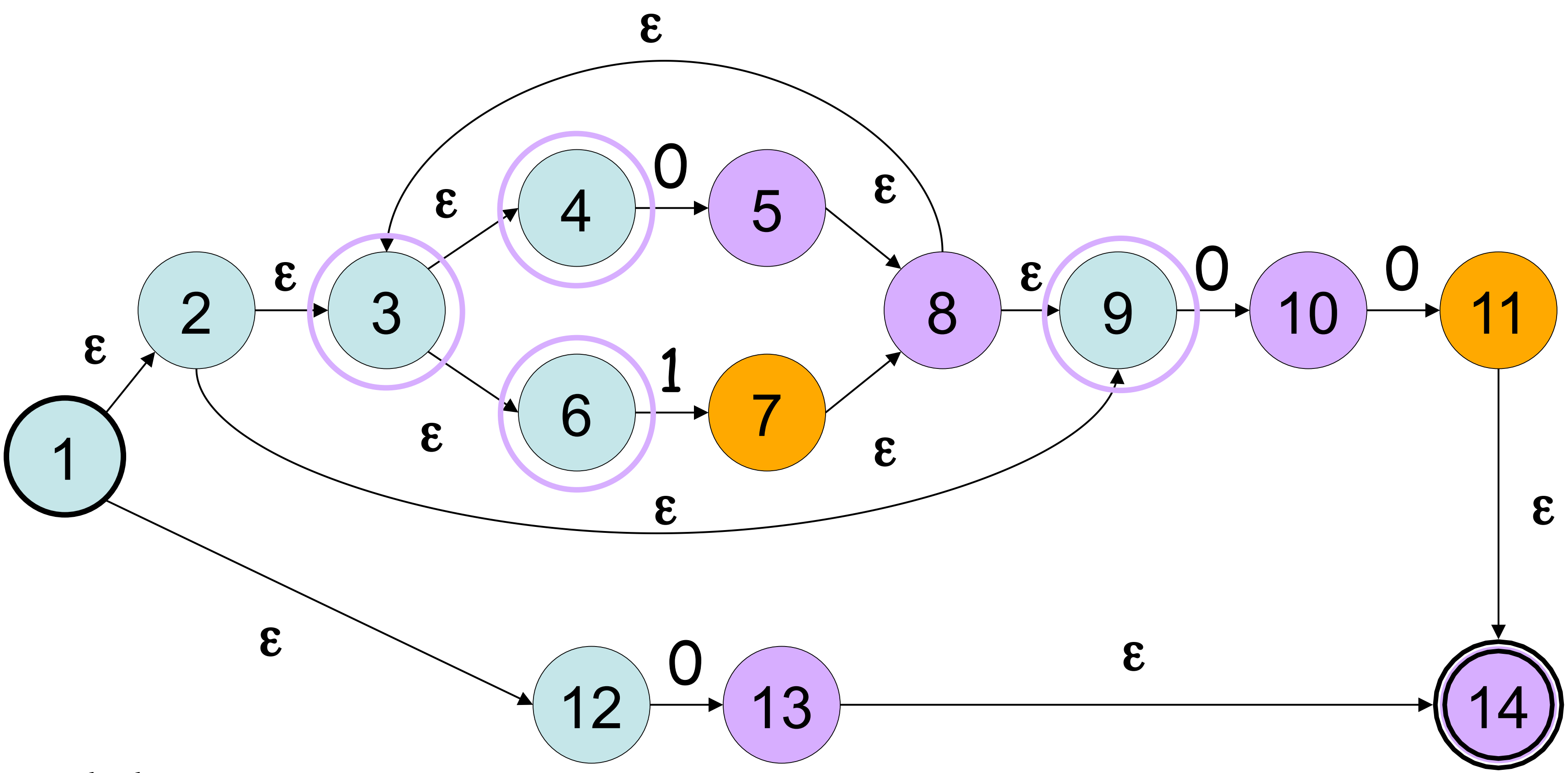


DFAedge(ε-*closure*(q_0), 0)

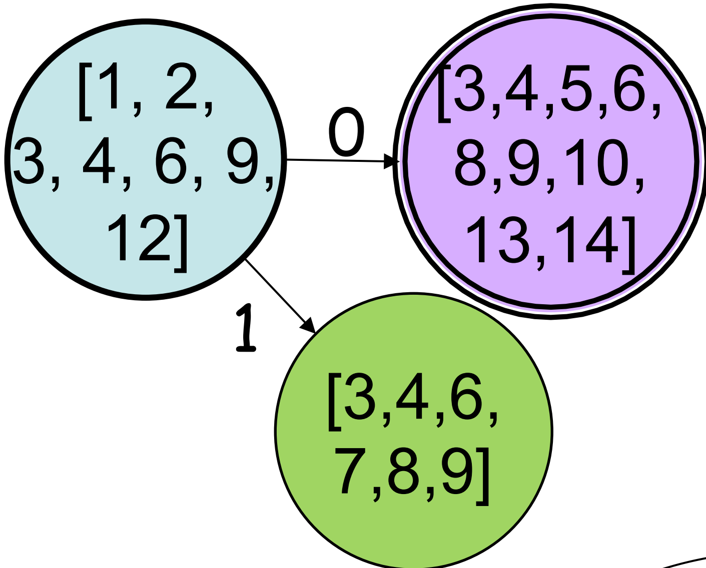


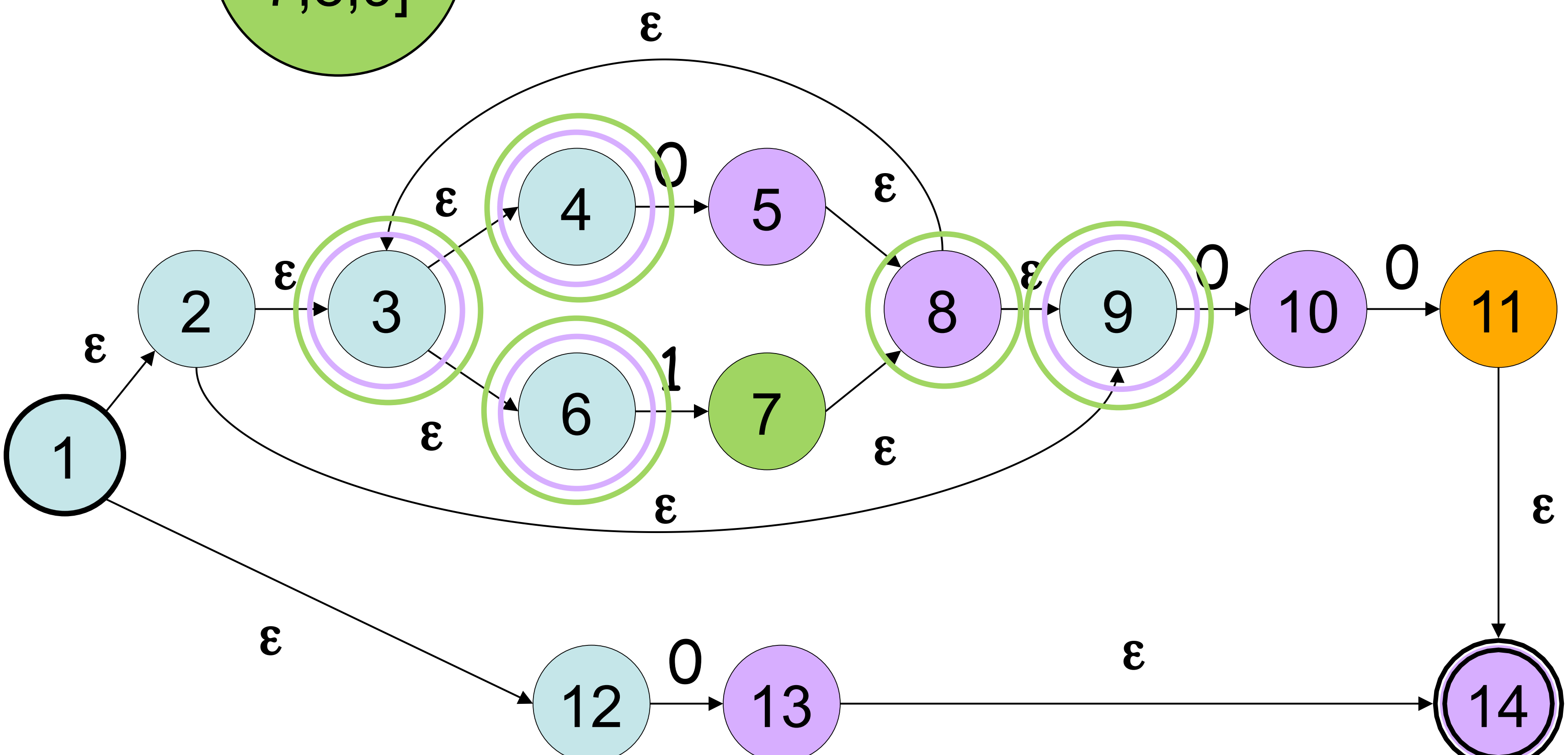


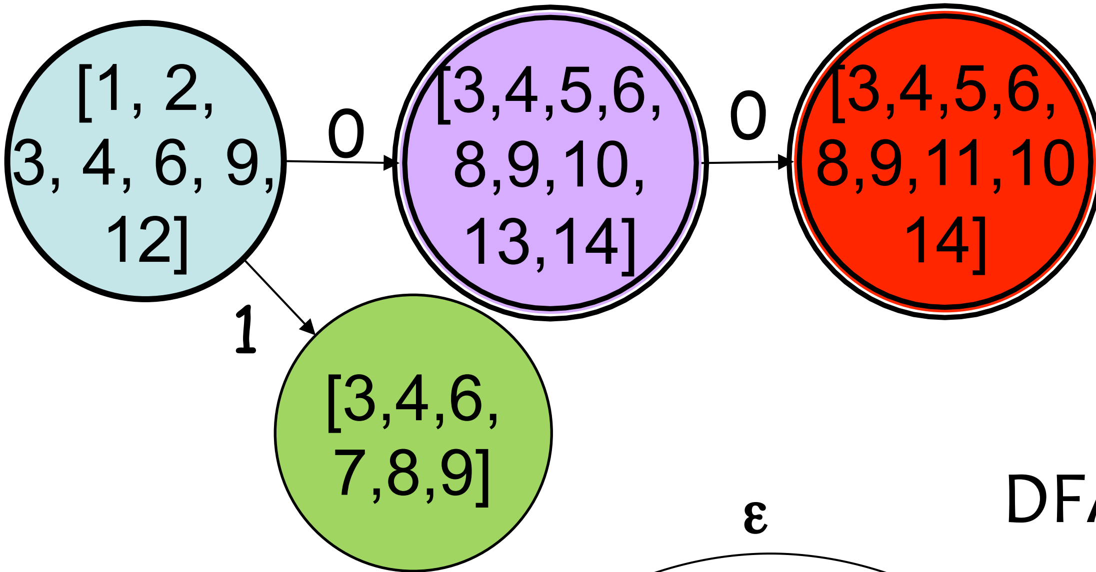
DFAedge(ε-*closure*(q_0), 0)



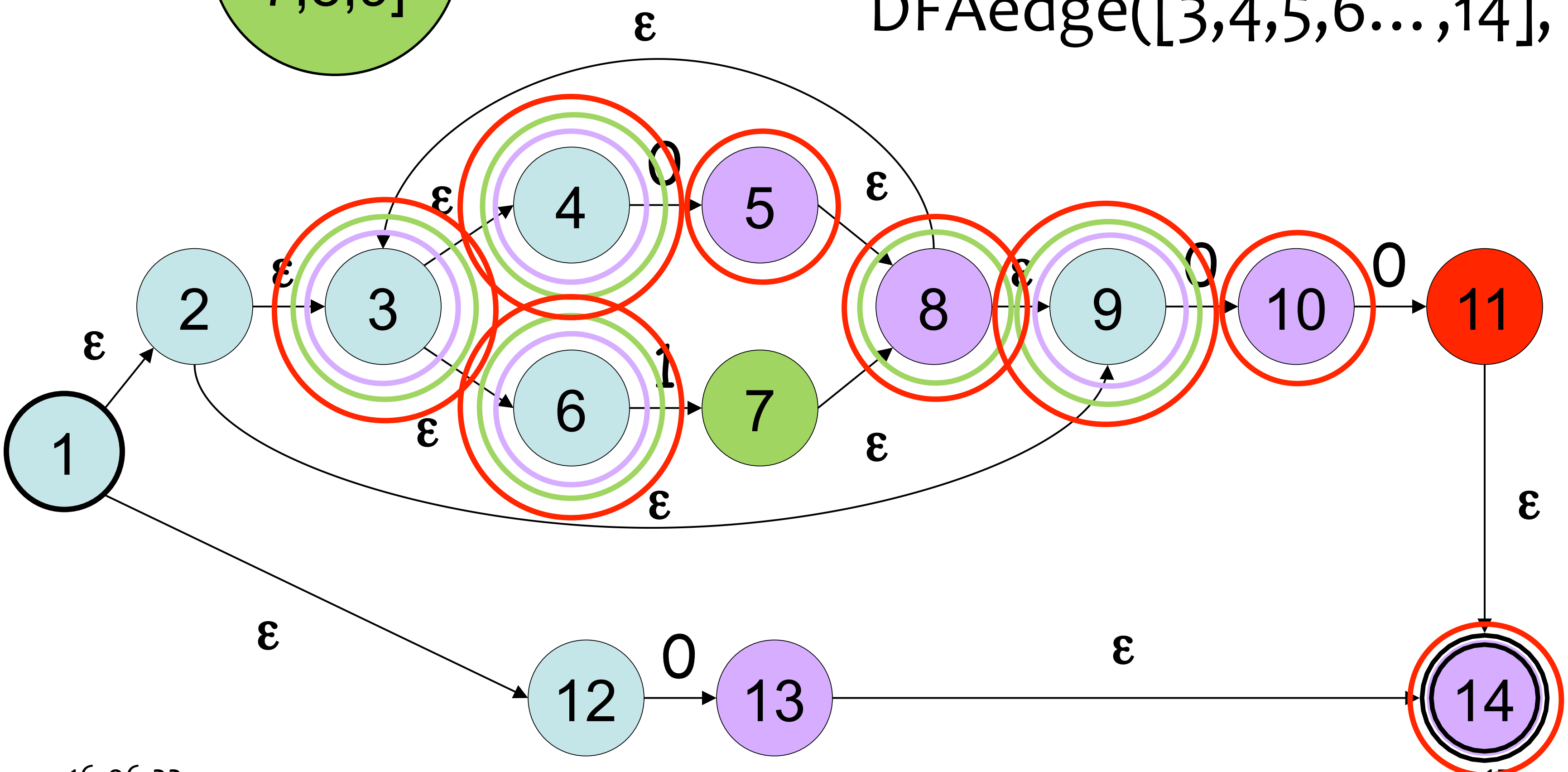
$DFAedge(\varepsilon\text{-}closure(q_0), 1)$



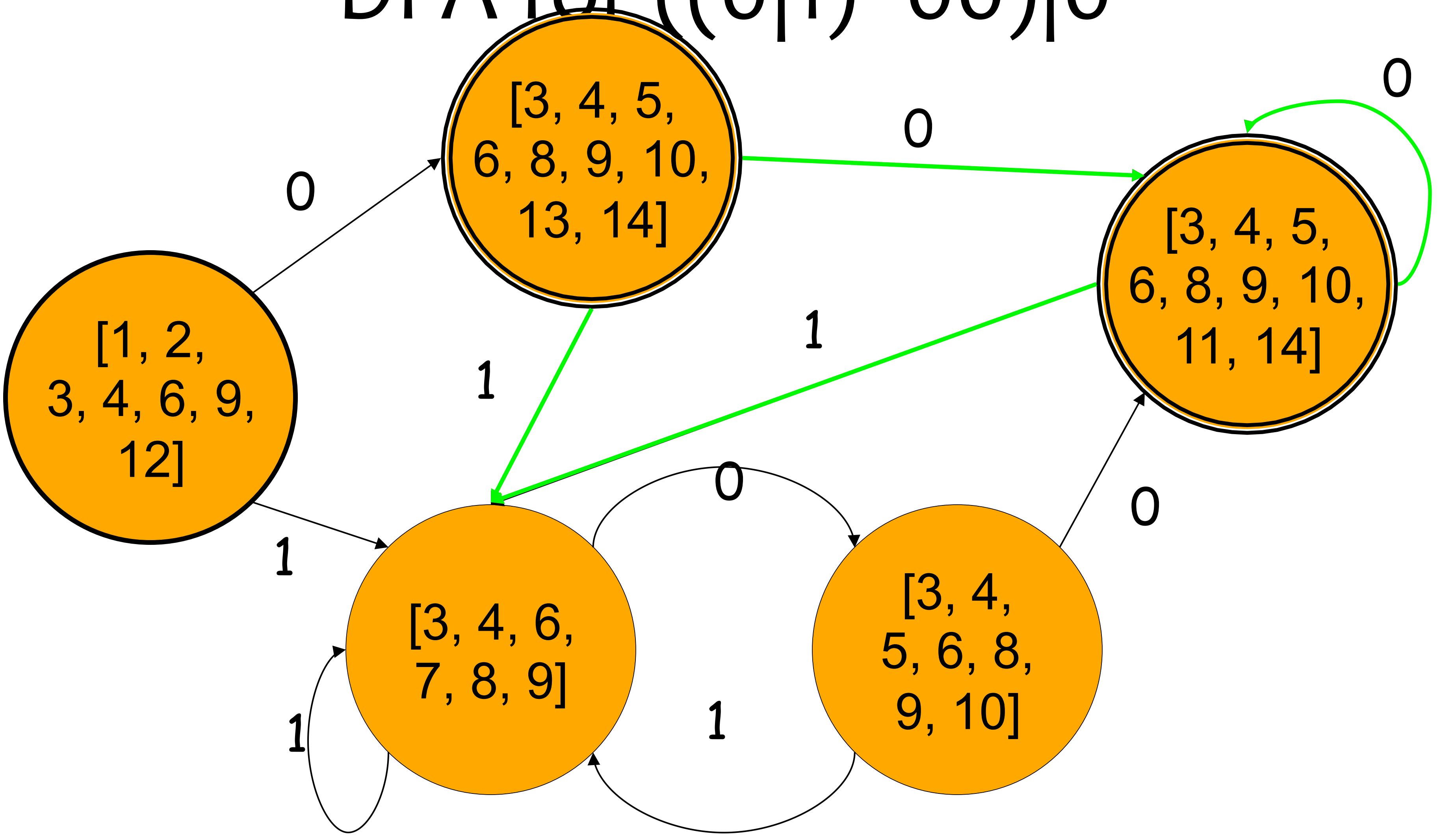




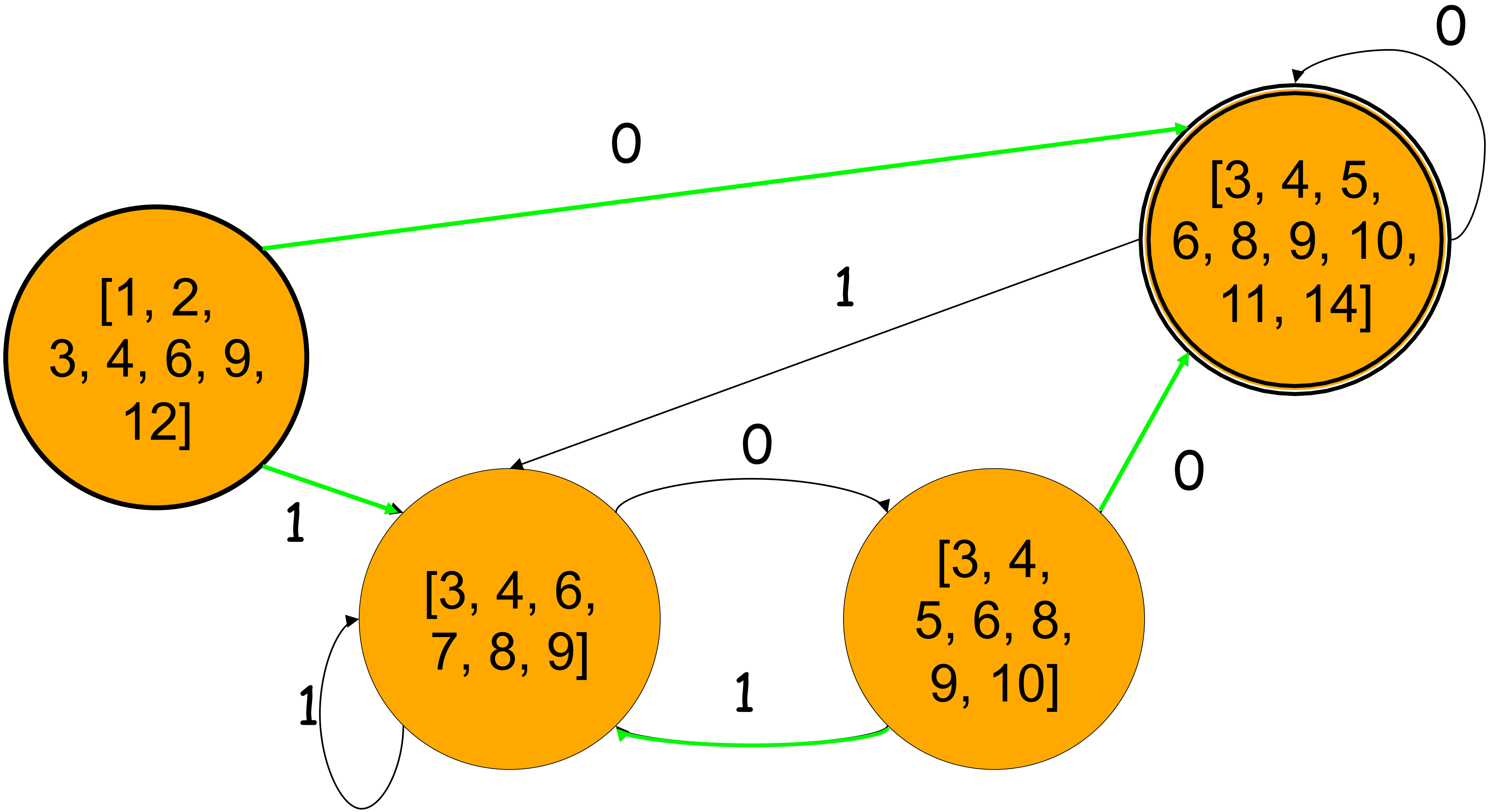
DFAedge([3,4,5,6…,14], 0)



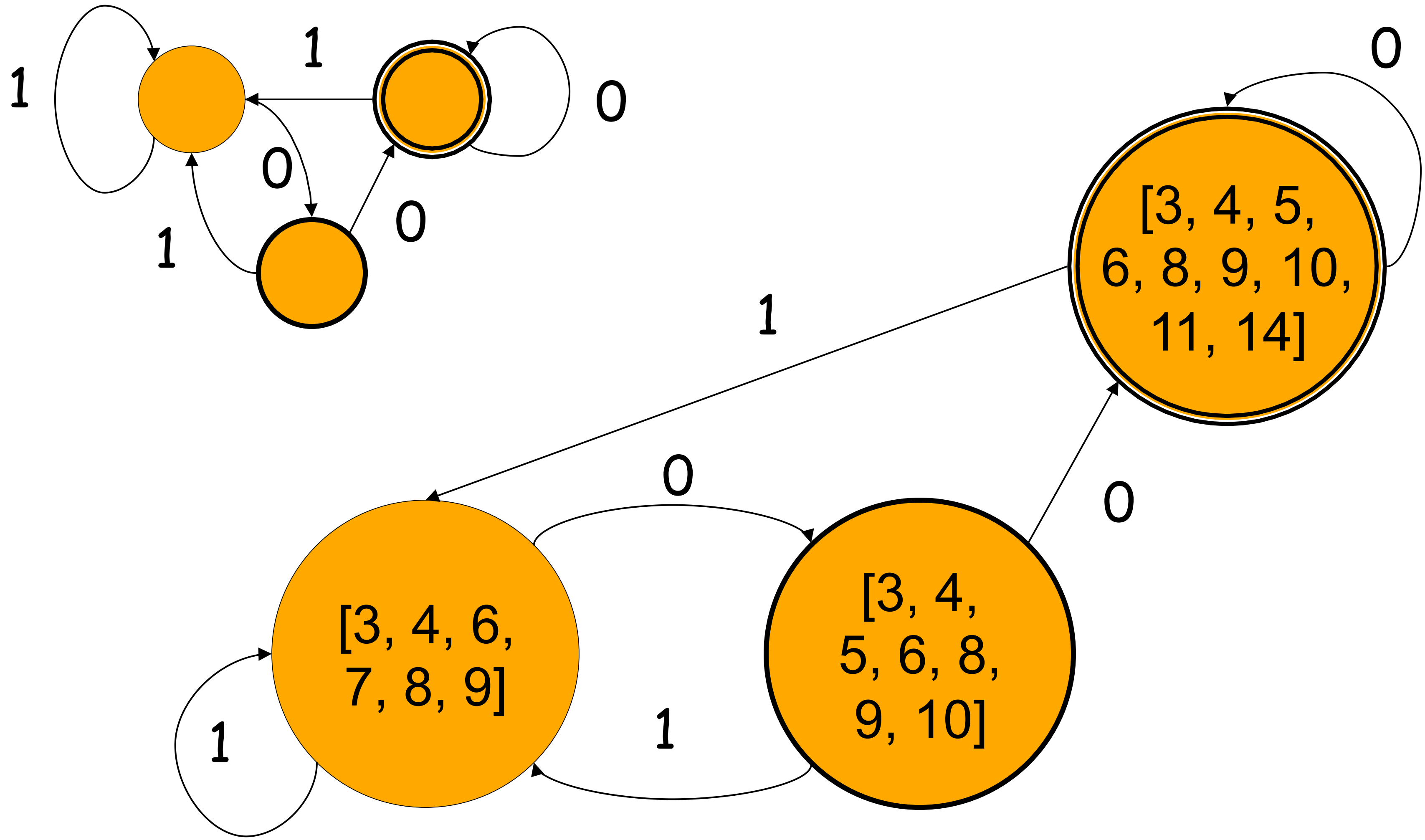
DFA for ((0|1)*00)|0



Minimization of DFAs



Minimization of DFAs



NFA to DFA Conversion

- Conversion method closely follows the NFA simulation algorithm
- Instead of simulating, we can collect those NFA states that behave identically on the same input
- Group this set of states to form one state in the DFA

NFA to DFA

```
states[0] = ε-closure({q0})
p = j = 0
while j ≤ p do
        for each symbol c ∈∑↑▒ do
                e = DFAedge(states[j], c)
                if e = states[i] for some i ≤ p
                then    Dtrans[j, c] = i
                else    p = p+1
                        states[p] = e
                        Dtrans[j, c] = p
        j = j + 1
```