CMPT 379
Compilers

Anoop Sarkar

http://www.cs.sfu.ca/~anoop

Matching Patterns using Non-deterministic Automata (conversion from NFA to DFA)

Simulating NFAs

- Simulation == Given a NFA and input string, does the string match the pattern?
- Similar to DFA simulation
- But have to deal with ε transitions and multiple transitions on the same input
- Instead of one state, we have to consider *sets* of states

NFA to DFA Conversion

- Simulation implicitly converts NFA -> DFA
- Subset construction
- Idea: subsets of set of all NFA states are *equivalent* and become one DFA state
- Algorithm simulates movement through NFA
- Key problem: how to treat ε-transitions?

ε-Closure

- Start state: q_0
- ε-closure(S): S is a set of states

$$
\begin{aligned}
&\textbf{initialize: } S \leftarrow \{q_0\} \\
&T \leftarrow S \\
&\textbf{repeat } T' \leftarrow T \\
&\quad T \leftarrow T' \cup \left[\cup_{s \in T'} \textbf{move}(s, \epsilon)\right] \\
&\textbf{until } T = T'
\end{aligned}
$$

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

NFA Simulation

- After computing the ε-*closure* move, we get a set of states
- On some input extend all these states to get a new set of states

$$\mathbf{DFAedge}(T, c) = \epsilon\textbf{-closure}\left(\cup_{q \in T} \mathbf{move}(q, c)\right)$$

NFA Simulation

- $\mathbf{DFAedge}(T, c) = \epsilon\textbf{-closure}\left(\cup_{q \in T} \mathbf{move}(q, c)\right)$
- Start state: q_0
- Input: $c_1, \ldots, c_k$

$$T \leftarrow \epsilon\textbf{-closure}(\{q_0\})$$

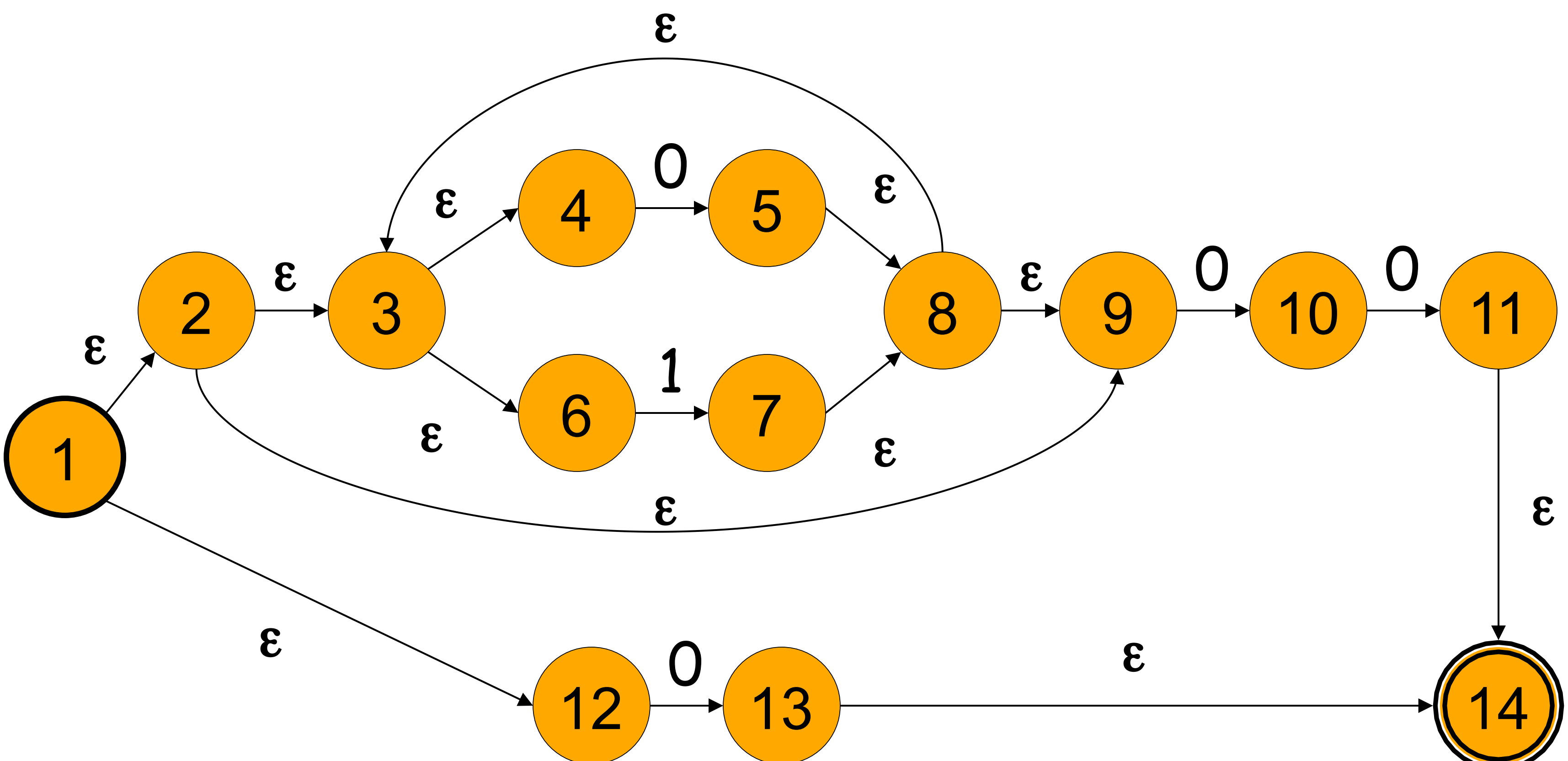
$$\textbf{for } i \leftarrow 1 \textbf{ to } k$$

$$\quad T \leftarrow \mathbf{DFAedge}(T, c_i)$$

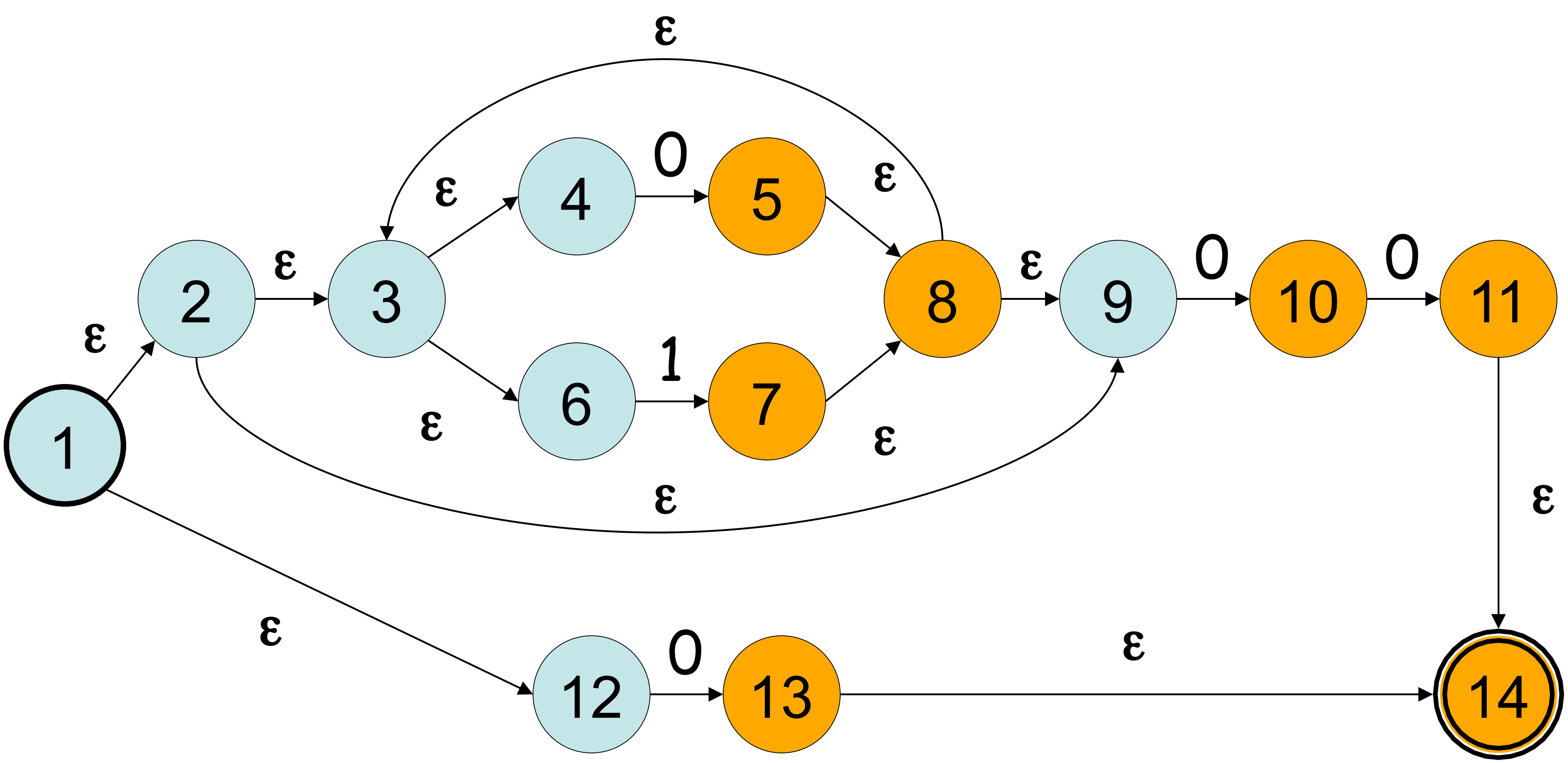
Conversion from NFA to DFA

- 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

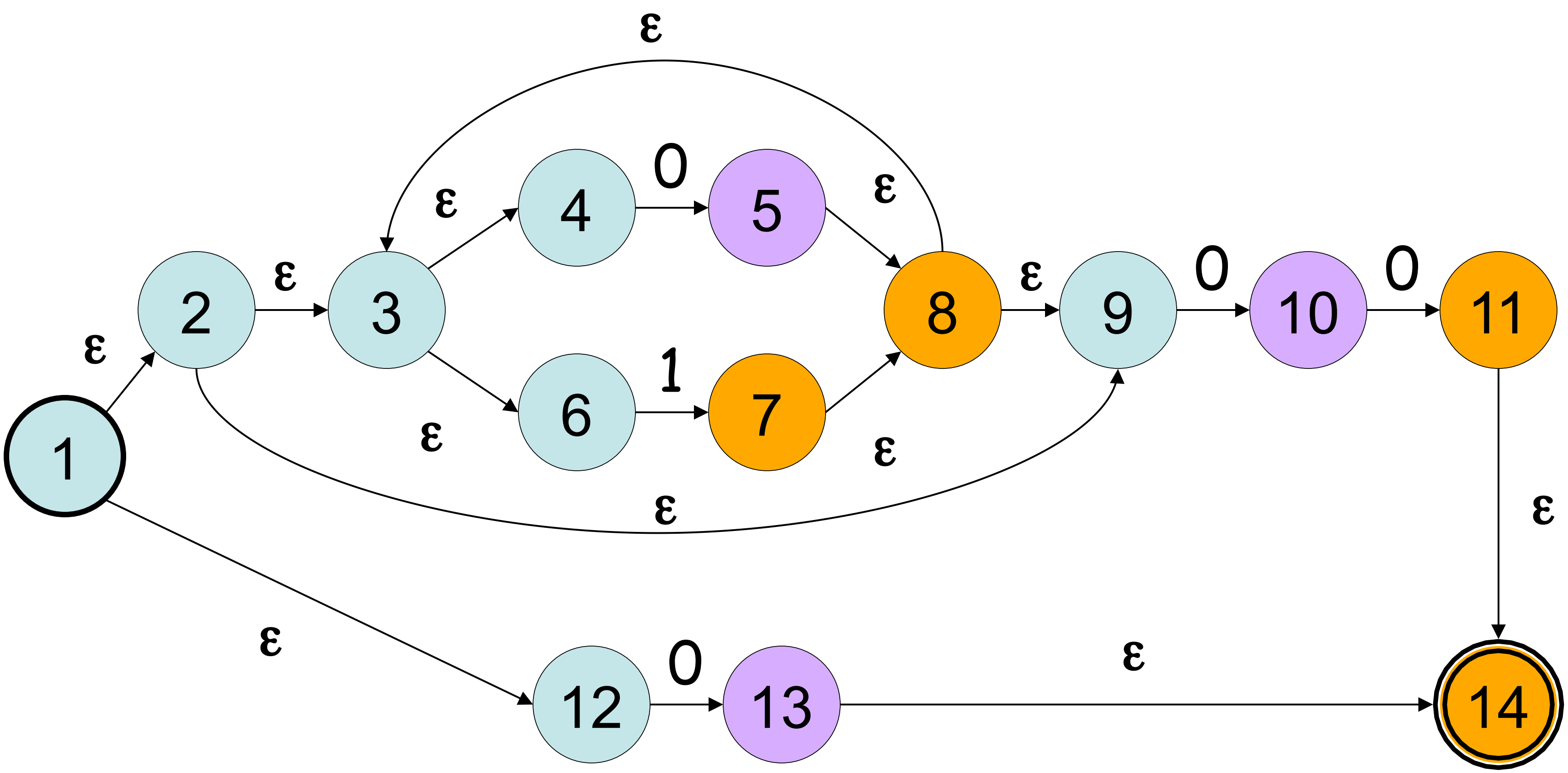
Example: subset construction



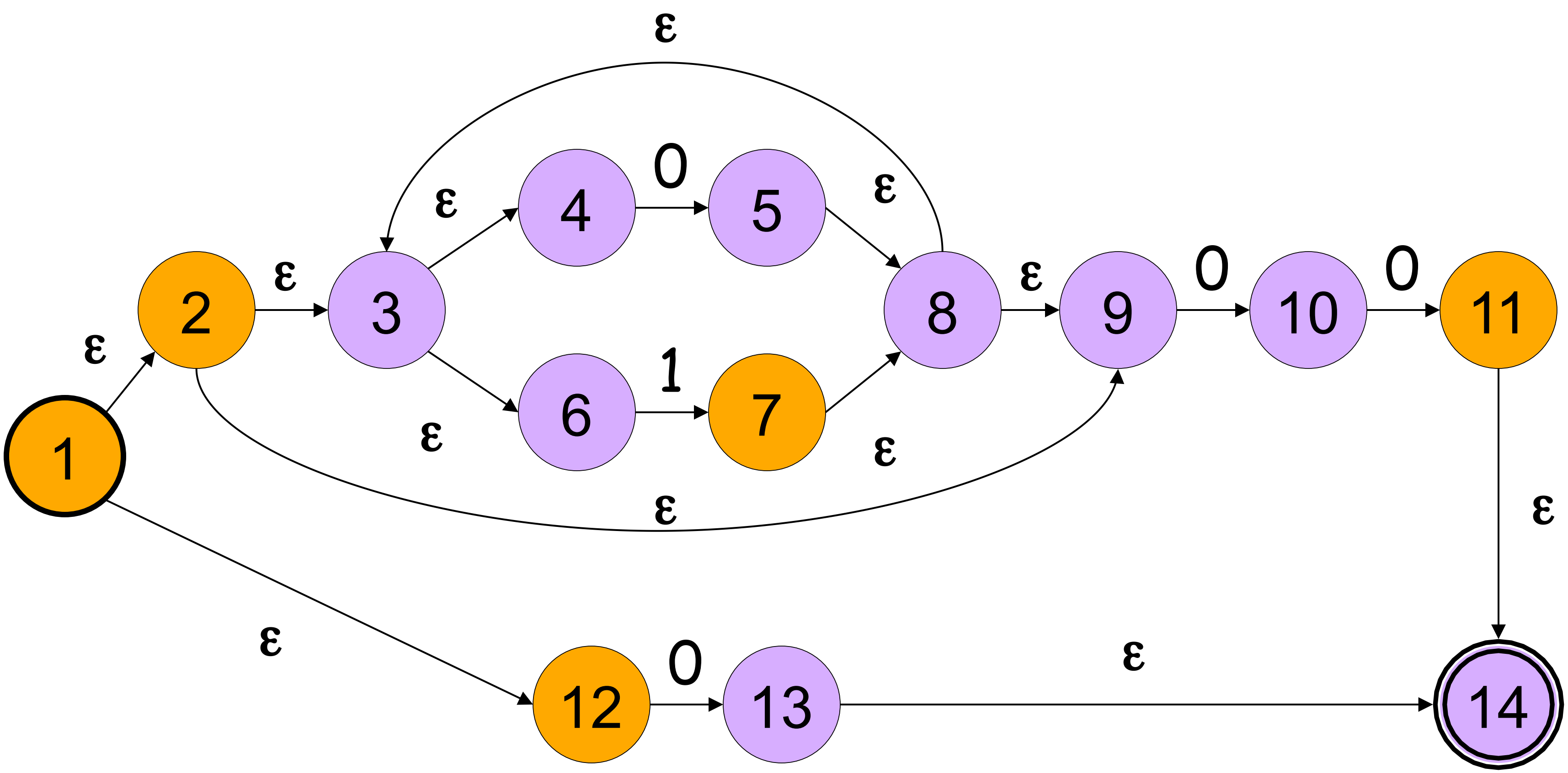
ε-*closure*(q_0)



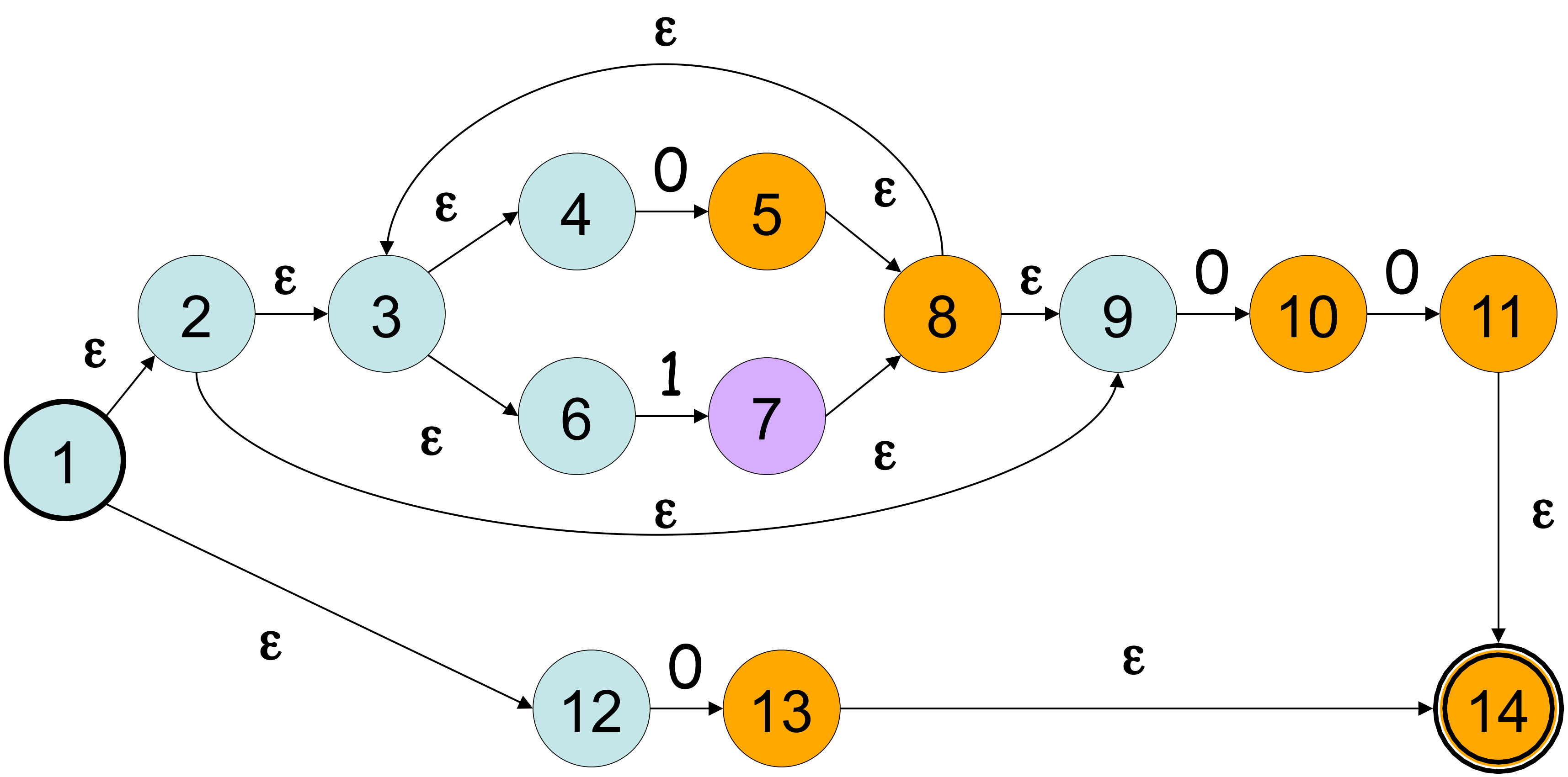
move(ε-*closure*(q_0), 0)



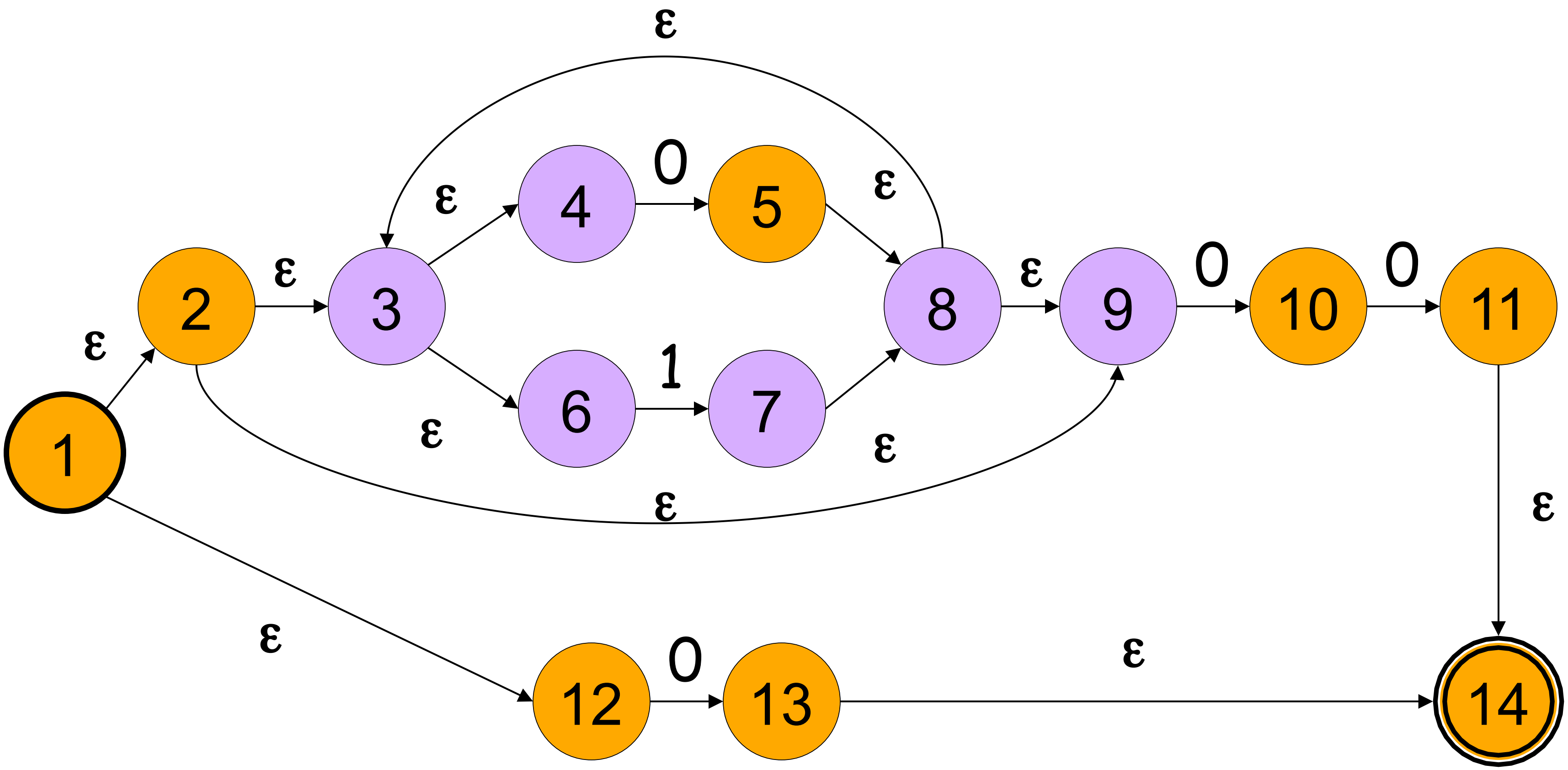
ε-*closure*(move(ε-*closure*(q_0), 0))



move(ε-*closure*(q_0), 1)



ε-*closure*(move(ε-*closure*(q_0), 1))



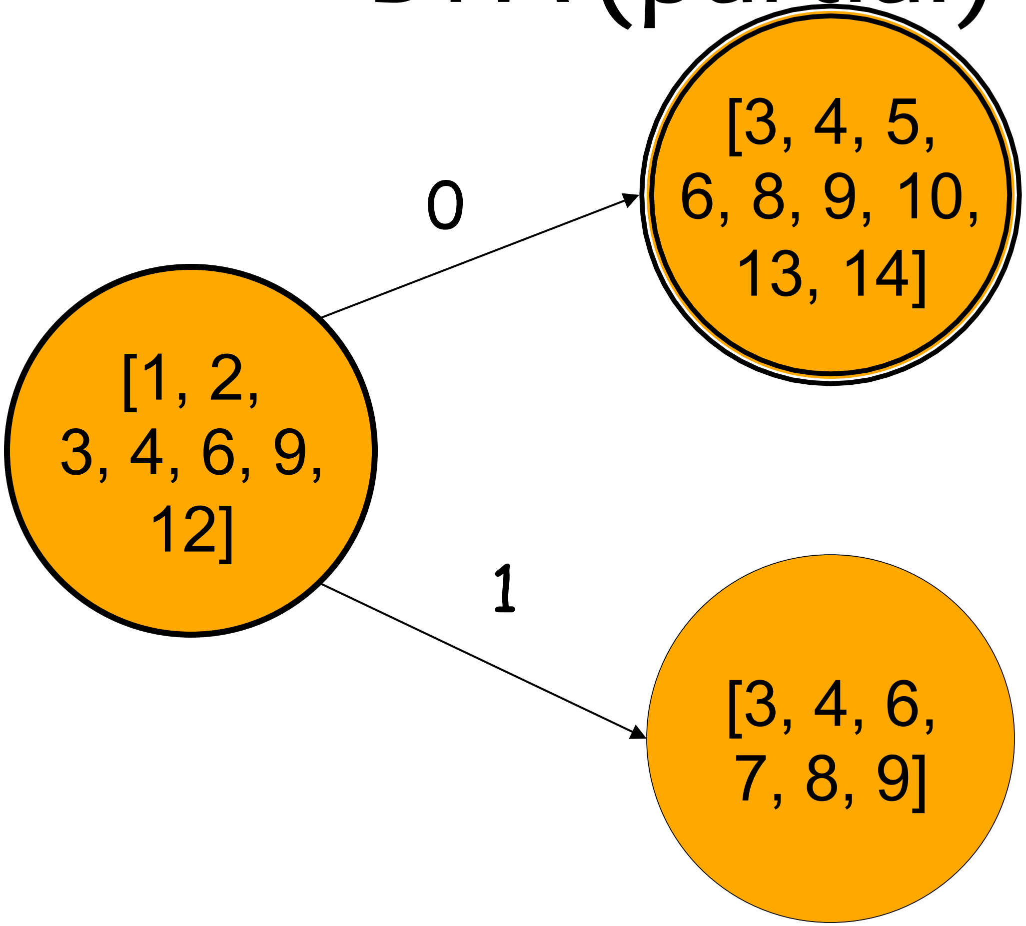
Subset Construction

```
add ε-closure(q0) to Dstates unmarked
while ∃ unmarked T ∈ Dstates do begin
    mark T;
    for each symbol c do begin
        U := ε-closure(move(T, c));
        if U ∉ Dstates then
            add U to Dstates unmarked
        Dtrans[d, c] := U;
    end
end
```

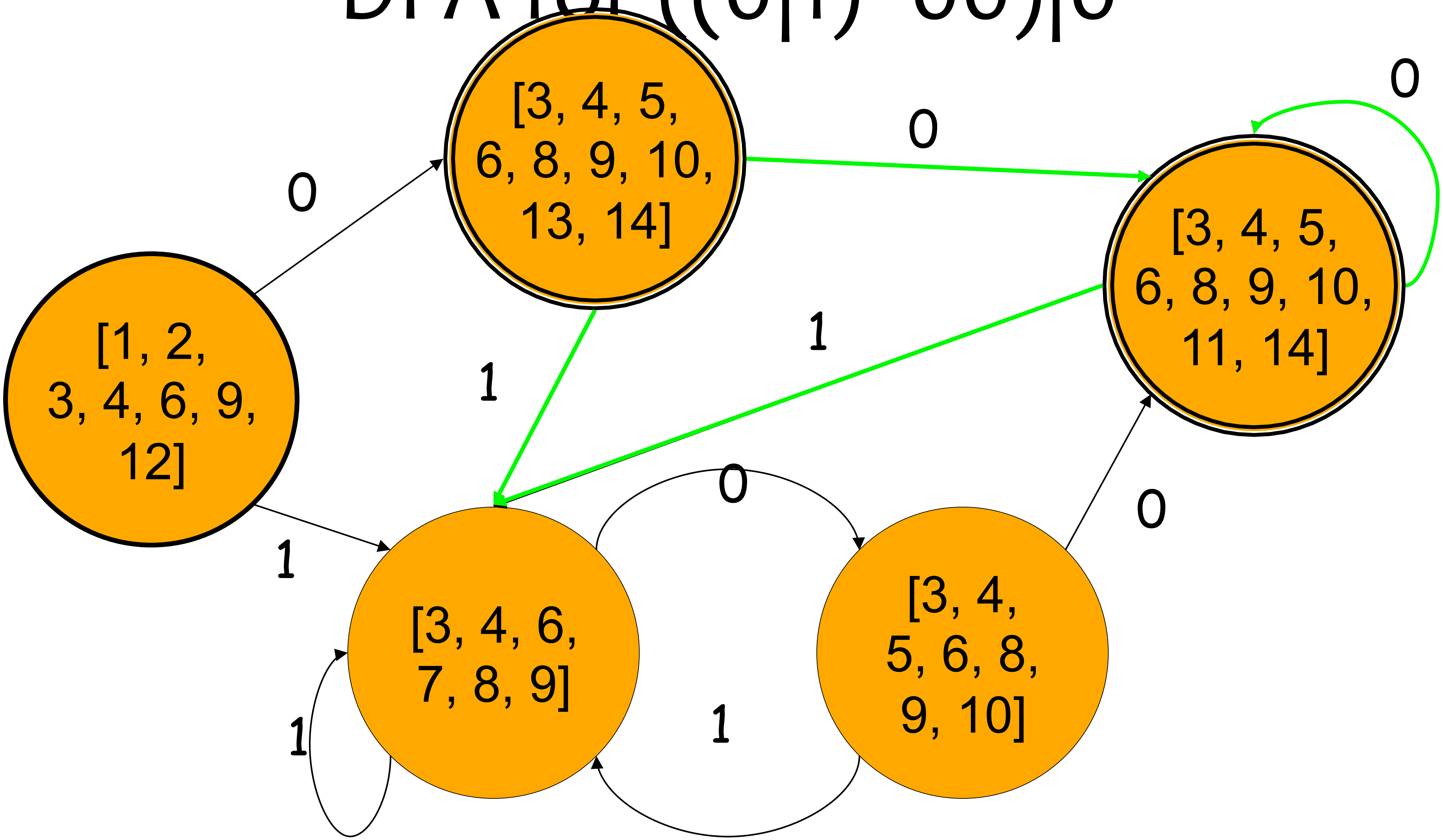
Subset Construction

```
states[0] = ε-closure({q0})
p = j = 0
while j ≤ p do begin
        for each symbol c do begin
                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
        end
end
```

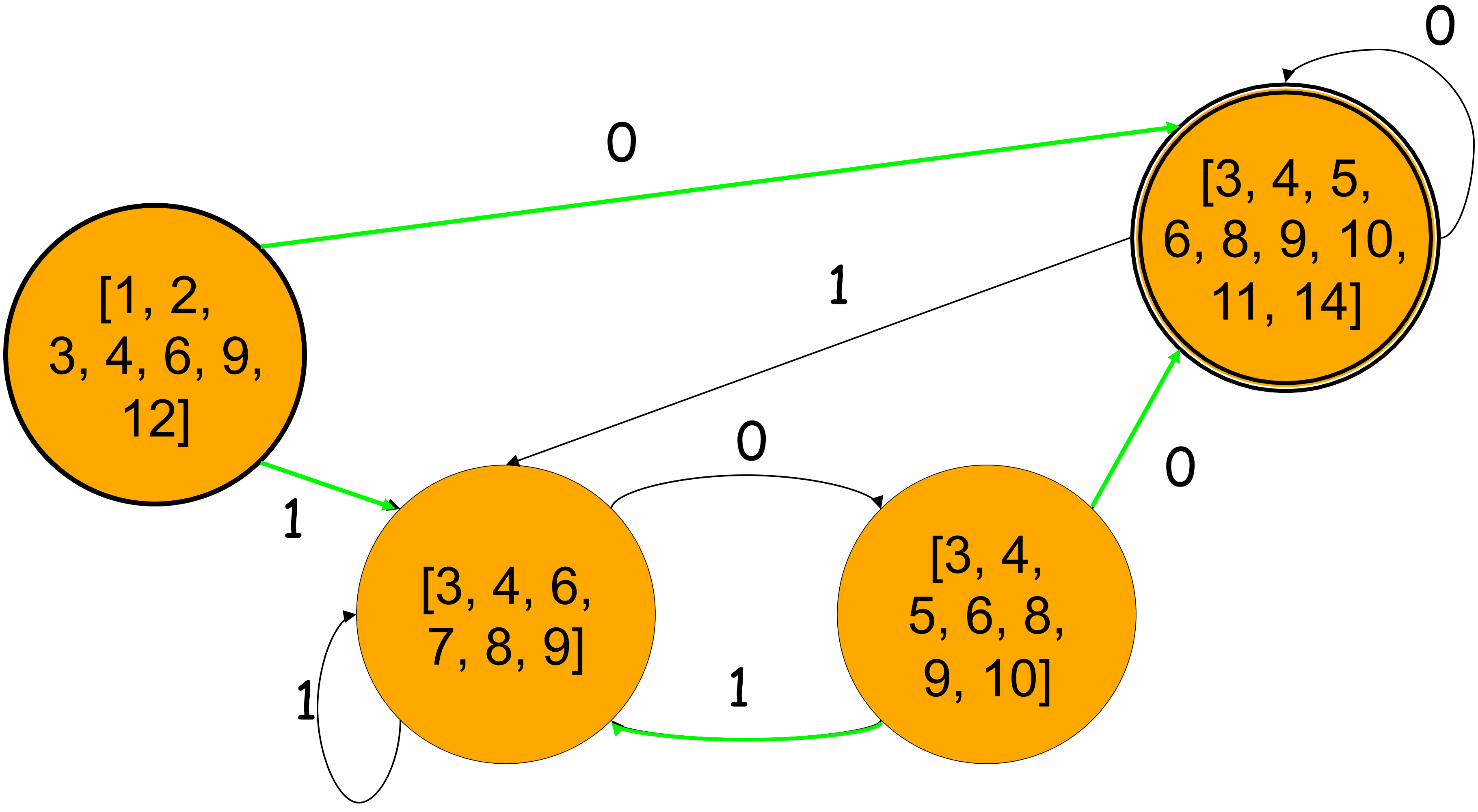
DFA (partial)



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



Minimization of DFAs



Minimization of DFAs