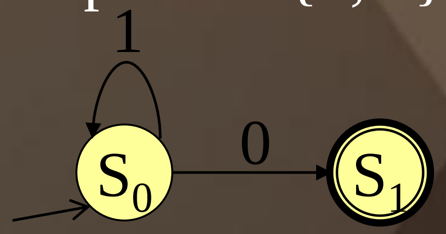
# *Lecture #4*

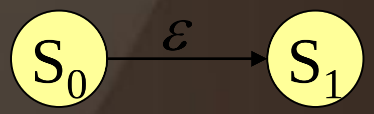
## Lexical Analysis - II

# *Finite Automata*

- Language of a FA ≡ Set of all accepted strings
    - Eg: Any number of 1's followed by a '0'
    - Alphabet: {0, 1}



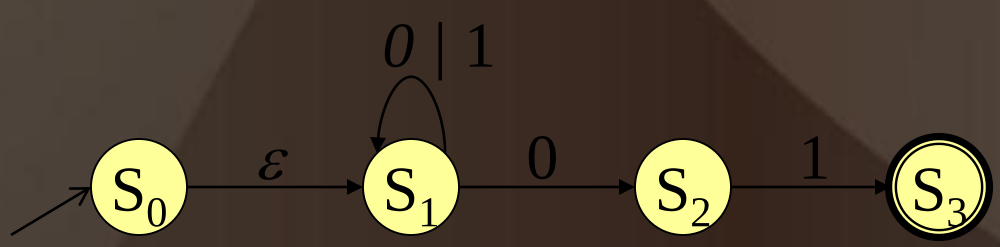

- Another kind of transition: *ε-moves*




    - *Control can move to $S_1$ on all input symbols that takes control to $S_0$*

# *DFA and NFA*

- Deterministic Finite Automata (DFA)
  - One transition per input per state
  - No $\varepsilon$-moves
- Non-deterministic Finite Automata (NFA)
  - Can have multiple transitions for one input at a given state
  - Can have $\varepsilon$-moves
  - eg: (0| 1)* 01



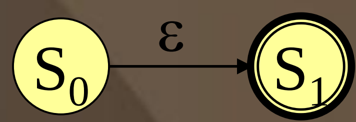
# *DFAs and NFAs*

- DFA takes only one path the state space graph
- NFA accepts a string if there exists at least one path through the graph that leads to an accepting state.
    - DFA is a special case of NFA
- *DFAs can lead to faster recognisers but are exponentially bigger than NFAs.*

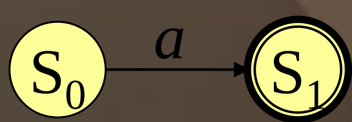
- Automation steps
    - RE → NFA → DFA → Table-Driven Implementation
- *For each kind of RE, an equivalent NFA can be designed*

# *RE to NFA (Thompson's Construction)*

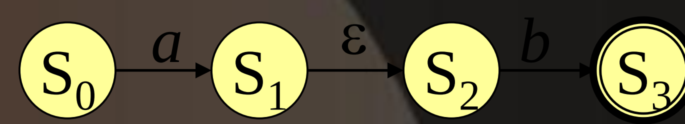
*For each kind of RE, an equivalent NFA can be designed*



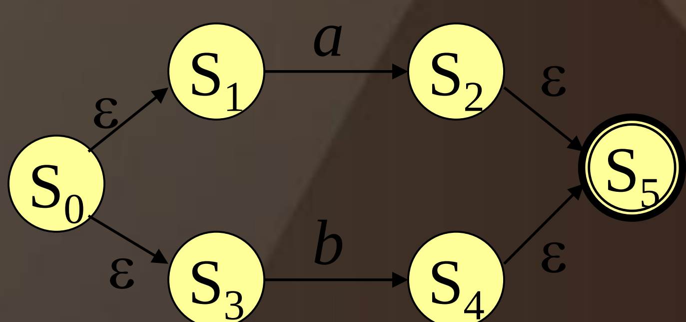

NFA for ε



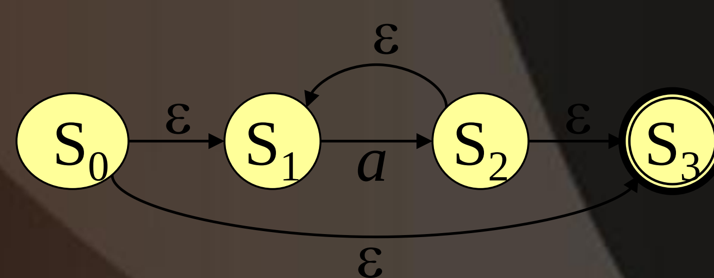

NFA for *a*




NFA for *ab*



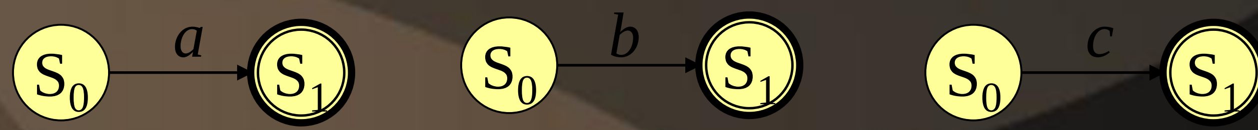

NFA for *a* | *b*

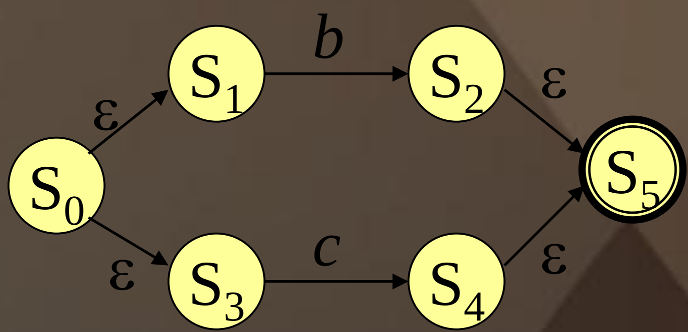



NFA for *a**

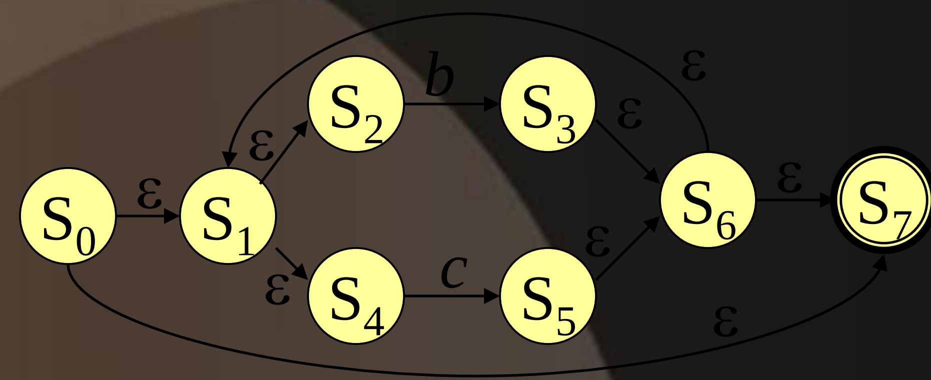
# *NFA Construction for the RE a(b|c)**

Step 1: NFAs for *a*, *b*, *c*



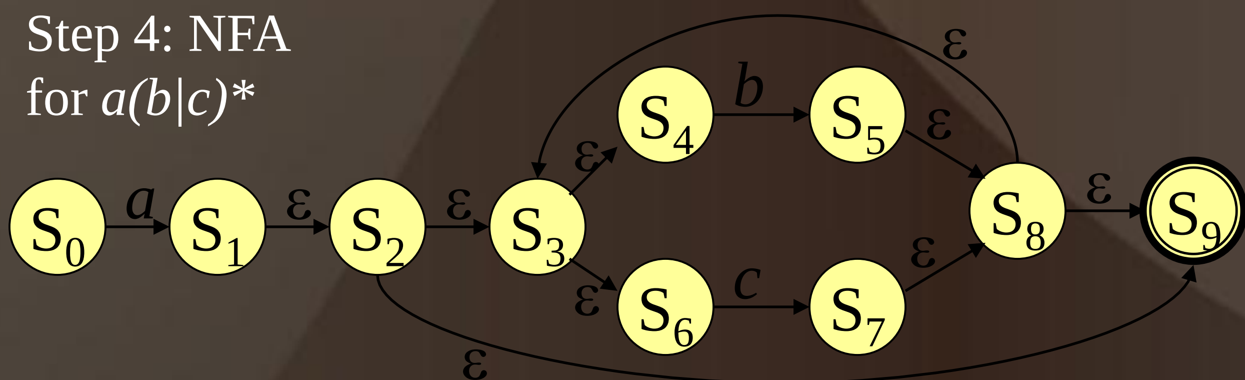




Step 2: NFA for *b*|*c*




Step 3: NFA for *(b*|*c)**

Step 4: NFA for *a(b*|*c)**

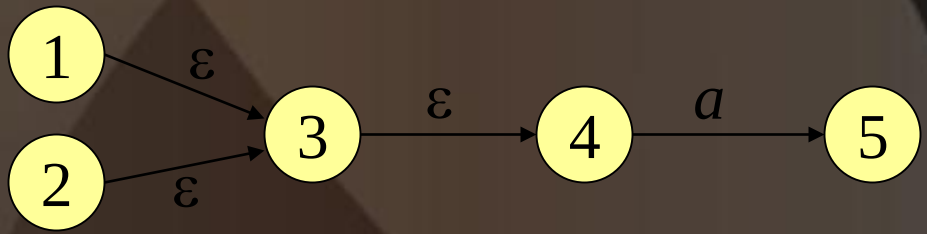



*Key Idea:*

- Design NFA pattern for each symbol and/or operator
- Join them in precedence order

# *NFA to DFA*

- **Two key functions:**
  - **move($s_i$, a):** The (union of the) set of states to which there is a transition on input symbol **a** from state $s_i$
  - **ε-closure($s_i$):** The (union of the) set of states reachable by **ε** from $s_i$.
- Example: ε-closure(2)={2,3,4}; ε-closure({2,1})={2,3,4,1};
  - move(ε-closure({2,1}), *a*) =5;




- The Algorithm:

  Start with the ε-closure of $s_0$ from NFA.

  Do for each unmarked state until there are no unmarked states:

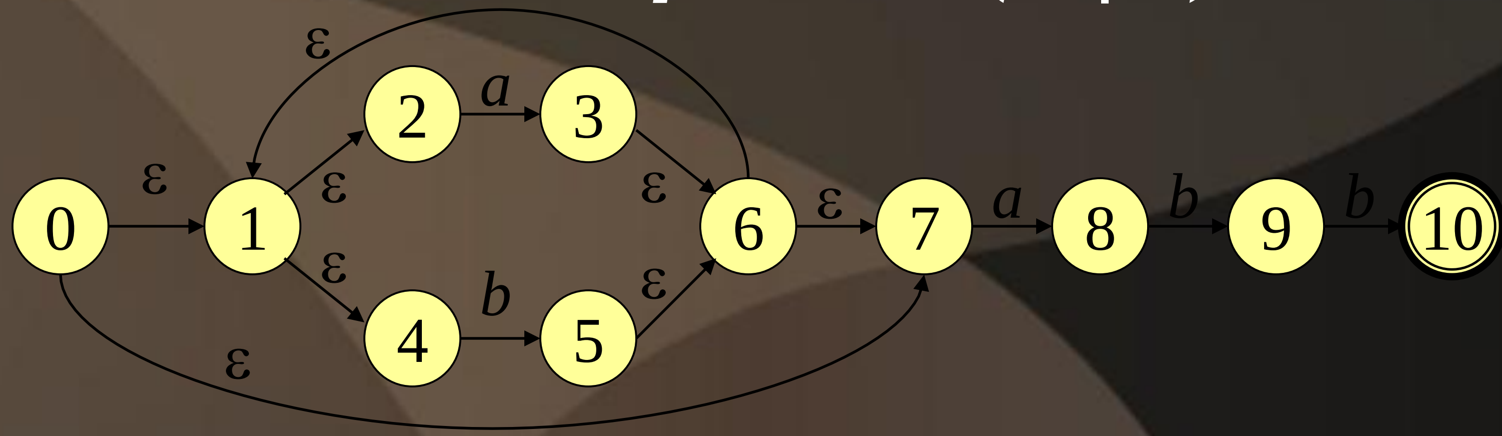
  for each symbol take their ε-closure(move(state,symbol))

# *NFA to DFA*

Initially, ε-closure is the only state in Dstates and it is unmarked.
**while** there is an unmarked state T in Dstates
    mark T
    **for each** input symbol a
        U:=ε-closure(move(T,a))
        **if** U is not in Dstates then add U as unmarked to Dstates
        Dtable[T,a]:=U

- From Dstates (set of states for DFA) and Dtable form the DFA.
- Each state of DFA corresponds to a set of NFA states that NFA could be in after reading some sequences of input symbols.
- Given an NFA with *n* states, the corresponding DFA may have up to $2^n$ states

# *NFA to DFA: for RE (a | b)*abb*




- A=ε-closure(0)={0,1,2,4,7}
- for each input symbol (that is, *a* and *b*):
  - B=ε-closure(move(A, *a*))=ε-closure({3, 8})={1,2,3,4,6,7,8}
  - C=ε-closure(move(A, *b*))=ε-closure({5})={1,2,4,5,6,7}
  - Dtable [A, *a*] = B; Dtable [A, *b*] = C
- B and C are unmarked. Repeating the above we end up with:
  - C={1,2,4,5,6,7}; D={1,2,4,5,6,7,9}; E={1,2,4,5,6,7,10}; and
  - Dtable [B, *a*] = B; Dtable [B, *b*] = D; Dtable [C, *a*] = B; Dtable [C, *b*] = C; Dtable [D, *a*] = B; Dtable [D, *b*] = E; Dtable [E, *a*] = B; Dtable [E, *b*] = C;
  - No more unmarked sets at this point!

# *NFA to DFA: for RE (a | b)*abb*

Transition table:

| state | a | b |
|---|---|---|
| A | B | C |
| B | B | D |
| C | B | C |
| D | B | E |
| E(final) | B | C |