NFA vs DFA

Nondeterministic FA vs Deterministic FA

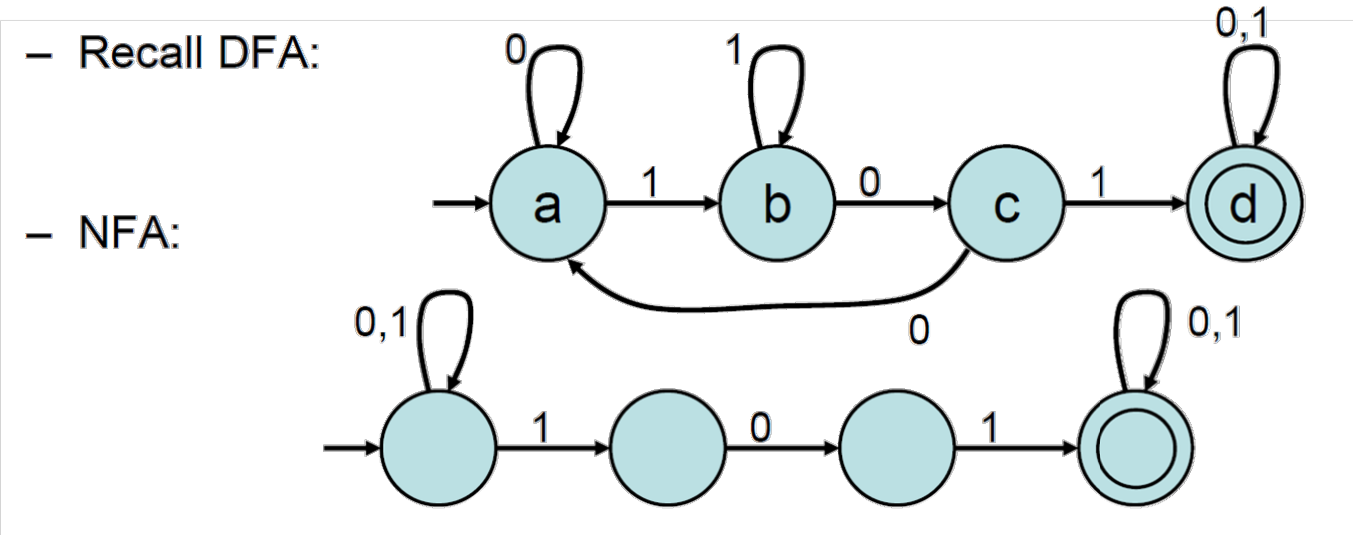
- NFA can be easier to construct
  - NFA diagrams are usually smaller than DFA
  - NFA states may be easier to understand

- NFA and DFA can recognize the same languages
  - If a language is DFA-recognizable it is also NFA-recognizable <u>and</u> vice versa.
  - Two machines are **equivalent** if they recognize the same language.

- Theorem: Every NFA has an equivalent DFA
  - NFA can always be converted into DFA
  - DFA may have many more states

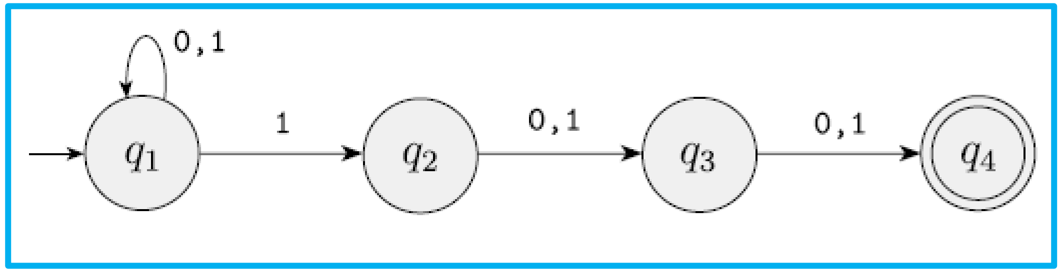
NFA vs DFA

- NFAs and DFAs have same power
  - NFAs can be "simpler" than equivalent DFAs
- Example: L = Strings having substring 101
- NFA "guesses" by following a path that goes through those states
  - Easier to see the required 101 pattern

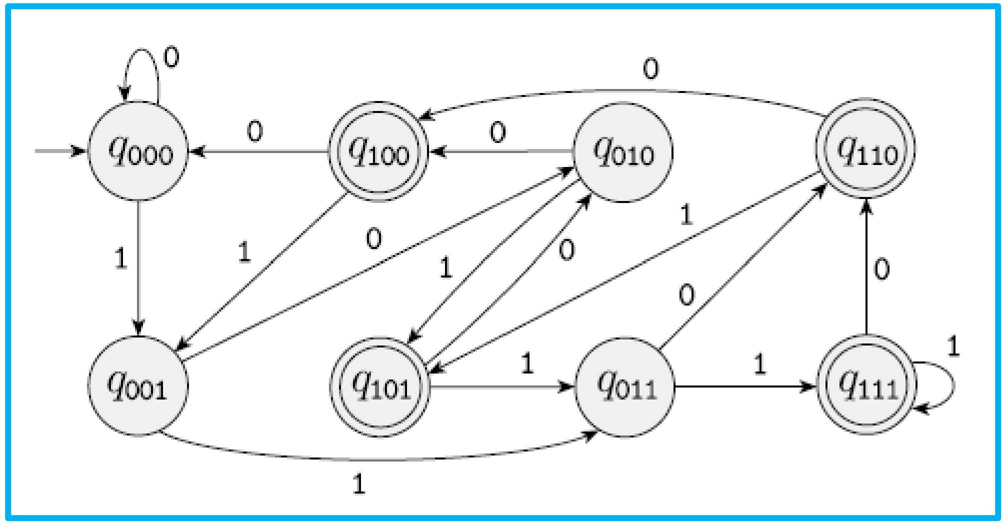


NFA vs DFA diagram

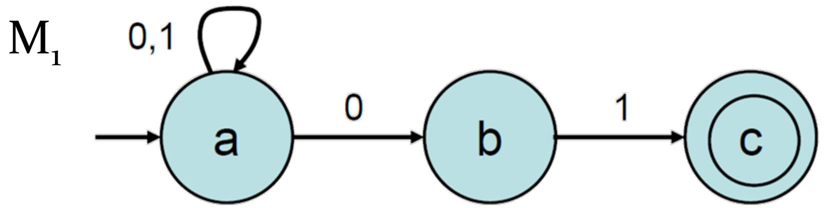
- Let A be the language consisting of binary strings with a
  - 1 in the 3rd position from the end
- NFA that recognizes A



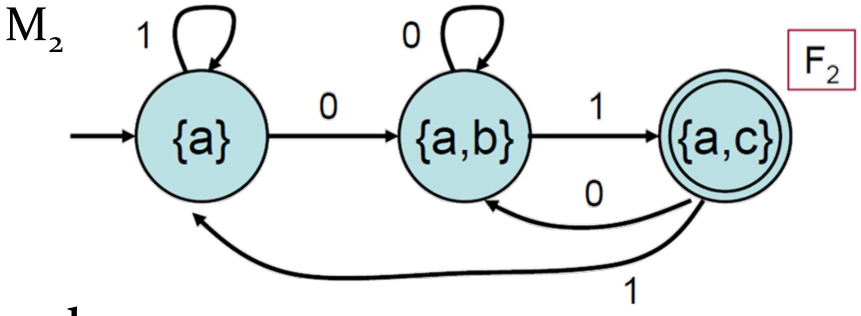
- DFA that recognizes A



Example: NFA to DFA



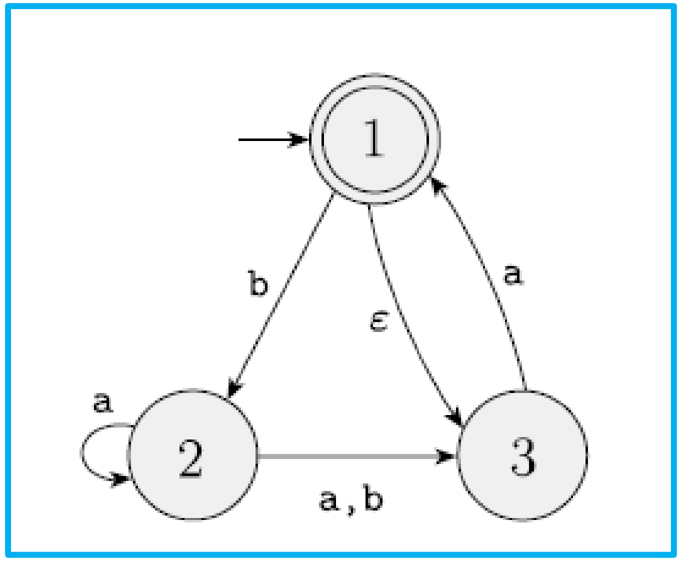
- Convert NFA M_1 to DFA M_2
- List all possible states M_2
  - Powerset P(Q) where Q = {a,b,c}
  - P(Q) = { {}, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c} }
    - Each set is now a state



- Determine start and accept states of M_2
  - q_0 = {a} and any state with c is an accept state.
- Determine the transition function of M_2
  - $\delta(p, a)$ = set of all states that are reachable from p by traveling along edge with symbol a in M_1
    - p maybe multiple states in a
  - Draw new node and edge in diagram or note in transition table
- Remove/ignore unreachable elements in P(Q)

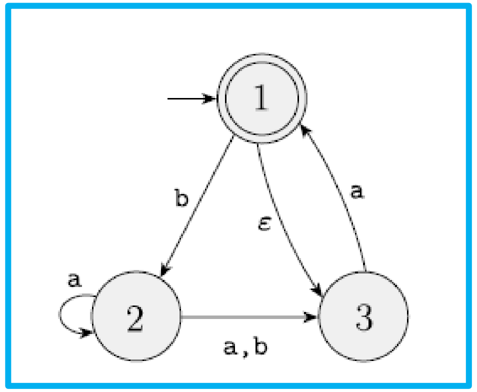
Example 2: NFA to DFA

- NFA *N* with $Q_N = \{ 1, 2, 3 \}$

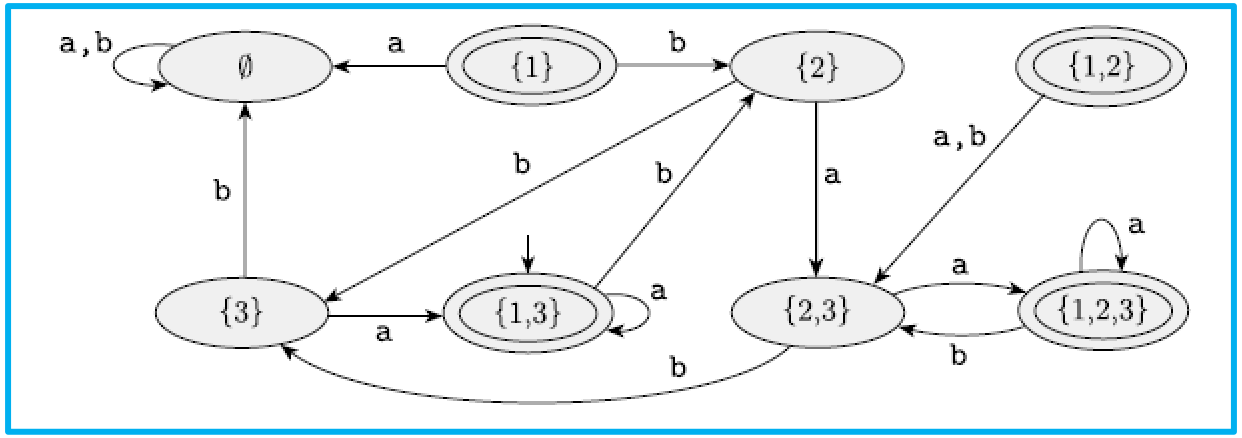


- Corresponding states for DFA *D*
  - $Q_D = \{ \{\}, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\} \}$.
- Determine start state E{1}
  - E({a}) = set of all states that are reachable from a by traveling along ε-arrows, plus a itself
  - Start state: {1,3}
- Determine accept states
  - $F_D = \{ \{1\}, \{1,2\}, \{1,3\}, \{1,2,3\} \}$

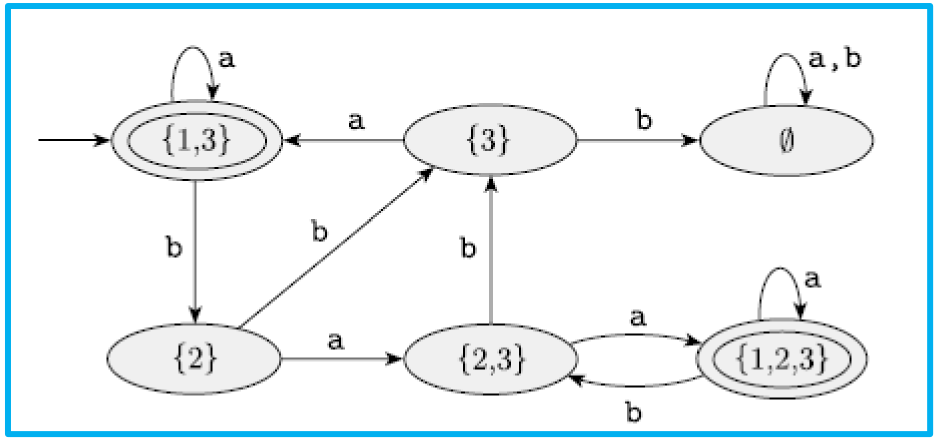
Example 2: NFA to DFA



- Determine *D*'s transition function with table or diagram



- Remove all unreachable states



Alternative Method: NFA to DFA

1. Create NFA state table from the given NFA

2. Create a blank DFA state table under possible input alphabets for the equivalent DFA

3. Mark the start state of DFA qo = E[q] as current state
   - q + states after ε-transitions

4. Find the set of all NFA states that are reachable from the current DFA state

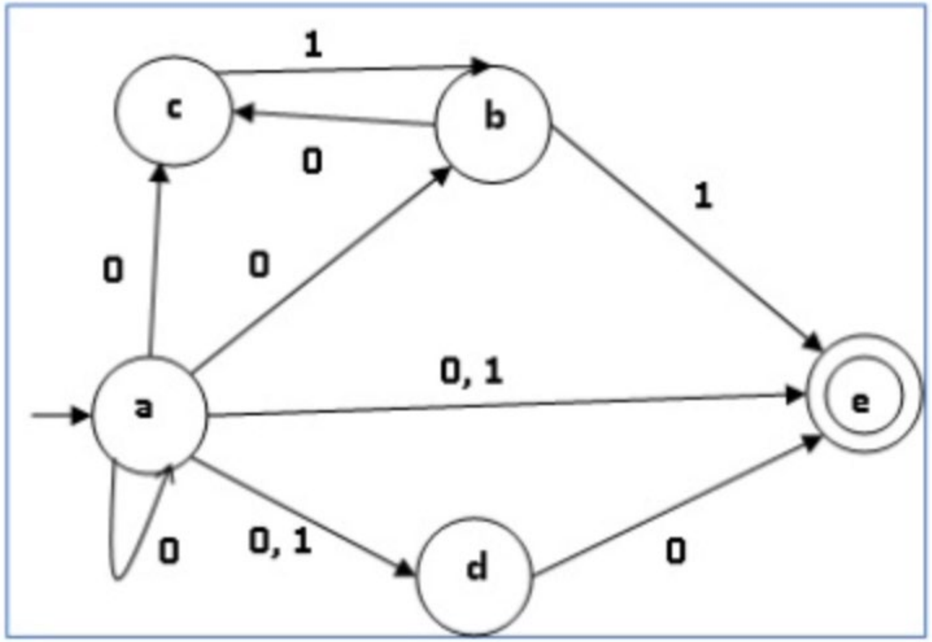
5. Each time we generate a new DFA state under the input alphabet return to step 5

6. When no new edges can be created
   - Draw diagram and mark all reachable accept states

Alternative Method: NFA to DFA

- Step 1: Create state table from given NFA diagram

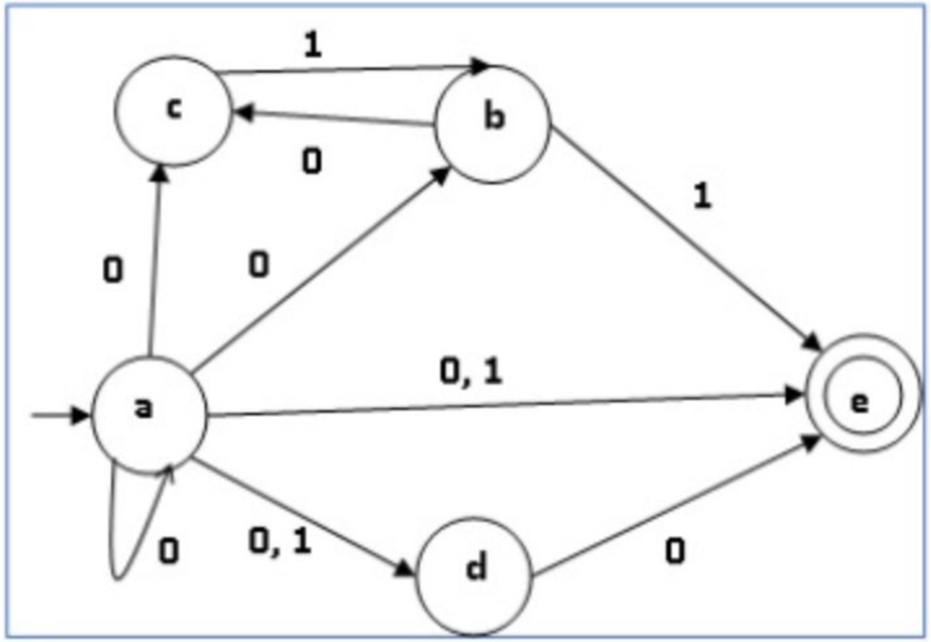
NFA: q	δ(q,0)	δ(q,1)
a	{a,b,c,d,e}	{d,e}
b	{c}	{e}
c	∅	{b}
d	{e}	∅
e	∅	∅



Alternative Method: NFA to DFA

- Steps 2-5: Create DFA table, start state, find all transitions

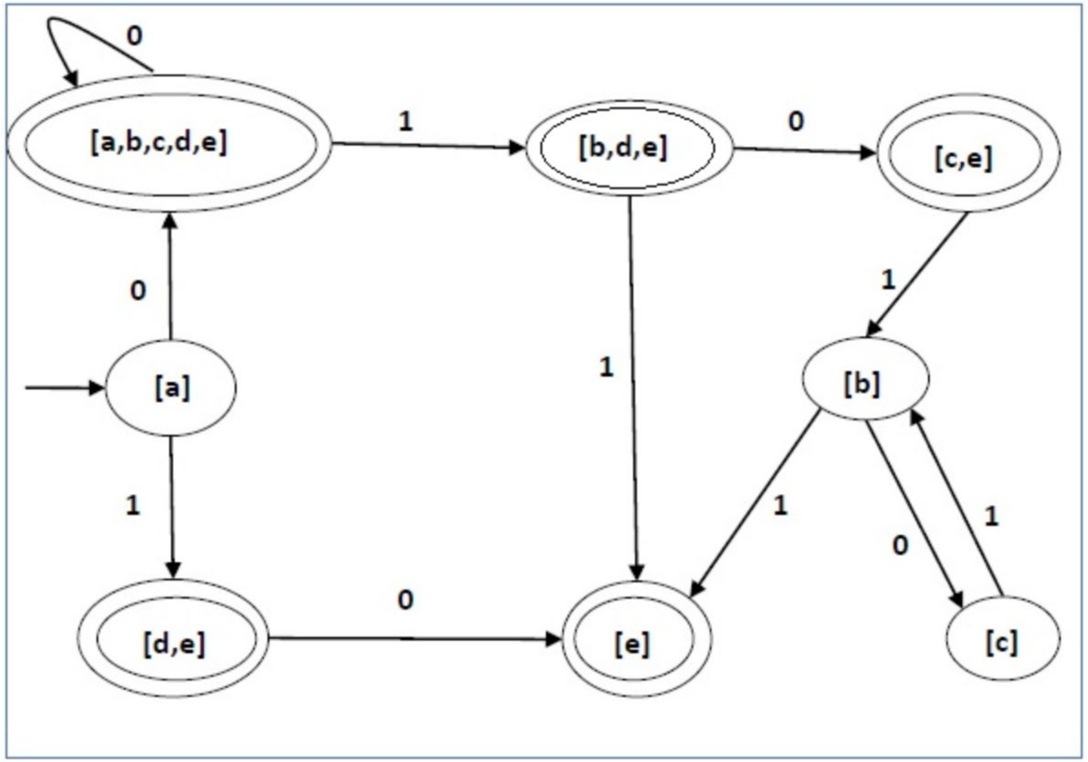
DFA: q	δ(q,0)	δ(q,1)
{a}	{a,b,c,d,e}	{d,e}
{a,b,c,d,e}	{a,b,c,d,e}	{b,d,e}
{d,e}	{e}	∅
{b,d,e}	{c,e}	{e}
{e}	∅	∅
{c,e}	∅	{b}
{b}	{c}	{e}
{c}	∅	{b}



Alternative Method: NFA to DFA

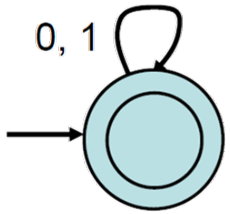
- Steps 6: Draw transition diagram and mark accept states

DFA: q	δ(q,0)	δ(q,1)
{a}	{a,b,c,d,e}	{d,e}
{a,b,c,d,e}	{a,b,c,d,e}	{b,d,e}
{d,e}	{e}	∅
{b,d,e}	{c,e}	{e}
{e}	∅	∅
{c,e}	∅	{b}
{b}	{c}	{e}
{c}	∅	{b}

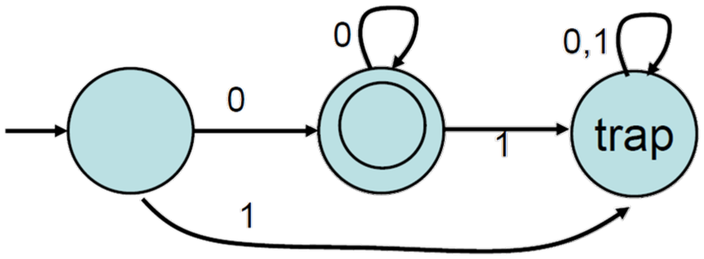


DFA Closure under Concatenation

- Example
  - $\Sigma = \{0,1\}$, $L_1 = \Sigma^*$, $L_2 = \{0\}\{0\}^*$ (just zeros, at least one)
  - L_1L_2 = Strings that end with a block of at least one 0
    - M_1:



M_2:

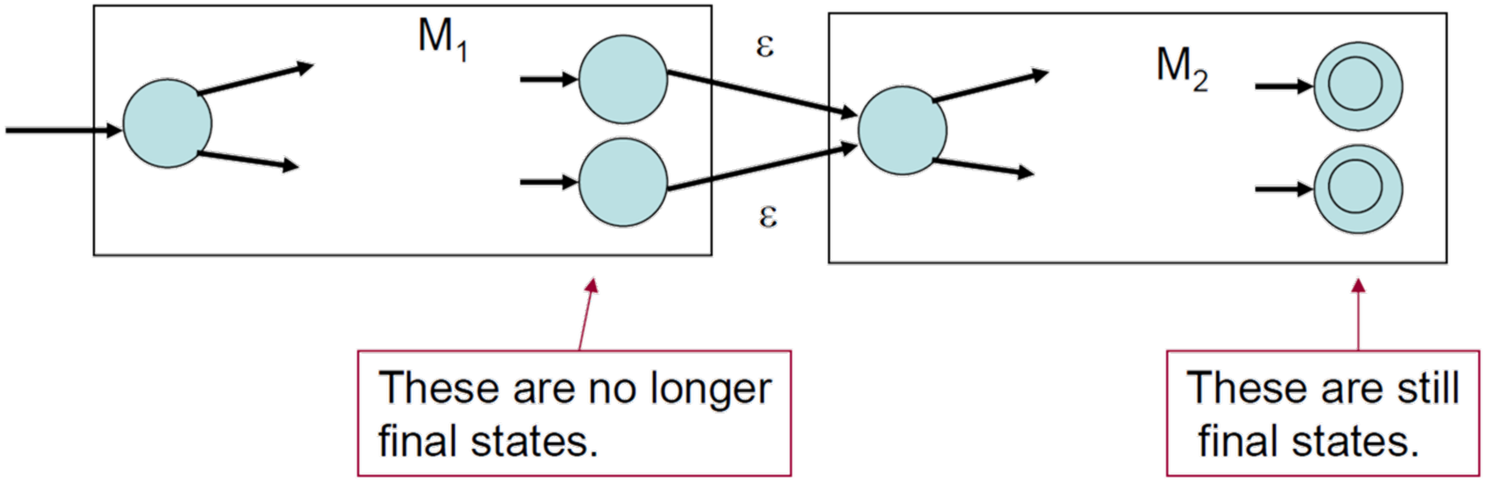


- How to combine?
  - Need to "guess" when to shift to M_2
  - Leads to our next model, Nondeterministic Finite Automata
    - FAs that can guess
- Closure under star operation is an extension of this.

NFA Closure under Concatenation

- $L_3 = L_1 \circ L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$

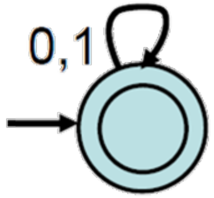
- Start with NFAs M_1 and M_2
  - Start state of M_1 is now the start of M_3
  - Connect ε–transitions from all M_1 accept states to M_2 start state
  - Accept states of M_1 become non-accept states
  - M_3 accepts are M_2 accept states



NFA Closure under Concatenation

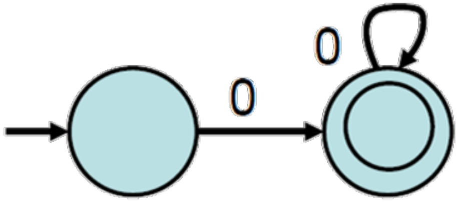
- $L_1 = \{0,1\}^*$
  - Any string

– M_1:



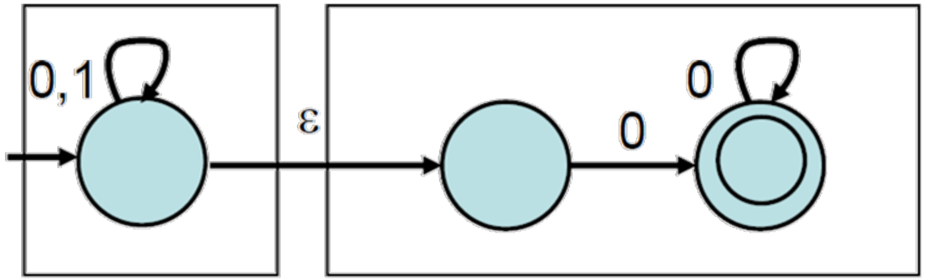
- $L_2 = \{0\}\{0\}^*$
  - String of all zeros
  - At least 1 zero

– M_2:



NFAs

– Now combine:



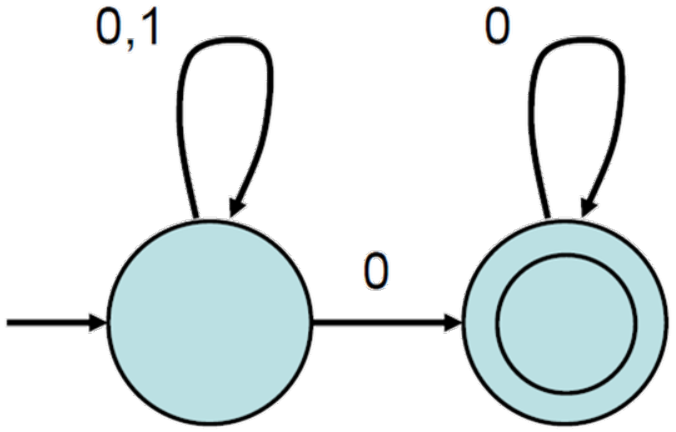
- $L_3 = \{0,1\}^*\{0\}\{0\}^*$
  - String ends in a zero block with at least one zero

NFA Closure Under Concatenation

- Could not show with DFA

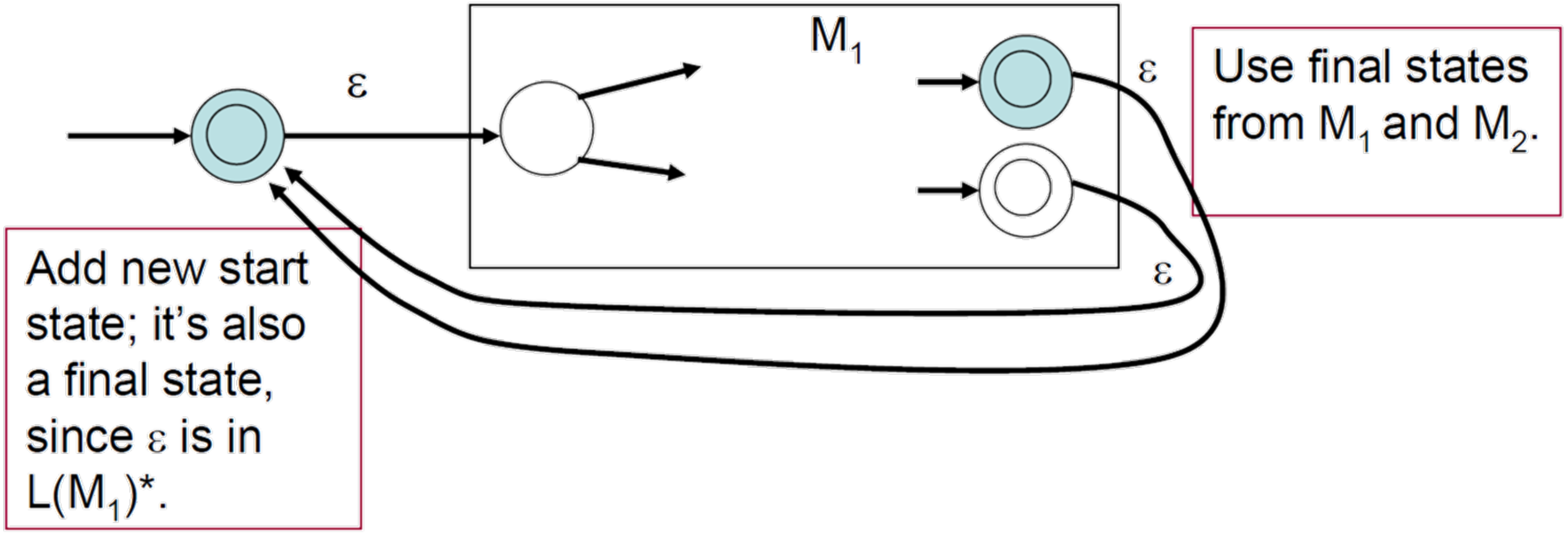
- L = {0,1}*{0}{0}*
  - Strings that consist of a 0 between
  - a binary string of any length and
  - a 0 string of any length.

- NFA can guess when the critical 0 occurs



Closure under Star Operation

- Star Operation
  - $L^{*} = \{x | x = y_1 y_2 \dots yk \; for \; some \; k \geq 0, every \; y \; in \; L\}$
  - Advanced form of concatenation plus ε

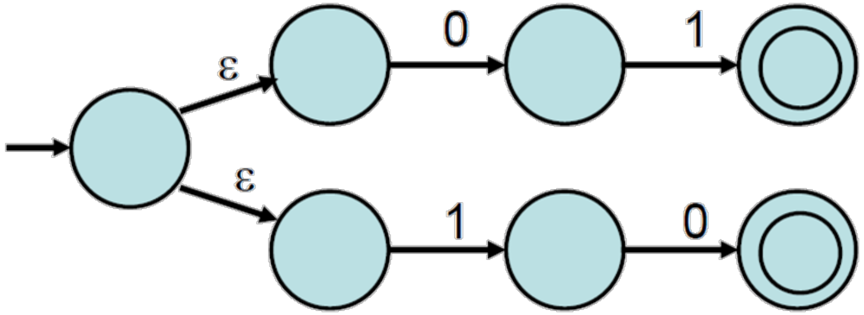


- Proof
  - Start with FA M_1
  - Create NFA M_2
    - $L(M_2) = L(M_1)^*$
- New start state for ε
- Connect accept states to new start state

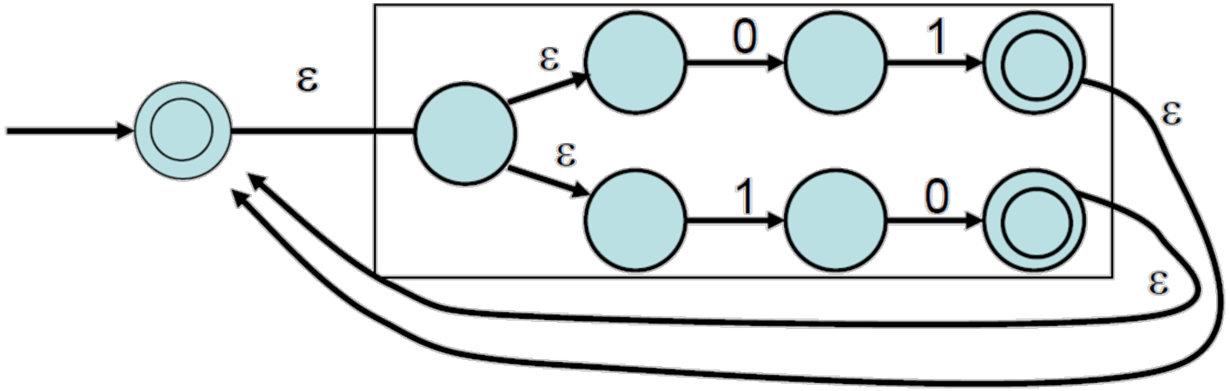
Closure under Star Operation

- Example
  - Σ={ 0, 1 }
  - L_1 = { 01, 10 }
  - $(L_1)^*$ = even-length strings where each pair consists of a 0 and a 1

– M_1:



– Construct M_2:



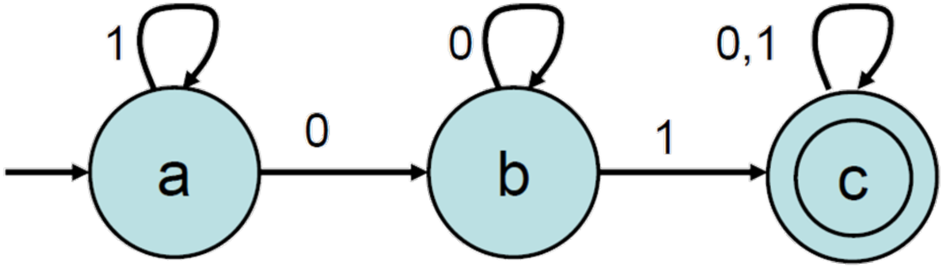
DFA Closure under Union

- Theorem: FA-recognizable languages are close under union

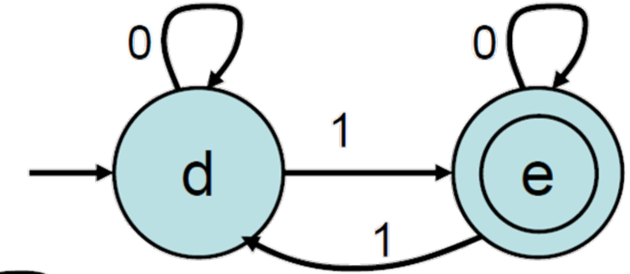
- DFA proof
  - Start with 2 DFAs
  - Create 3rd DFA by running the original to in parallel
  - If either reaches an accepting state, accept

- Example:

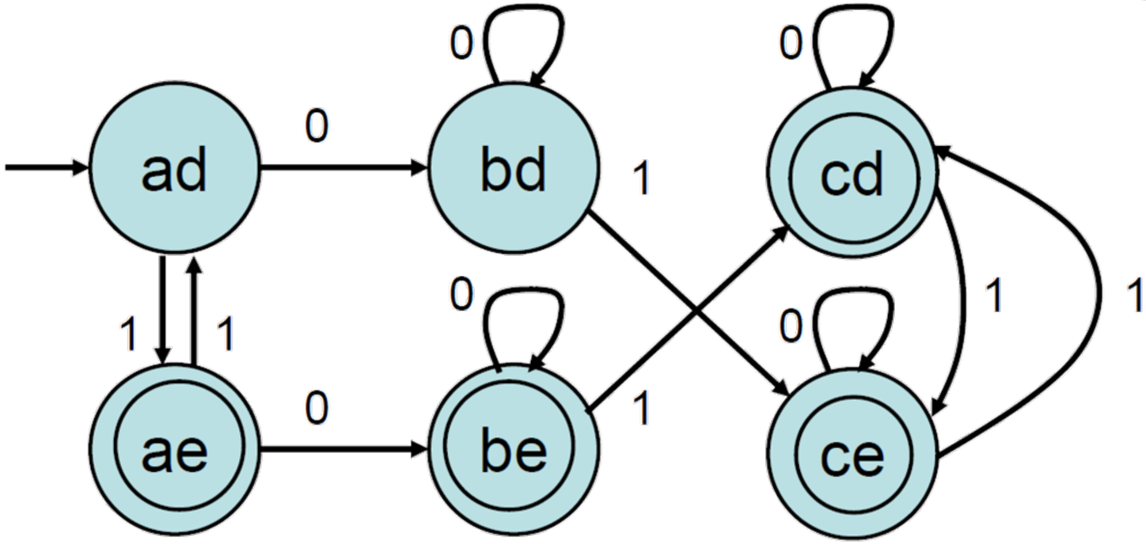
M_1: Substring 01



M_2: Odd number of 1s



M_3: 1



NFA Closure under Union

- NFA proof
  - Start with 2 NFAs
  - Create 3rd by adding a new start state and ε arrows connecting to the 2 original NFAs
- Note: NFAs don't help with Intersection

- Theorem: FA-recognizable languages are closed under union.
- New Proof:
  - Start with NFAs M_1 and M_2.
  - Get another NFA, M_3, with $L(M_3) = L(M_1) \cup L(M_2)$.