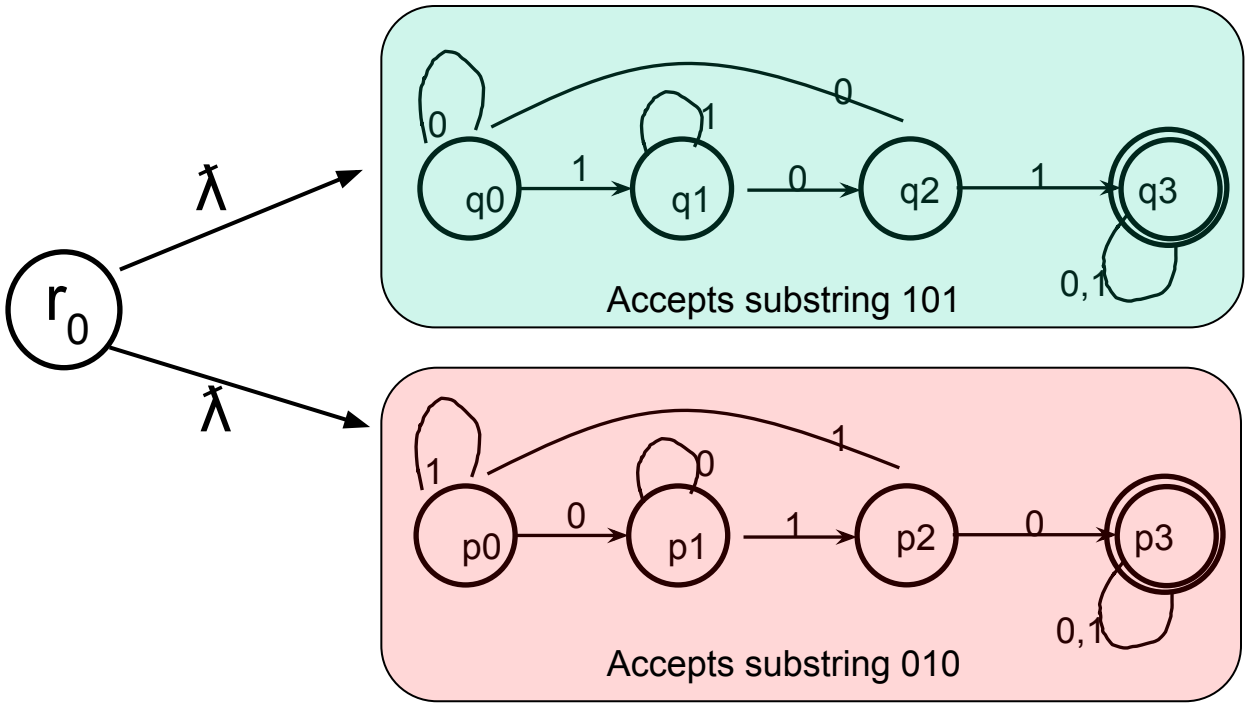
CS 3313
Foundations of Computing:

NFA Review

Extending the NFA model: NFA's With ε-Transitions

- We allow state-to-state transitions on empty string input λ (also denoted ϵ) .
- These transitions are done spontaneously, without looking at the input string.
- A convenience at times, but still only regular languages are accepted.
  - Allowing λ-transitions can make it easier to define and build the automaton
- Analogous to program going to several next states before reading the next input

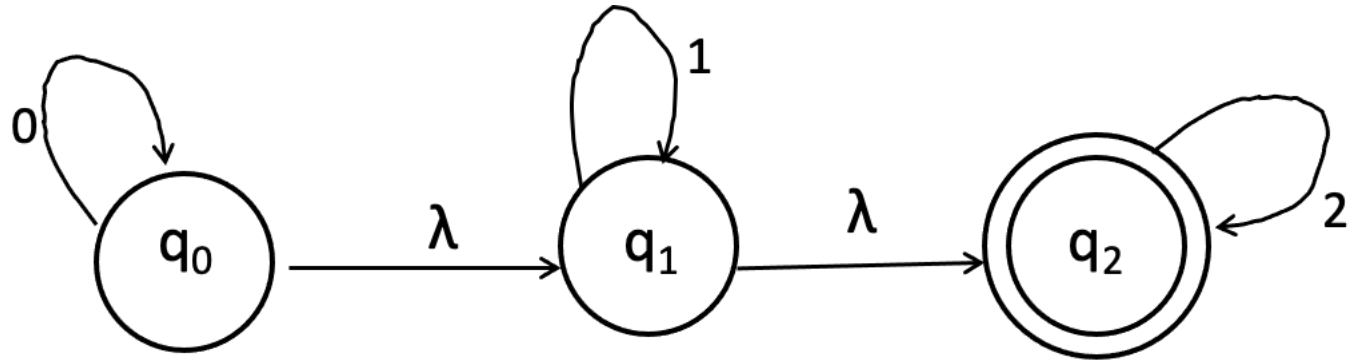
Example: recognizes Substring 101 or 010



From new start state r0,
Start both automaton (green and pink) at the same time
If one of them goes to a final state then accept input

Example 1:

- L = {w | w has 0's followed by 1's followed by 2's}



Any advantage to NFA model with empty string input ?

- w is a string in L1 or L2
  - Construct M1 for L1 and M2 for L2, then on E-transition from start go to both M1 and M2

- w is a string x.y where x is in L1 and y is in L2; x has 00 and y has 11
  - Construct M1 for L1 and M2 for L2, start in M1 and if it goes to final state then start M2.

- What are we doing here…..simplification of the language/problem

Recall Exercise from class:

- Provide an NFA M (with E moves) that accepts the language L over alphabet {0,1,2} where

L = { w | (a) w=x and x has two consecutive 0's or (b) w=y and y has substring 101 and ends with two 2's }

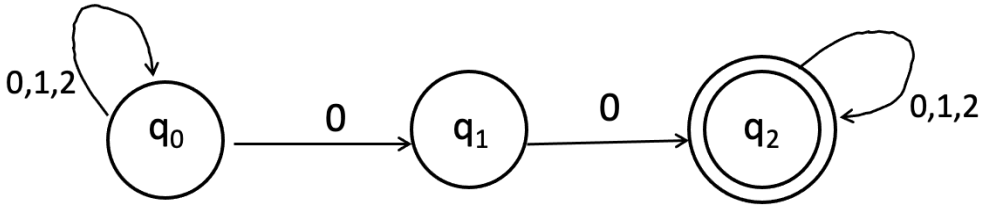
Property (a): build NFA M1 that recognizes substring 00

Property (b): build NFA M2 that recognizes two properties in sequence – substring 101 and then ends with two 2's.

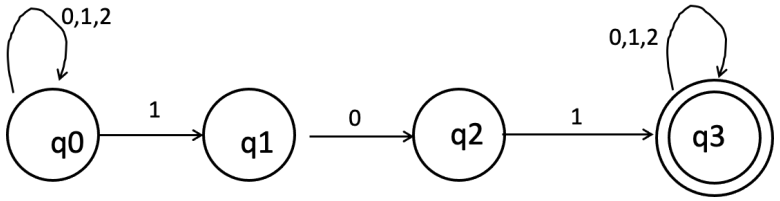
To design NFA M, start M and then go and start both M1 and M2.

Exercise:

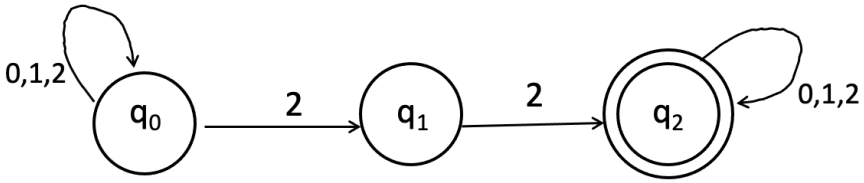
- NFA for property (a)



- NFA for property (b): view it as concatenation of two strings each has specific property/predicate
  - (i) x has substring 101

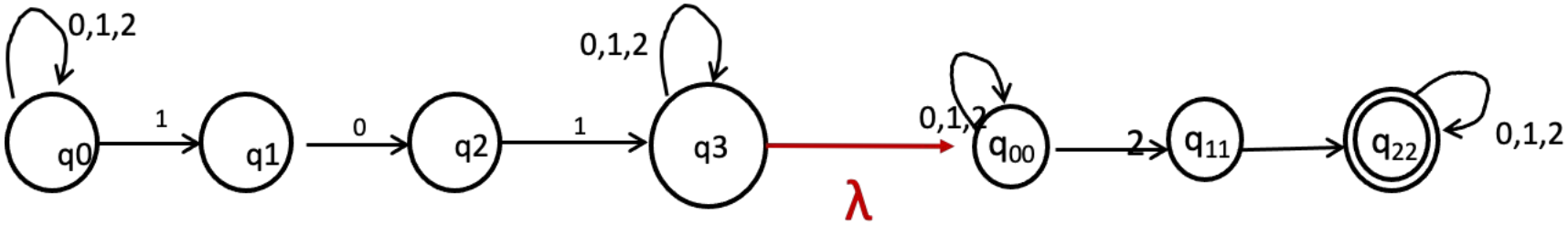


  - (ii) y ends with two 2's.

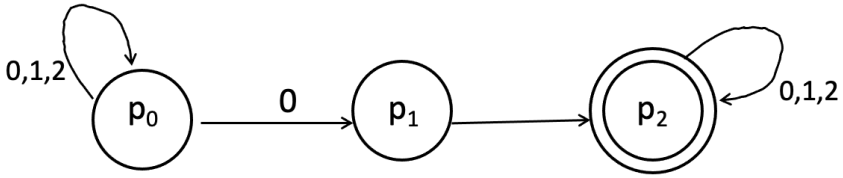


Exercise…

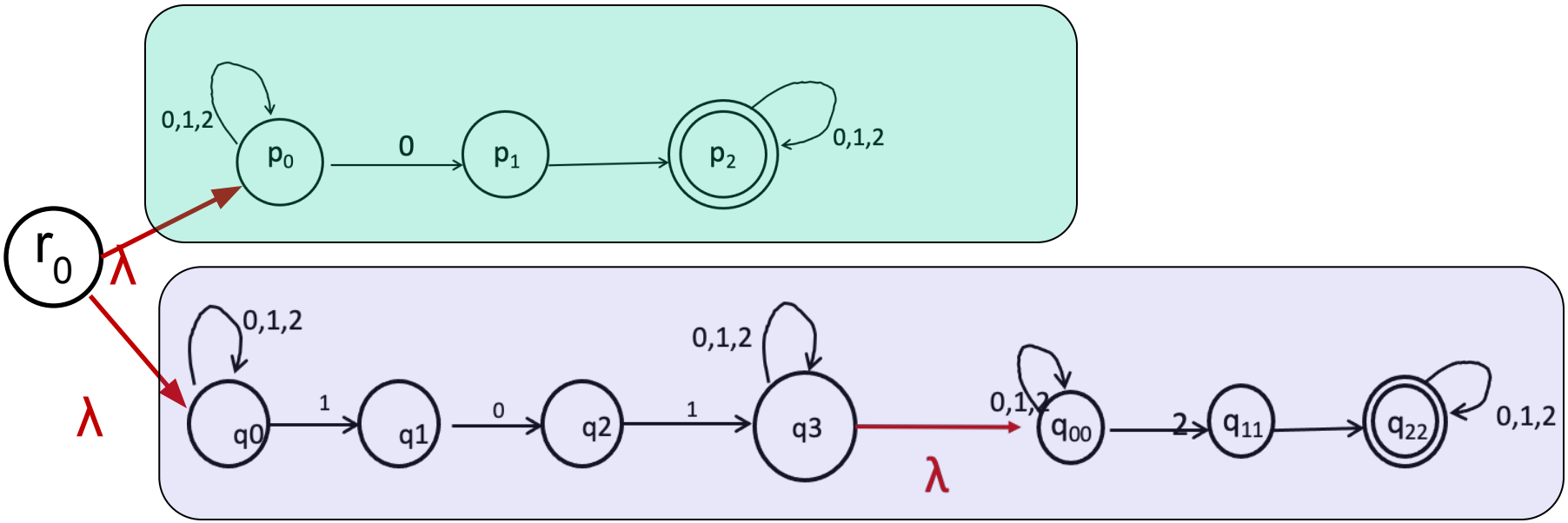
- NFA M2 for property (b): connect the two ”in series”
  - Relabel the states



- Recall NFA M1 for property (a) with new labels for states



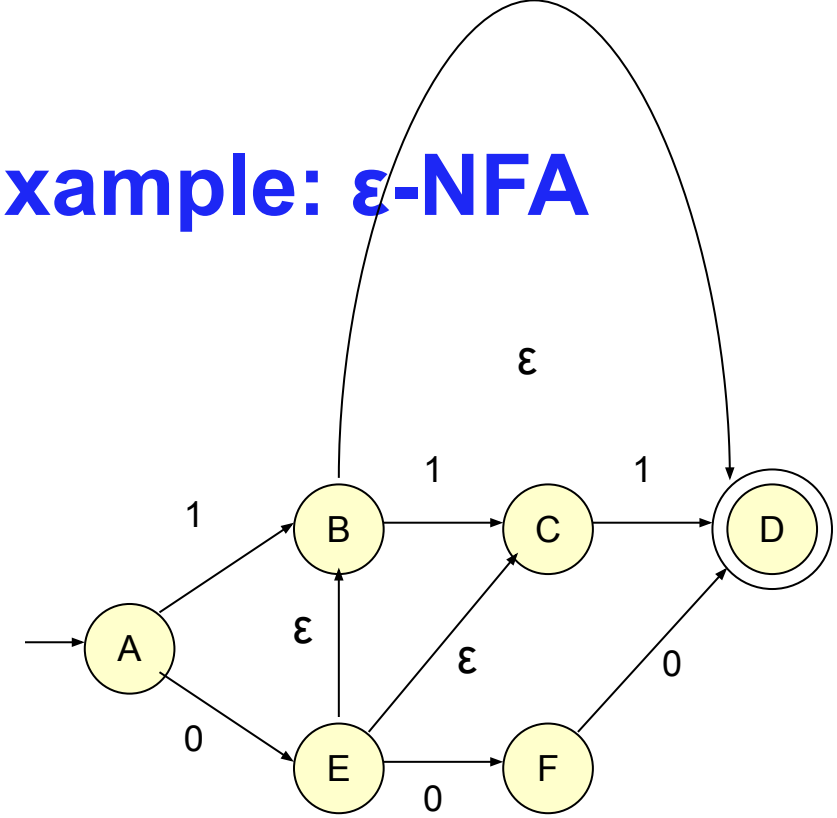
Connecting M1 and M2



Important Observation:
Paths in the NFA and Concept of E-Closure

- A path from state p to state q is labelled with symbols from alphabet OR labeled with empty string
- To transform an NFA with E-moves to an NFA (or DFA) without E-moves, of particular interest are paths labeled with empty string
  - An edge labeled with empty string implies from a state q, we can go to another state p without reading an input
- **Definition:** E-closure of a state = Path where all edges are labeled with empty string

Example: ε-NFA

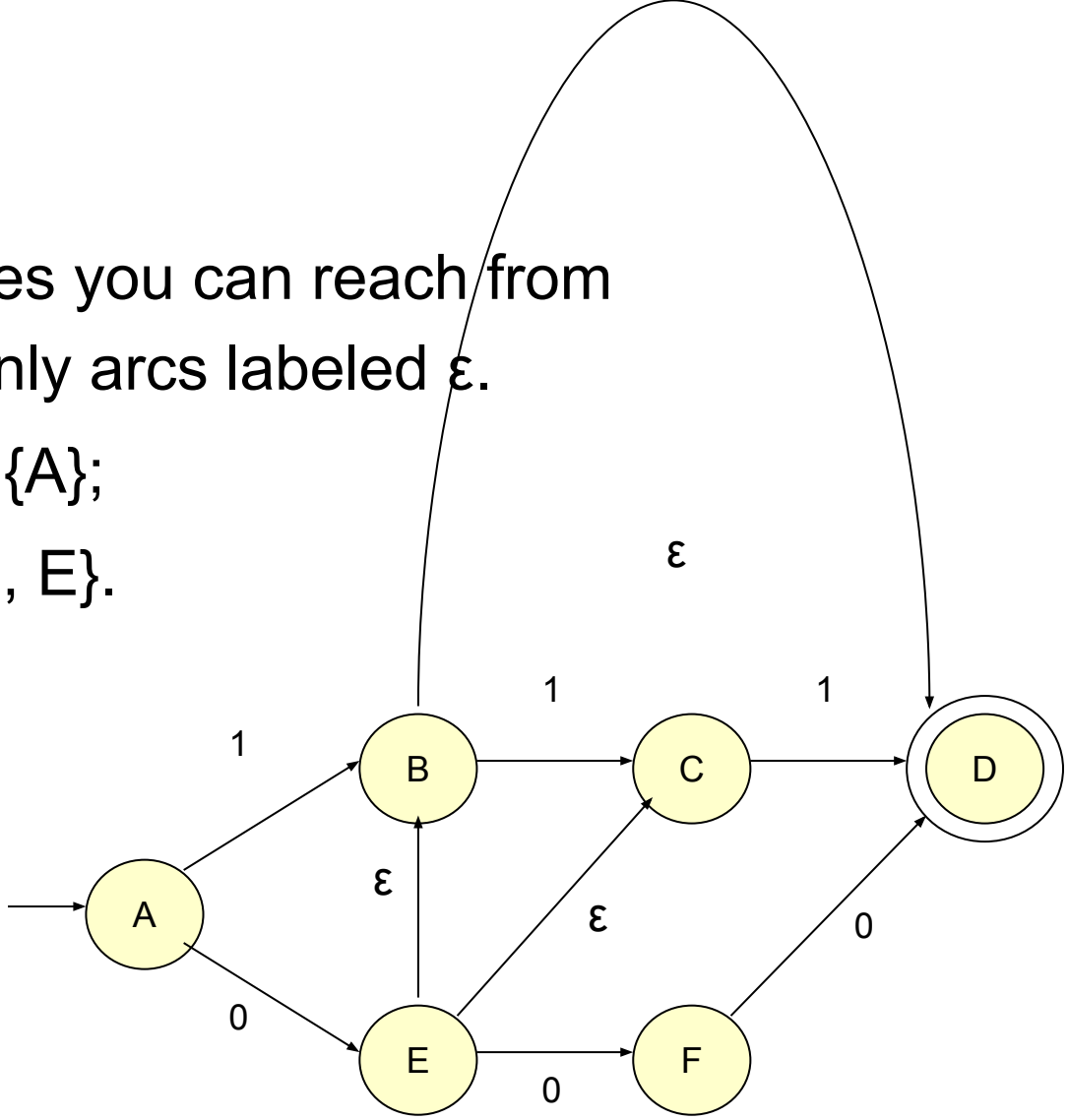


		0	1	ε
→	A	{E}	{B}	∅
	B	∅	{C}	{D}
	C	∅	{D}	∅
*	D	∅	∅	∅
	E	{F}	∅	{B, C}
	F	{D}	∅	∅

Closure of States

- CL(q) = set of states you can reach from state q following only arcs labeled ε.
- Example: CL(A) = {A};

  CL(E) = {B, C, D, E}.



- Closure of a set of states = union of the closure of each state.

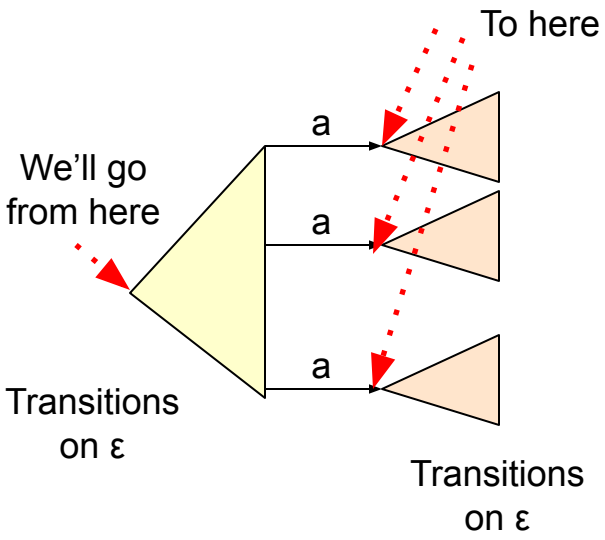
Why do we need E-closure…Extended Delta

- We need to extend transition function to apply over strings
- Intuition: $\hat{\delta}(q, w)$ is the set of states you can reach from q following a path labeled w.
- Basis: $\hat{\delta}(q, \varepsilon) = CL(q)$.
- Induction: $\hat{\delta}(q, xa)$ is computed by:

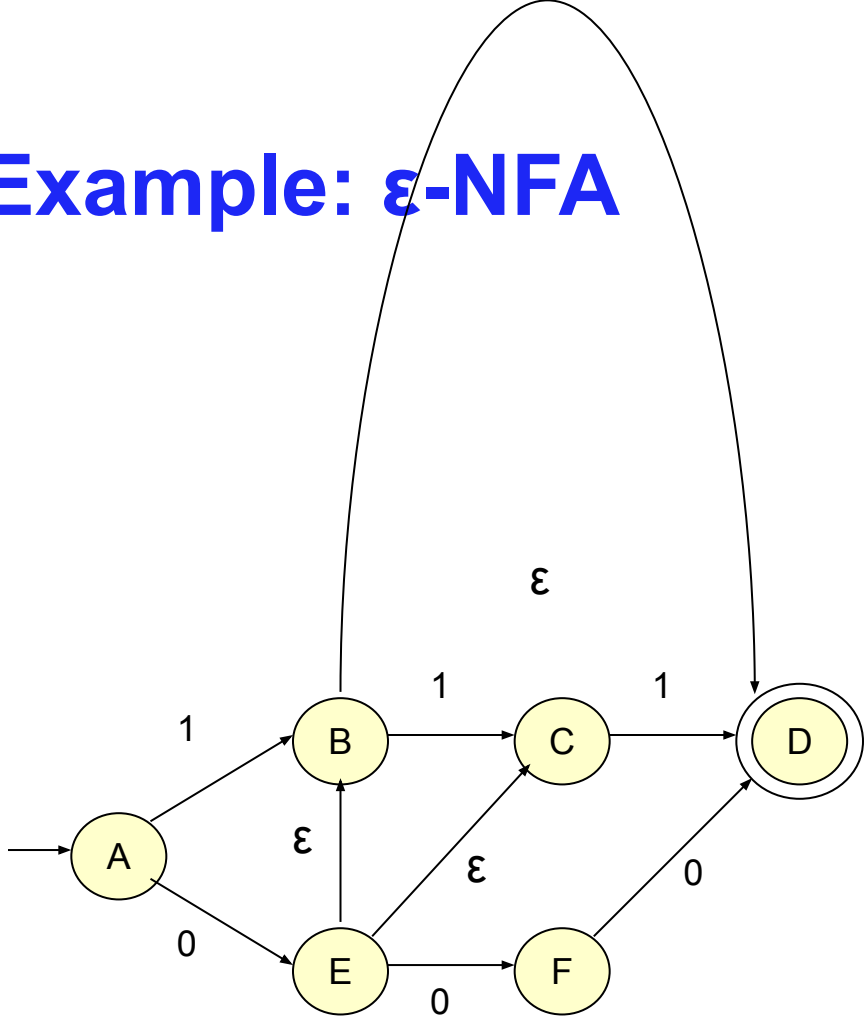
1. Start with $\hat{\delta}(q, x) = S$.
2. Take the union of $CL(\delta(p, a))$ for all p in S.

More in lecture session….

Picture of ε-Transition Removal

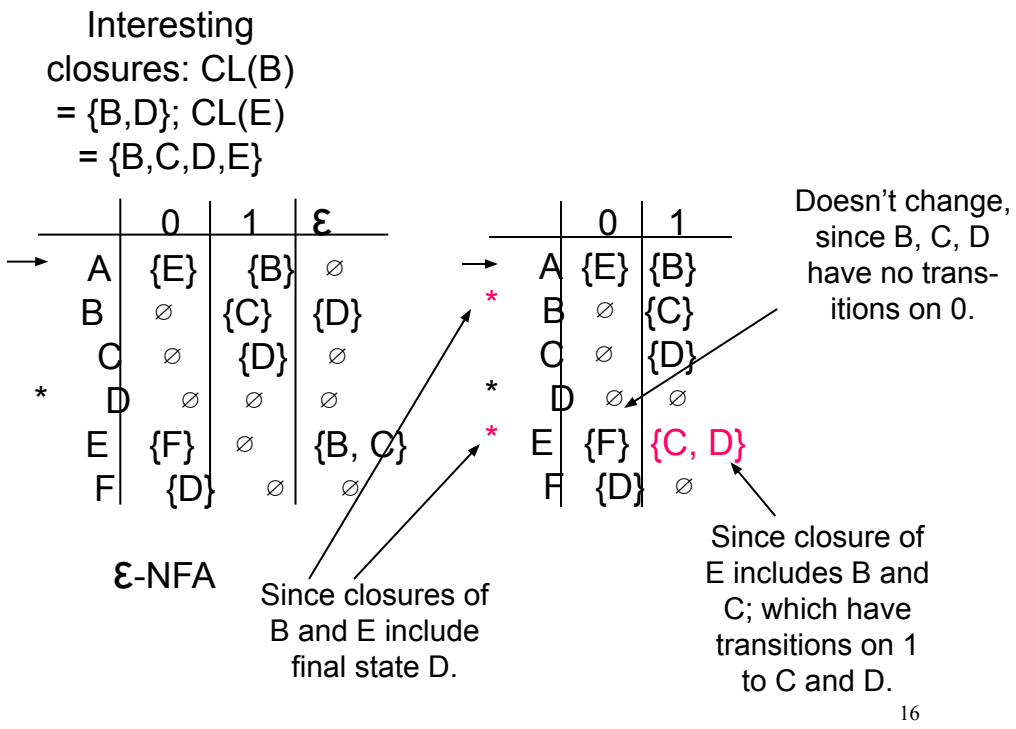


Example: ε-NFA



		0	1	ε
→	A	{E}	{B}	∅
	B	∅	{C}	{D}
	C	∅	{D}	∅
*	D	∅	∅	∅
	E	{F}	∅	{B, C}
	F	{D}	∅	∅

Example: ε-NFA-to-NFA



Breakout Exercise: (1) Remove E-transitions, (2) Convert Resulting NFA to DFA