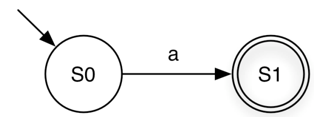
CMSC 330: Organization of Programming Languages

Finite Automata, con't.

Reducing Regular Expressions to NFAs

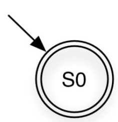
- Goal: Given regular expression e, construct NFA <e> = $(\Sigma, Q, q_0, F, \delta)$ that accepts the same language
  - invariant: $|F| = 1$ in our NFAs
- Base case: a



  - <a> = ({a}, {S0, S1}, S0, {S1}, {(S0, a, S1)})

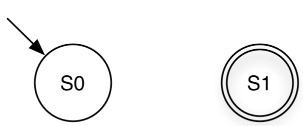
Reduction (cont'd)

- Base case: ε



  - <ε> = ({ε}, {S0}, S0, {S0}, ∅)

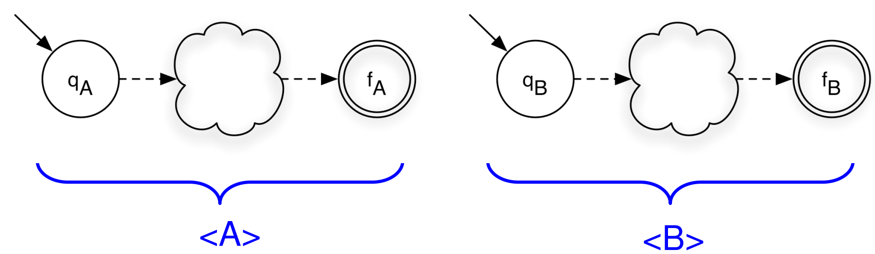
- Base case: ∅



  - <∅> = (∅, {S0, S1}, S0, {S1}, ∅)

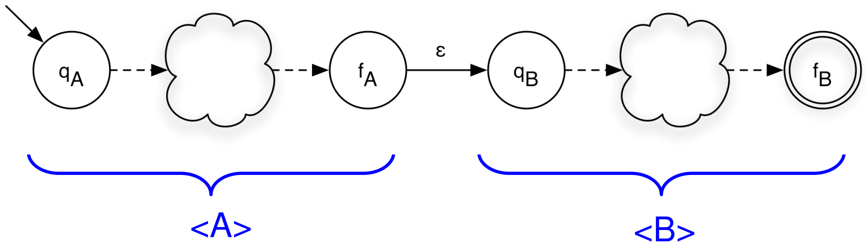
Reduction (cont'd)

- Induction: AB



Reduction (cont'd)

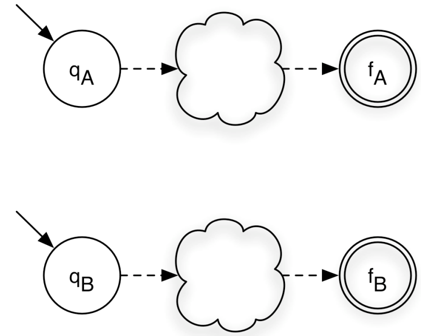
- Induction: AB



- $\langle A\rangle = (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- $\langle B\rangle = (\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B)$
- $\langle AB\rangle = (\Sigma_A \cup \Sigma_B, Q_A \cup Q_B, q_A, \{f_B\}, \delta_A \cup \delta_B \cup \{(f_A, \varepsilon, q_B)\})$

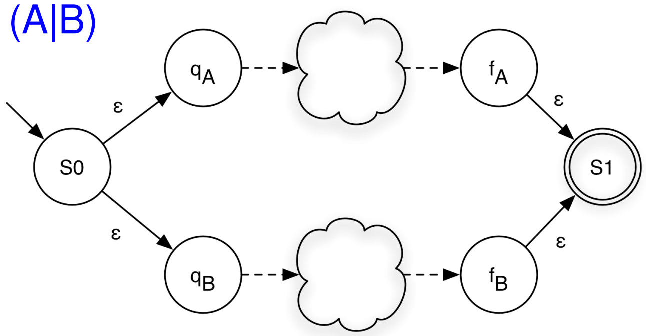
Reduction (cont'd)

- Induction: (A|B)



Reduction (cont'd)

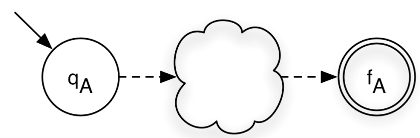
- Induction: (A|B)



- $\langle A\rangle = (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- $\langle B\rangle = (\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B)$
- $\langle (A|B)\rangle = (\Sigma_A \cup \Sigma_B, Q_A \cup Q_B \cup \{S0,S1\}, S0, \{S1\},$
  $\delta_A \cup \delta_B \cup \{(S0,\varepsilon,q_A), (S0,\varepsilon,q_B), (f_A,\varepsilon,S1), (f_B,\varepsilon,S1)\})$

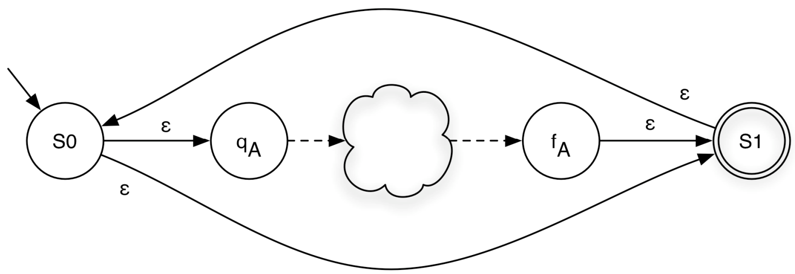
Reduction (cont'd)

- Induction: A*



Reduction (cont'd)

- Induction: A*



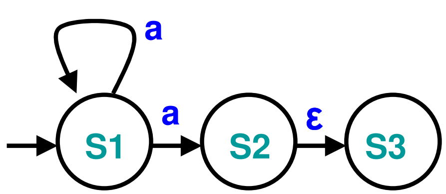
- <A> = $(\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- <A*> = $(\Sigma_A, Q_A \cup \{S0,S1\}, S0, \{S1\},$
  $\delta_A \cup \{(f_A,\varepsilon,S1), (S0,\varepsilon,q_A), (S0,\varepsilon,S1), (S1,\varepsilon,S0)\})$

Reduction Complexity

- Given a regular expression A of size n...
  - size = # of symbols + # of operations
- How many states does <A> have?
  - O(n)
    that's pretty good!
- NFA to DFA reduction
  - intuition: Build DFA where each DFA state represents a set of NFA states
  - given NFA with n states, DFA may have 2^n states
  - not so good, since DFAs are what we can implement easily

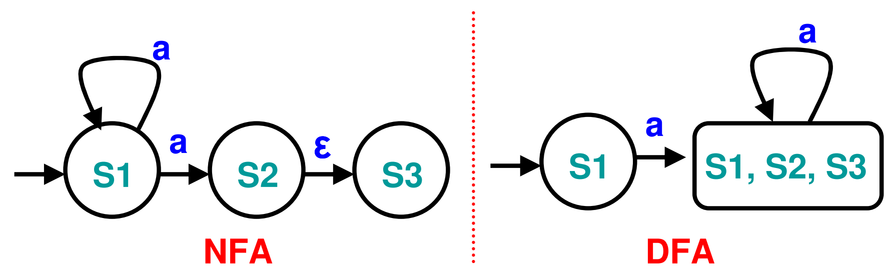
How an NFA Works

- When an NFA processes a string it may be in several possible states
  - Multiple transitions with same label
  - ε-transitions
- Example
  - After processing "a" NFA may be in states
    S1
    S2
    S3



Reducing NFA to DFA

- NFA may be reduced to DFA by explicitly tracking the set of NFA states
- Intuition: build DFA where each DFA state represents a set of NFA states
- Example



Reducing NFA to DFA (cont.)

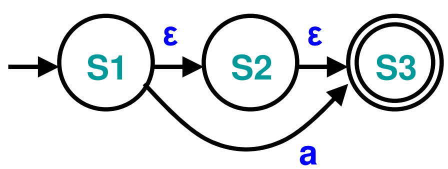
- Reduction applied using the subset algorithm
  - DFA state is a subset of set of all NFA states
- Algorithm
  - Input
    - NFA $(\Sigma, Q, q_0, F_n, \delta_n)$
  - Output
    - DFA $(\Sigma, R, r_0, F_d, \delta_d)$
  - Using
    - ε-closure(p)
    - move(p, a)

ε-transitions and ε-closure

- We say $p \underset{\varepsilon}{\rightarrow} q$ if it is possible to go from state p to state q by taking only ε-transitions
  - If $\exists$ p, p_1, p_2, … p_n, q $\in$ Q, such that $\{p,\varepsilon,p_1\} \in \delta$, $\{p_1,\varepsilon,p_2\} \in \delta$, … , $\{p_n,\varepsilon,q\} \in \delta$
- ε-closure(p) is defined as the set of states reachable from p using ε-transitions alone
  - Set of states q such that $p \underset{\varepsilon}{\rightarrow} q$
  - ε-closure(p) = $\{q \mid p \underset{\varepsilon}{\rightarrow} q\}$
  - Note:
    - ε-closure(p) always includes p
    - ε-closure() may be applied to set of states (take union)

ε-closure: Example 1

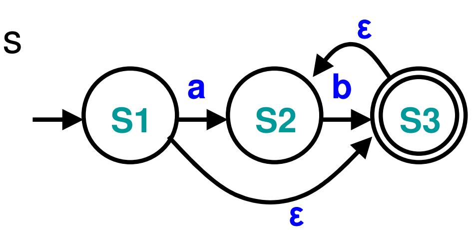
- Following NFA contains
  - S1 $\underset{\varepsilon}{\rightarrow}$ S2
  - S2 $\underset{\varepsilon}{\rightarrow}$ S3
  - S1 $\underset{\varepsilon}{\rightarrow}$ S3



- ε-closures
  - ε-closure(S1) = **{S1, S2, S3}**
  - ε-closure(S2) = **{S2, S3}**
  - ε-closure(S3) = **{S3}**
  - ε-closure({ S1, S2 }) = **{S1, S2, S3}** $\cup$ **{S2, S3}**

ε-closure: Example 2

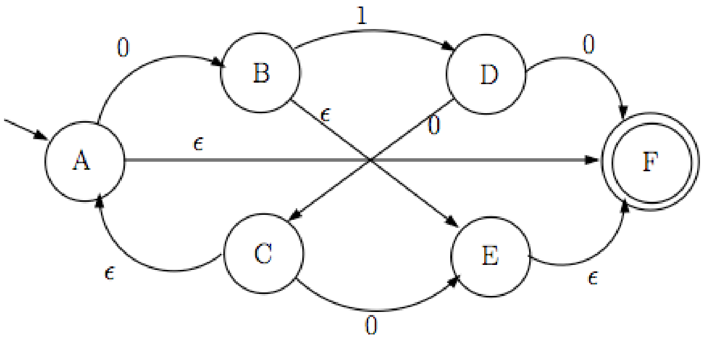
- Following NFA contains
  - S1 $\underset{\varepsilon}{\rightarrow}$ S3
  - S3 $\underset{\varepsilon}{\rightarrow}$ S2
  - S1 $\underset{\varepsilon}{\rightarrow}$ S2



- ε-closures
  - ε-closure(S1) = **{S1, S2, S3}**
  - ε-closure(S2) = **{S2}**
  - ε-closure(S3) = **{S2, S3}**
  - ε-closure({ S2,S3 }) = **{S2}** $\cup$ **{S2, S3}**

ε-closure: Practice

- Find ε-closures for following NFA



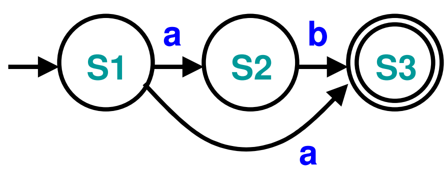
- Find ε-closures for the NFA you construct for the regular expression (0|1*)111(0*|1)

Calculating move(p, a)

- move(p, a) is defined as the set of NFA states reachable from p using exactly one transition on a
  - Set of all states q such that {p, a, q} ∈ δ
  - move(p,a) = {q | {p, a, q} ∈ δ}
  - Note that move(p,a) may be empty Ø, if there is no transition from p with label a

move(a,p) : Example 1

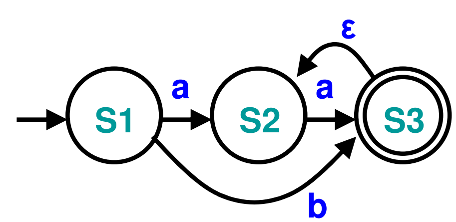
- Following NFA
  - Σ = {a, b}



- Move
  - move(S1, a) = **{S2, S3}**
  - move(S1, b) = **Ø**
  - move(S2, a) = **Ø**
  - move(S2, b) = **{S3}**
  - move(S3, a) = **Ø**
  - move(S3, b) = **Ø**

move(a,p) : Example 2

- Following NFA
  - Σ = {a, b}



- Move
  - move(S1, a) = **{S2}**
  - move(S1, b) = **{S3}**
  - move(S2, a) = **{S3}**
  - move(S2, b) = **Ø**
  - move(S3, a) = **Ø**
  - move(S3, b) = **Ø**

NFA → DFA Reduction Algorithm

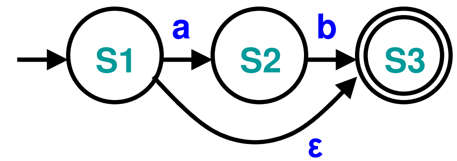
- Input NFA (Σ, Q, q_0, F_n, δ_n), output DFA (Σ, R, r_0, F_d, δ_d)
- Algorithm

Let r_0 = ε-closure(q_0), add it to R	// DFA start state
While ∃ an unmarked state r ∈ R	// process DFA state r
Mark r	// each state visited once
For each a ∈ Σ	// for each letter a
Let S = {s | q ∈ r & move(q, a) = s}	// states reached via a
Let e = ε-closure(S)	// states reached via ε
If e ∉ R	// if state e is new
Let R = e ∪ R	// add e to R (unmarked)
Let $\delta_d = \delta_d \cup$ {r, a, e}	// add transition r→e
Let F_d = {r | ∃ s ∈ r with s ∈ F_n}	// final if include state in F_n

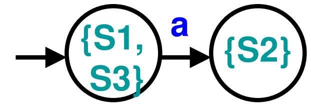
NFA → DFA Example 1

- Start = ε-closure(S1) = { {S1,S3} }
- R = { {S1,S3} }
- r ∈ R = {S1,S3}
- Move({S1,S3}, a}) = {S2}
  - e = ε-closure({S2}) = {S2}
  - R = R ∪ {S2} = { {S1,S3}, {S2} }
  - δ = δ ∪ {{S1,S3}, a, {S2}}
- Move({S1,S3}, b}) = Ø

NFA



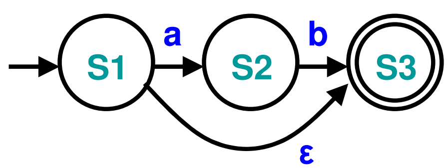
DFA



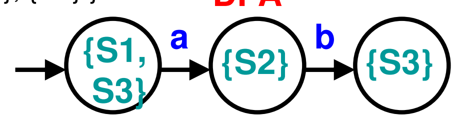
NFA → DFA Example 1 (cont.)

- R = { {S1,S3}, {S2} }
- r ∈ R = {S2}
- Move({S2}, a}) = Ø
- Move({S2}, b}) = {S3}
  - e = ε-closure({S3}) = {S3}
  - R = R ∪ {S3} = { {S1,S3}, {S2}, {S3} }
  - δ = δ ∪ {{S2}, b, {S3}}

NFA



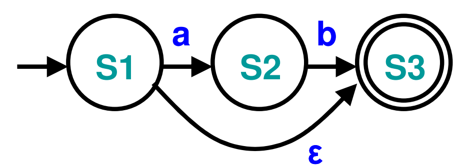
DFA



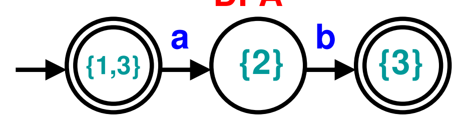
NFA → DFA Example 1 (cont.)

- R = { {S1,S3}, {S2}, {S3} }
- r ∈ R = {S3}
- Move({S3}, a}) = Ø
- Move({S3}, b}) = Ø
- F_d = {{S1,S3}, {S3}}
  - Since S3 ∈ F_n
- Done!

NFA

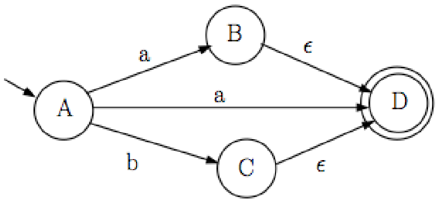


DFA

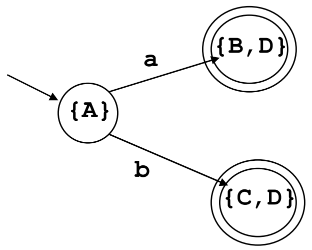


NFA → DFA Example 2

- NFA



- **DFA**



R = { {A}, {B,D}, {C,D}