CMSC 330: Organization of Programming Languages

Theory of Regular Expressions
DFAs and NFAs

Reminders

- Project 1 due Sep. 24
- Homework 1 posted
- Exam 1 on Sep. 25

- Exam topics list posted
- Practice homework (and solutions) posted

Previous Course Review

- {s | s defined} means the set of string s such that s is chosen or defined as given
- $s \in A$ means s is an element of the set A
- De Morgan's Laws:

$$(A \cap B)^C = A^C \cup B^C$$
$$(A \cup B)^C = A^C \cap B^C$$

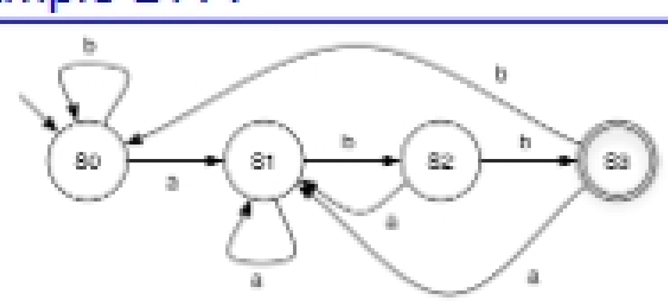
- There exists and for all symbols
- Etc…

Review

- Basic parts of a regular expression?
  - concatenation, |, *, ε, Ø, {a}
- What does a DFA do?

- Basic parts of a DFA?
  - alphabet, set of states, start state, final states, transition function $(\Sigma, Q, q_0, F, \delta)$

Example DFA



- S0 = "Haven't seen anything yet" OR "seen zero or more b's" OR "Last symbol seen was a b"
- S1 = "Last symbol seen was an a"
- S2 = "Last two symbols seen were ab"
- S3 = "Last three symbols seen were abb"

- Language?
- (a|b)*abb

Notes about the DFA definition

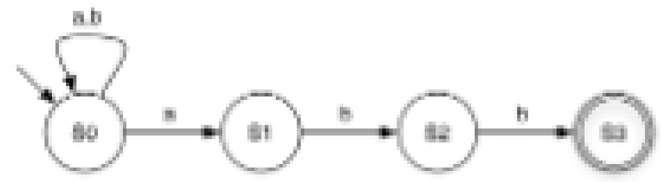
- Can not have more than one transition leaving a state on the same symbol
  - the transition function must be a valid function)

- Can not have transitions with no or empty labels
  - the transitions must be labeled by alphabet symbols

Nondeterministic Finite Automata (NFA)

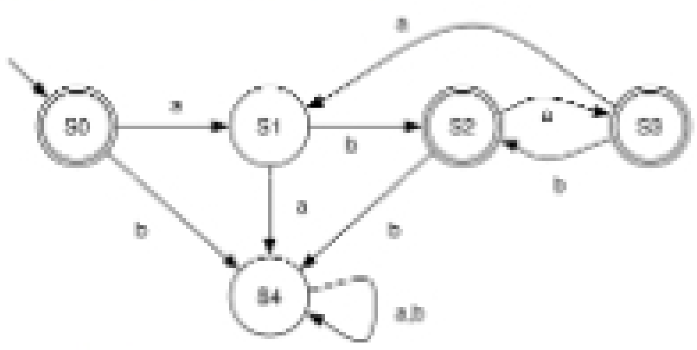
- An *NFA* is a 5-tuple (Σ, Q, q_0, F, δ) where
  - Σ is an alphabet
  - Q is a nonempty set of states
  - $q_0 \in Q$ is the start state
  - $F \subseteq Q$ is the set of final states
    - There may be 0, 1, or many
  - $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ specifies the NFA's transitions
    - Transitions on ε are allowed – can optionally take these transitions without consuming any input
    - Can have more than one transition for a given state and symbol
- An NFA accepts s if there is *at least one* path from its start to final state on s

NFA for (a|b)*abb



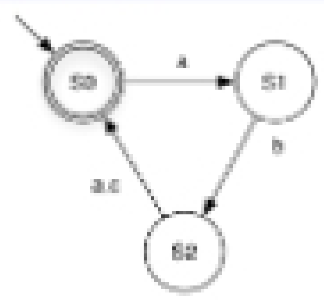
- ba
  - Has paths to either S0 or S1
  - Neither is final, so rejected
- babaabb
  - Has paths to different states
  - One leads to S3, so accepted

Another example DFA



- Language?
- (ab|aba)*

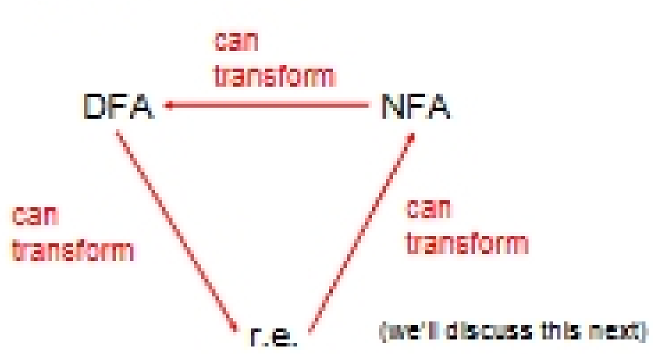
NFA for (ab|aba)*



- aba
  - Has paths to states S0, S1
- ababa
  - Has paths to S0, S1
  - Need to use ε-transition

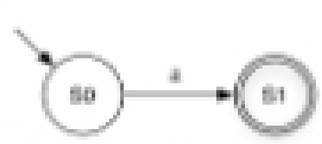
Relating R.E.'s to DFAs and NFAs

- Regular expressions, NFAs, and DFAs accept the same languages!



Reducing Regular Expressions to NFAs

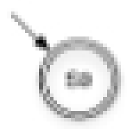
- Goal: Given regular expression e, construct NFA: <e> = (Σ, Q, q_0, F, δ)
  - Remember r.e. defined recursively from primitive r.e. languages
  - Invariant: |F| = 1 in our NFAs
    - Recall F = set of final states
- Base case: a



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

Reduction (cont'd)

- Base case: ε



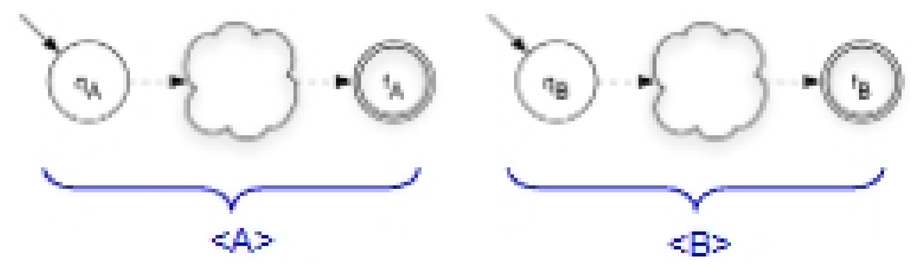
<ε> = (ε, {S0}, S0, {S0}, ∅)

- Base case: ∅

<∅> = (∅, {S0, S1}, S0, {S1}, ∅)

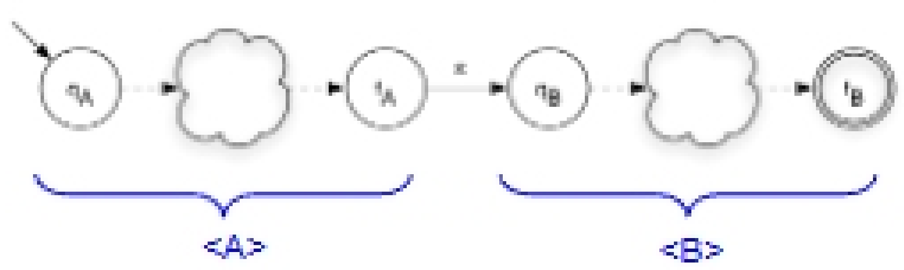
Reduction (cont'd)

- Induction: AB



Reduction (cont'd)

- Induction: AB



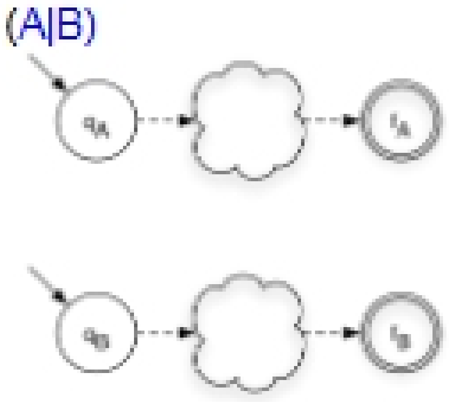
- <A> = $(\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- <B> = $(\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B)$
- <AB> = $(\Sigma_A \cup \Sigma_B, Q_A \cup Q_B, q_A, \{f_B\}, \delta_A \cup \delta_B \cup \{(f_A, \varepsilon, q_B)\})$

Practice

- Draw the NFA for these regular expressions using exactly the reduction method:
  - ab
  - hello
- Write the formal (5-tuple) NFA for the same regular expressions

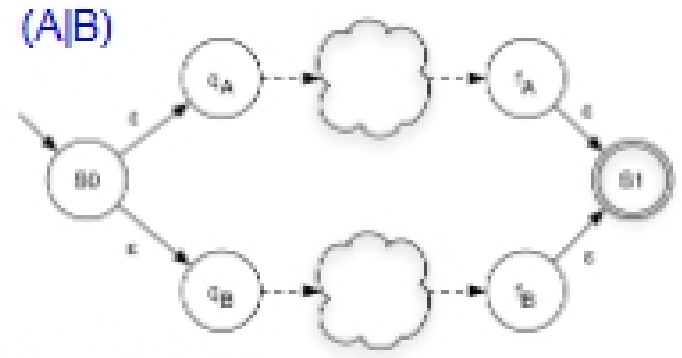
Reduction (cont'd)

- Induction: (A|B)



Reduction (cont'd)

- Induction: (A|B)



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