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 ∈ A means s is an element of the set A
- De Morgan's Laws:

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

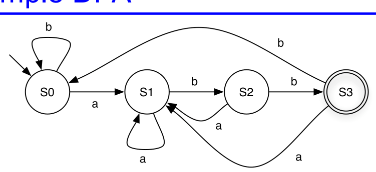
- There exists and for all symbols
- Etc…

Review

- Basic parts of a regular expression?
  `concatenation, |, *, ε, ∅, {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

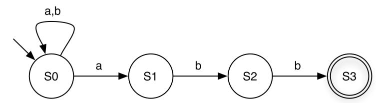
- Can not have more than one transition leaving a state on the same symbol
  - the transition function must be a valid function)

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

Nondeterministic Finite Automata (NFA)

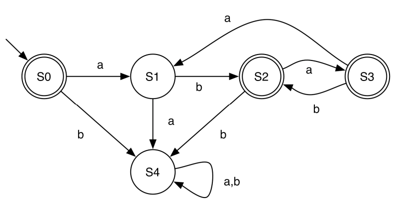
- An *NFA* is a 5-tuple $(\Sigma, Q, q_0, F, \delta)$ where
  - Σ is an alphabet
  - Q is a nonempty set of states
  - $q_0 \in Q$ is the start state
  - $F \subseteq Q$ is the set of final states
    - There may be 0, 1, or many
  - $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ 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



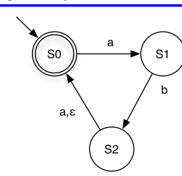
- ba
  - Has paths to either S0 or S1
  - Neither is final, so rejected
- babaabb
  - Has paths to different states
  - One leads to S3, so accepted

Another example DFA



- Language?
- (ab|aba)*

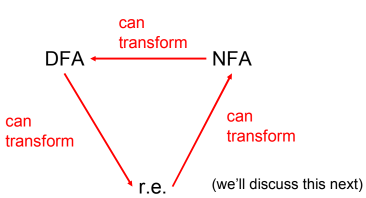
NFA for (ab|aba)*



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

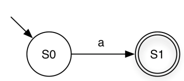
Relating R.E.'s to DFAs and NFAs

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



Reducing Regular Expressions to NFAs

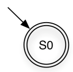
- Goal: Given regular expression e, construct NFA: <e> = $(\Sigma, Q, q_0, F, \delta)$
  - Remember r.e. defined recursively from primitive r.e. languages
  - Invariant: $|F| = 1$ in our NFAs
    - Recall F = set of final states
- Base case: a



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

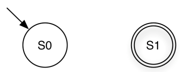
Reduction (cont'd)

- Base case: ε



$\langle\varepsilon\rangle = (\varepsilon, \{S0\}, S0, \{S0\}, \emptyset)$

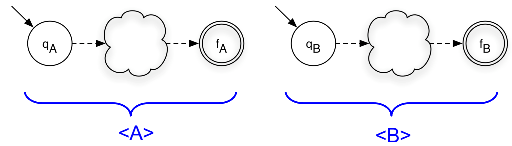
- Base case: ∅



$\langle\emptyset\rangle = (\emptyset, \{S0, S1\}, S0, \{S1\}, \emptyset)$

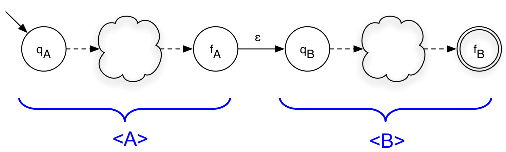
Reduction (cont'd)

- Induction: AB



Reduction (cont'd)

- Induction: AB



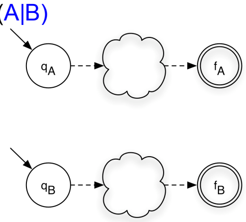
- $\langle A\rangle = (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- $\langle B\rangle = (\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B)$
- $\langle AB\rangle = (\Sigma_A \cup \Sigma_B, Q_A \cup Q_B, q_A, \{f_B\}, \delta_A \cup \delta_B \cup \{(f_A, \varepsilon, q_B)\}\ )$

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

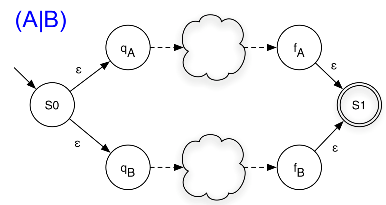
Reduction (cont'd)

- Induction: (A|B)



Reduction (cont'd)

- Induction: (A|B)



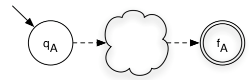
- $\langle A\rangle = (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- $\langle B\rangle = (\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B)$
- $\langle (A|B)\rangle = (\Sigma_A \cup \Sigma_B, Q_A \cup Q_B \cup \{S0,S1\}, S0, \{S1\},$
  $\delta_A \cup \delta_B \cup \{(S0,\varepsilon,q_A), (S0,\varepsilon,q_B), (f_A,\varepsilon,S1),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

Practice

- Draw the NFA for these regular expressions using exactly the reduction method:
  - ab | bc
  - hello | hi
- Write the formal NFA for the same regular expressions

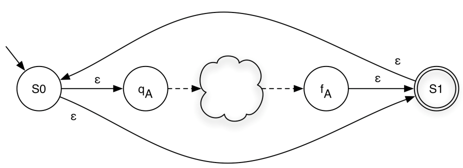
Reduction (cont'd)

- Induction: A*



Reduction (cont'd)

- Induction: A*



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

Practice

- Draw the NFA for these regular expressions using exactly the reduction method:
  - (ab | bc*)*
  - hello | (hi)*
- Write the formal NFA for the same regular expressions

Reduction Complexity

- Given a regular expression A of size n...
  Size = # of symbols + # of operations
- How many states does <A> have?
  - 2 added for each |, 2 added for each *
  - O(n)
  - That's pretty good!

Practice

Draw NFAs for the following regular expressions and languages:

- (0|1)*110*
- 101*|111
- all binary strings ending in 1 (odd numbers)
- all alphabetic strings which come after "hello" in alphabetic order
- (ab*c|d*a|ab)d

Handling ε-transitions

What if we want to remove all those unneeded ε-transitions?

First, some definitions:

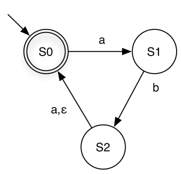
- We say: $p \xrightarrow{\epsilon} q$
  - if it is possible to transition from state p to state q taking only ε-transitions
  - if $\exists$ p, p_1, p_2, … p_n, q $\in$ Q (p $\neq$ q) such that $\{p, \epsilon, p_1\} \in \delta, \{p_1, \epsilon, p_2\} \in \delta, \ldots, \{p_n, \epsilon, q\} \in \delta$

ε-closure

- For any state p, the <u>ε-closure</u> of p is defined as the set of states q such that $p \xrightarrow{\epsilon} q$
- {q | $p \xrightarrow{\epsilon} q$ }

Example

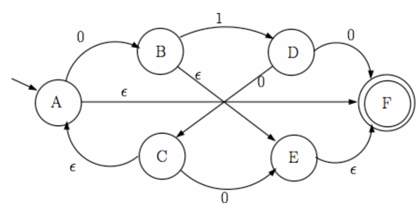
- What's the ε-closure of S2 in this NFA?



- {S2, S0}

Example

- Find the ε-closure for each of the states in this NFA:



Example

- Make the NFA for the regular expression
  - (0|1*)111(0*|1)
- Find the epsilon closure for each of the states of your NFA