CSE420 Compiler Design

Lecture: 3
Lexical Analysis (Part 2)

Finite State Automata (FSAs)

- **AKA “Finite State Machines”, “Finite Automata”, “FA”**
- A *recognizer* for a language is a program that takes as input a string x and answers “yes” if x is a sentence of the language and “no” otherwise.
  - The regular expression is compiled into a recognizer by constructing a generalized transition diagram called a finite automaton.
- One start state
- Many final states
- Each state is labeled with a state name
- Directed edges, labeled with symbols
- Two types
  - Deterministic (DFA)
  - Non-deterministic (NFA)

Nondeterministic Finite Automata

A nondeterministic finite automaton (NFA) is a mathematical model that consists of

1. **A set of states S**
2. **A set of input symbols Σ**
3. **A transition function that maps state/symbol pairs to a set of states**
4. **A special state s_0 called the start state**
5. **A set of states F (subset of S) of final states**

INPUT: string

OUTPUT: yes or no

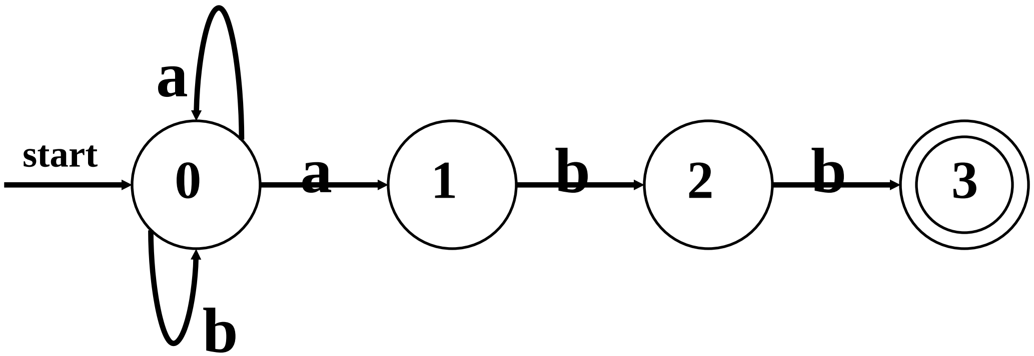
Example – NFA : (a|b)*abb

$S = \{0, 1, 2, 3\}$

$s_0 = 0$

$F = \{3\}$

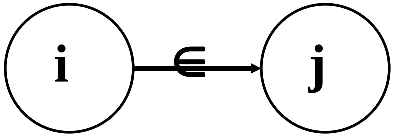
$\Sigma = \{a, b\}$



state	input a	input b
0	{ 0, 1 }	{ 0 }
1	--	{ 2 }
2	--	{ 3 }

Transition Table

∈ (null) moves possible



Switch state but do not use any input symbol

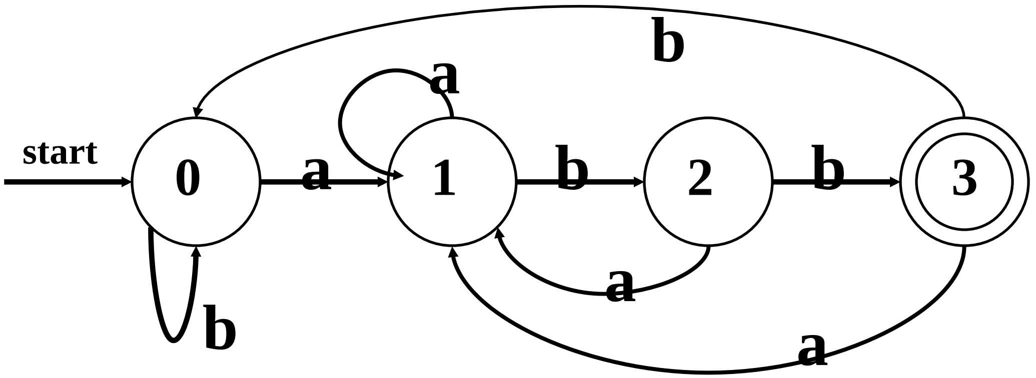
Deterministic Finite Automata

A DFA is an NFA with the following restrictions:

- **$\in$ moves are <u>not</u> allowed**
- **For every state s $\in$S, there is one and only one path from s for every input symbol a $\in$ Σ.**

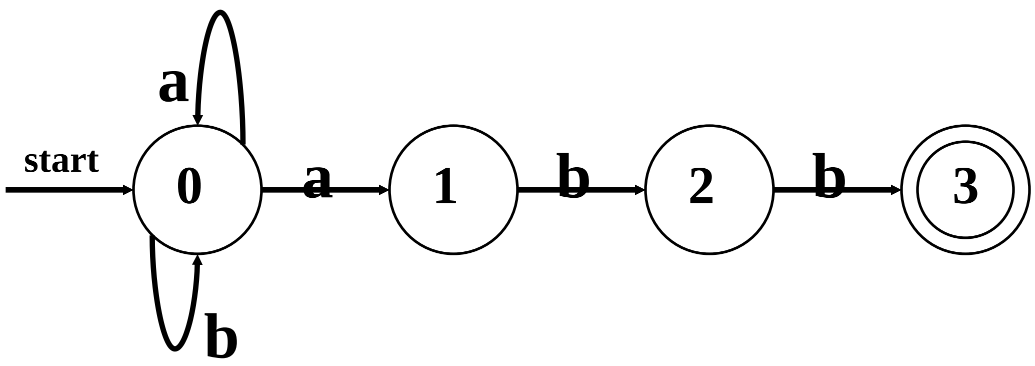
Since transition tables don't have any alternative options, DFAs are easily simulated via an algorithm.

Example – DFA : (a|b)*abb



What Language is Accepted?

Recall the original NFA:



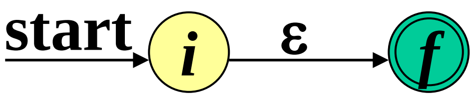
DFA vs NFA

- Both DFA and NFA are the recognizers of regular sets.
- But – time-space trade space exists
- DFAs are faster recognizers
  - Can be much bigger too..

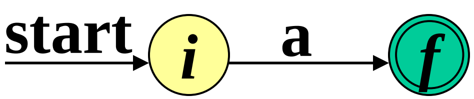
Converting Regular Expressions to NFAs

Thompson's Construction

- Empty string ε is a regular expression denoting { ε }



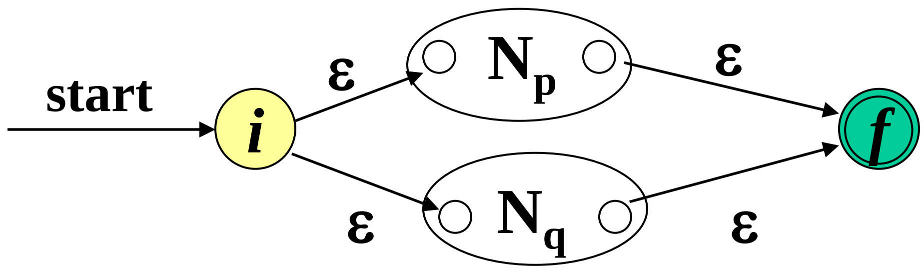
- *a* is a regular expression denoting {*a*} for any *a* in Σ



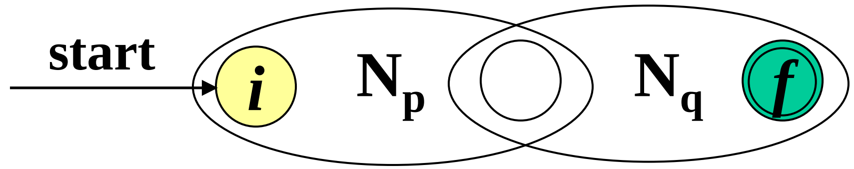
Converting Regular Expressions to NFAs

If P and Q are regular expressions with NFAs N_p, N_q:

P | Q (union)



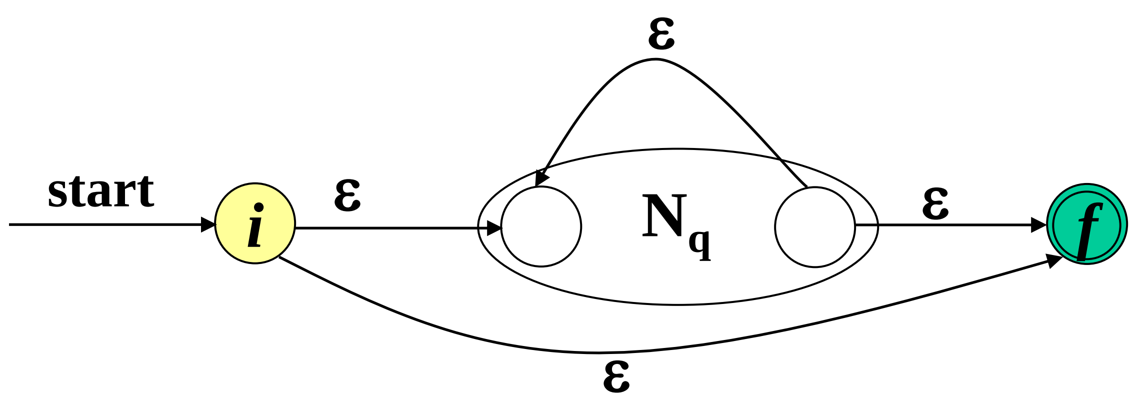
PQ (concatenation)



Converting Regular Expressions to NFAs

If Q is a regular expression with NFA N_q:

Q* (closure)



NFA Construction

RE: (a|b)*abb

NFA Construction

RE: (a|b)*abb

H.W: Construct NFA for the following RE

(ab*c) | (a(b|c*))

NFA Construction

id $\rightarrow$ **letter (letter | digit)***

End of slide