UCF | College of Engineering and Computer Science

UNIVERSITY OF CENTRAL FLORIDA

Regular Expressions to Finite Automata

COP-3402 Systems Software

Paul Gazzillo

UCF

Automate Lexing by Generating Automata

- We can automatically generate lexers
    - Specify tokens via regular expressions
    - Algorithm to convert regex to automata
- Two methods
    - Simulate an NFA
    - Convert an NFA to a DFA: subset construction

Convert Via the Subset Construction

- NFA can be in multiple states at once
- Finite automaton means finite number of states
- Therefore, finite number of combinations of states
  - How many subsets of states are there? (Hint: powerset)

Translate Each Regex in Order of Operations

- Convert each subexpression to an NFA
- Combine the NFAs for each subexpression
- Each regex operation corresponds to an NFA template
  - Concatenation
  - Alternation
  - Kleene closure

Demo: Regex Operations as NFAs

Concatenation

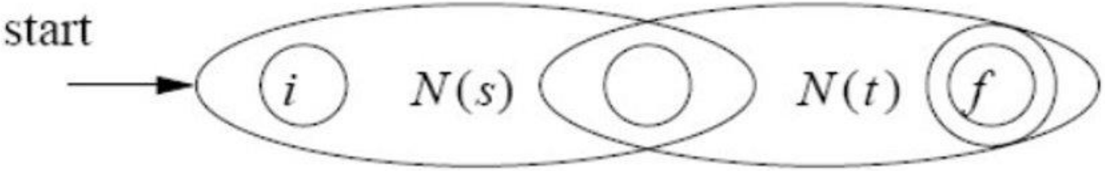

Figure 3.41: NFA for the concatenation of two regular expressions

Alternation

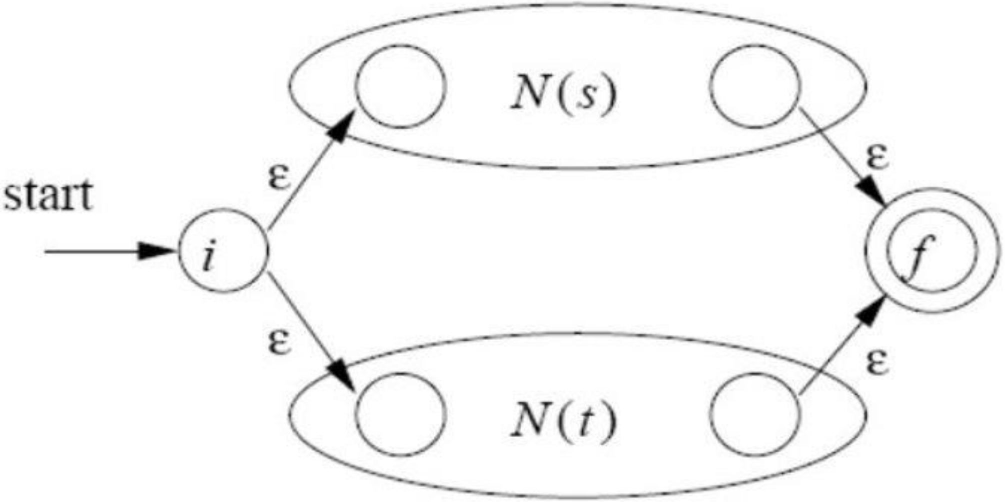

Figure 3.40: NFA for the union of two regular expressions

Kleene Closure

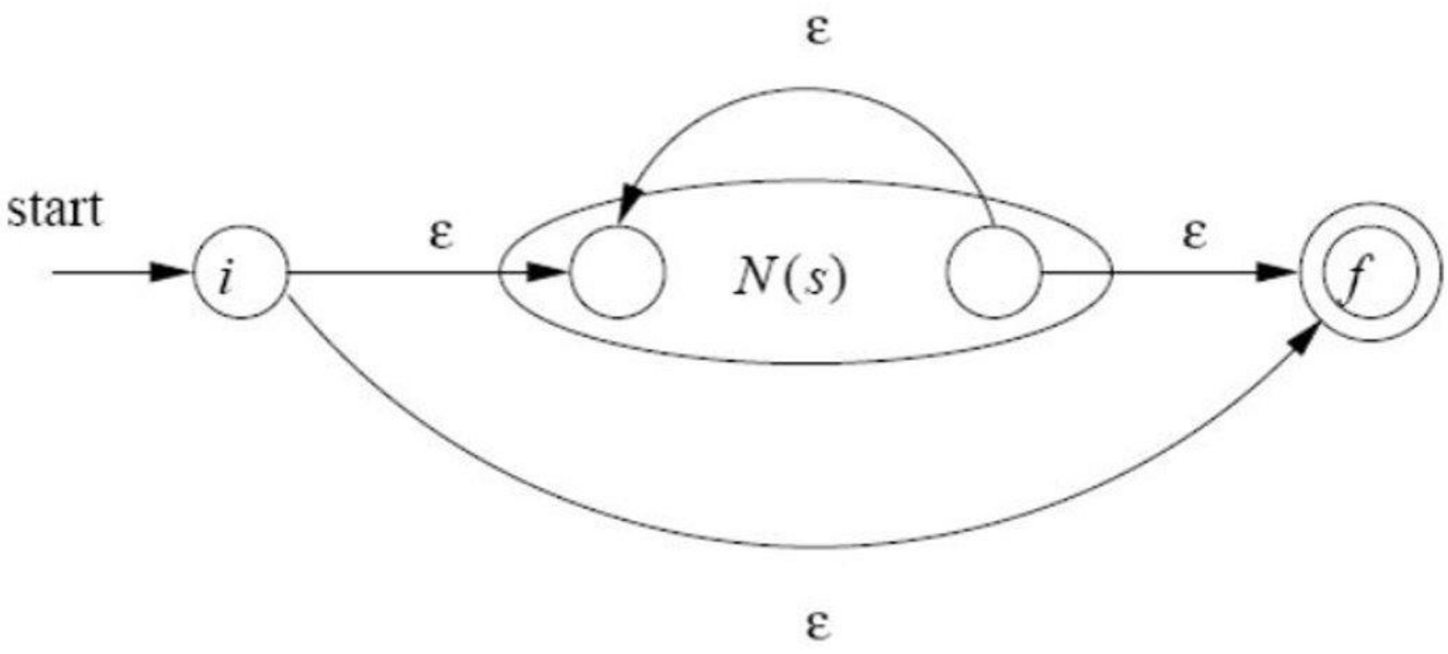

Figure 3.42: NFA for the closure of a regular expression

The NFA for (a|b)*abb

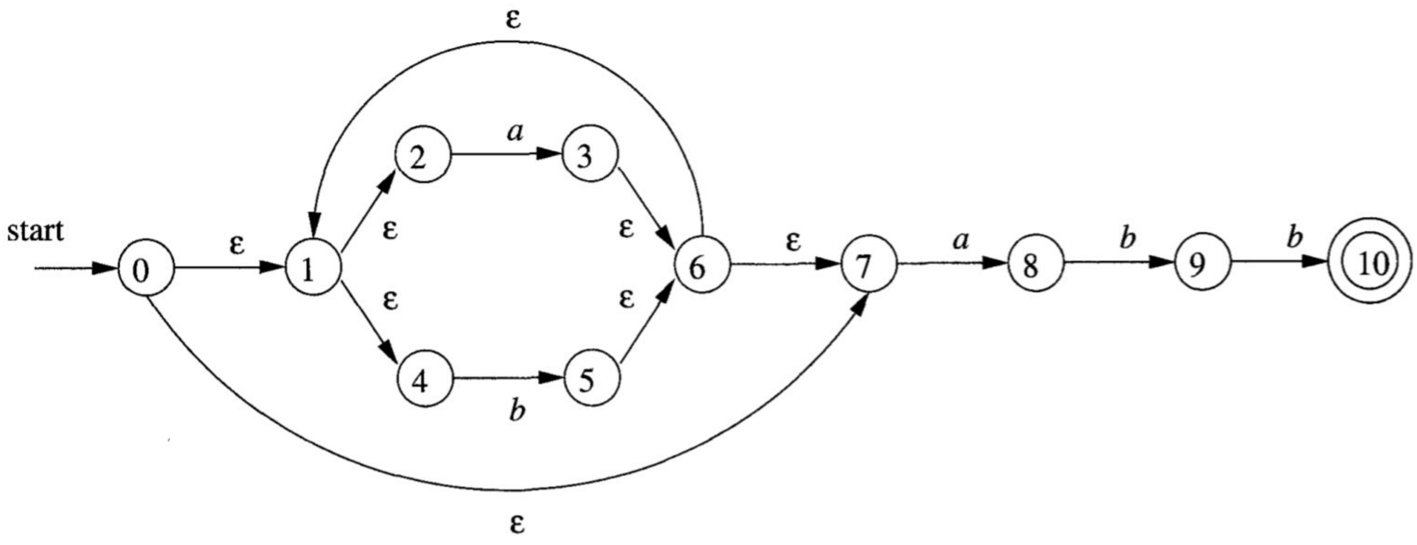

Figure 3.34: NFA N for $(\mathbf{a}|\mathbf{b})^*\mathbf{abb}$

Demo: Converting Regex to NFA

(a|b)*abb

aa*|bb*

((a|bc)b)*

Construct a DFA from an NFA Systematically

- Each DFA state created from subset of NFA states
    - Remember: can be in multiple states
- "Simulate" being in multiple states using a single state
    - Dragon book 3.7
- The multiple states are a *subset* of the NFA states
- Create the DFA by calling each subset a single DFA state

Sketch of the Subset Construction Algorithm

- Start at the starting state of the NFA
- Group all states reachable by ε (epsilon)
    - This is the *ε-closure*
    - Call this group of states the initial state for the DFA
- For each symbol **s** in the alphabet (remember its finite)
    - Get all that states that **s** transitions to
    - Find the ε-closure of those states
    - Call this group of states a single state of the DFA
- Repeat for all combinations of NFA states and symbols
    - Stop when we have covered them all

Demo: Converting NFA to DFA

(a|b)*abb

aa*|bb*

((a|bc)b)*

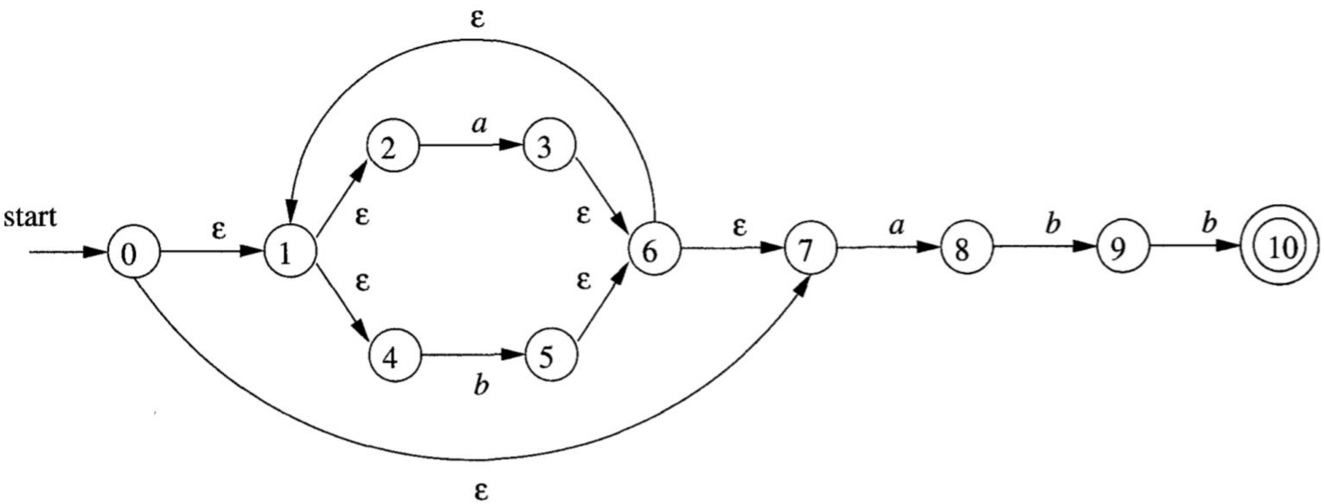

Figure 3.34: NFA N for $(\mathbf{a}|\mathbf{b})^*\mathbf{abb}$

NFA State	DFA State	a	b
$\{0,1,2,4,7\}$	A	B	C
$\{1,2,3,4,6,7,8\}$	B	B	D
$\{1,2,4,5,6,7\}$	C	B	C
$\{1,2,4,5,6,7,9\}$	D	B	E
$\{1,2,3,5,6,7,10\}$	E	B	C

Figure 3.35: Transition table *Dtran* for DFA D

Conclusion

- We can automatically generate lexers
- Regular expressions correspond to automata
  - Automata implemented with transition tables or if statements and while loops
- Simpler to generate NFAs from regular expressions
- Subset construction to convert NFA to DFA
  - Algorithm in Dragon Book 3.7.1
  - Alternative: simulate an NFA (Dragon Book 3.7.2)