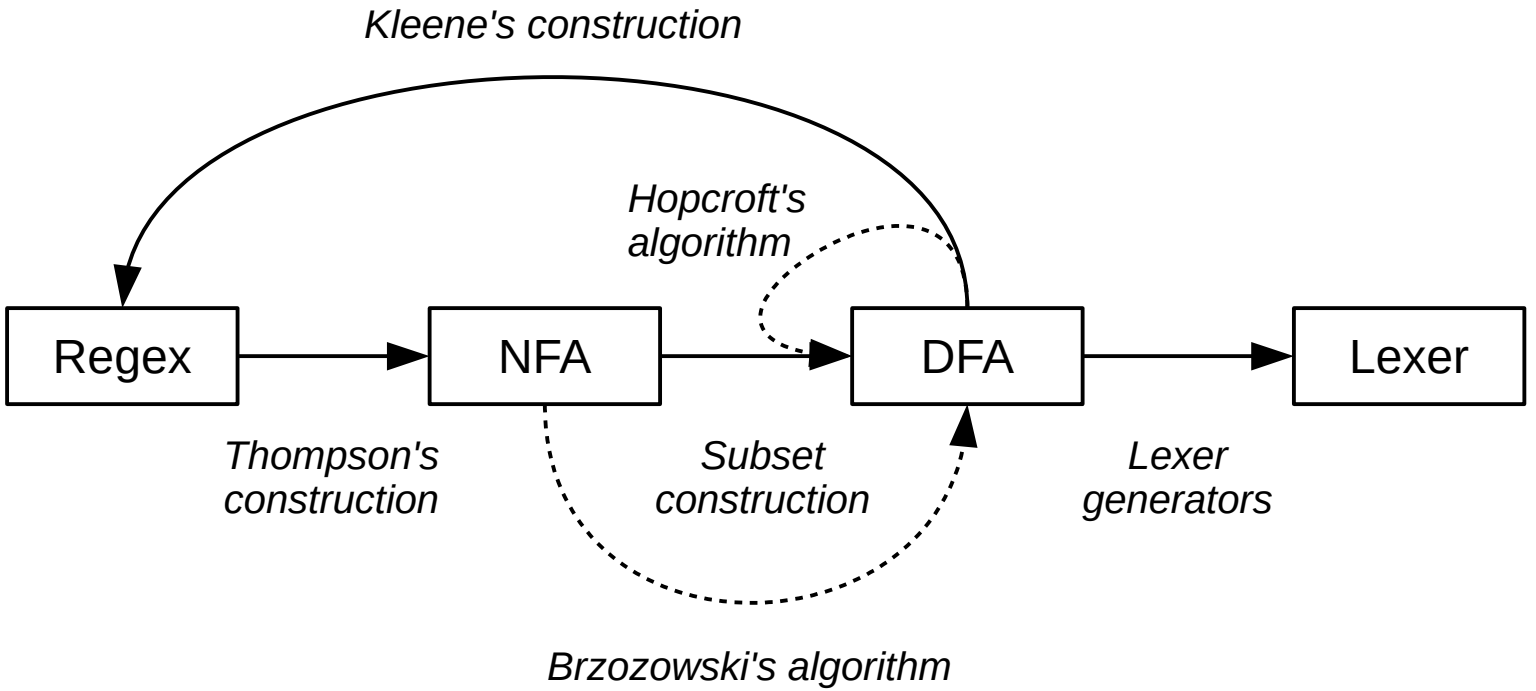
CS 480
Fall 2015

Mike Lam, Professor

Finite Automata Conversions and Lexing

Finite Automata

- Finite automata transitions:

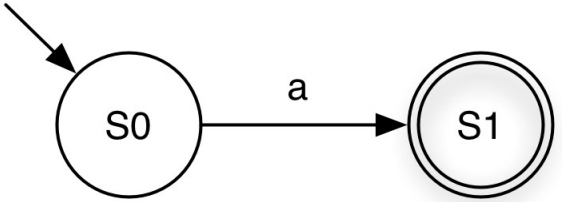


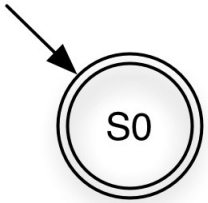
(dashed lines indicate transitions to a minimized DFA)

Finite Automata

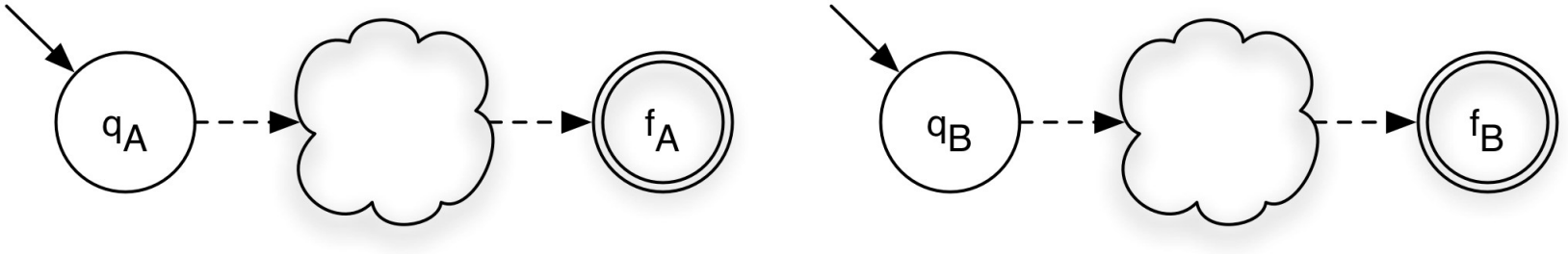
- RE to NFA: Thompson's construction
  - Core insight: build up NFA using "**templates**"
- NFA to DFA: Subset construction
  - Core insight: DFA node = **subset** of NFA nodes
  - Core concept: use **null closure** to calculate subsets
- DFA minimization
  - Core insight: create **partitions**, then keep splitting

Thompson's: Base cases

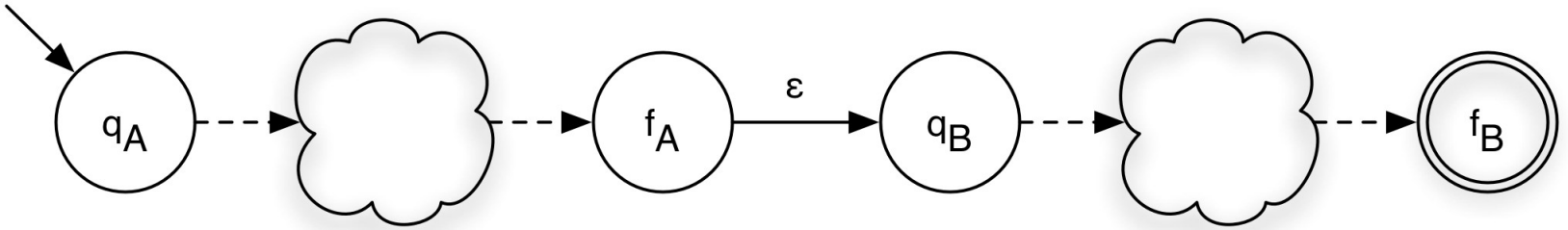




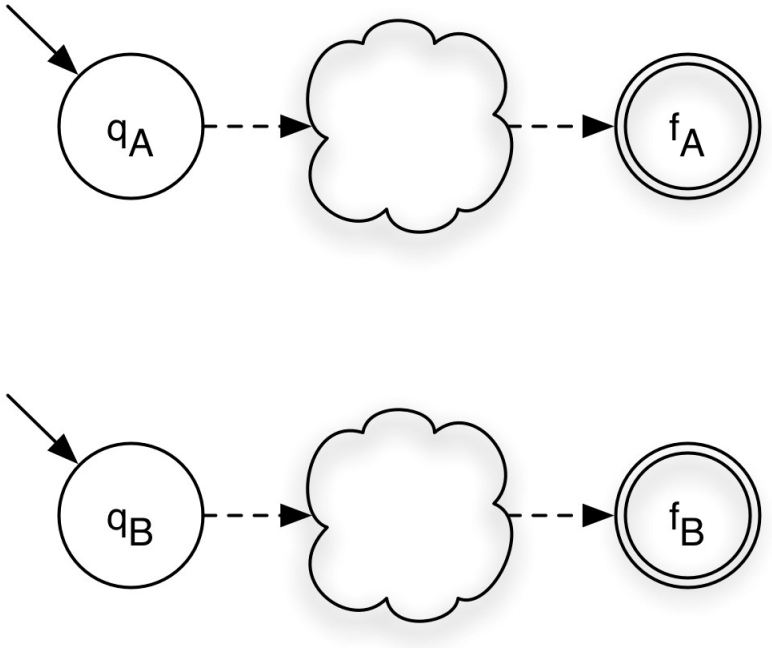
Thompson's: Concatenation



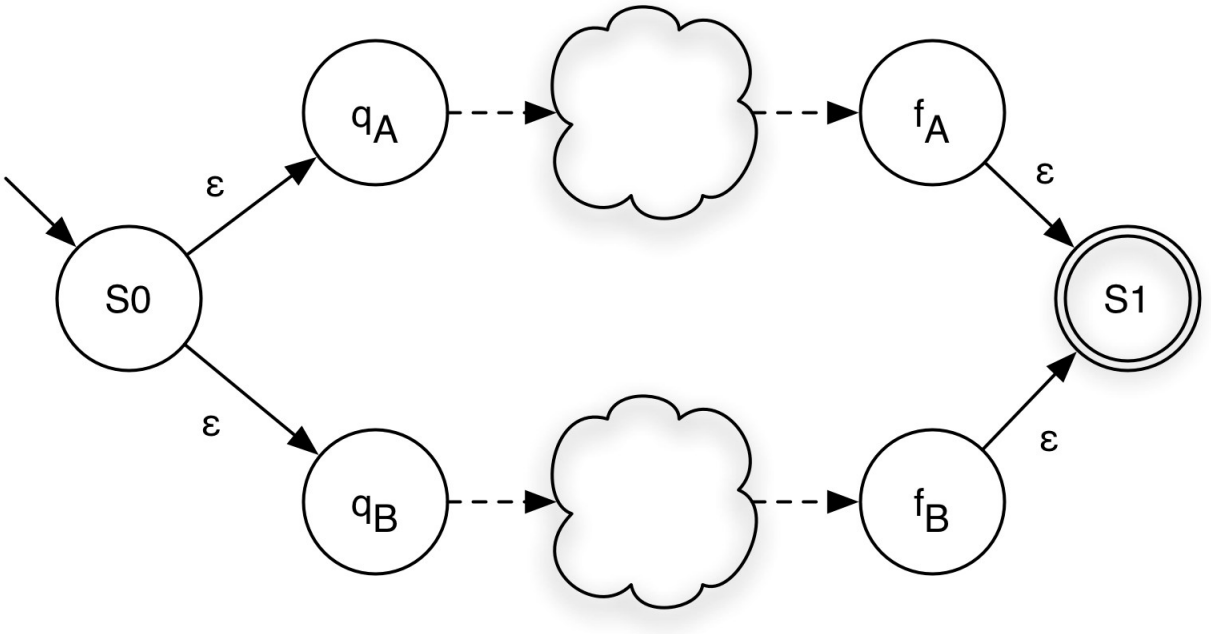
Thompson's: Concatenation



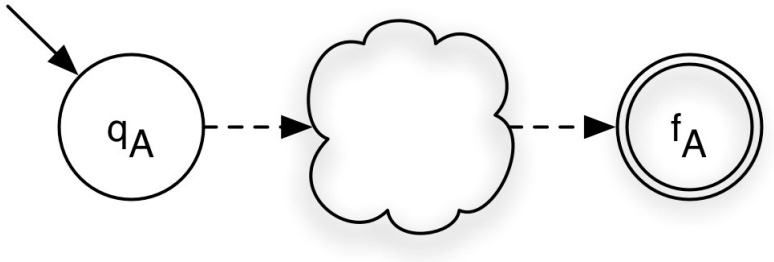
Thompson's: Union



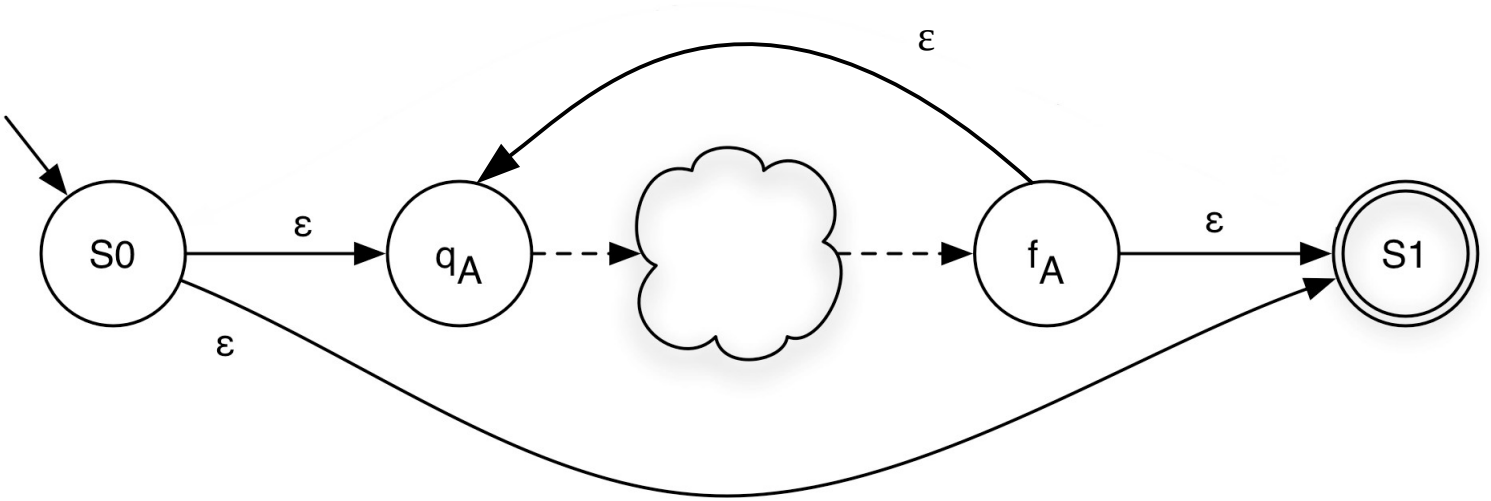
Thompson's: Union



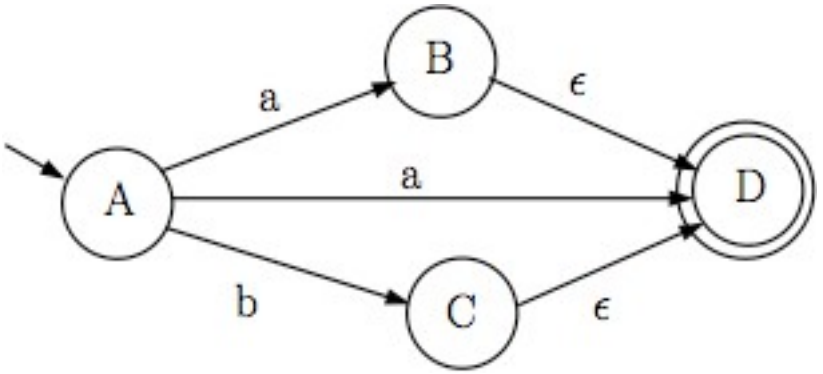
Thompson's: Closure



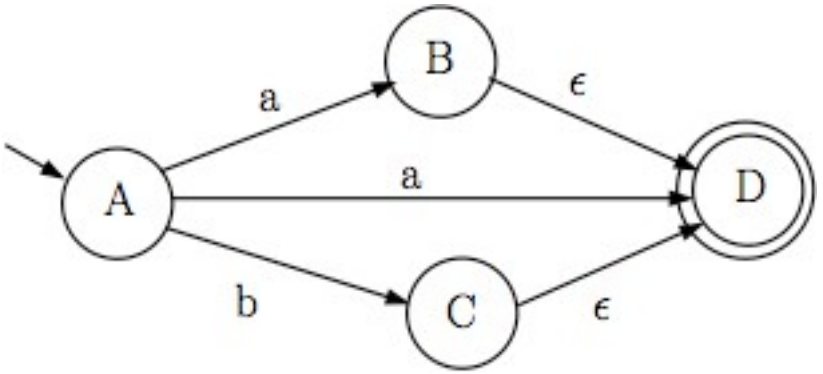
Thompson's: Closure

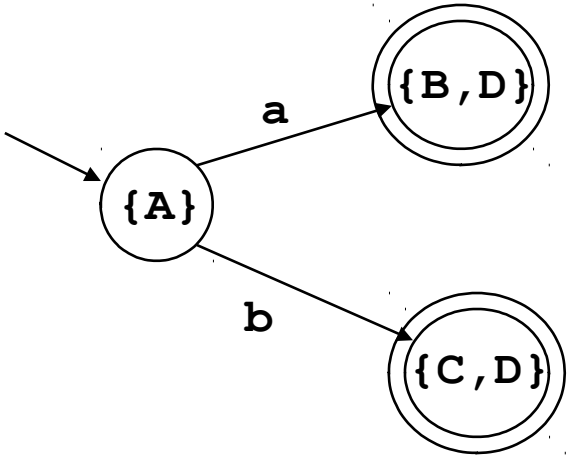


Subset Example

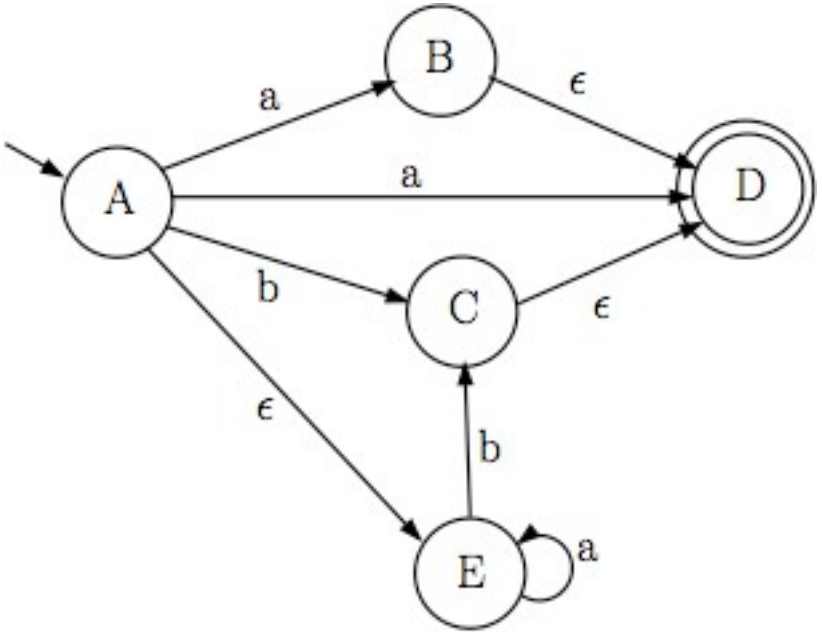


Subset Example

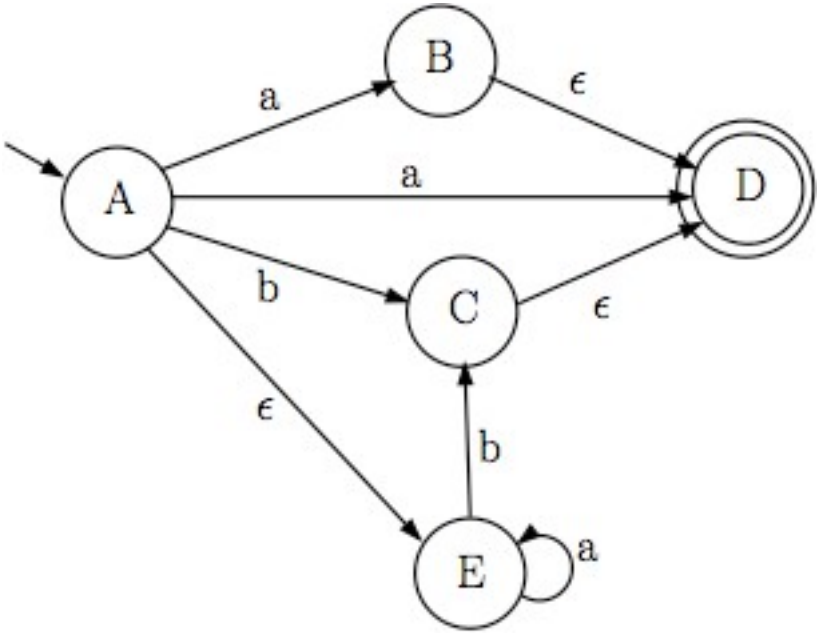


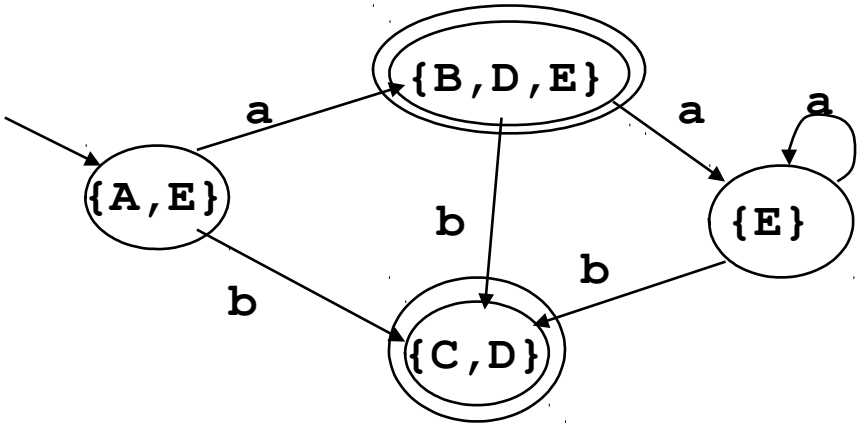


Subset Example



Subset Example





Discussion Questions

- How long does it take to...
  - Build an NFA?
  - Run an NFA?
  - Build a DFA?
  - Run a DFA?

Efficiency Concerns

- Thompson's construction
  - Runs in linear time to # of regex characters
  - Results in linear space increase
- NFA execution
  - Proportional to both NFA size and input string size
- Subset construction
  - Potential exponential state space explosion
  - A n-state NFA could require up to 2^n DFA states
  - However, this rarely happens in practice
- DFAs execution
  - Proportional to input string size only

NFA/DFA complexity

- NFAs build quicker (linear) but run slower
  - Better if you will only run the FA a few times
- DFAs build slower (worst case exponential) but run faster
  - Better if you will run the FA many times

	NFA	**DFA**
Build time	$O(m)$	$O(2^m)$
Run time	$O(m \times n)$	$O(n)$

m = length of regular expression
n = length of input string

Lexers

- Auto-generated
  - Table-driven: generic scanner, auto-generated tables
  - Direct-coded: hard-code the tables into the scanner
  - Common tools: lex/flex and similar
- Hand-coded
  - Better I/O handling
  - Easier interfacing w/ other phases

Handling Keywords

- Embed into NFA/DFA
  - Easier/faster for generated scanners
- Use lookup table
  - Easier for hand-coded scanners