

# Lexical Analysis

CMPT 379: Compilers

Instructor: Anoop Sarkar

[anoopsarkar.github.io/compilers-class](https://anoopsarkar.github.io/compilers-class)

# Building a Lexical Analyzer

- Token  $\Rightarrow$  Pattern
- Pattern  $\Rightarrow$  Regular Expression
- Regular Expression  $\Rightarrow$  NFA
- NFA  $\Rightarrow$  DFA
- DFA  $\Rightarrow$  Table-driven implementation of DFA

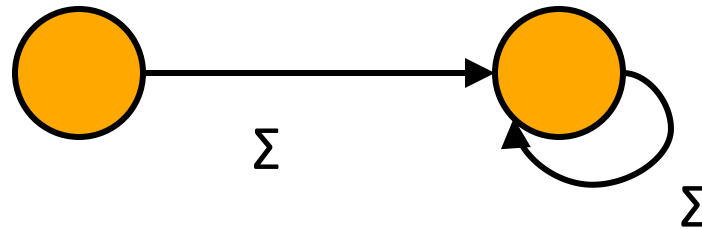
# Thompson's construction

- Converts regexps to equivalent NFA
- Six simple rules
  - Empty language
  - Symbols ( $\Sigma$ )
  - Empty String ( $\epsilon$ )
  - Alternation ( $r_1$  or  $r_2$ )
  - Concatenation ( $r_1$  followed by  $r_2$ )
  - Repetition ( $r_1^*$ )

Used by Ken Thompson for pattern-based search in text editor QED (1968)

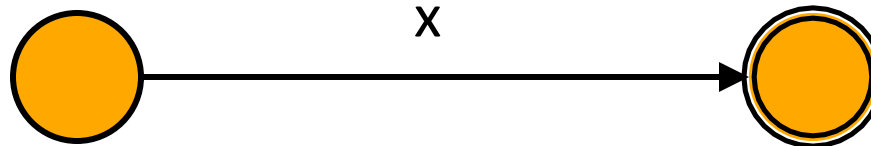
# Thompson Rule 0

- For the empty language  $\phi$  (optionally include a *sinkhole* state)



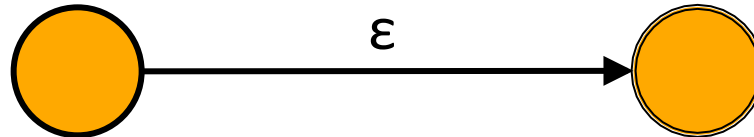
# Thompson Rule 1

- For each symbol  $x$  of the alphabet, there is a NFA that accepts it



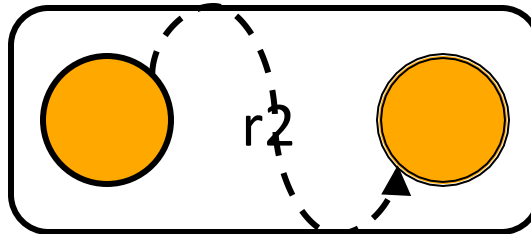
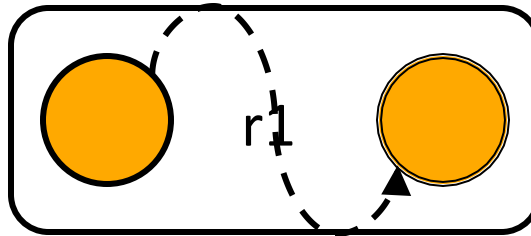
# Thompson Rule 2

- There is an NFA that accepts only  $\epsilon$



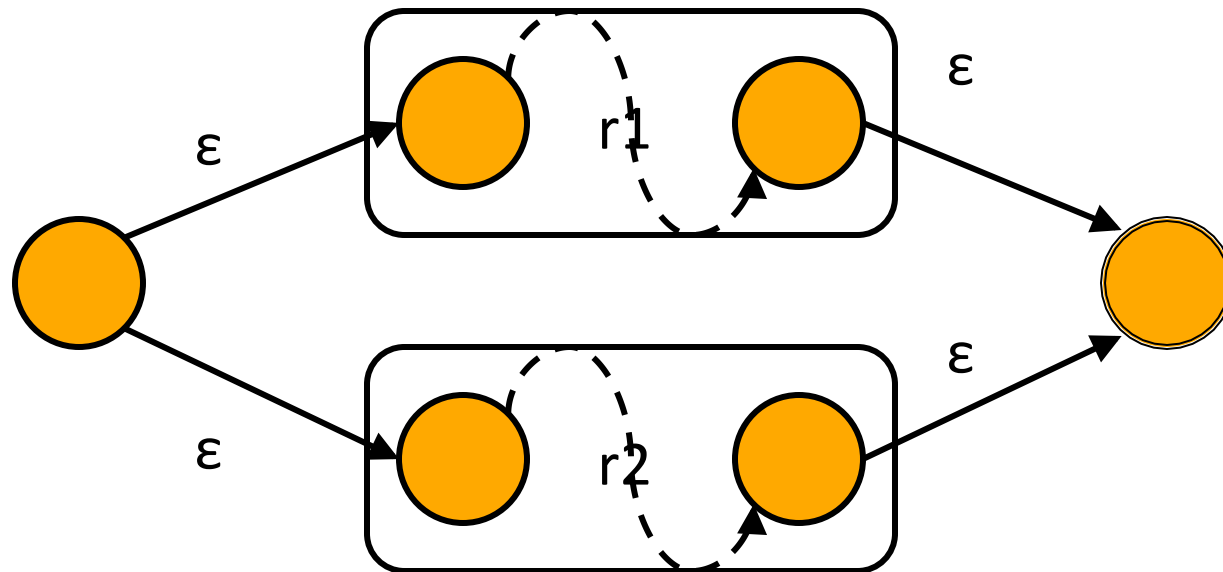
# Thompson Rule 3

- Given two NFAs for  $r_1$ ,  $r_2$ , there is a NFA that accepts  $r_1 | r_2$



# Thompson Rule 3

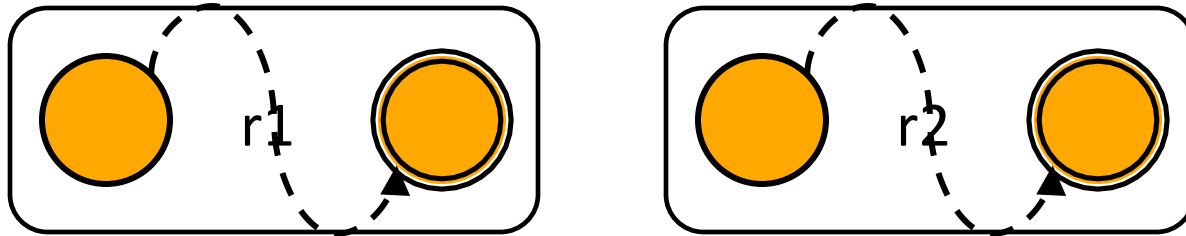
- Given two NFAs for  $r_1$ ,  $r_2$ , there is a NFA that accepts  $r_1 | r_2$





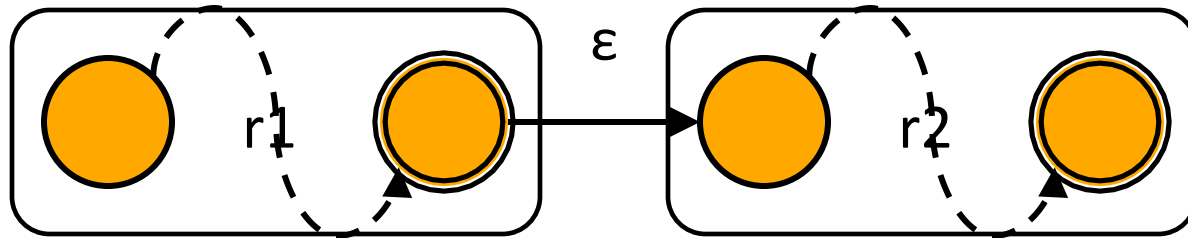
# Thompson Rule 4

- Given two NFAs for  $r_1$ ,  $r_2$ , there is a NFA that accepts  $r_1r_2$



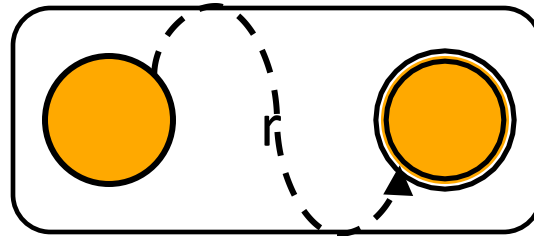
# Thompson Rule 4

- Given two NFAs for  $r_1$ ,  $r_2$ , there is a NFA that accepts  $r_1r_2$



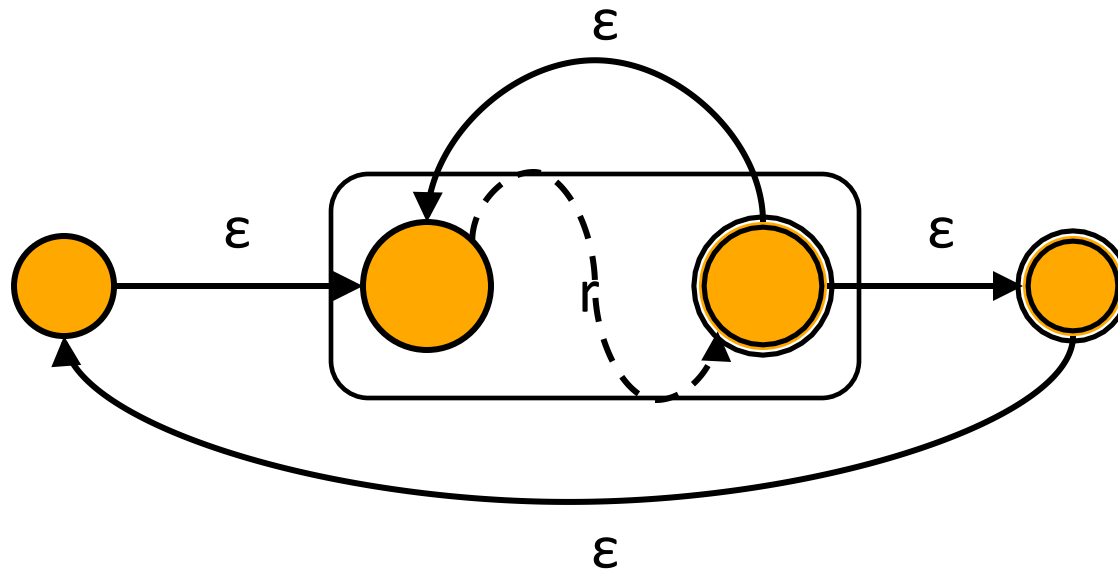
# Thompson Rule 5

- Given a NFA for  $r$ , there is an NFA that accepts  $r^*$



# Thompson Rule 5

- Given a NFA for  $r$ , there is an NFA that accepts  $r^*$



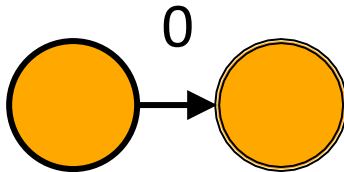
# Example

- Set of all binary strings that are divisible by four (include 0 in this set)
- Defined by the regexp:  $((0|1)^*00) | 0$
- Apply Thompson's Rules to create an NFA

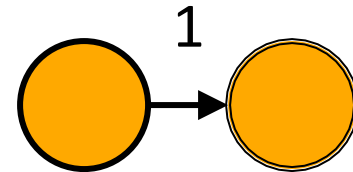
# Basic Blocks 0 and 1

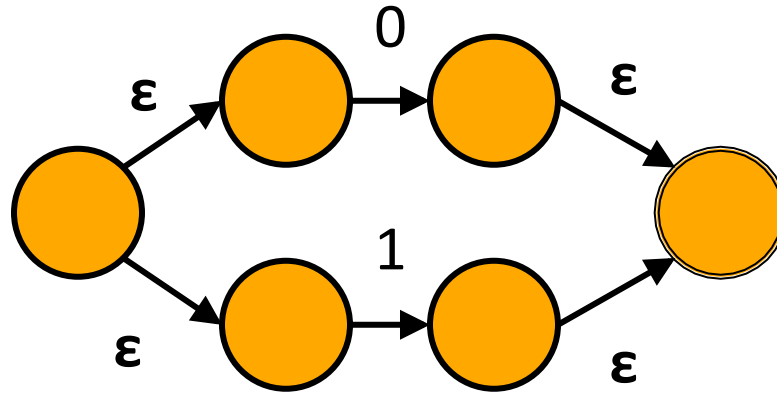
$((0|1)^*00) | 0$

NFA for 0



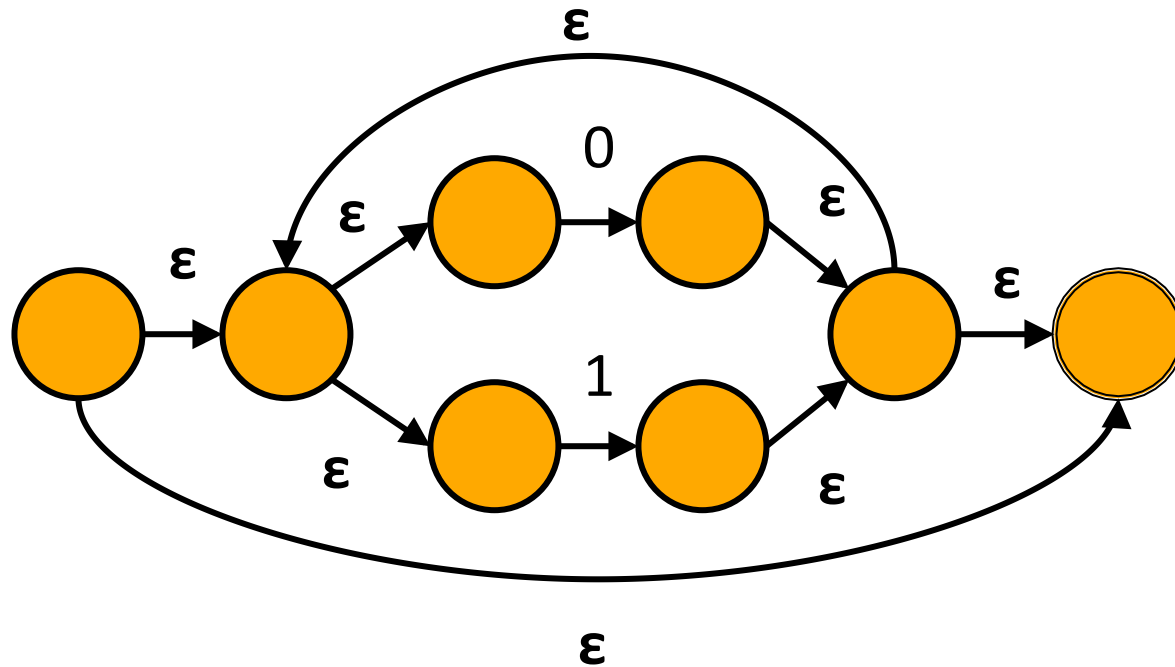
NFA for 1





$0|1$

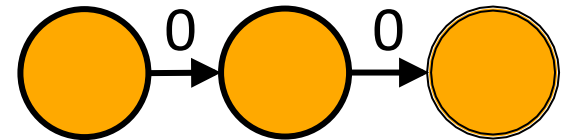
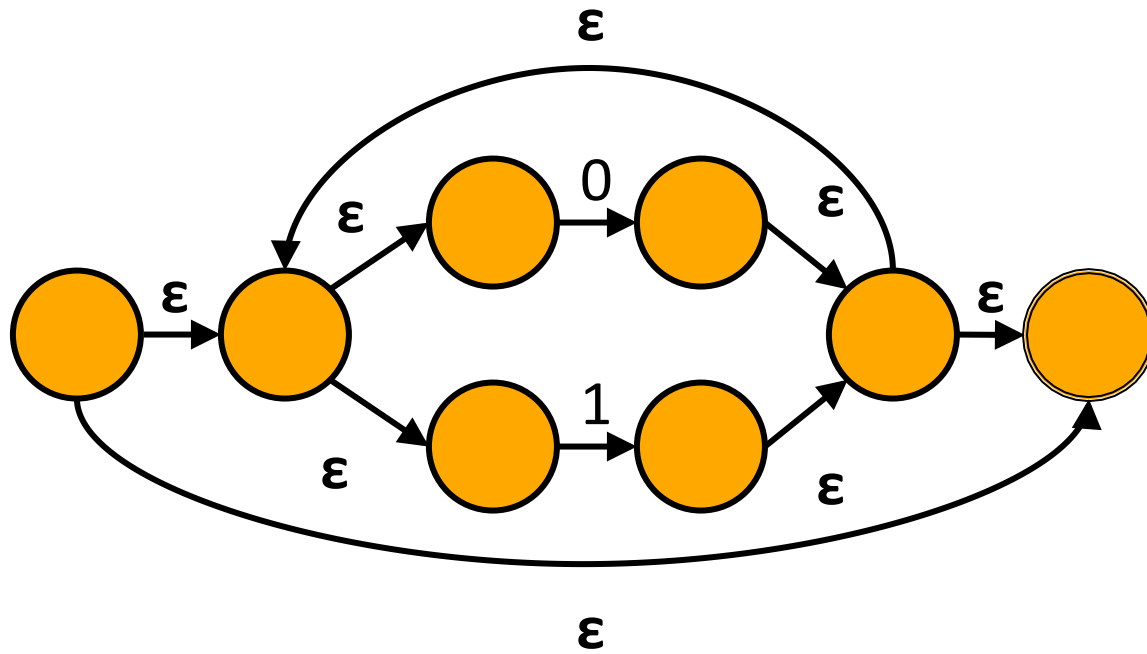
$((0|1)^*00) | 0$



$(0|1)^*$

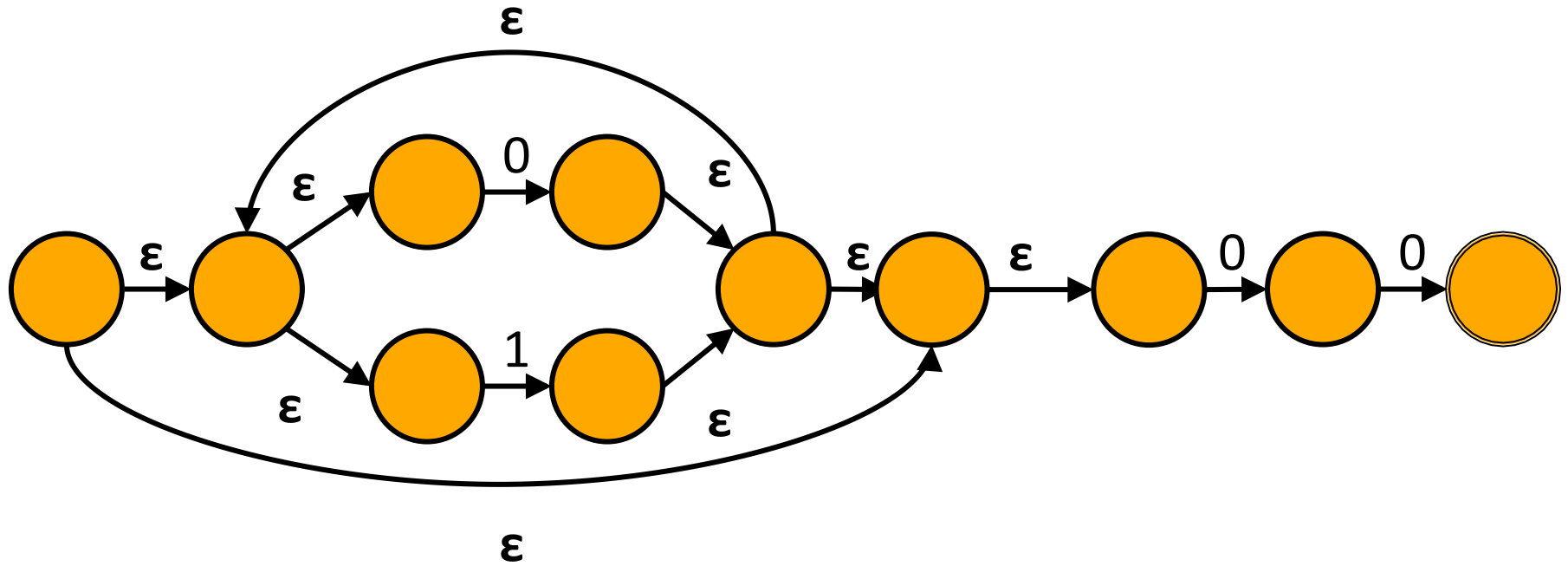
$((0|1)^*00) | 0$





$(0|1)^*00$

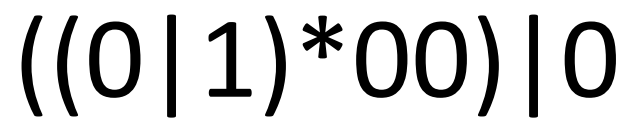
$((0|1)^*00) | 0$



$(0|1)^*00$

$((0|1)^*00) | 0$





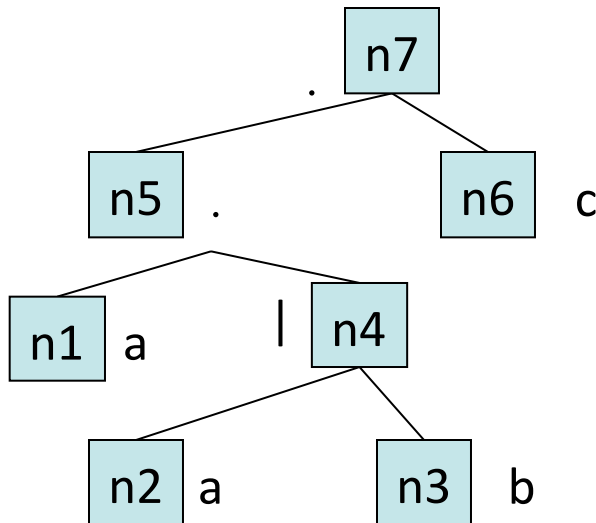
# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$   
 $aab|c.$

Post-order traversal of  
regexp tree

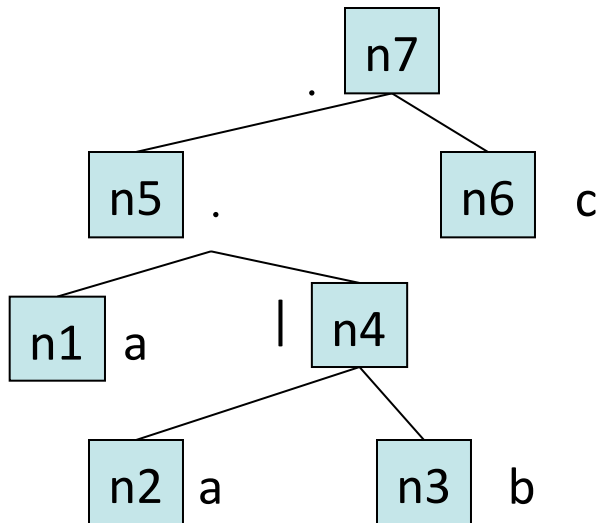


$n1 = \text{nfa}(a)$   
 $n2 = \text{nfa}(a)$   
 $n3 = \text{nfa}(b)$   
 $n4 = \text{nfa}(n2, n3, |)$   
 $n5 = \text{nfa}(n1, n4, .)$   
 $n6 = \text{nfa}(c)$   
 $n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

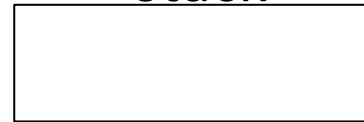
Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$

stack



Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

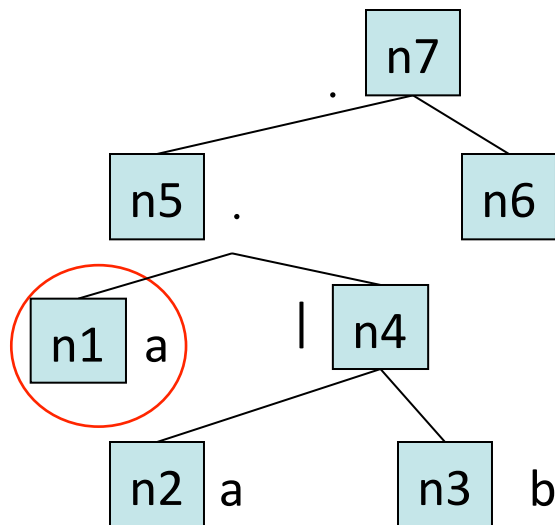
Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$

$aab|.c.$

Post-order traversal of  
regexp tree



c push n1

stack

n1

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

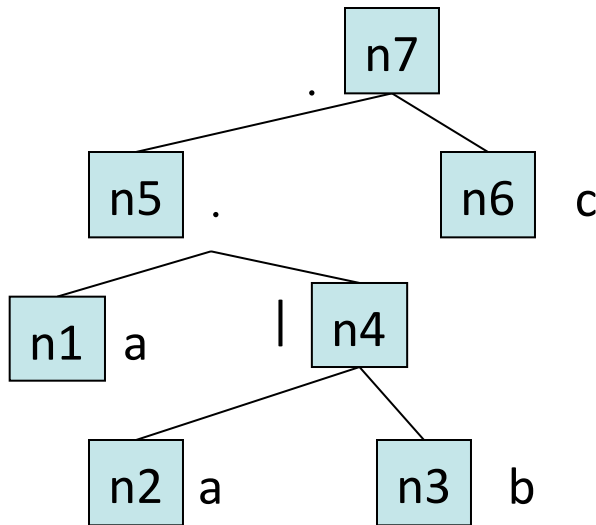
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$

stack

n1

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$



# Thompson's construction

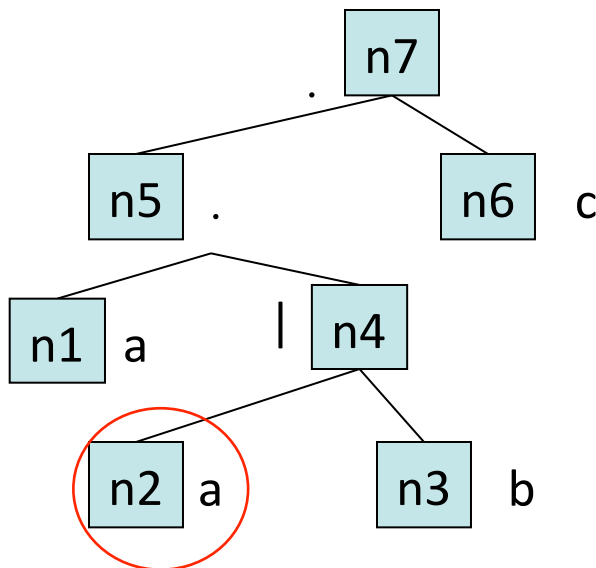
Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$

$aab|.c.$

Post-order traversal of  
regexp tree



push n2

stack

n2, n1

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

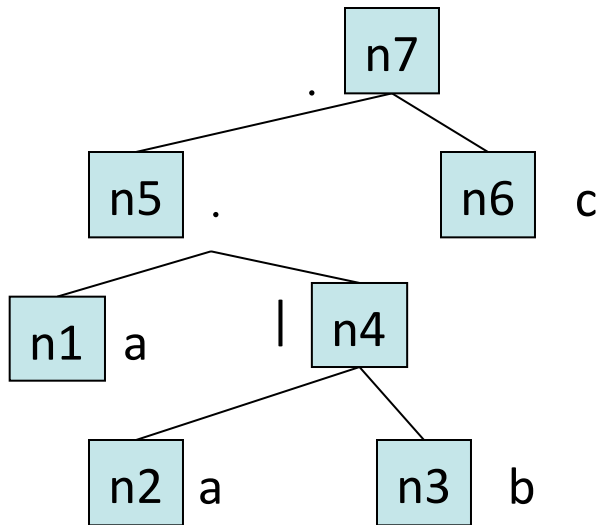
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$



stack

n2, n1

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

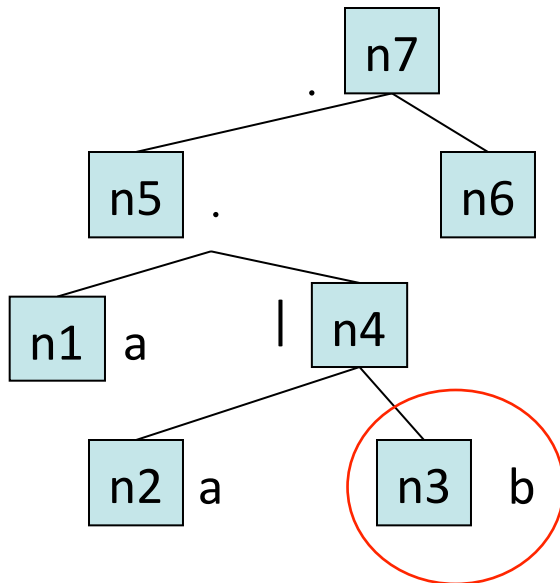
Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$

$aab|.c.$

Post-order traversal of  
regexp tree



c push n3

stack

n3, n2, n1

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

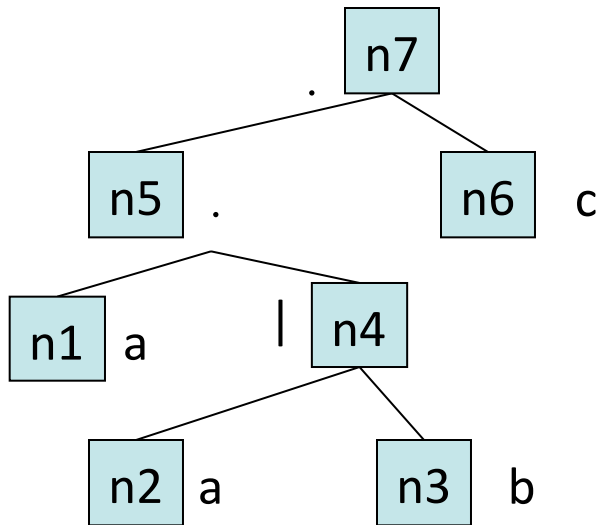
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$



stack

n3, n2, n1

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

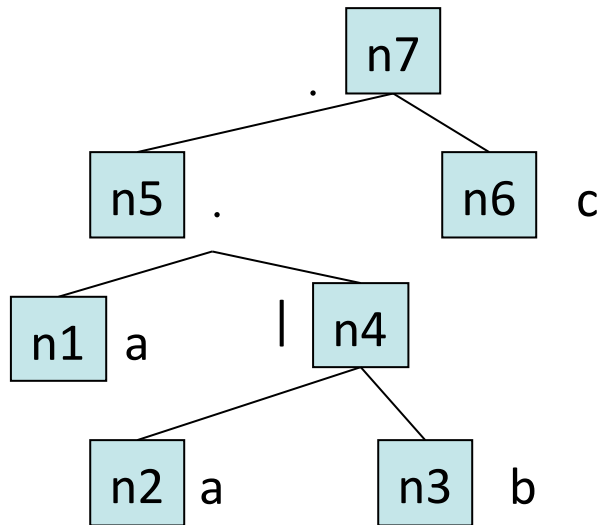
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$



stack

pop n3,n2

n1

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

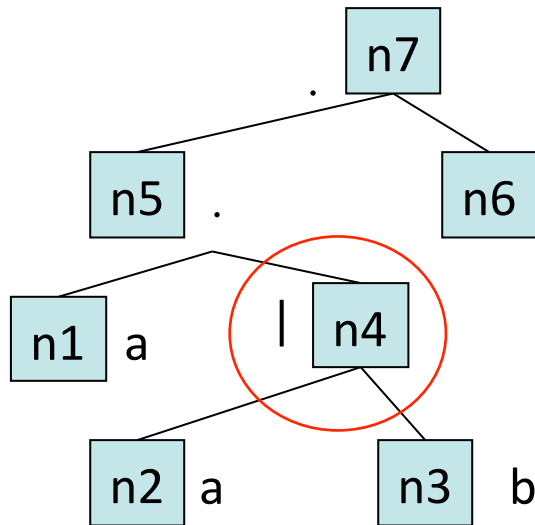
Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$

$aab|.c.$

Post-order traversal of  
regexp tree



c push n4

stack

n4, n1

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

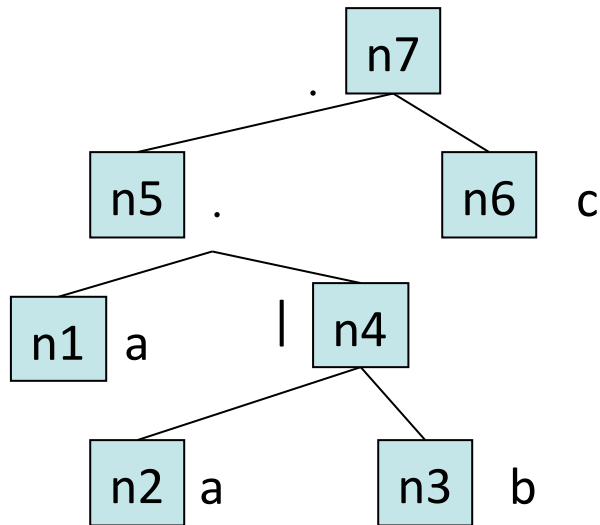
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|c.$



stack

n4, n1

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

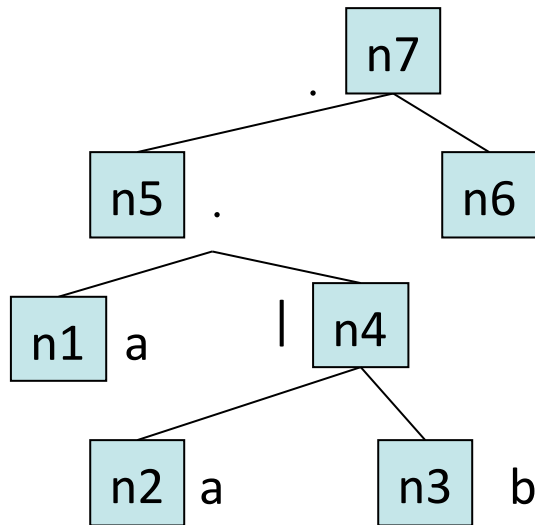
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|c$



stack

pop n4,n1

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$



# Thompson's construction

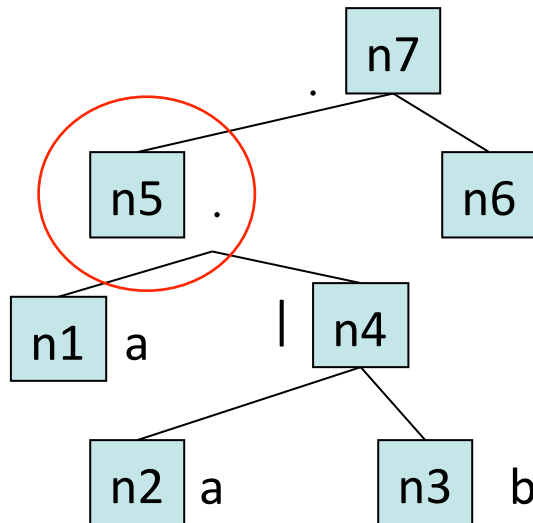
Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$

$aab|c$

Post-order traversal of  
regexp tree



c push n5

stack

n5

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

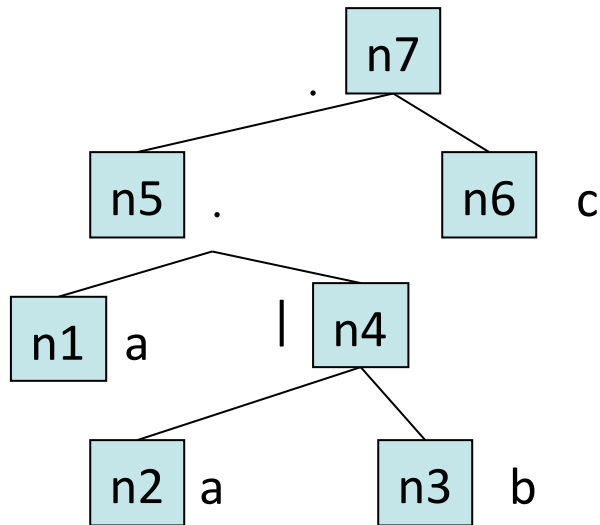
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$



stack

n5

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

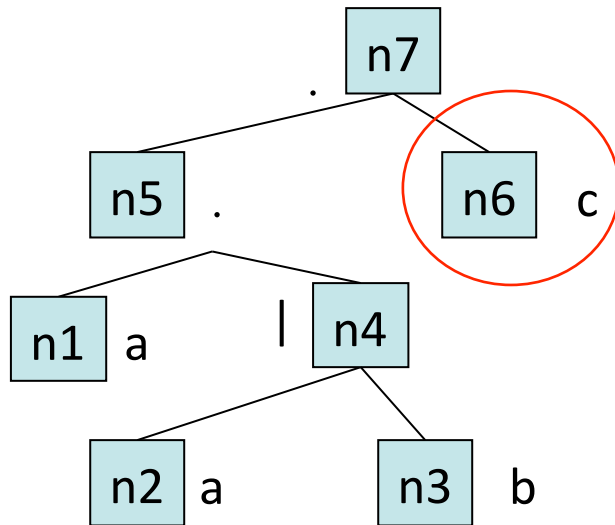
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$

push n6



stack

n6, n5

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

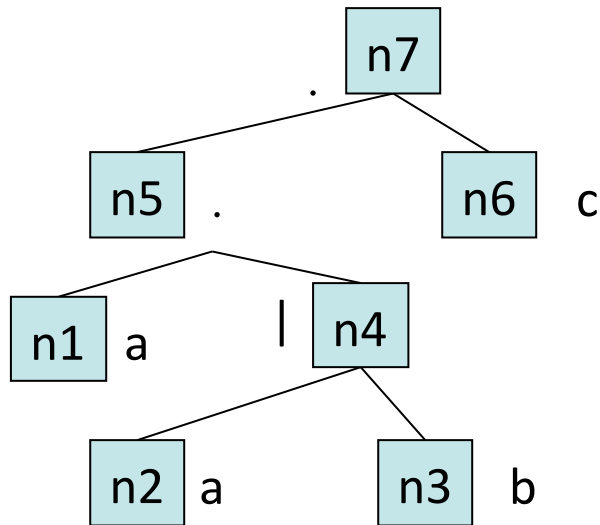
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$



stack

n6, n5

Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

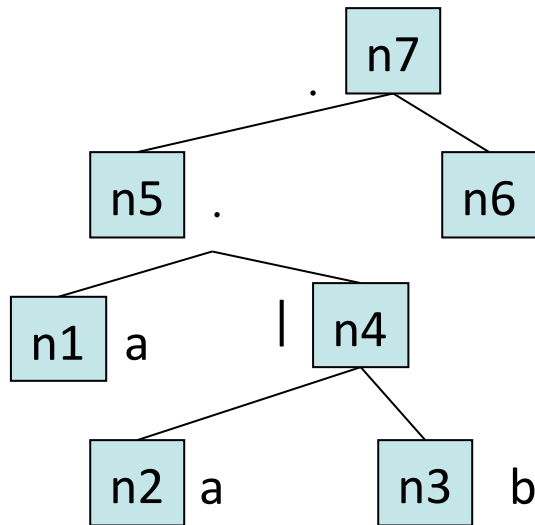
Converts regexps to NFA

Build NFA recursively  
from regexp tree

$(a(a|b))c$

$aab|.c.$

Post-order traversal of  
regexp tree



c pop n6, n5



stack



$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

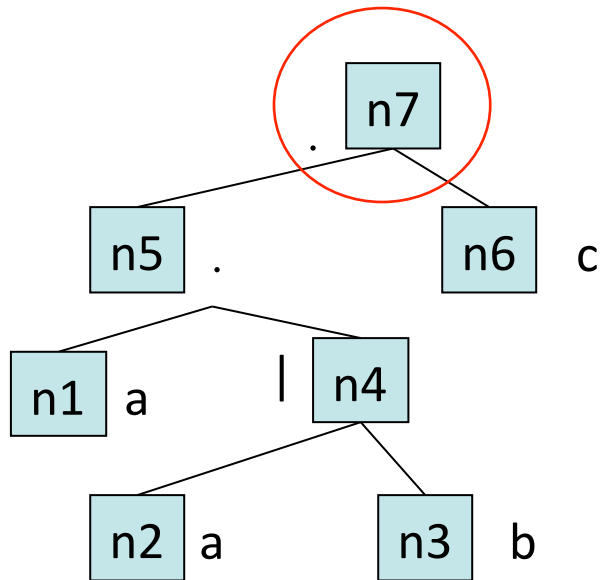
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

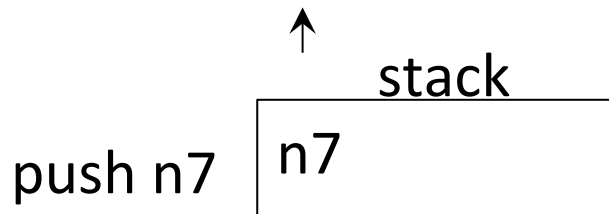
Converts regexps to NFA

Build NFA recursively  
from regexp tree



$(a(a|b))c$

$aab|.c.$



Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

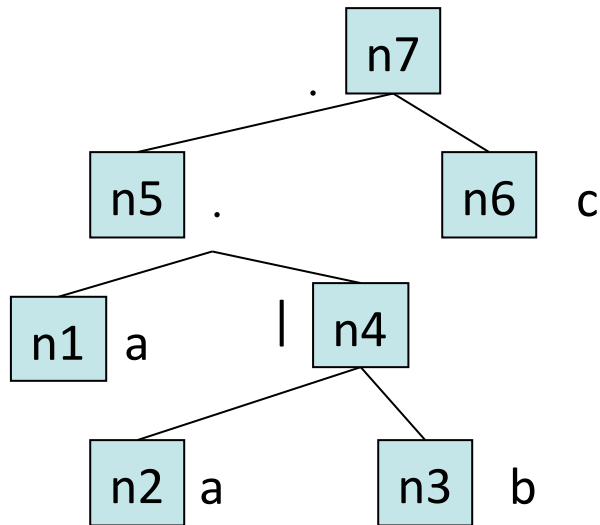
$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$

# Thompson's construction

Converts regexps to NFA

Build NFA recursively  
from regexp tree



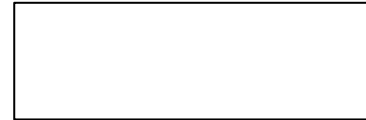
$(a(a|b))c$

$aab|.c.$

pop n7



stack



Post-order traversal of  
regexp tree

$n1 = \text{nfa}(a)$

$n2 = \text{nfa}(a)$

$n3 = \text{nfa}(b)$

$n4 = \text{nfa}(n2, n3, |)$

$n5 = \text{nfa}(n1, n4, .)$

$n6 = \text{nfa}(c)$

$n7 = \text{nfa}(n5, n6, .)$