Regex NFA

Introduction to Regular Expressions and NFA

1. Examine non-deterministic finite automata (NFA)

DFA Quiz

1. A DFA is a five-tuple consisting of:
   - An alphabet
   - A set of states
   - A transition function
   - A start state
   - One or more accepting states

DFA Language Quiz

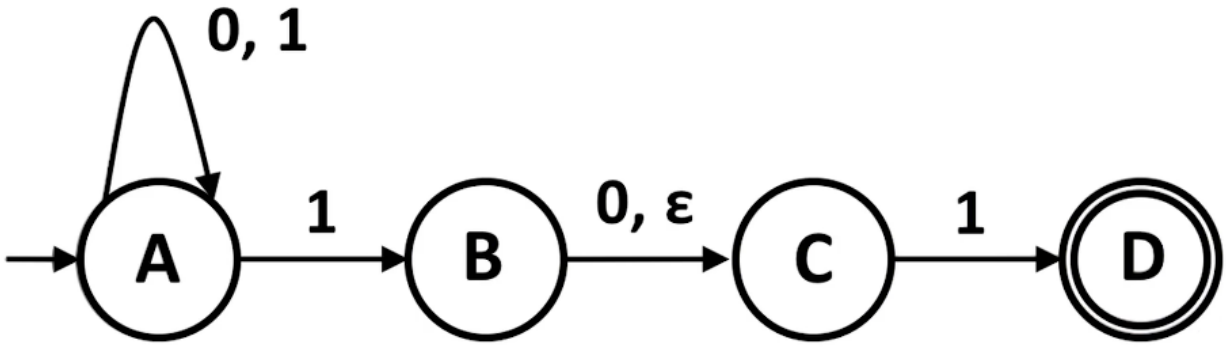
1. Check the box next to the statements that are true:
   - DFAs can only accept finite sized strings and can not deal with infinite sized strings (false)
   - All DFAs have finite number of states (true)
   - DFA is a whimsical machine, even on the exact same input, it behaves differentl every time visiting different set of states when invoked (false)

NFA

1. Nondeterministic Finite Automaton
   - Same input may produce multiple paths
   - Allows transition with an empty string (epsilon)
   - Transition from one state to different states on a given character
2. NFA can clone itself
   - Empty string produces a clone
   - Two transitions based on the same input produce a clone

NFA Example

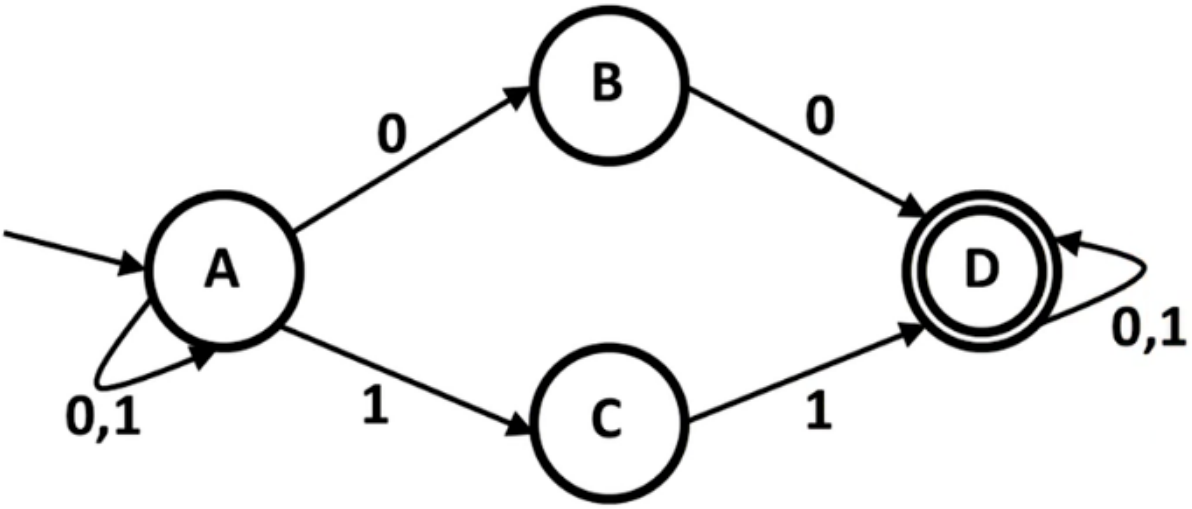
1. NFA can produce multiple copies that operate in parallel on the same input
   - Allow NFAs to explore
   - Only one clone of the NFA has to reach the accepting state to be successful



NFA Example

NFA Quiz

1. What string does this NFA accept?
    - Strings with either two consecutive 0s or 1s
    - Instructor answer: Any binary string that contains 00 or 11



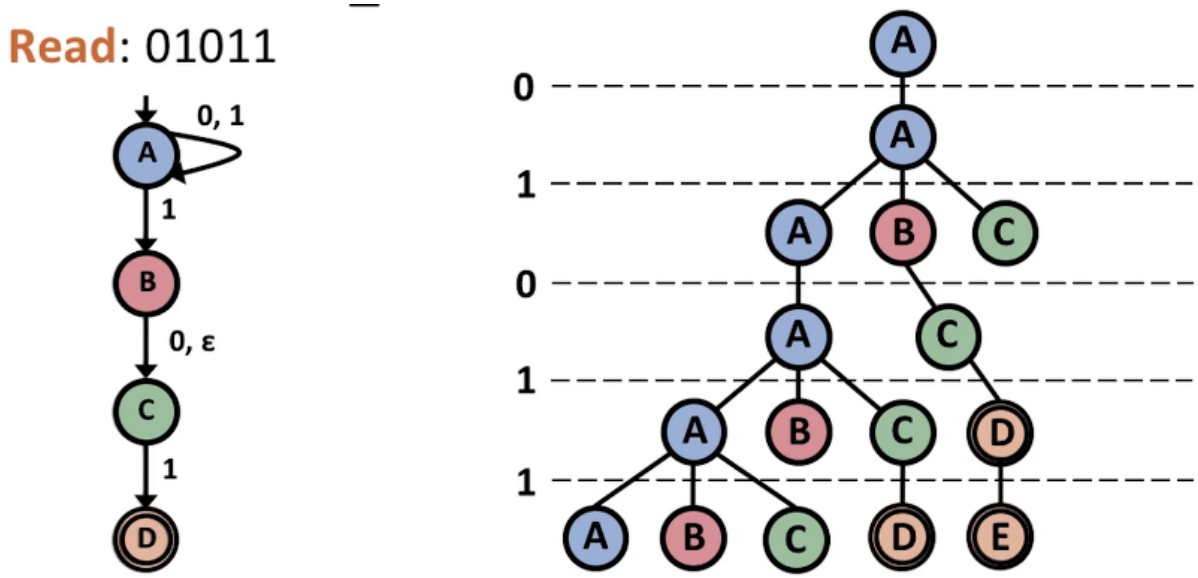
NFA Quiz 1

How NFA Works Part 1

1. Start in start state
    - If encounters and epsilon, clone the machine
    - If there are multiple legal transitions out of a given state, clone the machine
    - No legal transition from the state on incoming alphabet, machine dies
    - If at end of input, if in accepting state, accept input
    - If there are no epsilons or multiple transitions for a given symbol, state transitions act like a DFA

How NFA Works Part 2

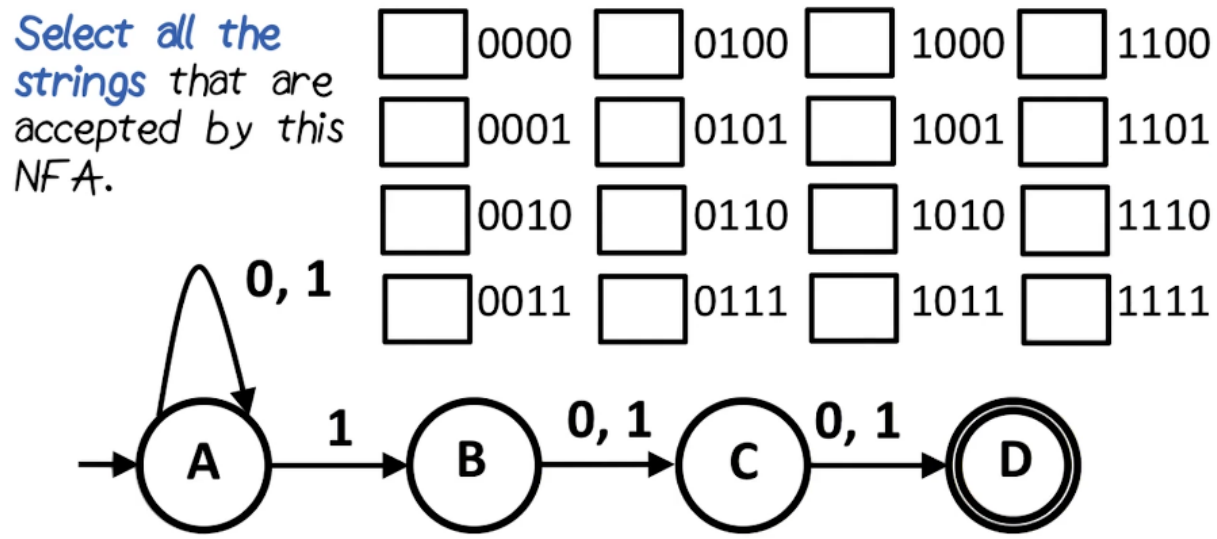
1. Important: It is insufficient to reach the accepting state at some point when processing the string
    - The NFA must process the entire string. If at least one clone is in the accepting state when the input is finished processing, the input is accepted



How an NFA Works

NFA Analysis Quiz

1. Select all the strings that are accepted by this NFA
   - 0000
   - 0001
   - 0010
   - 0011
   - 0100 (accepted)
   - 0101 (accepted)
   - 0110 (accepted)
   - 0111 (accepted)
   - 1000
   - 1001
   - 1010
   - 1011
   - 1100 (accepted)
   - 1101 (accepted)
   - 1110 (accepted)
   - 1111 (accepted)
2. Any string where the second character is a 1 will be accepted



NFA Quiz 2

NFA Defined

1. DFA vs NFA Comparison
   - NFA can transition to multiple states
   - NFA can use epsilon to transition on empty input
2. Language is:
   - DFA: The set of strings such that the DFA ends at an accepting state after processing the string
   - NFA: The set of strings such that NFA ends at an accepting state

DFA	NFA
Alphabet Σ	Alphabet Σ
A set of states Q	A set of states Q
One or more accepting states $F \subseteq Q$	One or more accepting states $F \subseteq Q$
One start state q_0	One start state q_0
A transition function $\delta : Q \times \Sigma \rightarrow Q$	A transition function $\delta : Q \times \Sigma\varepsilon \rightarrow P(Q)$ $\Sigma\varepsilon = \Sigma \cup \{\varepsilon\}$; P(Q) is power set of Q

DFA/NFA Comparison

DFA vs NFA Quiz

1. Put an N for NFA characteristics and D for DFA characteristics
   - The transition from a state is to a single particular next state for each input symbol (D)

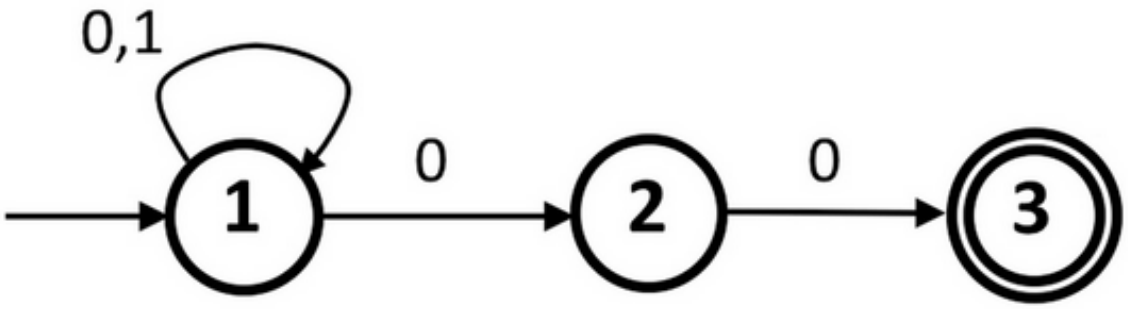
- Permits empty string transitions (N)
- A string is accepted if it ends at an accepting state (D/N)
- Machines are cloned and some clones die (N)

NFA Proof

1. Claim: NFA and DFA are equivalent
   - For every language L accepted by an NFA, there is a DFA that accepts L
   - In other words, DFA and NFA are equivalent computational models
2. Proof: When keeping track of nondeterministic computation of NFA N, use many 'fingers' to point at the set of states of N that can be reached on a given input string

NFA Language Quiz 1

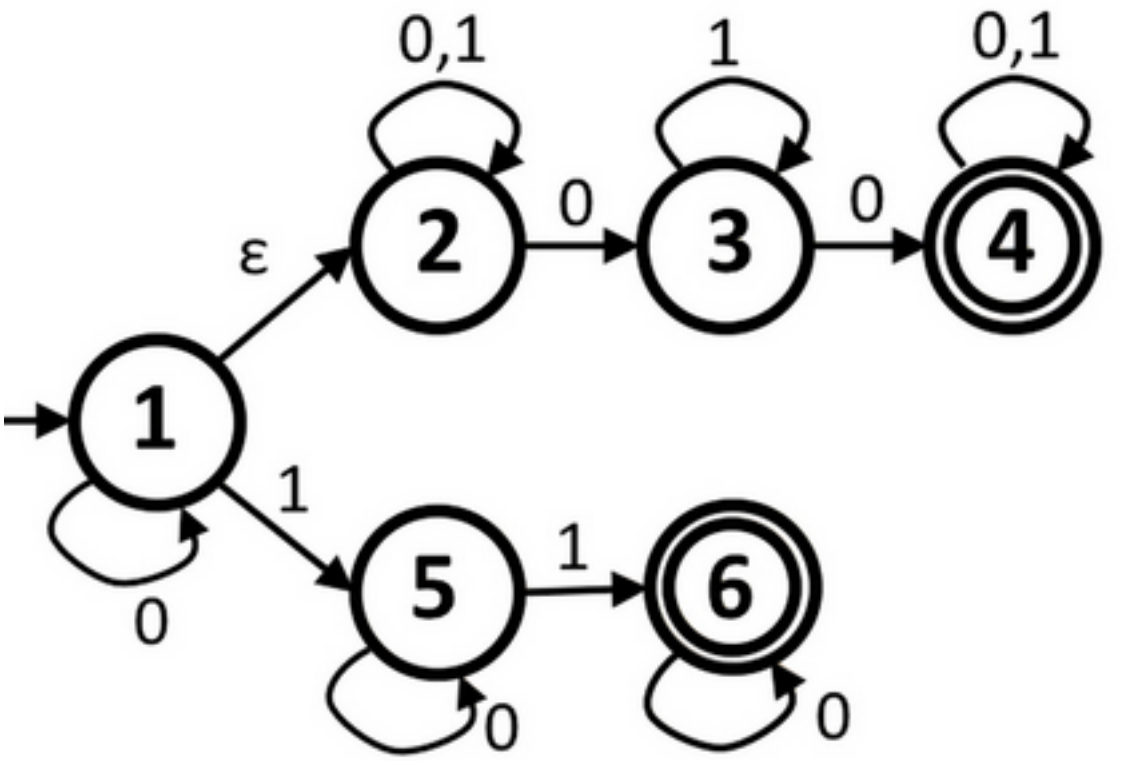
1. Which is the right language described by the NFA?
   - The language w where w ends with 00 (true)
   - The language w where w ends with 10



NFA Quiz 3

NFA Language Quiz 2

1. Which is the right language described by the NFA?
   - The language w where w contains at least two 0s, or exactly two 1s (true)
   - The language w where w contains exactly two 0s, or at least two 1s



NFA Quiz 4

Closure Property Quiz

1. A set is said to be closed for a specific operation, if, when the operation is performed on members of the set, it produces a member of the set
   - Real numbers are closed under division (true)
   - Integers are closed under division (false)
   - Even numbers are closed under division (false)

Closure Properties of Regular Expressions

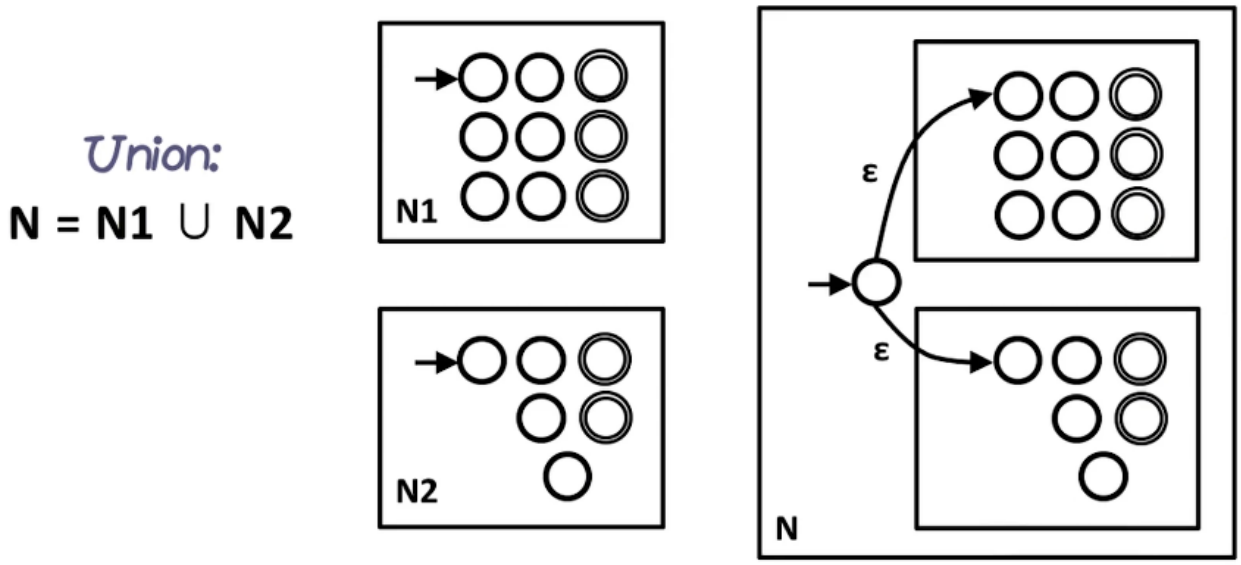
1. Closure Property
   - Certain operations performed on languages in a class will produce results that are in the same class

Regular Operations

1. For languages L1 and L2, we define the following regular operations:
   - Union: L1 U L2 = {x | x in L1 or x in L2}
   - Concatenation: L1 | L2 = {xy, x in L1 and y in L2}
   - Star/repetition: L1* = {x1...xk | xj in L1 for all 1 <= j <= k, k >= 0}

Closure Under Union

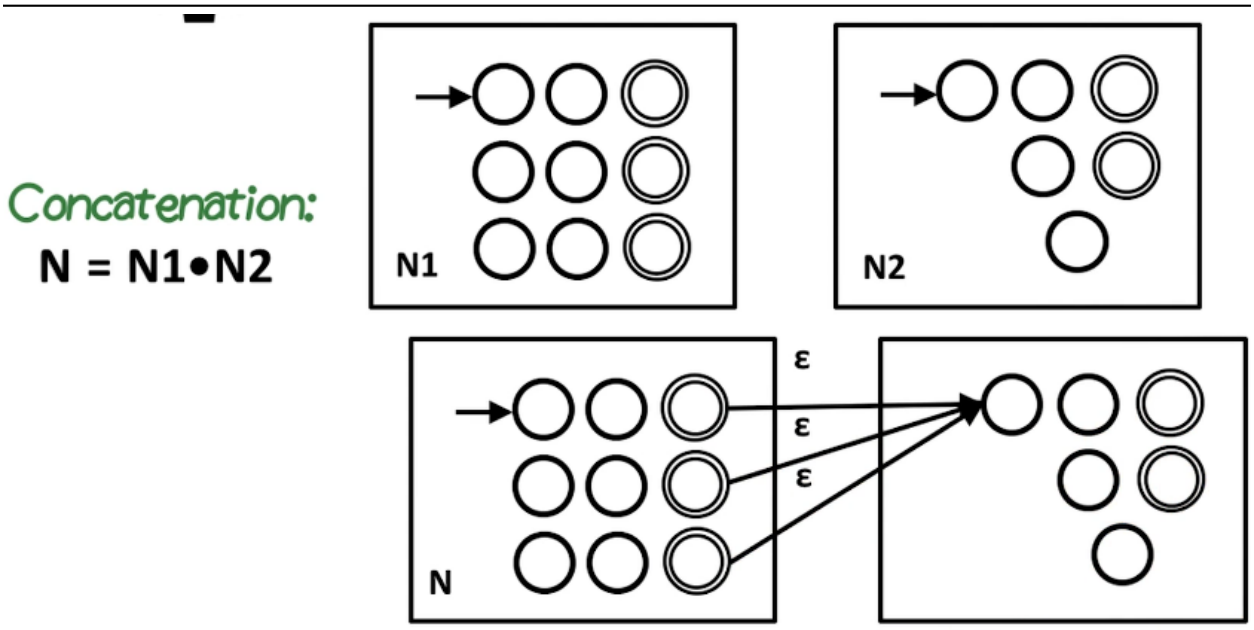
1. The regular languages are 'closed' under the union operation
   - If L1 and L2 are regular languages, then L1 U L2 is regular



Closure Under Union

Closure Under Concatenation

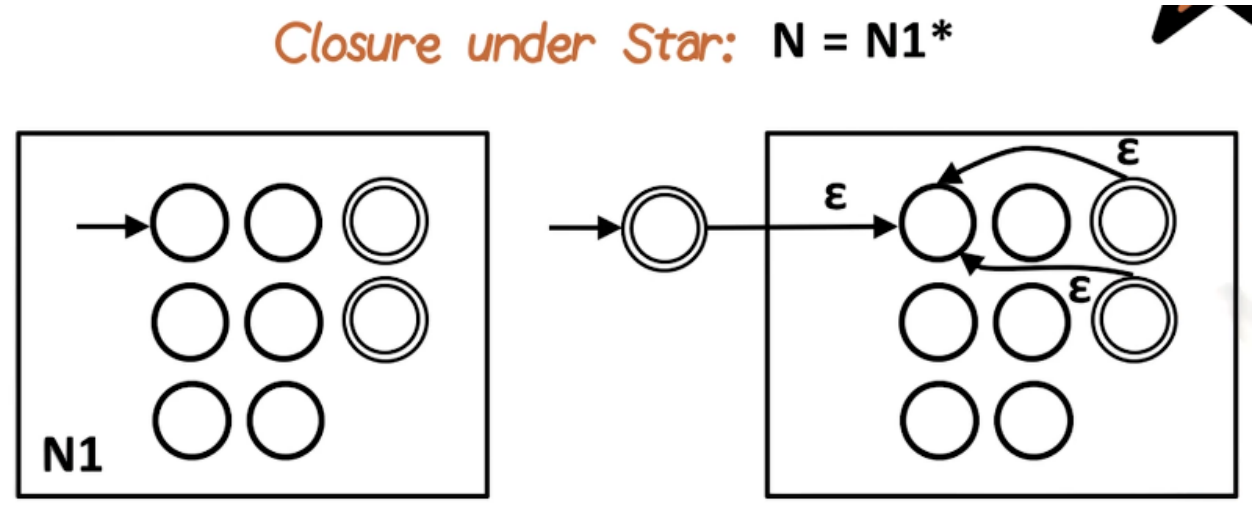
1. The regular languages are 'closed' under the concatenation operation
   - If L1 and L2 are regular languages, then L1 | L2 is regular



Closure Under Concatenation

Closure Under Star

1. Closure under Star: N = N1*



Closure Under Star

DFA Regular Language

1. Regular language = DFA
   - Regular expression <=> NFA <=> DFA = regular language
2. A language is regular if and only if some regular expression describes it
   - If part: If a regular expression describes it, then language is regular
   - Only if part: For every regular language (DFA) there exists a regular expression
   - Regular expression convertible to NFA, convertible to DFA. Thus, DFA exists for every regular expression and thus, lagnuage is regular

Lexical Analysis Introduction

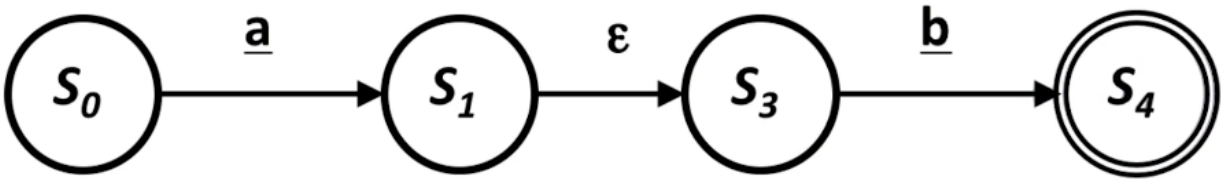
1. How do we convert a regular expression to a scanner?

Automating Scanner Construction

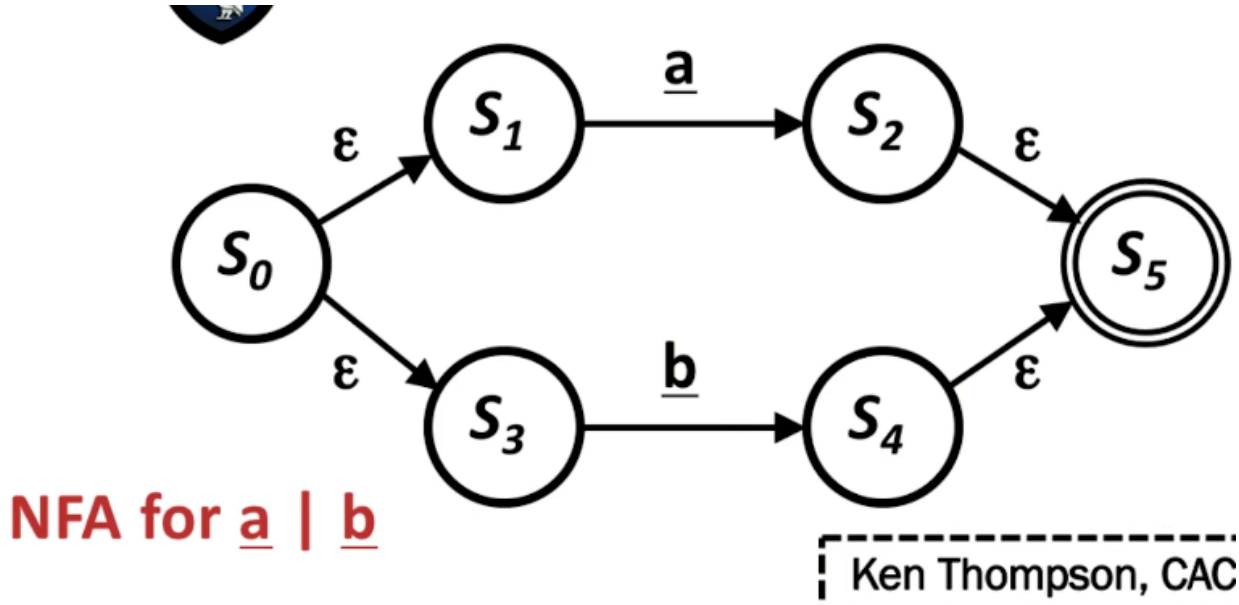
1. To convert a specification into code:
   - Write down the RE for the input language
   - Build a big NFA
   - Build the DFA that simulates the NFA
   - Systematically shrink the DFA
   - Turn it into code
2. Scanner generators
   - Lex and Flex work along these lines
   - Algorithms are well-known and well-understood
   - Key issue is interface to parser (define all parts of speech)
3. The Cycle of Constructions
   - RE -> NFA -> DFA -> Minimal DFA
   - This is a cyclical process

Thompson's Construction

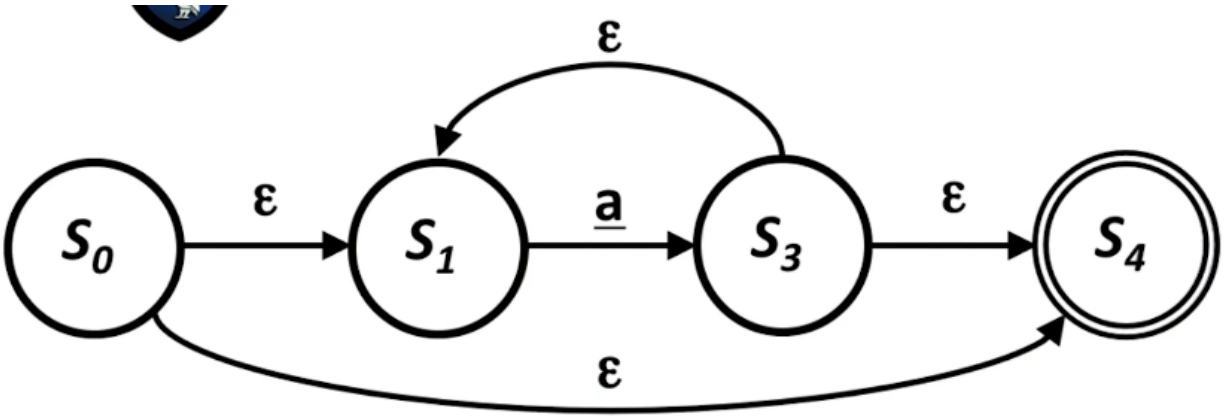
1. Key idea
   - NFA pattern for each symbol and each operator
   - Join them with epsilon moves in precedence order



NFA for ab



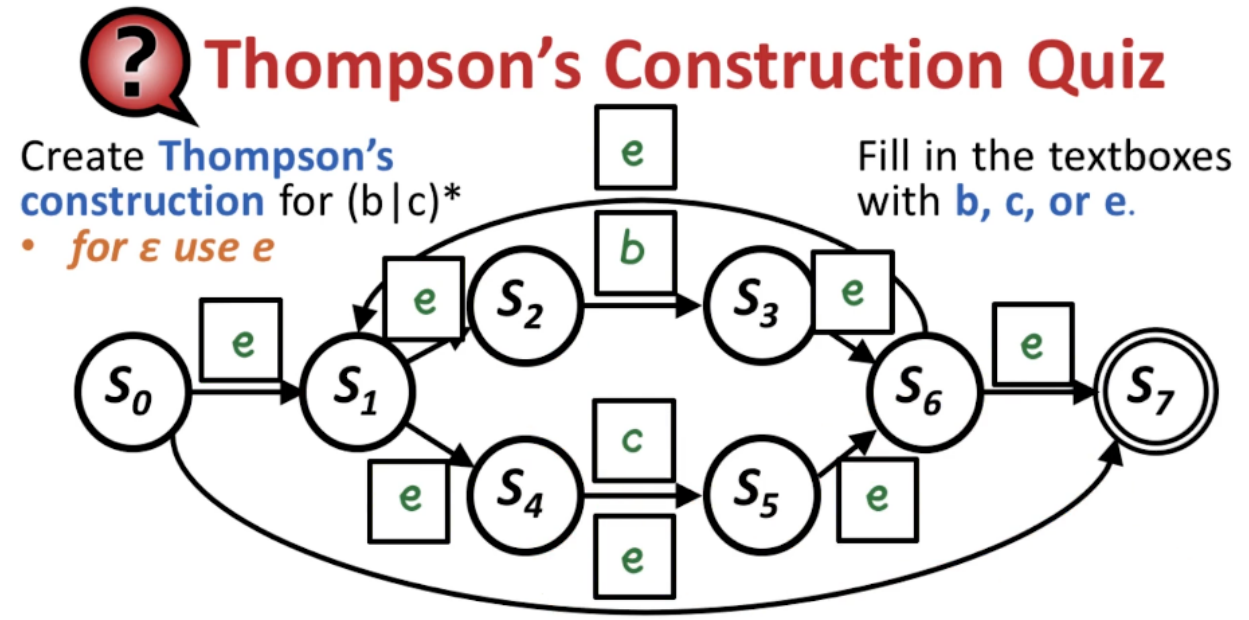
NFA for a



NFA for a*

Thompson's Construction Quiz

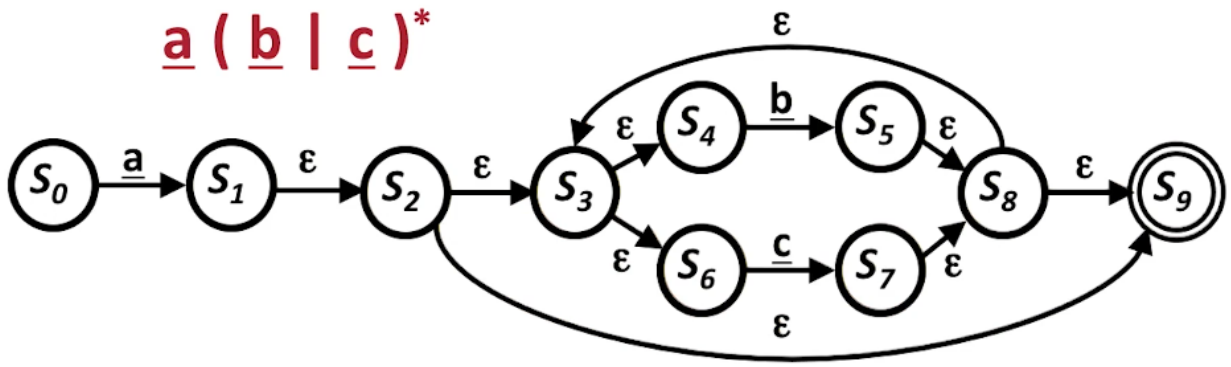
1. Create Thompson's construction for (b|c)*



NFA Quiz 5

Thompson's Construction Continued

1. Thompson's Construction for a(b|c)*



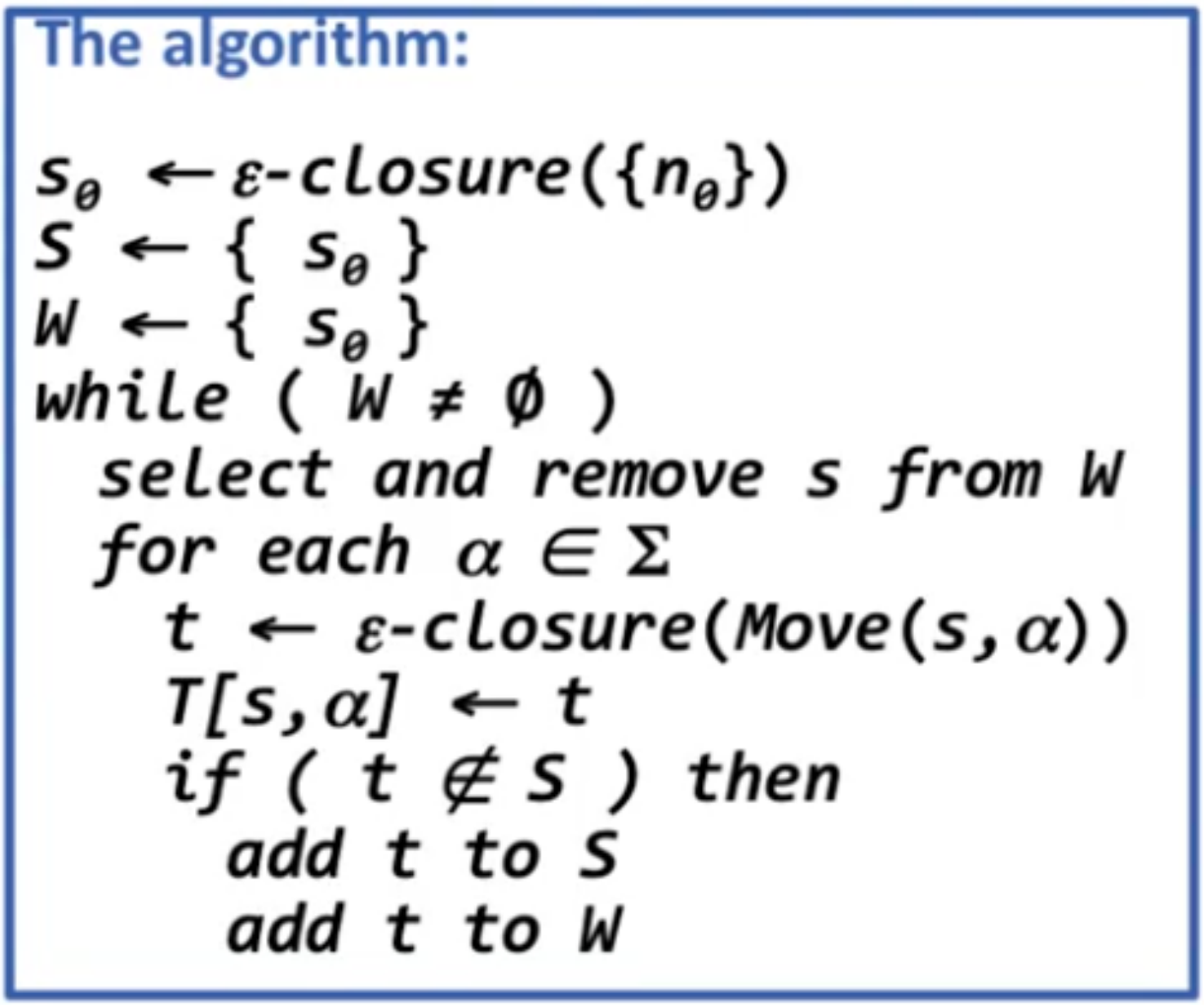
NFA Quiz 5

NFA to DFA Part 1

1. Two key functions:
   - Move(si, a) is the set of states reachable from si by a
   - e-closure(si) is the set of states reachable from si by e
2. High level algorithm
   - Start state derived from s0 of the NFA
   - Take its e-closure S0 = e-closure({s0})
   - Take the image of S0, Move(S0,a) for each a in E, and take its e-closure
   - Iterate until no more states are added

NFA to DFA: The Algorithm

1. The algorithm:



NFA to DFA Algorithm

2. The algorithm halts:
   - S contains no duplicates (test before adding)
   - 2^(NFA states) is finite
   - while loop adds to S, but does not remove from S (monotonic)
     - The loop halts
   - S contains all the reachable NFA states
     - It tries each character in each si
     - It builds every possible NFA configuration

NFA to DFA Part 2

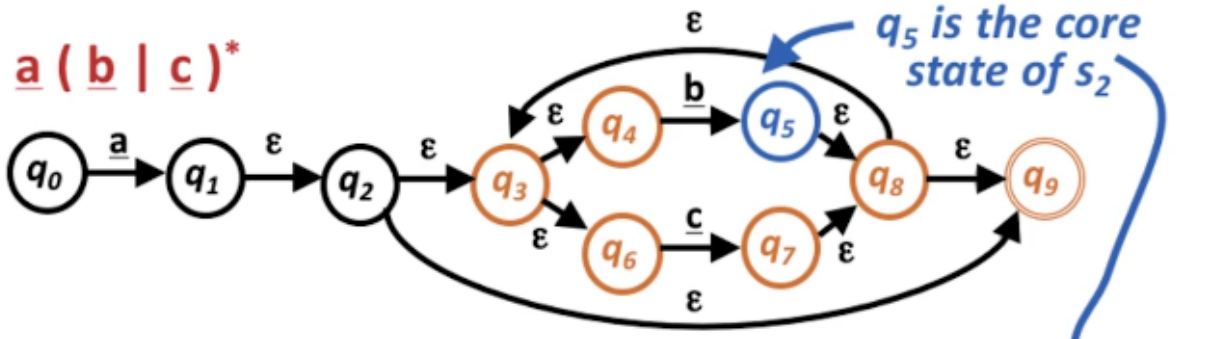
1. How do we determine the final state of the DFA and the different states of the DFA?
   - If the final state of the NFA is part of one of the subsets, then that particular subset is designated as the final state of the DFA
2. Example of a fixed-point computation

- Monotone construction of some finite set
- Halts when it stops adding to the set
- Proofs of halting and correctness are similar
- These computations arise in many contexts

3. Other fixed-point computations
   - Canonical construction of set of LR(1) items
   - Quite similar to the subset construction
   - Classic data-flow analysis (and Gaussian Elimination)
   - Solving sets of simultaneous set equation

Subset Construction Example Part 1 and 2

1. New states are generated when the combination of states hasn't been reached before
2. Epsilon closure means finding all of the states that are reachable given one of the variables
3. How do we determine the final states?
   - String is accepted if one of the NFAs is in the final state
   - s1, s2, and s3 are all the DFA final states



States		ε-closure(Move(s,*))		
DFA	NFA	a	b	c
s_0	q_0	$q_1, q_2, q_3, q_4, q_6, q_9$	none	none
s_1	$q_1, q_2, q_3, q_4, q_6, q_9$	none	$q_5, q_8, q_9, q_3, q_4, q_6$	$q_7, q_8, q_9, q_3, q_4, q_6$
s_2	$q_5, q_8, q_9, q_3, q_4, q_6$	none	s_2	s_3
s_3	$q_7, q_8, q_9, q_3, q_4, q_6$	none	s_2	s_3

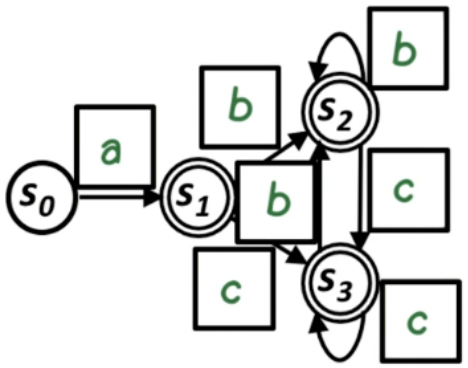
Transition Table for the DFA

NFA to DFA Example

DFA Quiz

Given the diagram and table for a(b|c)*, fill in the textboxes to complete the diagram.

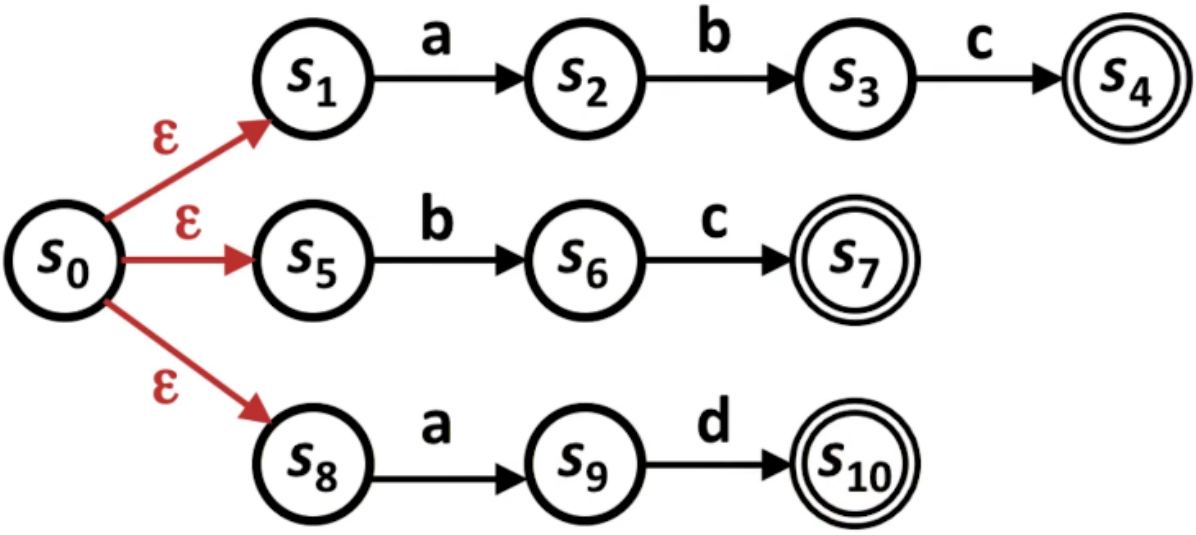
	a	b	c
s_0	s_1	none	none
s_1	none	s_2	s_3
s_2	none	s_2	s_3
s_3	none	s_2	s_3



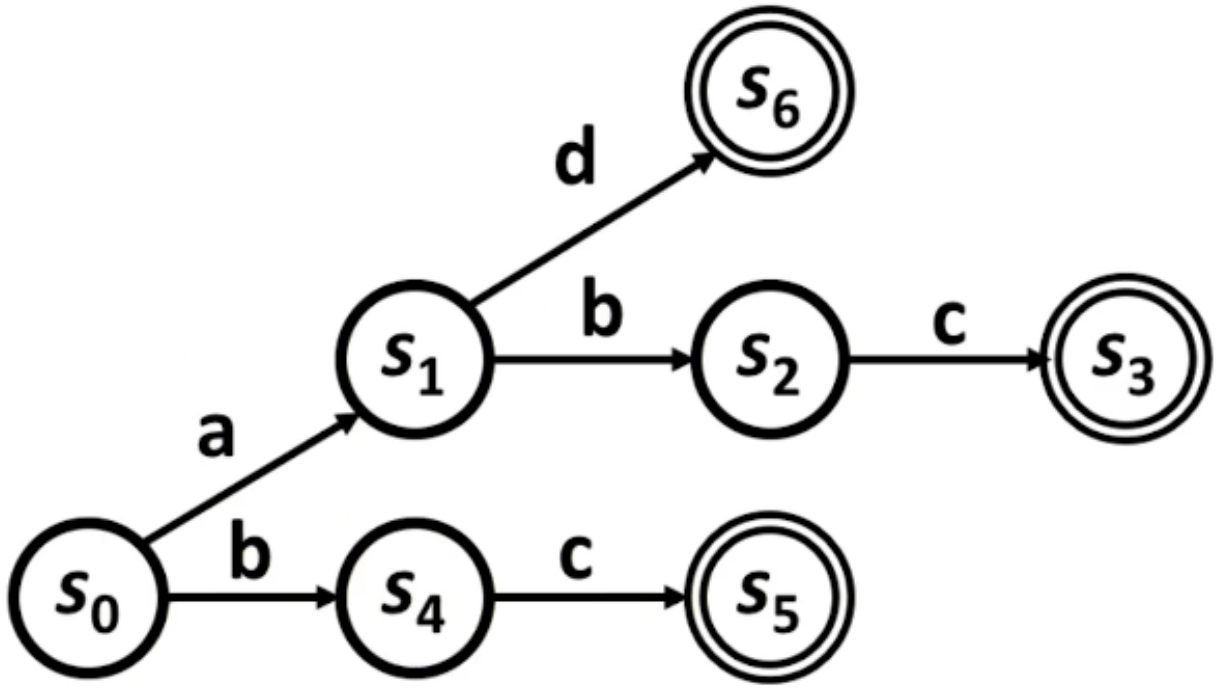
NFA Quiz 6

DFA Minimization Part 1

1. Minimal DFA: DFA with smallest number of states
   - Important to reduce memory consumption of compiler
     - Used to be very important when computers had less memory
     - Size of program you could compile was limited
   - The subset construction merges prefixes in the NFA
   - Idea: Merge states that have the same input and output
     - This only eliminates the prefixes, not the suffixes



Not-minimized NFA



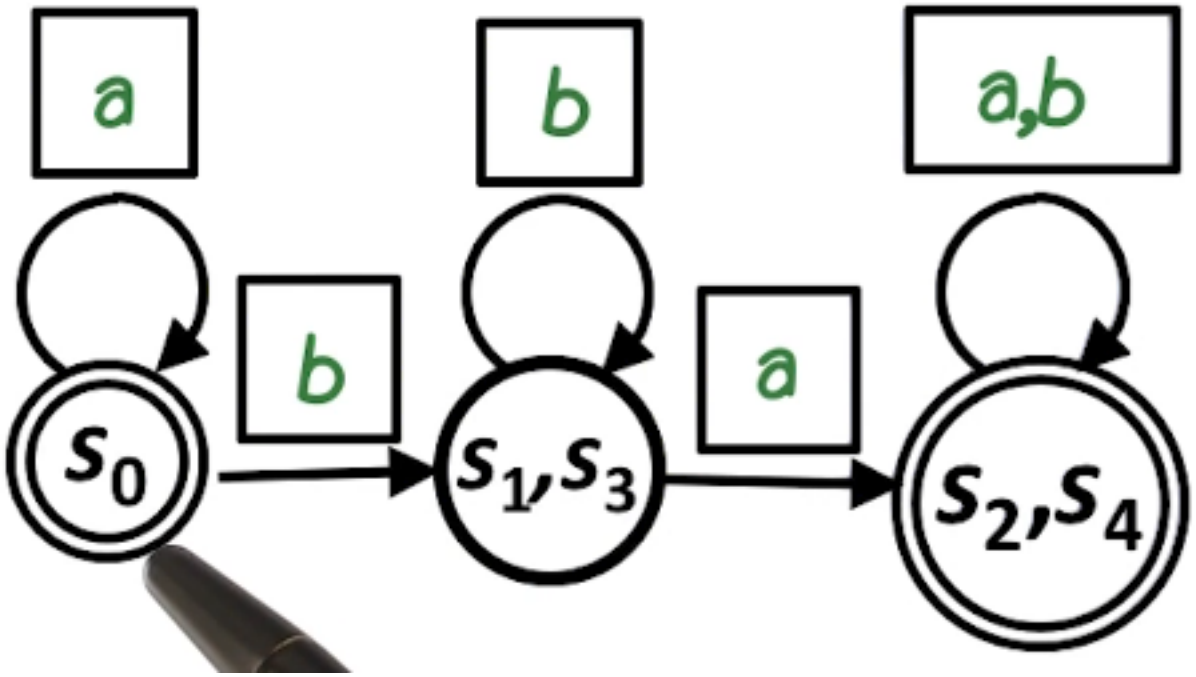
Minimized NFA

2. Idea: Use the subset construction twice
   - For an NFA N:
     - Let reverse(N) be the NFA constructed by making initial states final (& vice-versa) and reversing the edges
     - Let subset(N) be the DFA that results from applying the subset construction to N
     - Let reachable(N) be N after removing all states that are not reachable from the initial state
   - Then,
     - reachable(subset(reverse([reachable(subset(reverse(N))])))
   - This will remove both suffixes and prefixes
     - Brzozowski, 1962

DFA Minimization Part 2

1. Make the initial state the final state and reverse the arrows
   - Requires making a new initial state
2. Run the subset construction
   - Remove the epsilon closures to get to a DFA, combine states
3. Convert the DFA to NFA by adding epsilons and reverse
4. Perform subset construction again
5. DFA minimization reduces the size of the table
   - Does not cut down time for token generation

Minimization Quiz



NFA Quiz 7