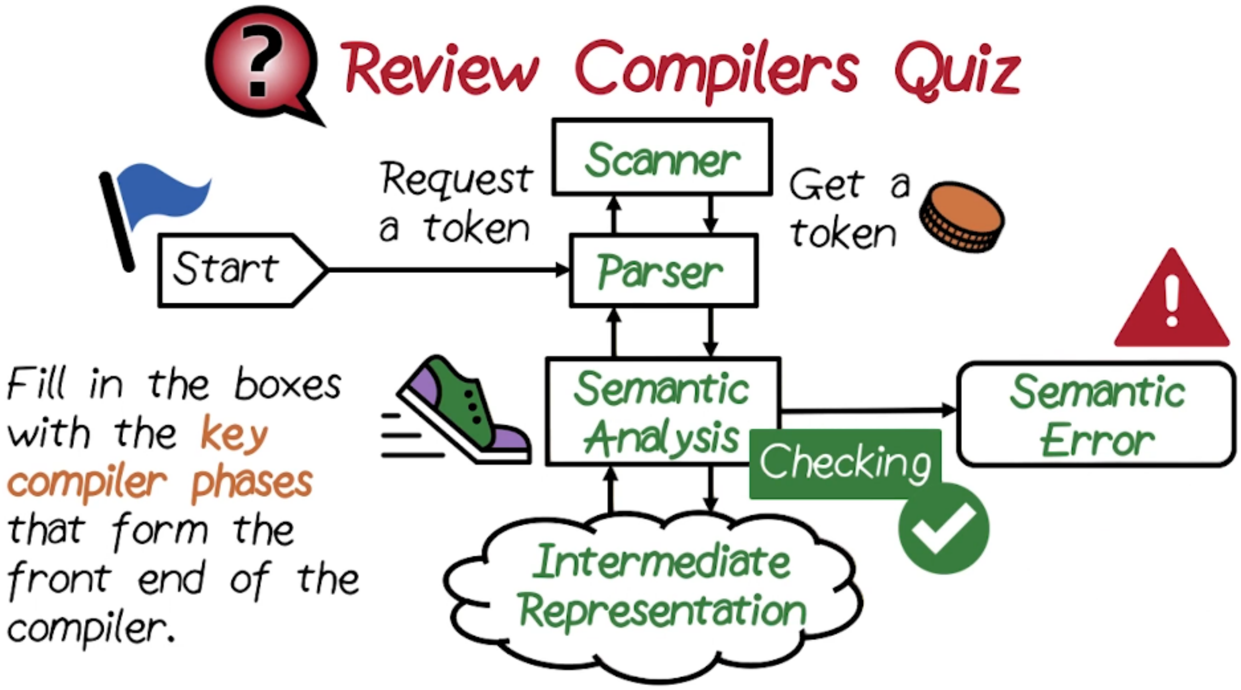
Regex DFA

Introduction to Regular Expressions and DFA

1. First phase of the compiler deals with lexical analysis to group characters into tokens
    - Called a scanner
    - Focus of this lecture
2. Examine regular expressions which serve as a lexical specification of a language
3. Also examine deterministics finite automata which serves as a implementation vehicle of regular expressions

Review Compilers Quiz

1. Fill in the boxes with the key compiler phases that form the front end of the compiler


Review Compilers Quiz
Scanner
Request a token
Get a token
Start
Parser
Fill in the boxes with the key compiler phases that form the front end of the compiler.
Semantic Analysis
Checking
Semantic Error
Intermediate Representation

Phases of a Compiler

Lexical Analysis (Scanner)

1. Job of scanner
    - Read input one character at a time
    - Group characters into tokens
    - Remove whitespaces and comments
    - Encode token types and form tuples and return to parser
        - Token Tuple: <type, value>
        - Example: <FloatConst, 123.45>
        - Example: <VAR, String = DaysOfWeek>
        - Example: <Operator, Value = +>
2. Lexical Language: A collection (set) of legal strings
3. Lexical rules: How to form legal strings

Representation of Strings: Basics of Regular Expression

1. Regular expression r (a pattern of characters) and its language L(r)
    - Used in many Unix programs (grep, vi, etc.)
    - State machine (finite automata)
        - Finite state machine
2. Basics of Regular Expression
    - Alphabet: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz1234567890~!@#$%^&*()_+=-{}|[]:";'<>?,/
    - Symbol: Valid character in a language
    - Metacharacters/metasymbols that have special meanings:
        - Defining reg-ex operations (e.g., |, (,), *, +, etc.)
        - Escape character () to turn off special meanings
        - Empty string and empty set = {}

Representation of Strings: Precedence of Operations

1. Basic Regular Expression
    - Single characters (a = {a})
    - Empty string = ""
    - Empty set = {}
2. Basic Regular Expression Operations
    - Union, denoted by a | b
    - Concatentation, denoted by ab
    - Repetition (Kleen closure, 0 or more times) a*
3. Precedence of operations
    - Repetition (a*) > Concatenation (ab) > Alternation (a | b)
    - Parentheses can be used to change precedence

Representation of Strings: Examples of Regular Expressions

1. Examples of regular expressions:
    - a | b* = {a, "", b, bb, bbb, . . . }
    - (a | b)* = {"", a, b, aa, ab, ba, bb, aaa, aab, . . . }
        - Any number (including 0) of a's and b's in any order
    - (a|c)*b(a|c)*
        - Any number of a's and c's (including 0) in any order followed by a single b followed by any number of a's and c's (including 0) in any order

Unix Style Regular Expressions

Symbol	Definition
.	Denotes any character in alphabet
[]	Character class, allowing range and complement
+	Repeating one or more times
?	Optional (zero or one time)
^ and $	Beginning and end of line

1. Examples
    - [a-d] same as [abcd]
    - [^1-3] denotes characters other than 1, 2, 3

Regular Expressions: Examples

1. [a-zA-Z_][a-zA-Z_0-9]*
    - Identifiers starting with lower case or upper case letters or underscore followed by zero or more of upper or lower case letter or underscore or digits
        - Variable name
2. [0-9][0-9]*
    - Unsigned integers
    - Could have leading zeros, many specifications don't allow this
3. (+|-)?[0-9][0-9]*
    - Signed integers with optional sign

Regular Expressions: Examples Continued

1. [+-]?[0-9]+(\.[0-9]*)?([eE][+-]?[0-9]+)?
    - Floating point numbers - various possibilities
        - 9
        - +8
        - 5.8
        - 5e-88
2. [ˆaeiouAEIOU]
    - Match any character not a vowel
3. [b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z]
    - Match any upper case or lower case consonant

Regular Expressions

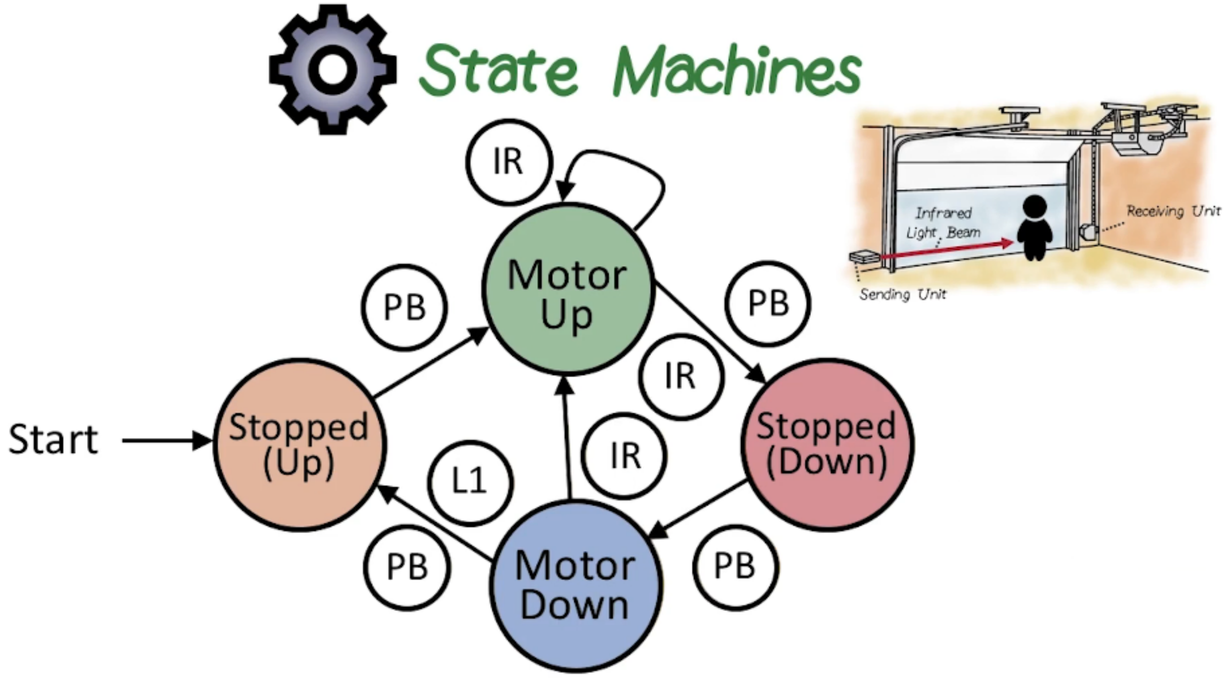
1. Regular Expressions are used in grep, sed, awk, perl, vi, shells, lex, yacc
    - Each may use slightly different convention

RegEx Quiz

1. Write a regular expression consisting of 0's and 1's which may have a 0 but whenever it occurs it must be followed by a 1, empty string ok:
    - (1|01)*
2. An identifier that starts with a lowercase letter or underscore and followed by one or more of lowercase letters and digits and underscores. A letter or digit must follow an underscore:
    - (L | U) (L | D | U (L | D))+
3. A string that consists of at least 3 consecutive 0's:
    - (0 | 1)* 000 (0|1)*

State Machines

1. Use state machines to parse a string
    - Progress from one state to the next
    - If we're in a final state when the parsing finishes, it was a valid string
        - Otherwise, it was illegal
    - PB = push button
    - IR = infrared sensor



Garage Door State Machine

Finite State Machine and Finite Automata

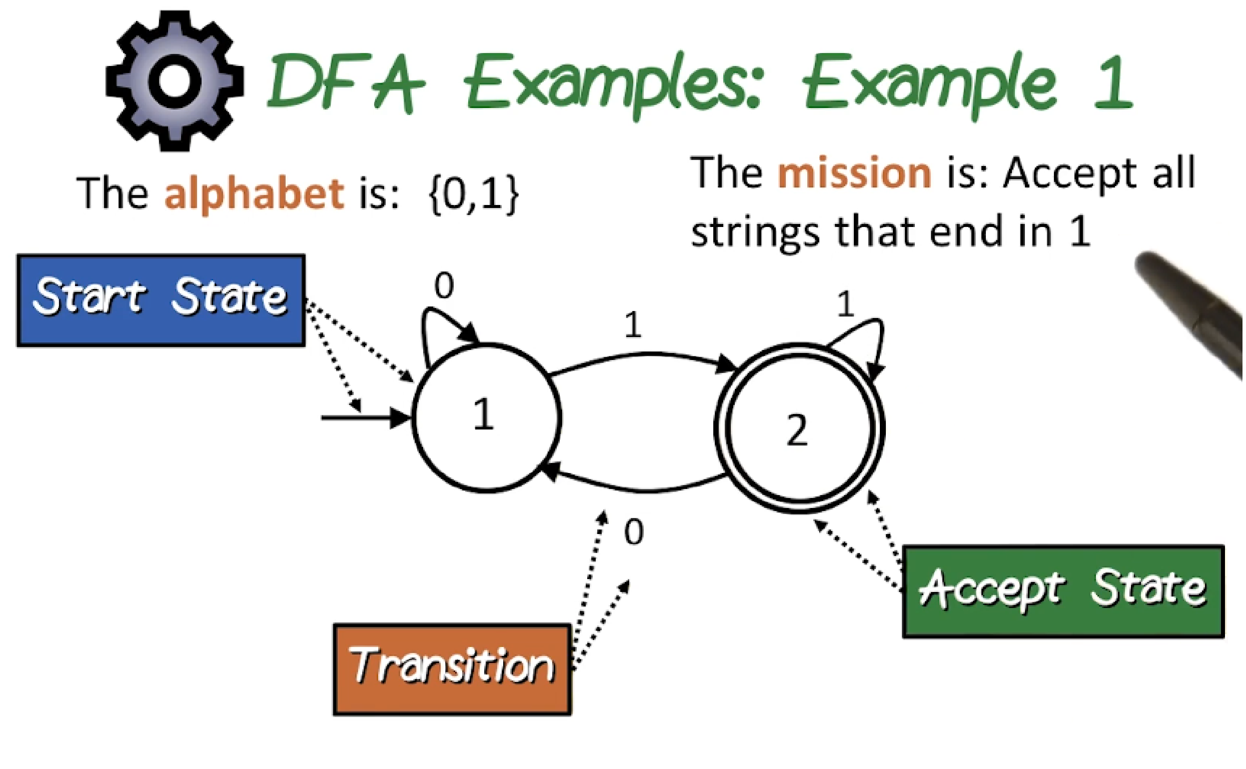
1. Deterministic Finite Automata
   - A simplest model for computing
2. Deterministic: Machine is in a state
   - Upon receipt of a symbol will go to a unique state
3. Finite: Have a finite number of state
4. Automata: Self operating machine
   - Automata: Plural of automaton
5. DFA: Finite-state machine without ambiguity
6. DFA can recognize strings
   - String is input
   - If DFA ends at accept state, string is recognized. Otherwise, it is rejected
7. A language is called a regular language if some finite automaton recognizes it
   - Every regular expression has a DFA associated with it an vice versa

State Machine Quiz

1. Fill in the blanks with Modified, Shared, or Invalid
   - When in the modified state, if a BusRd command is detected, the machine will go to state: Shared
   - When in the Invalid state, if a BusRd command is detected, the machine will go to state: Shared

DFA Example 1

1. The alphabet is {0,1}
2. Mission: Accept all strings that end in 1



DFA Examples: Example 1

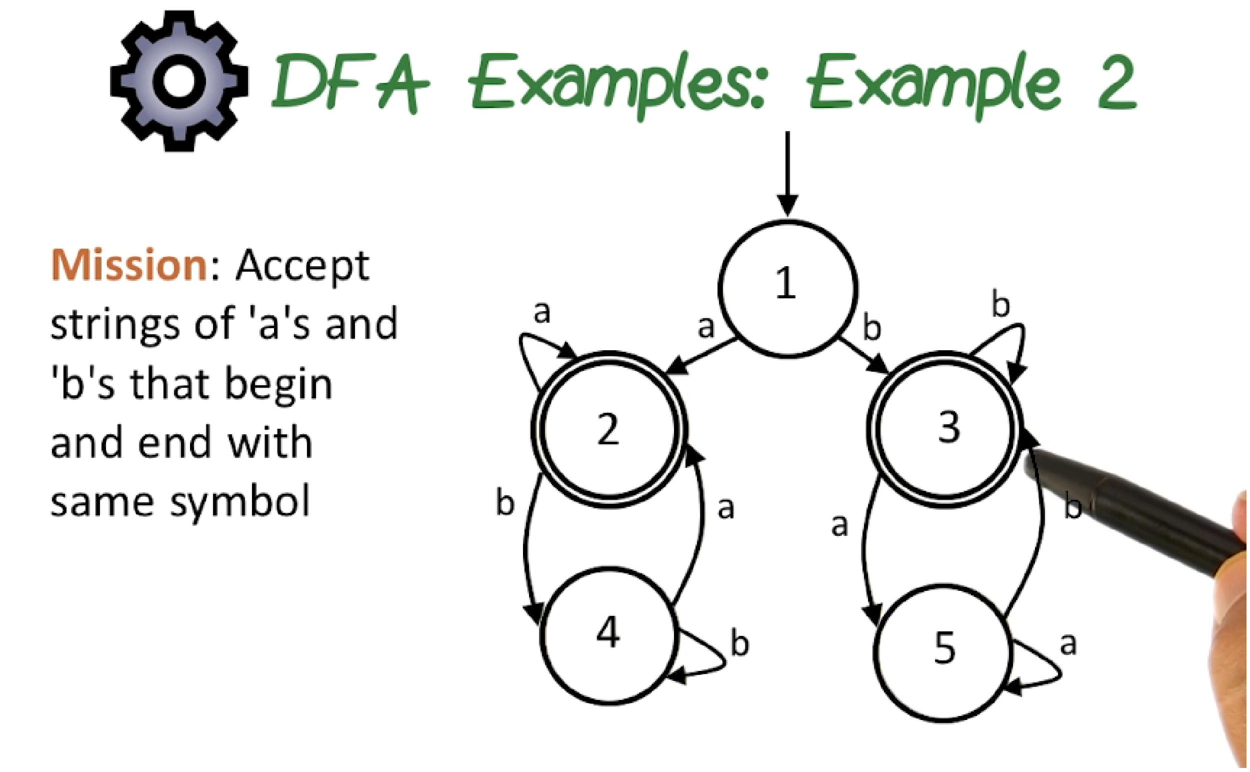
The alphabet is: {0,1}

The mission is: Accept all strings that end in 1

DFA Example 1

DFA Example 2

1. The alphabet is {a,b}
2. Mission: Accept strings of 'a's and 'b's that begin and end with same symbol



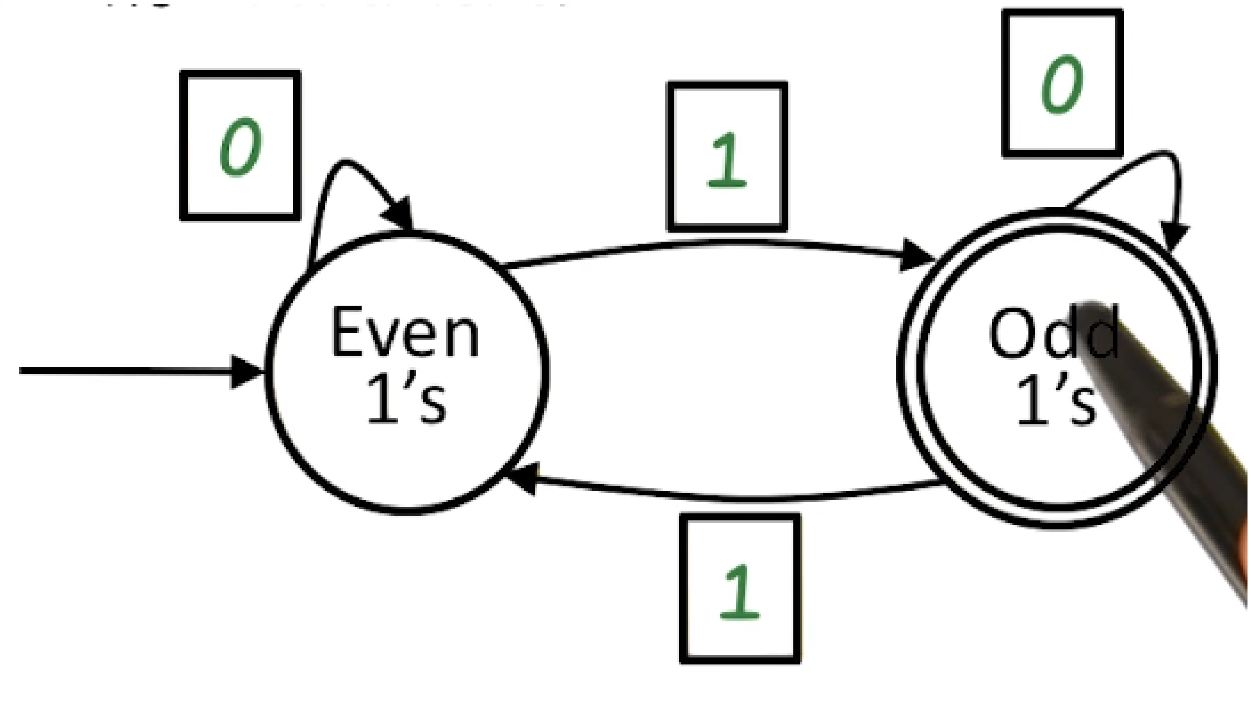
DFA Examples: Example 2

Mission: Accept strings of 'a's and 'b's that begin and end with same symbol

DFA Example 2

DFA Odd Ones Quiz

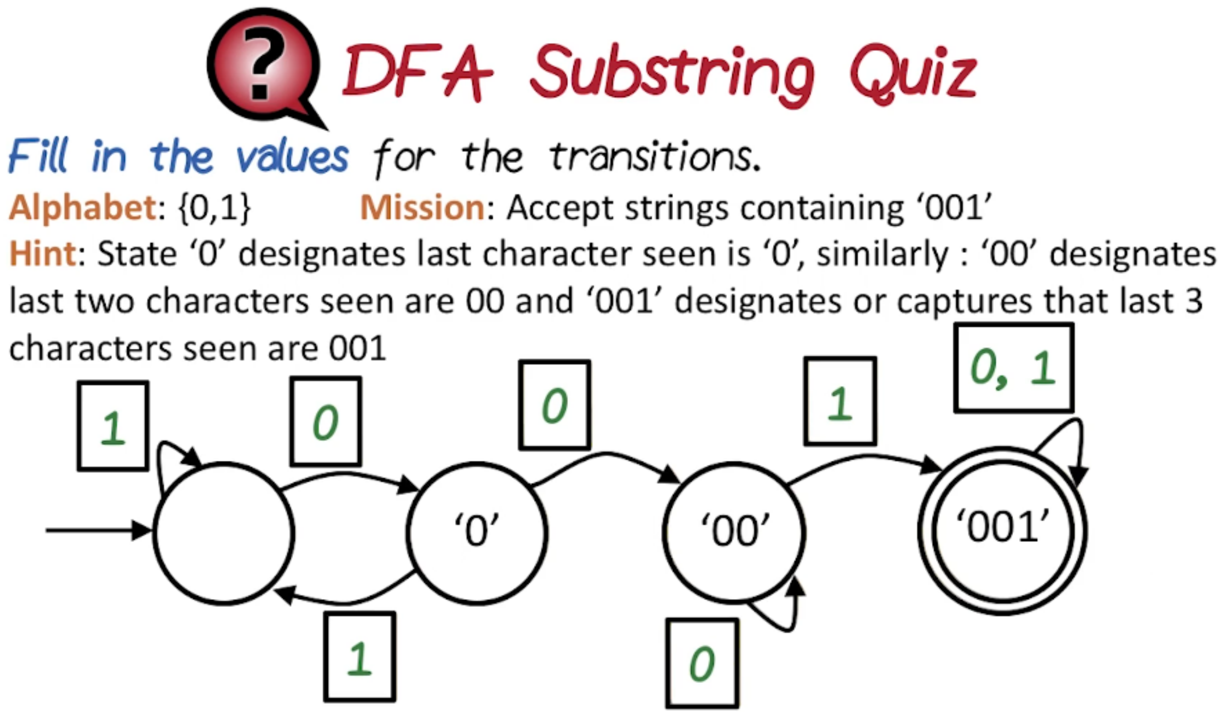
1. Alphabet: {0,1}
2. Mission: Accept strings with an odd number of ones
3. Fill in the values for the transitions.



DFA Quiz 1

DFA Substring Quiz

1. Alphabet: {0,1}
2. Mission: Accept strings containing '001'
3. Fill in the values for the transitions.
    - Hint: State '0' designates last character seen is '0', similarly '00' designates last two characters seen are 00 and '001' designates or captures that last 3 characters seen are 001



DFA Substring Quiz

Fill in the values for the transitions.

Alphabet: {0,1} Mission: Accept strings containing '001'

Hint: State '0' designates last character seen is '0', similarly : '00' designates last two characters seen are 00 and '001' designates or captures that last 3 characters seen are 001

DFA Quiz 2

Formal Definition of DFA

1. A DFA consists of:
   - Alphabet: S
   - A set of states: Q
   - A transition function: T = Q xS -> Q
   - One start state: q0
   - One or more accepting states: F is a subset of Q
2. Language accepted by a DFA is the set of strings such that DFA ends at an accepting state
   - Each string is c1c2. . . cn with ci in S
   - States are qi = T(qi-1,ci) for i=1. . . n
   - qn is an accepting state

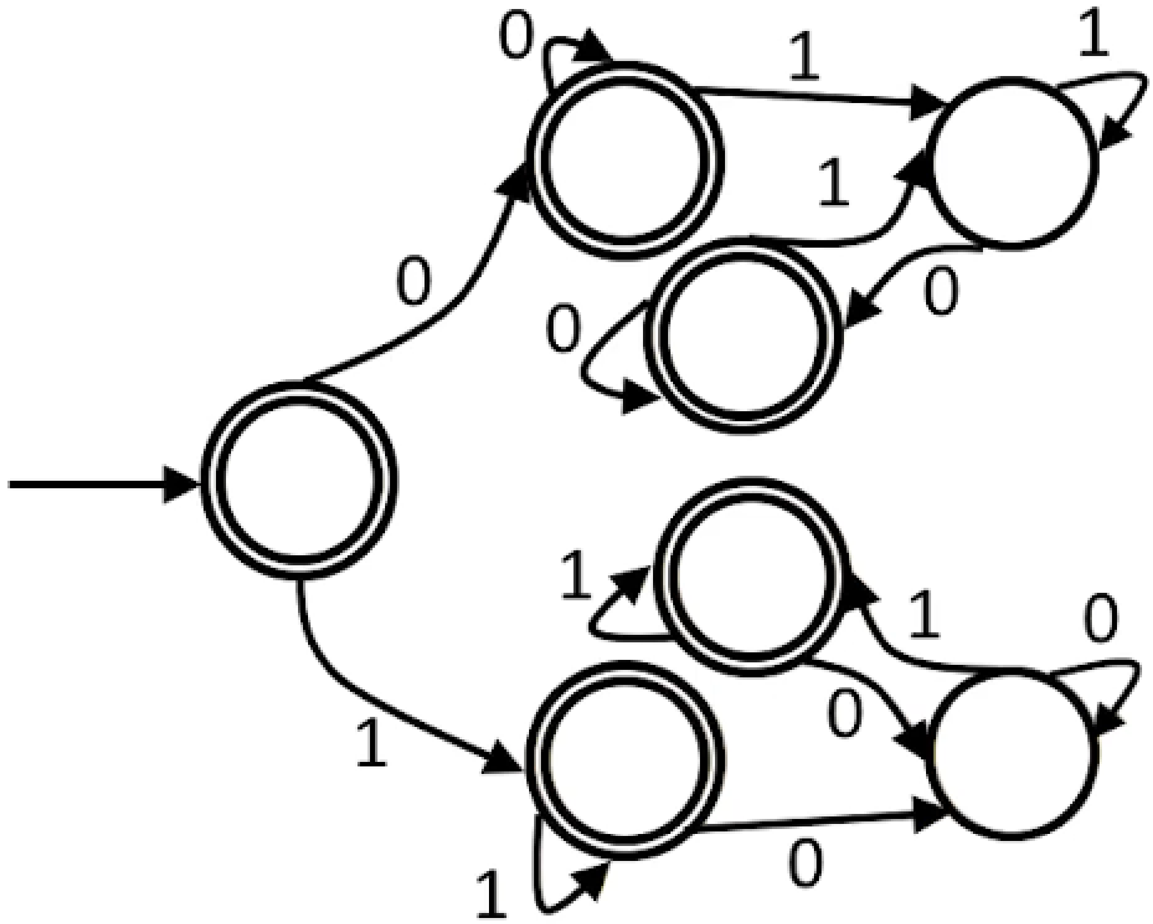
DFA Quiz

1. Can DFA's be designed to accept any string?
   - No
   - We require a finite number of states which means certain strings cannot be recognized by a DFA

DFA String Recognition Quiz

1. Select the strings that a DFA can be designed to detect.
   - Strings that start out with k zeros followed by k ones
     - No; Must keep track of the number of zeros/ones that we've seen, but this number is unbounded. Require infinite states.
   - Strings with an equal number of ones and zeros
     - No; Must keep track of the number of zeros/ones that we've seen, but this number is unbounded. Require infinite states.

- Strings with an equal number of strings “01” and “10”
  - Yes



DFA Quiz 3