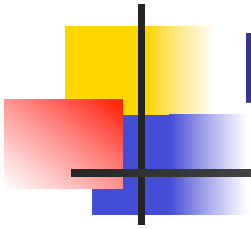
Programming Languages and Compilers (CS 421)

Elsa L Gunter

2112 SC, UIUC

http://courses.engr.illinois.edu/cs421

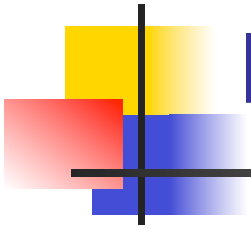
Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha



Regular Expressions

- Start with a given character set – **a, b, c**…

- Each character is a regular expression
  - It represents the set of one string containing just that character



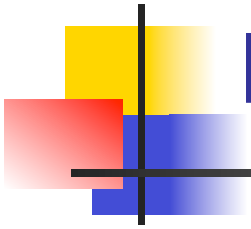
Regular Expressions

- If **x** and **y** are regular expressions, then **xy** is a regular expression
  - It represents the set of all strings made from first a string described by **x** then a string described by **y**

If x={a,ab} and y={c,d} then xy ={ac,ad,abc,abd}.

- If **x** and **y** are regular expressions, then **x**v**y** is a regular expression
  - It represents the set of strings described by either **x** or **y**

If x={a,ab} and y={c,d} then x v y={a,ab,c,d}



Regular Expressions

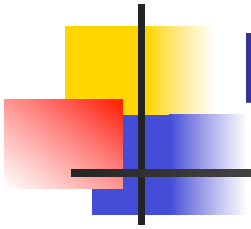
- If **x** is a regular expression, then so is (**x**)
  - It represents the same thing as **x**

- If **x** is a regular expression, then so is **x***
  - It represents strings made from concatenating zero or more strings from **x**

  If x = {a,ab}

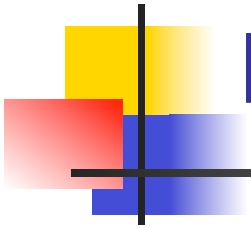
  then x* ={"",a,ab,aa,aab,abab,aaa,aaab,…}

- ε
  - It represents {""}, set containing the empty string



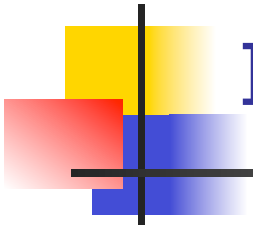
Example Regular Expressions

- **(0v1)*1**
  - The set of all strings of **0**'s and **1**'s ending in 1, **{1, 01, 11,…}**
- **a*b(a*)**
  - The set of all strings of a's and b's with exactly one b
- **((01) v(10))***
  - You tell me
- Regular expressions (equivalently, regular grammars) important for lexing, breaking strings into recognized words



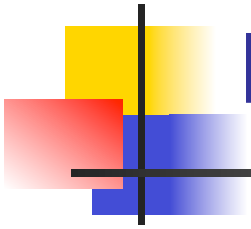
Example: Lexing

- Regular expressions good for describing lexemes (words) in a programming language
  - Identifier = (a ∨ b ∨ … ∨ z ∨ A ∨ B ∨ … ∨ Z) (a ∨ b ∨ … ∨ z ∨ A ∨ B ∨ … ∨ Z ∨ 0 ∨ 1 ∨ … ∨ 9)*
  - Digit = (0 ∨ 1 ∨ … ∨ 9)
  - Number = 0 ∨ (1 ∨ … ∨ 9)(0 ∨ … ∨ 9)* ∨ ~ (1 ∨ … ∨ 9)(0 ∨ … ∨ 9)*
  - Keywords: if = if, while = while,…



Implementing Regular Expressions

- Regular expressions reasonable way to generate strings in language
- Not so good for recognizing when a string is in language
- Problems with Regular Expressions
  - which option to choose,
  - how many repetitions to make
- Answer: finite state automata

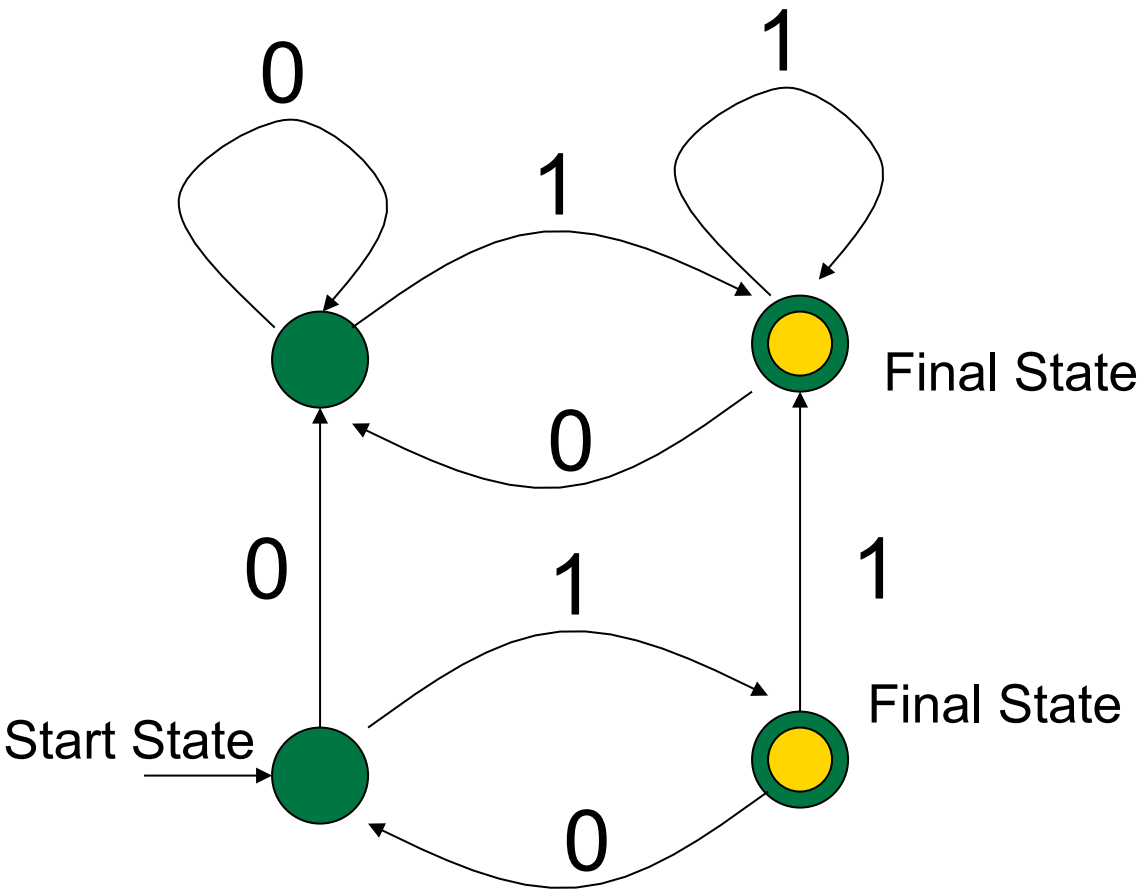


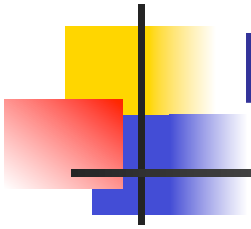
Finite State Automata

- A finite state automata over an alphabet is:
  - a directed graph
  - a finite set of states defined by the nodes
  - edges are labeled with elements of alphabet, or empty string; they define state transition
  - some nodes (or *states*), marked as final
  - one node marked as start state

- Syntax of FSA

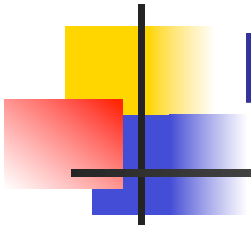
Example FSA





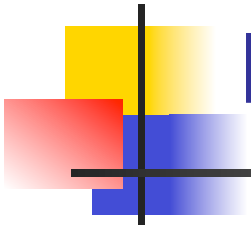
Deterministic FSA's

- If FSA has for every state *exactly one* edge for each letter in alphabet then FSA is *deterministic*
  - No edge labeled with ε
- In general FSA in *non-deterministic*.
  - NFSA also allows edges labeled by ε
- Deterministic FSA special kind of non-deterministic FSA



DFSA Language Recognition

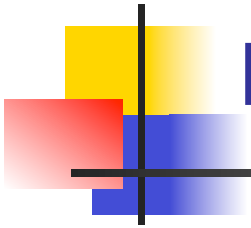
- Think of a DFSA as a board game; DFSA is board
- You have string as a deck of cards; one letter on each card
- Start by placing a disc on the start state



DFSA Language Recognition

- Move the disc from one state to next along the edge labeled the same as top card in deck; discard top card

- When you run out of cards,
  - if you are in final state, you win; string is in language
  - if you are not in a final state, you lose; string is not in language



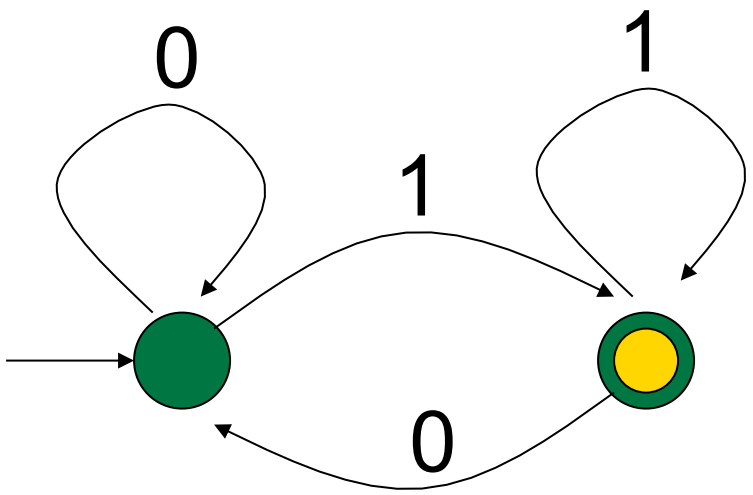
DFSA Language Recognition -Summary

- Given a string over alphabet
- Start at start state
- Move over edge labeled with first letter to new state
- Remove first letter from string
- Repeat until string gone
- If end in final state then string in language

- Semantics of FSA

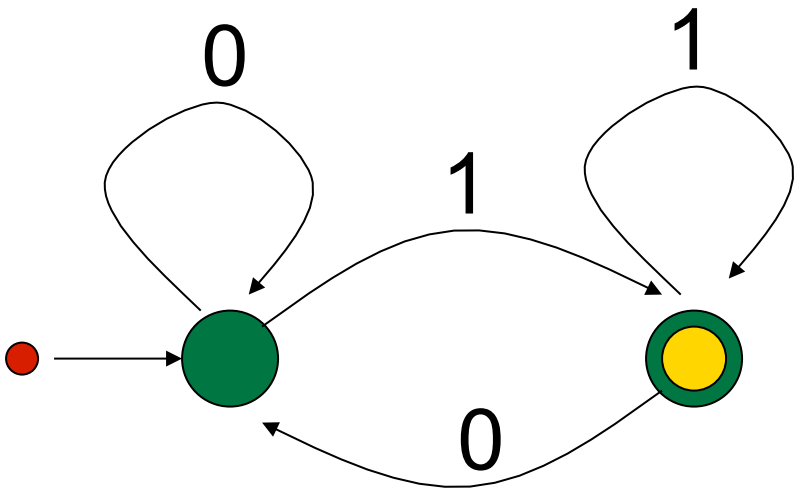
Example DFSA

- Regular expression: (0 v 1)* 1
- Deterministic FSA



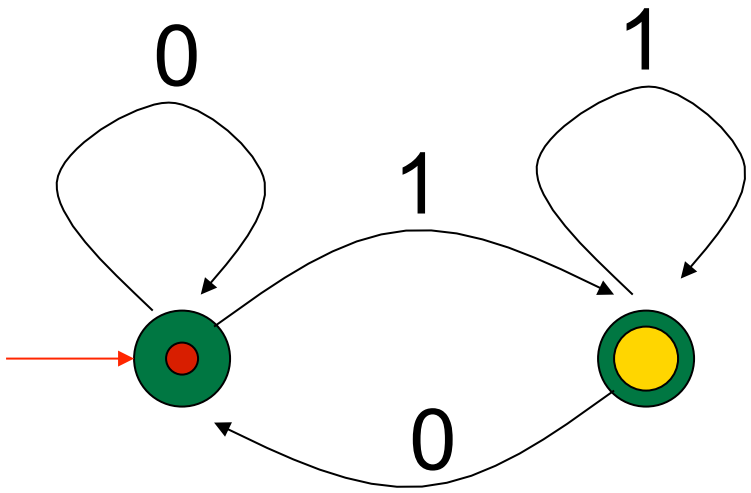
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



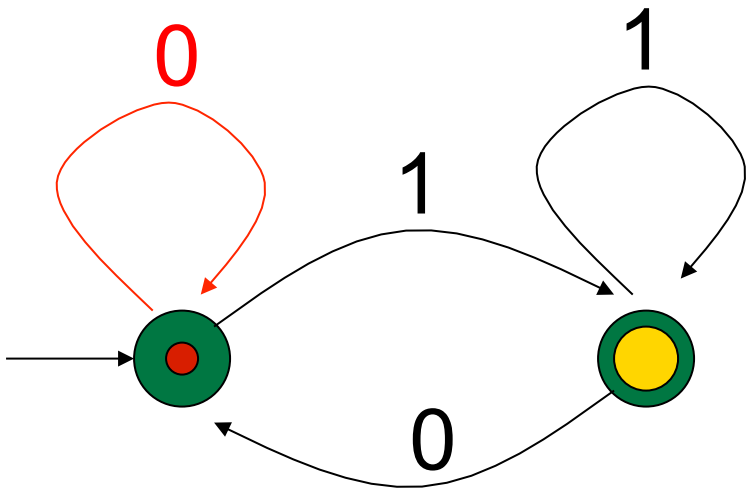
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



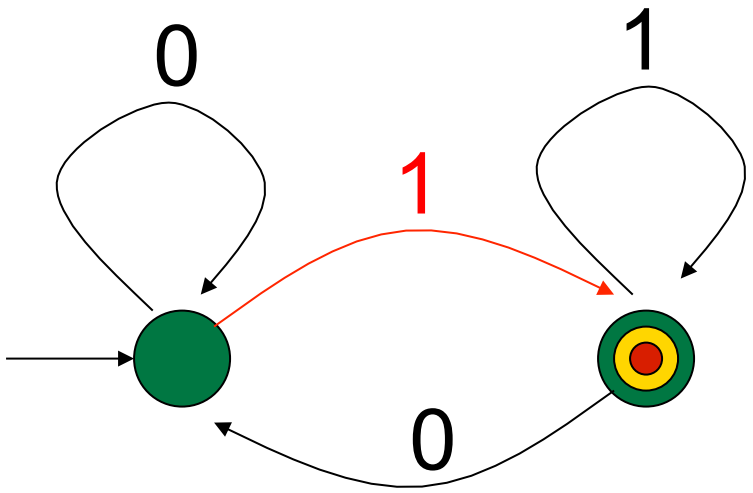
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string ~~0~~ 1 1 0 1



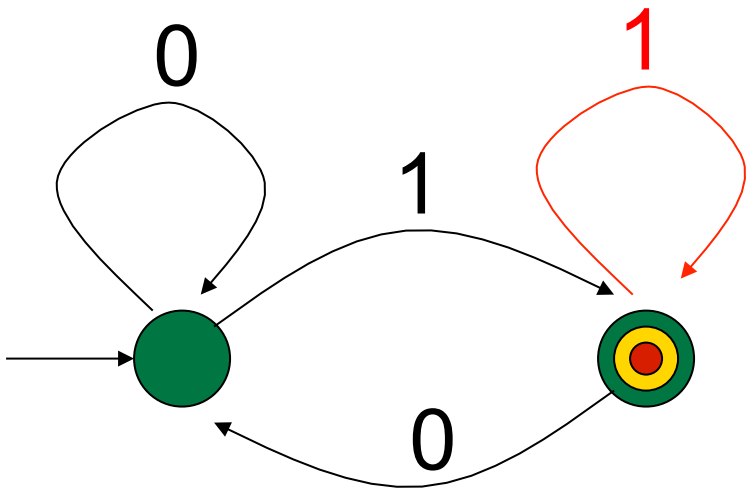
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



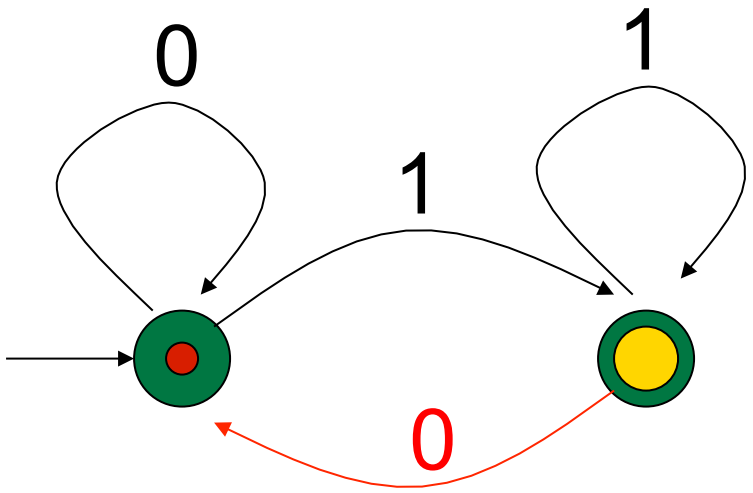
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



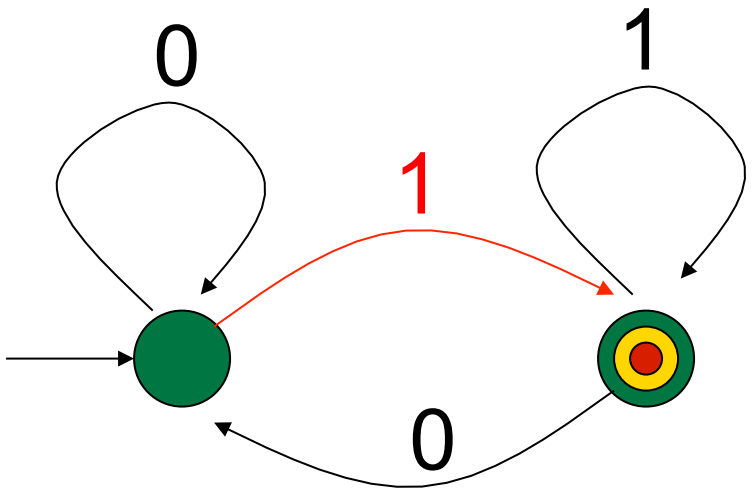
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



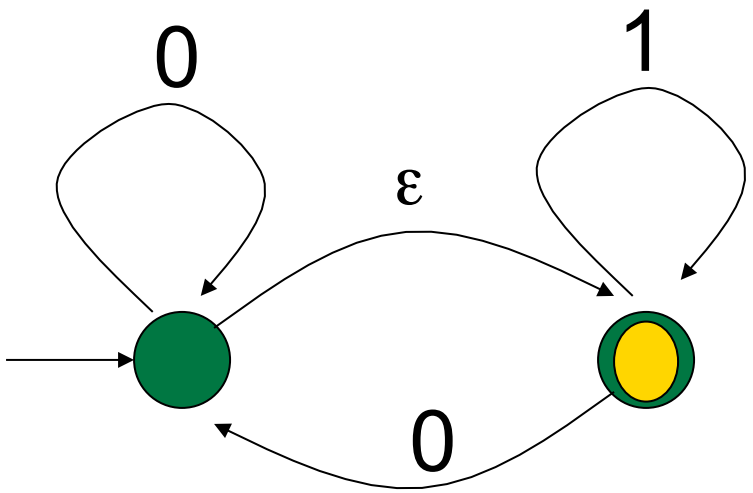
Example DFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



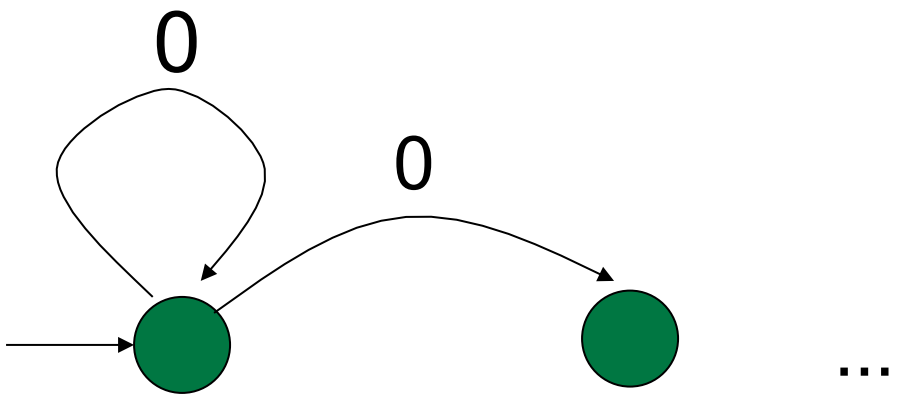
Non-deterministic FSA's

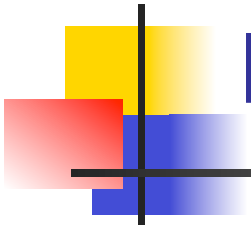
- NFSA generalize DFSA in two ways:
- Include edges labeled by ε
  - Allows process to non-deterministically change state



Non-deterministic FSA's

- Each state can have zero, one or more edges labeled by each letter
  - Given a letter, non-deterministically choose an edge to use



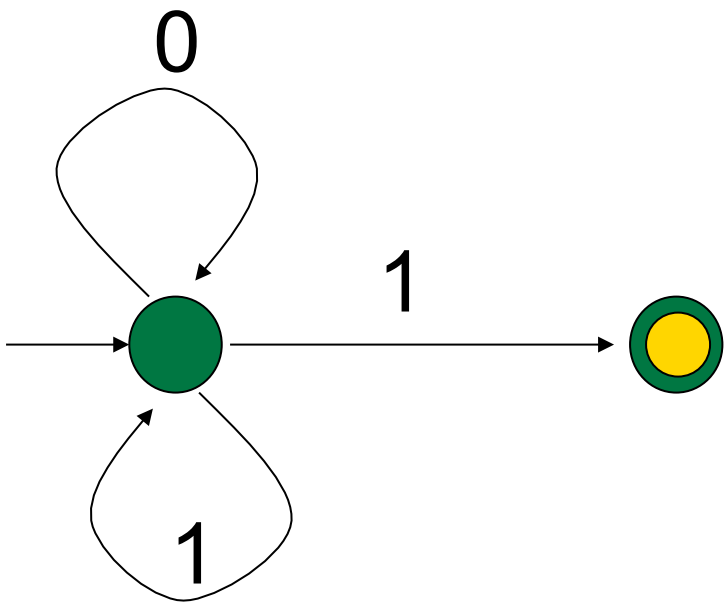


NFSA Language Recognition

- Play the same game as with DFSA
- Free move: move across an edge with empty string label without discarding card
- When you run out of letters, if you are in final state, you win; string is in language
- You can take one or more moves back and try again
- If have tried all possible paths without success, then you lose; string not in language

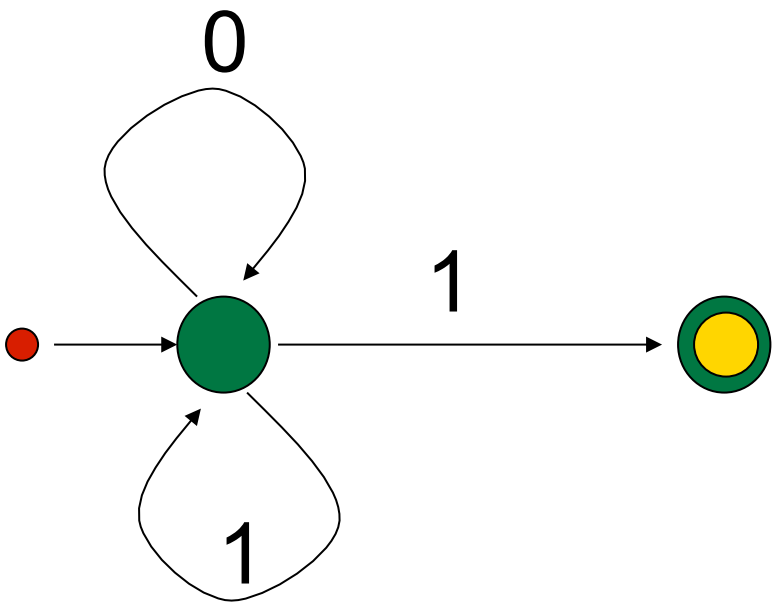
Example NFSA

- Regular expression: (0 v 1)* 1
- Non-deterministic FSA



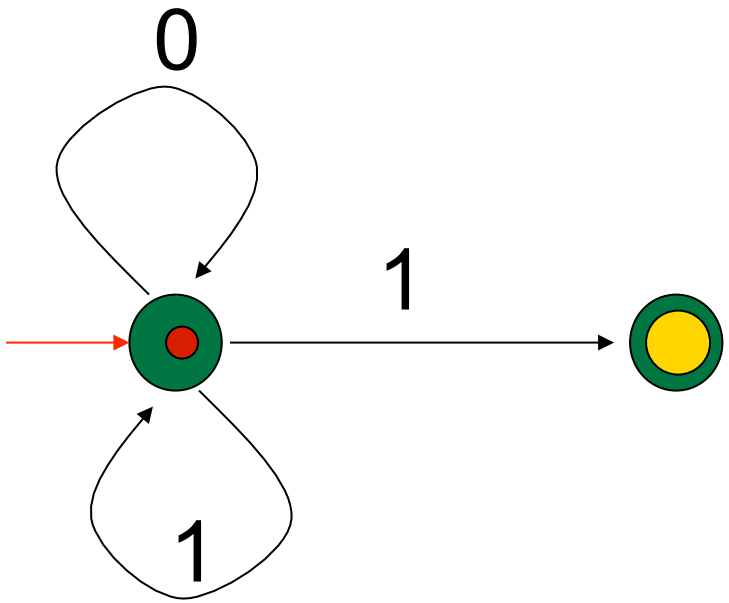
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



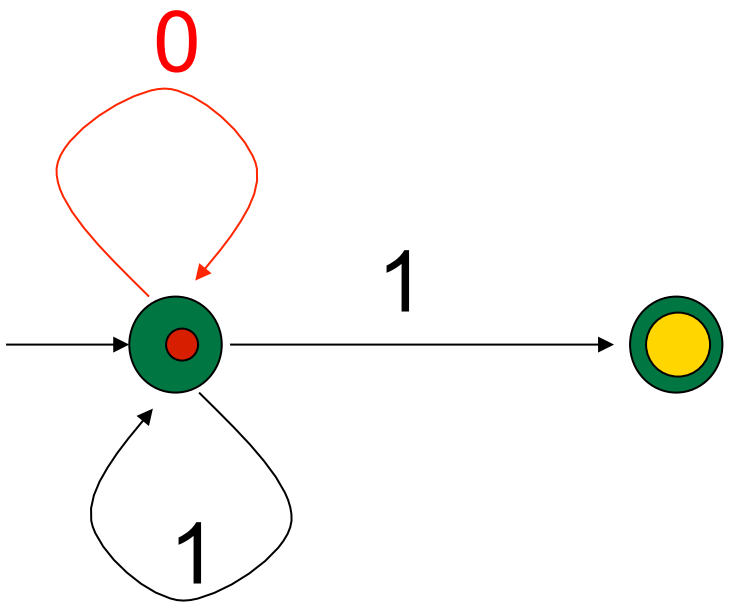
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



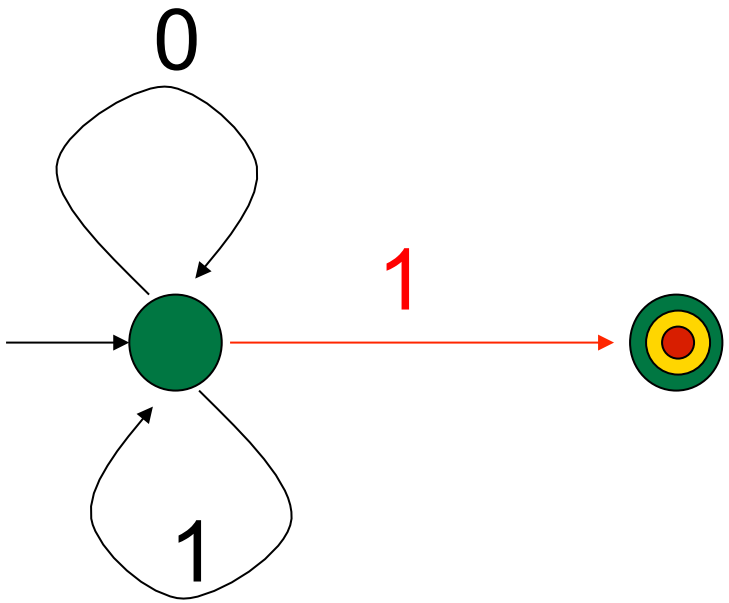
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string ~~0~~ 1 1 0 1



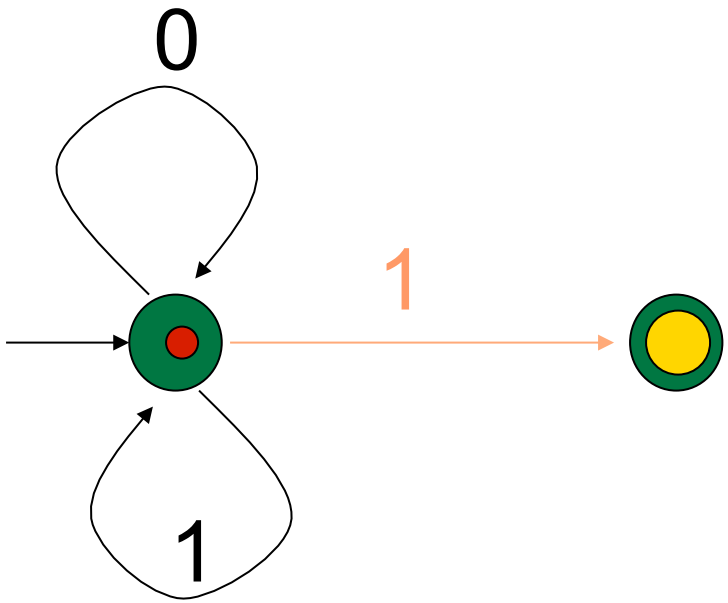
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1
- Guess



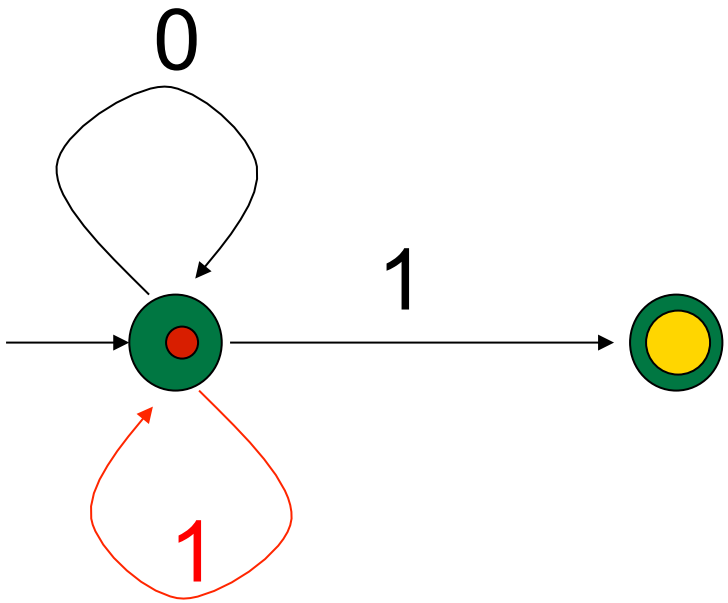
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1
- Backtrack



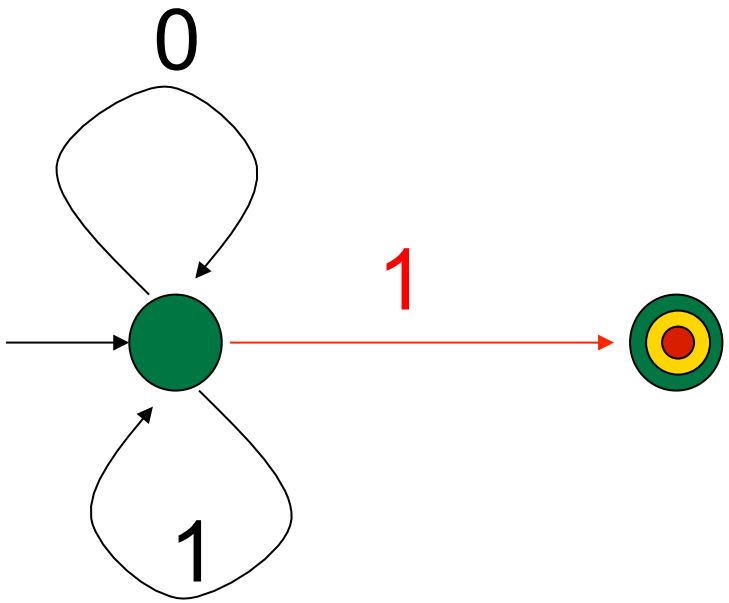
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1
- Guess again



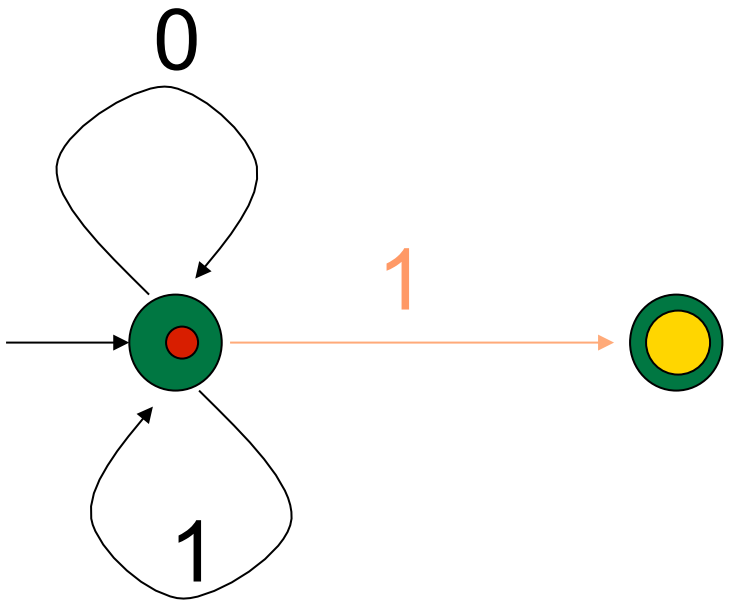
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1
- Guess



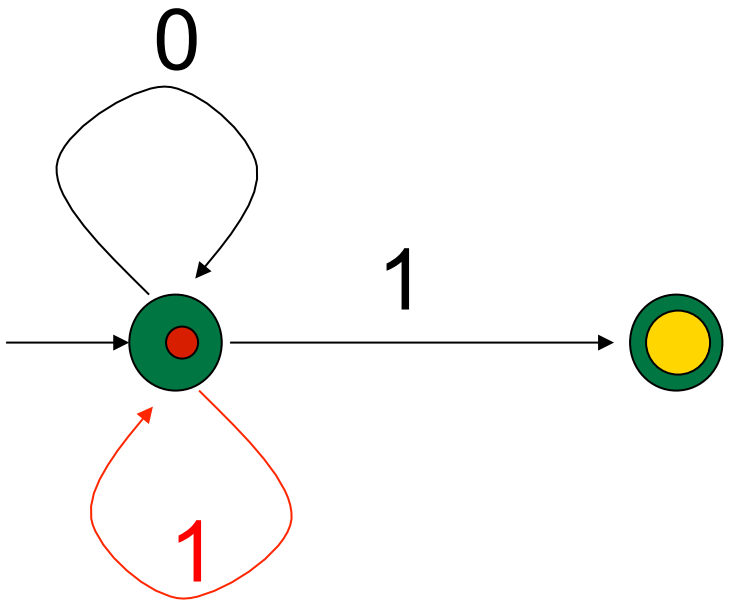
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1
- Backtrack



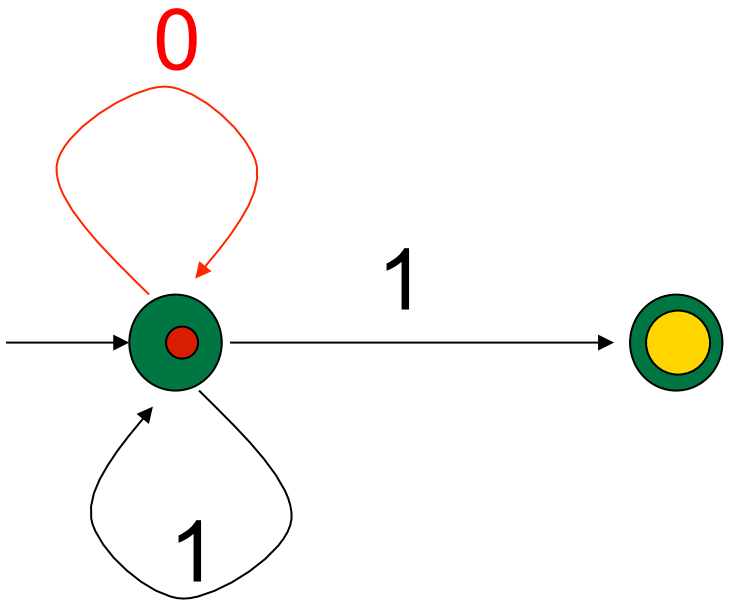
Example NFSA

- Regular expression: (0 v 1)* 1
- Accepts string ~~0 1 1~~ 0 1
- Guess again



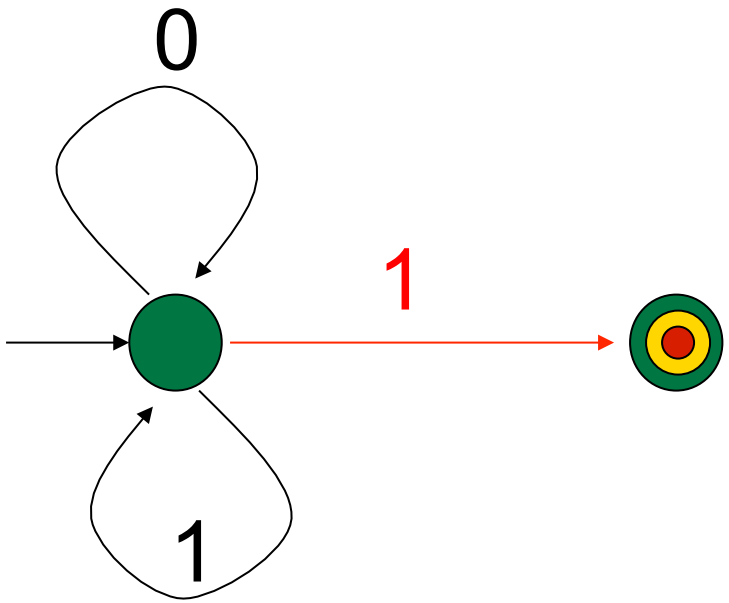
Example NFSA

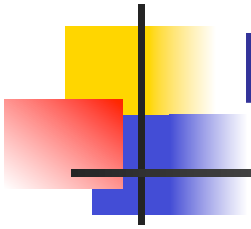
- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1



Example NFSA

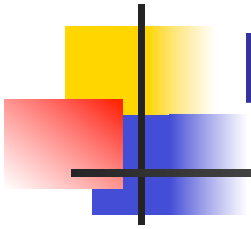
- Regular expression: (0 v 1)* 1
- Accepts string 0 1 1 0 1
- Guess (Hurray!!)





Rule Based Execution

- Search
- When stuck backtrack to last point with choices remaining
- Executing the NFSA in last example was example of rule based execution
- FSA's are rule-based programs; transitions between states (labeled edges) are rules; set of all FSA's is programming language



Rule Based Execution

- Search
- When stuck backtrack to last point with choices remaining

- FSA's are rule-based programs; transitions between states (labeled edges) are rules; set of all FSA's is programming language