Compilers

Finite Automata

Alex Aiken

Finite Automata

- Regular expressions = specification
- Finite automata = implementation

- A finite automaton consists of
  - An input alphabet Σ
  - A set of states S
  - A start state n
  - A set of accepting states $F \subseteq S$
  - A set of transitions state $\rightarrow^{\text{input}}$ state

Finite Automata

- Transition

$$s_1 \rightarrow^{a} s_2$$

- Is read

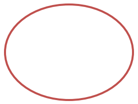
In state s_1 on input a go to state s_2

- If end of input and in accepting state => accept

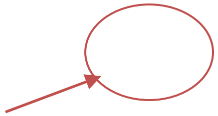
- Otherwise => reject

Finite Automata

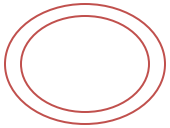
- A state



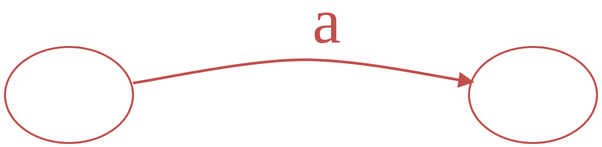
- The start state



- An accepting state



- A transition



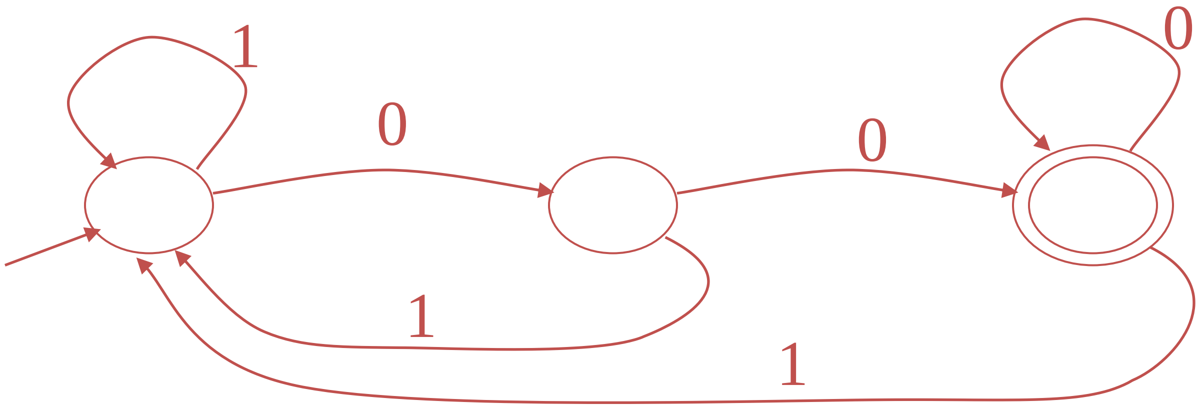
Finite Automata

- A finite automaton that accepts only "1"

Finite Automata

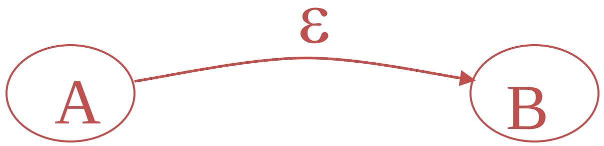
- A finite automaton accepting any number of 1's followed by a single 0
- Alphabet: {0,1}

Select the regular language that denotes the same language as this finite automaton



- $(0 + 1)^*$
- $(1^* + 0)(1 + 0)$
- $1^* + (01)^* + (001)^* + (000^*1)^*$
- $(0 + 1)^*00$

- Another kind of transition: ε-moves



Finite Automata

- Deterministic Finite Automata (DFA)
  - One transition per input per state
  - No ε-moves
- Nondeterministic Finite Automata (NFA)
  - Can have multiple transitions for one input in a given state
  - Can have ε-moves

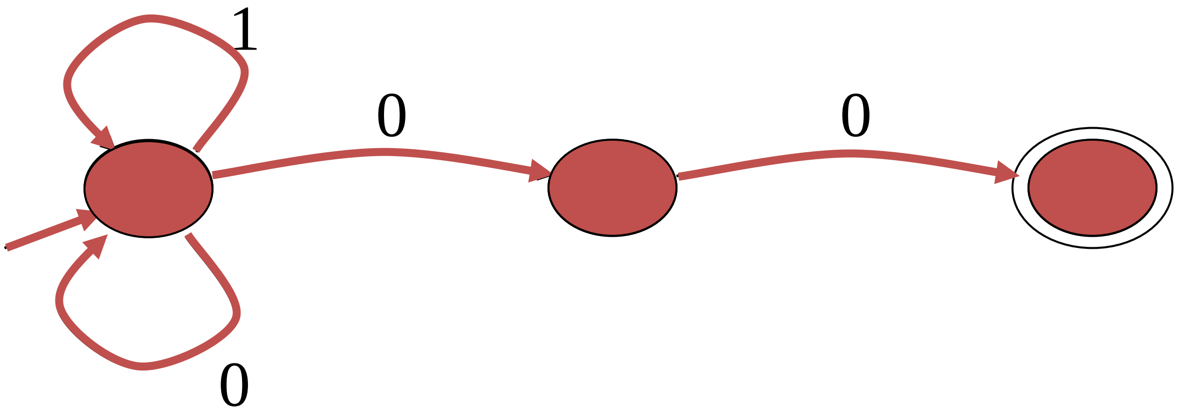
Finite Automata

- A DFA takes only one path through the state graph

- An NFA can choose

Finite Automata

- An NFA can get into multiple states



- Input: 1 0 0
- States:

Finite Automata

- NFAs and DFAs recognize the same set of languages
  - regular languages
- DFAs are faster to execute
  - There are no choices to consider
- NFAs are, in general, smaller