Formal languages and automata

Features of regular languages

https://t.me/fla_uog

mh.olyaee@gmail.com

Brief review

- Symbols, Alphabet, Strings, Σ^*, Languages, 2^{Σ^*}
- Deterministic Finite State Automata
  - States, Labels, Start State, Final States, Transitions
  - Extended State Transition Function
  - DFAs accept regular languages

Regular languages

- Since regular languages are sets, we can combine them with the usual set operations
  - Union
  - Intersection
  - Difference

Theorem

If L_1 and L_2 are regular languages, so are $L_1 \cup L_2$, $L_1 \cap L_2$ and $L_1 - L_2$.

Proof Idea

Construct cross-product DFAs

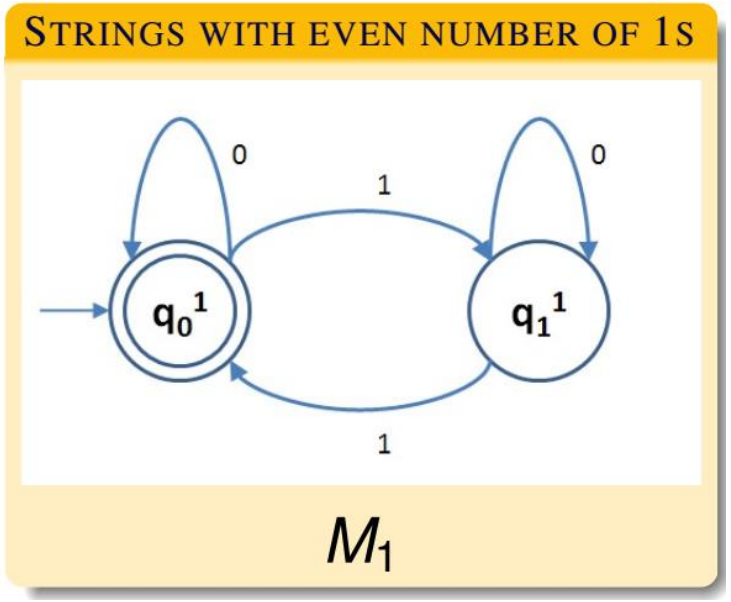
Cross-product DFA

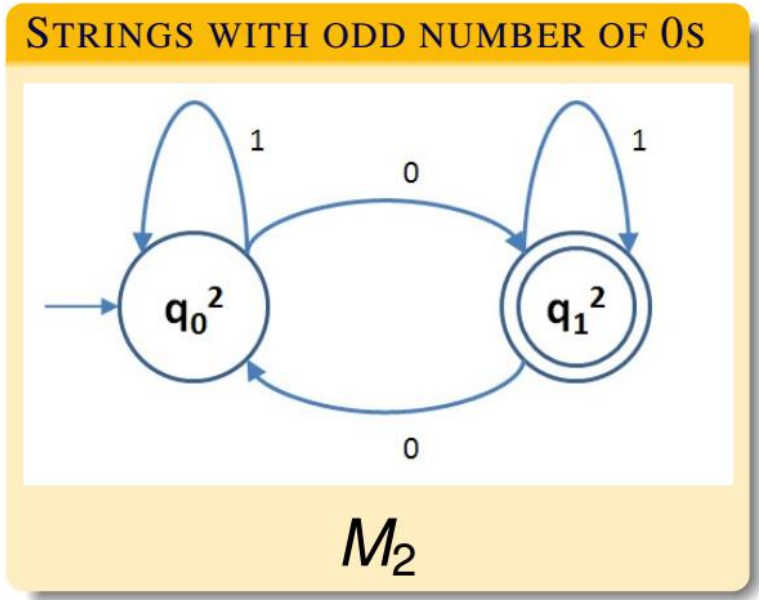
- A single DFA which simulates operation of two DFAs in parallel!
- Let the two DFAs be M_1 and M_2 accepting regular languages L_1 and L_2
  1. $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$
  2. $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$
- We want to construct DFAs $M = (Q, \Sigma, \delta, q_0, F)$ that recognize
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $L_1 - L_2$

Constructing M

- We need to construct $M = (Q, \Sigma, \delta, q_0, F)$
- Q =pairs of states, one from M_1 and one from M_2
  $Q = \{(q_1, q_2) | q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$
  $Q = Q_1 \times Q_2$
- $q_0 = (q_0^1, q_0^2)$
- $\delta((q_i^1, q_j^2), x) = (\delta_1(q_i^1, x), \delta_2(q_j^2, x))$
- Union: $F = \{(q_1, q_2) | q_1 \in F_1 \text{ or } q_2 \in F_2\}$
- Intersection: $F = \{(q_1, q_2) | q_1 \in F_1 \text{ and } q_2 \in F_2\}$
- Difference: $F = \{(q_1, q_2) | q_1 \in F_1 \text{ and } q_2 \notin F_2\}$

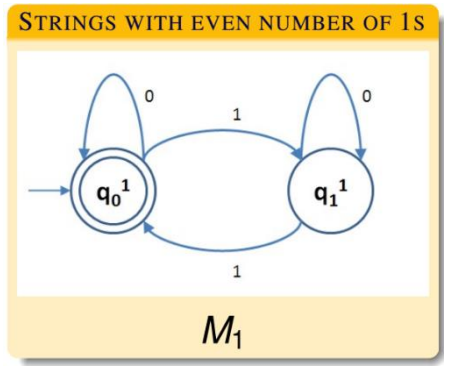
Example

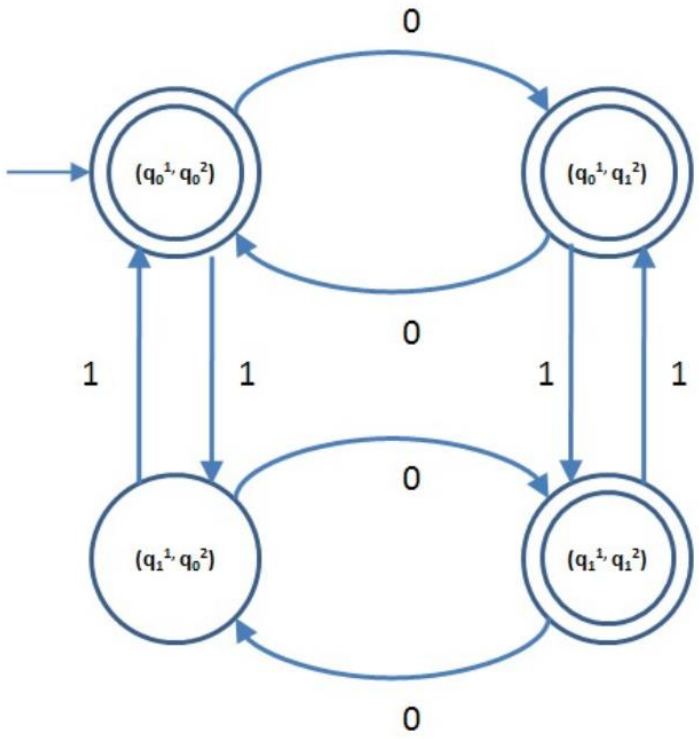

STRINGS WITH EVEN NUMBER OF 1S
0
1
0
q_0^1
q_1^1
1
M_1


STRINGS WITH ODD NUMBER OF 0S
1
0
1
q_0^2
q_1^2
0
M_2

DFA for Union

- DFA for $L_1 \cup L_2$ accepts when either M_1 or M_2 accepts.

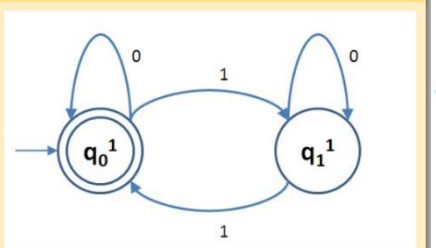




DFA for Intersection

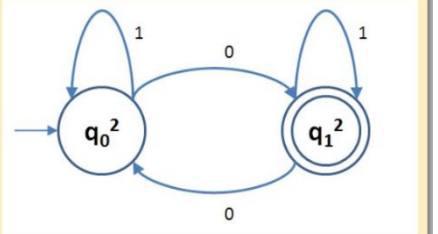
- DFA for $L_1 \cap L_2$ accepts when both M_1 and M_2 accept.

STRINGS WITH EVEN NUMBER OF 1S

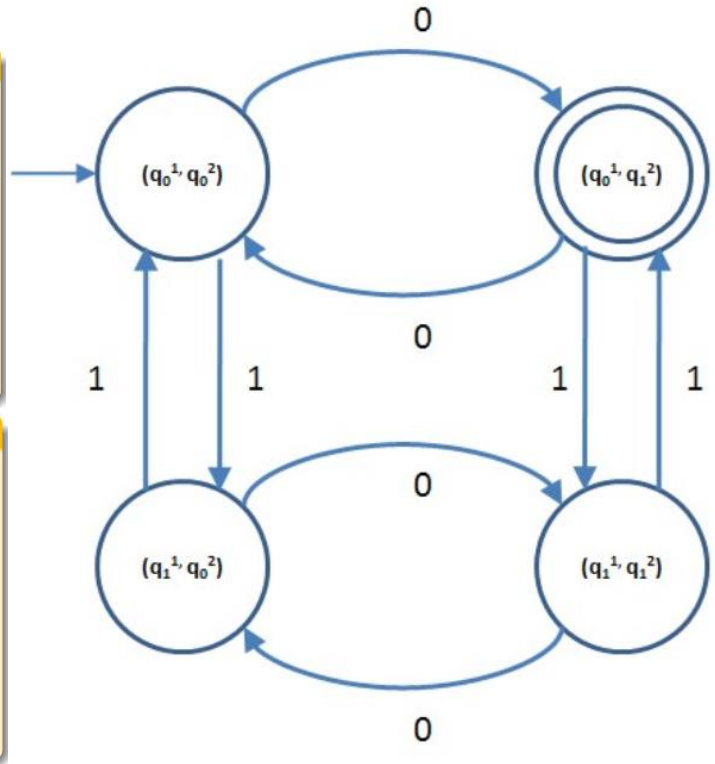


M_1

STRINGS WITH ODD NUMBER OF 0S



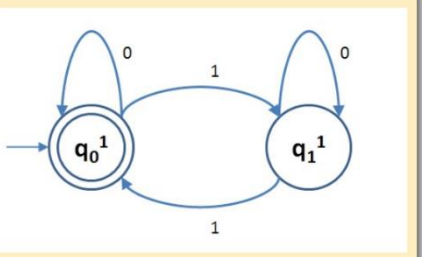
M_2



DFA for Difference

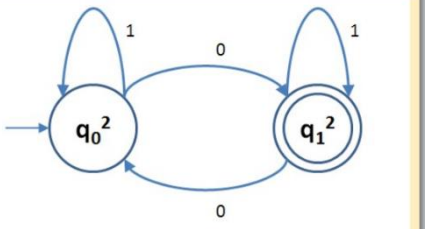
- DFA for $L_1 - L_2$ accepts when M_1 accepts and M_2 rejects.

STRINGS WITH EVEN NUMBER OF 1S

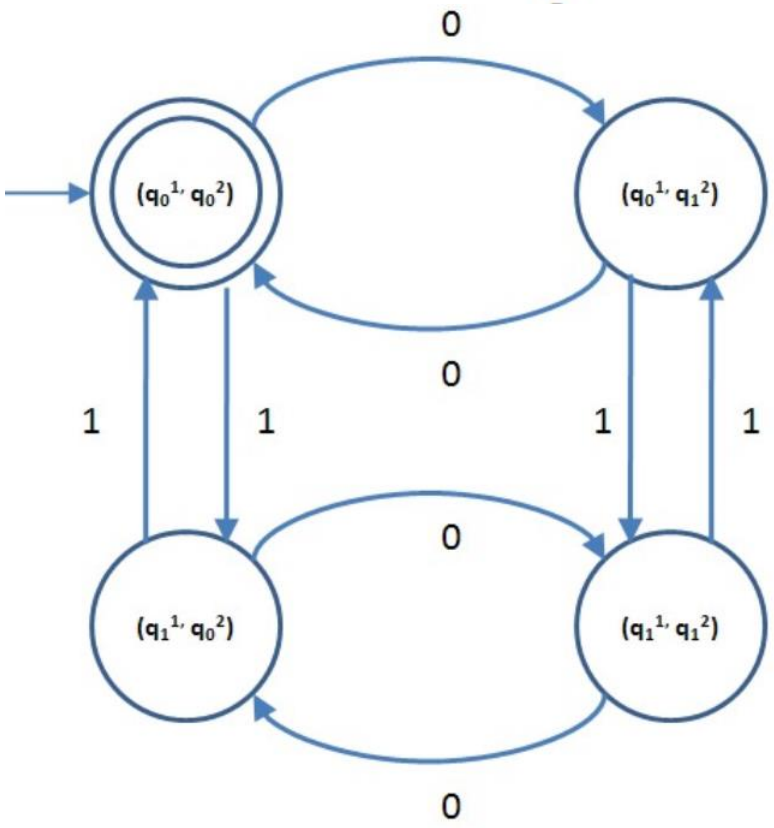


M_1

STRINGS WITH ODD NUMBER OF 0S



M_2

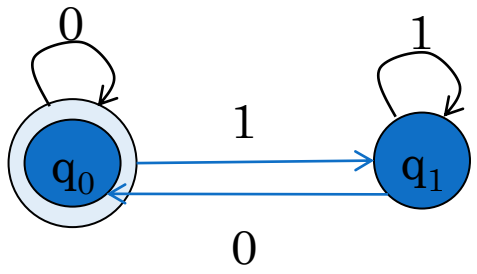


Example

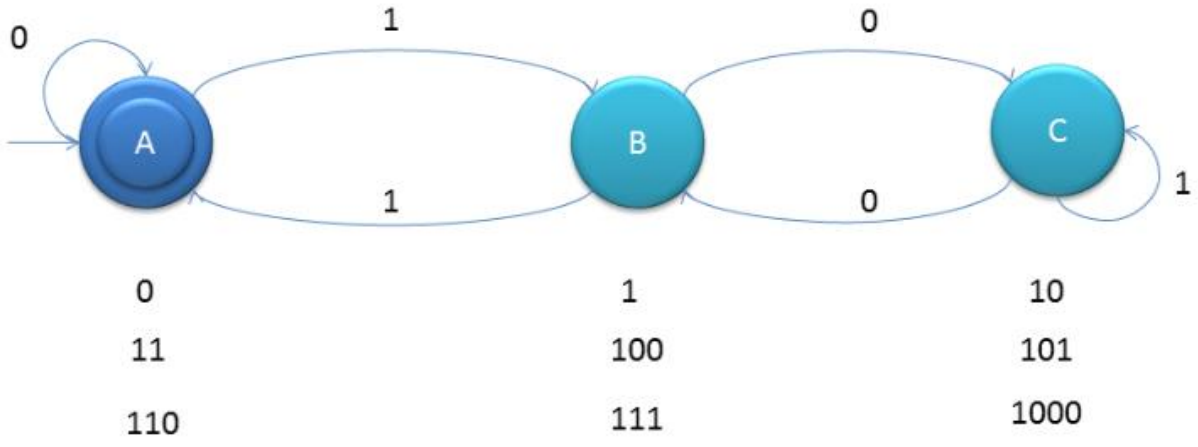
- DFA for binary numbers divisible by 3
- DFA for binary numbers divisible by 2

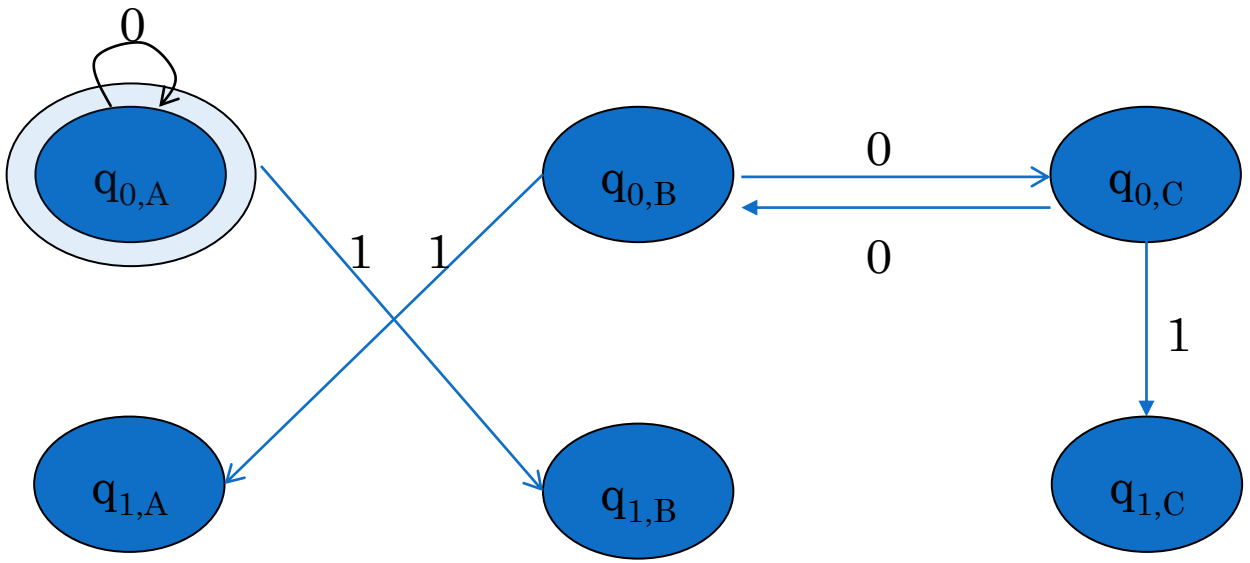
Divided by 2

- Consider the below set
- {0, 01, 11, 110, …}



Divided by 3




0
$q_{0,A}$
1
1
$q_{0,B}$
0
0
$q_{0,C}$
1
$q_{1,A}$
$q_{1,B}$
$q_{1,C}$

Other regular operators

- Reverse: $L^R = \{\omega = a_1 \dots a_n | \omega^R = a_n \dots a_1 \in L\}$
- Concatenation: $L_1 \cdot L_2 = \{\omega\nu | \omega \in L_1 \text{ and } \nu \in L_2\}$
- Star Closure: $L^* = \{\omega_1\omega_2 \dots \omega_k | k \geq 0 \text{ and } \omega_i \in L\}$

$L1 = \{a, aa, ba\}, L2 = \{b, ba\} \longrightarrow L1.L2 = \{ab, aba, aab, aaba, bab, baba\}$

$L = \{b, ba\}, L^* = \lambda, b, ba, bb, bba, baba, bab, bbb, bbba, bbaba, \ldots = L^0 \cup L^1 \cup \cdots.$

The reverse of a regular language

THEOREM

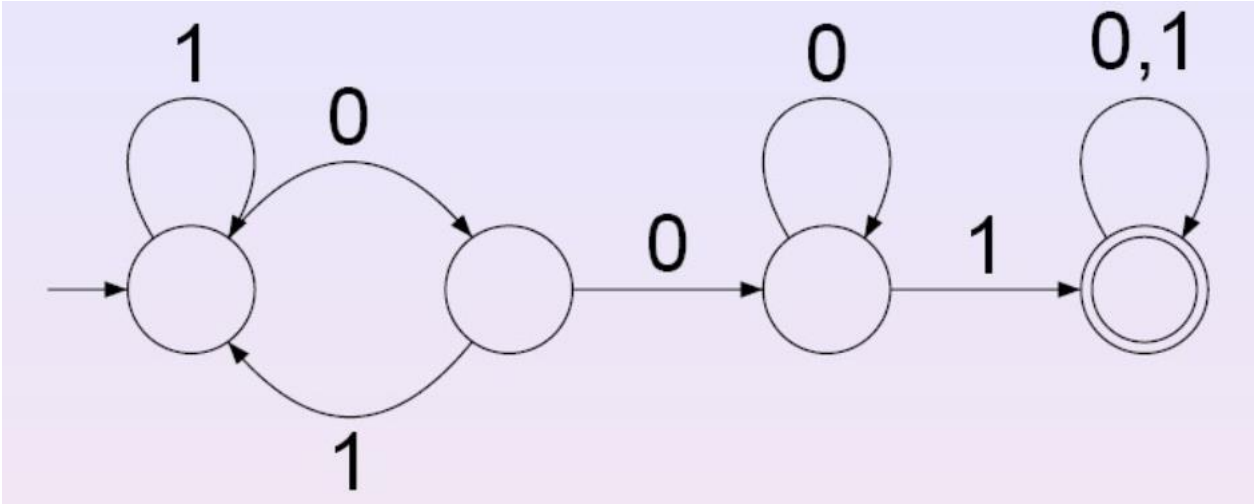
The reverse of a regular language is also a regular language.

- If a language can be recognized by a DFA that reads strings from right to left, then there is an "normal" DFA (one that reads from left to right) that accepts the same language.

Reversing a DFA

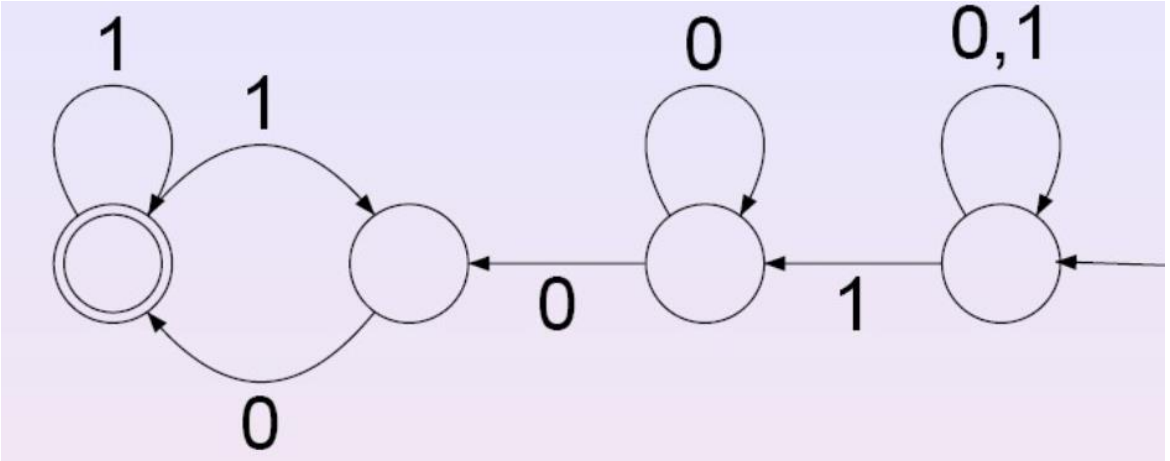
- Assume L is a regular language. Let M be a DFA that recognizes L
- We will build a machine M^R that accepts L^R
- If M accepts ω, then ω describes a directed path, in M, from the start state to a final state.
- First attempt: Try to define M^R as M as follows
  - Reverse all transitions
  - Turn the start state to a final state
  - Turn the final states to start states!
- But, as such, M^R is not always a DFA.
  - It could have many start states.
  - Some states may have too many outgoing transitions or none at all!

Example



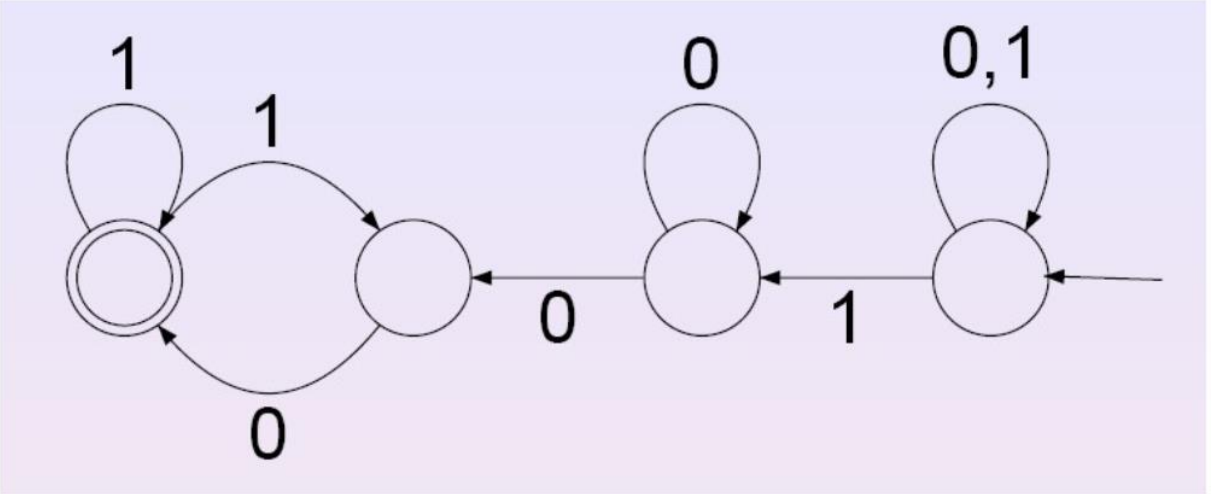
- What language does this DFA recognize?

Example



- What happens with input 100?
  - There are multiple transitions from a state labeled with the same symbol.
  - State transitions are not deterministic any more: the next state is not uniquely determined by the current state and the current input. $\rightarrow$ Nondeterminism

Example



- We will say that this machine accepts a string if there is some path that reaches an accept state from a start state.