Programming Languages and Software Verification

02 Regular languages and finite-state automata

DFA: informal view

A **Deterministic Finite-State Automaton** (DFA) is a machine that takes in input a sequence of symbols, and it answers with yes (or no) if the sequence is accepted (or rejected)

- State: represents the basic memory support for the automaton, such that reading a symbol makes the automaton move from state to state
- Finite: means that while analyzing the sequence of symbols the automaton can only remember a finite set of facts about the sequence
- Deterministic: means that given a state and an input symbol, the automaton has exactly one way to proceed

DFA: formal definition

A Deterministic Finite-State Automaton (DFA) is a tuple

$$A = (Q, \Sigma, \delta, q_0, F)$$

where

- Q is a finite set of states
- Σ is an alphabet of input symbols
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final (or accepting) states

DFA: how does it work?

Reading the input string $a_1 a_2 \ldots a_n$ induces the visit of a path in the automaton:

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} q_n$$

such that:

- q_0 is the initial state
- $q_{i+1} = \delta(q_i, a_{i+1})$
- if $q_n \in F$ the string $a_1 a_2 \cdots a_n$ is accepted, otherwise it is rejected

DFA: example

Create a DFA that recognizes the language

$$L = \{x \in \{0,1\}^* \mid 01 \text{ is a substring of } x\}$$

We use 3 states:

- q_0 "I have seen no symbols, or all the symbols I have seen so far were 1"
- q_1 "the last symbol I saw was 0"
- q_2 "I have seen a 01 substring"

The alphabet is $\Sigma = \{0,1\}$
The initial state is q_0
The set of final states is $F = \{q_2\}$

DFA: example

Create a DFA that recognizes the language

$$L = \{x \in \{0,1\}^* \mid 01 \text{ is a substring of } x\}$$

We specify the transitions:

- from q_0, read 0, go to q_1
- from q_0, read 1, stay in q_0
- from q_1, read 0, stay in q_1
- from q_1, read 1, go to q_2
- from q_2, read 0 or 1, stay in q_2

DFA: example

Create a DFA that recognizes the language

$$L = \{x \in \{0,1\}^* \mid 01 \text{ is a substring of } x\}$$

Formally, the automaton we have defined is the following:

$$A = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$$

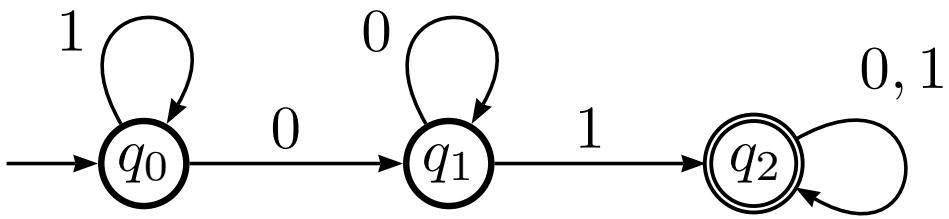
where:

$$\begin{aligned}
\delta(q_0, 0) &= q_1 \\
\delta(q_0, 1) &= q_0 \\
\delta(q_1, 0) &= q_1 \\
\delta(q_1, 1) &= q_2 \\
\delta(q_2, 0) &= q_2 \\
\delta(q_2, 1) &= q_2
\end{aligned}$$

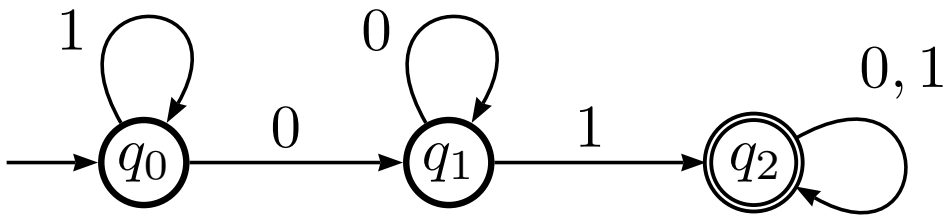
Transition diagrams

A more convenient way of representing (small) DFAs $(Q, \Sigma, \delta, q_0, F)$:

- for each state $q \in Q$ there is a node labeled q
- for each state $q \in Q$ and each symbol $a \in \Sigma$ there is an arc labeled a from the node labeled q to the node labeled $\delta(q, a)$
- the initial node q_0 has an incoming arrow
- final states $q \in F$ are emphasized



Transition tables



Tabular representation of the transition diagram

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$* q_2$	q_2	q_2

Language accepted by a DFA

Extend the transition function to strings

$$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$$

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a) \end{aligned}$$

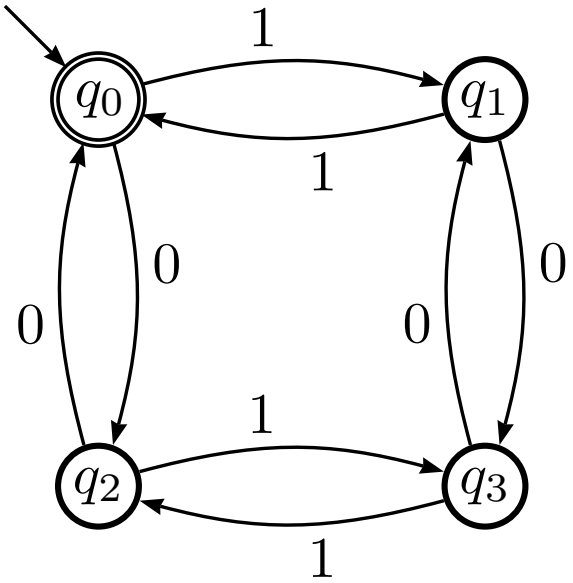
Language accepted by DFA $A = (Q, \Sigma, \delta, q_0, F)$

$$L(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

Example

$$L = \{w \in \{0,1\}^* \mid w \text{ has an even number of 0s and 1s}\}$$

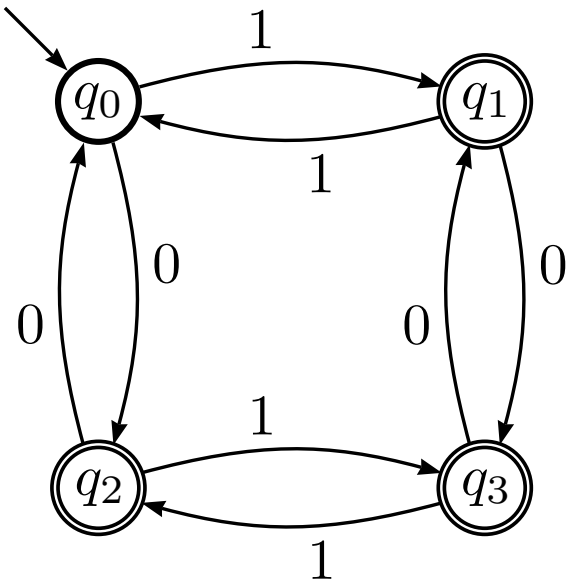
The automaton that accepts L



Example

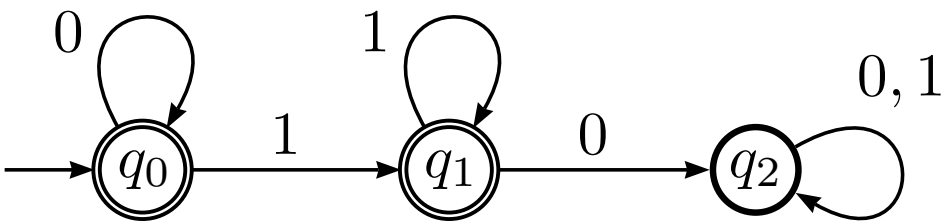
$$L = \{w \in \{0,1\}^* \mid w \text{ has an even number of 0s and 1s}\}$$

The automaton that accepts $\overline{L}$



Example

$$L = \{0^n 1^m \mid n \geq 0, m \geq 0\}$$



Example

$$L = \{(01)^n \mid n \geq 0\}$$