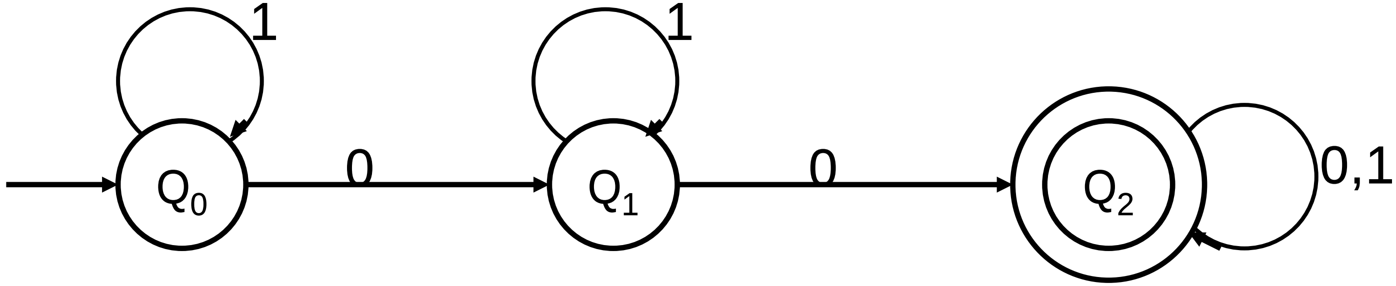
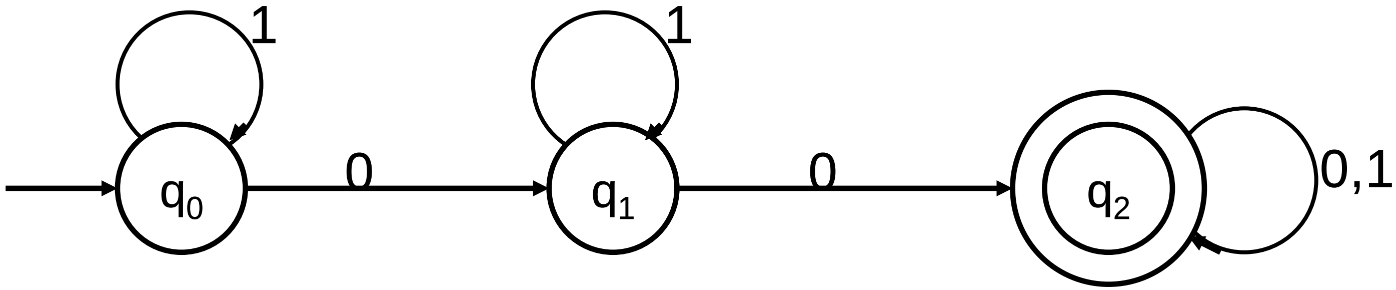
Deterministic Finite Automata (DFA)

- DFAs are easiest to present pictorially:



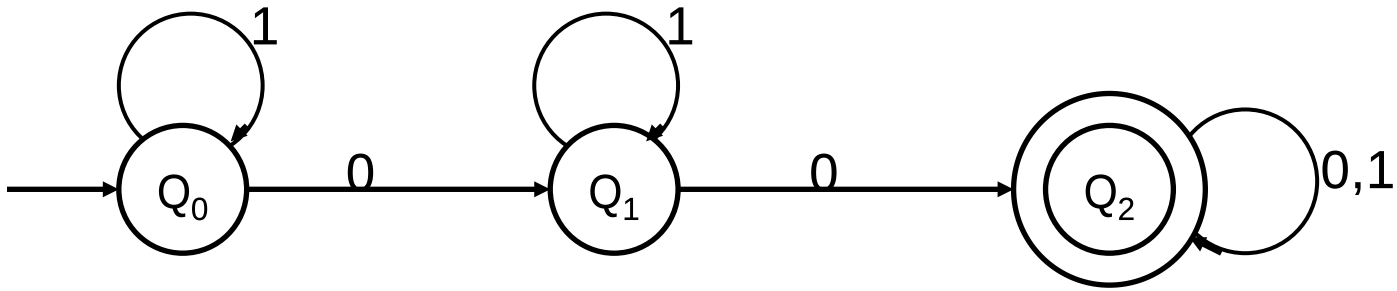
They are directed graphs whose nodes are *states* and whose arcs are labeled by one or more symbols from some alphabet Σ.

Here Σ is {0,1}.

- One state is *initial* (denoted by a short incoming arrow), and several are *final/accepting* (denoted by a double circle). For every symbol $a \in \Sigma$ there is an arc labeled *a* emanating from every state.
- 

- Automata are string processing devices. The arc from q_1 to q_2 labeled 0 shows that when the automaton is in the state q_1 and receives the input symbol 0, its next state will be q_2.

- Every path in the graph spells out a string over *S*. Moreover, for every string $w \in \Sigma^*$ there is a unique path in the graph labelled *w*. (Every string can be processed.) The set of all strings whose corresponding paths end in a final state is the *language of the automaton*.



- In our example, the language of the automaton consists of strings over {*0,1*} containing at least two occurrences of *0*.

- Modify the automaton so that its language consists of strings containing *exactly two* occurrences of 0.
-

Formal Definition

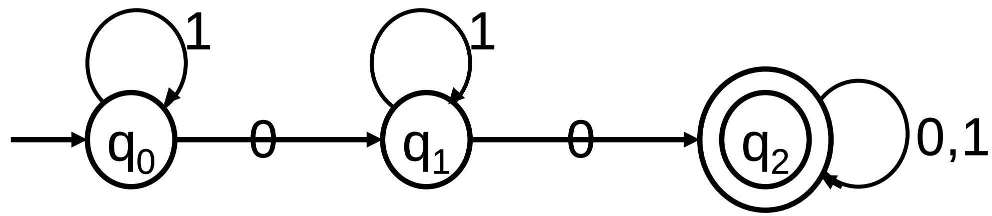
- **A D**FA is a quintuple $\mathbf{A}=(\mathbf{Q}, \Sigma, \mathbf{s}, \mathbf{F}, \delta)$, where
  - **Q** is a set of *states*
  - Σ is the alphabet of *input symbols*
  - **s** is an element of **Q** --- the *initial state*
  - **F** is a subset of **Q** ---the set of *final states*
  - δ: $\mathbf{Q} \times \Sigma \longrightarrow \mathbf{Q}$ is the *transition function*

Example

- In our example,
- $\mathbf{Q}=\{q_0,q_1,q_2\}$,
  $\mathbf{\Sigma}=\{0,1\}$,
  $\mathbf{s}=q_0$,
  $\mathbf{F}=\{q_2\}$,
- and

δ is given by 6 equalities

- $\delta(q_0,0)=q_1$,
- $\delta(q_0,1)=q_0$,
- $\delta(q_2,1)=q_2$
- …



Transition Table

- All the information presenting a DFA can be given by a single thing -- its *transition table*:

	0	1
Q_0	Q_1	Q_0
$\rightarrow Q_1$	Q_2	Q_1
$*Q_2$	Q_2	Q_2

- The initial and final states are denoted by $\rightarrow$ and * respectively.

Extension of δ to Strings

- Given a state **q** and a string **w**, there is a unique path labeled **w** that starts at **q** (why?). The endpoint of that path is denoted $\underline{\delta}$**(q,w)**

- Formally, the function $\underline{\delta} : Q \times \Sigma^* \rightarrow Q$
- is defined recursively:

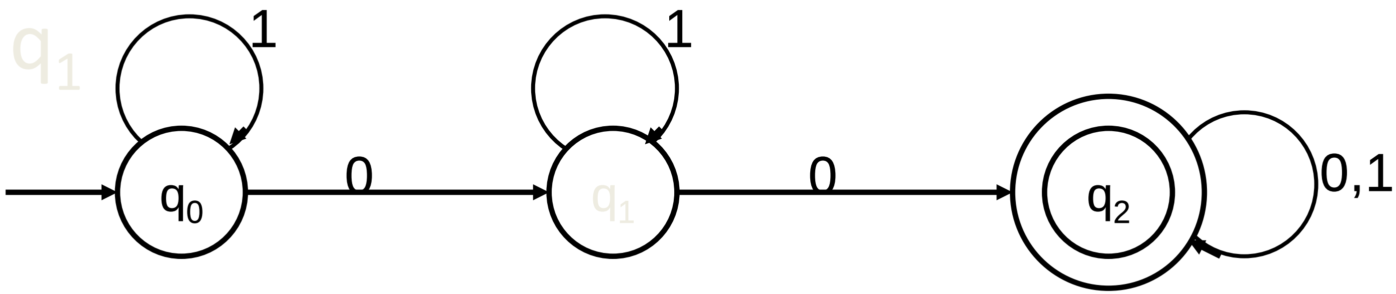
  – $\underline{\delta}(q,\varepsilon)=q$
  – $\underline{\delta}(q,ua)= \delta(\underline{\delta}(q,u),a)$

- Note that $\underline{\delta}(q,a)= \delta(q,a)$ for every $a\in\Sigma$;
- so $\underline{\delta}$ does extend δ.

Example trace

- Diagrams (when available) make it very easy to compute $\underline{\delta}(q,w)$ --- just trace the path labeled w starting at q.

- E.g. trace 101 on the diagram below starting at q_1



- Implementation and precise arguments need the formal definition.
-
- $\underline{\delta}(q_1,101)=\delta(\underline{\delta}(q_1,10),1)$
- $=\delta(\delta(\underline{\delta}(q_1,1),0),1)$
- $=\delta(\delta(\delta(q_1,1),0),1)$
- $=\delta(\delta(q_1,0),1)$
- $=\delta(q_2,1)$
- $=q_2$

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

Language of accepted strings

A DFA $=(\mathbf{Q},\Sigma,\mathbf{s},\mathbf{F},\delta)$, *accepts* a string $\mathbf{w}$ iff $\underline{\delta}(\mathbf{s},w) \in \mathbf{F}$

The language of the automaton A is

$L(A)=\{w \mid A \text{ accepts } w\}$.

More formally

$L(A)=\{w \mid \underline{\delta}(Start(A),w) \in Final(A)\}$

Example:

Find a DFA whose language is the set of all strings over $\{a,b,c\}$ that contain aaa as a substring.

DFA's as Programs

```
data DFA q s = DFA { states :: [q],
                     symbols :: [s],
                     delta :: q -> s -> q,
                     start :: q,
                     final :: [q]}
```

Transition function

```
trans :: (q -> s -> q) -> q -> [s] -> q
trans d q [] = q
trans d q (s:ss) = trans d (d q s) ss

accept :: (Eq q) => DFA q s -> [s] -> Bool
accept
  m@(DFA{delta = d,start = q0,final = f}) w
  = elem (trans d q0 w) f
```

An Example

```
ma = DFA { states = [0,1,2],
           symbols = [0,1],
           delta = \p a ->
                    (2*p+a) `mod` 3,
           start = 0,
           final = [2]
         }
```