# Finite-State Machines (Automata) lecture 13

- a simple form of computation
- used widely
- one way to find patterns
- one way to describe reactive systems

# Reactive Systems


- Wait to receive an input
- Respond with:

  an output (possibly changing to a new state)
  or

  change to new state without output
- Response depends on (finite) history

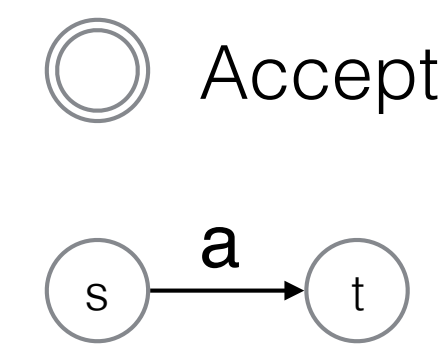
# Finite State Machines

- A conceptual tool for modelling reactive systems.
- Not limited to software systems.
- Used to specify required system behaviour in a precise way.
- Then implement as software/hardware (and perhaps verify behaviour against FSM).

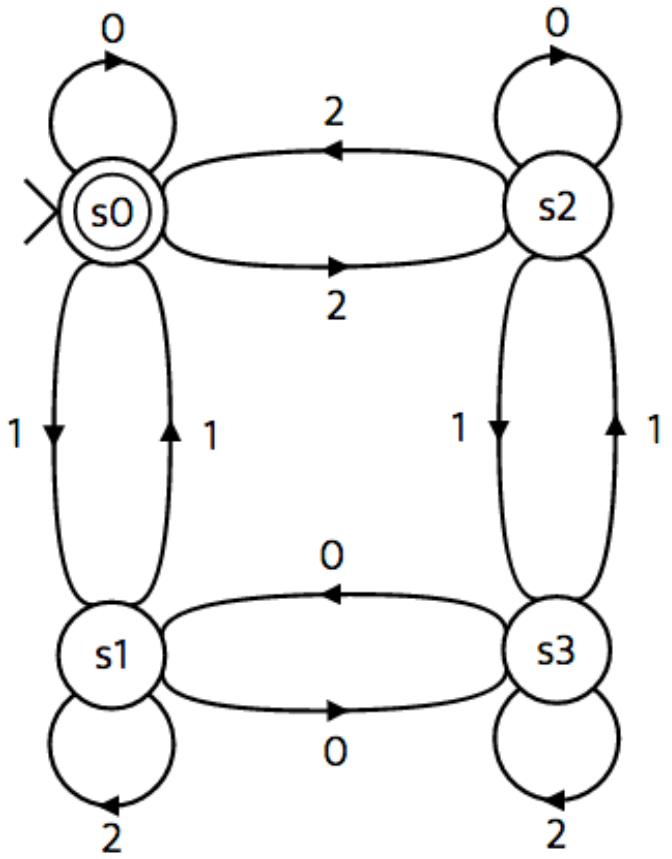
State

Start state

Accepting state




Transition




depending on the application,
**a** is a letter, symbol, token, action, …

This machine has four states,

$Q = \{ s0, s1, s2, s3 \}$

Its alphabet has three symbols

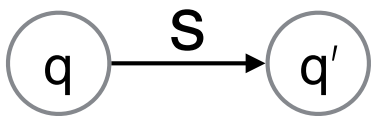
$\Sigma = \{ 0, 1, 2 \}$

For each

(state, symbol) pair, **(q,s)**

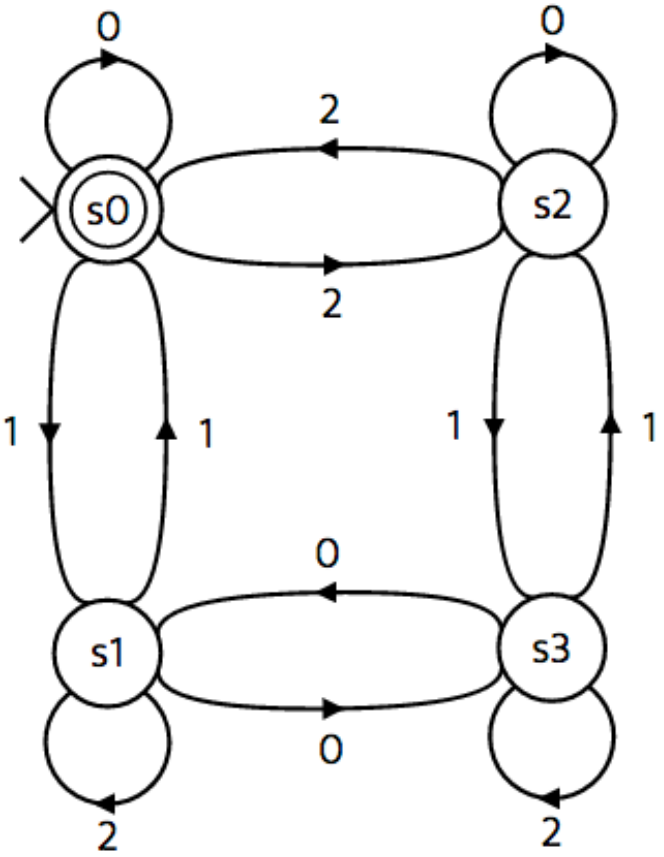
there is **exactly one** transition

from **q** with label **s**




This means that

**q′** is **determined** by **(q,s)**




This is a

**deterministic**

**finite-state**

**automaton**

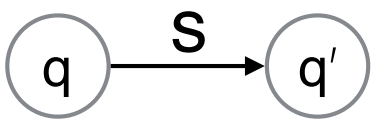
# DFA

For each

(state, symbol) pair, **(q,s)**

there is **exactly one** transition

from **q** with label **s**

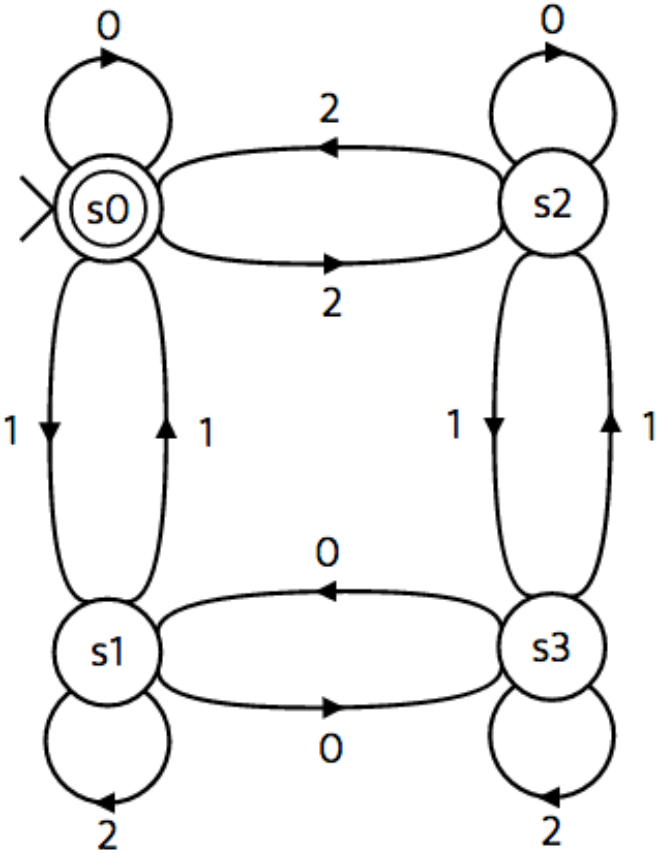



**q′** is **determined** by **(q,s)**

Every DFA has a

**next-state function**

$q' = F(q,s)$



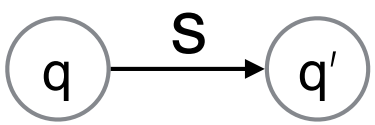

This is a

**deterministic finite-state automaton, DFA**

Every DFA has a

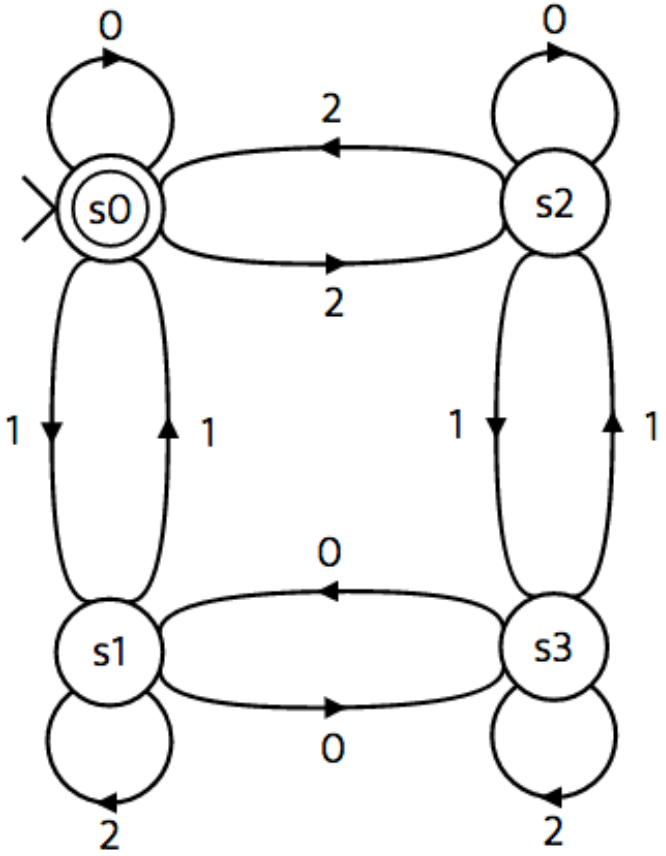
**next-state function**

$q' = F(q,s)$




Since we have
finitely many symbols and states
This can be given by a table

| | 0 | 1 | 2 |
|---|---|---|---|
| **s0** | s0 | | |
| **s1** | | | s2 |
| **s2** | | s3 | |
| **s3** | s1 | | |

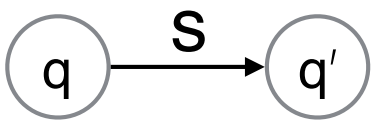



This is a
**deterministic finite-state automaton, DFA**

Every DFA has a

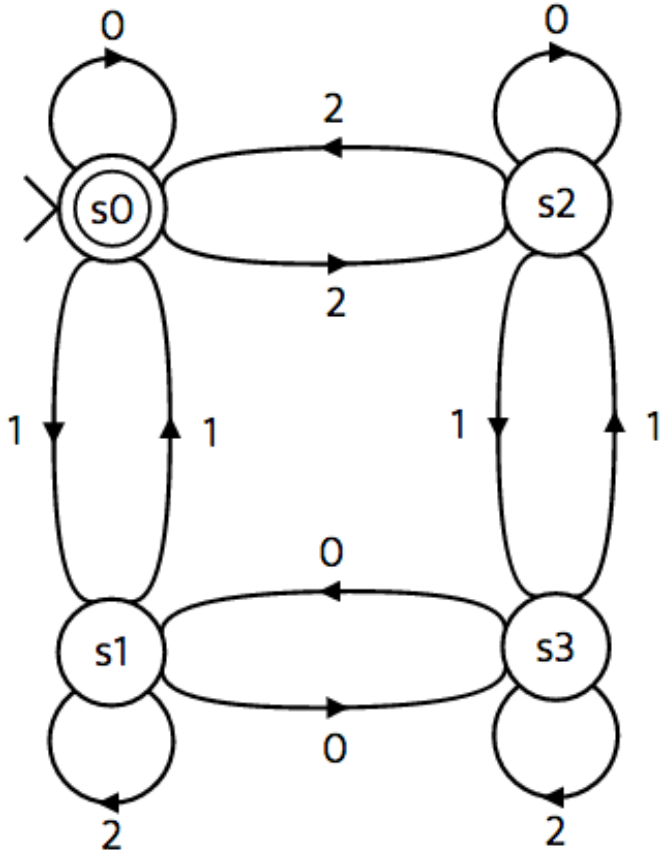
**next-state function**

$q' = F(q,s)$




Since we have
finitely many symbols and states
This can be given by a
**next-state table**

| q\s | 0 | 1 | 2 |
|---|---|---|---|
| **s0** | s0 | s1 | s2 |
| **s1** | s3 | s0 | s1 |
| **s2** | s2 | s3 | s0 |
| **s3** | s1 | s2 | s3 |

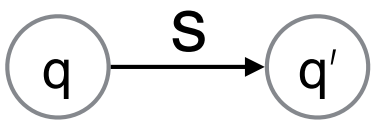



This is a
**deterministic finite-state automaton, DFA**

Every DFA has a

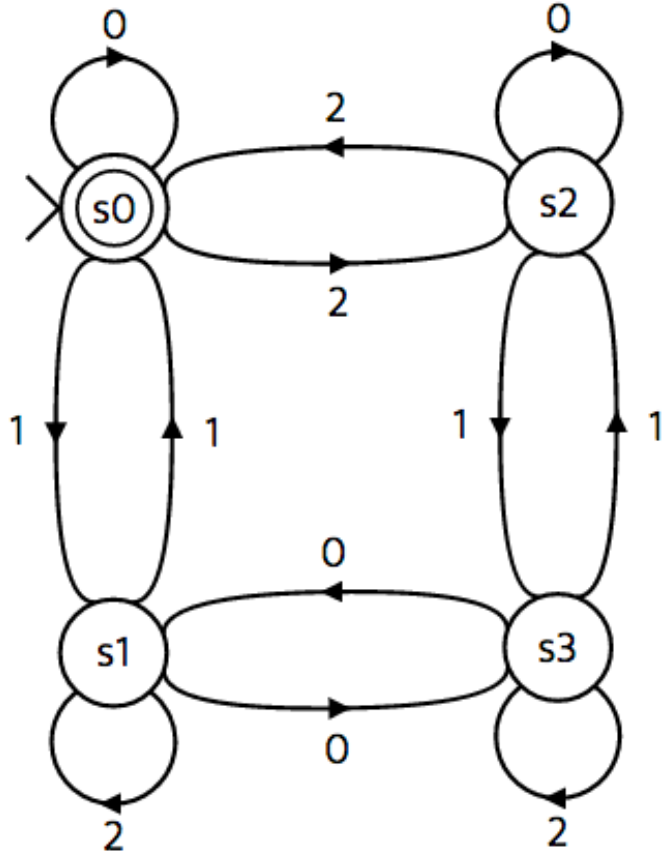
**next-state function**

$q' = F(q,s)$




We can also represent the transitions by a relation. Which symbol(s) take us from state **q** to state **r** ?

| q\r | s0 | s1 | s2 | s3 |
|---|---|---|---|---|
| s0 | 0 | 1 | 2 | |
| s1 | | | | |
| s2 | | | | |
| s3 | | 0 | 1 | |

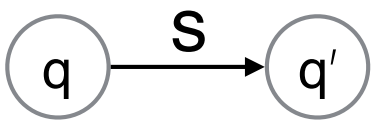



This is a **deterministic finite-state automaton, DFA**

Every DFA has a

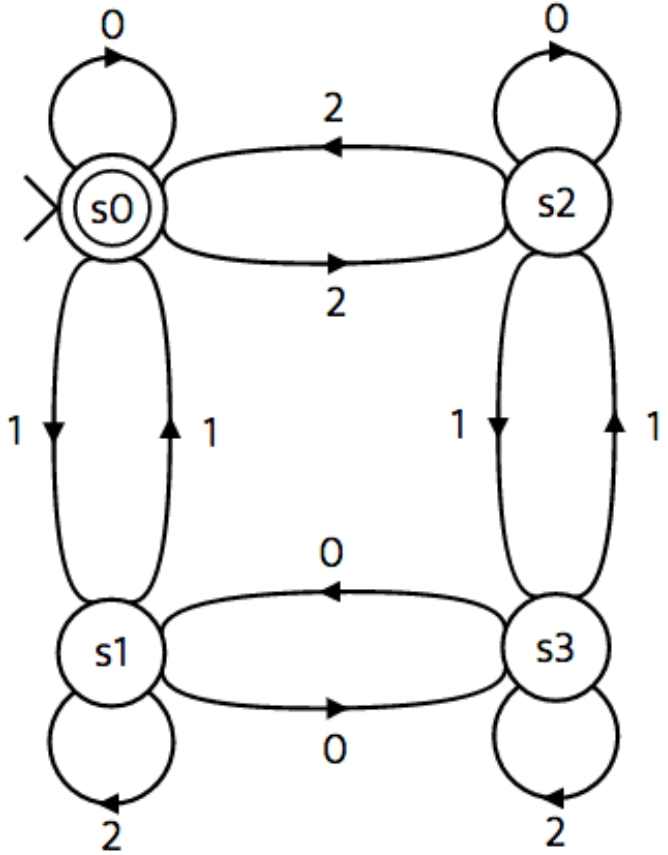
**next-state function**

$q' = F(q,s)$




We can also represent the transitions by a relation. Which symbol(s) take us from state **q** to state **r** ?

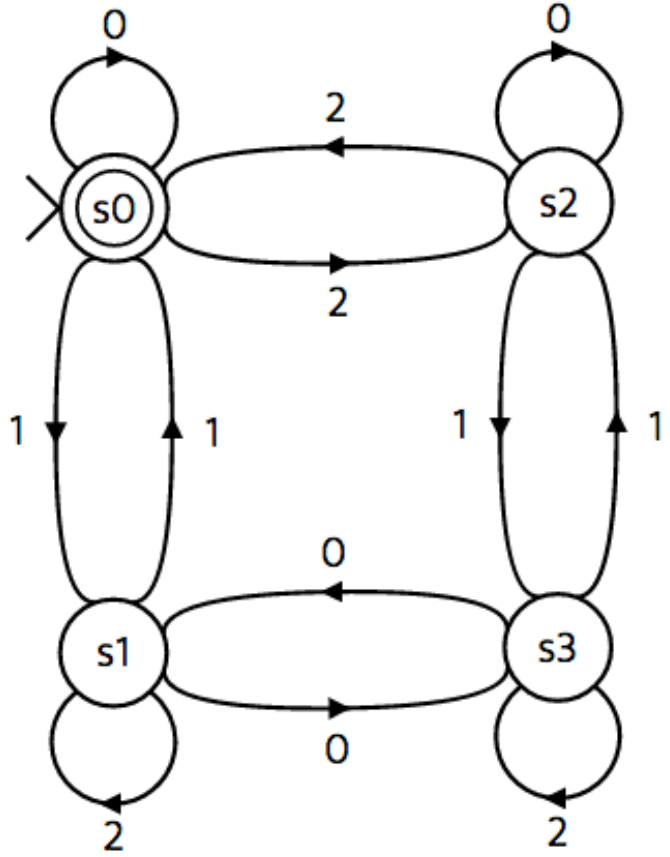
| q\r | s0 | s1 | s2 | s3 |
|---|---|---|---|---|
| **s0** | 0 | 1 | 2 | |
| **s1** | 1 | 2 | | 0 |
| **s2** | 2 | | 0 | 1 |
| **s3** | | 0 | 1 | 2 |




This is a **deterministic finite-state automaton, DFA**

For every DFA, any input string determines a path or **trace**

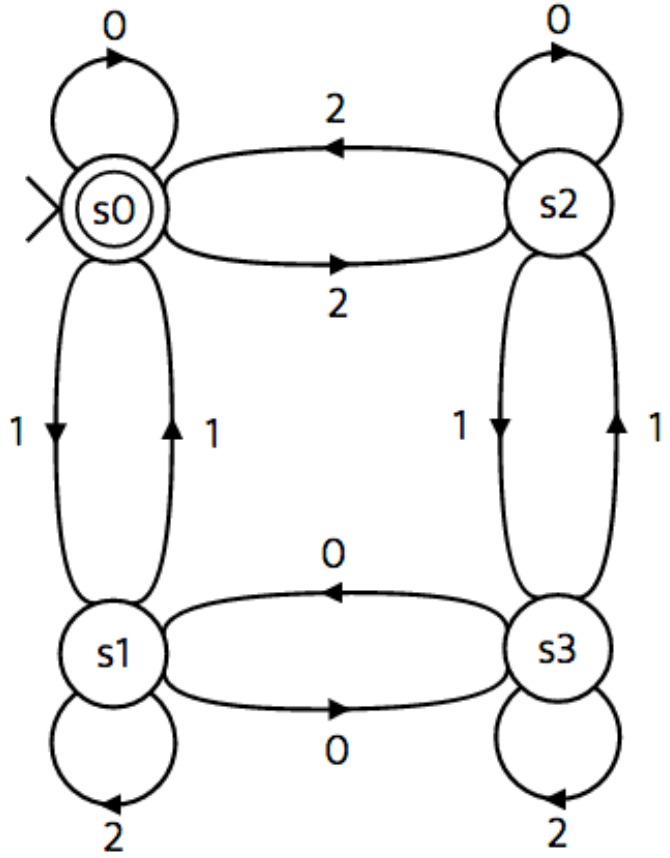
| input | path |
|---|---|
| 000 | s0 s0 s0 s0 |
| 021 | s0 s0 s2 s3 |
| 120 | s0 s1 s1 s3 |
| 11 | |
| 12 | |
| 21 | |
| 22 | |
| 111 | |
| 110 | |
| 212 | |




This is a **deterministic finite-state automaton, DFA**

For every DFA, any input string determines a path

| input | path |
|---|---|
| 000 | s0 s0 s0 s0 |
| 021 | s0 s0 s2 s3 |
| 120 | s0 s1 s1 s3 |
| 11 | s0 s1 s0 |
| 12 | s0 s1 s1 |
| 21 | s0 s2 s3 |
| 22 | s0 s2 s0 |
| 111 | s0 s1 s0 s1 |
| 110 | s0 s1 s0 s0 |
| 212 | s0 s2 s3 s3 |

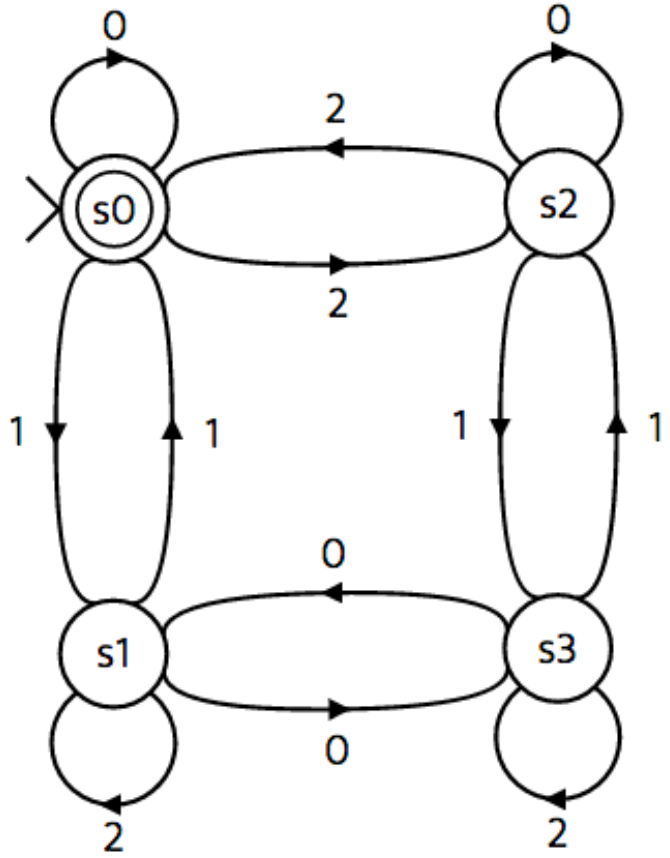



This is a **deterministic finite-state automaton, DFA**

Which paths

**end in an accepting state?**

| input | path |
|---|---|
| **000** | **s0 s0 s0 s0** |
| 021 | s0 s0 s2 s3 |
| 120 | s0 s1 s1 s3 |
| **11** | **s0 s1 s0** |
| 12 | s0 s1 s1 |
| 21 | s0 s2 s3 |
| **22** | **s0 s2 s0** |
| 111 | s0 s1 s0 s1 |
| **110** | **s0 s1 s0 s0** |
| 212 | s0 s2 s3 s3 |

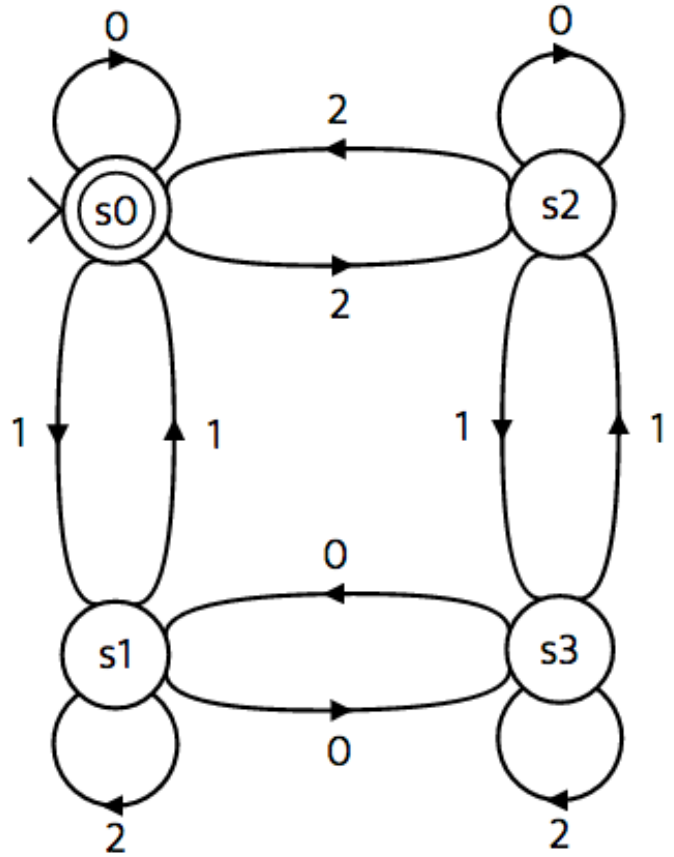



This is a **deterministic finite-state automaton, DFA**

Which paths

**end in an accepting state?**

| | input | path |
|---|---|---|
| **0** | **000** | **s0 s0 s0 s0** |
| 7 | 021 | s0 s0 s2 s3 |
| 15 | 120 | s0 s1 s1 s3 |
| **4** | **11** | **s0 s1 s0** |
| 5 | 12 | s0 s1 s1 |
| 7 | 21 | s0 s2 s3 |
| **8** | **22** | **s0 s2 s0** |
| 13 | 111 | s0 s1 s0 s1 |
| **12** | **110** | **s0 s1 s0 s0** |
| 23 | 212 | s0 s2 s3 s3 |

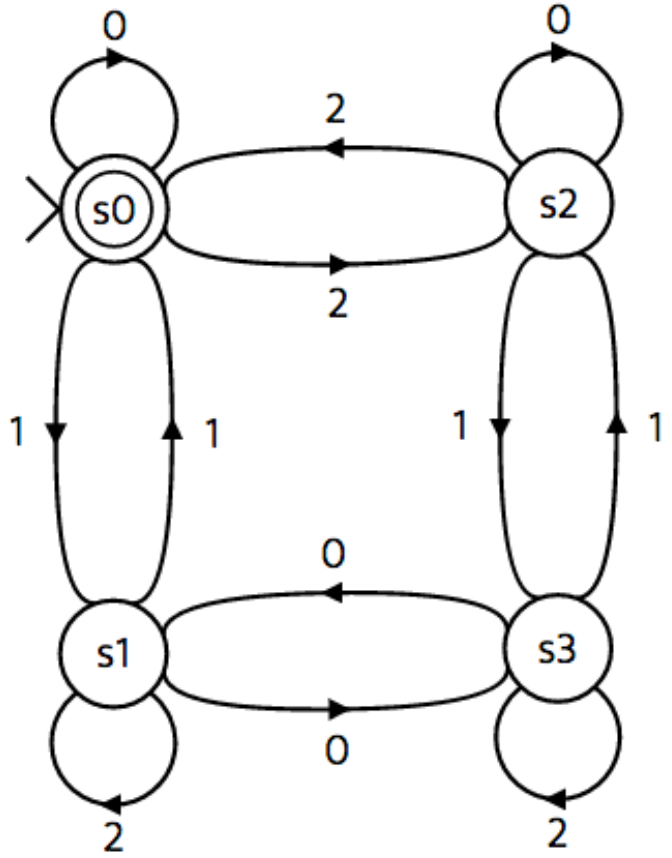



This is a **deterministic finite-state automaton, DFA**

What is the meaning of each state?

0
1
2
3
4
5
6
7
8
9

.Express each of these
decimal numbers
in ternary notation,
and find out
what state is reached
by the trace they determine



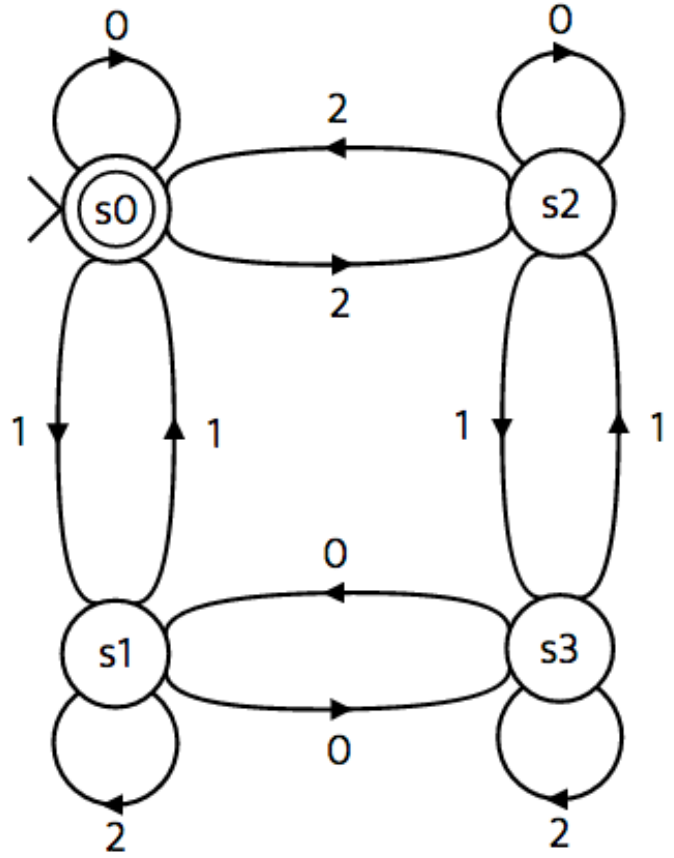

This is a
**deterministic finite-state automaton, DFA**

What is the meaning of each state?

0
1
2
3
4
5
6
7
8
9

This machine counts
ternary
mod 4

$F(q, s) = 3 * q + s \pmod 4$

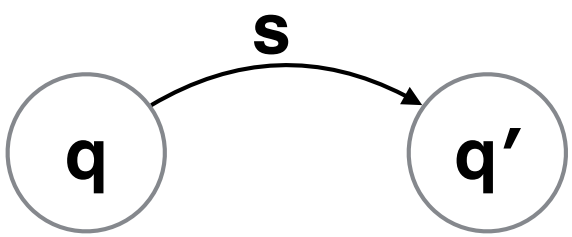



This is a
**deterministic finite-state automaton, DFA**

# Formal Definition DFA

Deterministic FSM, deterministic finite automaton
$D = (Q, s_0, F, \Sigma, N)$

- Set of states, Q
- Start state $s_0 \in Q$
- Accepting states $A \subseteq Q$
- Alphabet $\Sigma$
- Next state function,
- $q' = N(q, s)$
  where $q, q' \in Q$
  and $s \in \Sigma$.



# A mathematical definition of a Finite State Machine.

A machine M consists of:

**Q**: the set of states,
**Σ**: the alphabet of the machine
- the tokens the machine can process,

**B**: the set of **beginning** or start states of the machine
**A**: the set of the machine's **accepting** states.
**δ**: the set of **transitions**
is a set of (state, symbol, state) triples
$\delta \subseteq Q \times \Sigma \times Q$.

A finite-state automaton, or machine (FSM)

M consists of:

**Q**: the set of states,

**Σ**: the alphabet of the machine

- the tokens the machine can process,

**B**: the set of **beginning** or start states of the machine

**A**: the set of the machine's **accepting** states.

**δ**: the set of **transitions**

is a set of (state, symbol, state) triples

$$\delta \subseteq Q \times \Sigma \times Q.$$

An FSM is a deterministic automaton **DFA** if

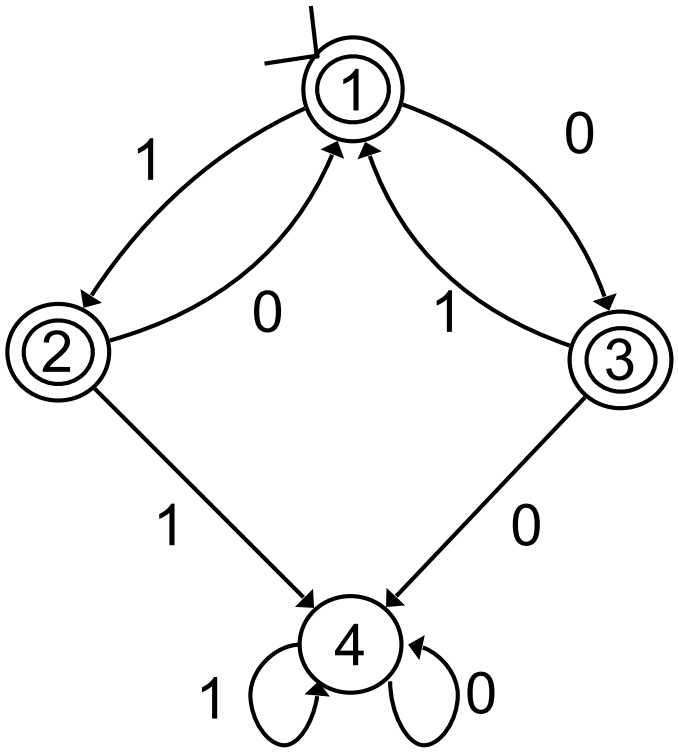
it has a single start state $B = \{ s0 \}$

it has a next-state function

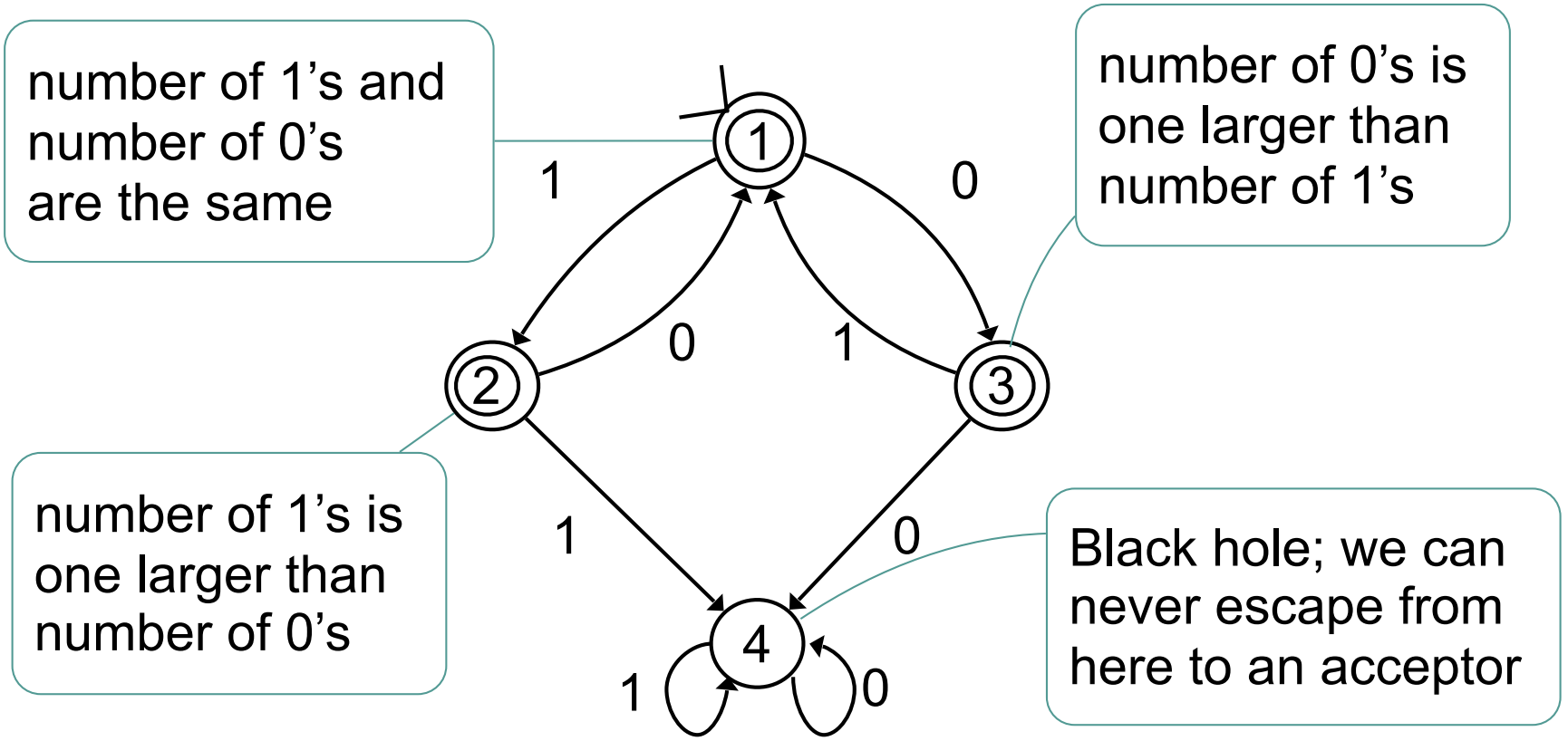
$$F : Q \times \Sigma \rightarrow Q$$

such that $\delta = \{ ( q, s, F(q,s)) \mid (q,s) \in Q \times \Sigma \}$

# Acceptor Example



# Acceptor Example




Accepts strings of 0's and 1's for which the difference between number of 0's and number of 1's in a subsequence is at most 1.