CSE 311

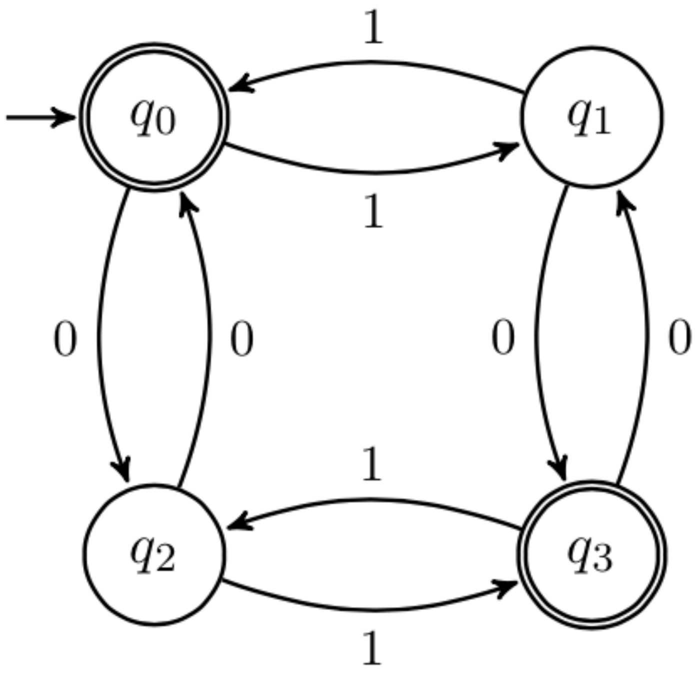
Foundations of Computing I

* All slides are a combined effort between previous instructors of the course

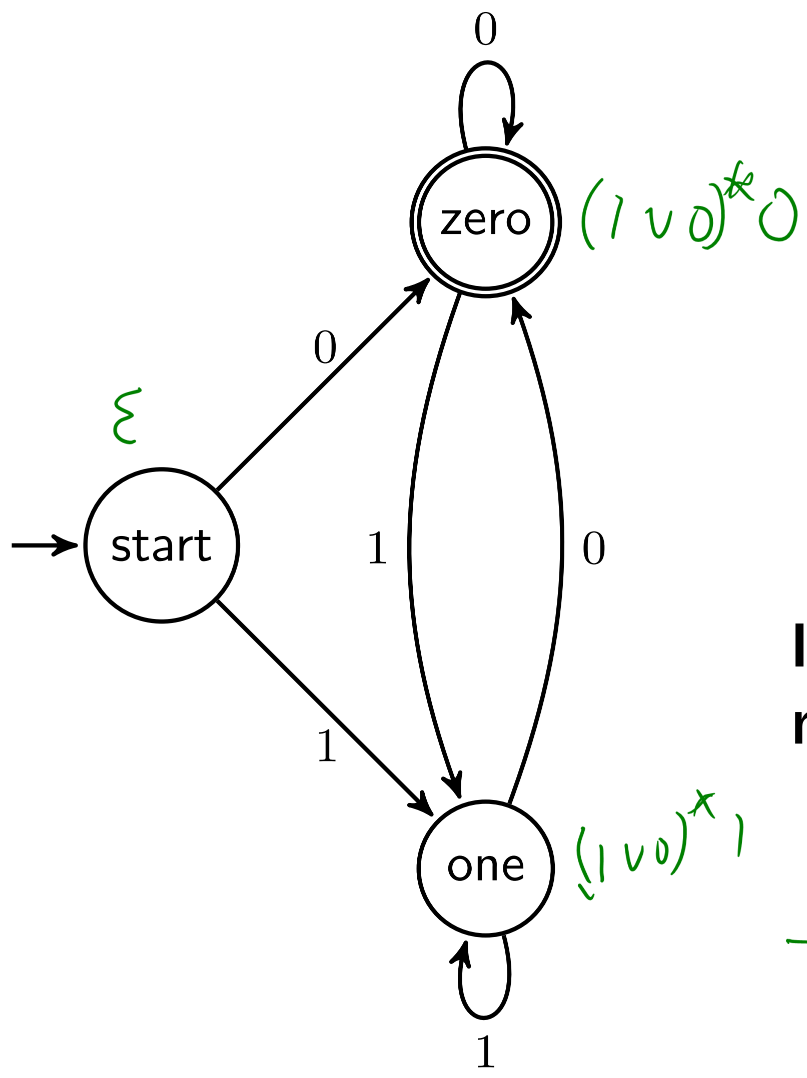
CSE 311: Foundations of Computing

Lecture 20: Finite State Machines (DFAs)

Hi!!



A Weird Sort of Programming!



0101 → one

101 → one

10

0

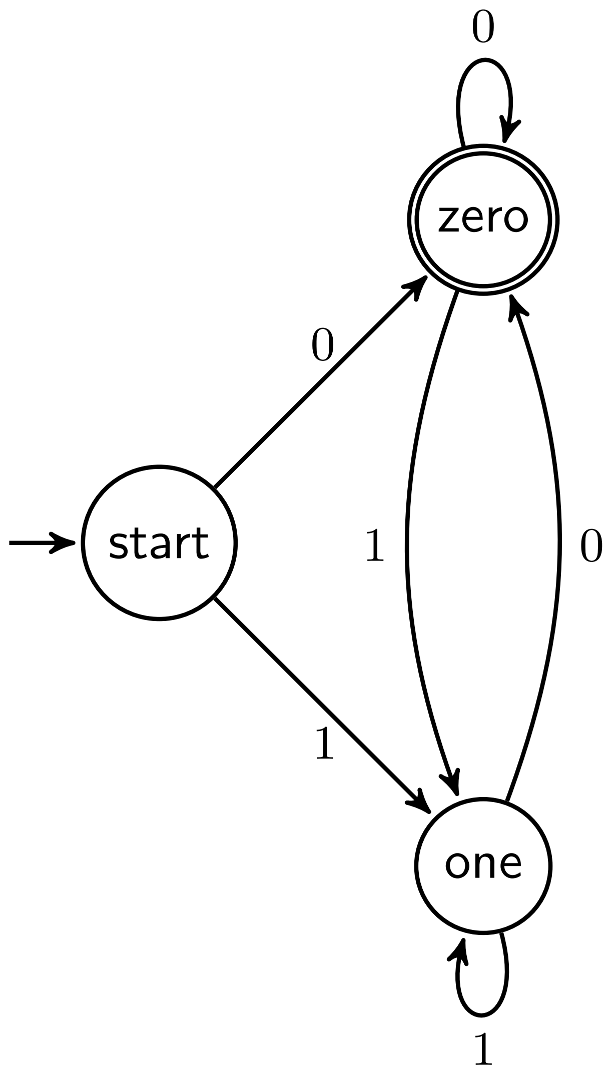
What does this “thing” do?

Take a guess!

If you had to give this “method” a name, what would it be?

boolean isEven (String s)

A Weird Sort of Programming!



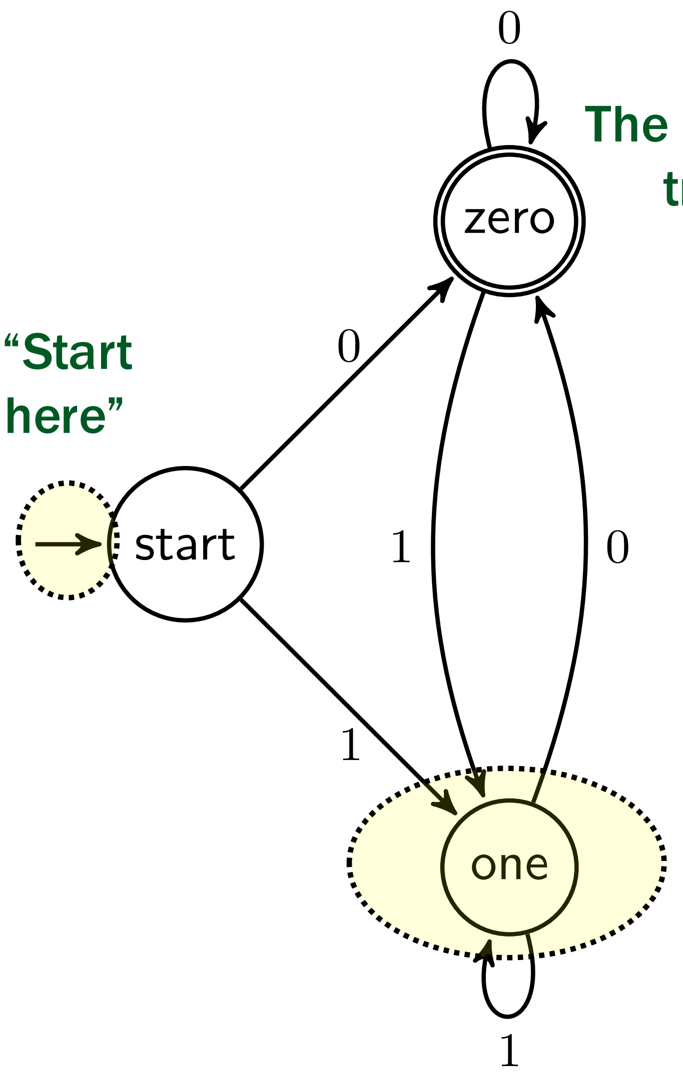
What does this "thing" do?

Take a guess!

If you had to give this "method" a name, what would it be?

```
boolean isEven(binary s)
```

Finite State Machines (“DFAs”)



“Start here”

The “double circle” means “return true if the input ends here”

“If I get X, follow the arrow…”

The circles are called “states”
We’re only in a single state at any point in time…

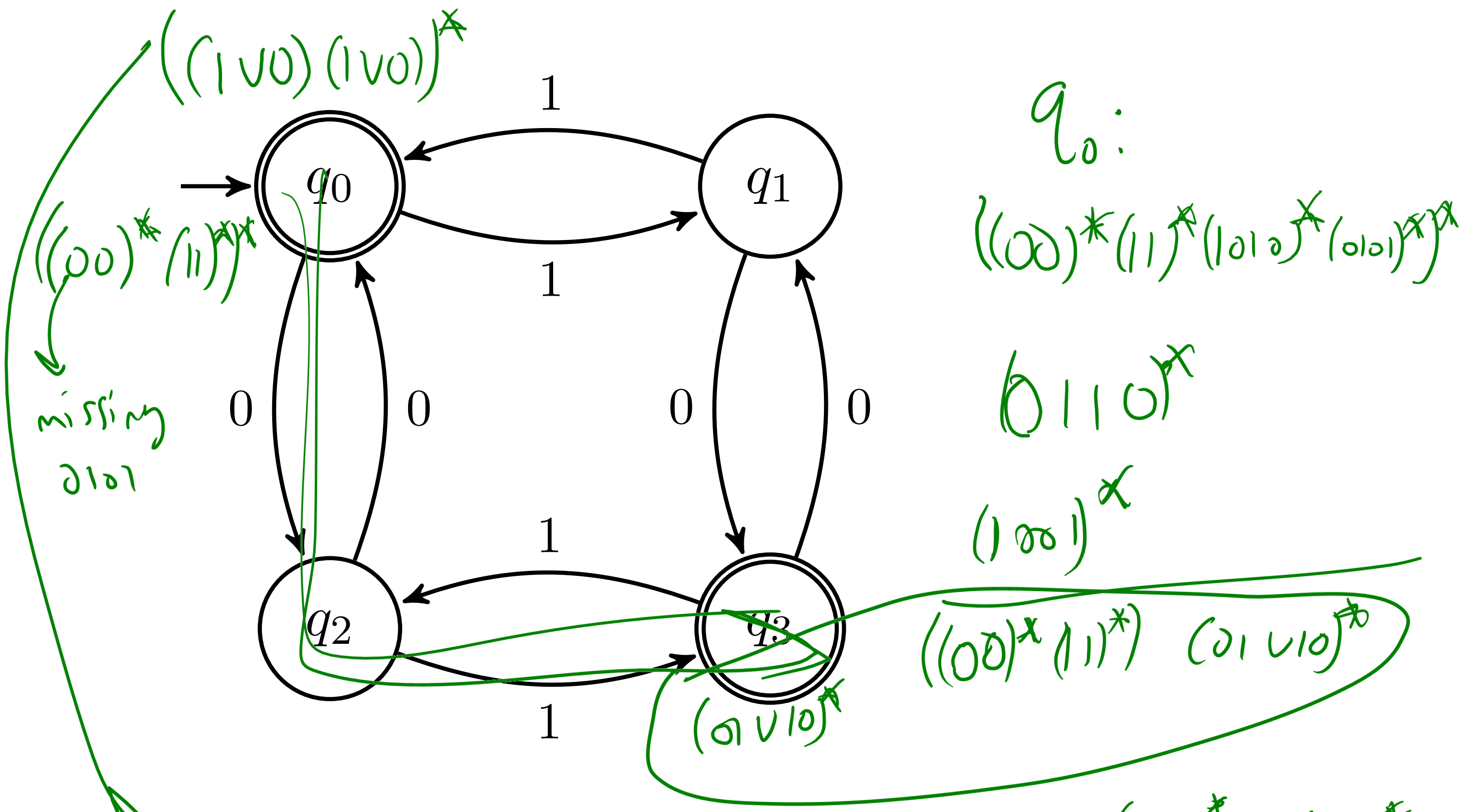
Applications of FSMs (a.k.a. finite automata)

- Implementation of regular expression matching in programs like `grep`
- Control structures for sequential logic in digital circuits
- Algorithms for communication and cache-coherence protocols
  - Each agent runs its own FSM
- Design specifications for reactive systems
  - Components are communicating FSMs

Applications of FSMs (a.k.a. finite automata)

- Formal verification of systems
  - Is an unsafe state reachable?
- Computer games
  - FSMs provide worlds to explore
- Minimization algorithms for FSMs can be extended to more general models used in
  - Text prediction
  - Speech recognition

What language does this machine recognize?



$$\begin{tikzcd}
\to q_0 \arrow[r, bend right, "1"'] \arrow[d, bend right, "0"'] & q_1 \arrow[l, bend right, "1"'] \arrow[d, bend right, "0"'] \\
q_2 \arrow[u, bend right, "0"'] \arrow[r, bend right, "1"'] & q_3 \arrow[u, bend right, "0"'] \arrow[l, bend right, "1"']
\end{tikzcd}$$

q_0:

$((00)^*(11)^*(1010)^*(0101)^*)^*$

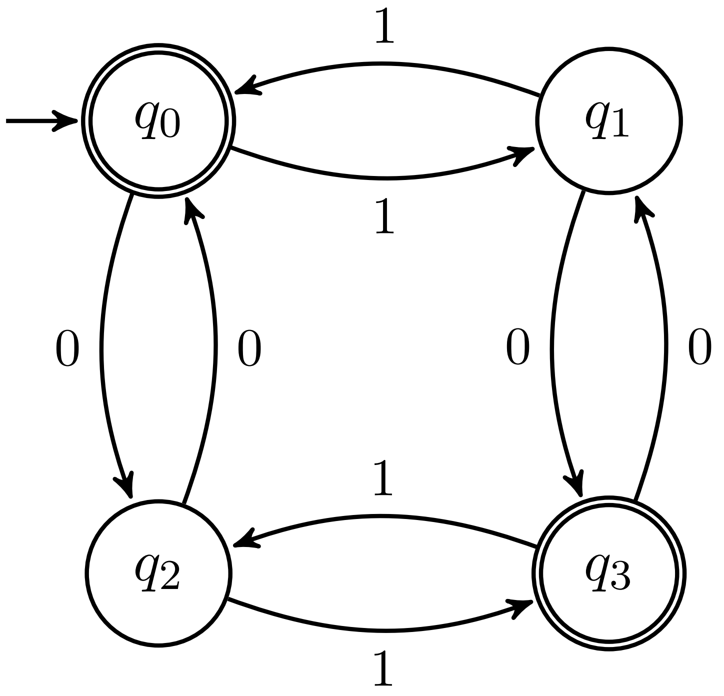
$(0110)^*$

$(1001)^*$

binary strings w/ even length

$(00)^* \cup (0101)^*$
$\cup (0011)^*$
$\cup (11)^*$
$\cup (01)^*$

What language does this machine recognize?

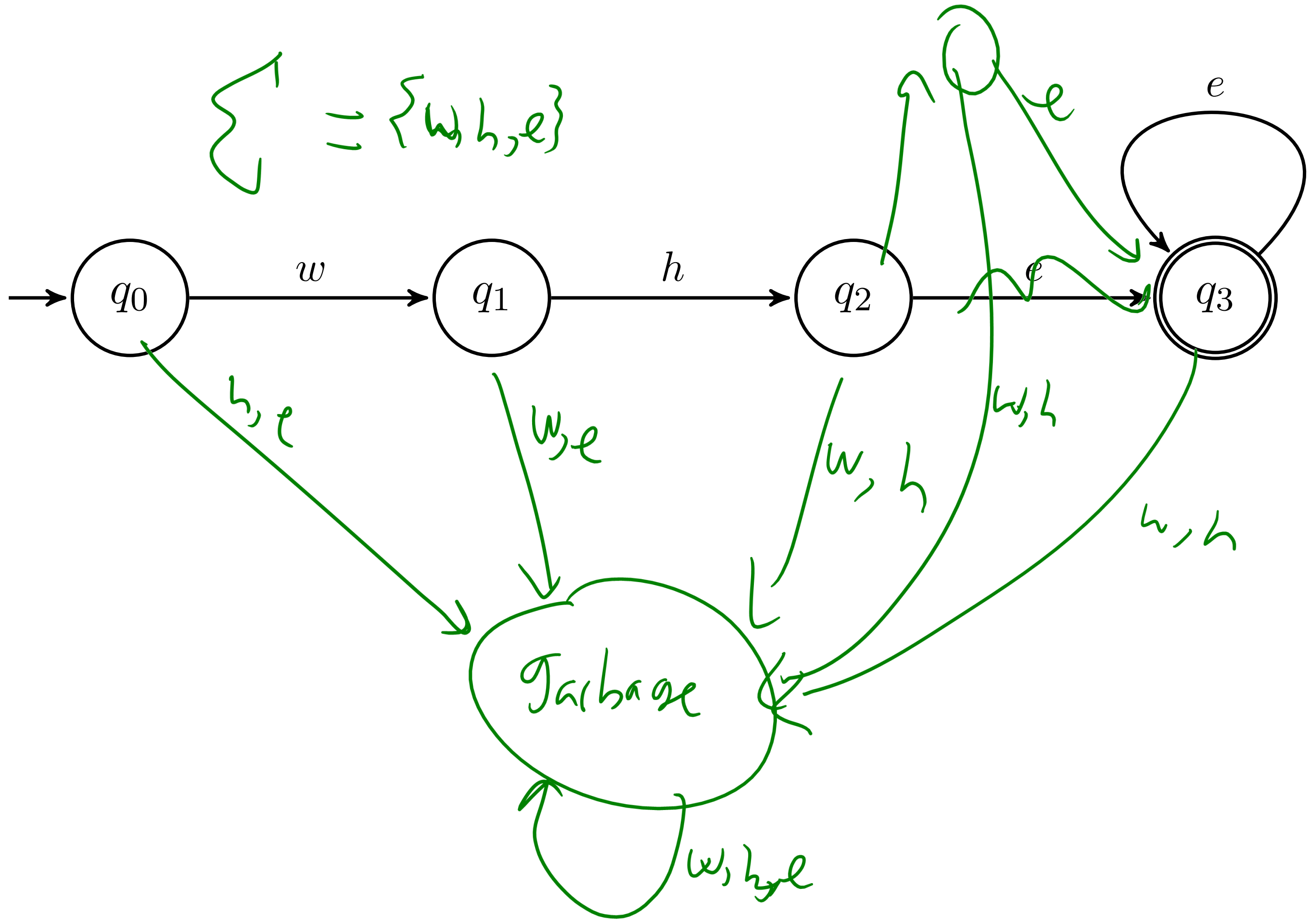


$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[r, bend right, "1"'] \arrow[d, bend right, "0"'] & q_1 \arrow[l, bend right, "1"'] \arrow[d, bend right, "0"'] \\
& q_2 \arrow[u, bend right, "0"'] \arrow[r, bend right, "1"'] & q_3 \arrow[l, bend right, "1"'] \arrow[u, bend right, "0"']
\end{tikzcd}$$

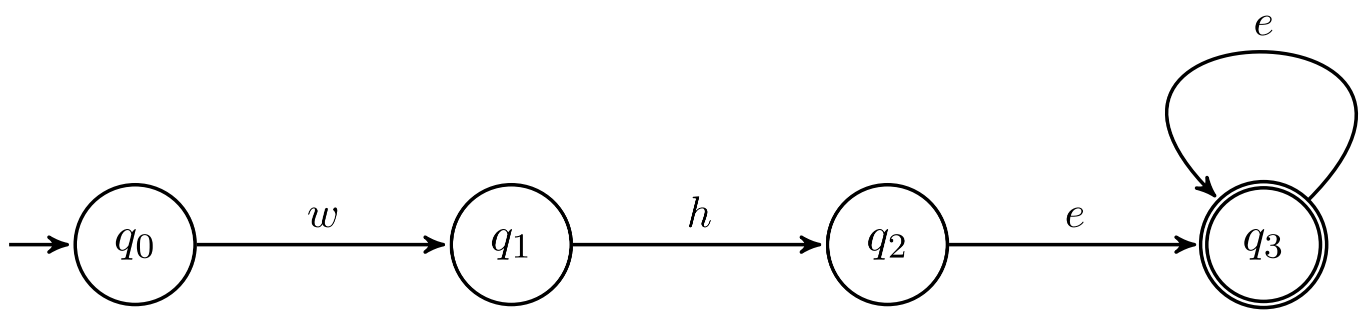
All binary strings with even length

Why is this not a DFA? Fix it!

$\Sigma = \{w, h, e\}$

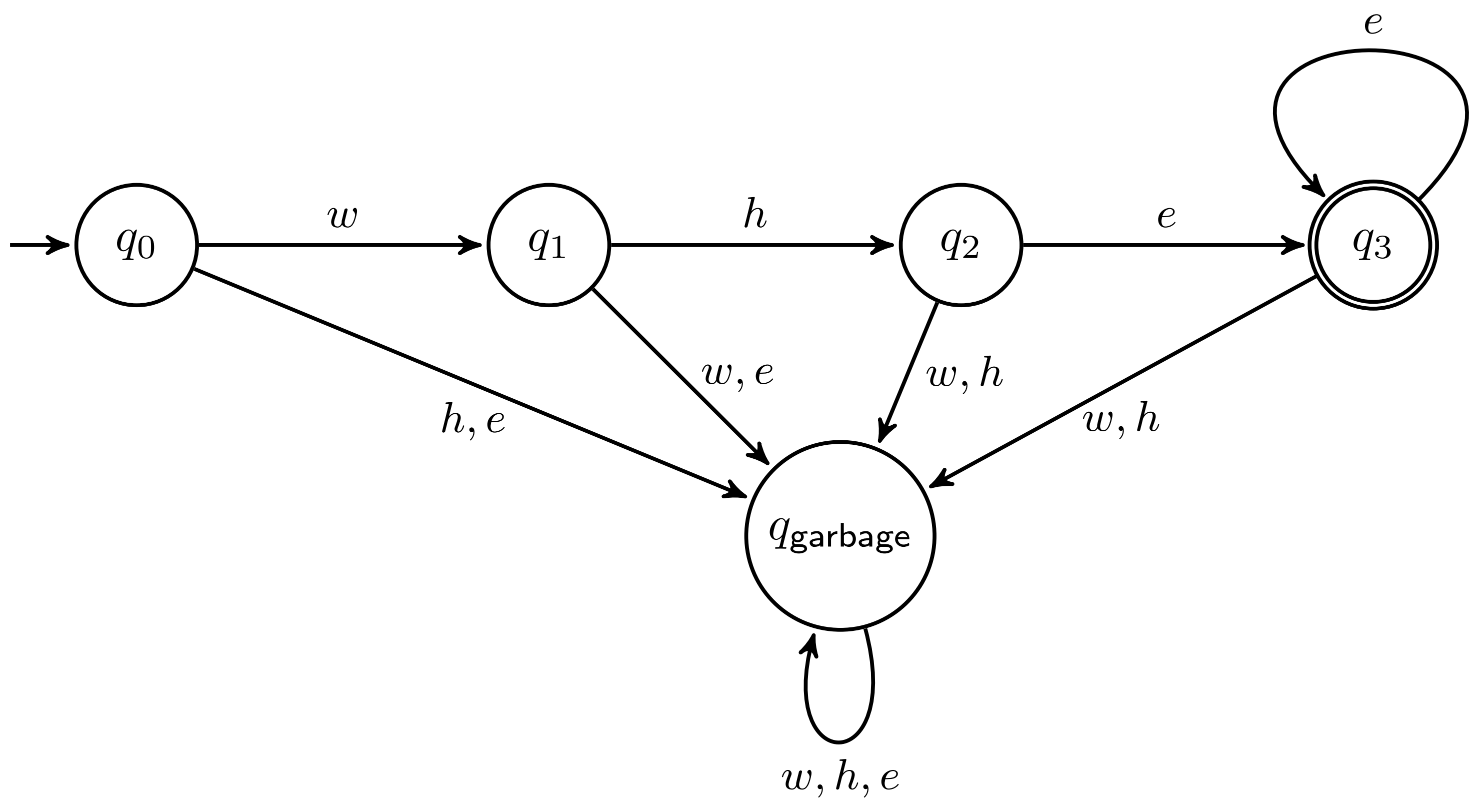


Why is this not a DFA? Fix it!



DFAs must have a transition for every character at every state!

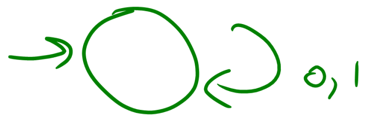
Why is this not a DFA? Fix it!

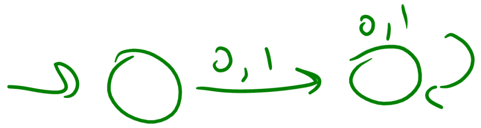


"Garbage states" are a useful concept. Whenever we KNOW we can't accept the string, just send it to a state that will always go back to itself. This is the way of saying "return false" in DFA-land.

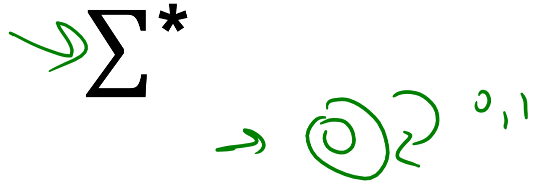
For each of the following languages, create a DFA

$\emptyset$ $\cancel{\{\}}$

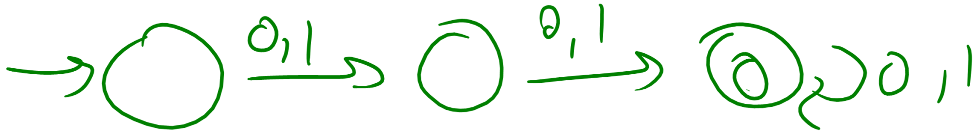




Σ^*



$\{x \in \{0,1\}^* : \text{len}(x) > 1\}$

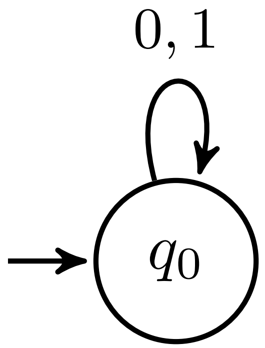


$\varepsilon \neq \emptyset$

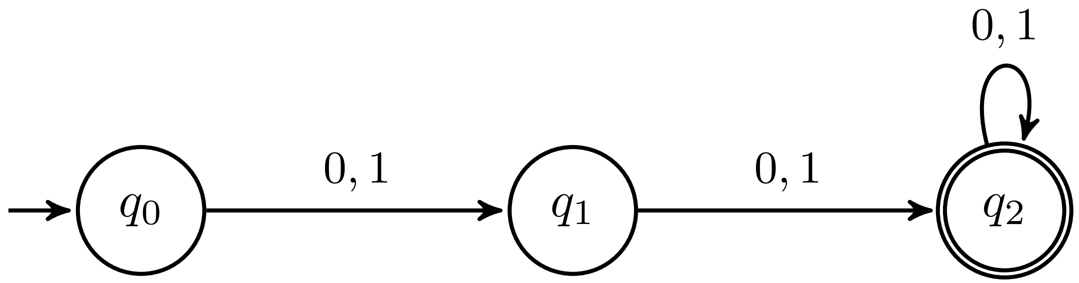
$\{\varepsilon\}$

For each of the following languages, create a DFA

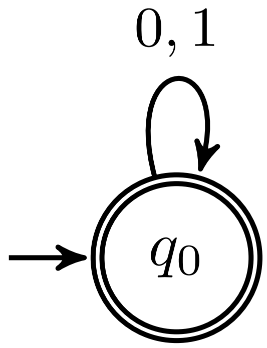
$\emptyset$



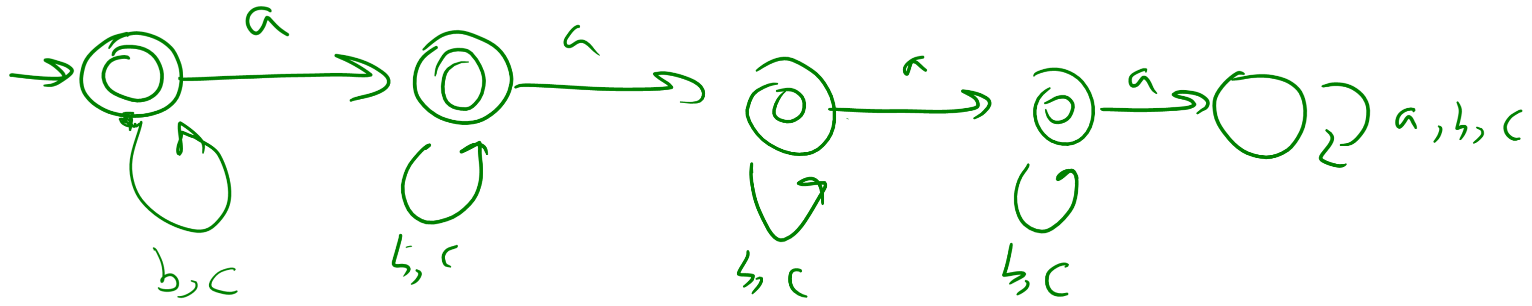
$\{x \in \{0,1\}^* : \text{len}(x) > 1\}$



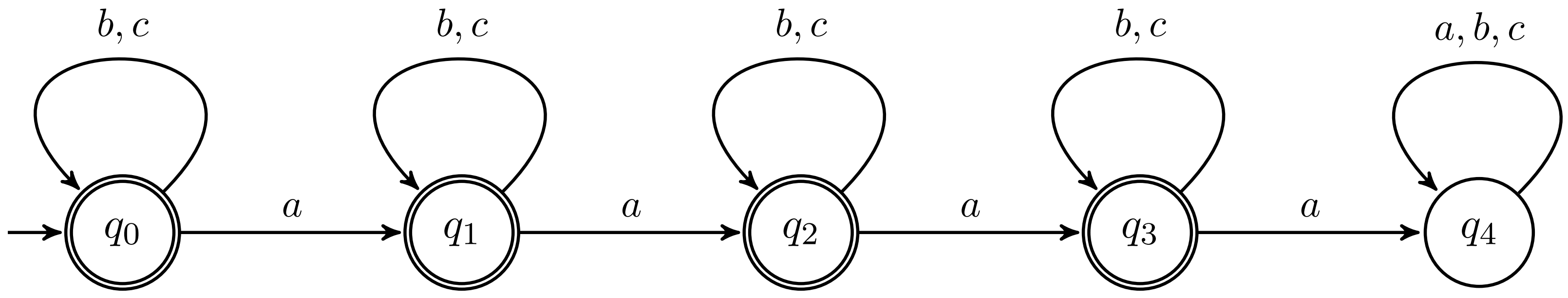
Σ^*



FSM that accepts strings of a's, b's, c's with no more than 3 a's

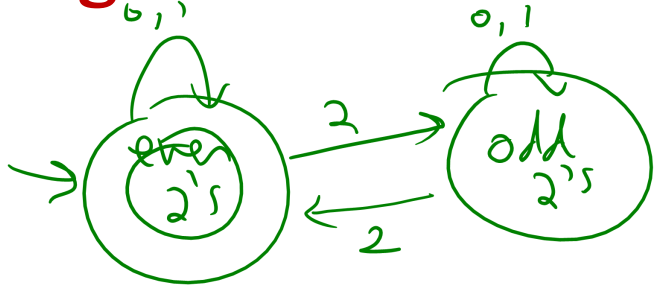


FSM that accepts strings of a’s, b’s, c’s with no more than 3 a’s

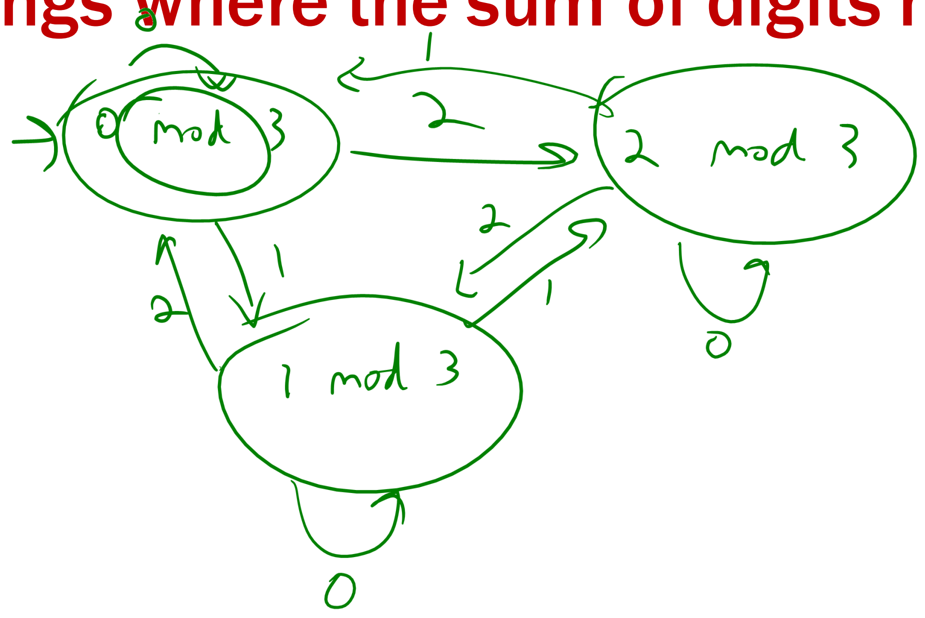


Strings over {0, 1, 2}*

M_1: Strings with an even number of 2's

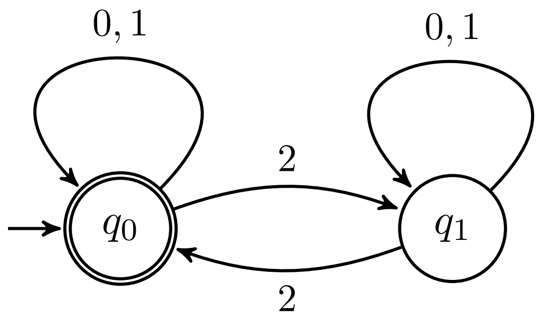


M_2: Strings where the sum of digits mod 3 is 0

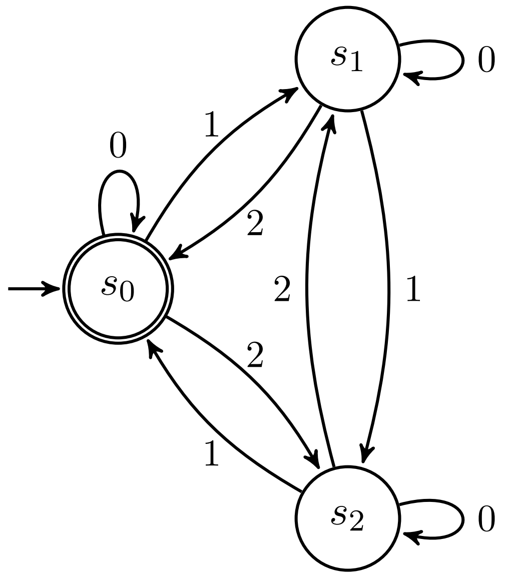


Strings over {0, 1, 2}*

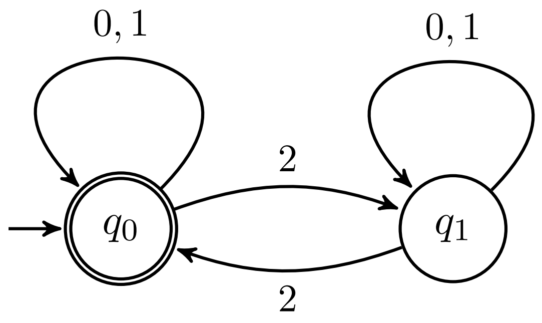
M_1: Strings with an even number of 2's

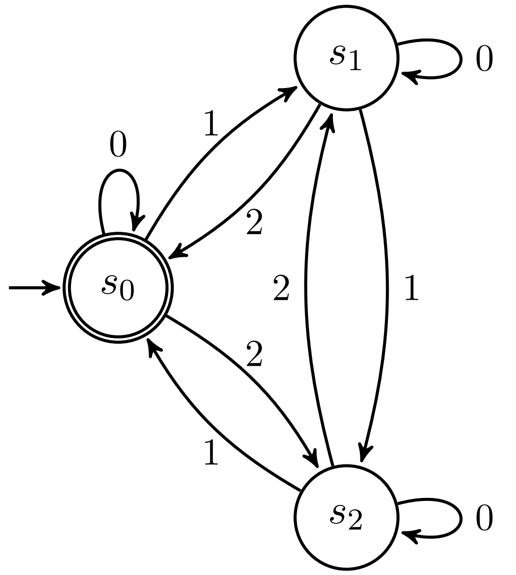


M_2: Strings where the sum of digits mod 3 is 0

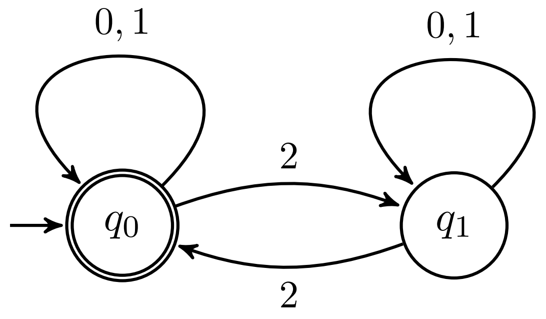


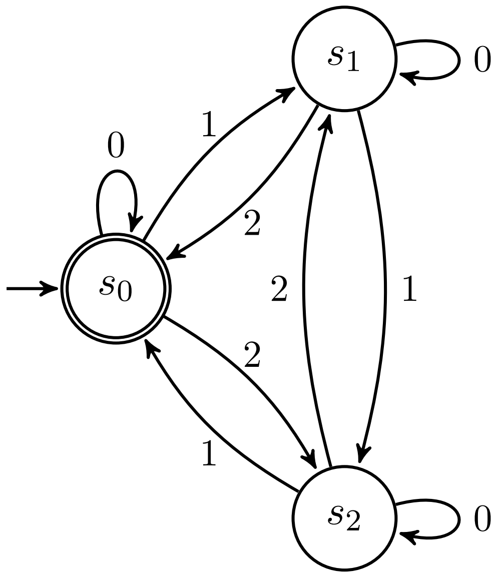
Strings with an even number of 2's AND a mod 3 sum of 0

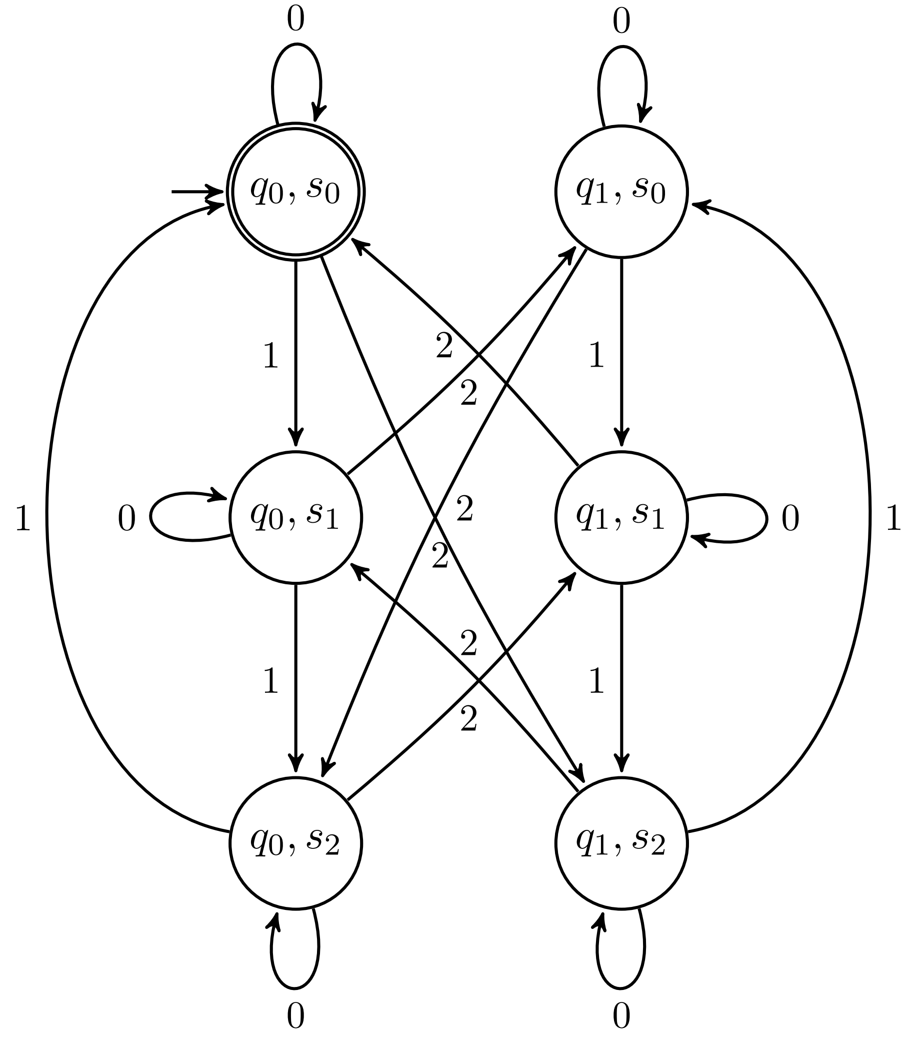




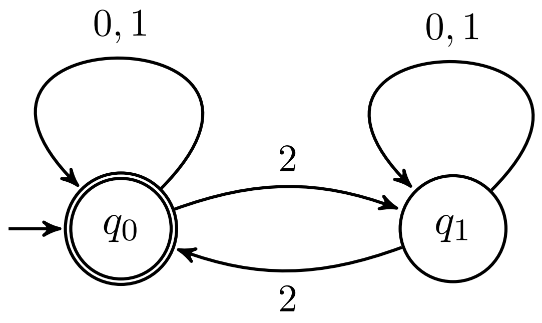
Strings with an even number of 2’s AND a mod 3 sum of 0

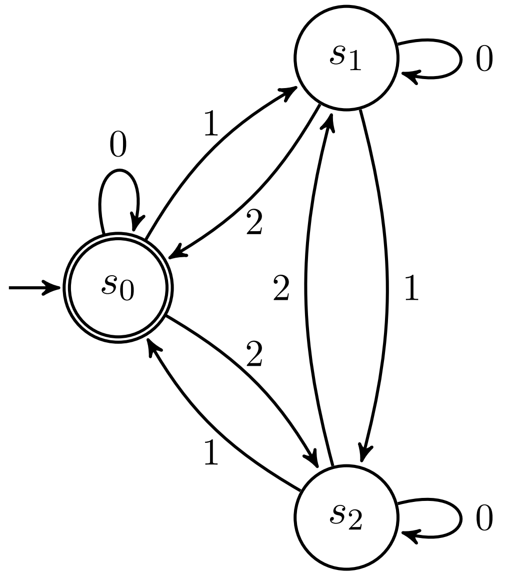




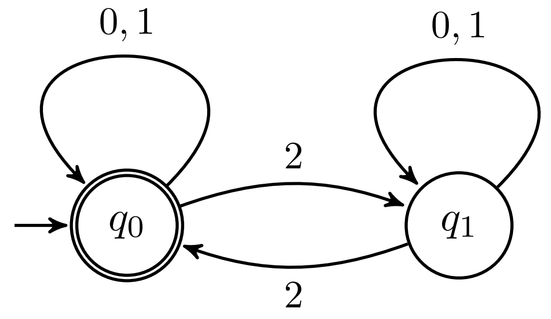


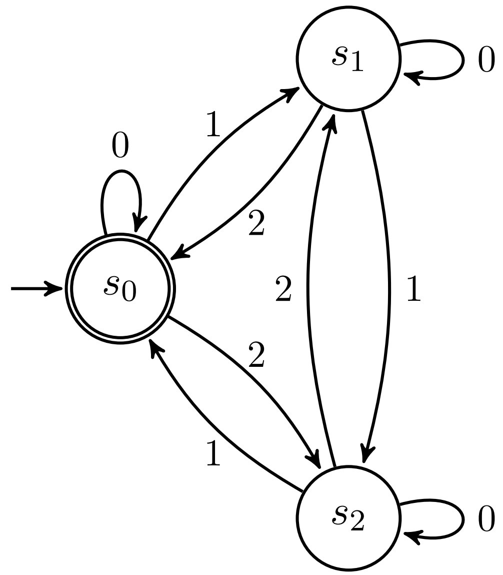
Strings with an even number of 2's OR a mod 3 sum of 0

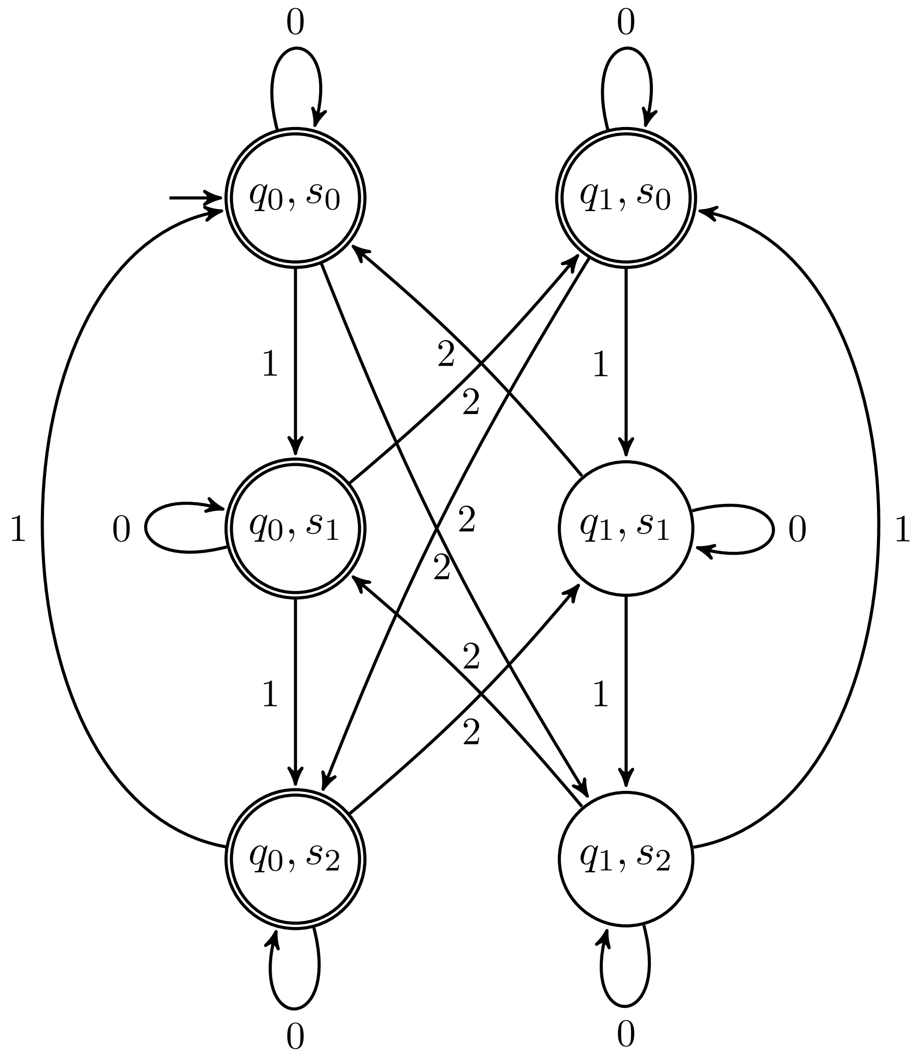




Strings with an even number of 2's OR a mod 3 sum of 0

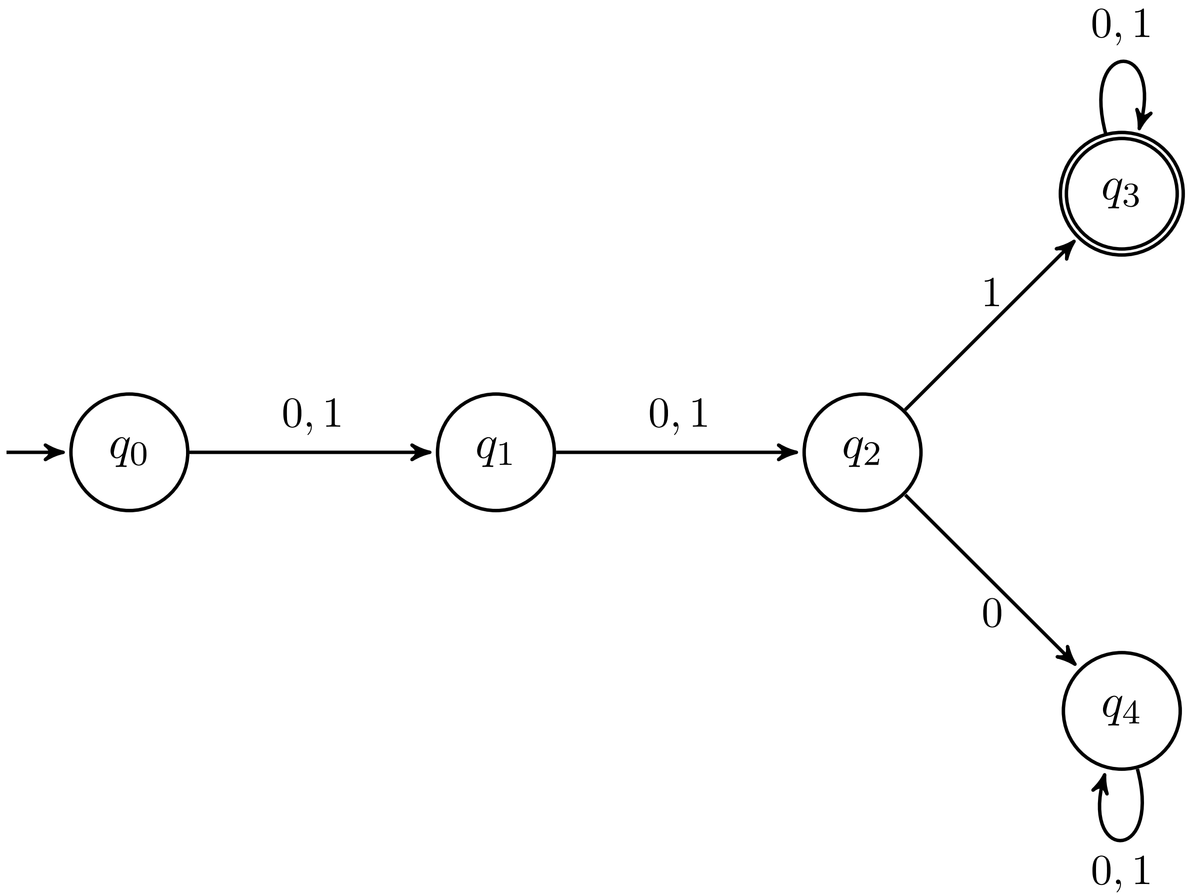






FSM that accepts binary strings with a 1 three positions from the start

FSM that accepts binary strings with a 1 three positions from the start


q_0
$0, 1$
q_1
$0, 1$
q_2
1
q_3
$0, 1$
0
q_4
$0, 1$