CSE 311 Lecture 23: Finite State Machines

Emina Torlak and Sami Davies

Topics

Finite state machines (FSMs)

Definition and examples.

Finite state machines with output

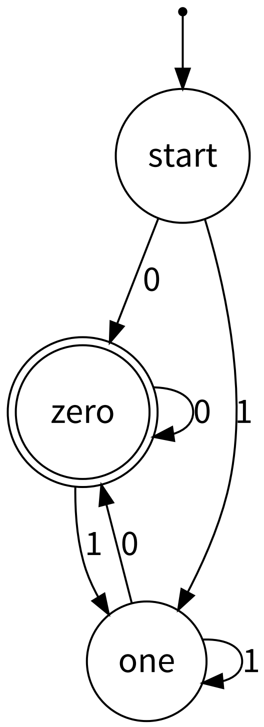
Definition and examples.

Finite state machines (FSMs)

Definition and examples.

Finite state machines by example

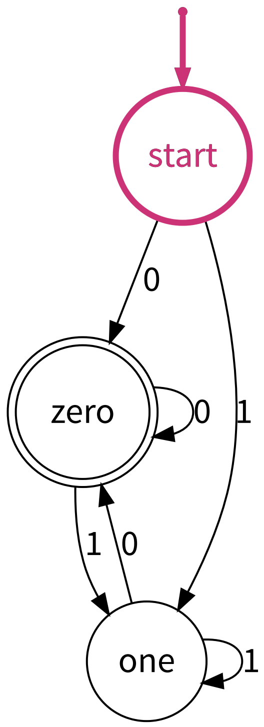
An FSM *recognizes* (or *accepts*) a set of strings.



Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.



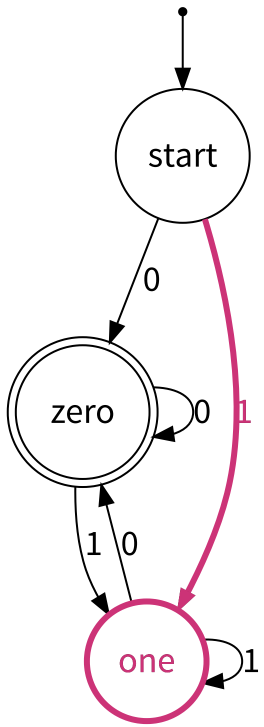
Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.



Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

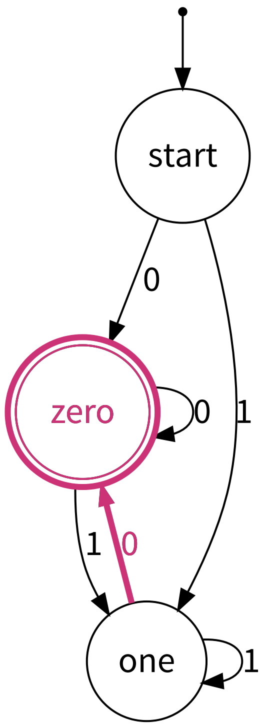
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

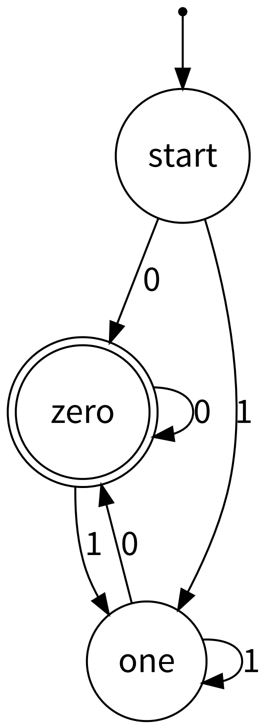
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Which strings are recognized by the example FSM?

ε

0

0110

1011

Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

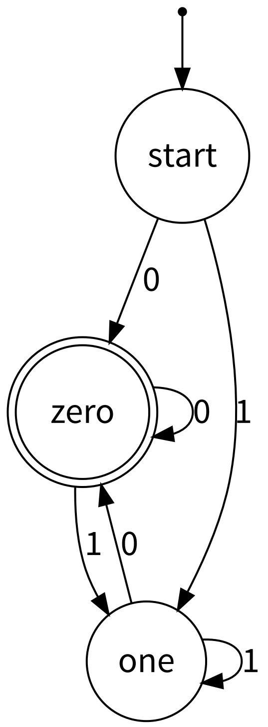
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Which strings are recognized by the example FSM?

ε no

0

0110

1011

Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

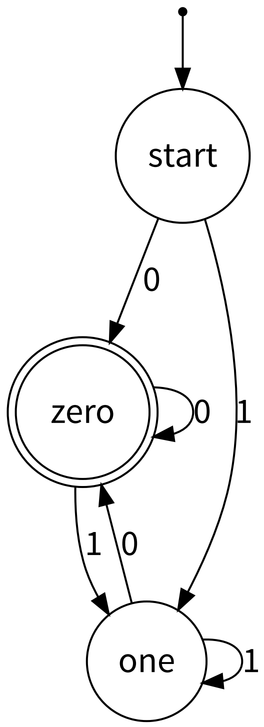
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Which strings are recognized by the example FSM?

ε	no
0	yes
0110	
1011	

Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

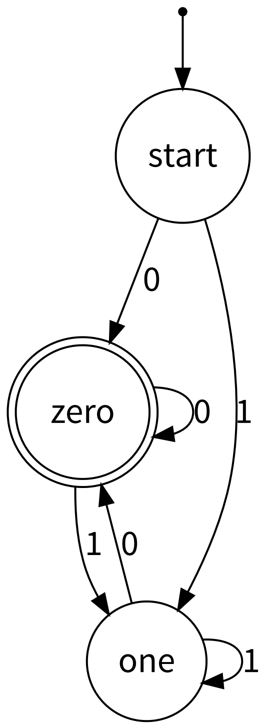
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Which strings are recognized by the example FSM?

ε	no
0	yes
0110	yes
1011	

Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

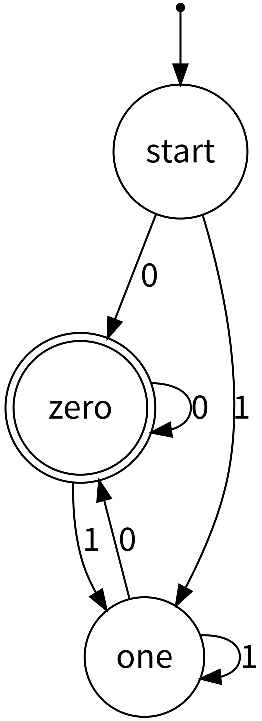
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Which strings are recognized by the example FSM?

ε	no
0	yes
0110	yes
1011	no

Finite state machines by example

An FSM *recognizes* (or *accepts*) a set of strings.

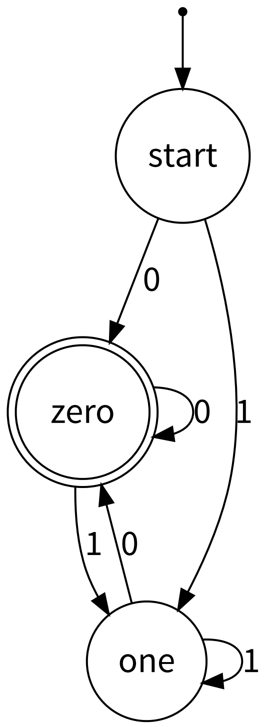
(1) Given a string s, begin at the *start state*, denoted by an incoming arrow with no source.

(2) If $s = aw$, take the edge labeled a to get to the next state.

(3) Otherwise (s is empty), stop.

(4) Let s be w and repeat (2)-(4) until the machine stops.

The FSM *accepts* s iff it stops in a *final state*, denoted by a double circle.



Which strings are recognized by the example FSM?

ε	no
0	yes
0110	yes
1011	no

All binary strings that end in 0.

Defining an FSM

Finite state machine (FSM)

A *finite state machine* (FSM) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

Defining an FSM

Finite state machine (FSM) or deterministic finite automaton (DFA)

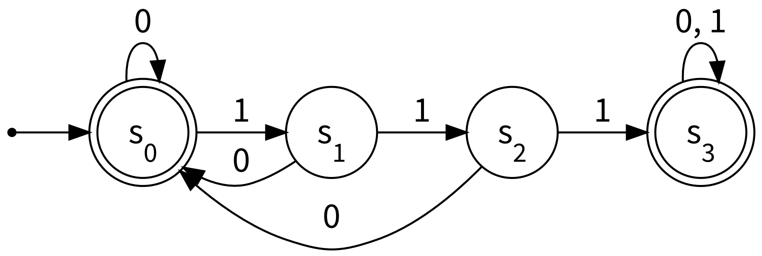
A *finite state machine* (FSM) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of
a finite set of *states* S, a finite *input alphabet* Σ_{in},
a *transition function* f that maps each state in S and input in Σ_{in} to a state in S,
a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

Defining an FSM

Finite state machine (FSM) or deterministic finite automaton (DFA)

A *finite state machine* (FSM) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

state	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_3
s_3	s_3	s_3



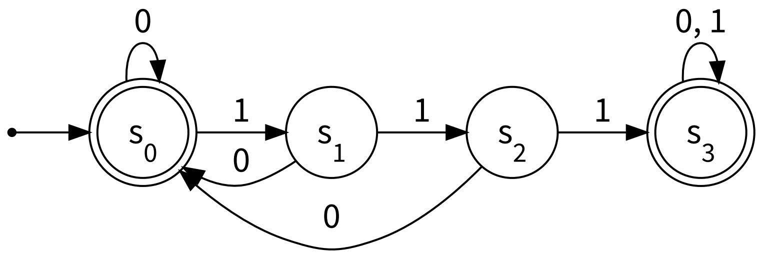
What language does this FSM accept?

Defining an FSM

Finite state machine (FSM) or deterministic finite automaton (DFA)

A *finite state machine* (FSM) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

state	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_3
s_3	s_3	s_3



What language does this FSM accept?

The set of all binary strings that contain 111 or end in 0.

Applications of FSMs

Implementation of string matching.

For example, in `grep`.

Algorithms for communication and cache-coherence protocols.

Each agent runs its own FSM.

Specifications for controllers.

For example, device drivers.

Specifications for reactive systems.

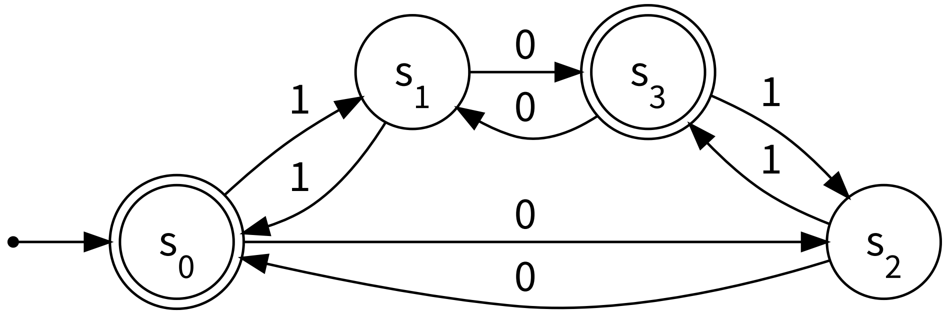
Components are communicating FSMs.

Verification.

Is an unsafe state reachable?

Example: an FSM over binary strings

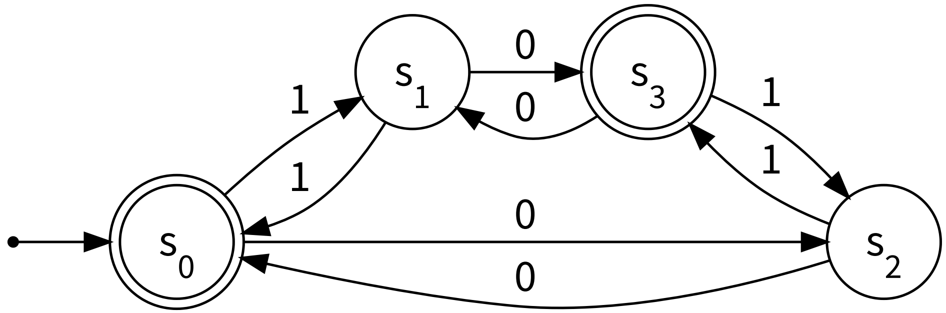
What language does this FSM recognize?



Example: an FSM over binary strings

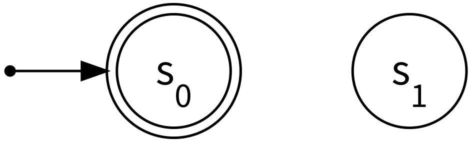
What language does this FSM recognize?

The set of all binary strings where the numbers of 1's and 0's are congruent mod 2. That is, both numbers are even or both are odd.

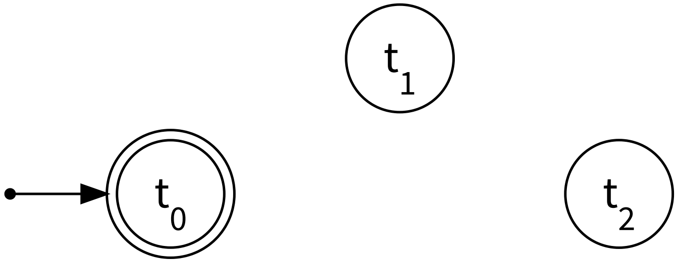


Example: FSMs over {0, 1, 2}

M_1: strings with an even number of 2's

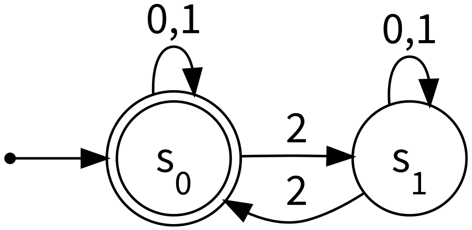


M_2: strings where the sum of digits mod 3 is 0

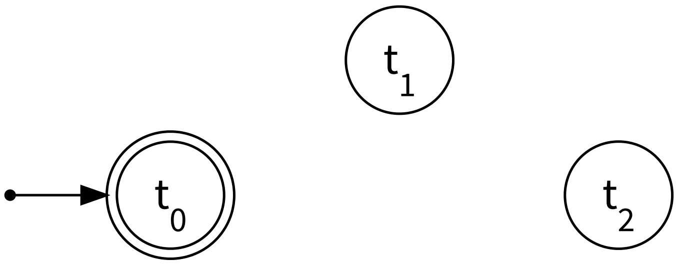


Example: FSMs over {0, 1, 2}

M_1: strings with an even number of 2's

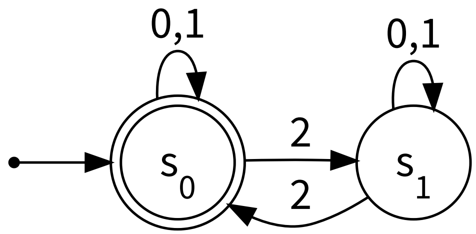


M_2: strings where the sum of digits mod 3 is 0

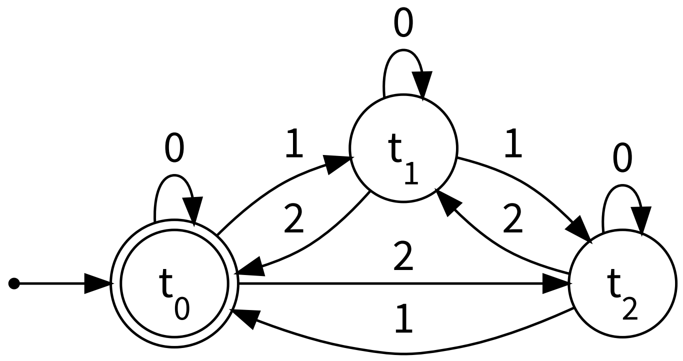


Example: FSMs over {0, 1, 2}

M_1: strings with an even number of 2's

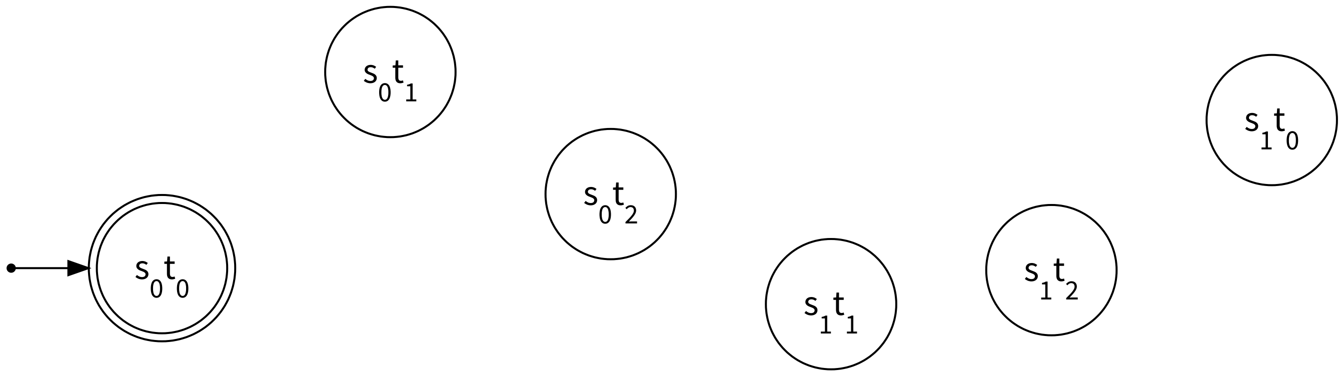


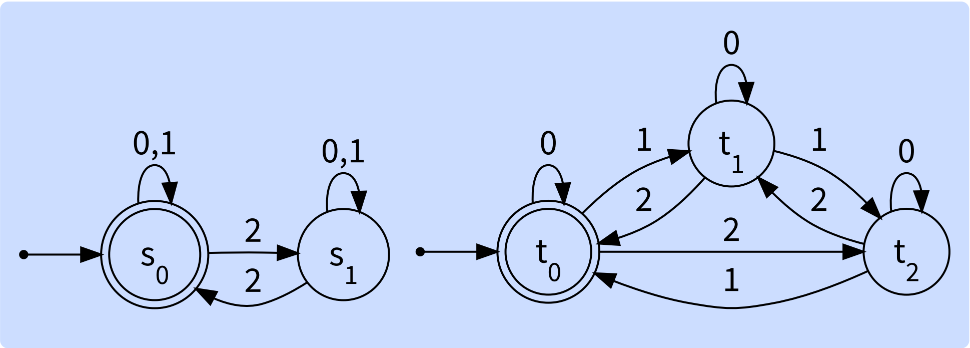
M_2: strings where the sum of digits mod 3 is 0



Example: a combined FSM over {0, 1, 2}

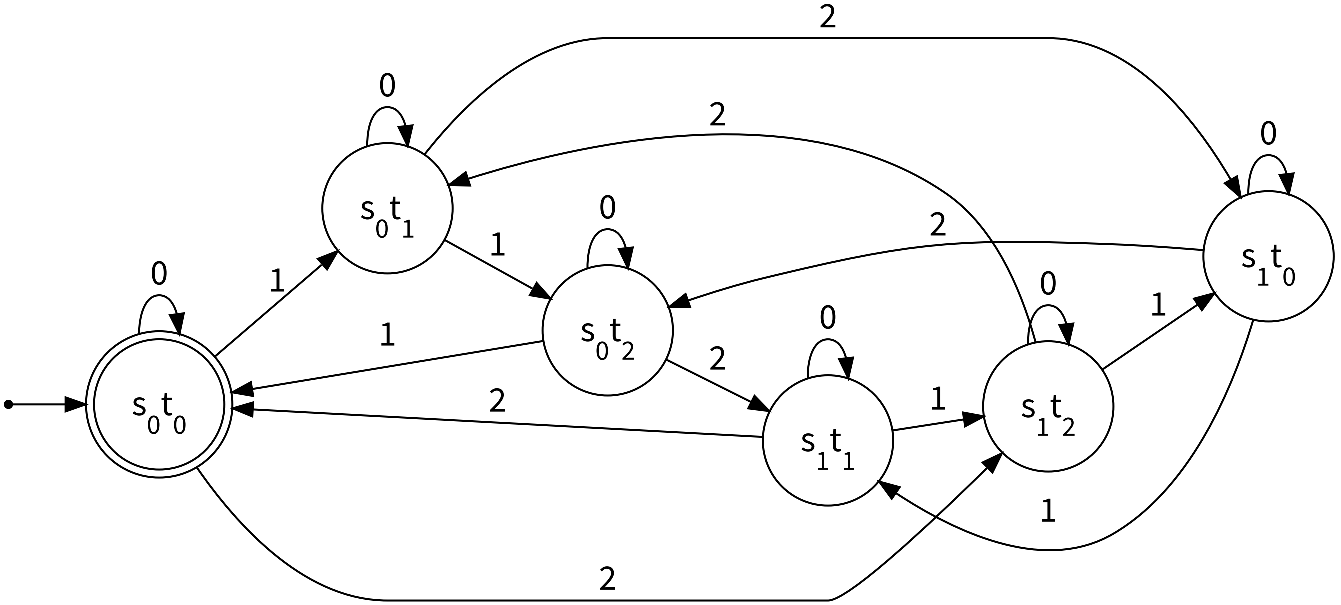
M_1 and M_2: strings with an even number of 2's where digit sum mod 3 is 0

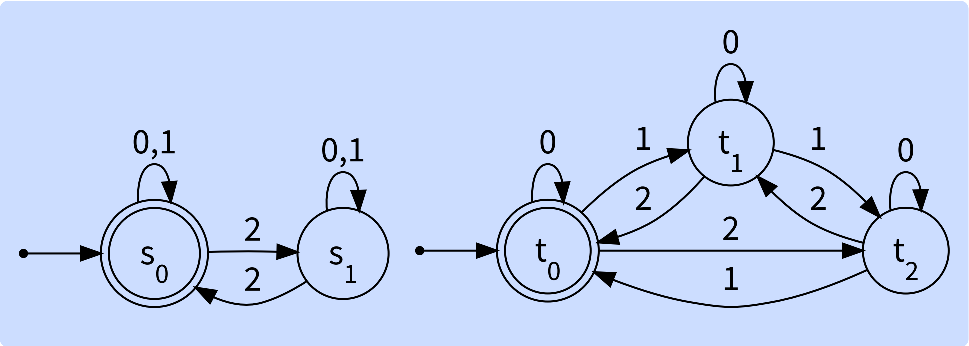




Example: a combined FSM over {0, 1, 2}

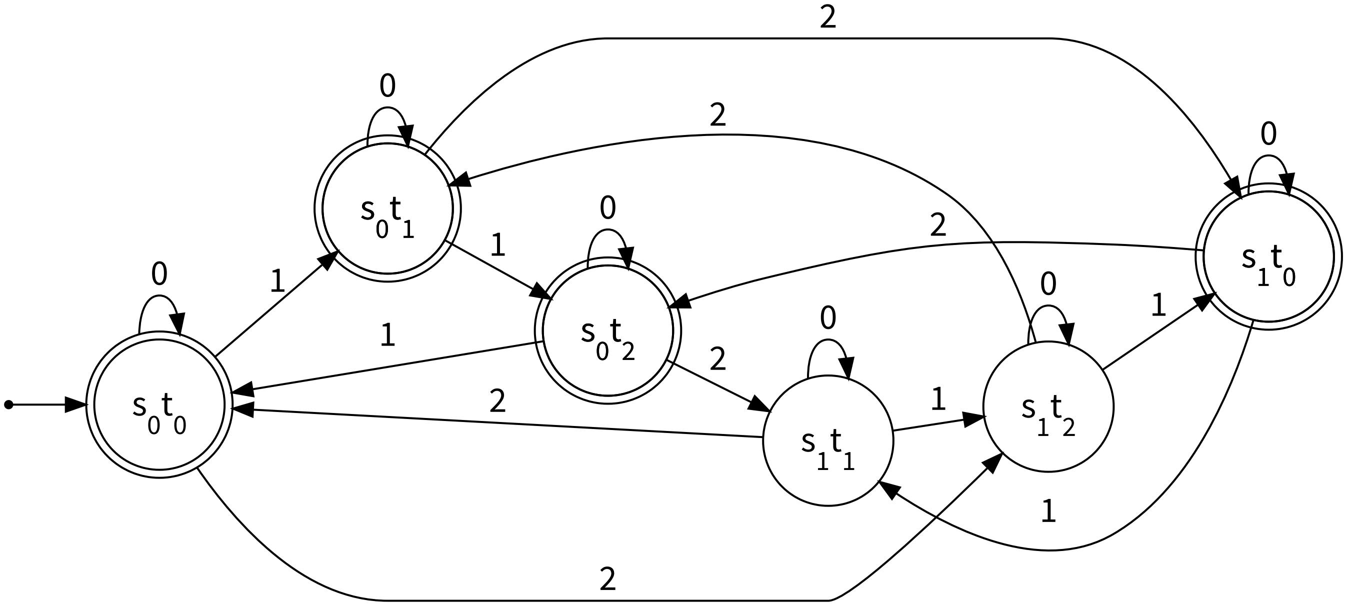
M_1 and M_2: strings with an even number of 2's where digit sum mod 3 is 0

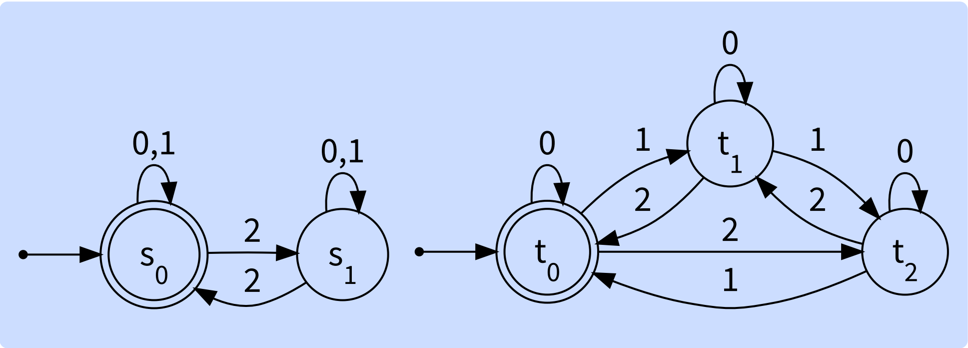




Example: another combined FSM over {0, 1, 2}

M_1 or M_2: strings with an even number of 2's or digit sum mod 3 is 0





Finite state machines with output

Definition and examples.

Adding output to FSMs

FSMs can do more than just accept or reject strings.

So DFAs aren't the only kind of FSM!

Another useful kind of FSM can also return output.

Whenever you enter specific states, the FSM produces an output.

These FSMs (Moore machines) are used as controllers.

Adding output to FSMs

FSMs can do more than just accept or reject strings.

So DFAs aren't the only kind of FSM!

Another useful kind of FSM can also return output.

Whenever you enter specific states, the FSM produces an output.

These FSMs (Moore machines) are used as controllers.

Finite state machine (FSM) with output

A *finite state machine* (FSM) $M = (S, \Sigma_{\text{in}}, \Sigma_{\text{out}}, f, g, s_0)$ consists of

a finite set of *states* S, a finite *input alphabet* Σ_{in},

a finite *output alphabet* Σ_{out},

a *transition function* f that maps each state in S and input in Σ_{in} to a state in S,

an *output function* g that maps each state in S to an output symbol in Σ_{out},

and *start state* $s_0 \in S$.

Example: vending machine

Consider a simple vending machine.

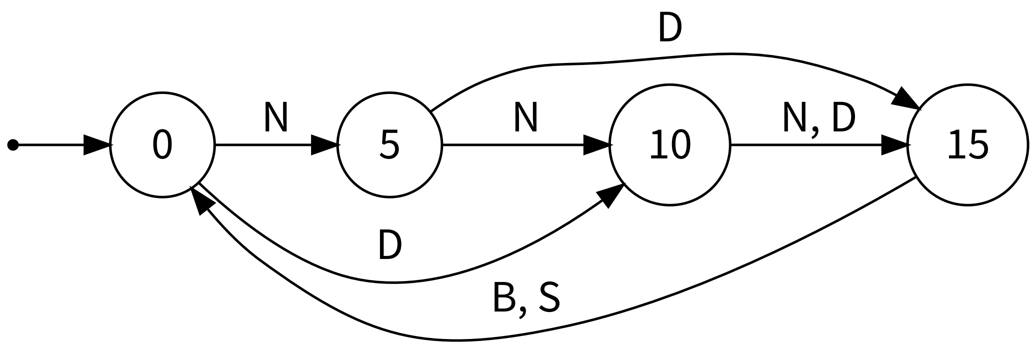
- Enter 15 cents in dimes or nickels.
- Press S (Snickers) or B (Butterfinger) for a candy bar.

Example: vending machine

Consider a simple vending machine.

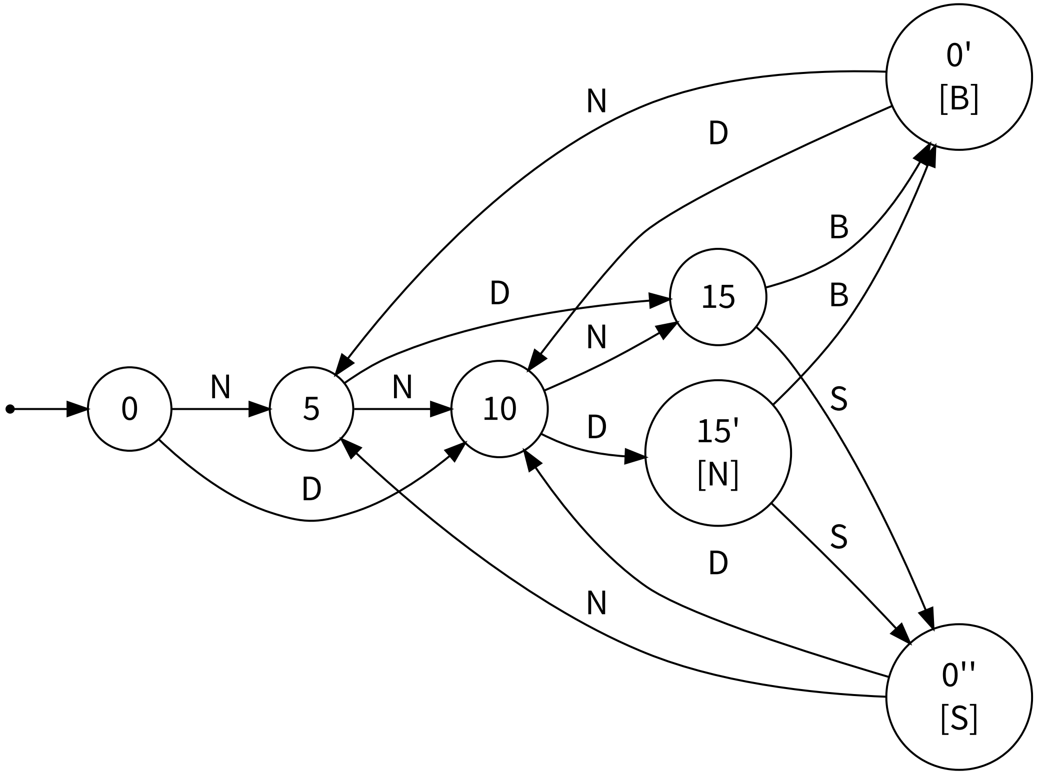
Enter 15 cents in dimes or nickels.

Press S (Snickers) or B (Butterfinger) for a candy bar.



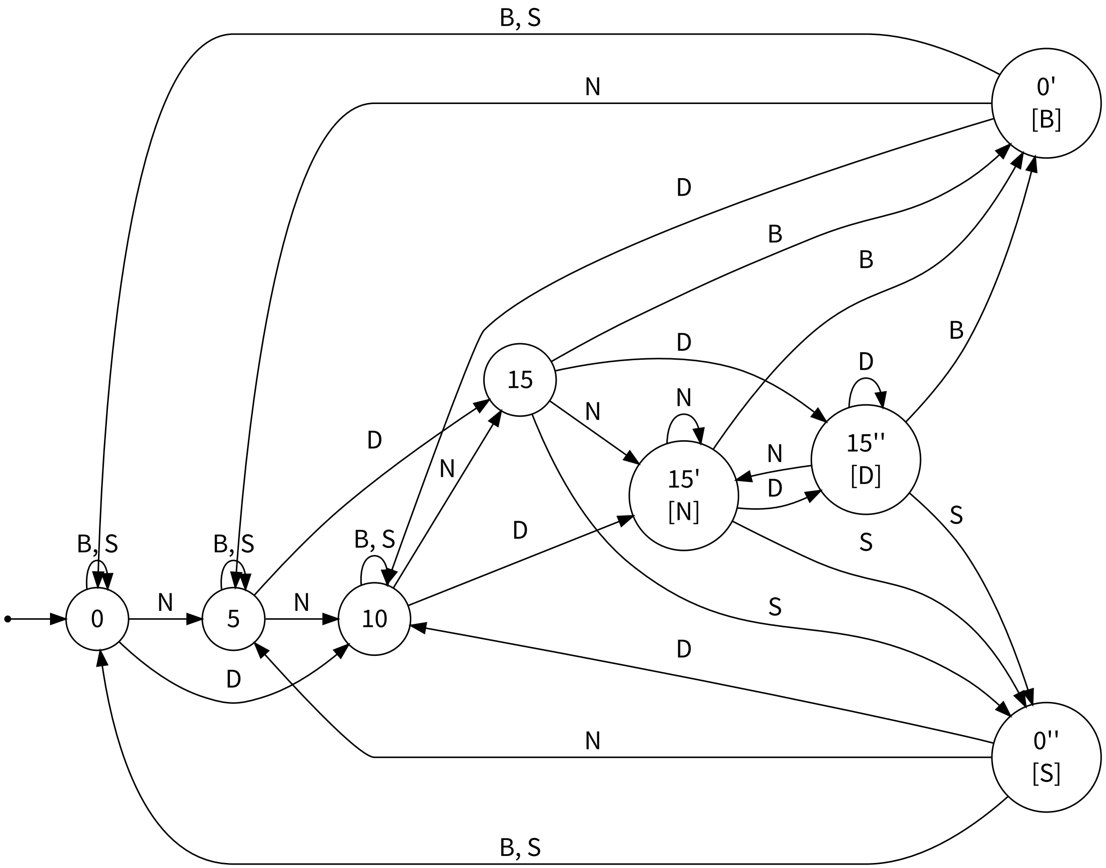
Basic transitions on N (nickel), D (dime), B (Butterfinger), and S (Snickers).

Example: vending machine with output



Adding output to states: N (nickel), B (Butterfinger), and S (Snickers).

Example: complete vending machine with output



Add transitions to cover all symbols for each state.

Summary

A finite state machine (FSM) consists of states and transitions.

States include the start state and final states.

Transitions map states and input symbols to states.

Also known as deterministic finite automata (DFAs).

An FSM recognizes a set of strings (language).

These are all strings that reach a final state from the start.

FSMs have many applications.

Regular expression matching, cache coherence protocols.

Verification (e.g., of OS kernels and devices).

Controllers (FSMs with output).