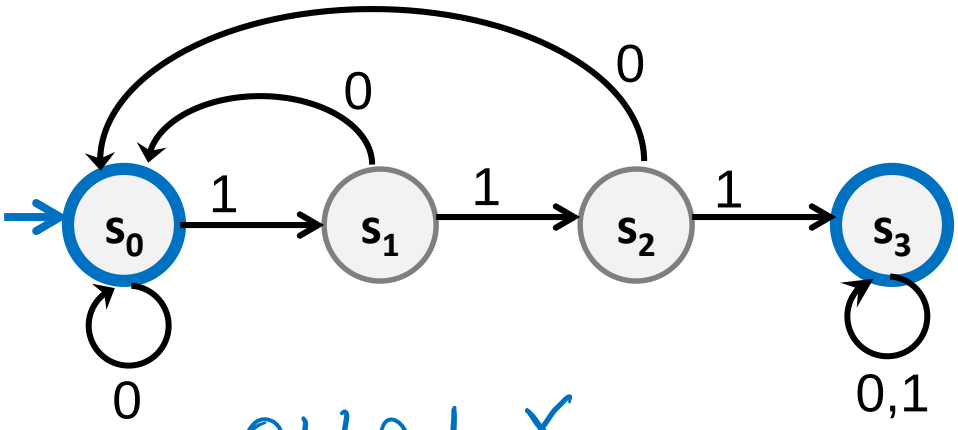
CSE 311: Foundations of Computing

Lecture 22: DFAs and Finite State Machines with Output

Finite State Machines

- States
- Transitions on input symbols
- Start state and final states
- The "language recognized" by the machine is the set of strings that reach a final state from the start

Old 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



01101 X

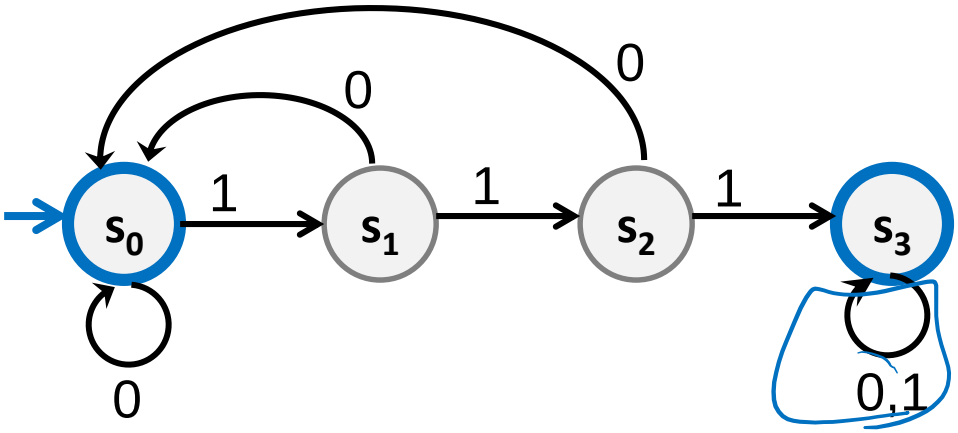
010 ✓

Finite State Machines

- Each machine designed for strings over some fixed alphabet Σ.

- Must have a transition defined from each state for ***every*** symbol in Σ.

Old 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



Applications of FSMs (a.k.a. Finite Automata)

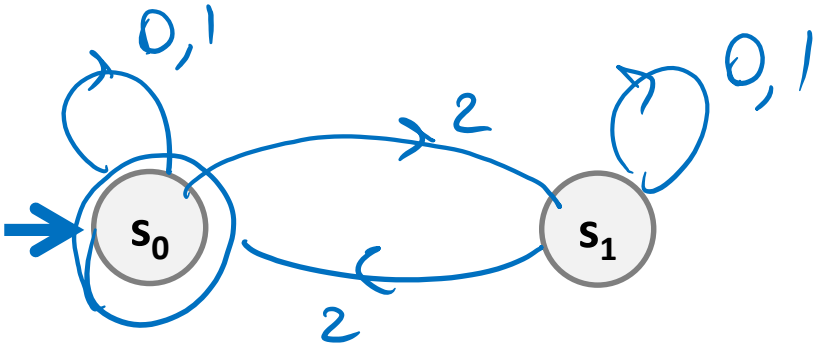
- Implementation of regular expression matching in programs like `grep`
- Control structures for sequential logic in digital circuits
- Algorithms for communication and cache-coherence protocols
  - Each agent runs its own FSM
- Design specifications for reactive systems
  - Components are communicating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

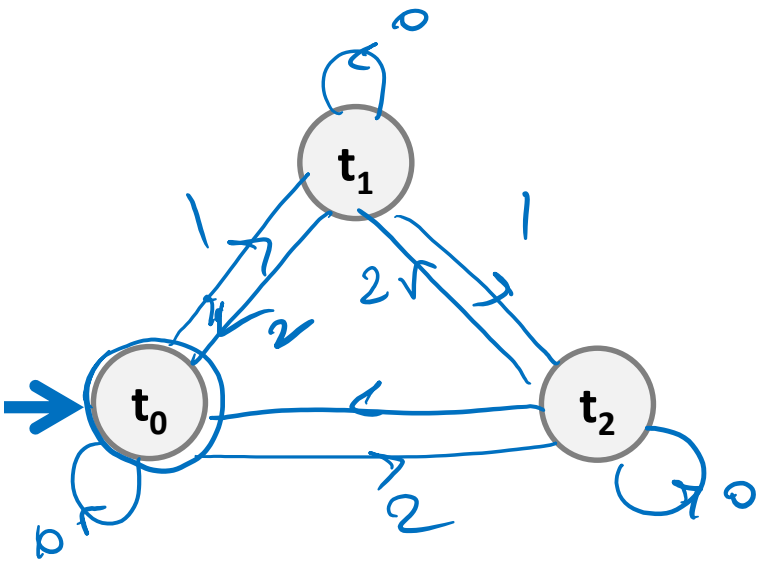
Strings over {0, 1, 2}

M_1: Strings with an even number of 2's



0212 ✓
021 X

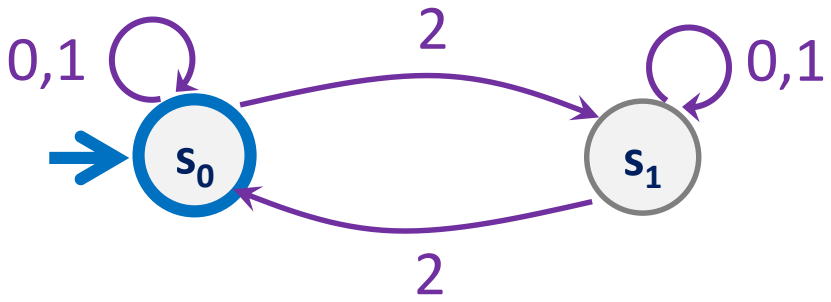
M_2: Strings where the sum of digits mod 3 is 0



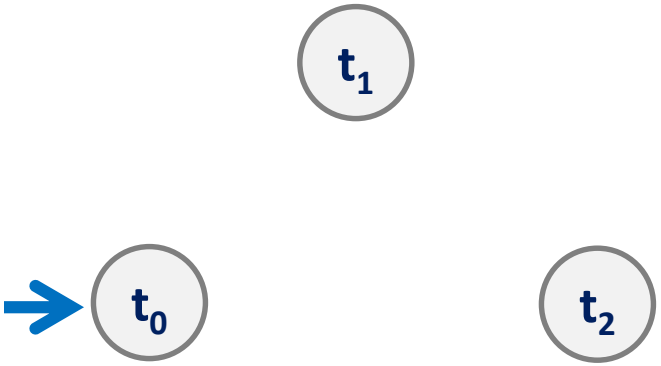
0212 X
021 ✓

Strings over {0, 1, 2}

M_1: Strings with an even number of 2's



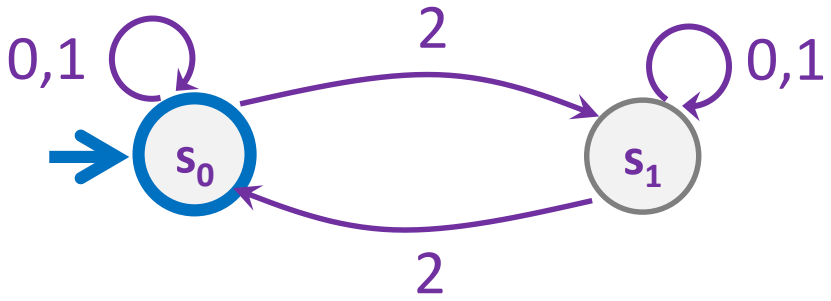
M_2: Strings where the sum of digits mod 3 is 0



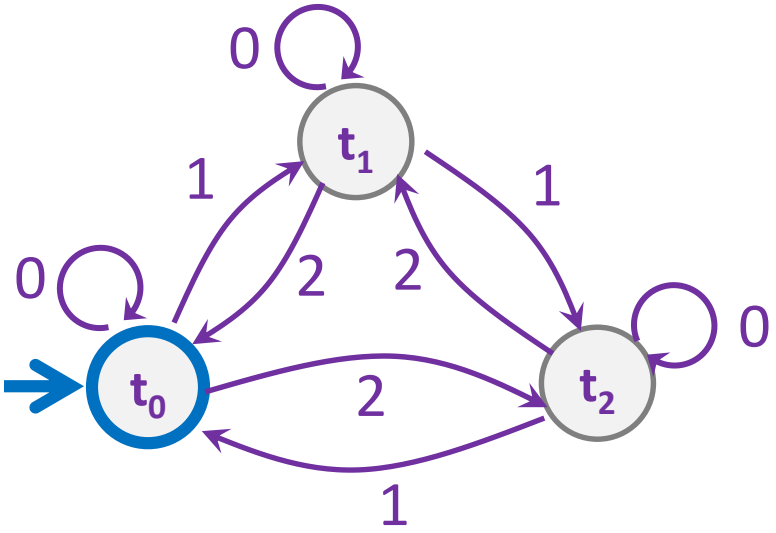
Strings over {0, 1, 2}

DFA

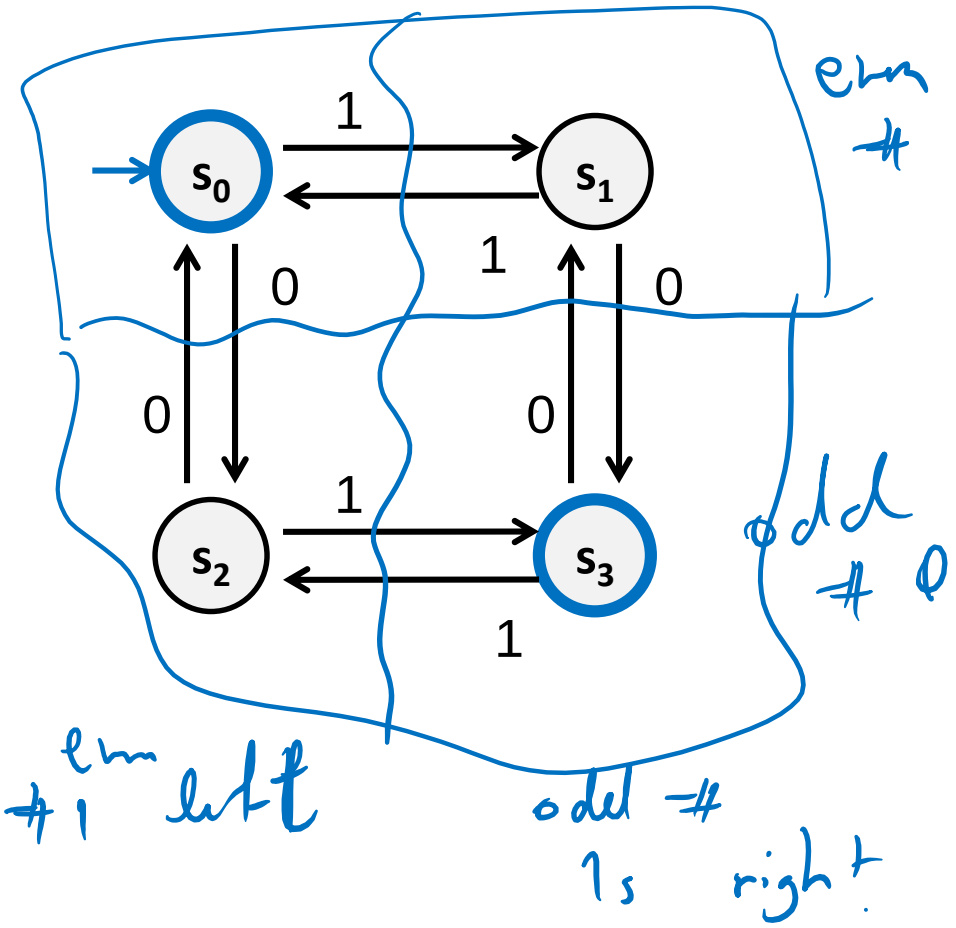
M_1: Strings with an even number of 2's



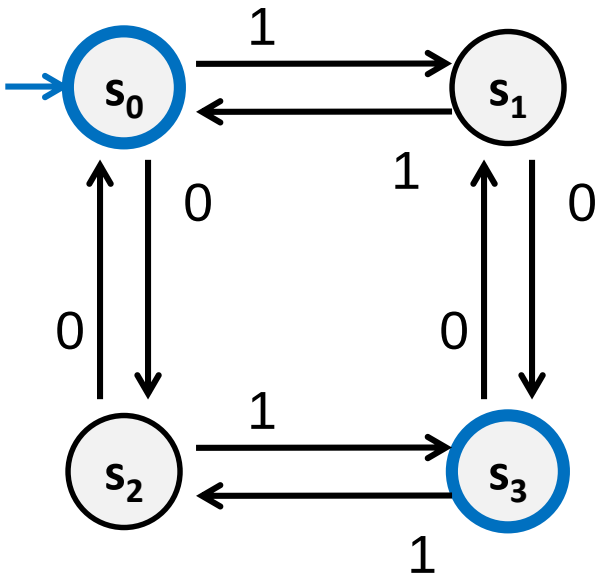
M_2: Strings where the sum of digits mod 3 is 0

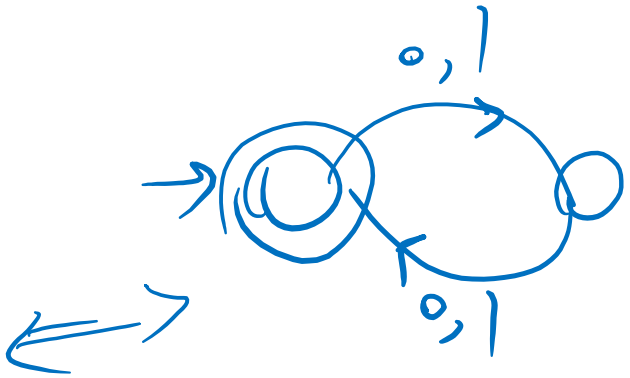


What language does this machine recognize?



What language does this machine recognize?

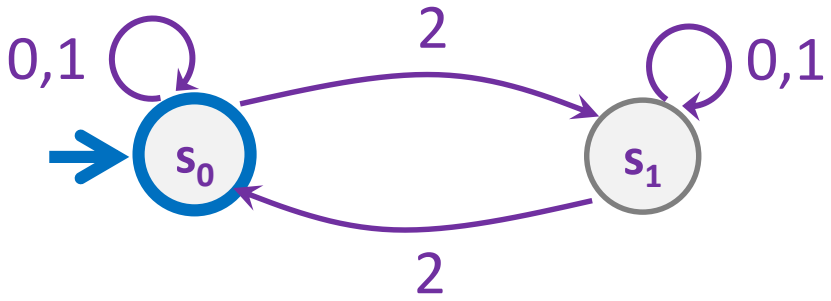




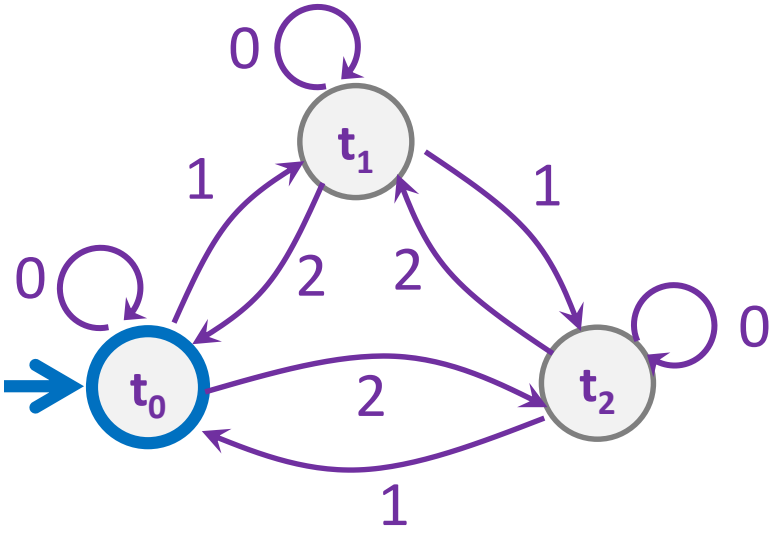
The set of all binary strings with # of 1's $\equiv$ # of 0's (mod 2) (both are even or both are odd).

Strings over {0, 1, 2}

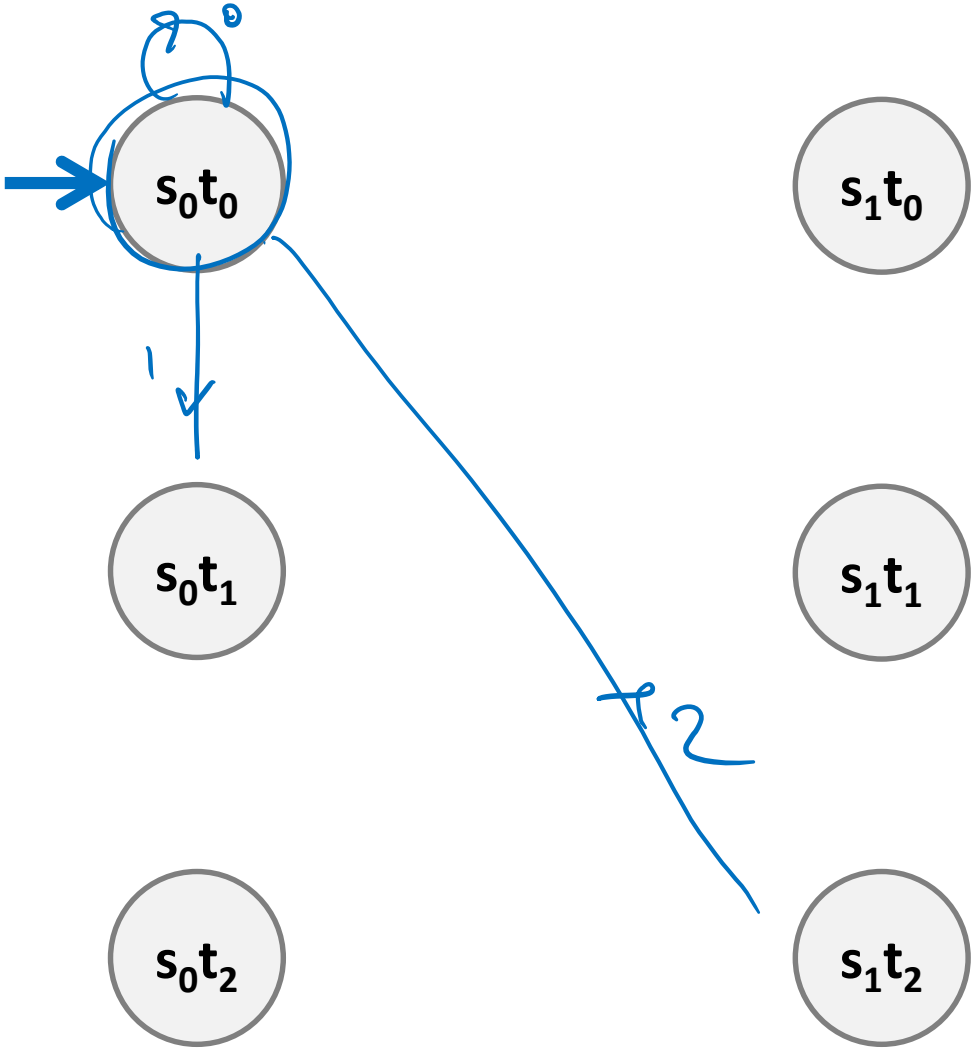
M_1: Strings with an even number of 2's



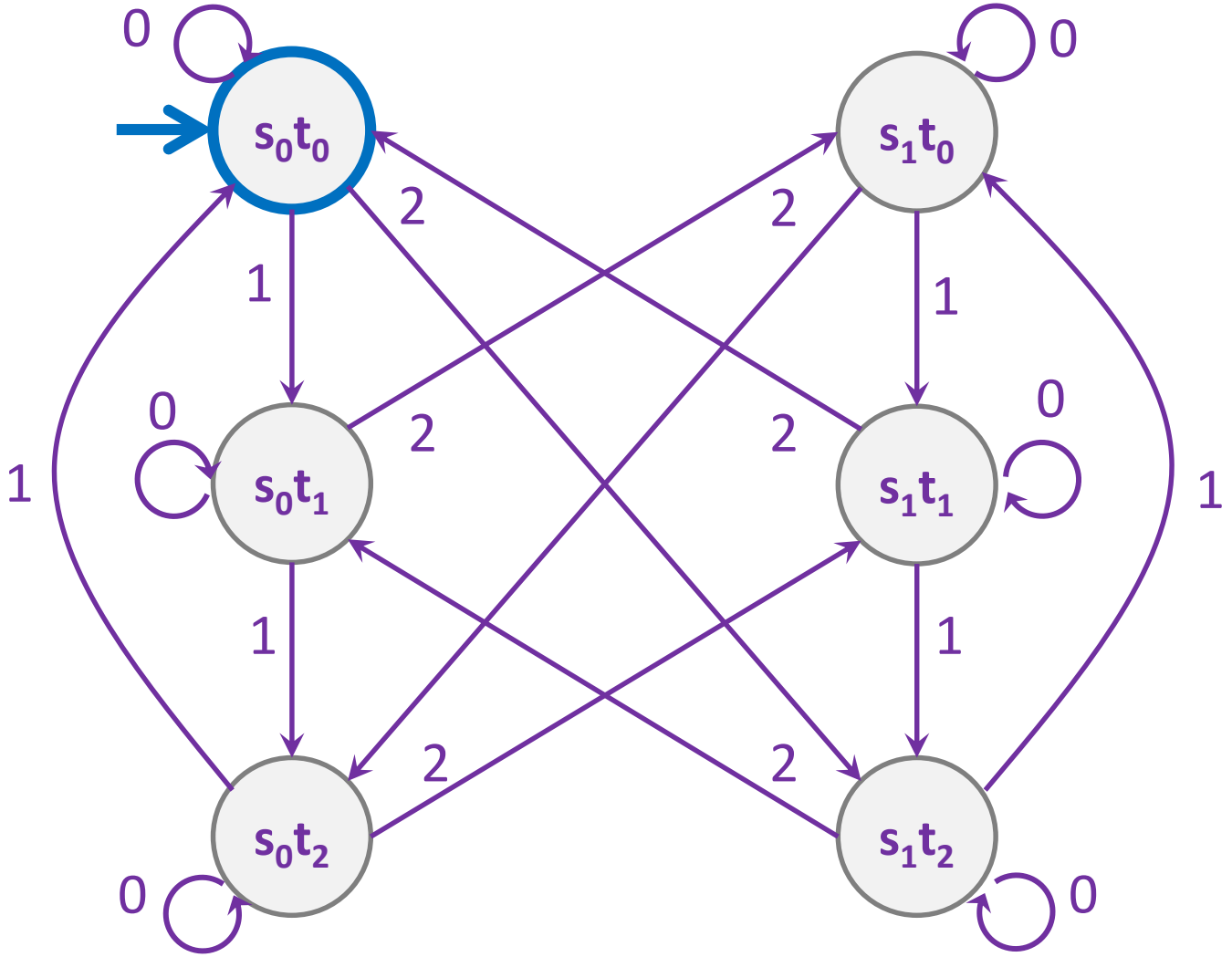
M_2: Strings where the sum of digits mod 3 is 0



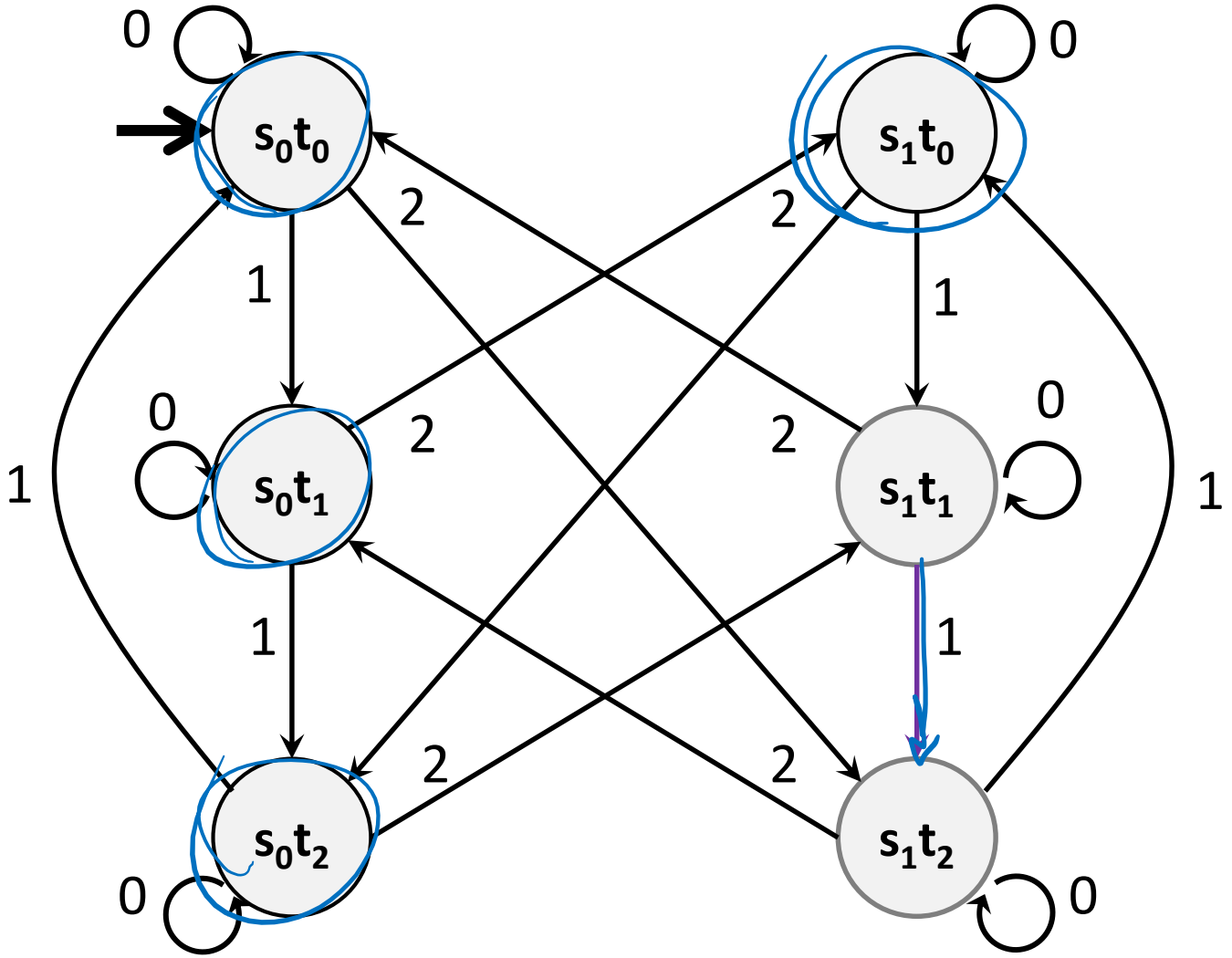
Strings over {0,1,2} w/ even number of 2's and mod 3 sum 0



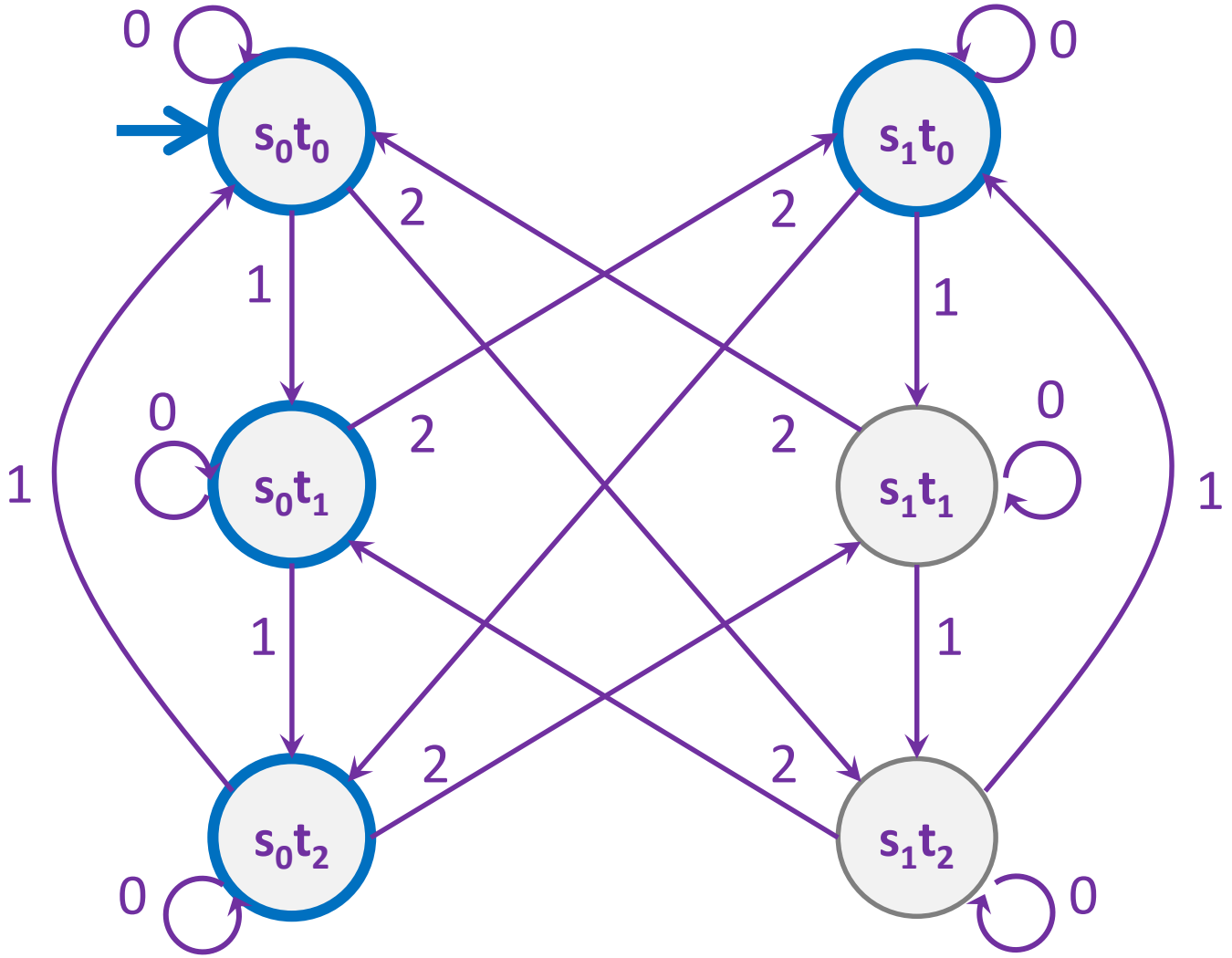
Strings over {0,1,2} w/ even number of 2's and mod 3 sum 0



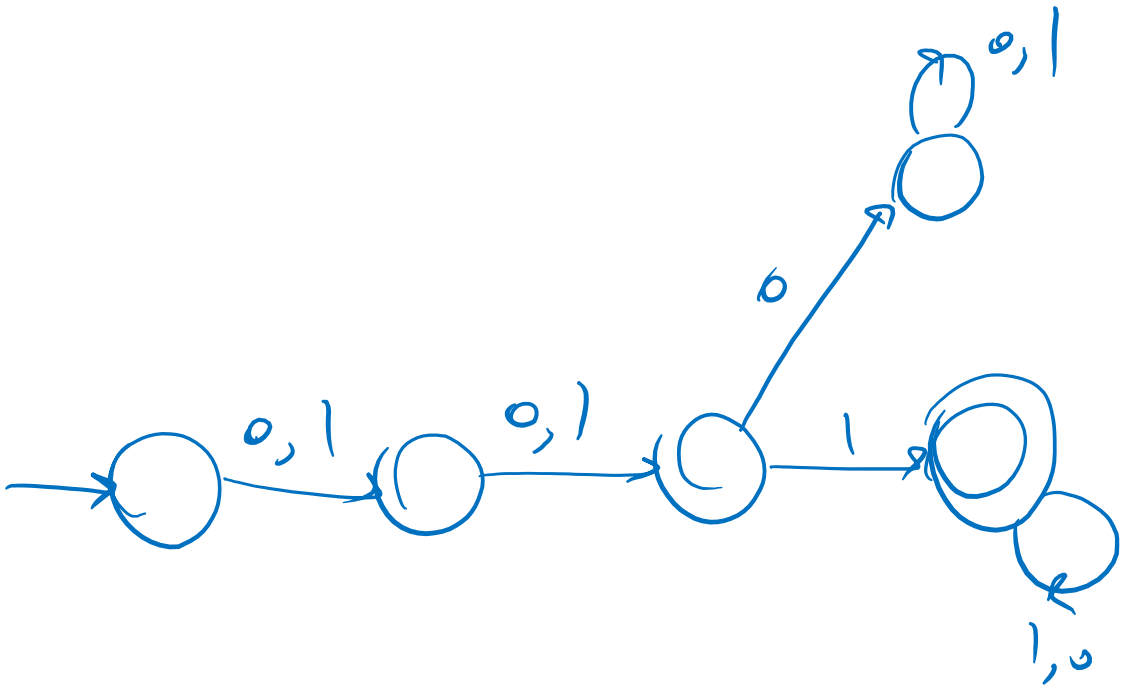
Strings over {0,1,2} w/ even number of 2's OR mod 3 sum 0?



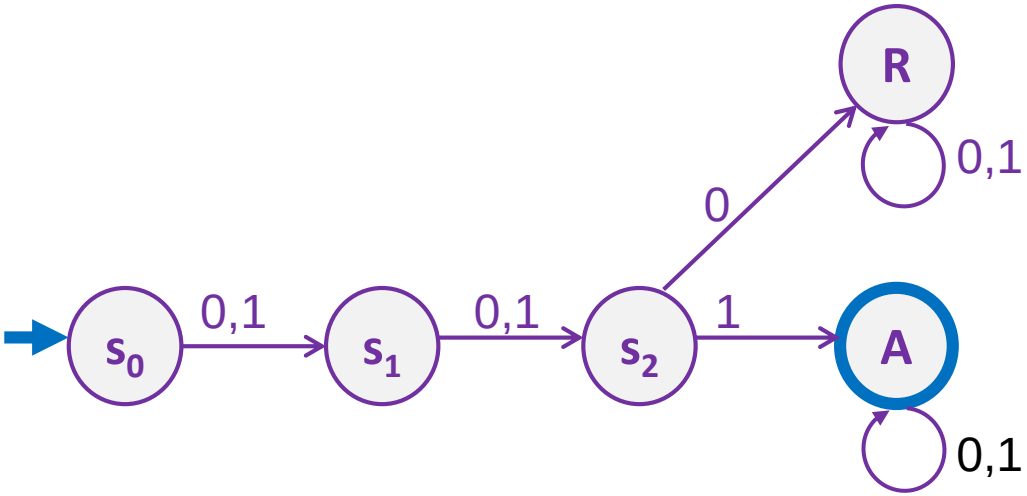
Strings over {0,1,2} w/ even number of 2's OR mod 3 sum 0



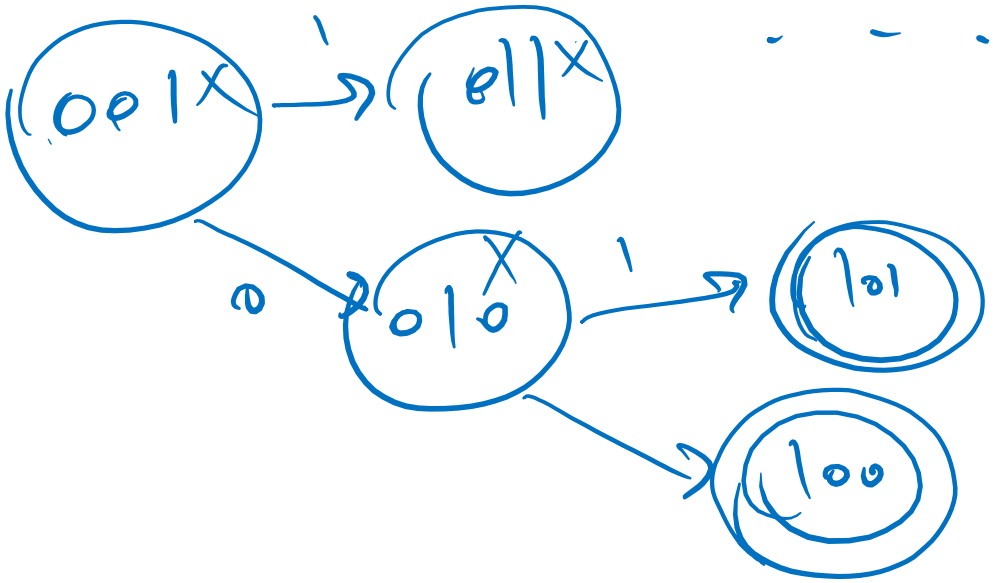
The set of binary strings with a 1 in the 3rd position from the start



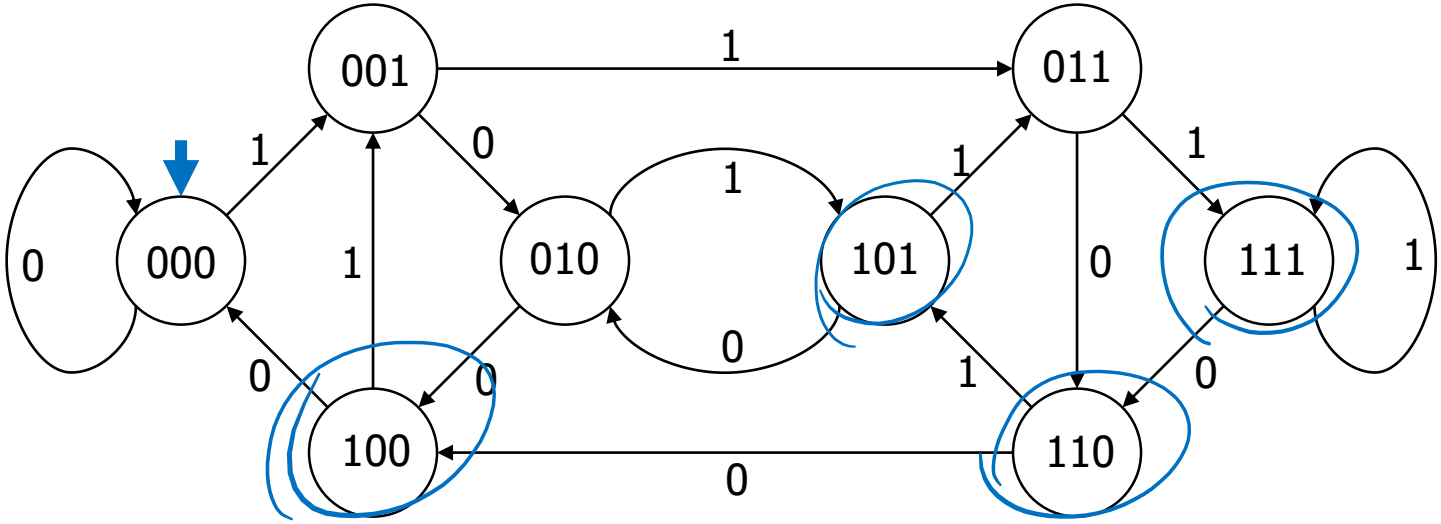
The set of binary strings with a 1 in the 3rd position from the start



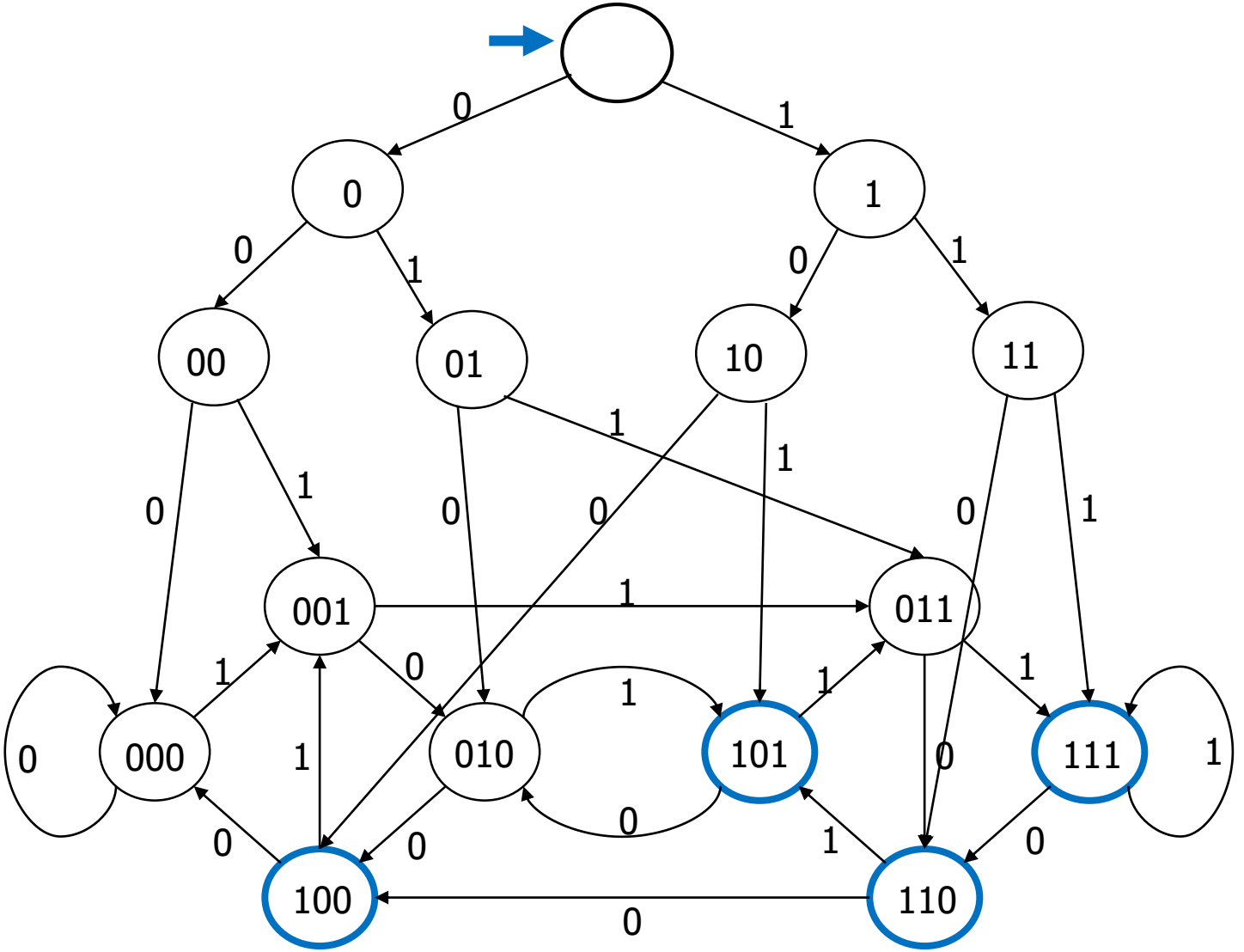
The set of binary strings with a 1 in the 3rd position from the end



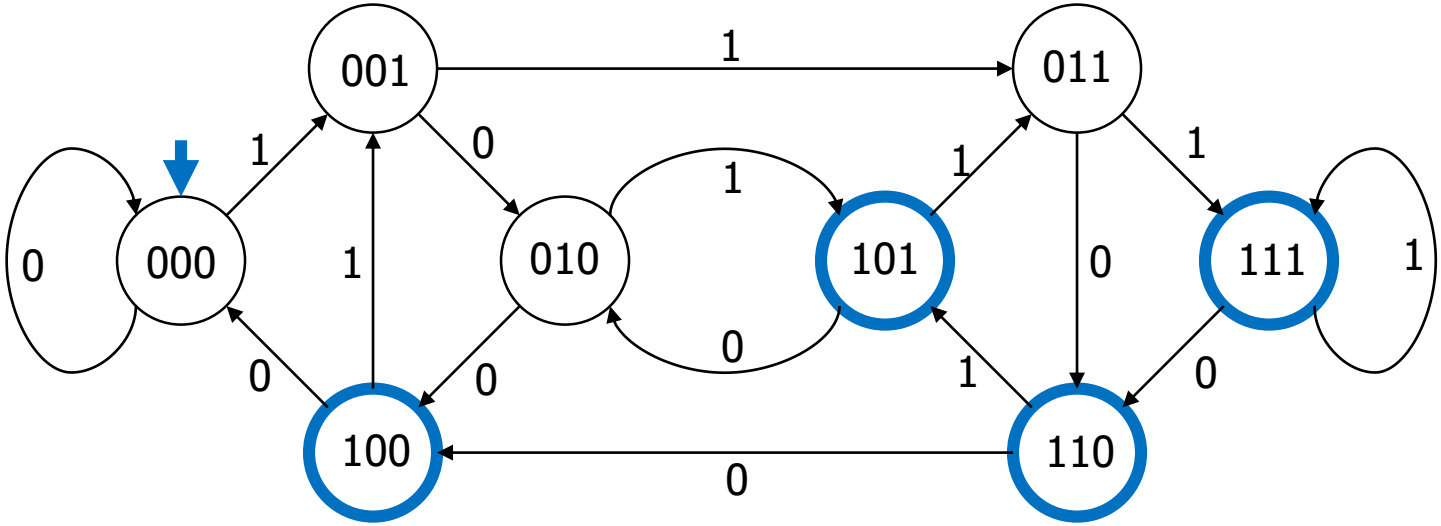
3 bit shift register “Remember the last three bits”



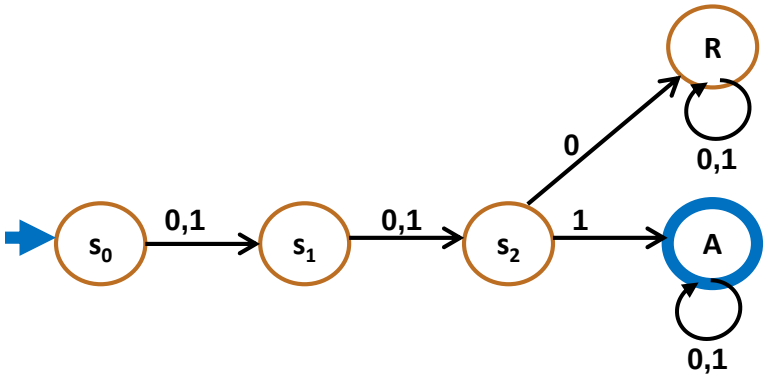
The set of binary strings with a 1 in the 3rd position from the end

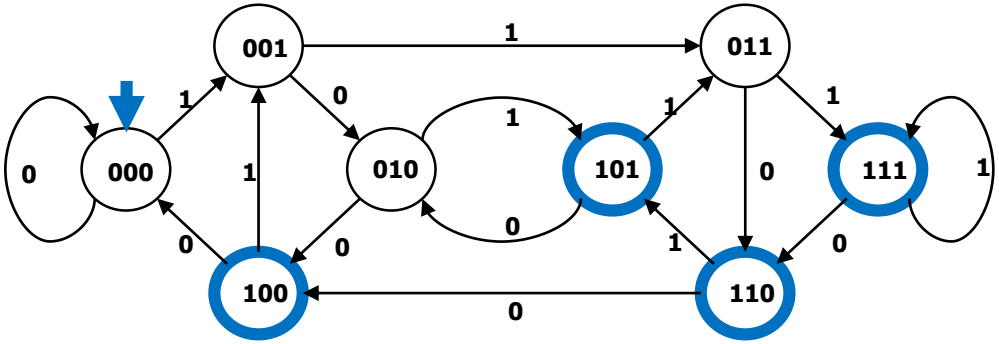


The set of binary strings with a 1 in the 3rd position from the end



The beginning versus the end





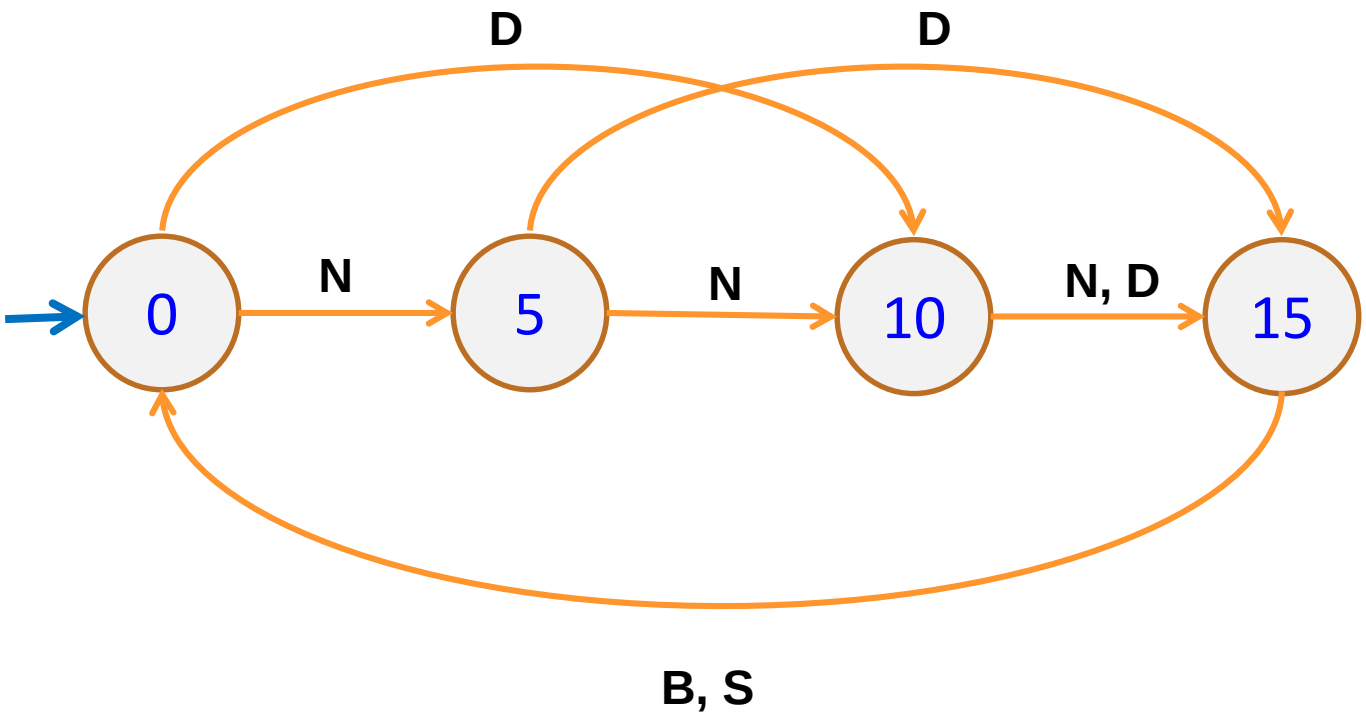
Adding Output to Finite State Machines

- So far we have considered finite state machines that just accept/reject strings
  - called “Deterministic Finite Automata” or DFAs
- Now we consider finite state machines that with output
  - These are the kinds used as controllers

Vending Machine

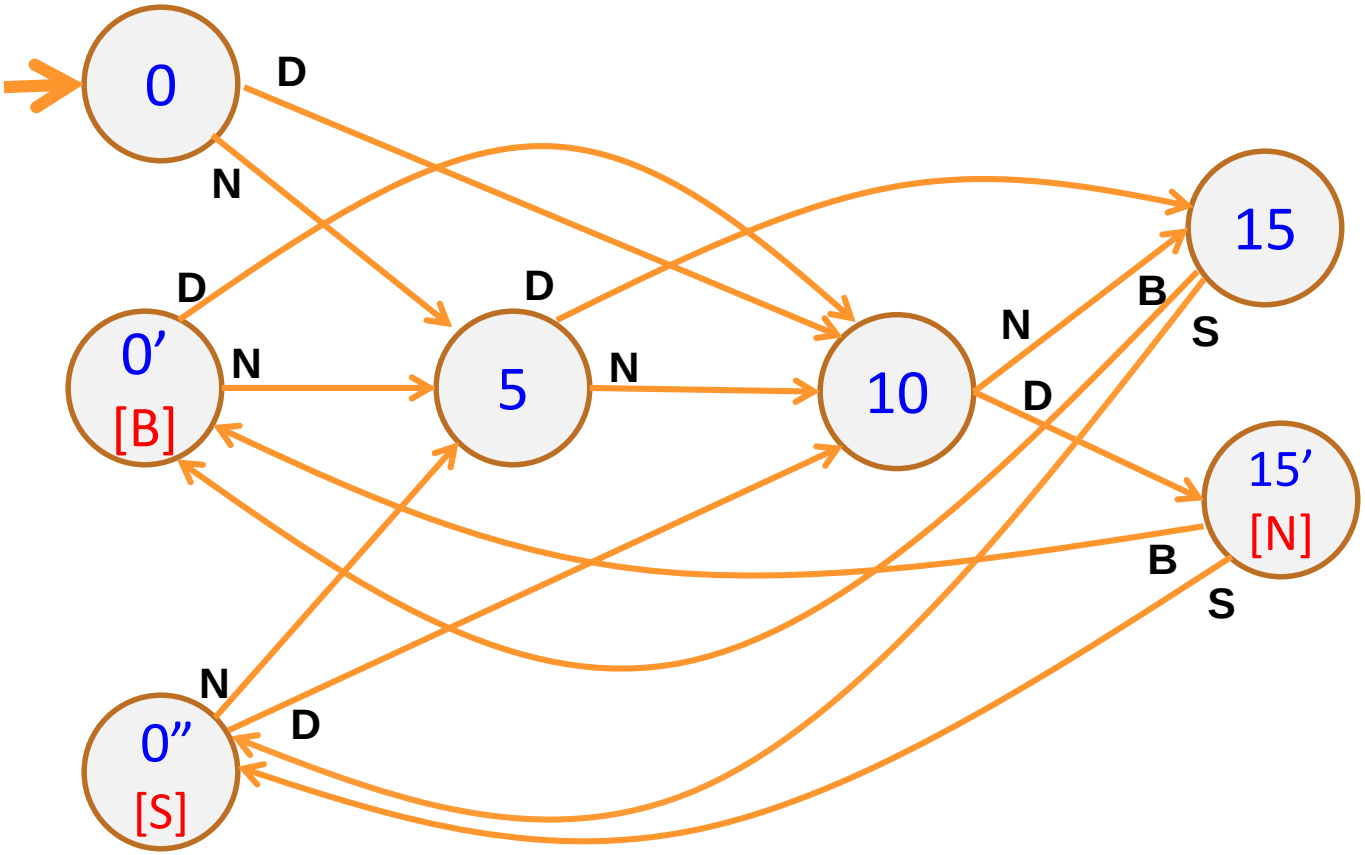
Enter 15 cents in dimes or nickels
Press S or B for a candy bar

Vending Machine, v0.1



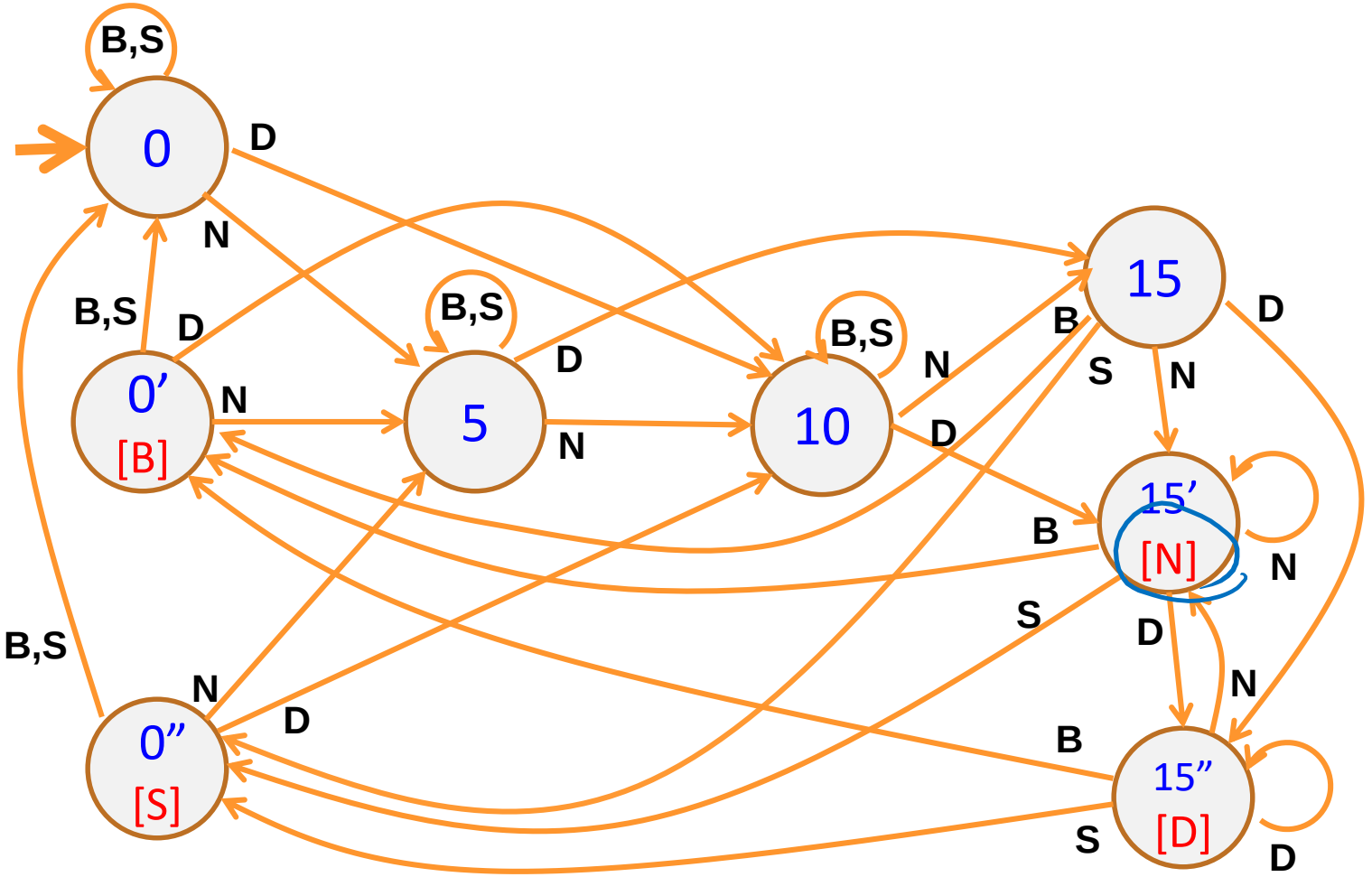
Basic transitions on **N** (nickel), **D** (dime), **B** (butterfinger), **S** (snickers)

Vending Machine, v0.2



Adding output to states: N – Nickel, S – Snickers, B – Butterfinger

Vending Machine, v1.0



Adding additional “unexpected” transitions to cover all symbols for each state