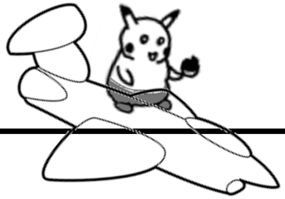


cse 311: foundations of computing

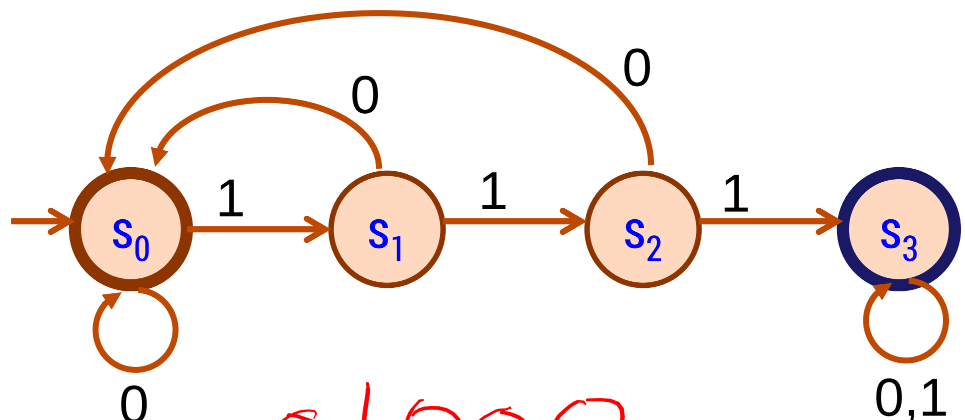
Spring 2015

Lecture 22: Finite state machines

- States
- Transitions on inputs
- Start state and final states
- The language recognized by a machine is the set of strings that reach a final state

010111

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



01000

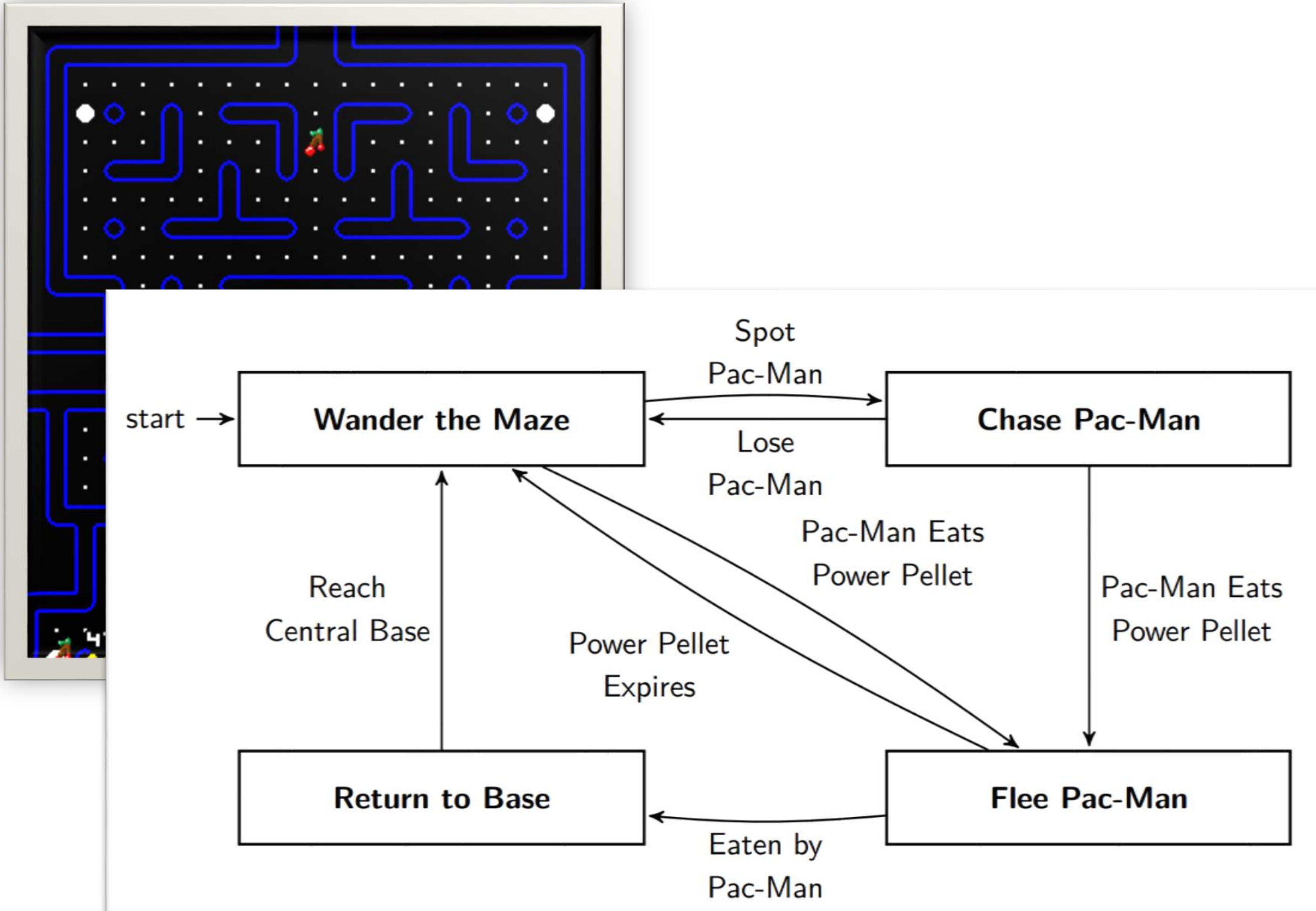
applications of FSMs (aka finite automata)

- 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 (aka finite automata)

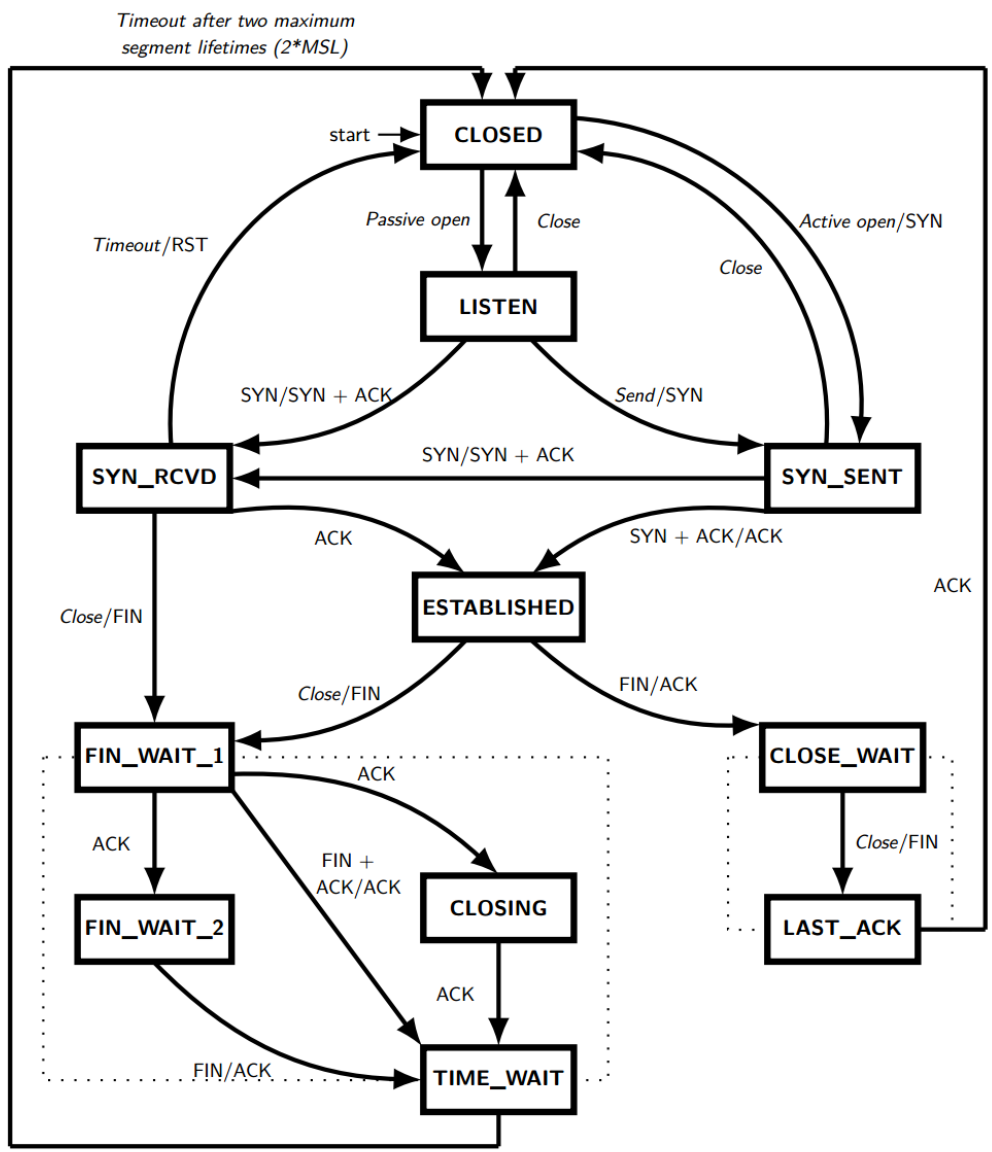
- Formal verification of systems
  - Is an unsafe state reachable?
- Computer games
  - FSMs provide worlds to explore
  - Character AI
- Minimization algorithms for FSMs can be extended to more general models used in
  - Text prediction
  - Speech recognition

waka waka



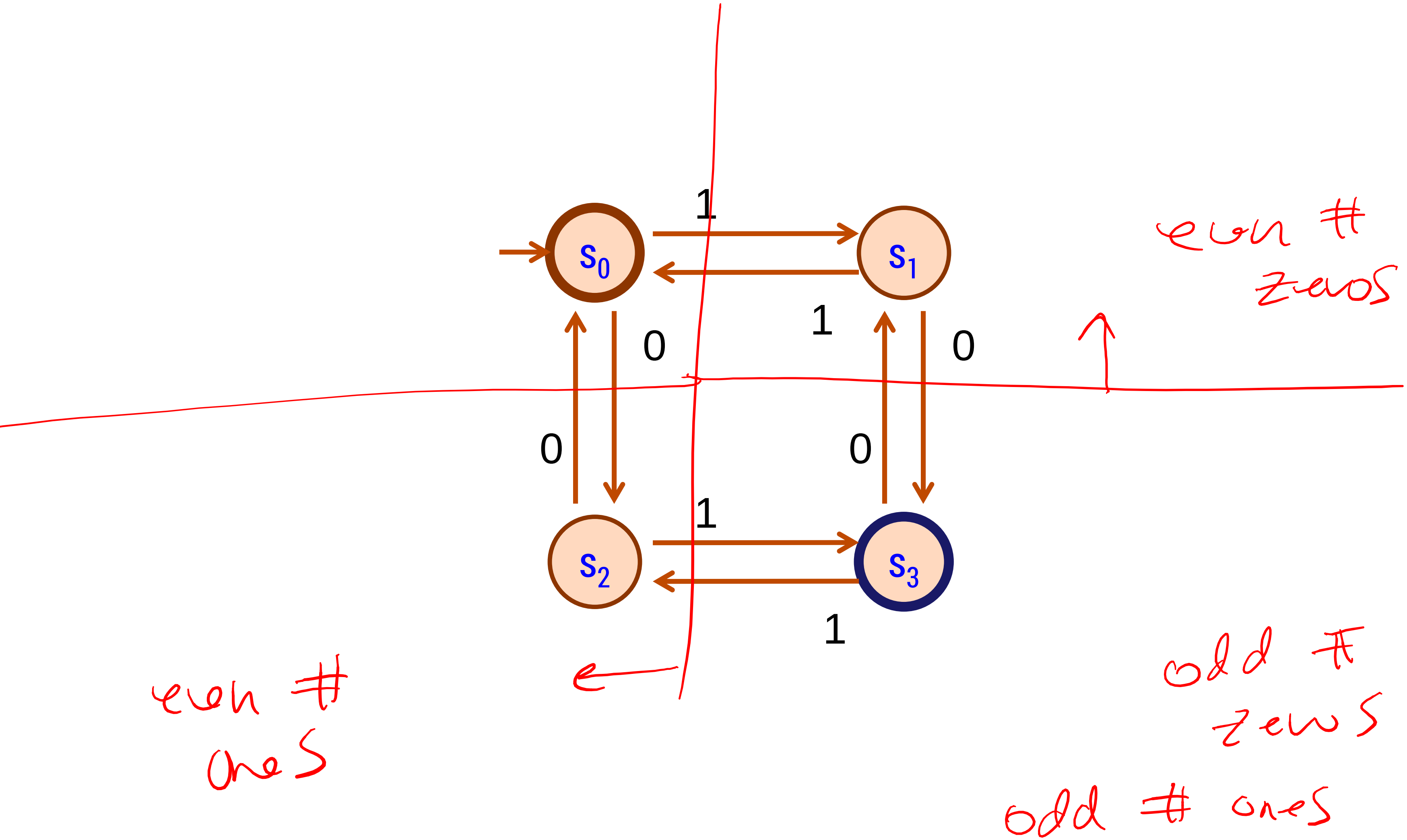
start
Wander the Maze
Spot Pac-Man
Chase Pac-Man
Lose Pac-Man
Pac-Man Eats Power Pellet
Reach Central Base
Power Pellet Expires
Pac-Man Eats Power Pellet
Return to Base
Flee Pac-Man
Eaten by Pac-Man

tcp



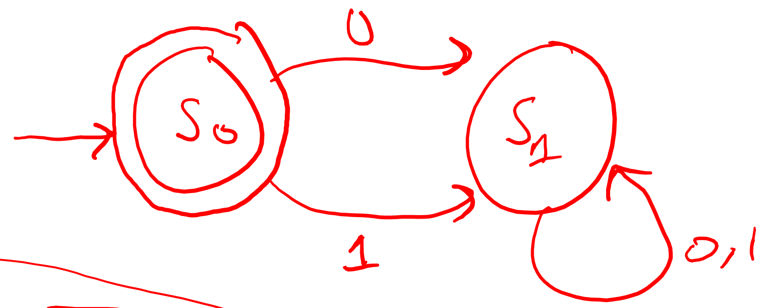
Timeout after two maximum segment lifetimes (2*MSL)
start
CLOSED
Passive open
Close
Timeout/RST
Active open/SYN
Close
LISTEN
SYN/SYN + ACK
Send/SYN
SYN_RCVD
SYN/SYN + ACK
SYN_SENT
ACK
SYN + ACK/ACK
ESTABLISHED
ACK
Close/FIN
Close/FIN
FIN/ACK
FIN_WAIT_1
CLOSE_WAIT
ACK
ACK
FIN + ACK/ACK
Close/FIN
FIN_WAIT_2
CLOSING
LAST_ACK
ACK
FIN/ACK
TIME_WAIT

what language does this machine recognize?

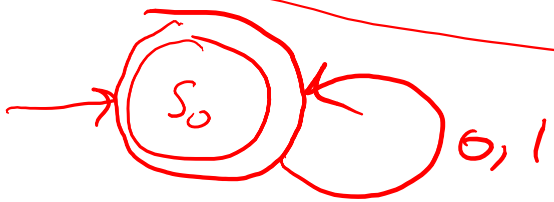


can we recognize these languages with DFAs?

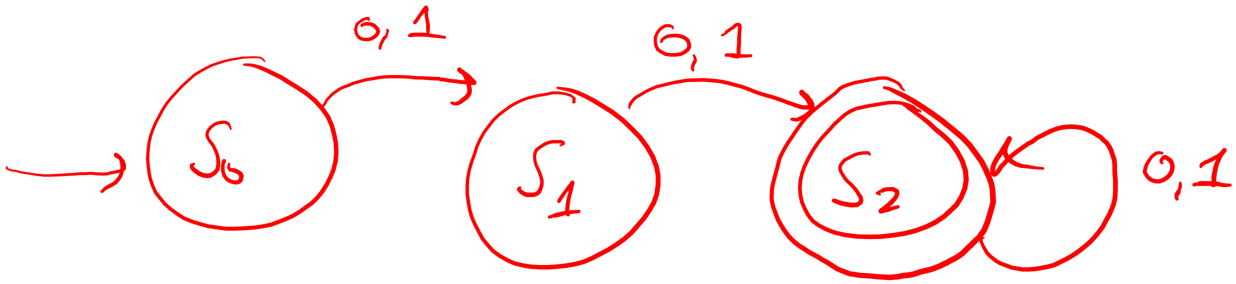
- $\emptyset$



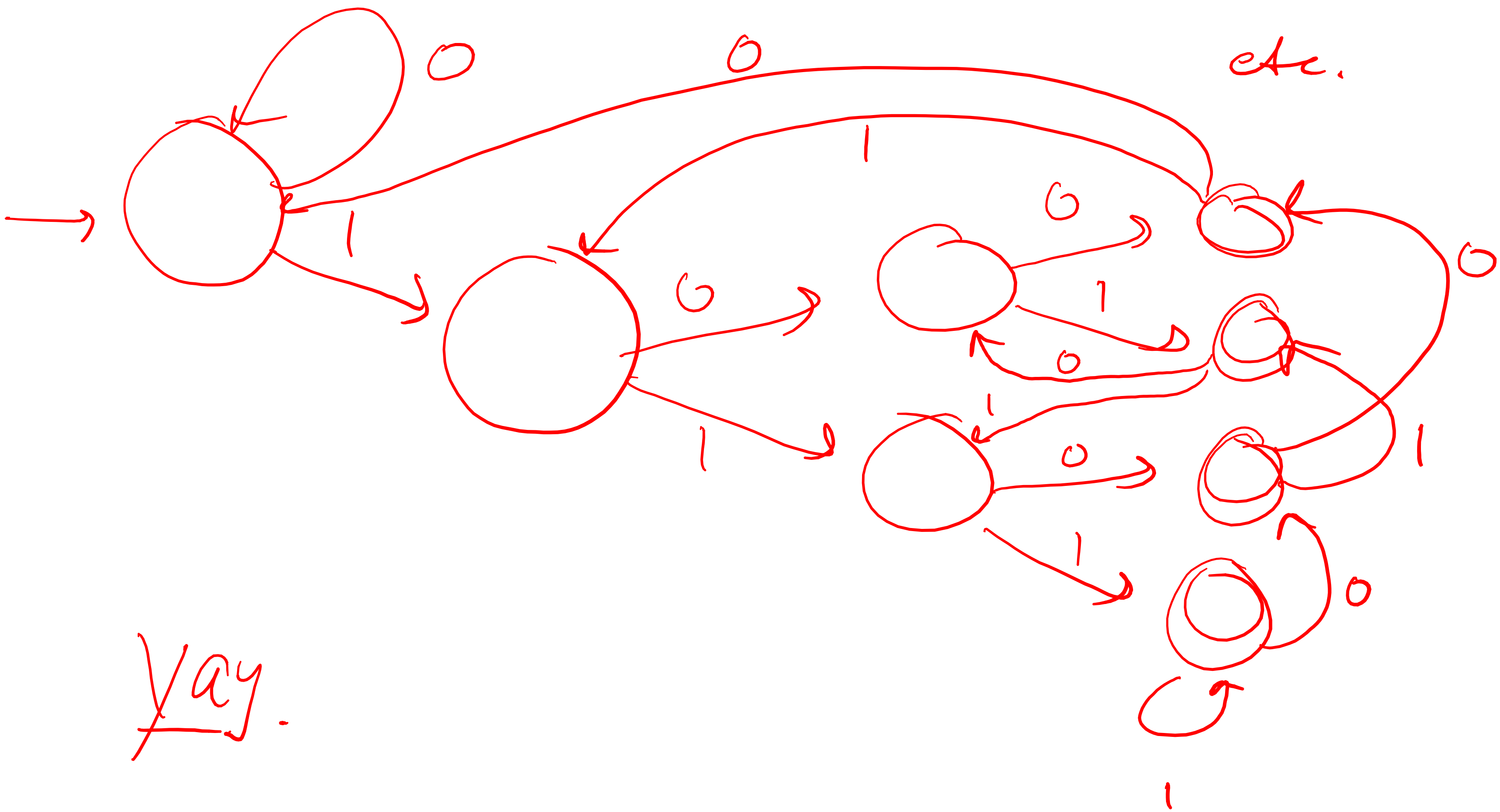
- Σ^*



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

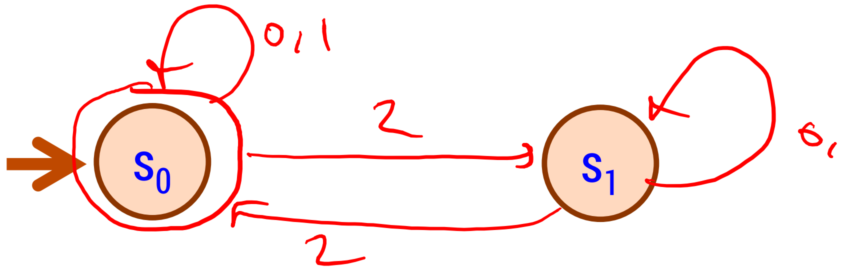


FSM that accepts binary strings with a 1 three positions from the end

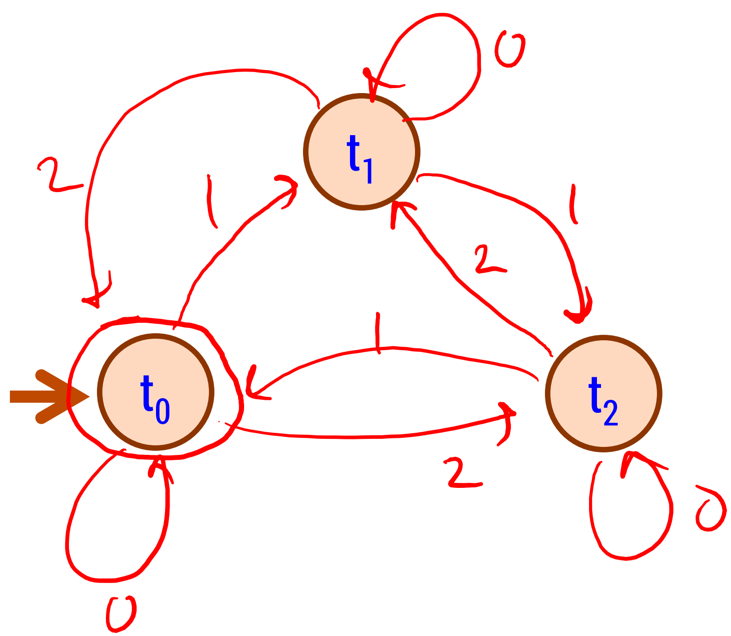


strings over {0, 1, 2}*

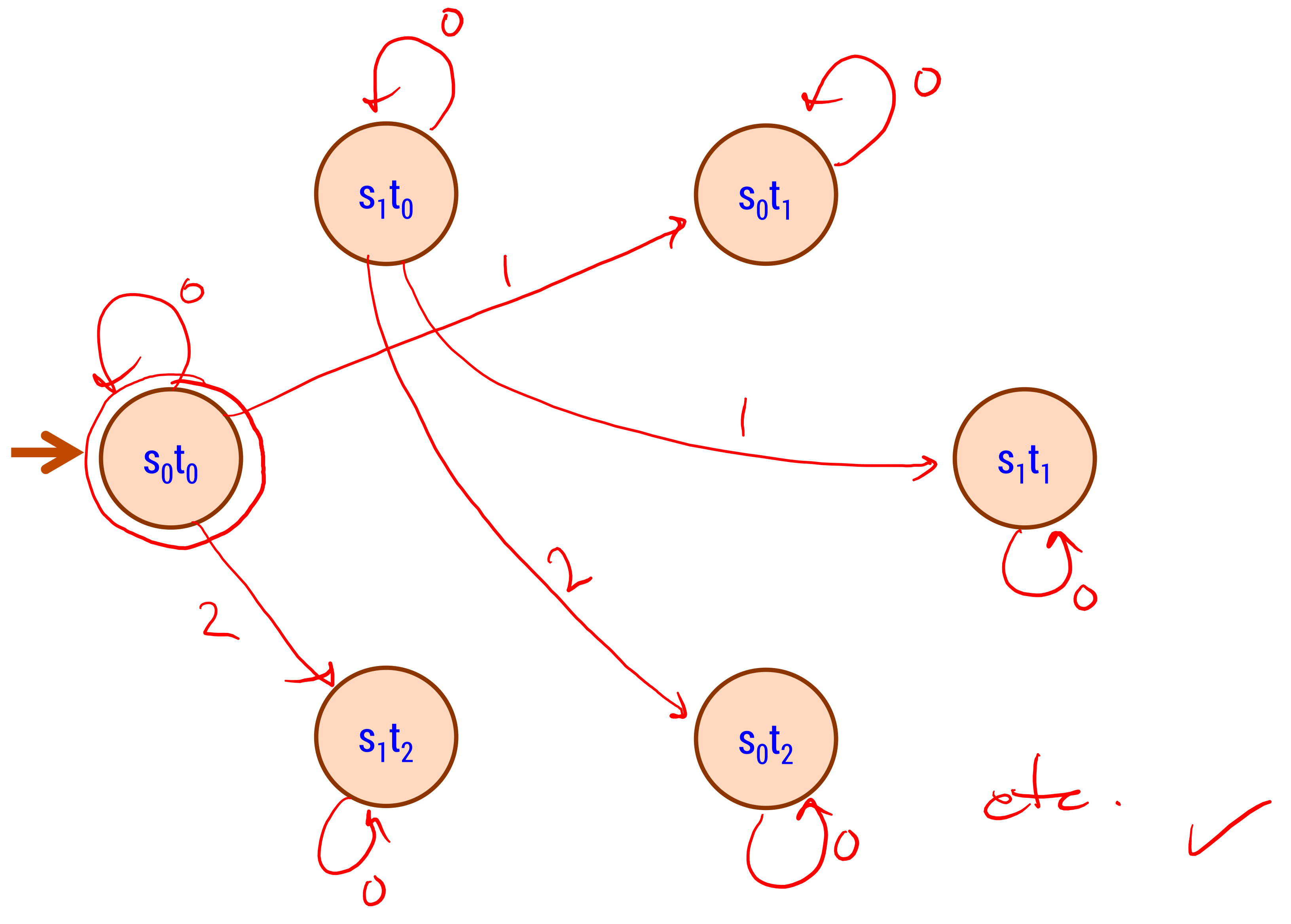
M_1: Strings with an even number of 2's



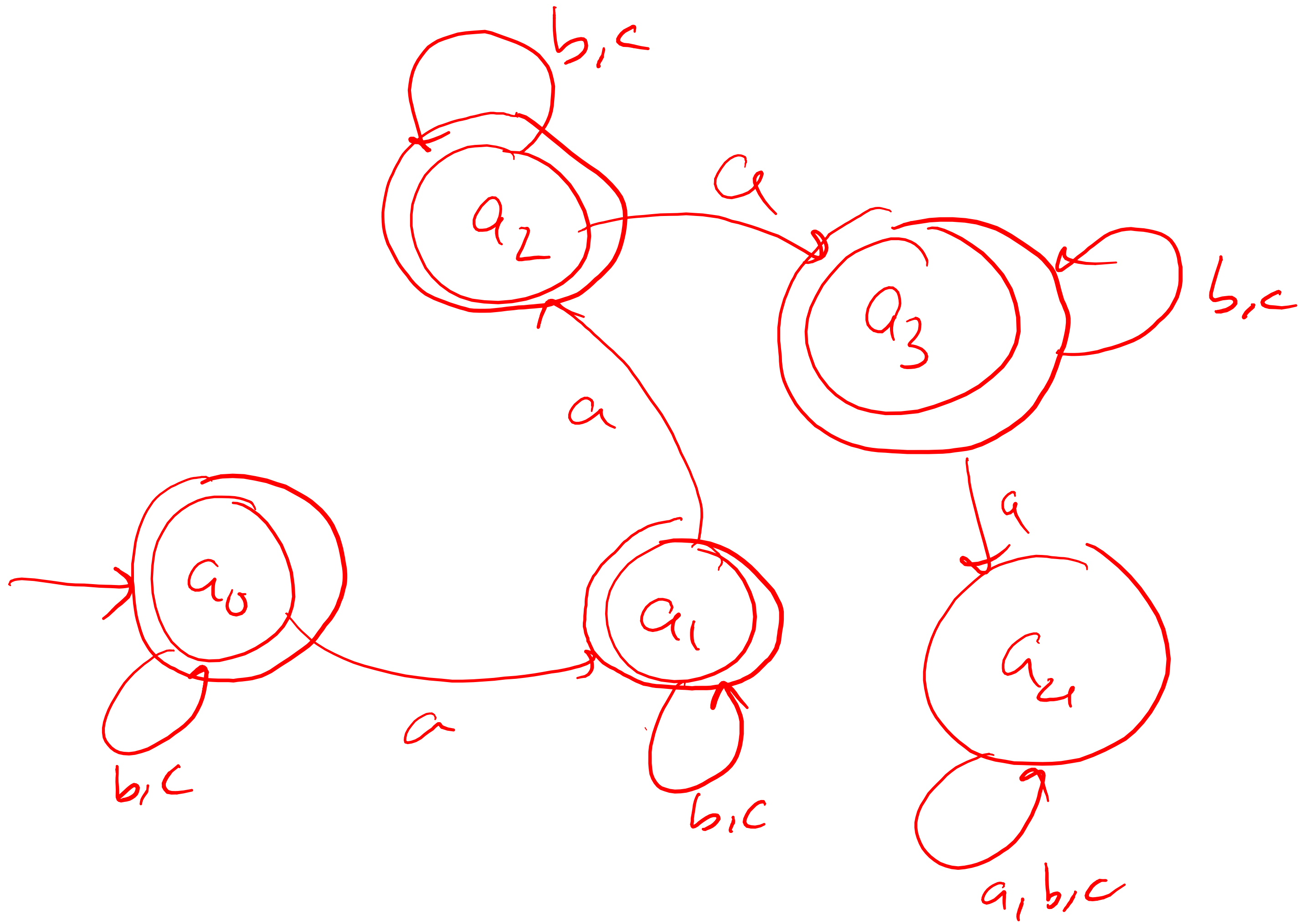
M_2: Strings where the sum of digits mod 3 is 0



both: even number of 2's and sum mod 3 = 0



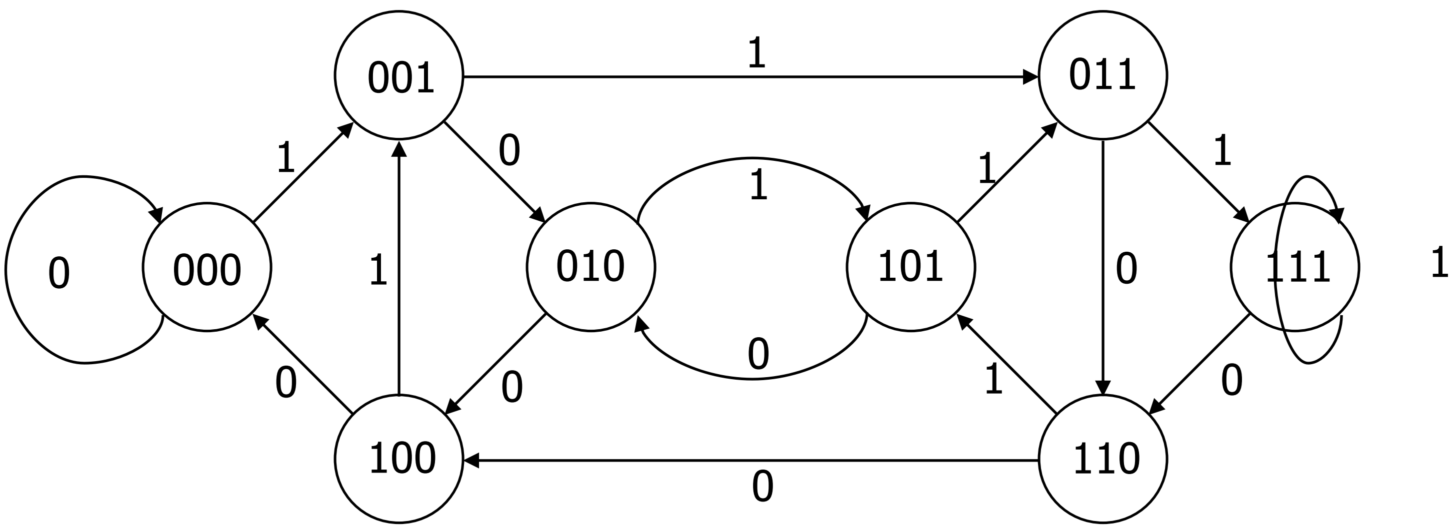
DFA that accepts strings of a's, b's, c's with no more than 3 a's

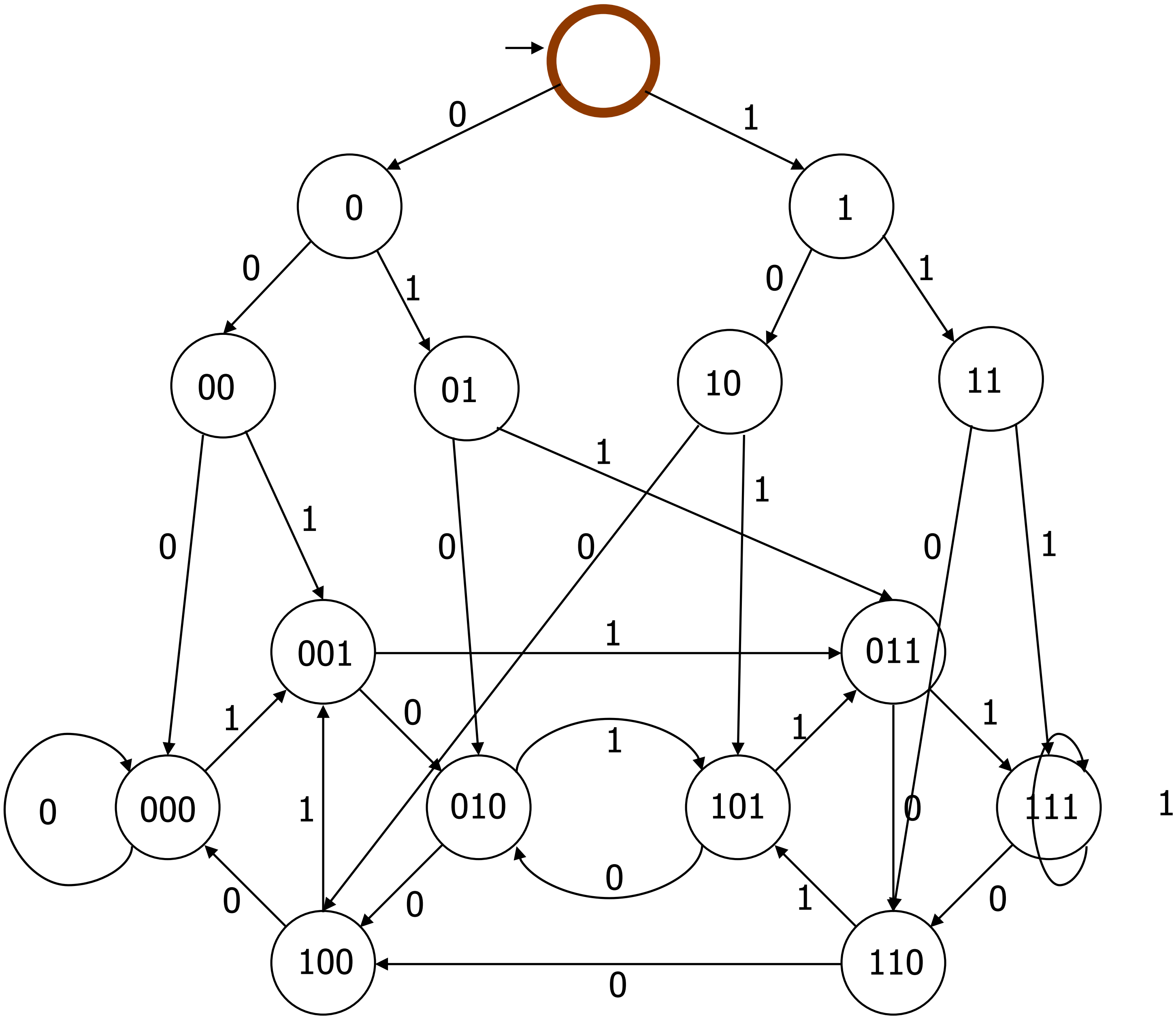


“Remember the last three bits”

3 bit shift register

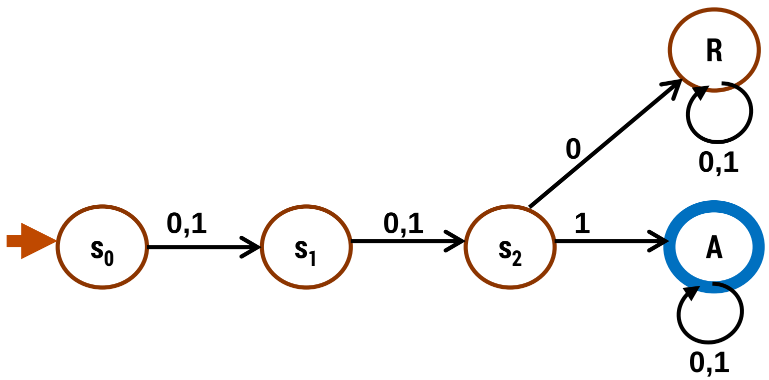
$000 \xrightarrow{1} 001 \xrightarrow{0} 010 \to \ldots$

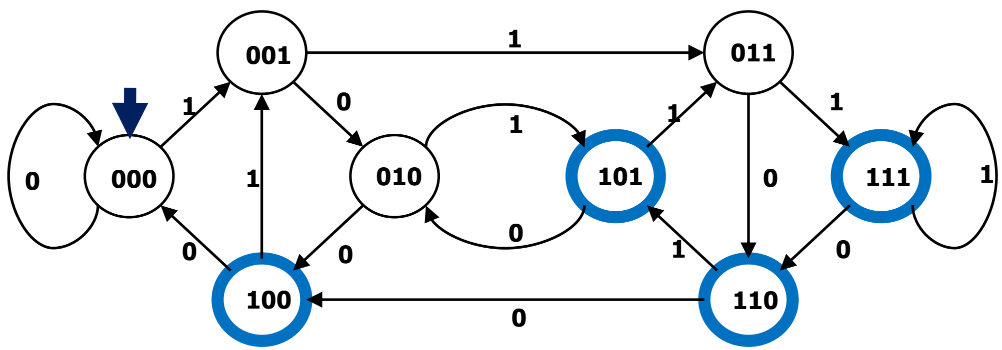




0
1
00
01
10
11
000
001
010
011
100
101
110
111

start and accept states

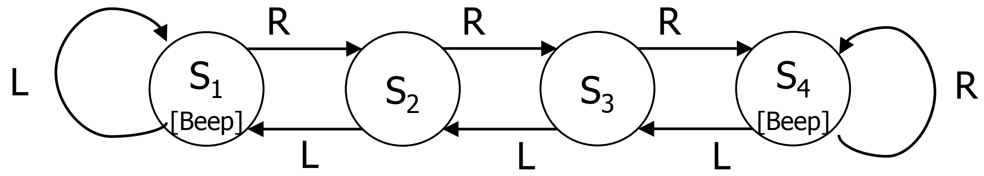




FSMs with output

"Tug-of-war"

	Input		Output
State	L	R	
s_1	s_1	s_2	Beep
s_2	s_1	s_3	
s_3	s_2	s_4	
s_4	s_3	s_4	Beep



vending machine

We're only making $5.50/hour writing regular expressions.

Let's design a vending machine.

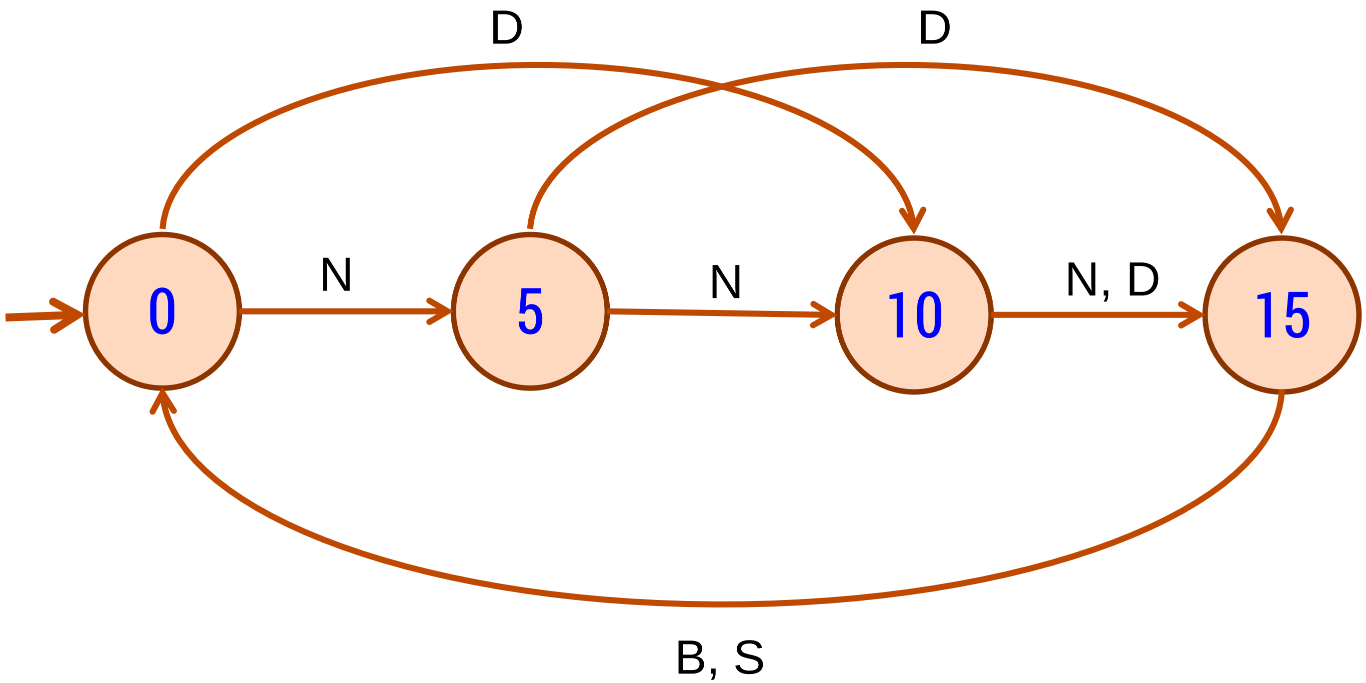
"He does not think like normal people, and as a result his tests are quite difficult. His lectures are amusing and get the material across, but his office hours are not always too helpful. **Beware the vending machine final.**"

Vending spec:

Enter 15 cents in dimes or nickels

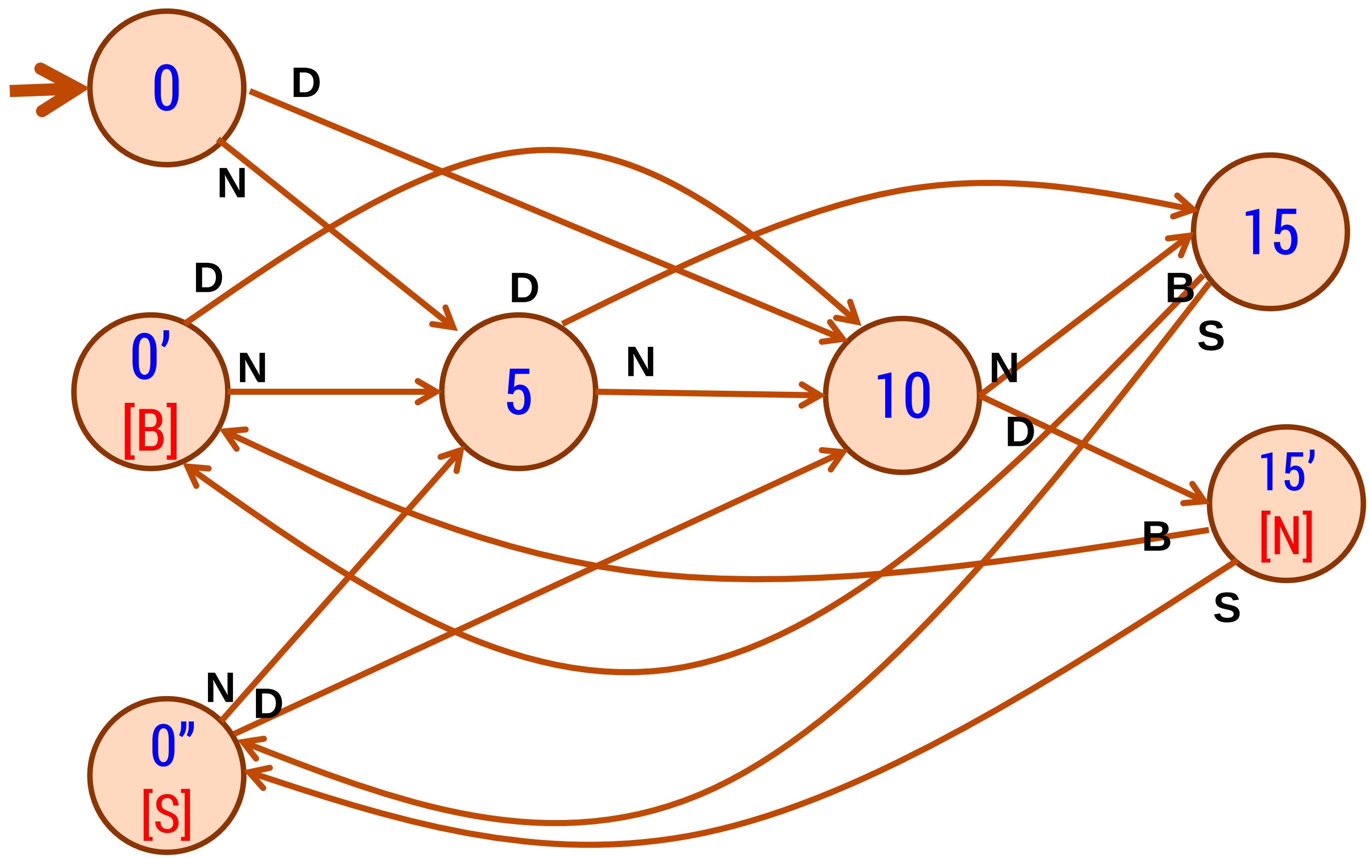
Press **S** or **B** for a candy bar

vending machine v0.1



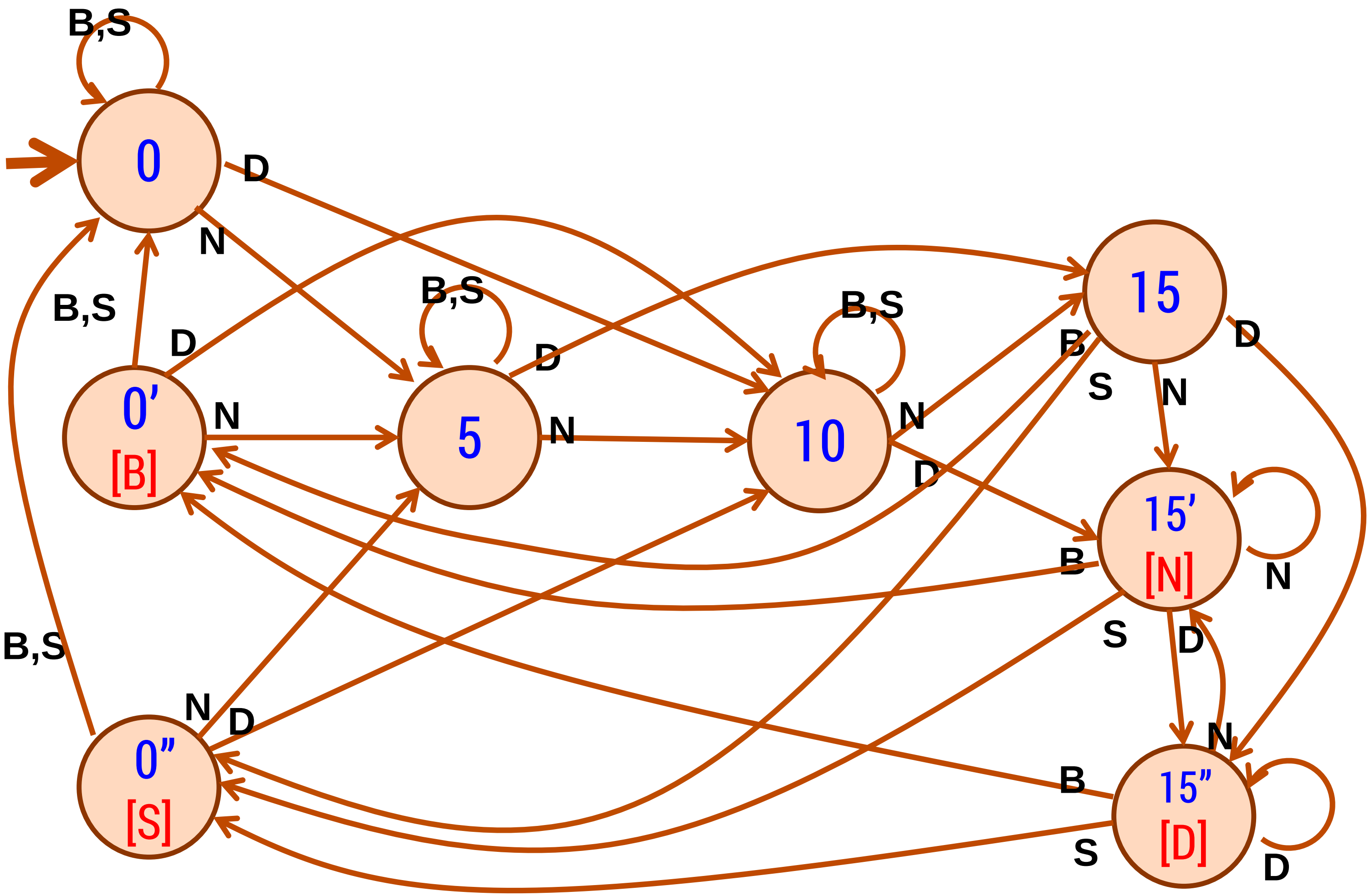
Basic transitions on N (nickel), D (dime), B (butterfinger), S (snickers)

vending machine v0.2



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

vending machine v1.0



Adding additional “unexpected” transitions