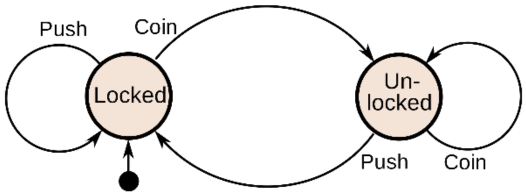
STATE MACHINES

Roberto Martin-Martin

Assistant Professor of Computer Science

What will you learn today?

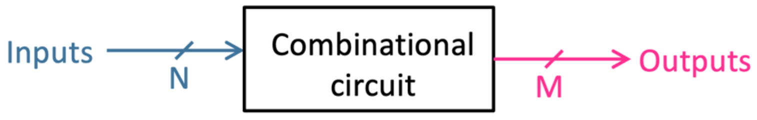
- What is a state machine?
  - Connection to combinatorial logic
  - When do we use state machines in robotics?
- Types of Finite Machines
- Deterministic Finite Automation (DFA)
- Tools for designing/implementing FSMs
- Other types of machines
  - Petri Nets
  - Hybrid Automata

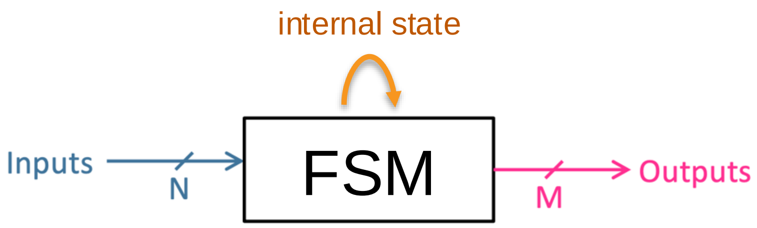


Combinational Logic vs. State Machines

- Previous: combinational logic
  - input → output
  - nothing depends on the past!

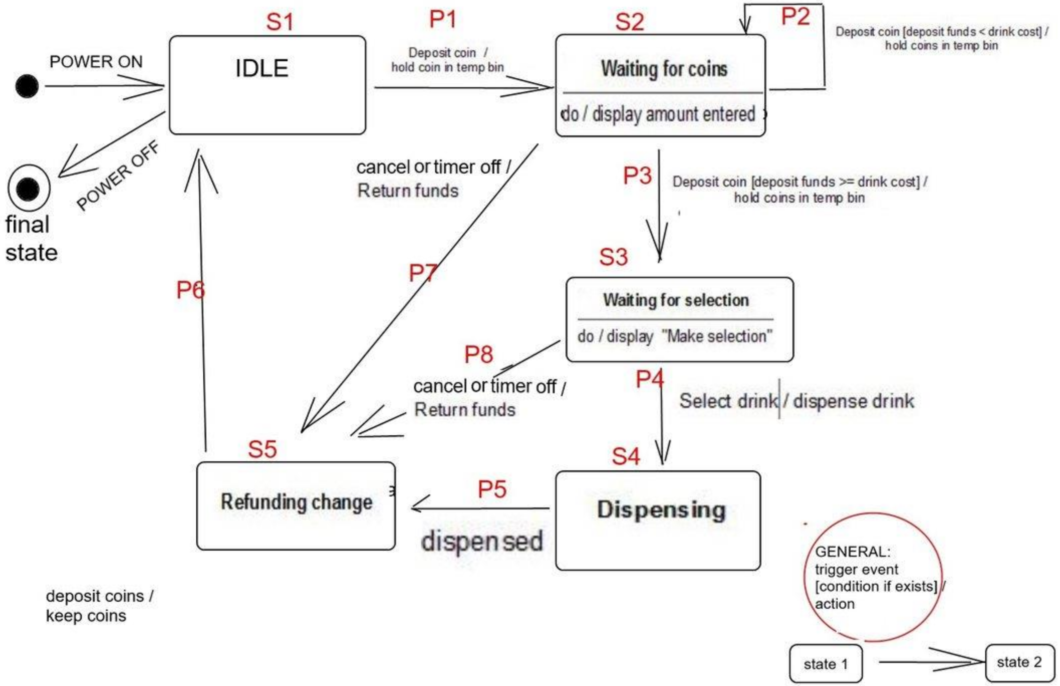
- Today: (Finite) state machines
  - like combinational logic
    WITH MEMORY
  - the memory is called "internal state"
  - FSMs take inputs and produce outputs, but outputs depend on the inputs AND the internal state
  - Apart from the "digital" role of processing inputs into outputs, they perform *actions* in robotics





What is a Finite State Machine (FSM)?

- Consider a vending machine



What is a Finite State Machine?

- Declared in patents to show a robot's Concept of Operation (ConOp)

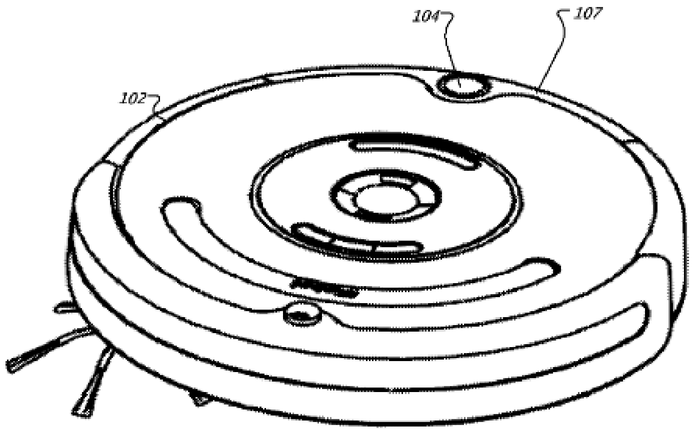


FIG. 1A

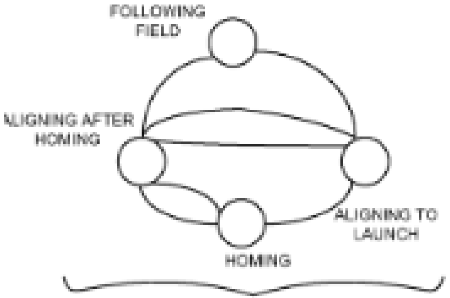


FIG. 25B

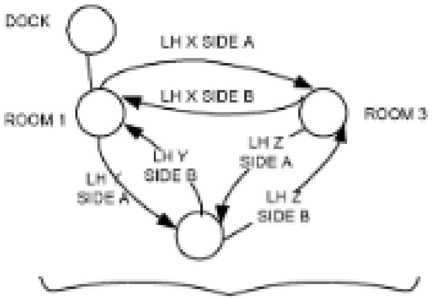
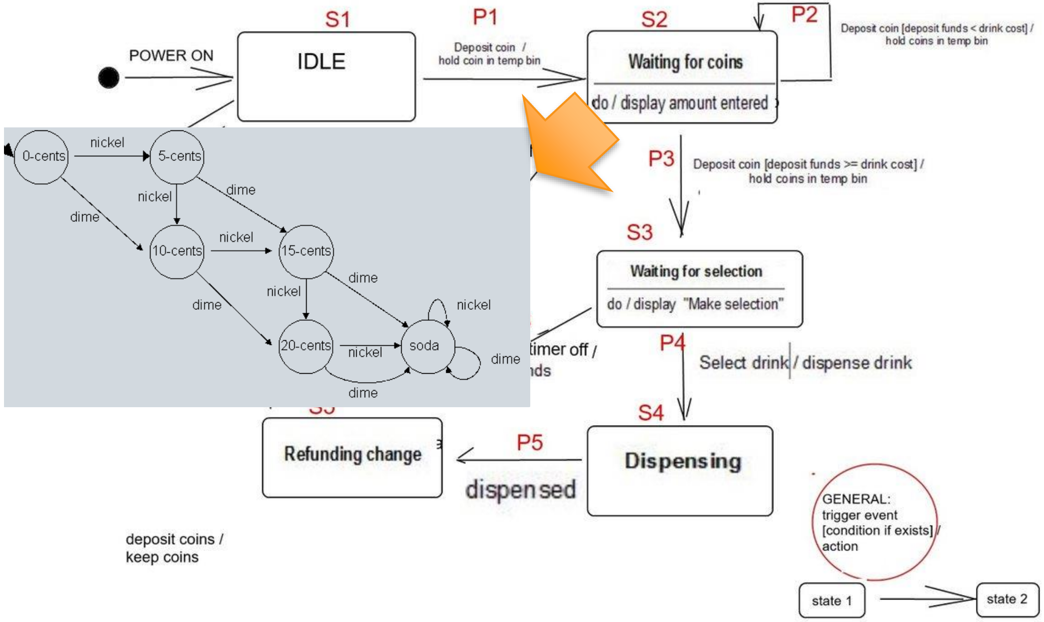


FIG. 25C

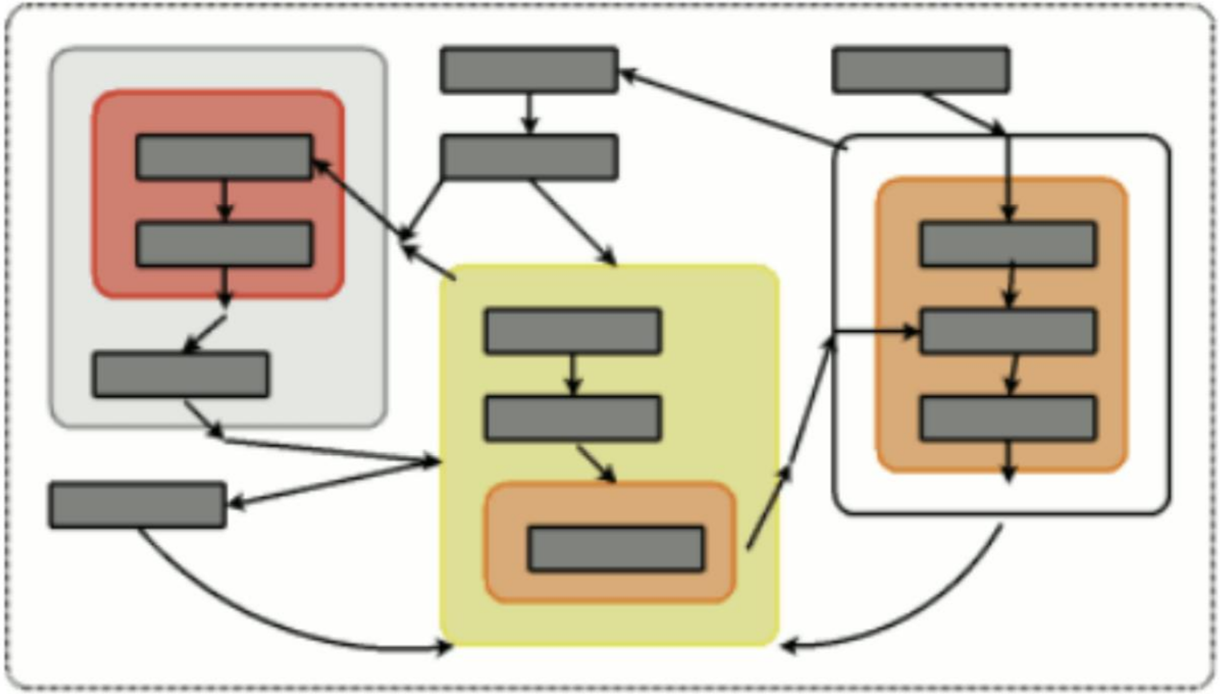
Link to Patent

What is a Finite State Machine?

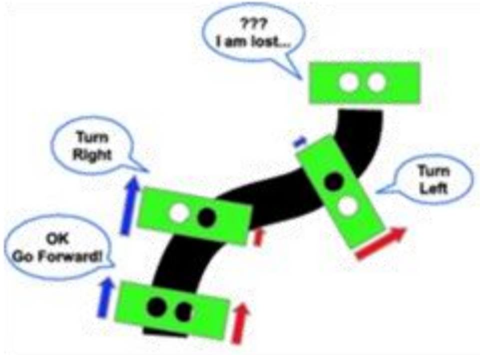
- And FSMs can be embedded in other FSMs

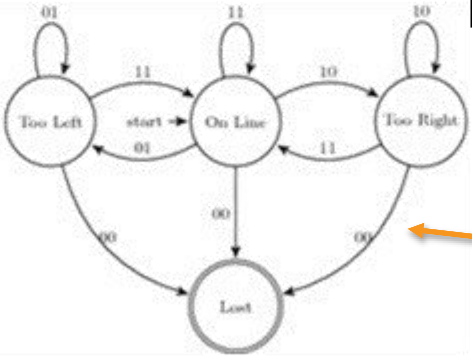


Hierarchical Finite State Machines



- Process digital inputs into digital outputs **with memory of the past inputs!**
- Behaviors in robots can be combined using state machines
- Way of composing skills





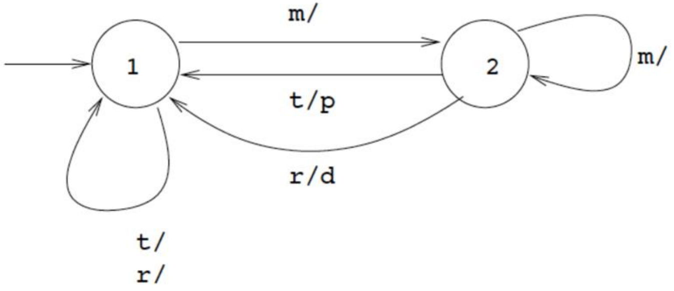
*Clear example, but there are better ways to do this.

Definitions

- **Finite State Machine (FSM)**: A reactive system whose response to a particular stimulus (a signal, or a piece of input) is not the same on every occasion, depending on its current "state"
- The system can be in ***exactly*** one state at a given time
- It is composed of a quintuple (S, s_0, Σ, T, s_g)
  - **States (S)**: The finite (and non-empty) set of possible values
  - **Initial State s_0:** The initial state of the machine $\mathbf{s_0}$
  - **Alphabet $\sum$**: A finite (and non-empty) set of possible inputs
  - **State Transitions Function, T**: Describes how $\sum$ changes the state **s.** $\boldsymbol{S \times \Sigma \to S}$
  - **Final State(s) (s_g)**: A finite set (and possibly empty) subset of **S**
  - We can also define **O**: set of possible outputs

Example: Ticket Machine

- Σ (m, t, r) : inserting money, requesting ticket, requesting refund
- S (1, 2) : unpaid, paid
- s0 (1) : an initial state, an element of S.
- T: transition function: $\delta : S \times \Sigma \rightarrow S$
- F : empty
- O (p/d) : print ticket, deliver refund



States (S): finite (and non-empty) set of possible values

Initial State s_0: initial state of the machine s_0

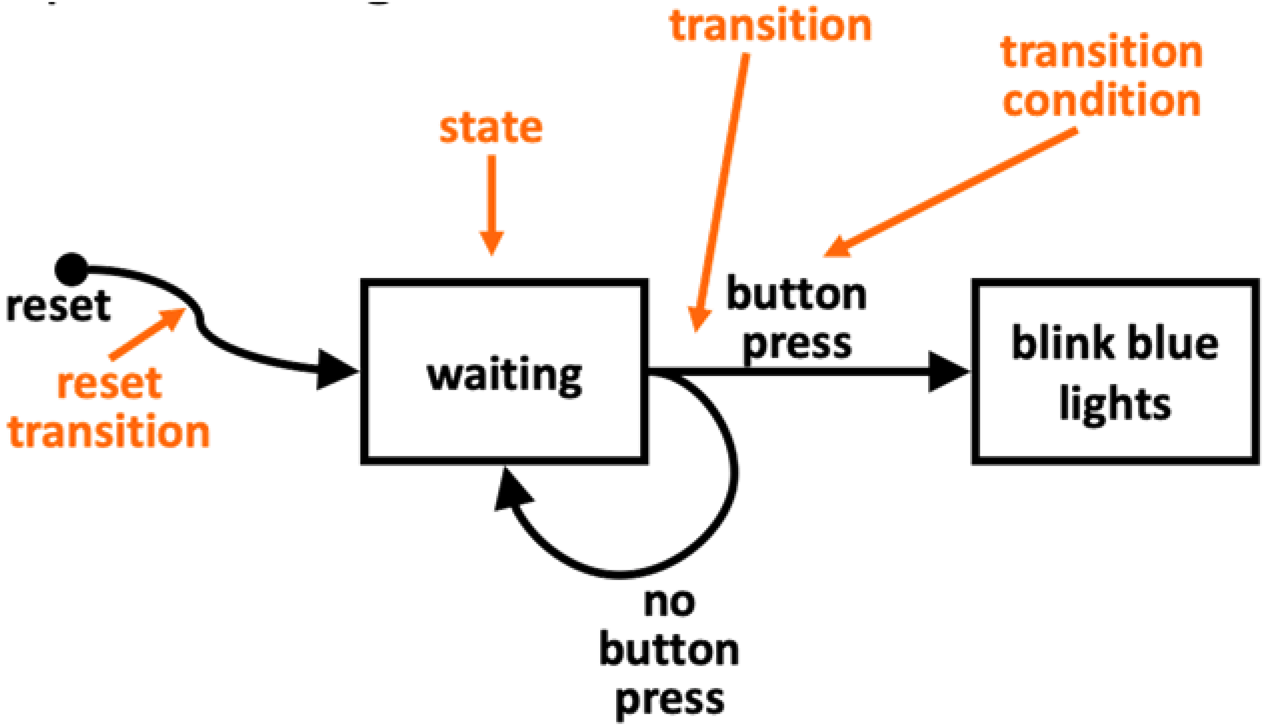
Alphabet ∑: A finite (and non-empty) set of possible inputs

State Transitions Function, T: Describes how ∑ changes the state s. $S \times \Sigma \rightarrow S$

Final State(s) (s_g): A finite set (and possibly empty) subset of **S**

O: set of possible outputs

Representing a FSM



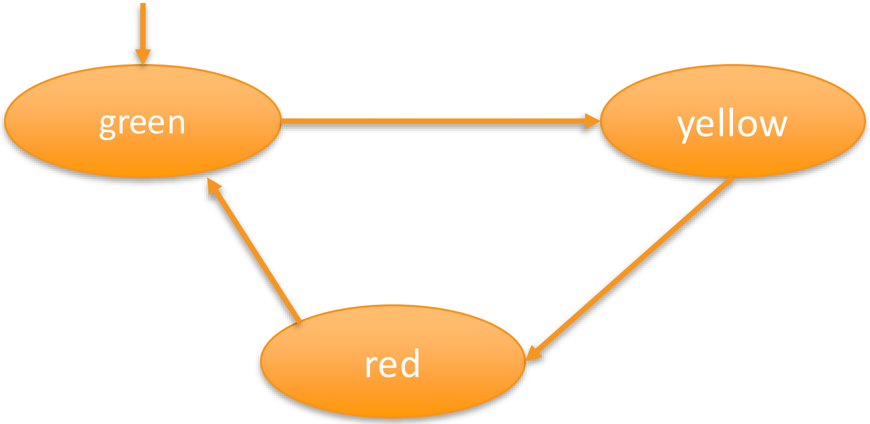
Example: Traffic Light (open loop example)

- **States:** Green, Yellow, and Red
- **Initial State**: Won't matter in the long run. Let's go with Green.
- **Final States:** Lights run indefinitely*, So no final state(s)
- **Alphabet:** Positive integers to represent minutes between 0 and 3
- **Transitions:**
  - When light is green, wait 2 minutes, then go to Yellow
  - When Yellow, wait 1 minute, then go to Red
  - When red, wait 3 minutes, then go to Green

*simplified open loop solution that neglects initialization and multilight coordination.

Exercise

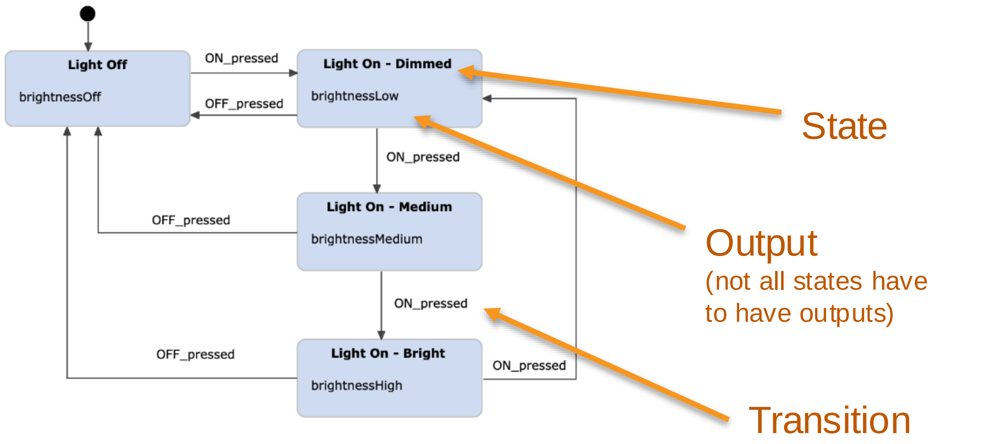
- What state machine would represent the traffic light?



Types of FSMs

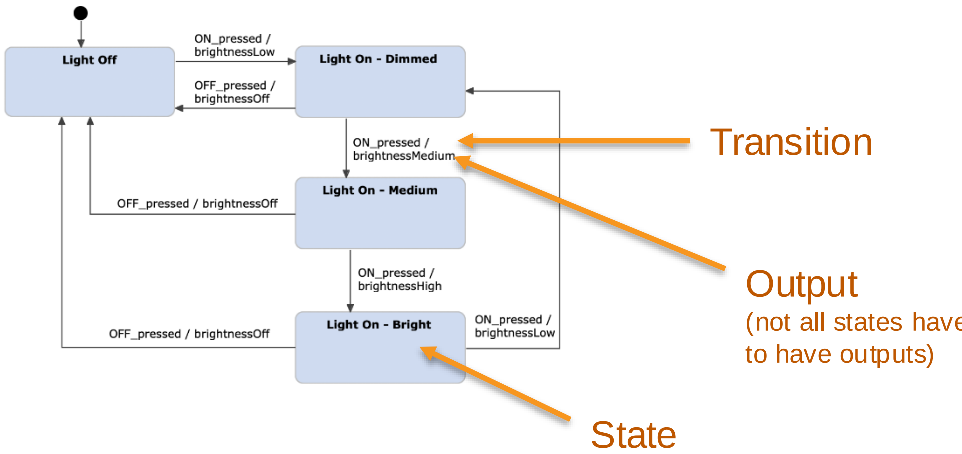
Moore Machine

- Output is only function of state

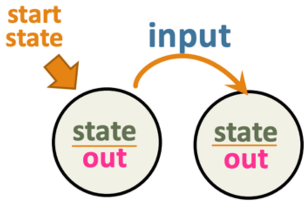


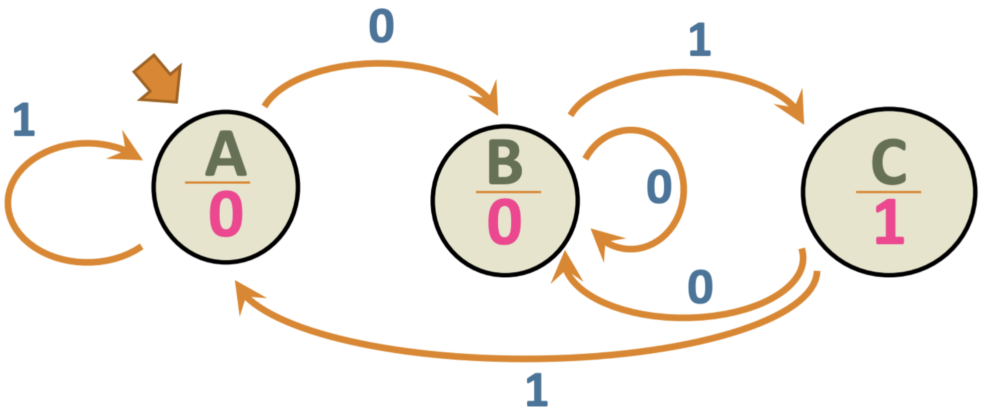
Mealy Machine

- Output is function of the state and the input
- The output is produced during the transition
- Often simpler diagrams (fewer states)



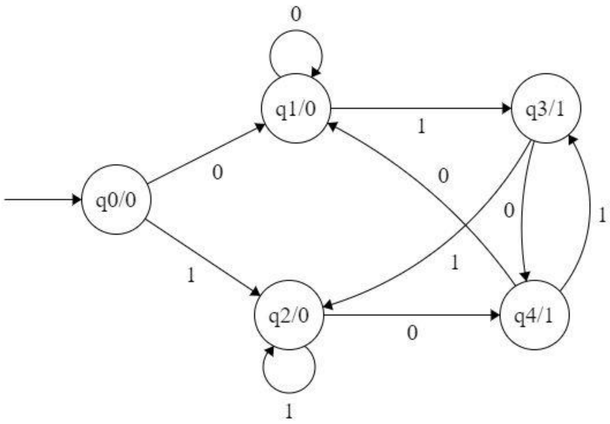
Input, Output and Internal State in a Moore Machine





[credit to K. Bala, Cornell CS]

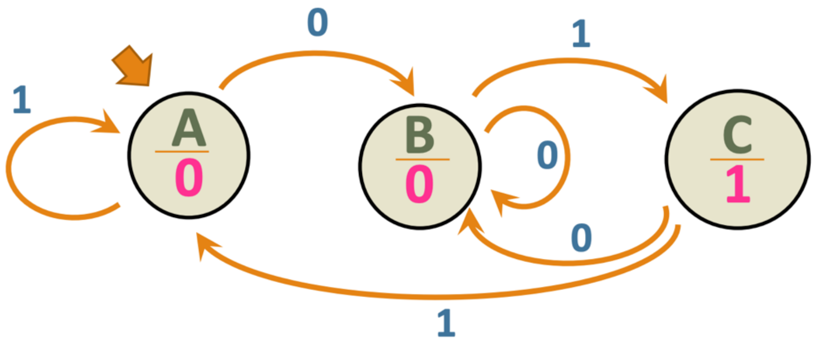
Moore Machine – Output depends only on the present state.



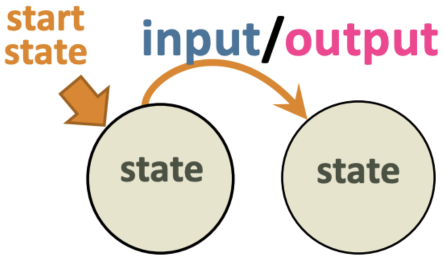
- Example:
  - Input: 11
  - Transition: δ (q0,11) → δ(q2,1)→q2
  - Output: 000

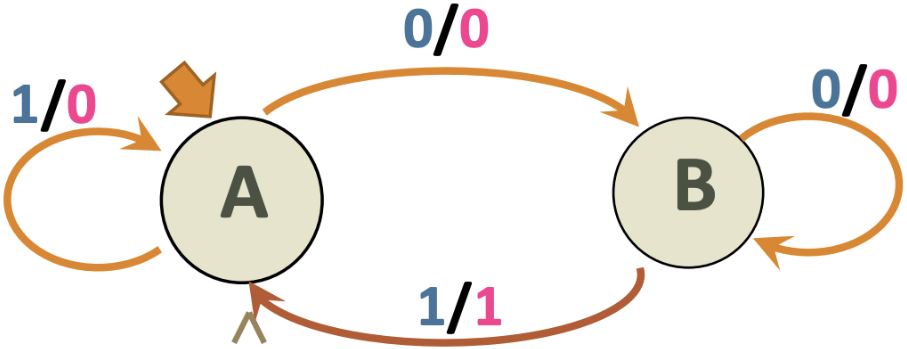
Exercise

- In the given Moore FSM, what is the output if the input is 10010?
- What is the minimum number of bits to represent the state of this FSM?



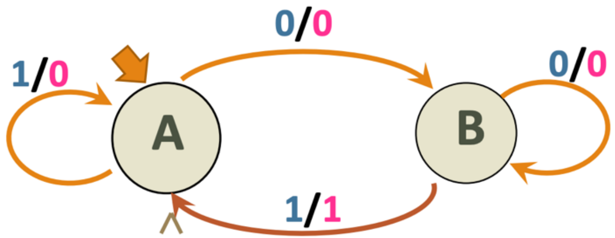
Input, Output and Internal State in a Mealy Machine





Exercise

- In the given Mealey FSM, what is the output if the input is 10010?
- What is the minimum number of bits to represent the state of this FSM?

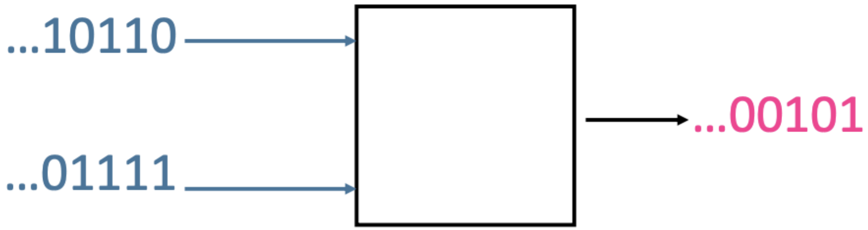


Designing a FSM

- Draw a state diagram
- Write output and next-state tables
- Encode states, inputs, and outputs as bits
- Determine logic equations for next state and outputs
- Draw the circuit

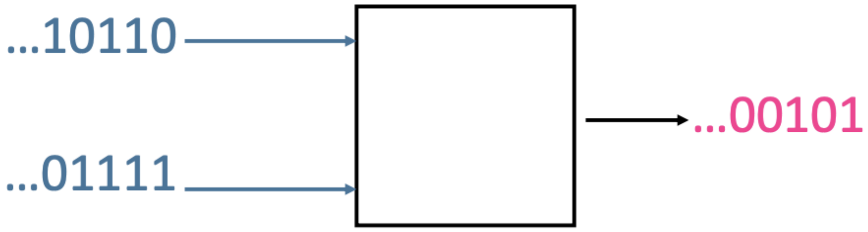
Example: Building a Circuit for a Serial Adder

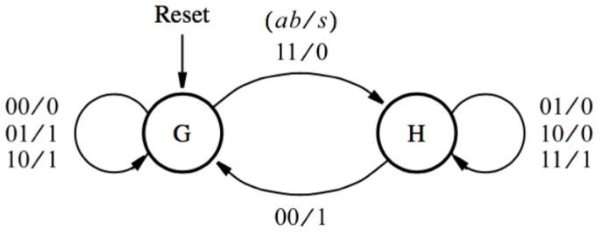
- least-significant-bit first (lsb)
- Mealey FSM



Example: Building a Circuit for a Serial Adder

- least-significant-bit first (lsb)

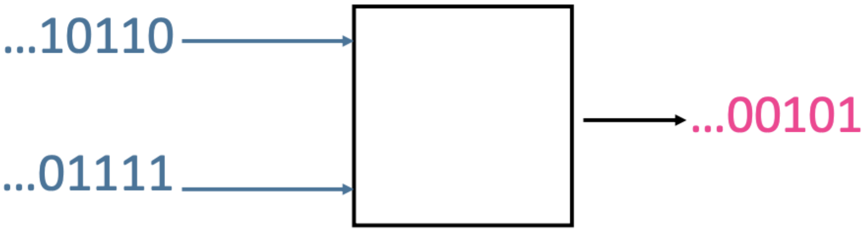




G: carry-in = 0
H: carry-in = 1

Example: Building a Circuit for a Serial Adder

- least-significant-bit first (lsb)



Present state	Next state				Output s			
	ab = 00	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

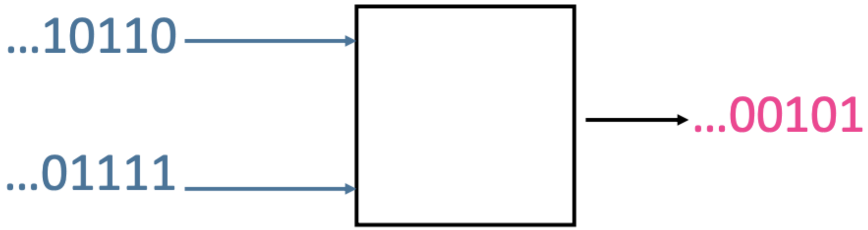
Fig: State table for the Mealy type serial adder FSM

Present state	Next state				Output			
	ab = 00	01	10	11	00	01	10	11
y	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Fig: State-assigned table for the Mealy type serial adder FSM

Example: Building a Circuit for a Serial Adder

- least-significant-bit first (lsb)

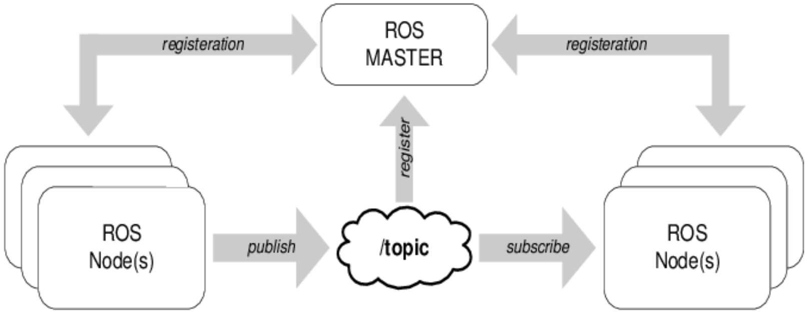


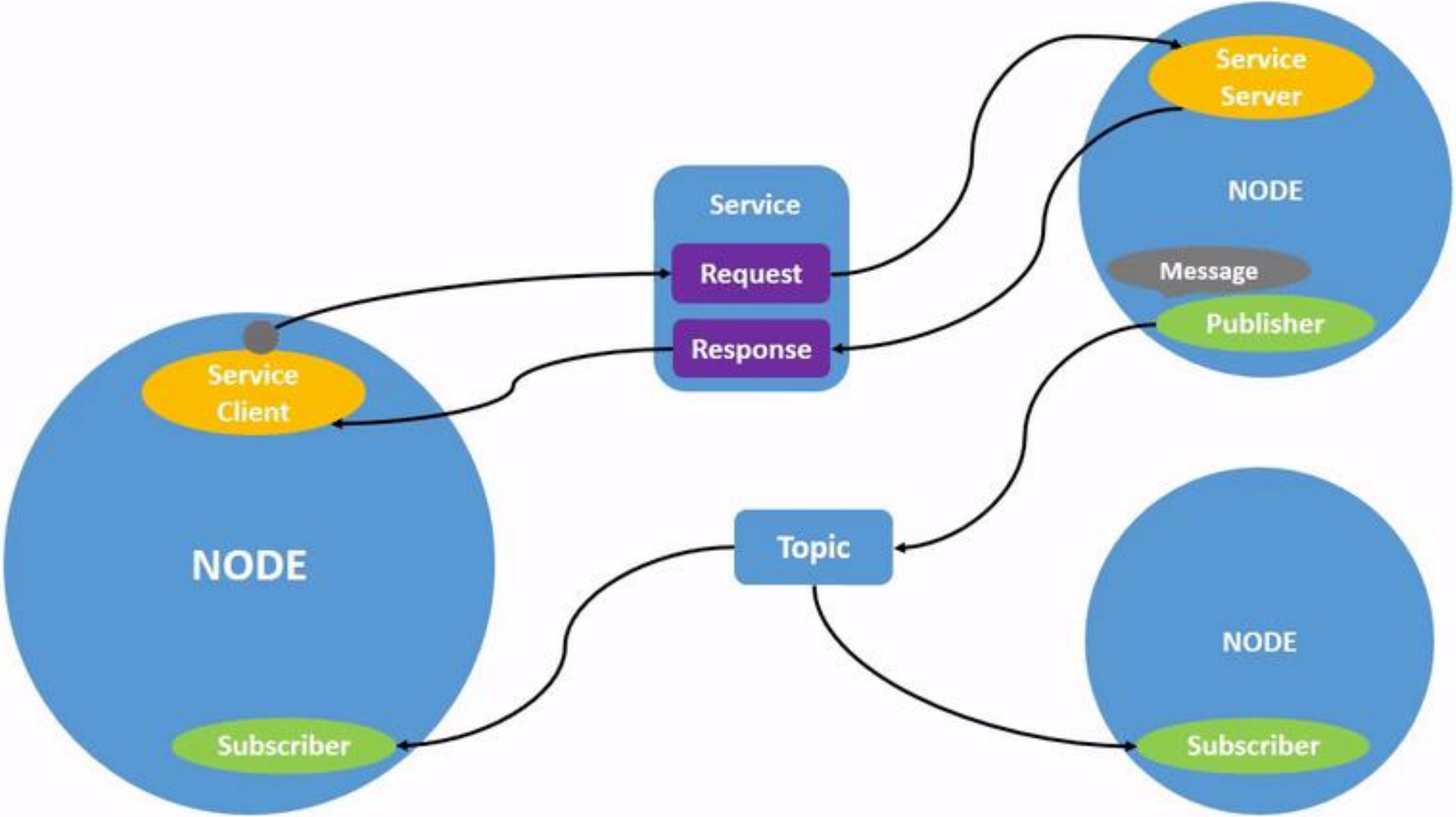
$$Y = ab + ay + by$$

$$s = a \oplus b \oplus y$$

How do we program FSM in Robotics?

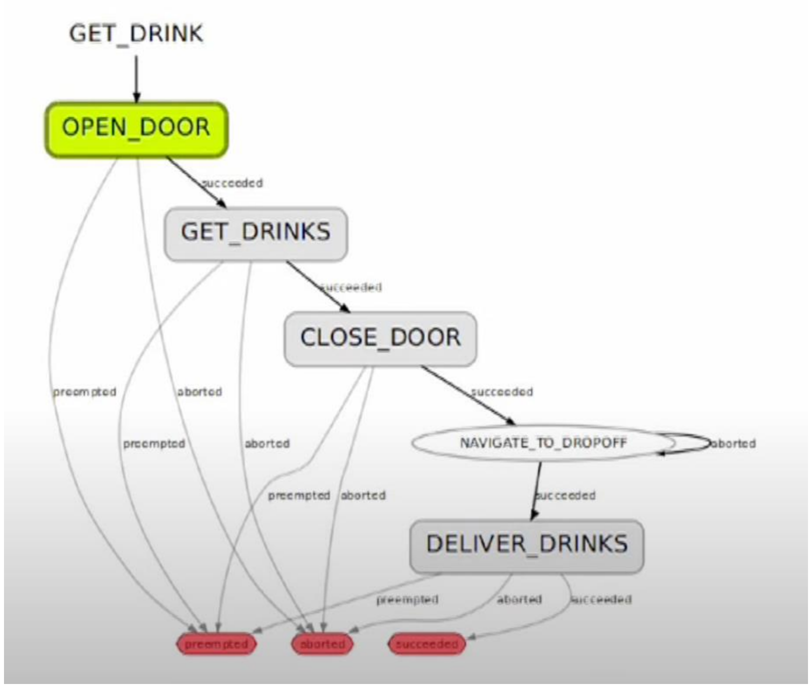
- Introducing ROS (Robot Operating System)
- Set of software libraries and tools that help you build robot applications
- System that intercommunicates processing units (ROS nodes)




Service
Server
NODE
Message
Publisher
Service
Request
Response
Service
Client
NODE
Topic
NODE
Subscriber
Subscriber

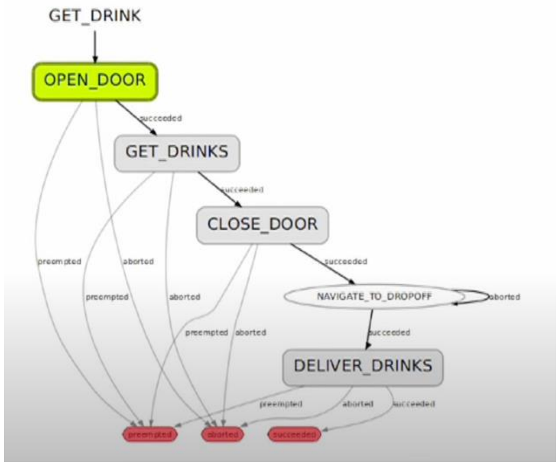
How do we program FSM in Robotics?

- SMACH in ROS
- Tools to create FSM
  - Formal language
    - Create states
      - What is the action in each state?
    - Define transitions
    - Build a machine
  - Visualization
  - ”Executor”



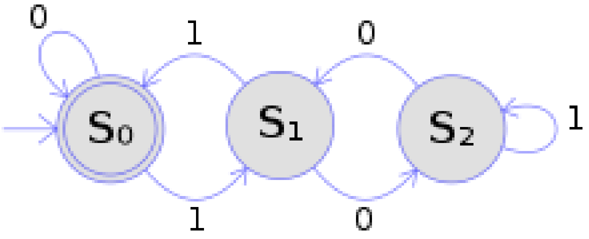
Benefits of FSM

- Separates the control logic (links) from the functionality (nodes)
- The control logic can be expressed concisely as a graph
- Provides an easy way to handle control problems such as:
  - fork/join
  - randomness
  - timeouts
- Easy way to trace/monitor execution

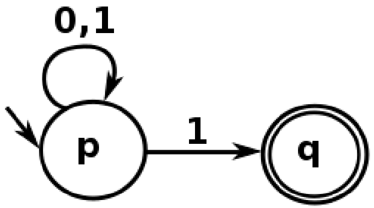


Types of FSMs

- Deterministic Finite Automaton

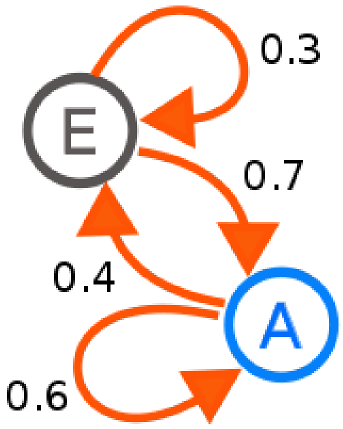


- Non-Deterministic Finite Automaton



Markov Chains

- Random FSMs are directly related to Markov Chains
- A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event
  - $p(s_{t+1}|s_{t-1}, s_{t-2}, \ldots) = p(s_{t+1}|s_{t-1})$

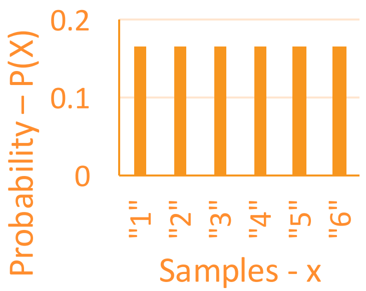


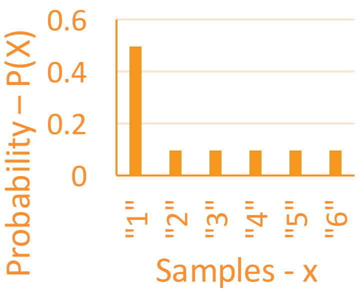
Primer on Probability

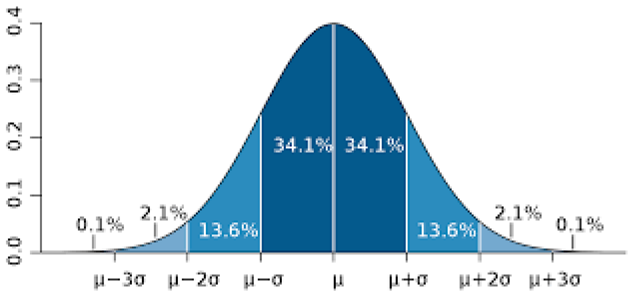
- X is a random variable if it can take different values with different probabilities and can be observed with an experiment
- X can be continuous or discrete
  - Outcome of a dice → Discrete random variable
  - Height of adult population of a country → Continuous random variable
- Sample space: all possible values that a random variable can take
  - X=outcome of a dice → S={1,2,3,4,5,6} → Discrete
  - X=Height of adult population → S=[1 feet, 9 feet] → Continuous
- P(X=x): Probability of an event (or probability of X taking the value x)
  - $P(X=x) \leq 1$
  - $\sum_x P(X = x) = 1$

Probability distribution

- Function P(X=x) (or directly P(x))
  - When we plot it:
    - x axis: the sample space
    - y axis: the probability of each sample, P(x)







Dependent and Independent Variables

- Let's assume X and Y are two random variables
  - X → outcome of a coin toss, Y → outcome of a dice toss
  - X → develop lung cancer, Y → be a smoker
- P(X,Y) is called the joint probability
  - P(X="tails", Y="6")
  - P(X="develop lung cancer", Y="smoker")

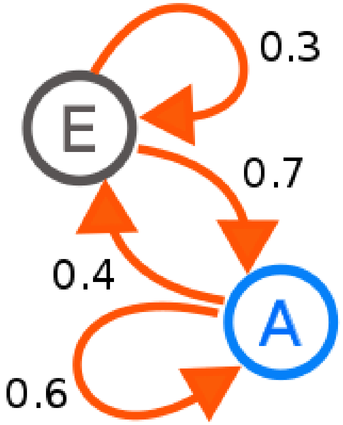
- P(X|Y) is called a conditional probability, probability of X conditioned on knowing Y
  - P(X|Y="6") → Probability of outcome of a coin toss given that the dice tossed "6"
  - P(X|Y="smoker) → Probability of developing lung cancer given that is a smoker

Dependent and Independent Variables

- X and Y are said to be <u>independent variables</u> if the distribution of X is not influenced by the value taken by Y and vice versa. In that case:
  - the probability of a joint event is the product of the probability of one event and the probability of the other
    - P(X="tails", Y="6") = P(X="tails") • P(Y="6")
  - the conditional probability of one knowing the other is the same as the original probability
    - P(X | Y="6") = P(X)
- X and Y are said to be <u>dependent variables</u> if knowing the value of one of the events affects the probability distribution over the other event. In that case:
    - P(X | Y="smoker") ≠ P(X)

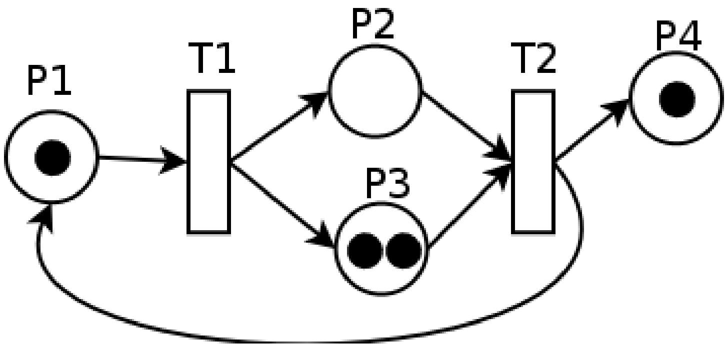
Back to Markov Chains / Markov Processes

- The state is only dependent on the previous state!
- If we can decide the action to take (the input to the process) we talk about a Markov Decision Process
  - Critical concept in Robot Learning!
  - Formalism for planning, reinforcement learning, imitation learning…
- Non-Deterministic Finite State Machines are Markov Processes!



Petri Nets (Place Transition (PT) Network)*

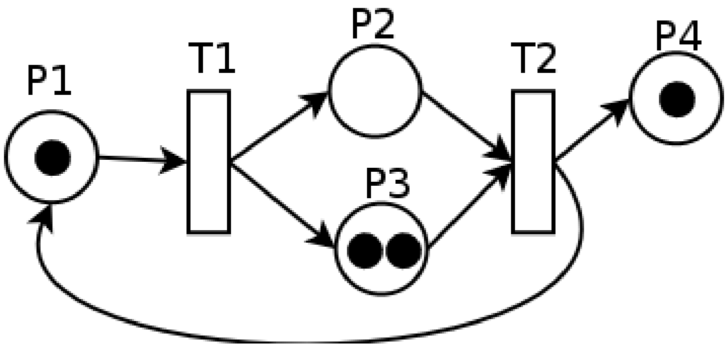
- Bipartite Graph → Two types of elements:
  - Places
  - Transitions
- Places can contain tokens
- A transition is enabled (can happen) if all places connected to contain at least 1 token
- After a transition triggers, all places connected to it gain a token
- Execution is clocked triggered



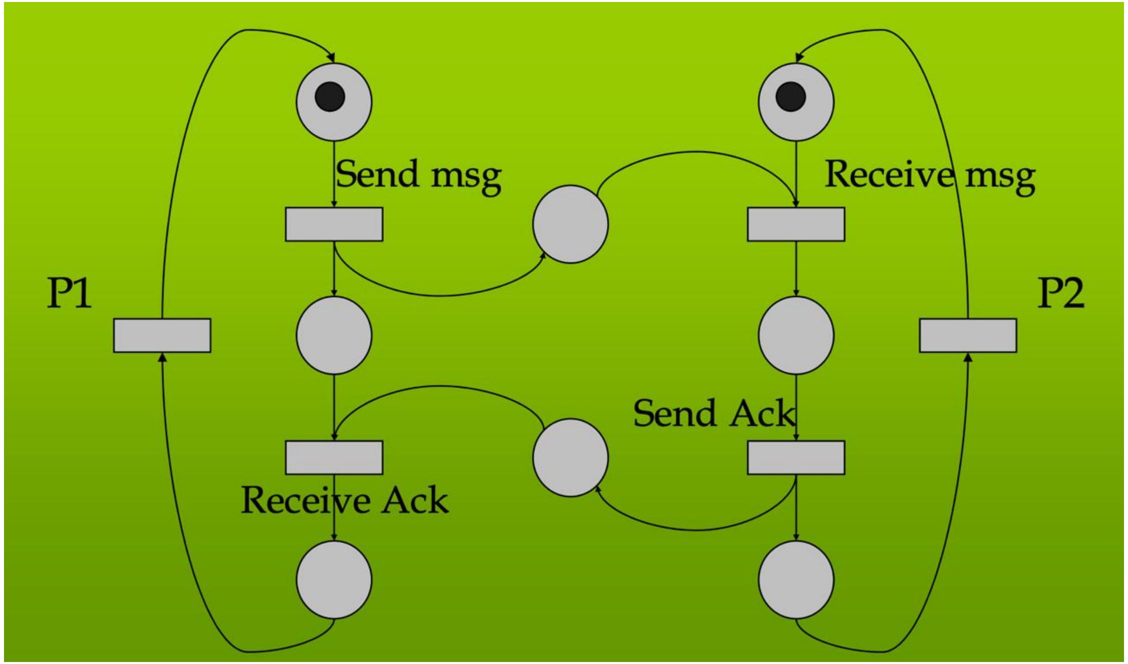
* actually named after Carl Petri

What are Petri Nets for?

- Modeling processes
  - Chemical processes
  - Business processes
  - Automation processes!
- Represents concurrency and interdependencies
- Related to workflows

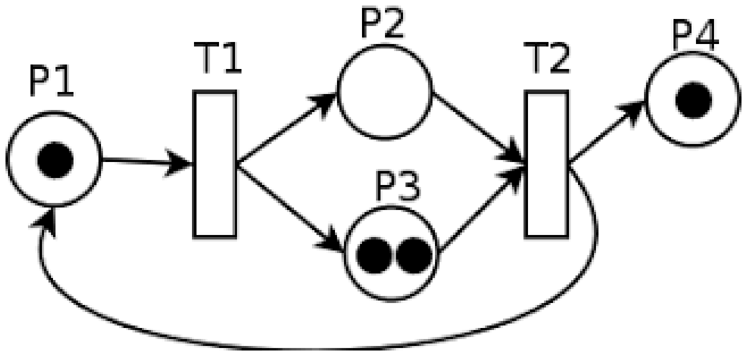


Example of Petri Net



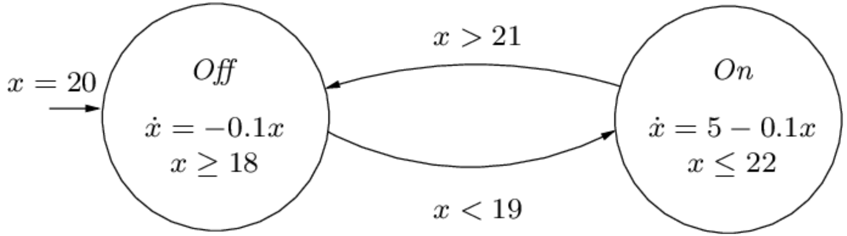
Exercise

- In the given Petri Net, what is the sequence of transitions after three clock steps?
- What is the final state of each of the places (number of tokens)?



Hybrid Automata

- Model to describe systems in which digital computational processes interact with analog physical processes
  - Robots!
- A HA is a bipartite graph (V,E)
  - V: vertices = control modes
  - E: edges = control switches
- Control modes change the state of the system (dynamics!) until a switch condition is reached



Example of Hybrid Automaton