Modeling

Context

Embedded systems involve SW + HW

We used FSM-based design for software

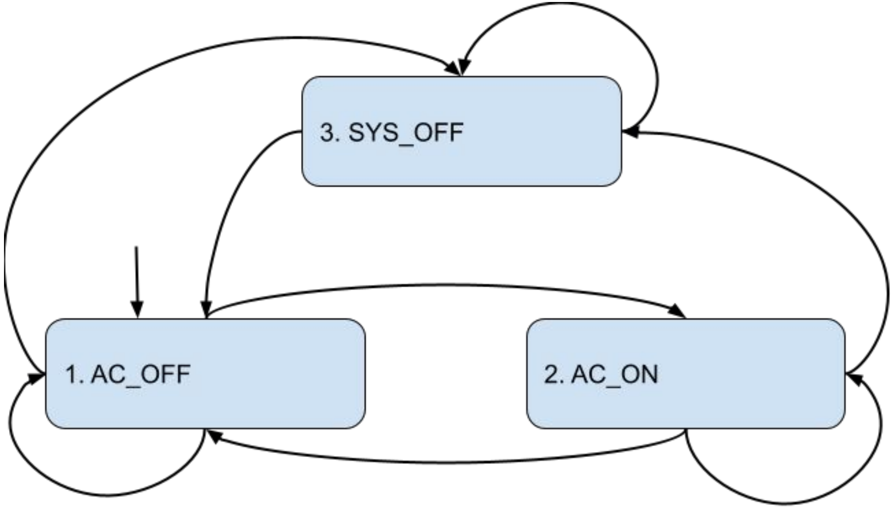
- Translates pretty easily to code
- Guides unit testing
- Helps us reason about the system?

Can we expand that reasoning? Can we incorporate HW/timing into that reasoning?

FSMs as models

FSM describes behavior of the system

Abstracts away some aspects of the system



What do you think when you hear "all models are wrong, some models are useful"

?

Formalizing FSMs

We handwaved some aspects of FSMs

- Role and behavior of inputs and outputs
- Presence/absence of self-loops
- Distinction between FSMs and extended FSMS

Back to the formal definition

(Lee/Seshia 3.3.3)

An FSM is a 5-tuple: (States, Inputs, Outputs, update, initialState)

- States is a finite set of states
- Inputs is a set of input valuations
- Outputs is a set out output valuations
- update: States x Inputs ⟶ States x Outputs is an update function
- initialState is the initial state

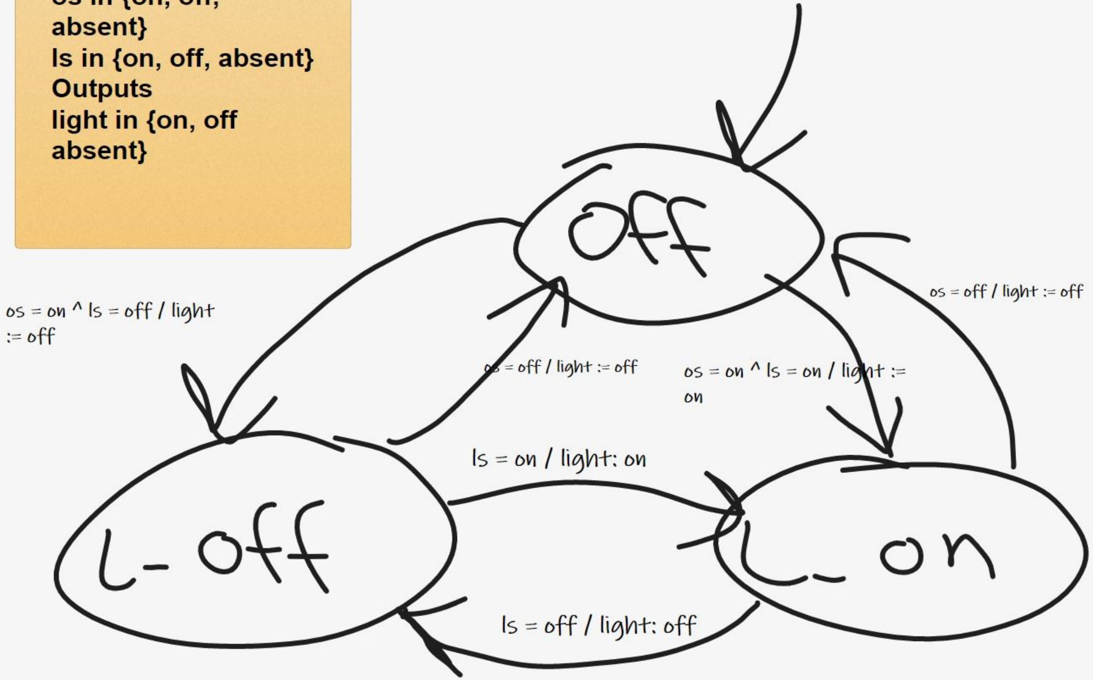
Valuation: a set of values that a signal can take on or the assertion that the value is absent

Numerical signals: R U {absent} or N U {absent}
Pure signals: {present, absent}
Categorical signals: examples {1, 2, 3, ... 8, absent} or {up, down, left, right, absent}

Board example: system with two switches

System with two switches and one indicator light
on/off switch: if off, no response to light switch and light is off
light switch: light on if on, off if off

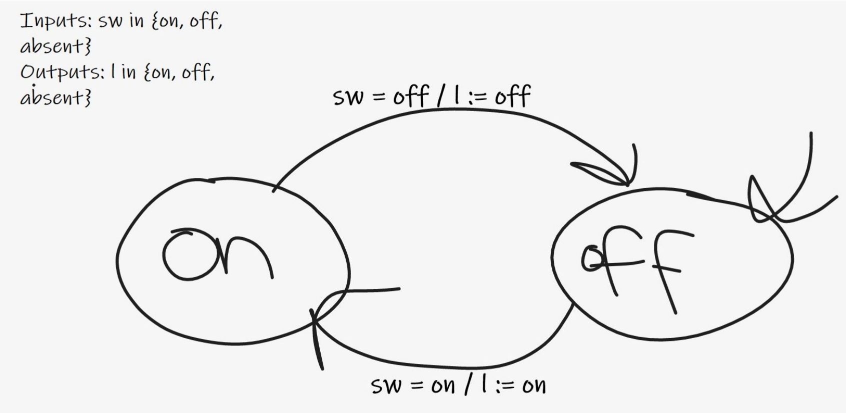
Inputs
os in {on, off, absent}
ls in {on, off, absent}
Outputs
light in {on, off absent}

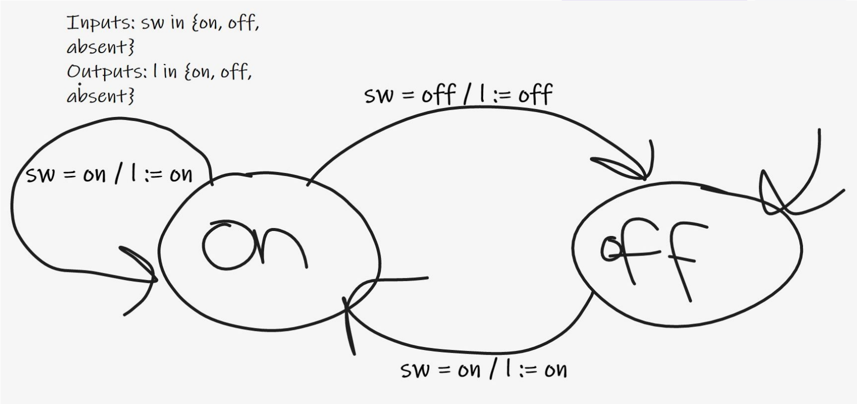


Self loops, present/absent I/O

Difference between the below two FSMs?

- FSM on the right: sw keeps being read and output keeps being asserted





Keeping track of data

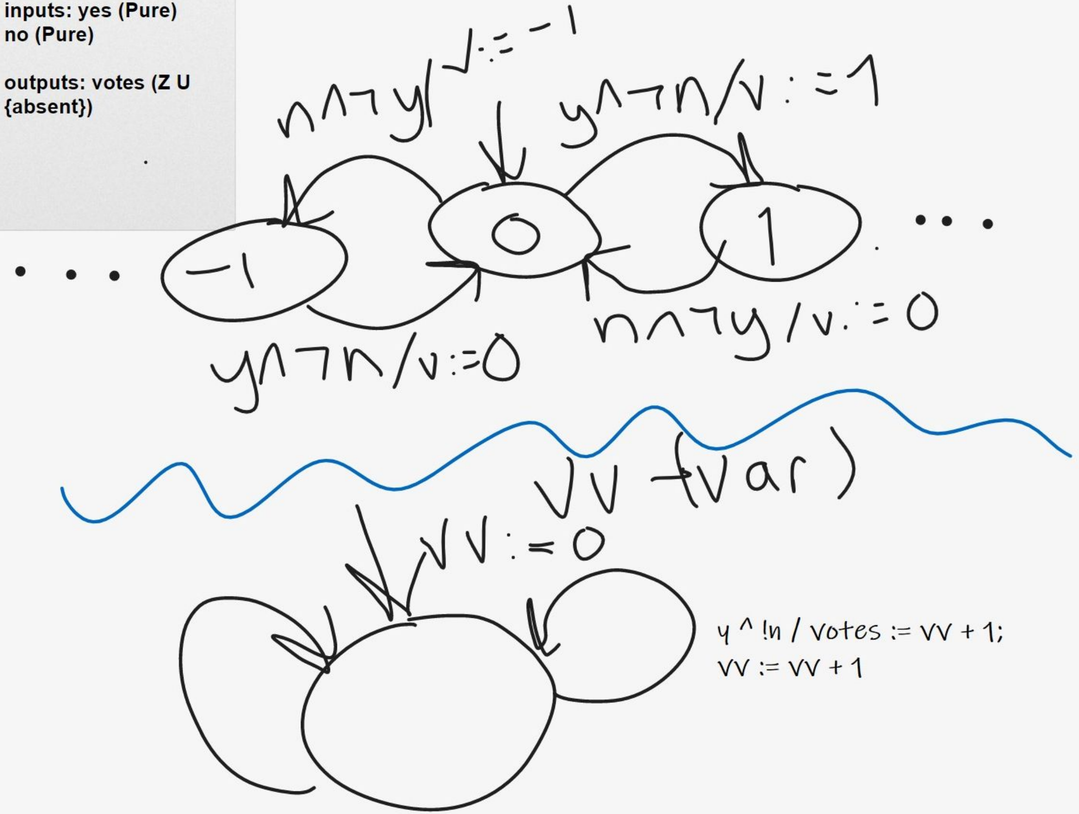
An FSM is a 5-tuple: (States, Inputs, Outputs, update, initialState)

- How do we keep track of internal data?
- Example: system with yes/no vote buttons, keep track of difference in votes (board example)

Vote buttons

inputs: yes (Pure)
no (Pure)

outputs: votes (Z U {absent})



FSM vs Extended SM

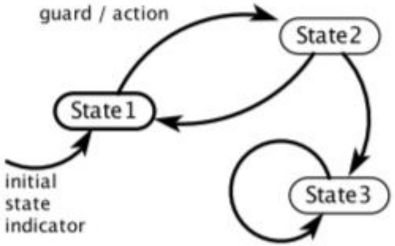


Figure 3.3: Visual notation for a finite state machine.

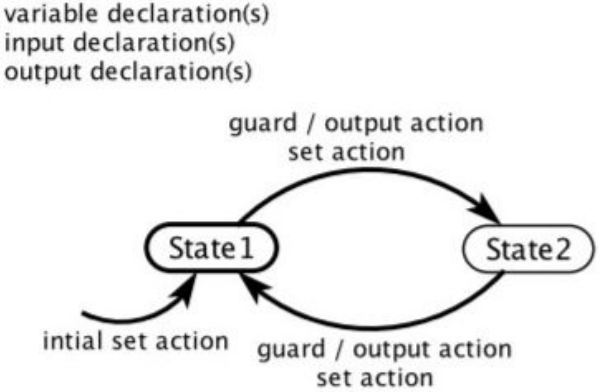
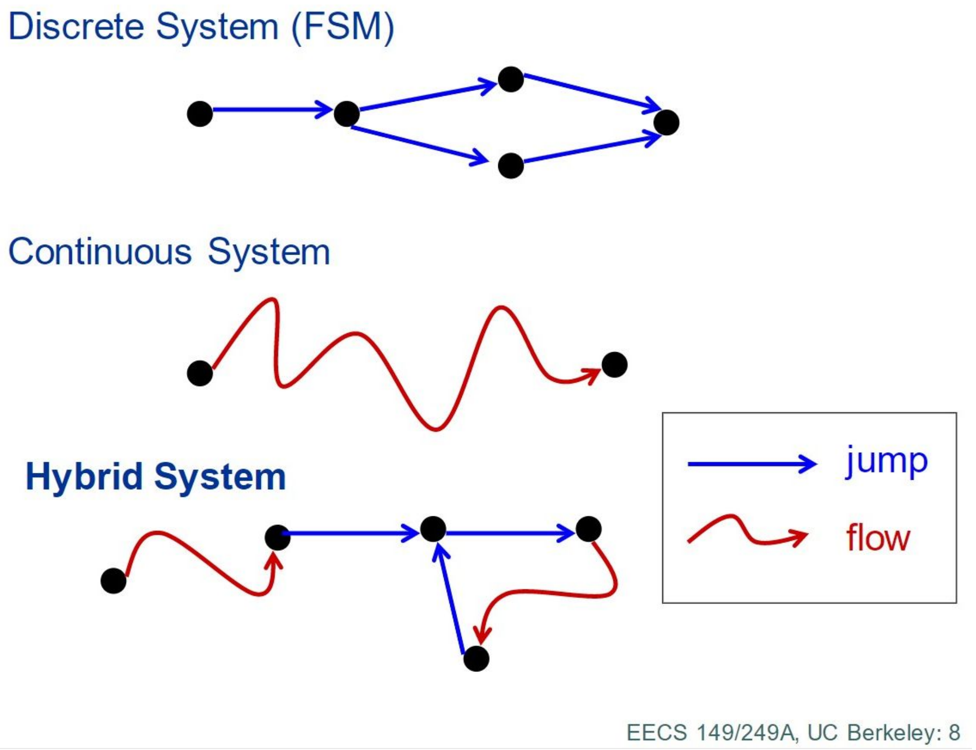


Figure 3.9: Notation for extended state machines.

Lee/Seshia chapter 3

What are we missing out on when we tell time by using "mils" as an input?

Hybrid systems



Slide from Prabal Dutta and Sanjal A. Seshia, 2019

Timed automata

Distinction between discrete and continuous variables

Continuous behavior defined in "states"

Now called "modes"

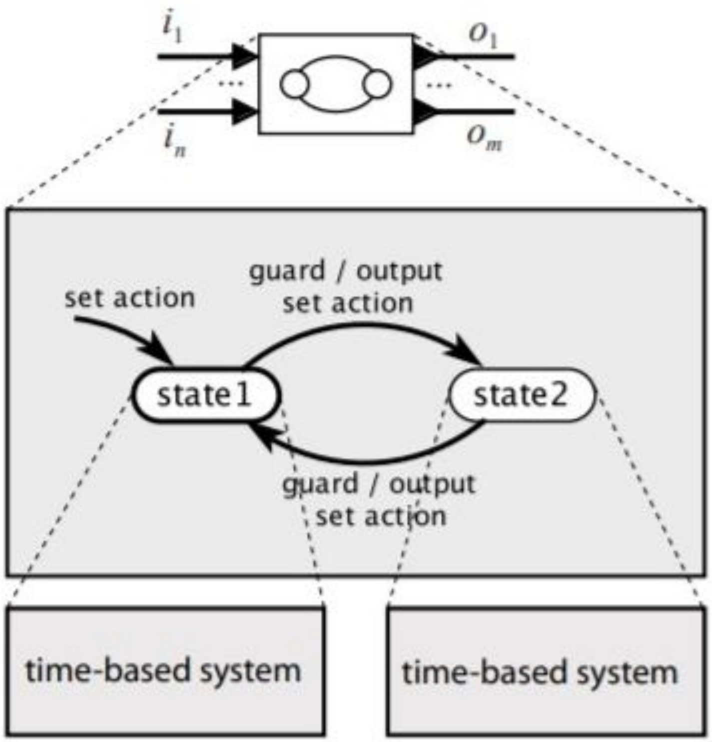


Figure 4.4: Notation for hybrid systems.

Lee/Seshia chapter 4

ODEs

Sometimes it is more desirable to describe a variable in terms of how it changes rather than its explicit form

- Useful for: modeling, reasoning

Define a function in terms of its derivative and possibly initial conditions

- Ordinary Differential Equation, or ODE

Solving general ODEs is beyond the scope of the class, but we will discuss some patterns here

Discussion of homework problems

Example: bouncing ball

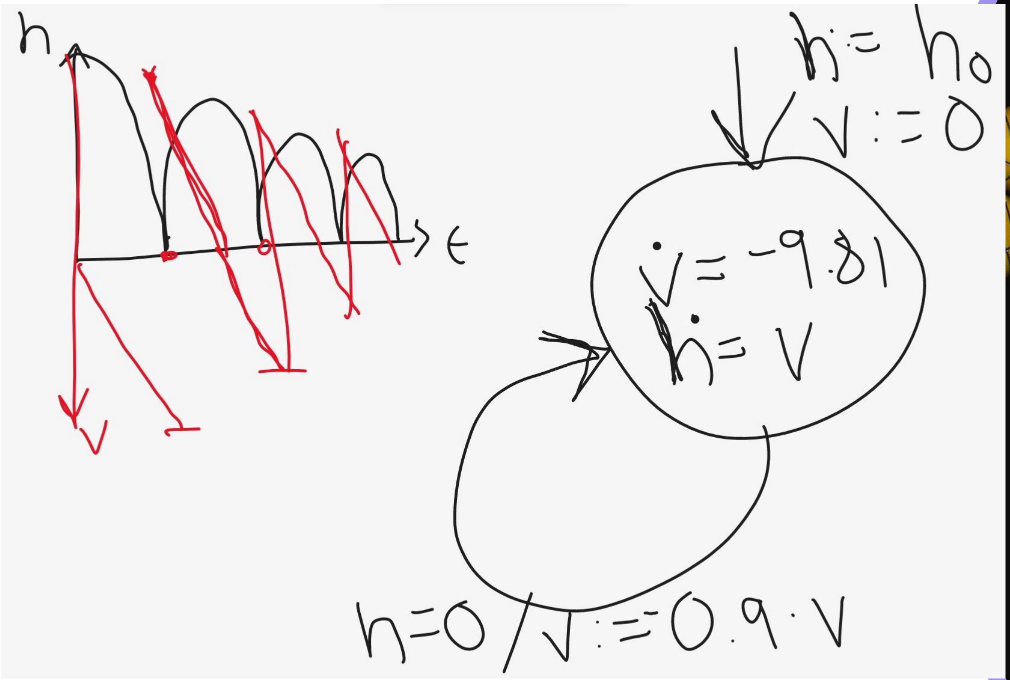
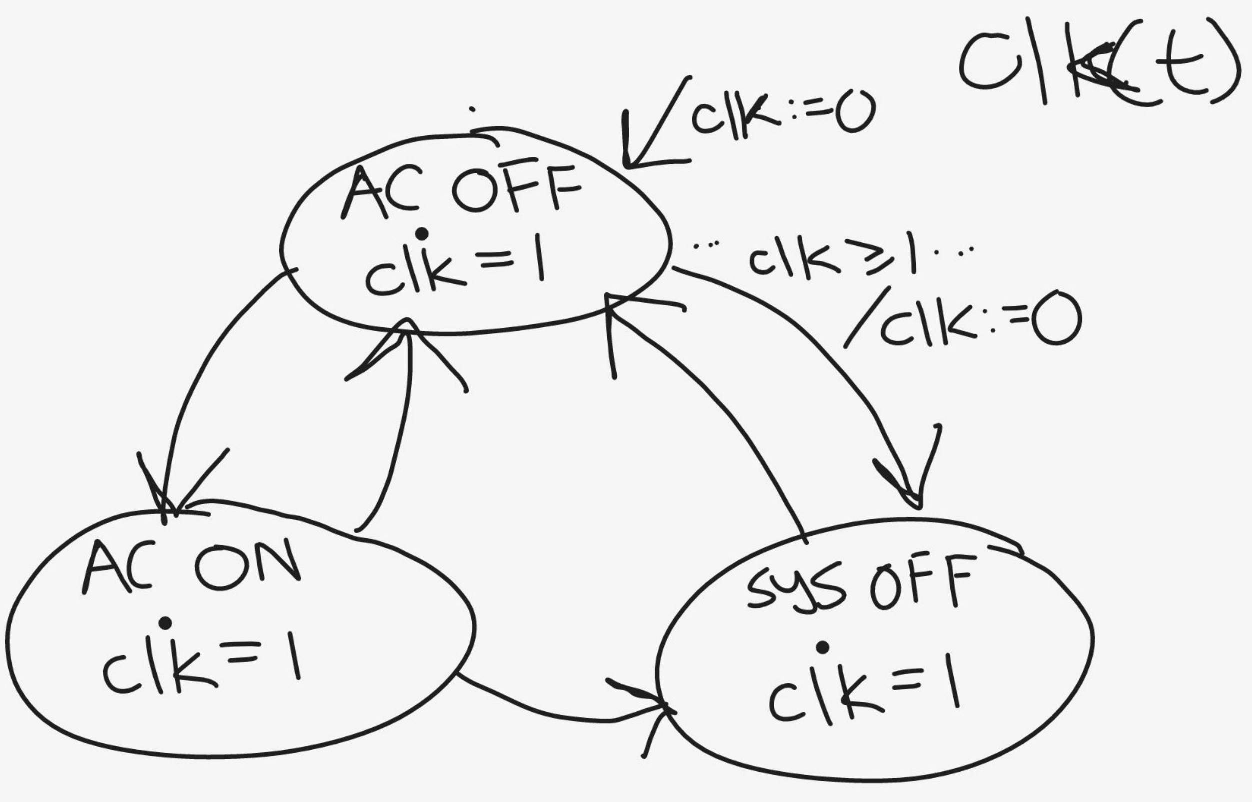
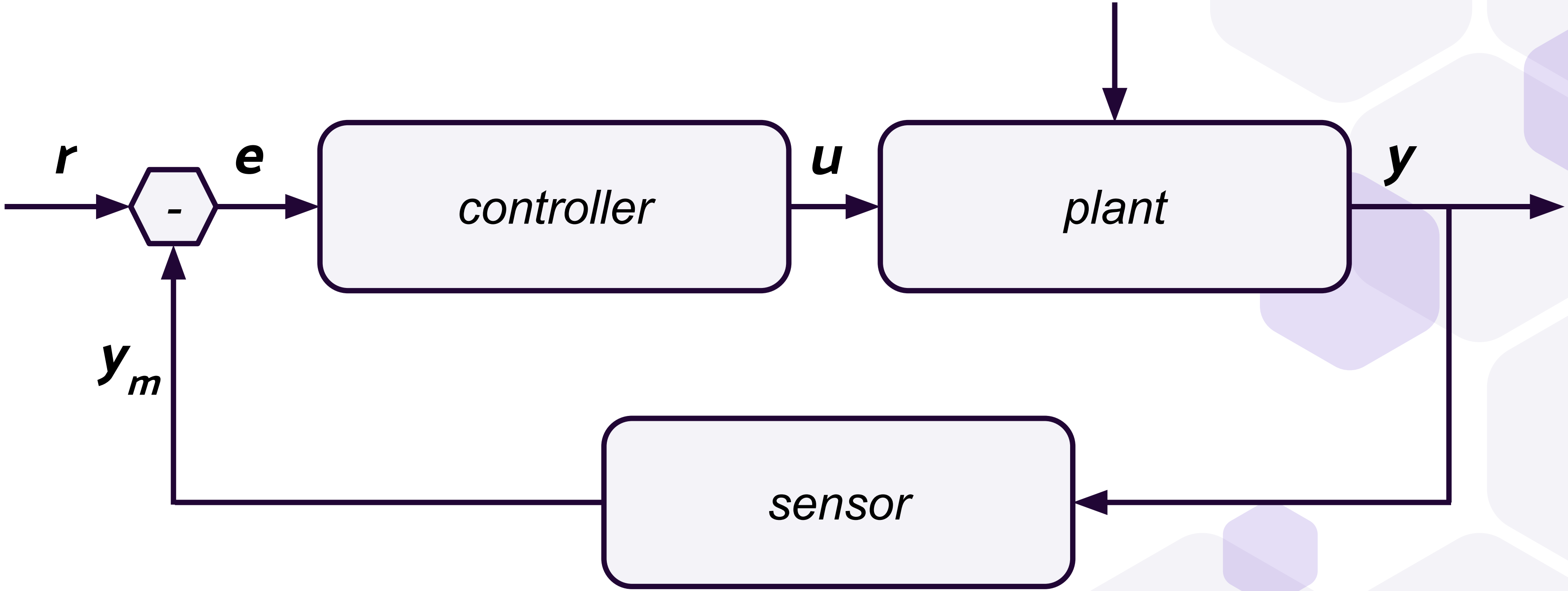


Image source

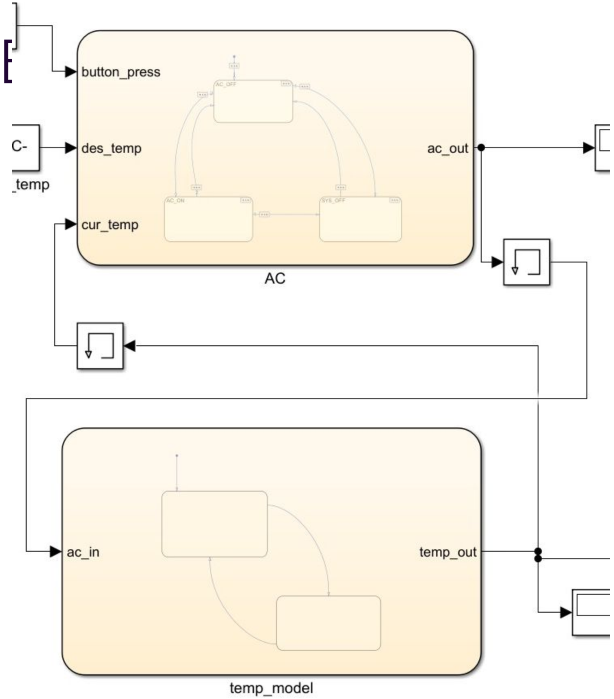
Air conditioner with continuous time

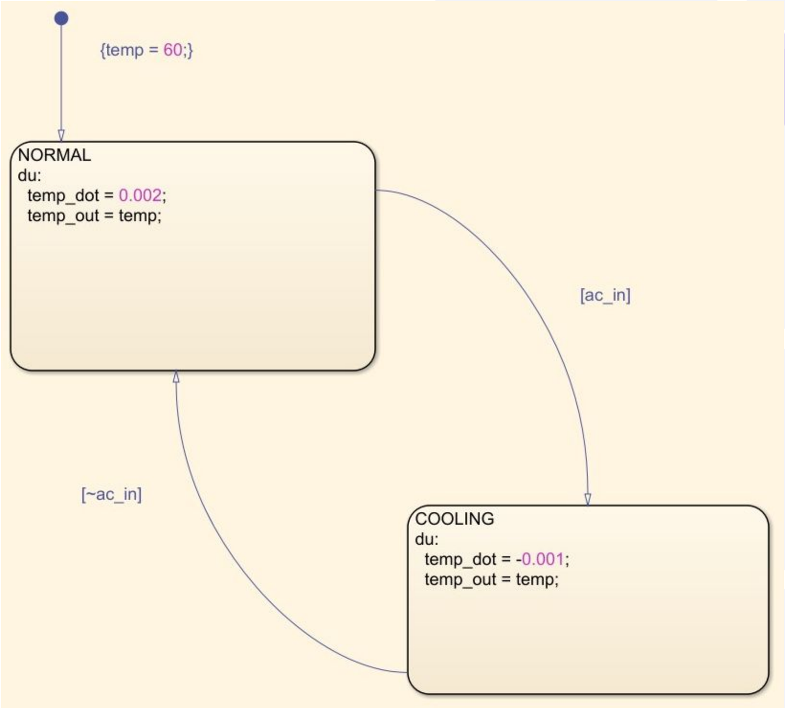


Feedback loops and control



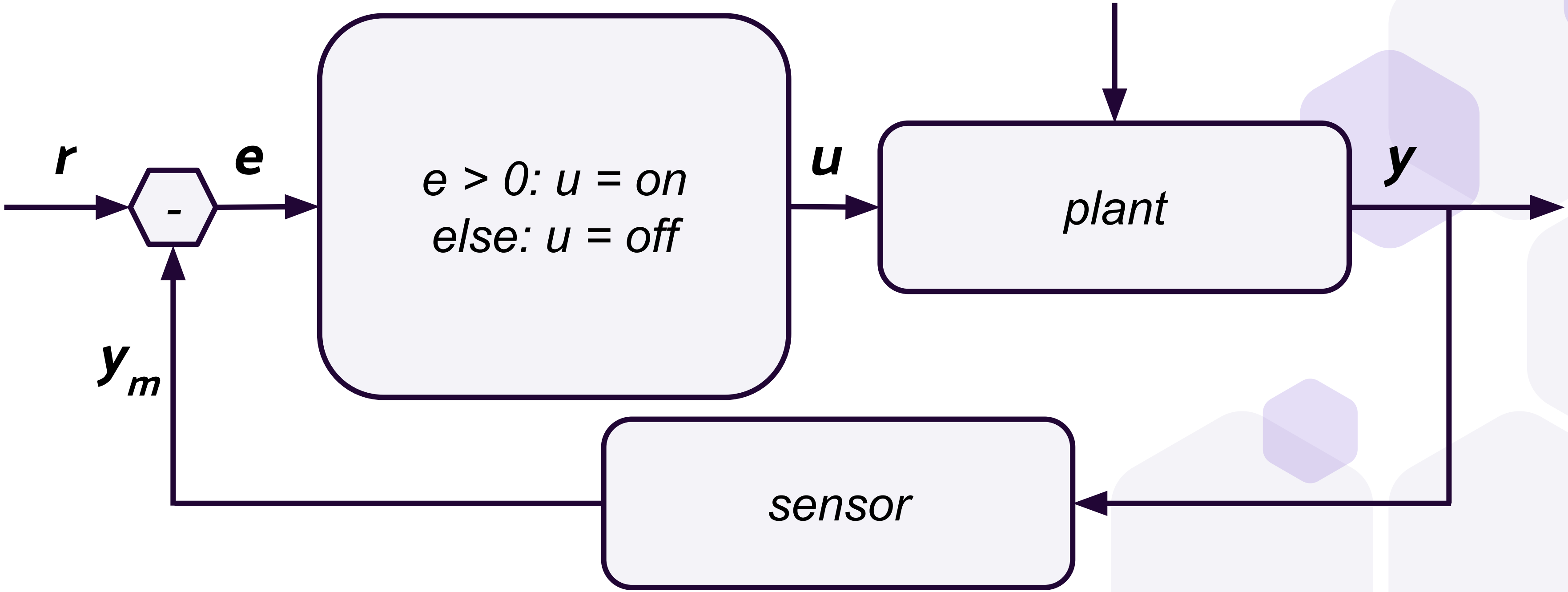
Air conditioner temperature model





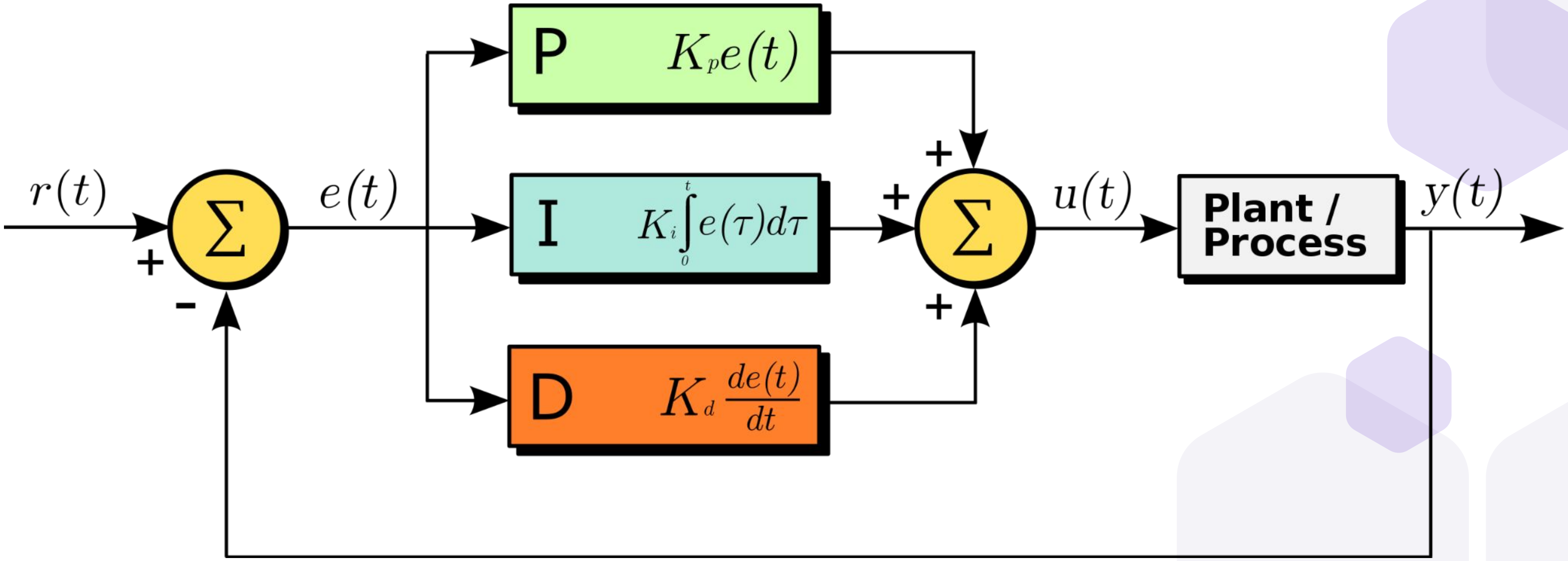
Bang-Bang controller

Controller output is 2-state ({on, off}, {up, down}, etc)

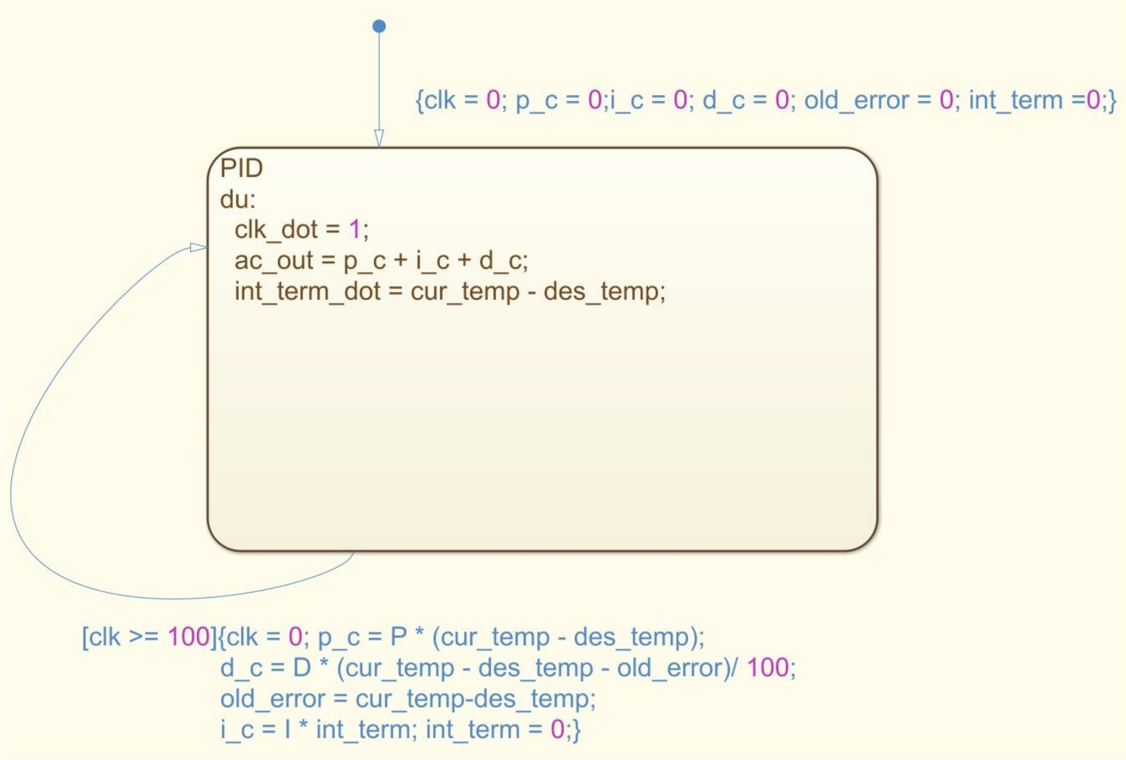


PID controller

For continuous controller outputs



Board and simulink discussion of PID



Bonus/if time: composition of automata

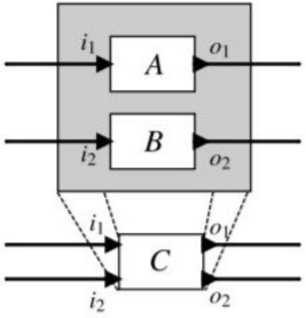


Figure 5.2: Side-by-side composition of two actors.

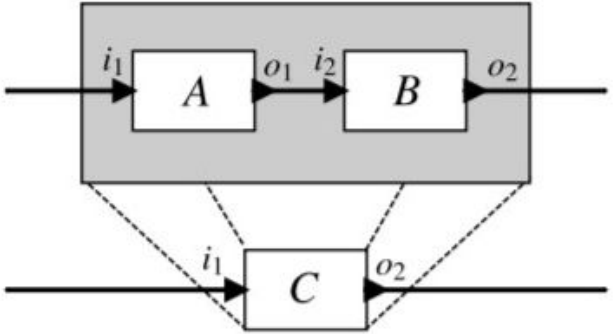


Figure 5.7: Cascade composition of two actors.

Lee/Seshia chapter 5

Bonus/if time: feedback loops in automata

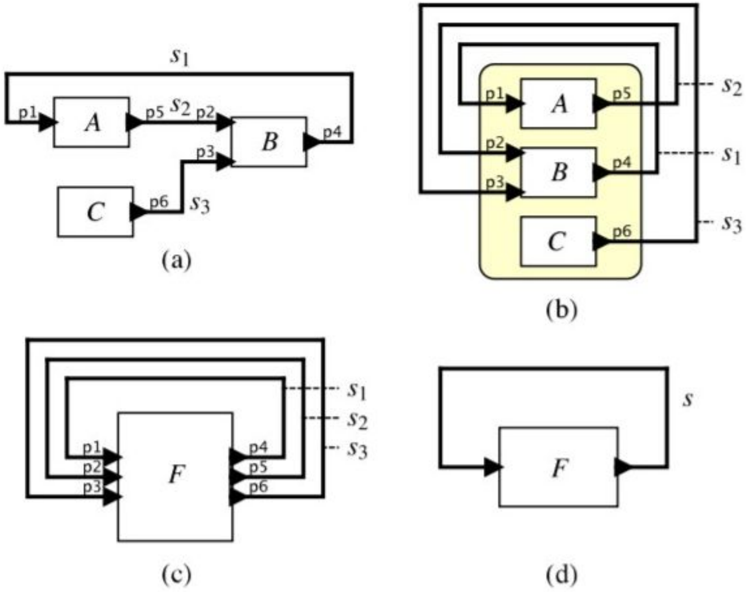


Figure 6.1: Any interconnection of actors can be modeled as a single (side-by-side composite) actor with feedback.

Lee/Seshia chapter 6