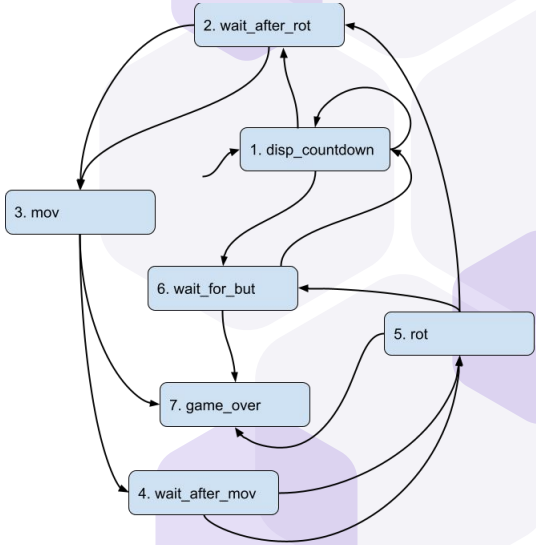
Project demo on Dec 12!

28: Modeling

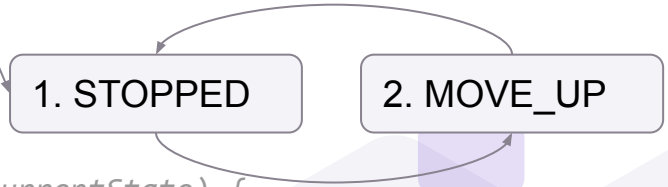
FSM-based design for software

- Incorporates reasoning about state into design
- Translates pretty easily to code
- Guides unit testing



Can we formalize how we talk about FSMS to help *prove* properties about the system?

Can we incorporate hardware/physics into our state-based model?




```
(currentState) {
e STOPPED:
// transition 1-2
if (buttonUPpressed && deskJeight != maxJeight) {
```

Formal verification

Does the model of the system meet the requirements?

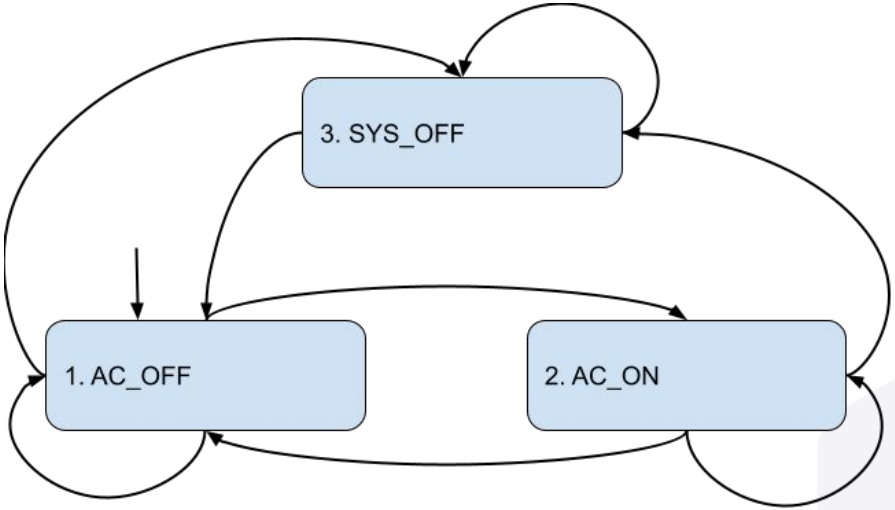
- Formal definition of model (today/next week)
- Formal definition of requirements (later)

Required or recommended by some safety standards at high SILs

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 SMs

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

FSM for system with indicator light -TT

Monday, November 14, 2022 10:42 AM

System with two switches and indicator light

On/off switch: if off, no response to light switch and light is off

Light switch: if system is on, light is on if switch is on and vice versa

FSM = (States, Inputs, Outputs, update, initialState)

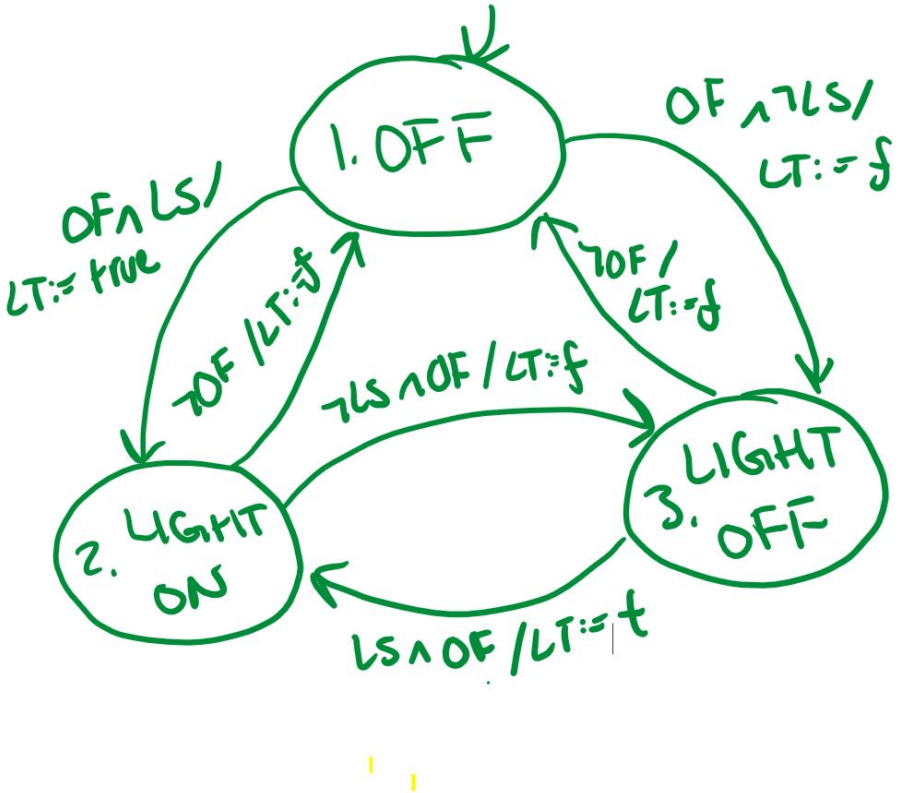
States = {OFF,LIGHT ON, LIGHT OFF}

Inputs = OF in {true, false}
LS in {true, false}

Outputs = LT in {true, false}

update(OFF, true, true) -> (LIGHT ON, true)
...
update(OFF, false, false) -> (OFF, false)

initialState = OFF



Valuation

(Lee/Seshia 3.3.3)

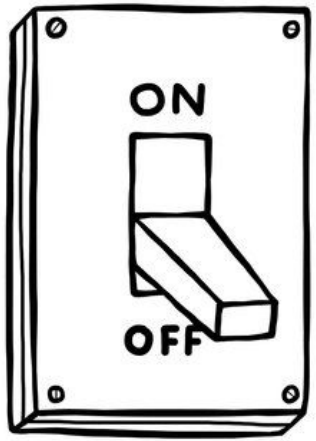
An FSM is a 5-tuple: (States, Inputs, Outputs, update, initialState)

- 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 {true, false, absent}

Present vs. Absent



For a *discrete* system, events for an input:

- Are discrete (separate, individual events)
- Happen in sequence

absent means “absence of an event,” while a value means “we received/sent this value”

changes depending on assumptions/implementation of model

Light switch: moment where it gets flipped to on or off

Push button as pure signal : event is the push of a button

FSM for system with indicator light -ET

Monday, November 14, 2022 10:42 AM

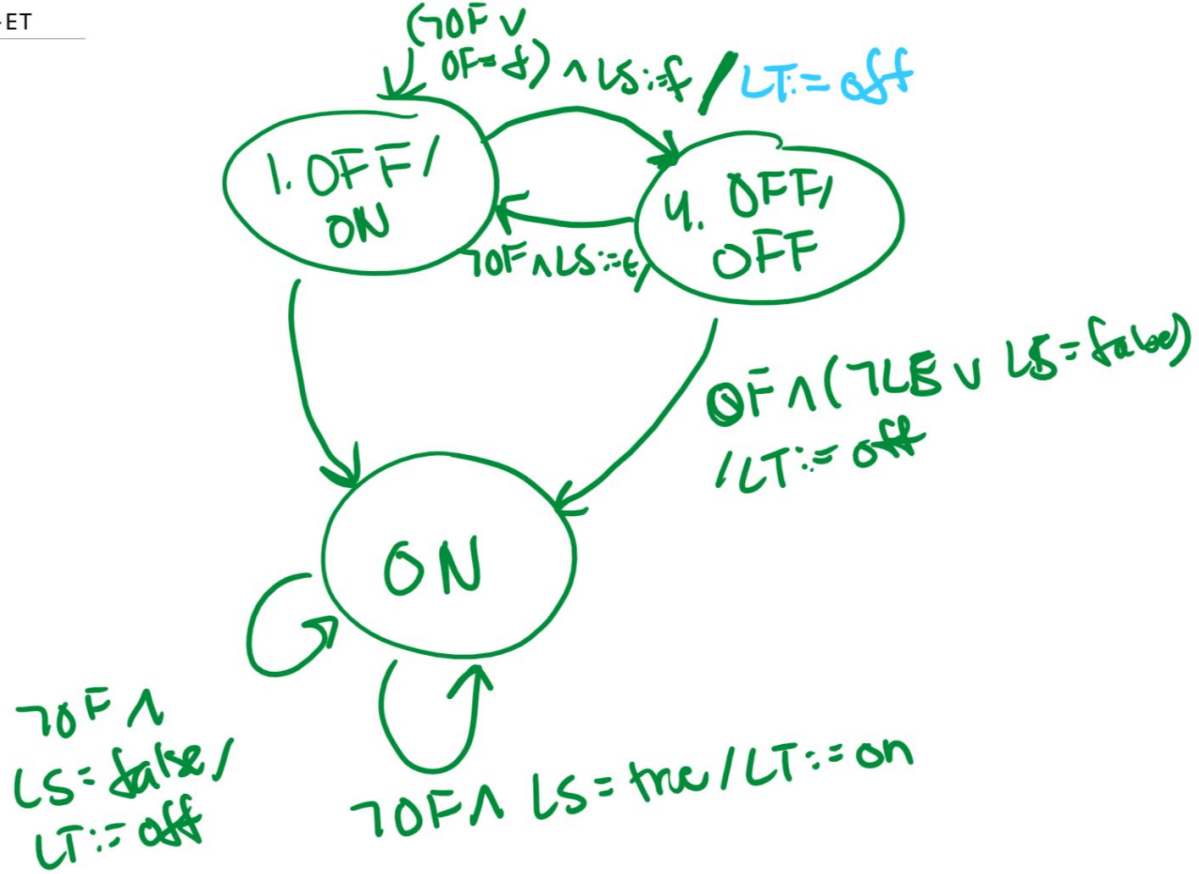
System with two switches and indicator light

On/off switch: if off, no response to light switch and light is off

Light switch: if system is on, light is on if switch is on and vice versa

OF = {true, false, absent}
LS = {true, false, absent}

LT = {on, off, absent}



Time/event triggered design

Purely time-triggered

(Our implementations are time-triggered)

We call updateFSM with all inputs periodically

All inputs are present for every update (event is the action of polling input)

Event-triggered

(Would get complicated to implement using our template)

Transition the FSM based on a change in input (e.g. interrupt)

Not all inputs may be present for every update