Advanced state modeling

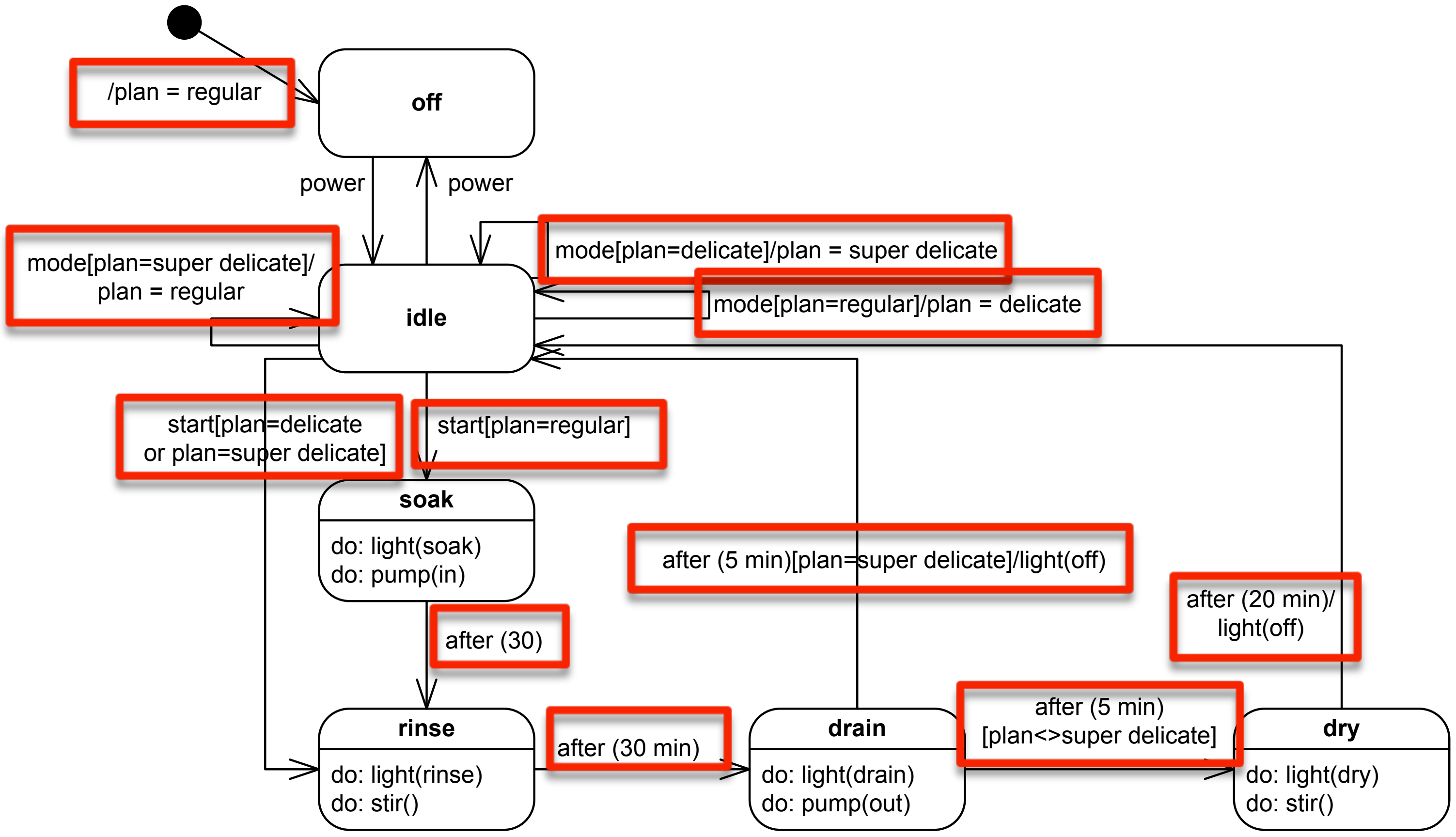
Marlon Dumas

Institute of Computer Science

How does a washing machine work?

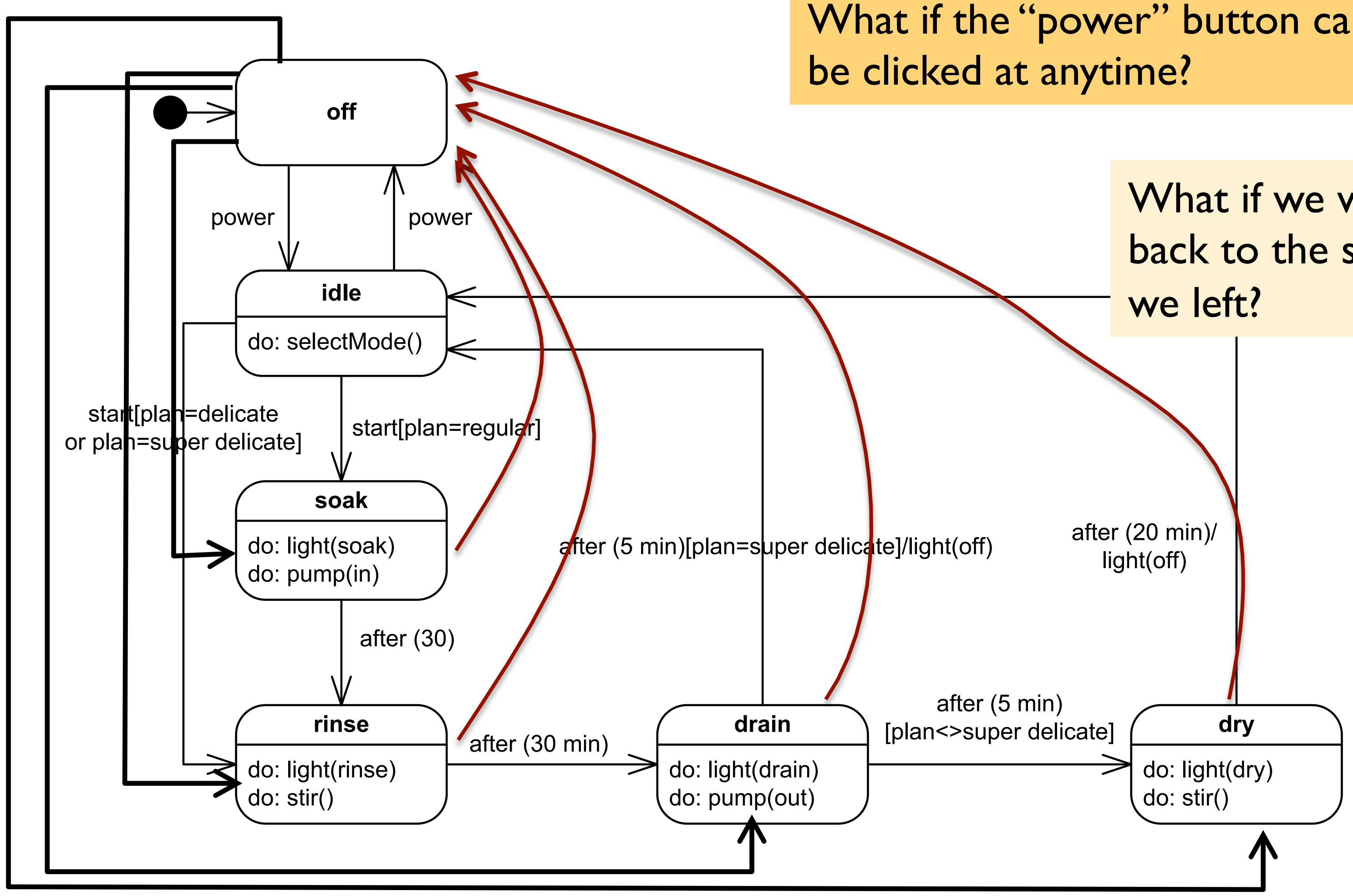
- On/off (power) button.
- Start button (no stop button!)
- Light indicates current stage
  - soaking, rinsing, draining, drying
- Three washing plans that can be changes using a “mode” button:
  - Regular
  - Delicate (no soaking)
  - Super delicate (no soaking, no drying)
- Off can be pushed only:
  - before starting
  - or after finishing

Statechart for the washing machine



/plan = regular
off
power
power
mode[plan=super delicate]/ plan = regular
idle
mode[plan=delicate]/plan = super delicate
mode[plan=regular]/plan = delicate
start[plan=delicate or plan=super delicate]
start[plan=regular]
soak
do: light(soak)
do: pump(in)
after (5 min)[plan=super delicate]/light(off)
after (20 min)/ light(off)
after (30)
after (30 min)
after (5 min) [plan<>super delicate]
rinse
do: light(rinse)
do: stir()
drain
do: light(drain)
do: pump(out)
dry
do: light(dry)
do: stir()

State explosion and history

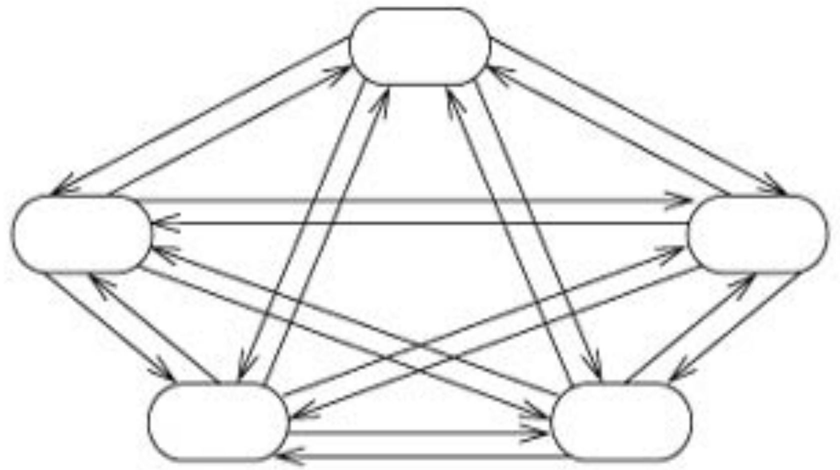


What if the “power” button can be clicked at anytime?

What if we want to come back to the same state we left?

State explosion

- If we have "n" classes with "m" (boolean) attributes each (let's assume that all classes have the same number of attributes)
  - Possible states of the whole system = 2^{nm}



Abstraction in Statecharts

Factor out common behavior

↳ **Composite States**

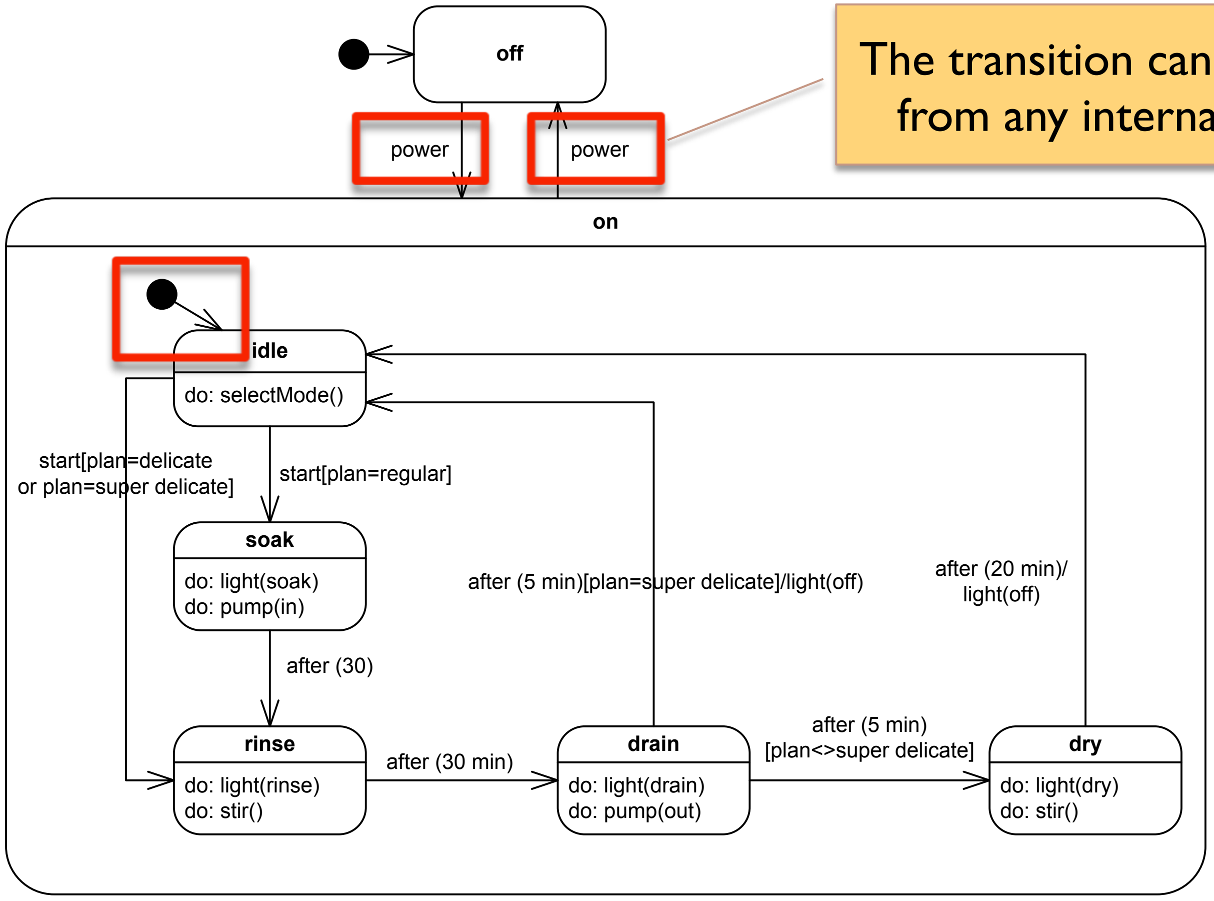
Remember history

↳ **History pseudo-states**

Segregate independent behavior

↳ **Orthogonal/ Parallel States**

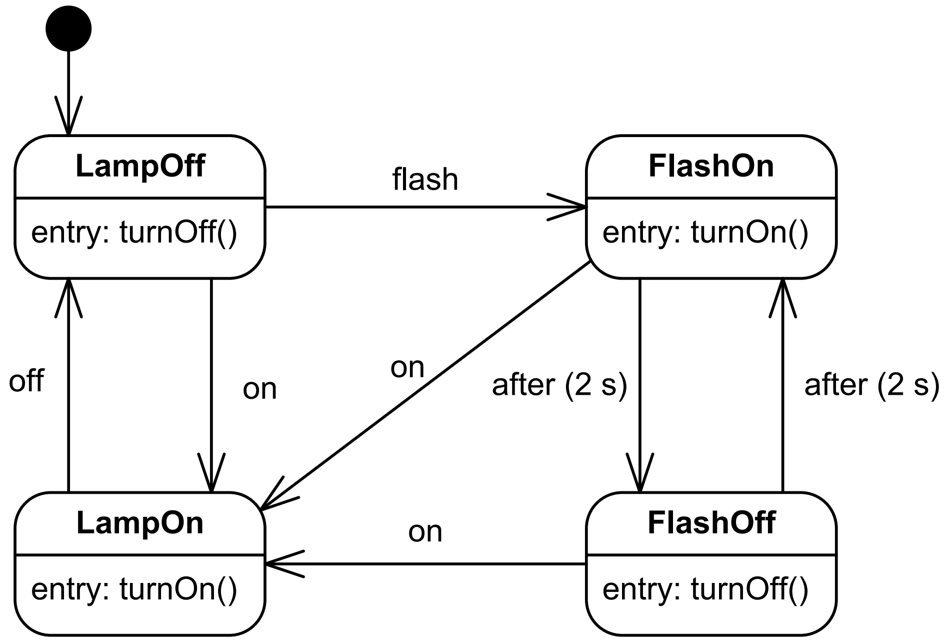
Composite states



The transition can be fired from any internal state

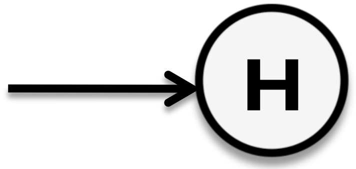
Exercise 1

- Group "FlashOn" and "FlashOff" states into a composite state "Flashing"

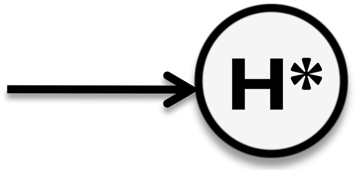


History pseudo-state

- Return to a previously visited hierarchical state
- Shallow history: just the current level

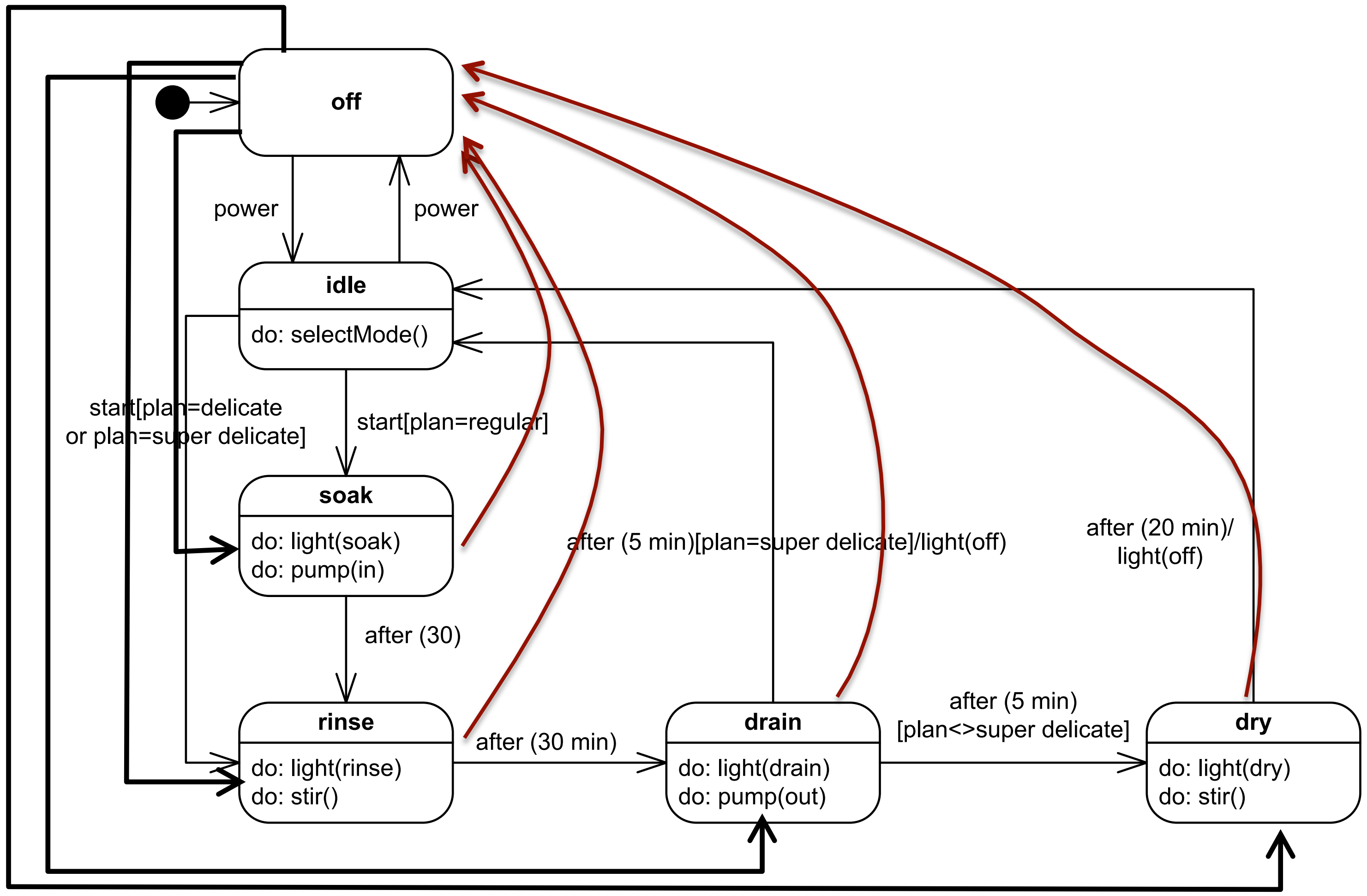


- Deep history: includes all nested states

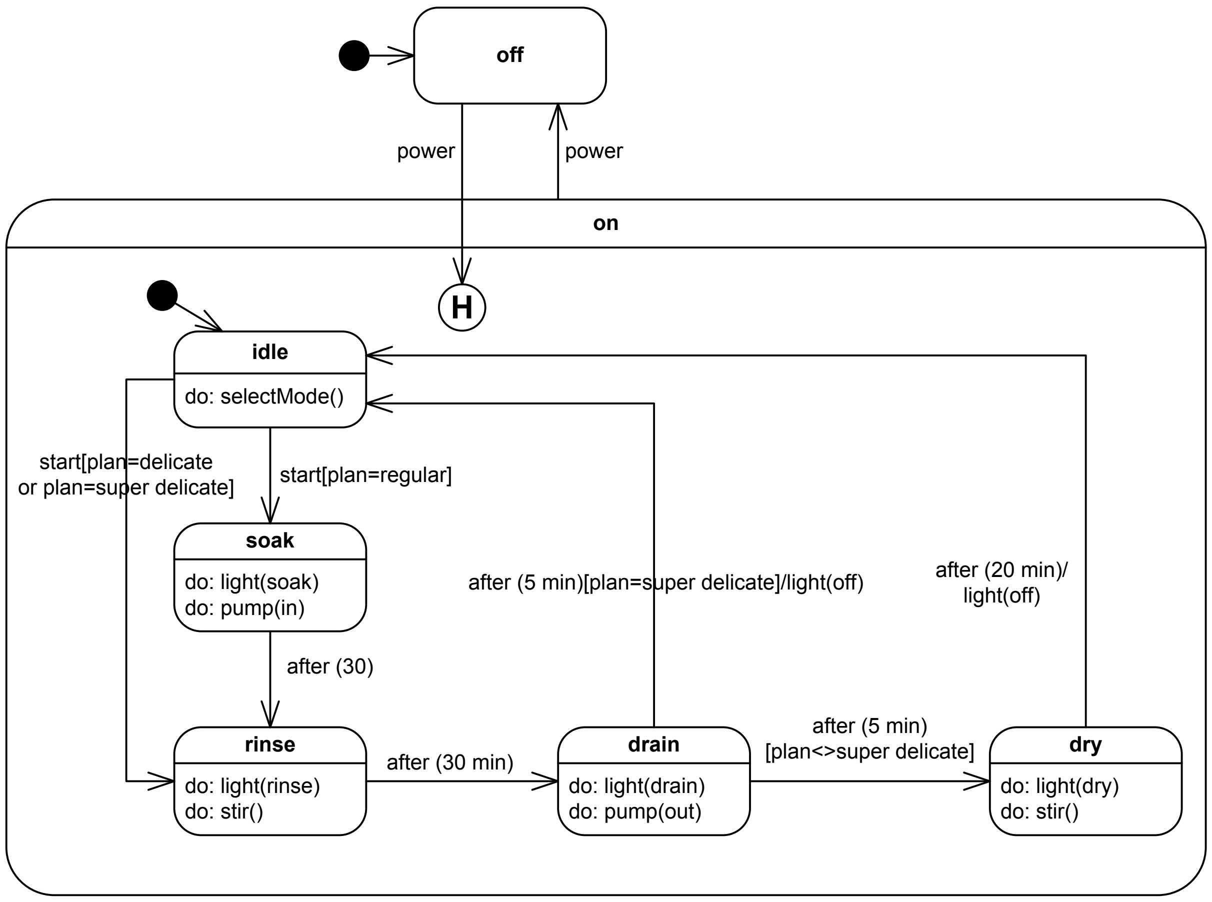


- Sometimes it is useful to clear history:
  - clear-history(state) clh(state)
  - clear-history(state*) clh(state*)

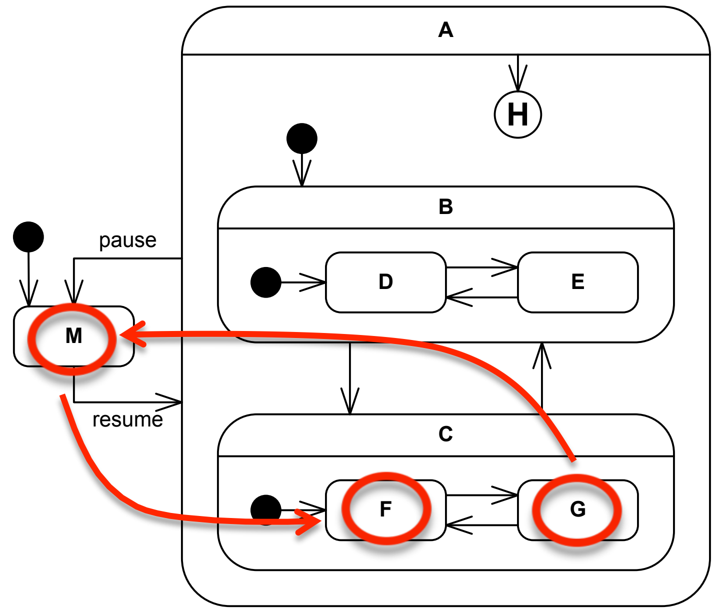
Back to the washing machine…

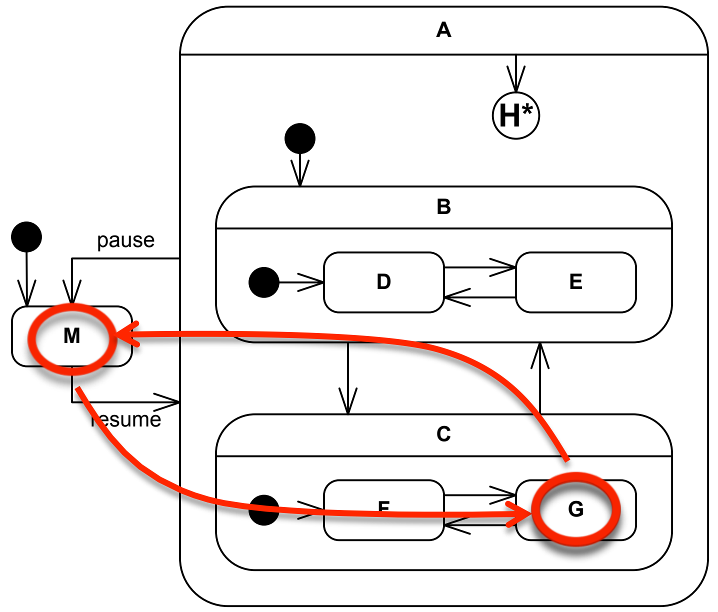


Washing machine with “history”



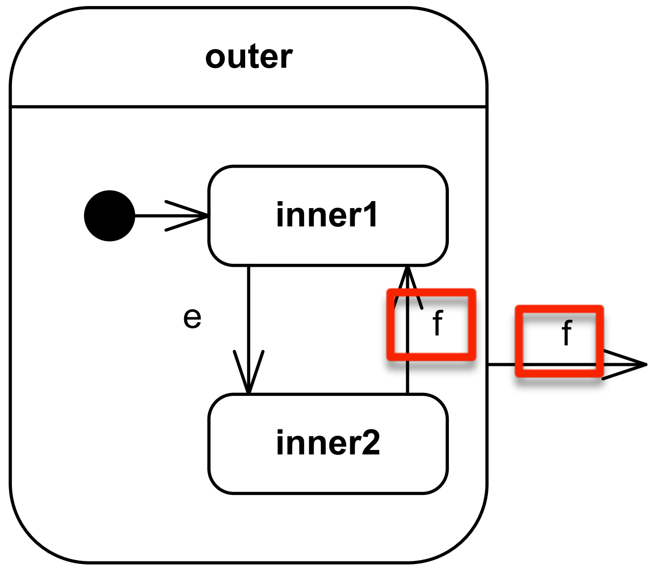
Shallow vs. Deep history





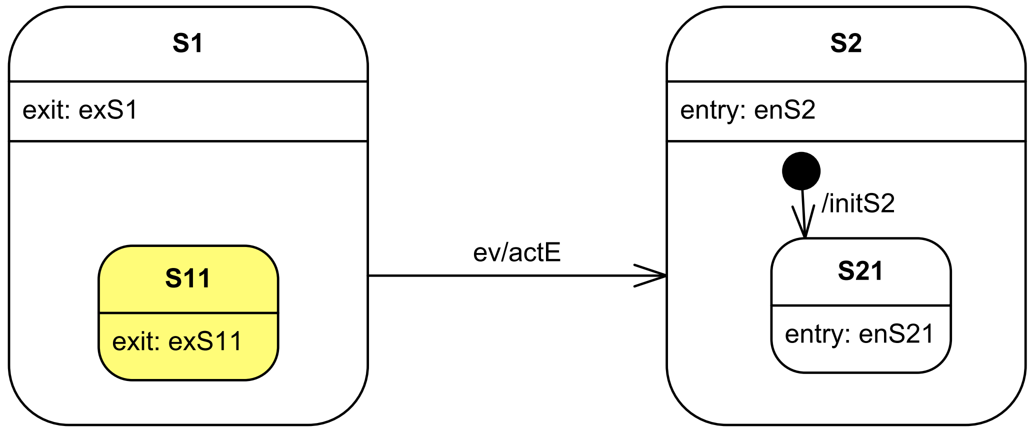
Note on transition precedence

- Two or more transitions may have the same event trigger
  - inner transition takes precedence
  - if no transition is triggered, event is discarded



Order of activities in nested models

- Same approach as for the simple case

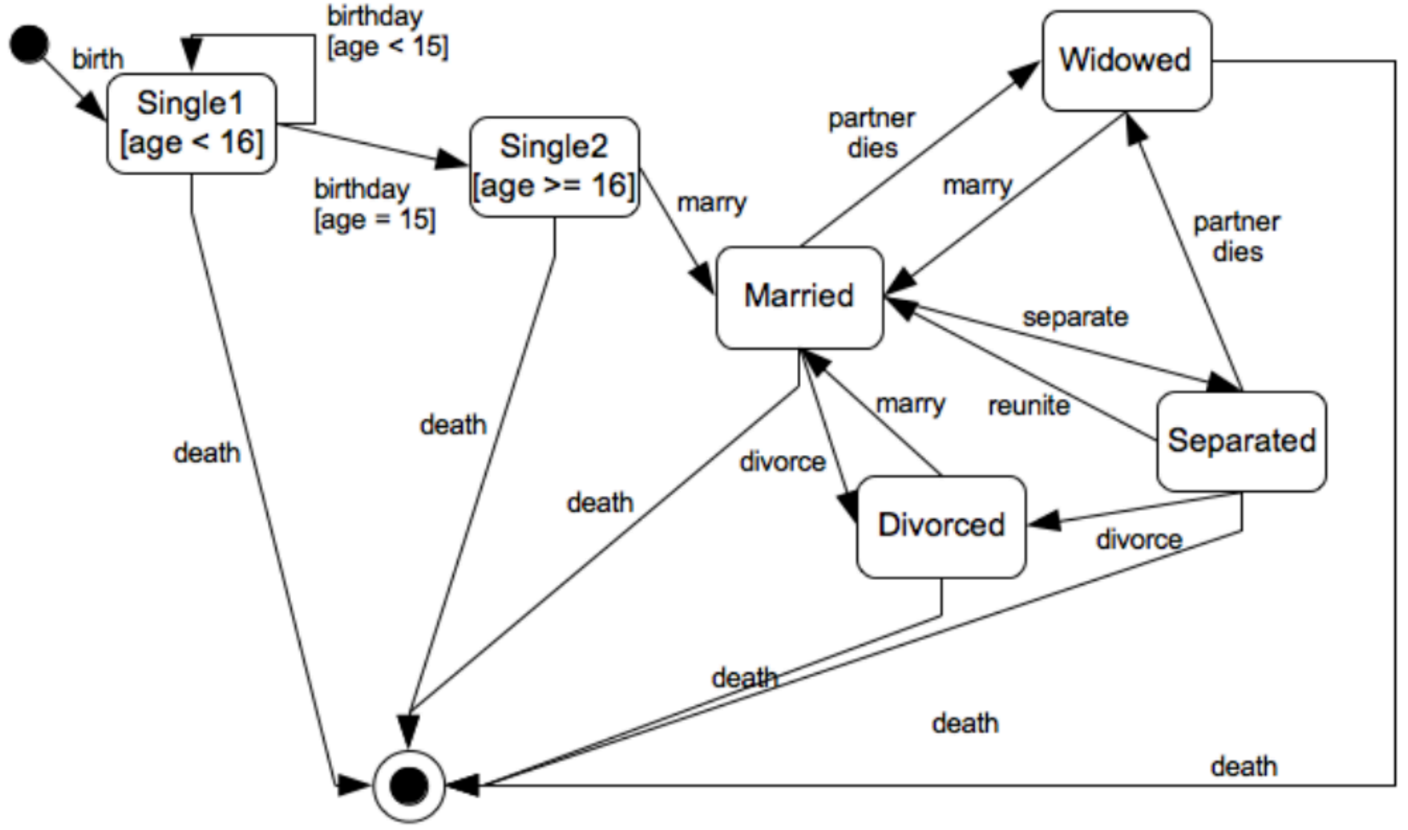


Execution sequence:

exS11 ⇨ exS1 ⇨ actE ⇨ enS2 ⇨ initS2 ⇨ enS21

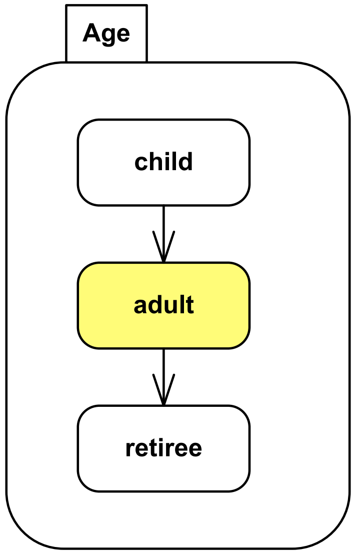
Exercise 2

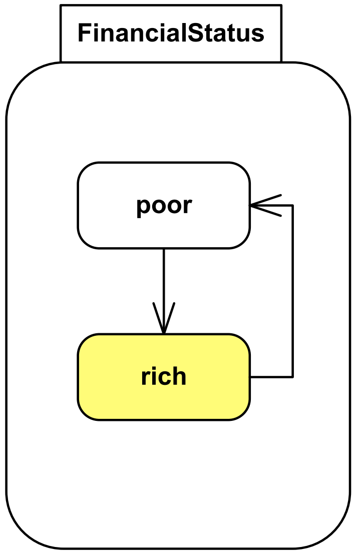
- Fix and simplify this state machine



Independent behavior

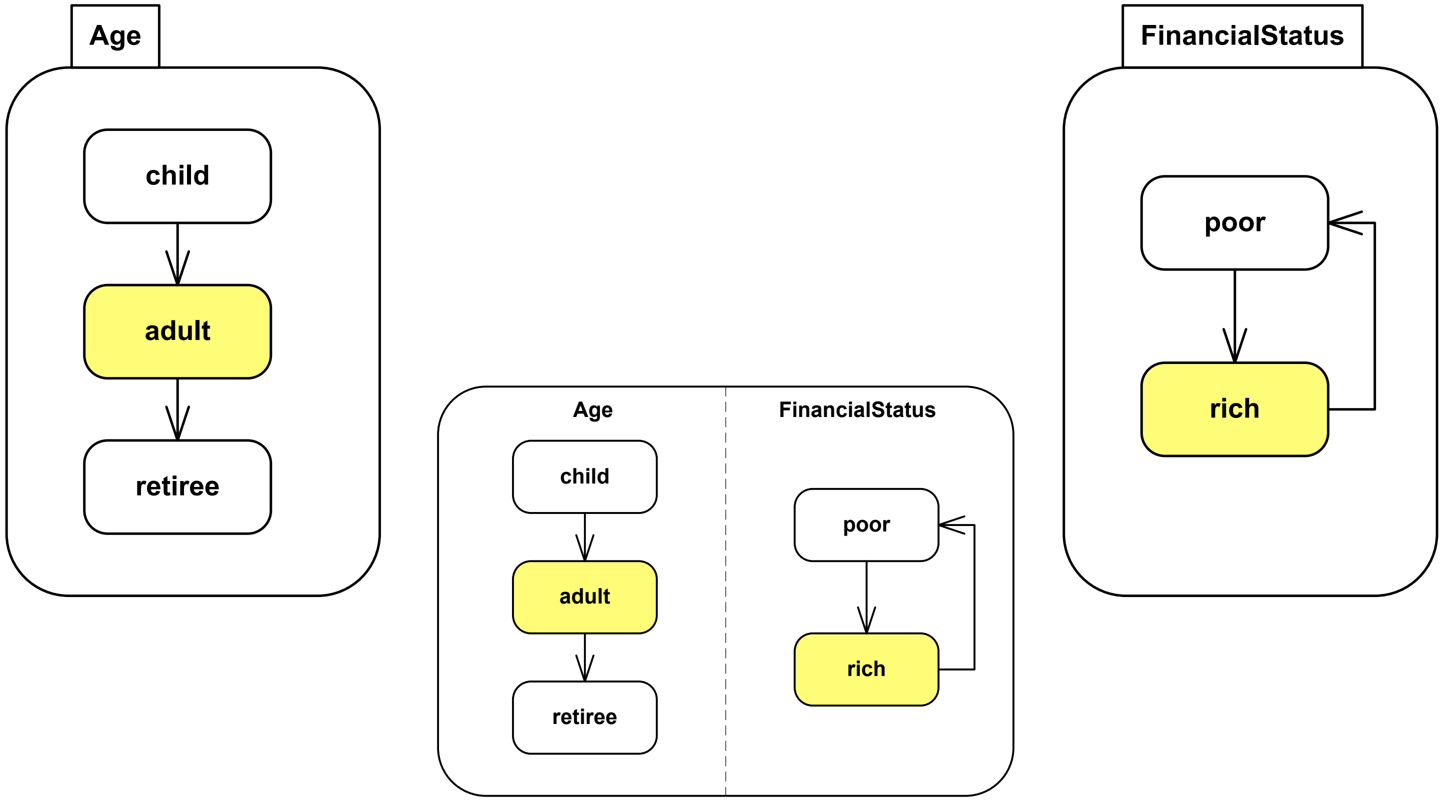
- Multiple simultaneous perspectives on the same entity





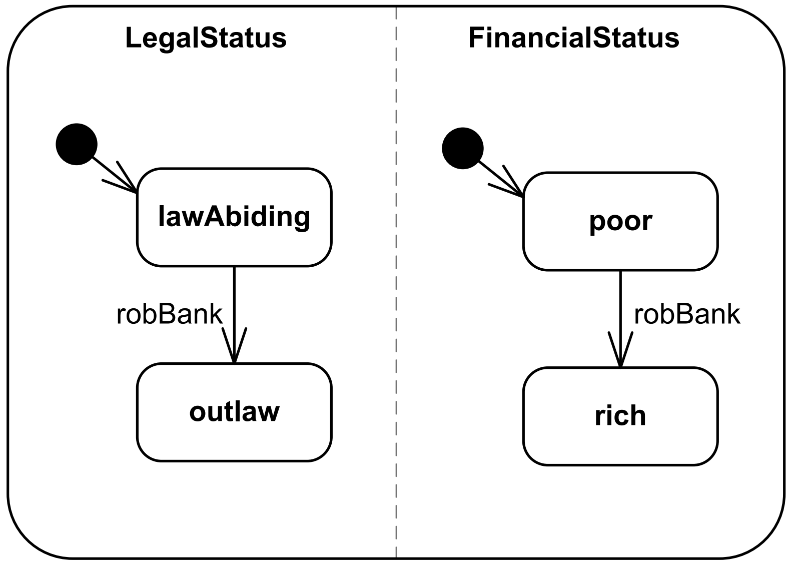
Parallelism: States with orthogonal regions

- Combine multiple simultaneous descriptions



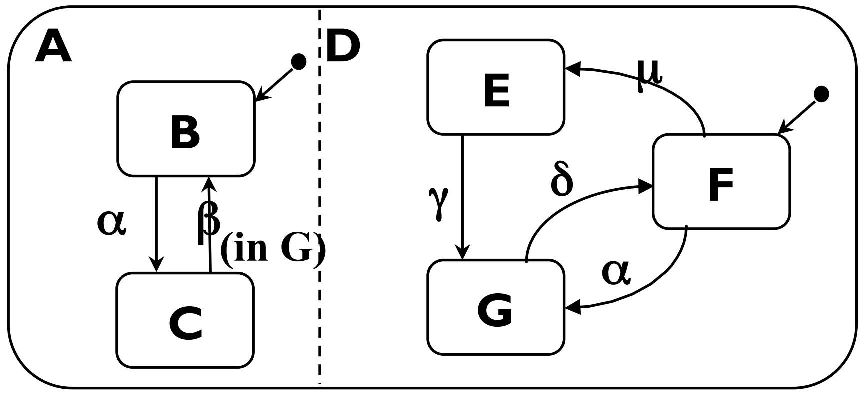
Parallelism: States with orthogonal regions

- All mutually orthogonal regions detect the same events and respond to them "simultaneously"
  - usually reduces to interleaving of some kind

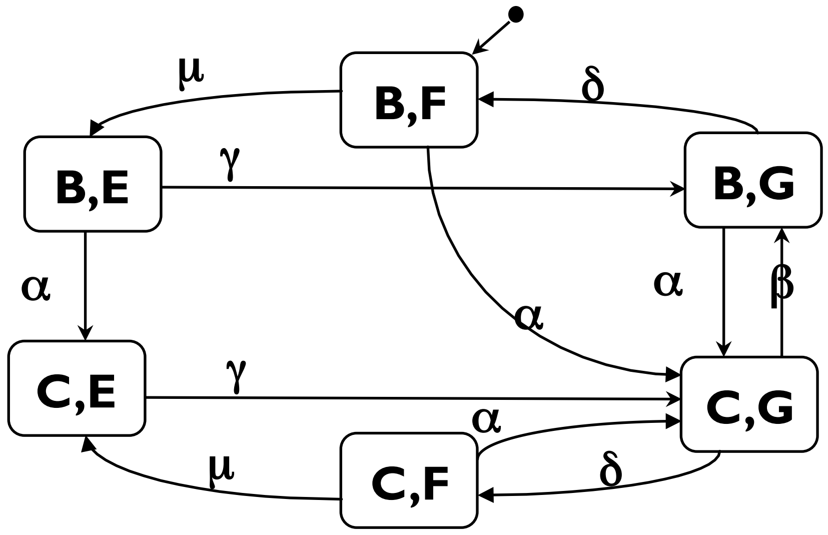


“Flat” vs. Parallel State Machines

- Every parallel machine can be transformed into a sequential machine:

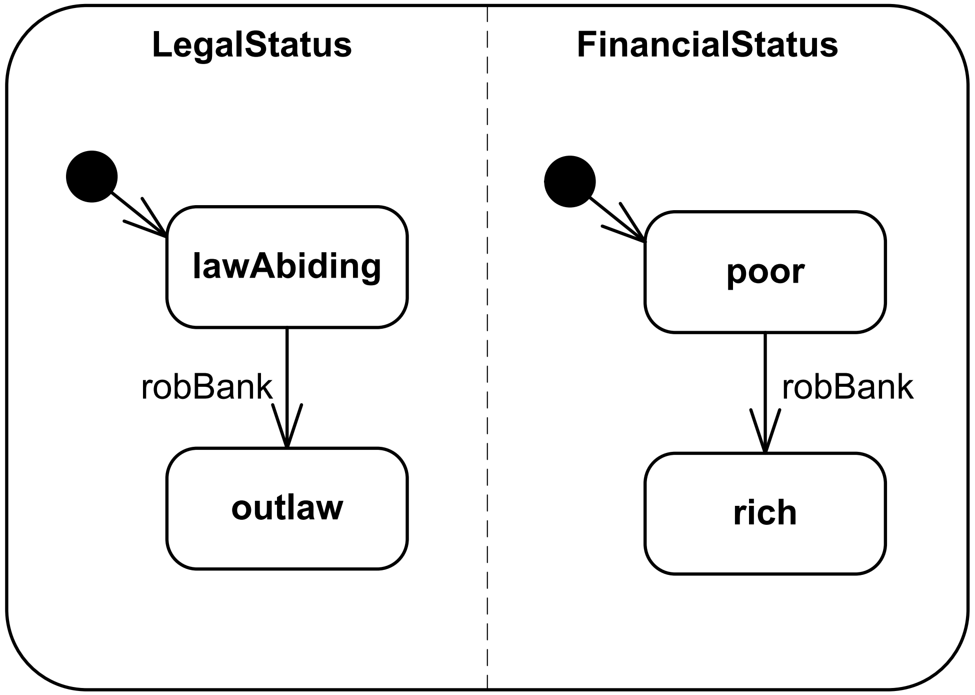


With Orthogonal Regions



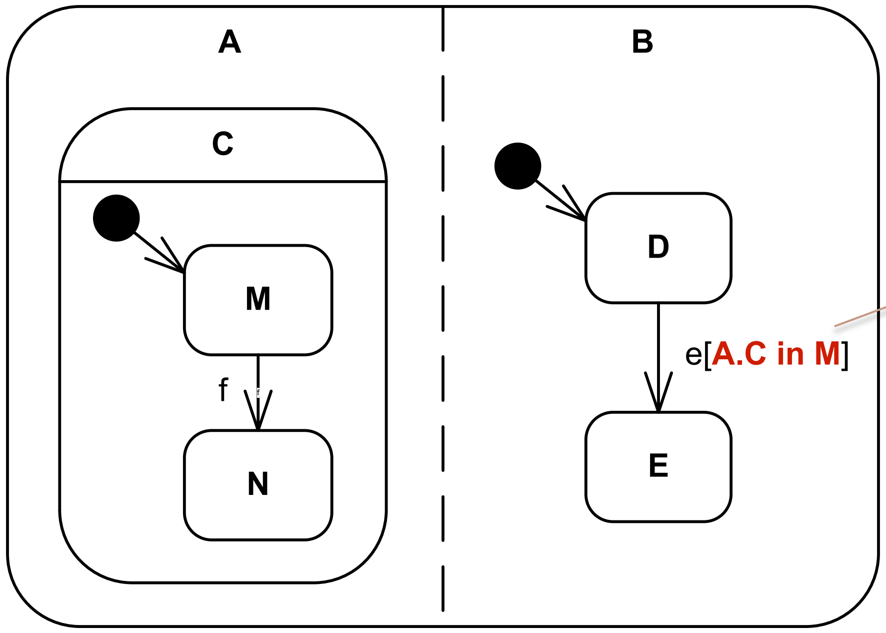
Without Orthogonal Regions

Exercise 3: Rewrite this without parallel regions



Synchronization

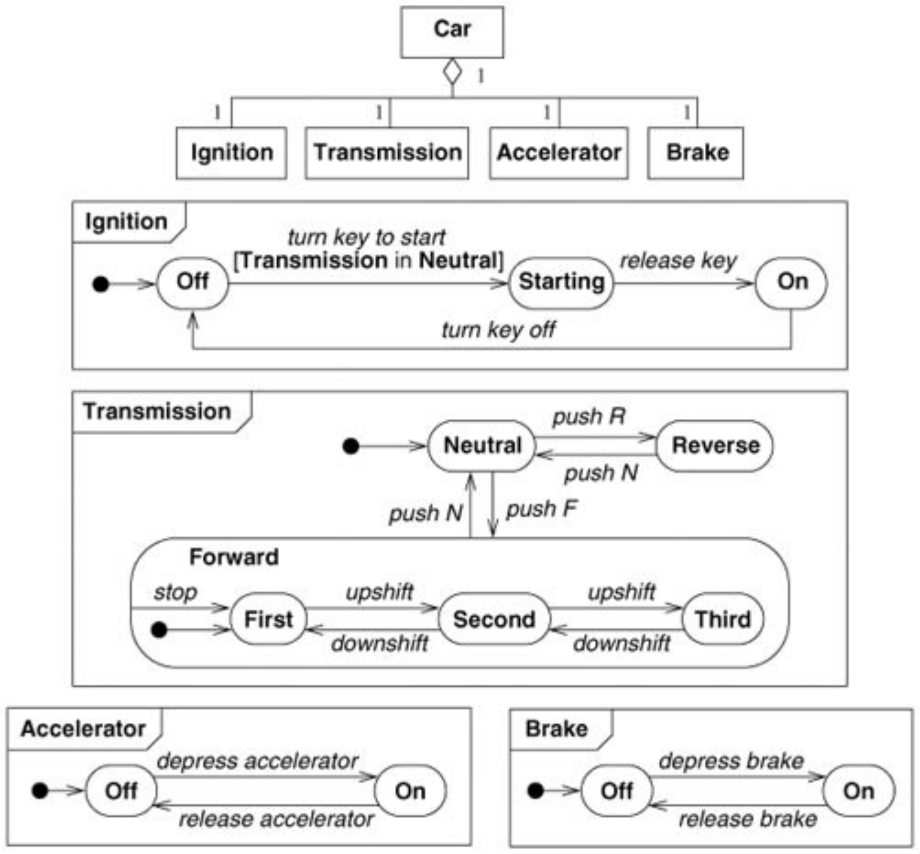
- Orthogonal regions/states can be synchronized via transition guards



This transition can only be fired when A.C is in M state

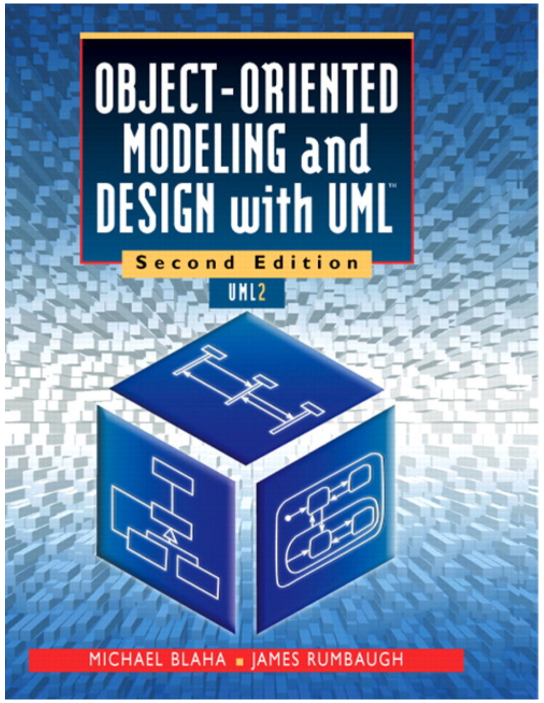
Class aggregation and their state diagrams

- A state diagram is a collection of state diagrams
  - Class aggregation will usually require to combine the state diagrams of all parts
- The whole can be thought as a set of orthogonal regions!



Readings & Resources

- Last week: Blaha & Rumbaugh, Chapter 5
- **This week:** Blaha & Rumbaugh, Chapter 6


OBJECT-ORIENTED MODELING and DESIGN with UML™
Second Edition
UML2
MICHAEL BLAHA ▪ JAMES RUMBAUGH