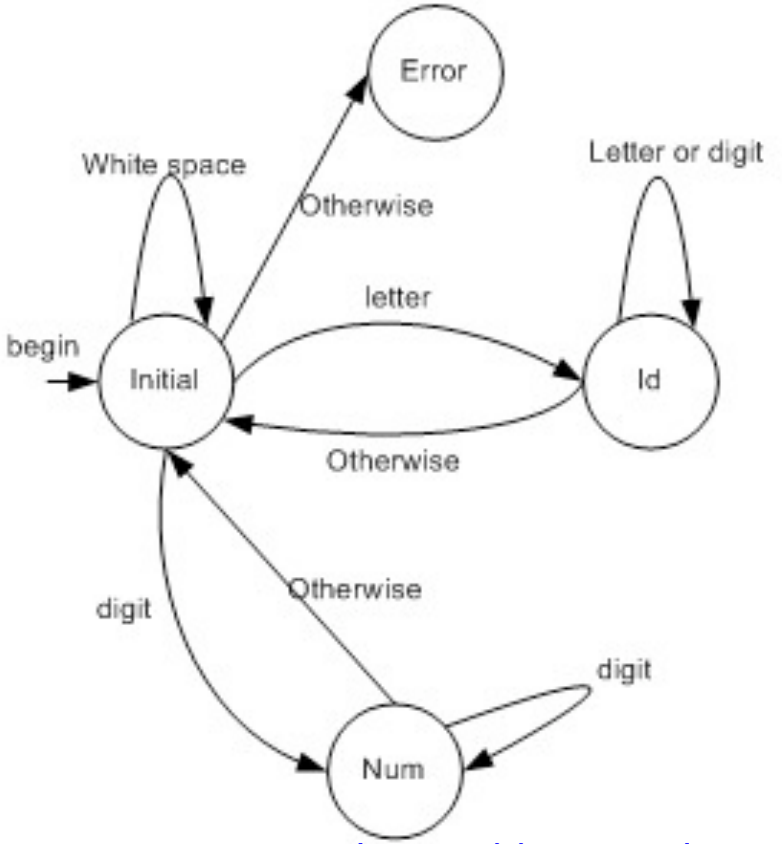
CS445 / ECE451 / CS645 / SE463

Software Requirements Specification & Analysis

State Machine Models

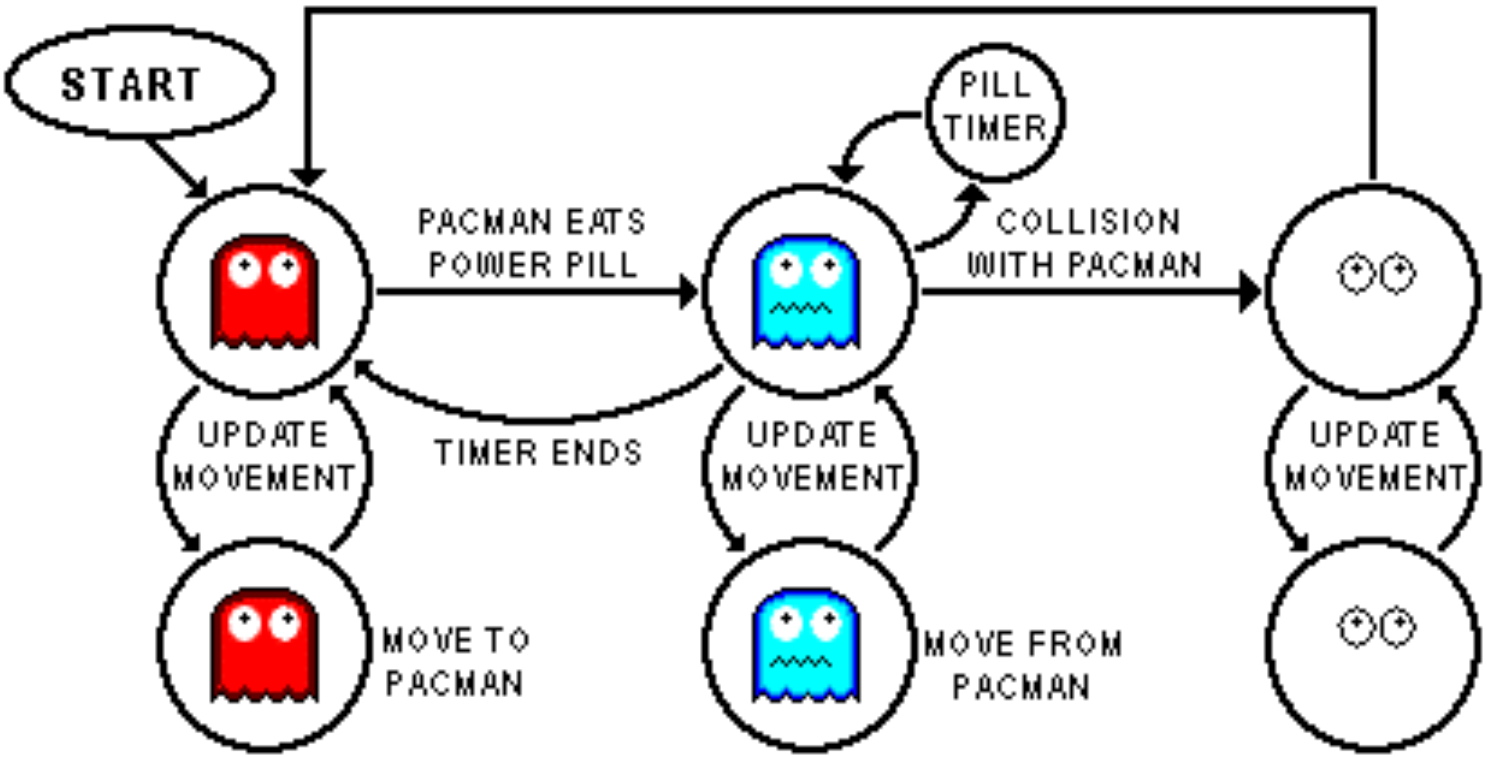
Finite State Machines



This FSM recognizes a token stream of identifiers and numbers

http://www.codeproject.com/KB/recipes/Parser_Expression.aspx

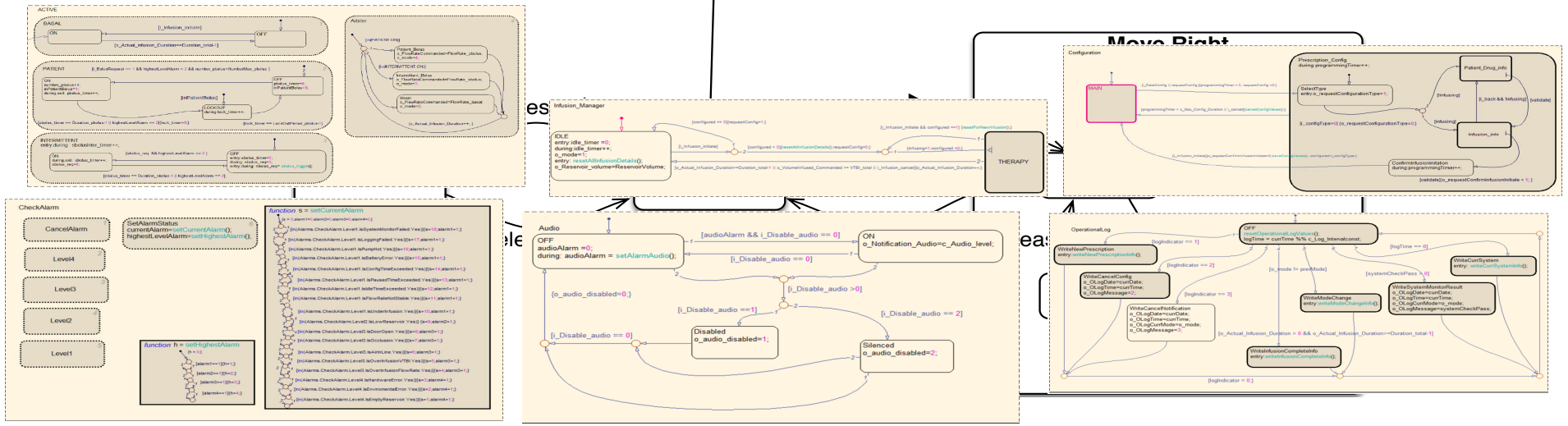
State Machines



State Machines in Practice

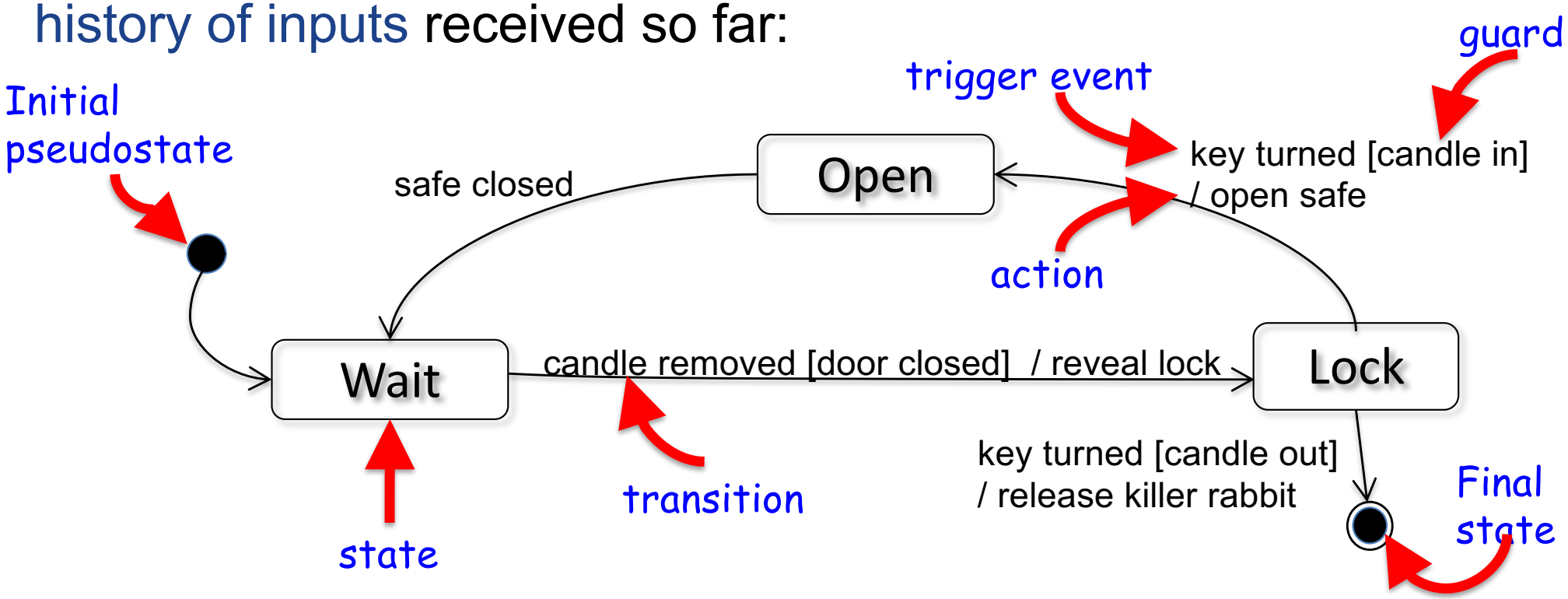
Used to understand complex behaviour

- multiple sequences, cases, conditions
- agent's AI (e.g., in games)
- critical software to be certified / verified



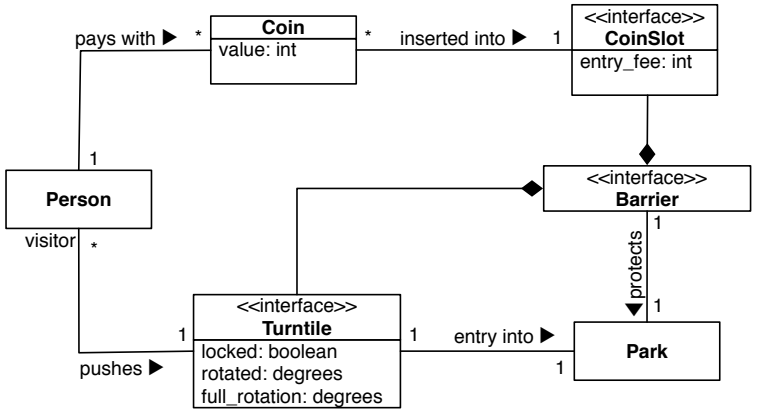
UML State Machines

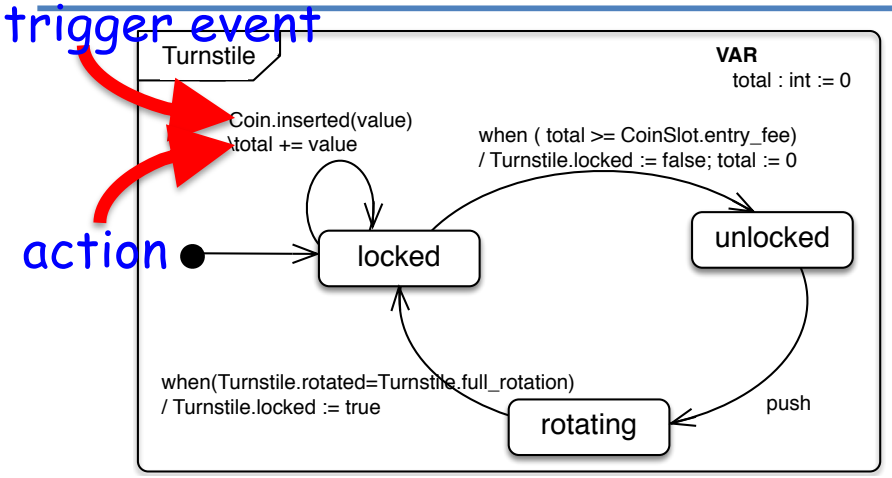
A UML state machine models behaviour that depends on the history of inputs received so far:



Fowler, UML Distilled,3ed, p.76

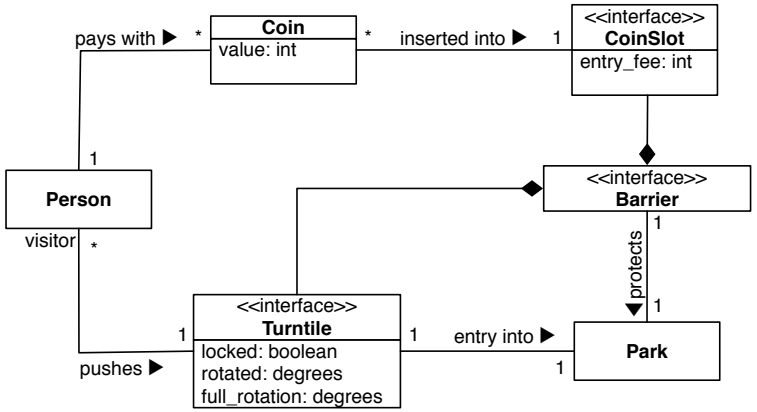
UML State Machines

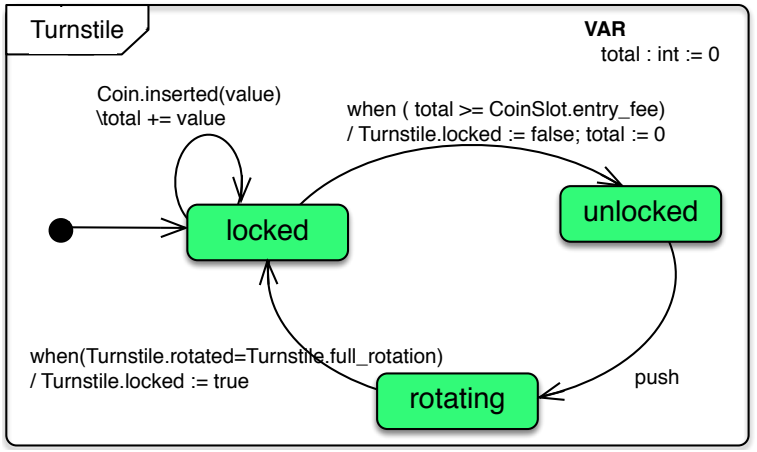




- **Inputs** are events and conditions on ≪interface≫ phenomena
- **Outputs** are actions on ≪interface≫ phenomena
- **States** represent equivalence classes of input traces
- State machines can have internal variables

States



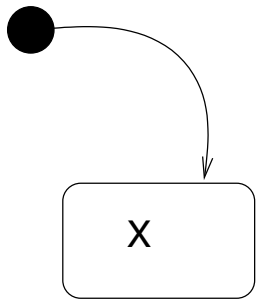


States partition system behaviour into distinct **modes of operation**

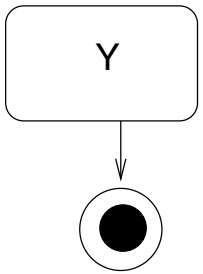
- equivalence classes of input traces
- equivalence classes of future behaviours
- distinct set of input events of interest
- distinct reactions to input events
- internal processing of input

States and Pseudo-states

- There always needs to be a designated starting/initial state.
- The designator of an initial state is a *pseudo-state.*
  - A pseudo-state is NOT a real state (no time is spent there)
  - Later, we will see the History pseudo-state



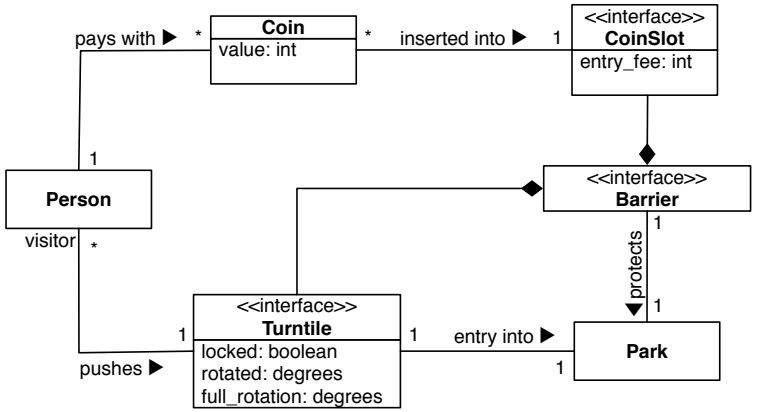
X is the initial State

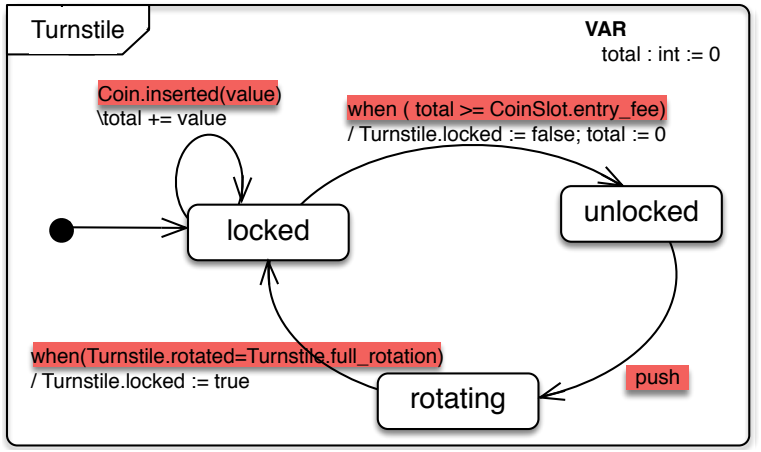


Bullseye is a final state

Often there is a designated final state. This is a real state.

Inputs





An **event** is some noteworthy ***change*** in the environment

- input message from the environment
- some change to ≪interface≫ phenomena
- passage of time

A **condition** is a ***persistent*** Boolean expression over

- ≪interface≫ phenomena
- event parameters
- state-machine variables

Time event

A time event is the occurrence of a specific date/time or the passage of time.

- Absolute time:
  - at (12:12 pm, 12 Dec 2012)
- Relative time:
  - after (10 seconds since exit from state A)
  - after (10 seconds since x)
  - after (20 minutes) // since the transition's source state was entered

Change event

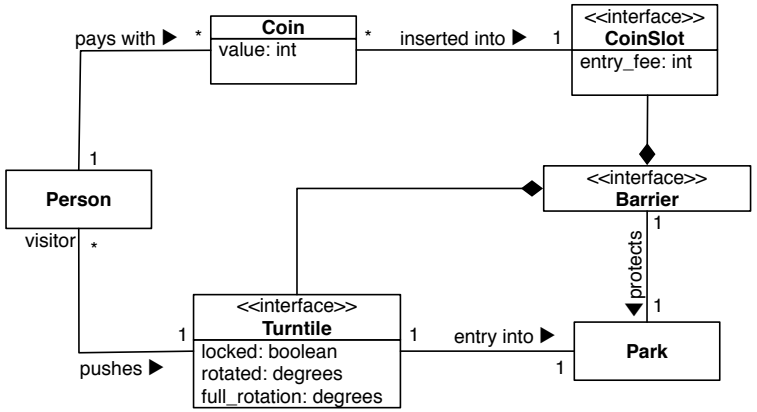
A change event is the event of a condition becoming true.

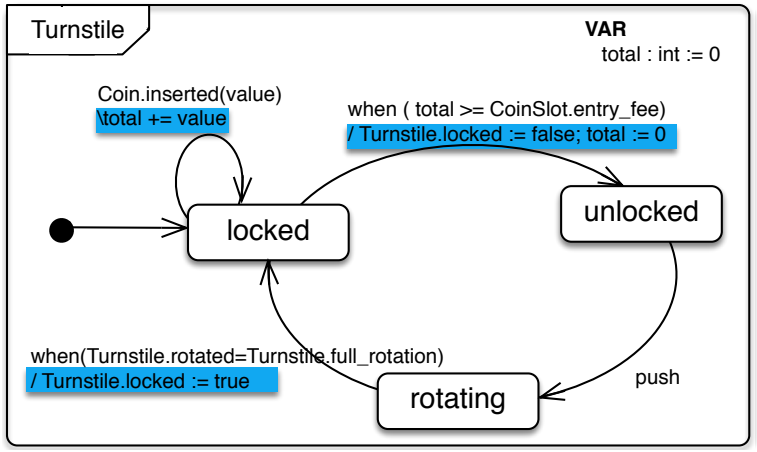
- The event "occurs" when the condition changes value from *false* to *true*.
  - when (temperature > 25 degrees)
  - when (on)
- The event does not reoccur unless the value of the condition becomes *false* and then returns to *true*.

when(X) vs. [X]

- A change event is what you want if the system is in a state waiting for a condition to become true
- A transition with a guard but no triggering event
  - UML semantics:
    - guard is checked once (when the source state is entered)
    - transition fires if the guard is true when checked
    - guard is not checked again (until source state is next entered)
  - not likely what you want to say

Actions





Actions are the system's response to an event

- output message
- some change to ≪interface≫ phenomena
  - e.g., Turnstile.locked := true

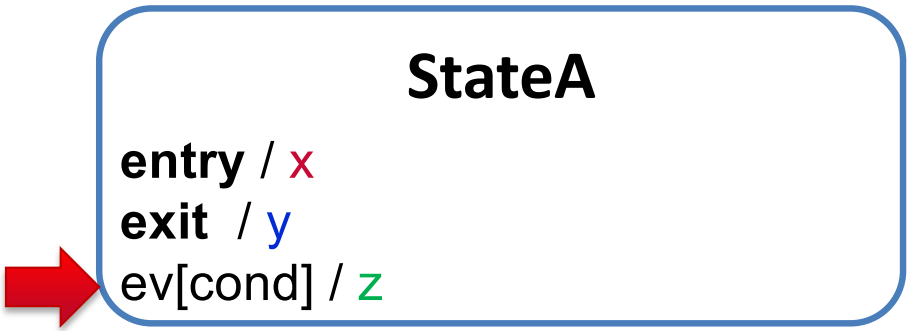
An action is non-interruptible (i.e., atomic)

State Actions

States can also be annotated with entry or exit actions, and with internal actions.

- Entry actions: – actions that occur every time the state is entered by an explicit transition.
- Exit actions: – actions that occur every time the state is exited by an explicit transition.
- Internal **actions**: on events

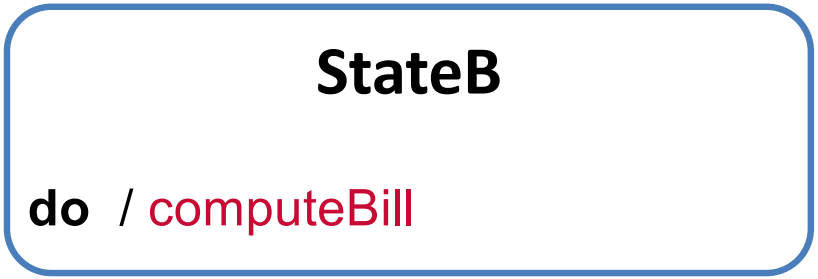
An action is uninterruptible.



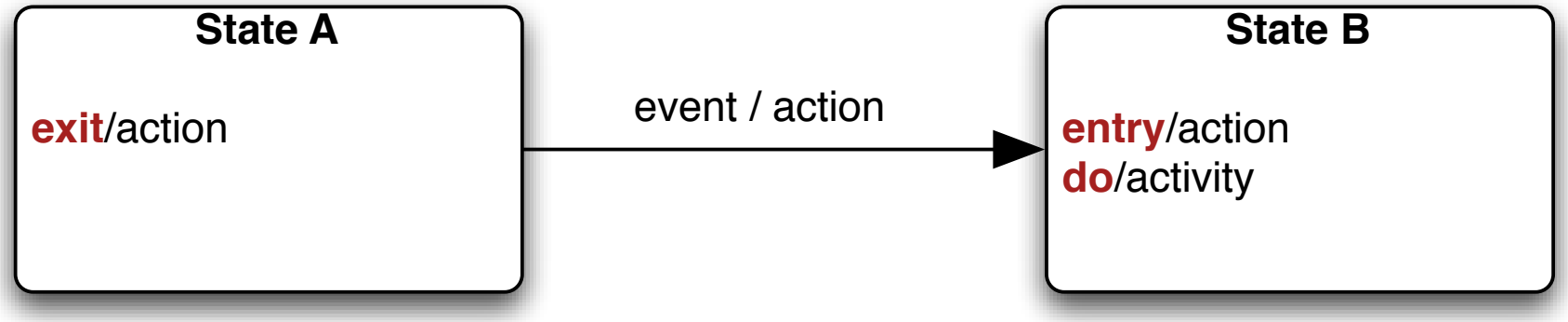
State Activities

An activity is computation of the system that takes time, and can be interrupted.

- c.f., an action, which is uninterruptible
- An activity may be associated with a state.
- States with activities are called *activity states*.



Execution Order

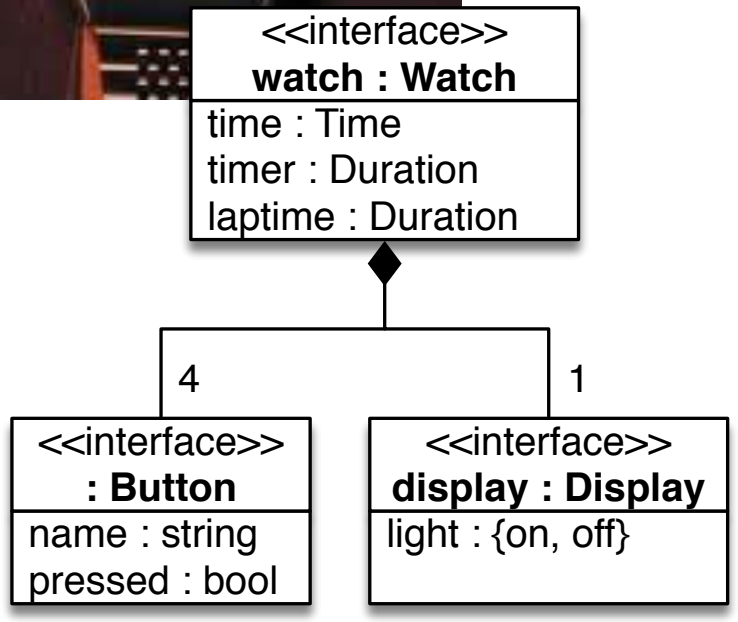


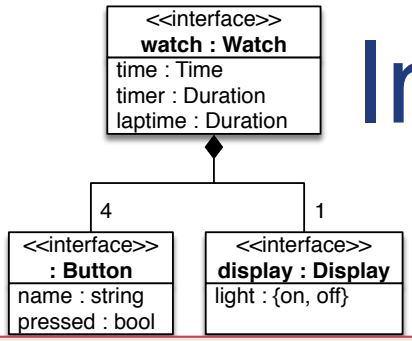
In an explicit transition (including self-looping transitions!), the order of effects is:

1. the exit action of the source state, then
2. transition actions (in listed order), then
3. the entry action of the destination state, then
4. the state activity

Stopwatch Example

Input	Meaning of Input Command
on	Turn watch on
off	Turn watch off
mode	Toggle between watch and stopwatch
but2 [watch]	Toggle between 12h and 24h display
but2 [stopwatch]	Pause/ resume timer
but3 [watch]	Turn light on for 3 sec
but3 [stopwatch, timer running, display timer]	Record laptime; display laptime; turn light on for 3 sec
but3 [stopwatch, timer stopped, display timer]	Reset timer; turn light on for 3 sec
but3 [stopwatch, timer running, display laptime]	Display timer; turn light on for 3 sec





Inputs and Outputs

Event Macros

on = when (On/Off:Button.pressed)
off = when (On/Off:Button.pressed)
mode = when (mode:Button.pressed)
but2 = when (but2:Button.pressed)
but3 = when (but3:Button.pressed)

Domain Model Variable Macros

time = watch.time
timer = watch.timer
laptime = watch.laptime
light = display.light

Vars

timerOn: bool

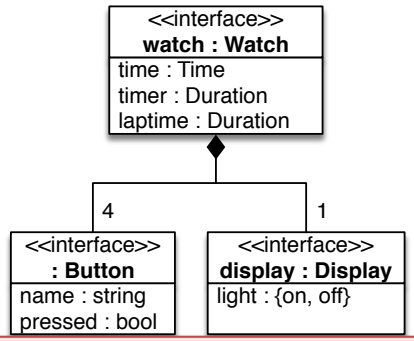
Inputs

- On
- Off
- Mode
- But2
- But3

Outputs

- Display nothing
- Display 12 hr time
- Display 24 hr time
- Display timer
- Display lap time (duration: Time)
- Turn on light (duration: Time)

Stopwatch Example



Event Macros

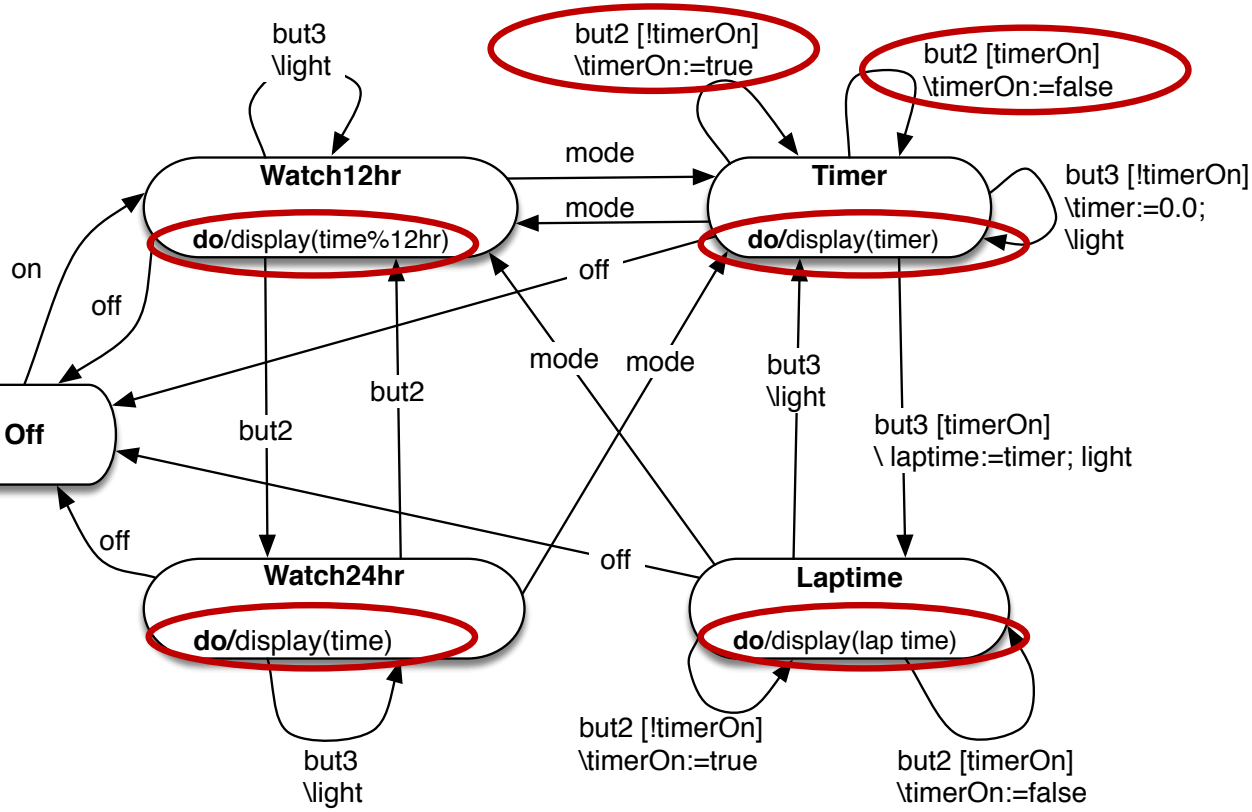
on = when (On/Off:Button.pressed)
off = when (On/Off:Button.pressed)
mode = when (mode:Button.pressed)
but2 = when (but2:Button.pressed)
but3 = when (but3:Button.pressed)

Domain Model Variable Macros

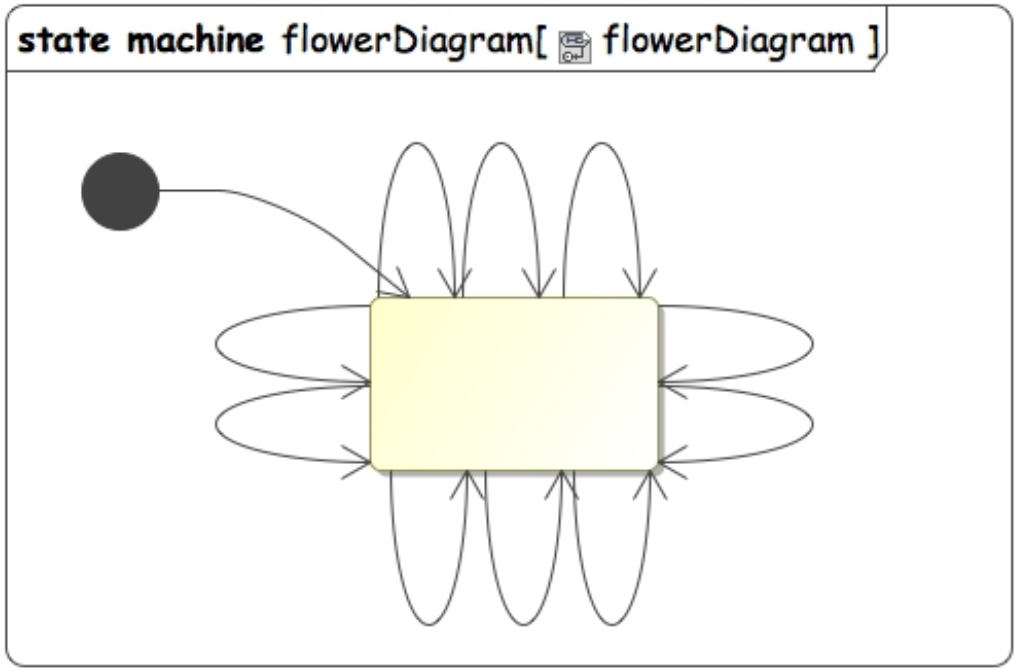
time = watch.time
timer = watch.timer
laptime = watch.laptime
light = display.light

Vars

timerOn: bool



“Flower Diagram”



State Machine vs. Scenarios

State machine diagrams	Scenarios
specifies behaviour	*Illustrates* behaviour
all allowable scenarios	one allowable scenario, showing end-to-end behaviour (better feel for overall system behaviour)
developer oriented	customer oriented
identifies system states, which represent equivalent input histories	
	can help developer validate state diagrams

References

Craig Larman, *Applying UML and Patterns, 3ed.,* Prentice Hall, 2004.

- Chapter 23: “UML State Machine Diagrams and Modeling”

Lenny Delligatti, *SysML Distilled: A Brief Guide to the Systems Modeling Language*, Addison-Wesley Professional, 2013.

- Chapter 8: “State Machine Diagrams”