Object Modeling with OMG UML Tutorial Series

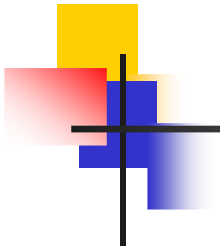
Behavioral Modeling

Gunnar Övergaard, Bran Selic and
Conrad Bock

UML Revision Task Force
November 2000

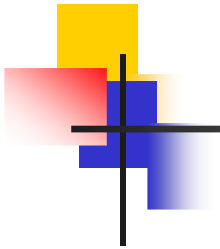
Part 2: Statecharts

Bran Selic, Rational Software



Behavioral Modeling

- Part 1: Interactions and Collaborations
- Part 2: Statecharts
- Part 3: Activity Diagrams

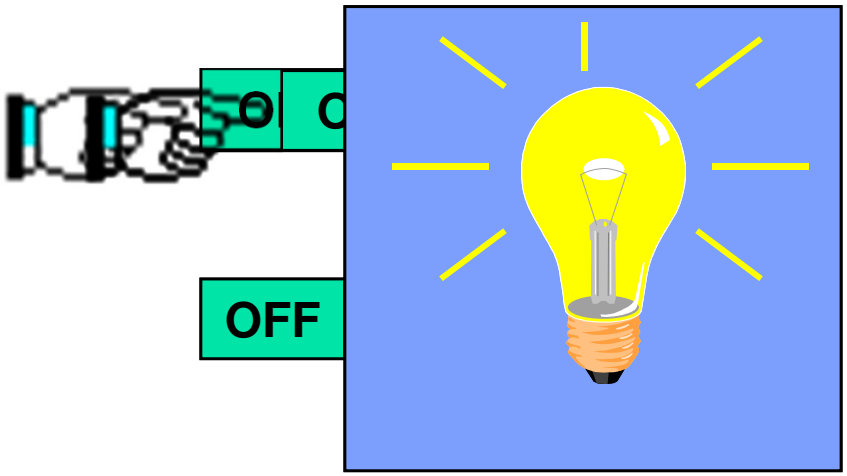


Overview

- Basic State Machine Concepts
- Statecharts and Objects
- Advanced Modeling Concepts
- Case Study
- Wrap Up

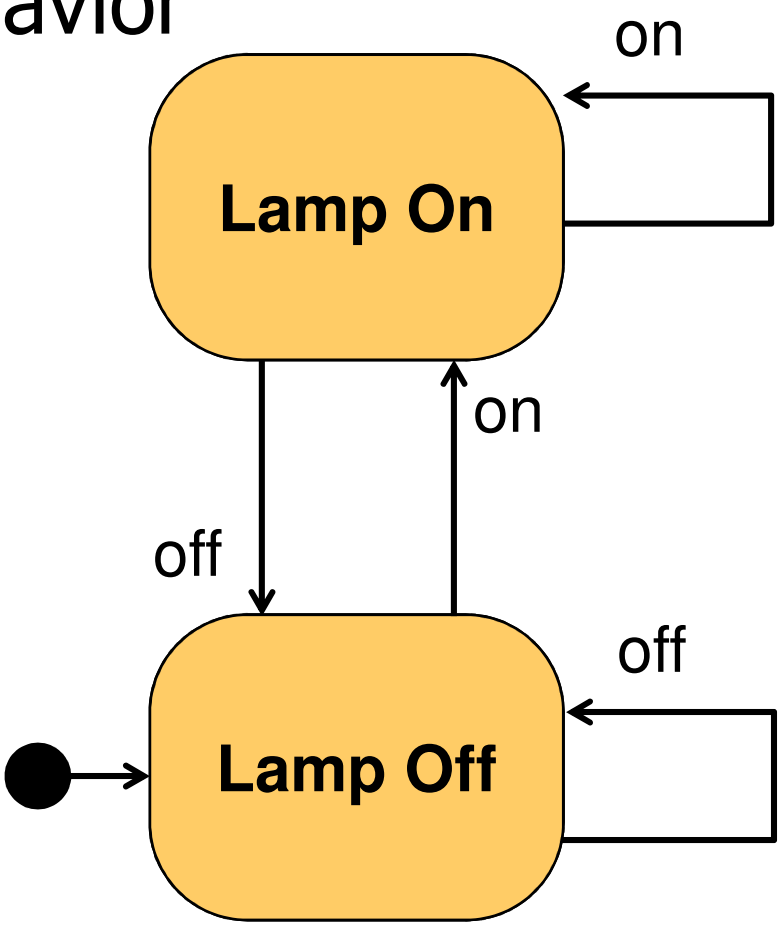
Automata

- A machine whose output behavior is not only a direct consequence of the current input, but of some past history of its inputs
- Characterized by an internal state which represents this past experience



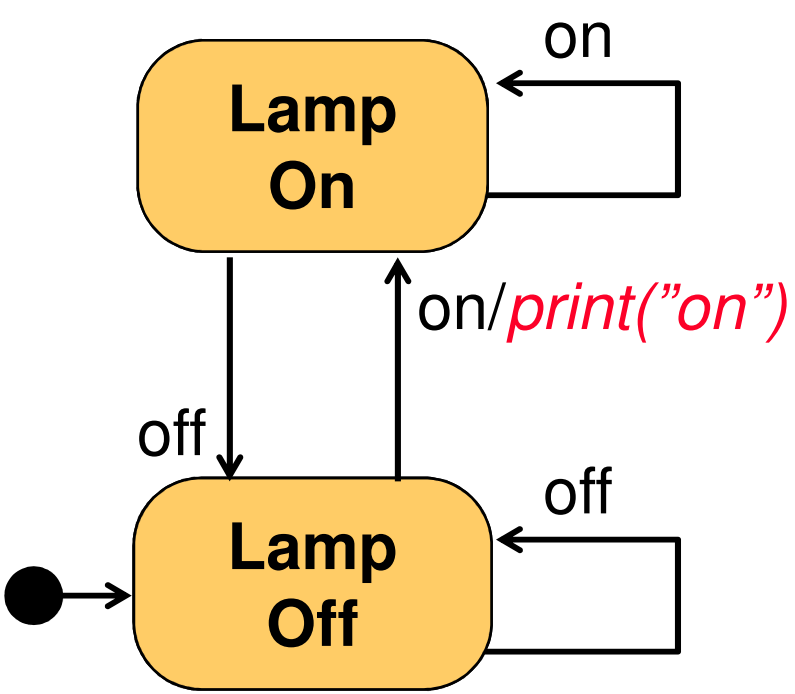
State Machine (Automaton) Diagram

- Graphical rendering of automata behavior

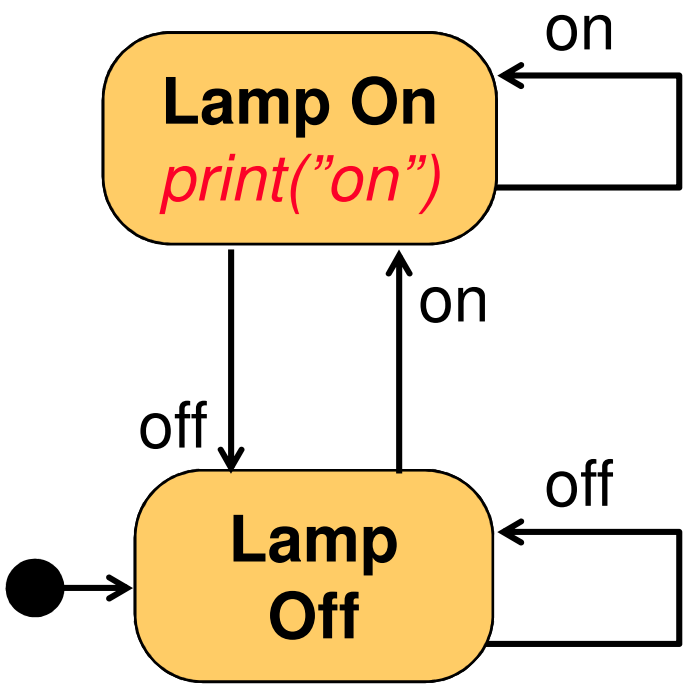


Outputs and Actions

- As the automaton changes state it can generate outputs:



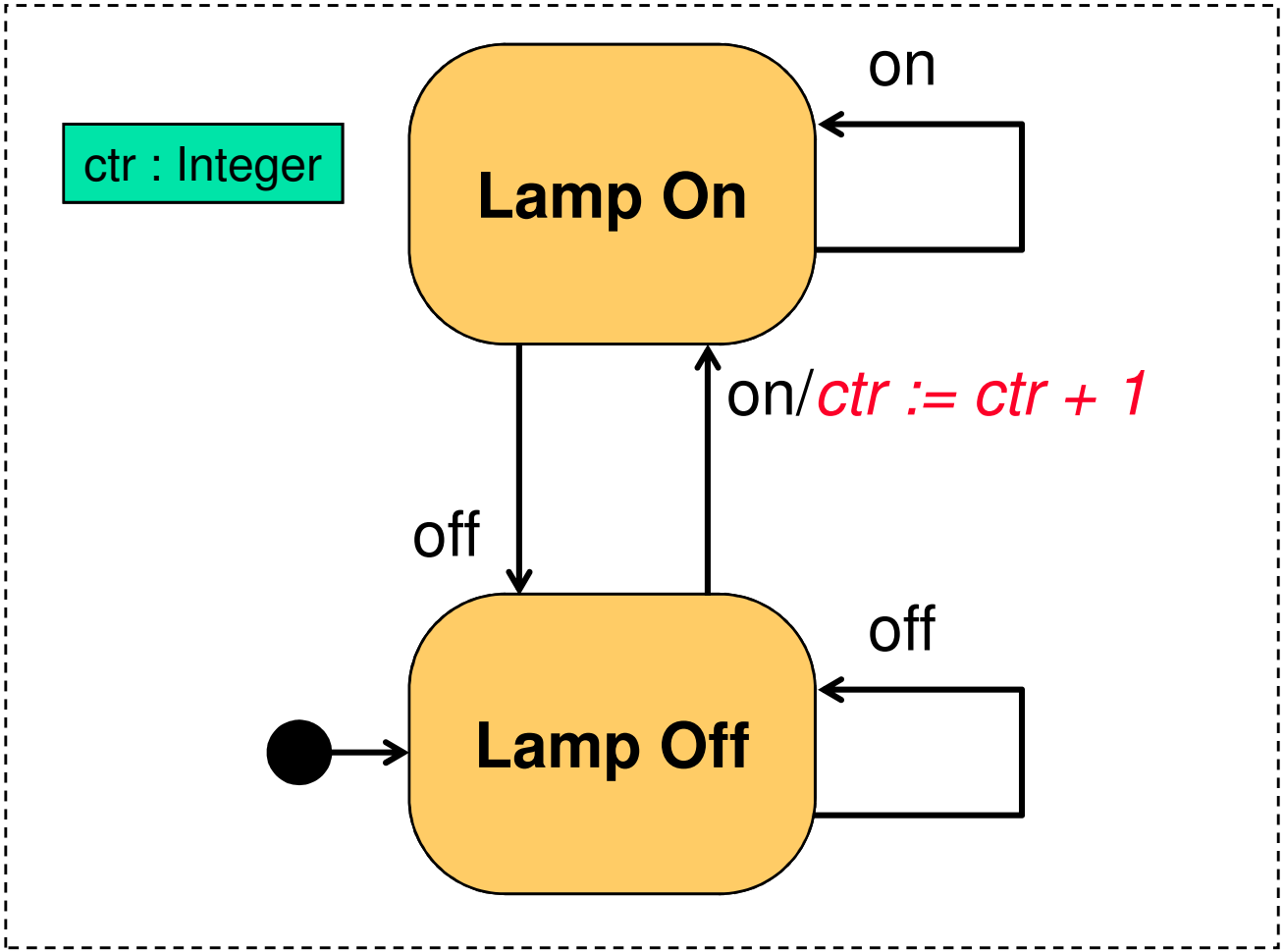
Mealy automaton

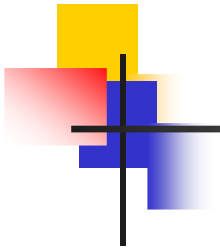


Moore automaton

Extended State Machines

- Addition of variables ("extended state")

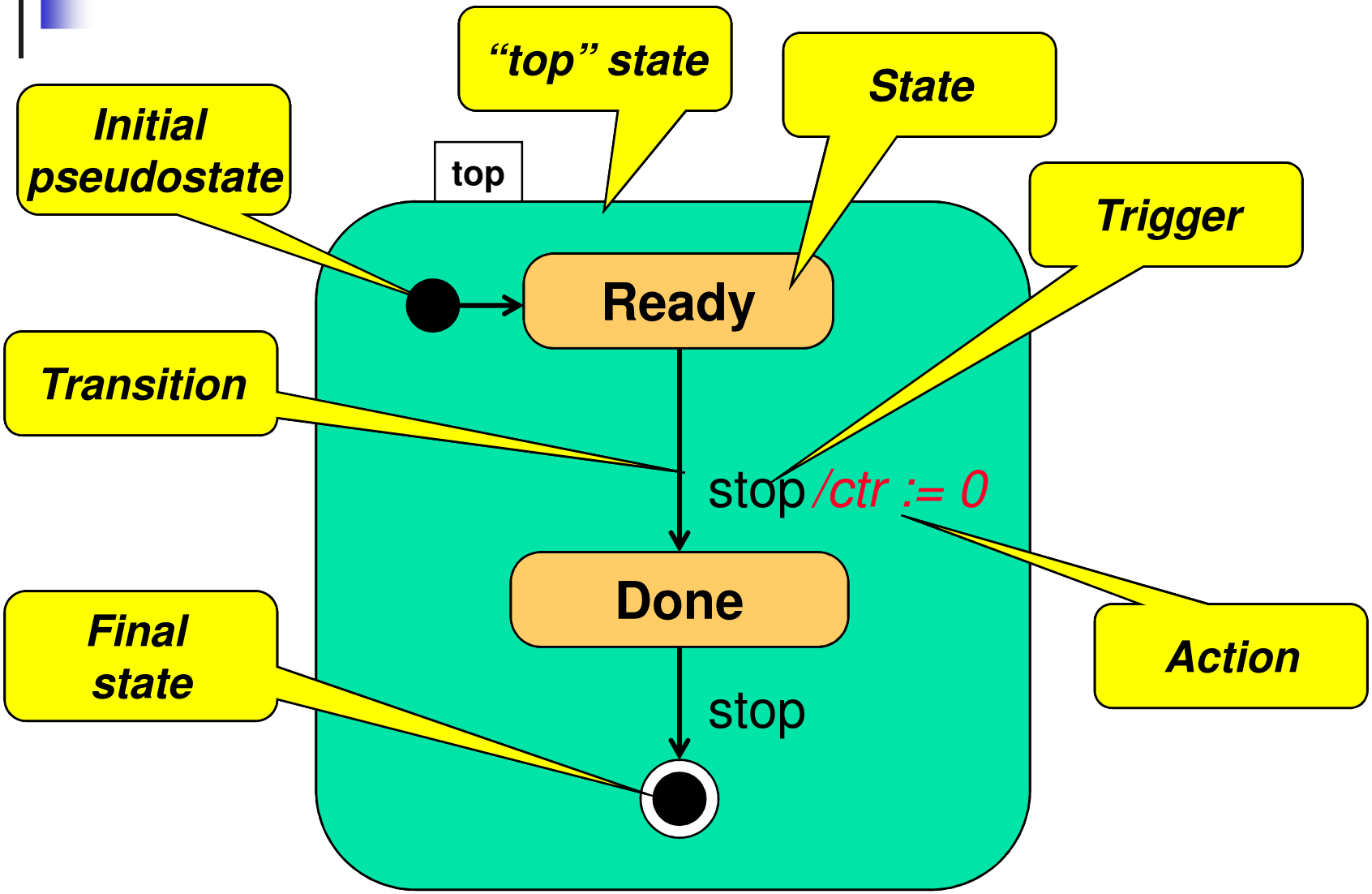




A Bit of Theory

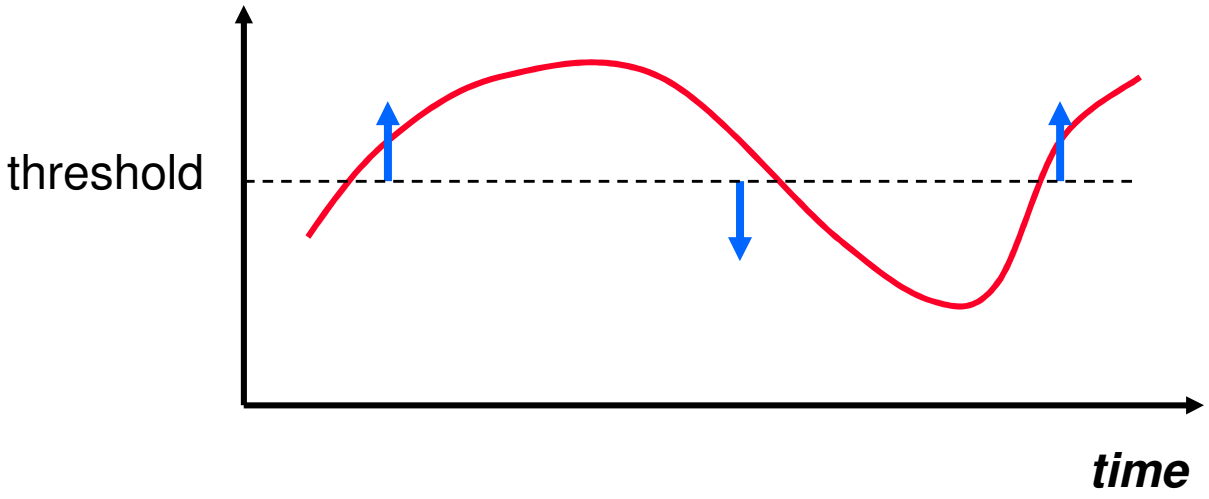
- An extended (Mealy) state machine is defined by:
  - a set of input signals (input alphabet)
  - a set of output signals (output alphabet)
  - a set of states
  - a set of transitions
    - triggering signal
    - action
  - a set of extended state variables
  - an initial state designation
  - a set of final states (if terminating automaton)

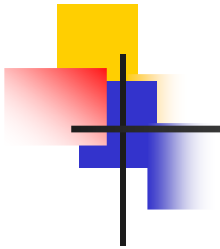
Basic UML Statechart Diagram



What Kind of Behavior?

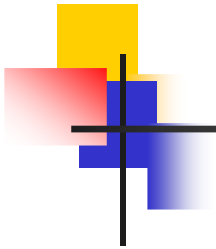
- In general, state machines are suitable for describing event-driven, discrete behavior
  - inappropriate for modeling continuous behavior





Event-Driven Behavior

- Event = a type of observable occurrence
  - interactions:
    - synchronous object operation invocation (call event)
    - asynchronous signal reception (signal event)
  - occurrence of time instants (time event)
    - interval expiry
    - calendar/clock time
  - change in value of some entity (change event)
- Event Instance = an instance of an event (type)
  - occurs at a particular time instant and has no duration



The Behavior of What?

- In principle, anything that manifests event-driven behavior
  - NB: there is no support currently in UML for modeling continuous behavior
- In practice:
  - the behavior of individual objects
  - object interactions
- The dynamic semantics of UML state machines are currently mainly specified for the case of active objects

Basic State Machine Concepts

Statecharts and Objects

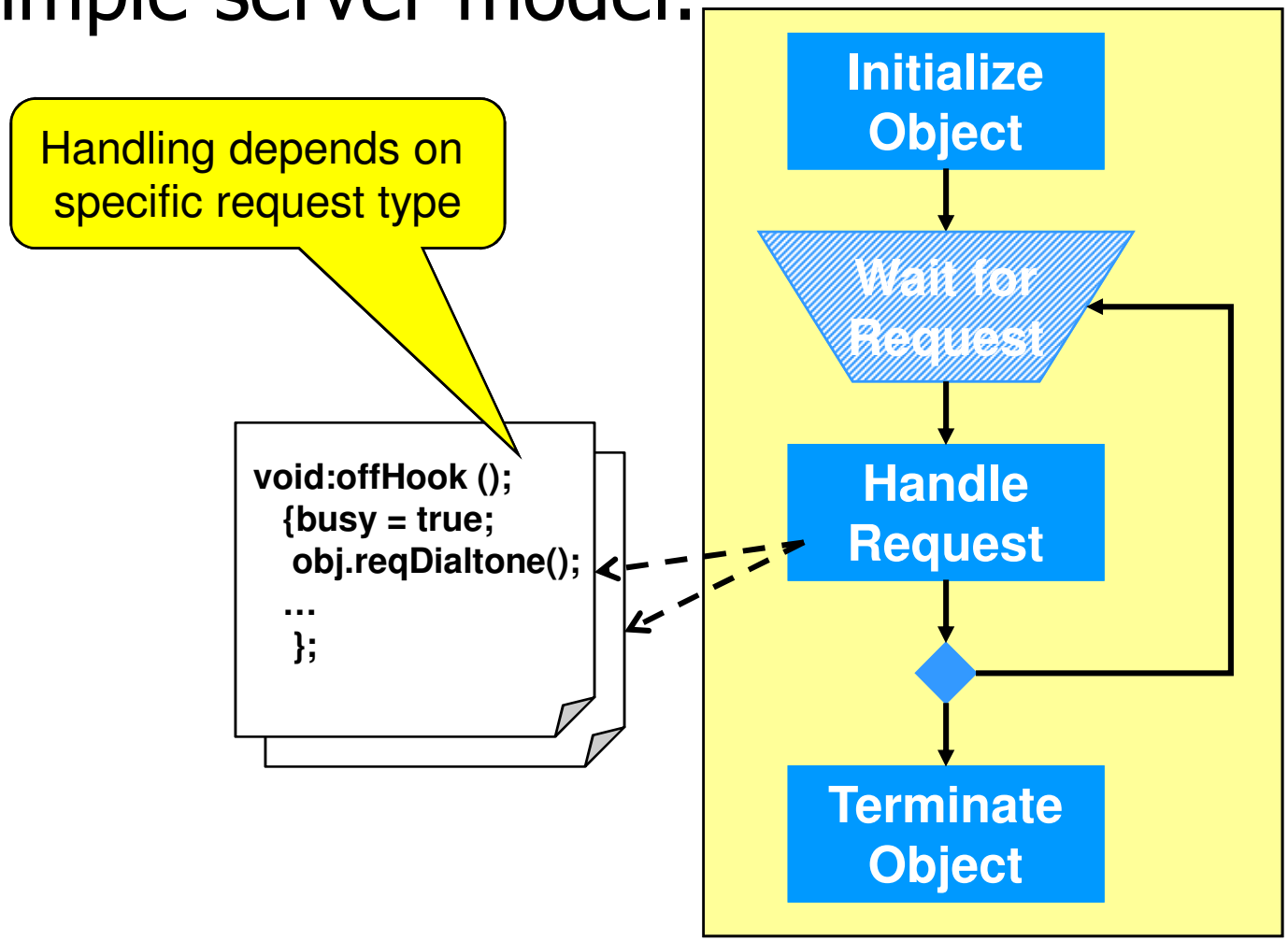
Advanced Modeling Concepts

Case Study

Wrap Up

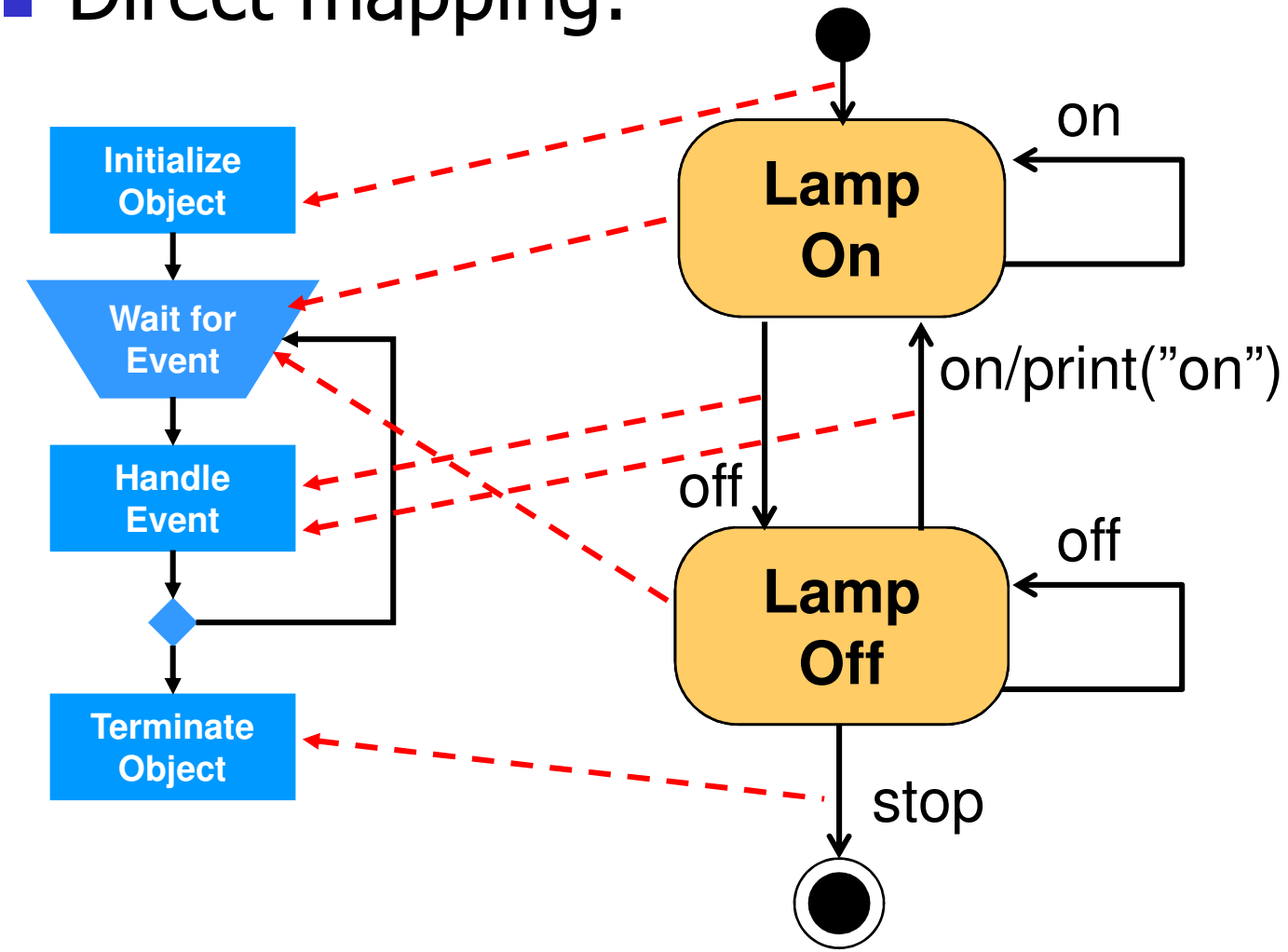
Object Behavior - General Model

- Simple server model:



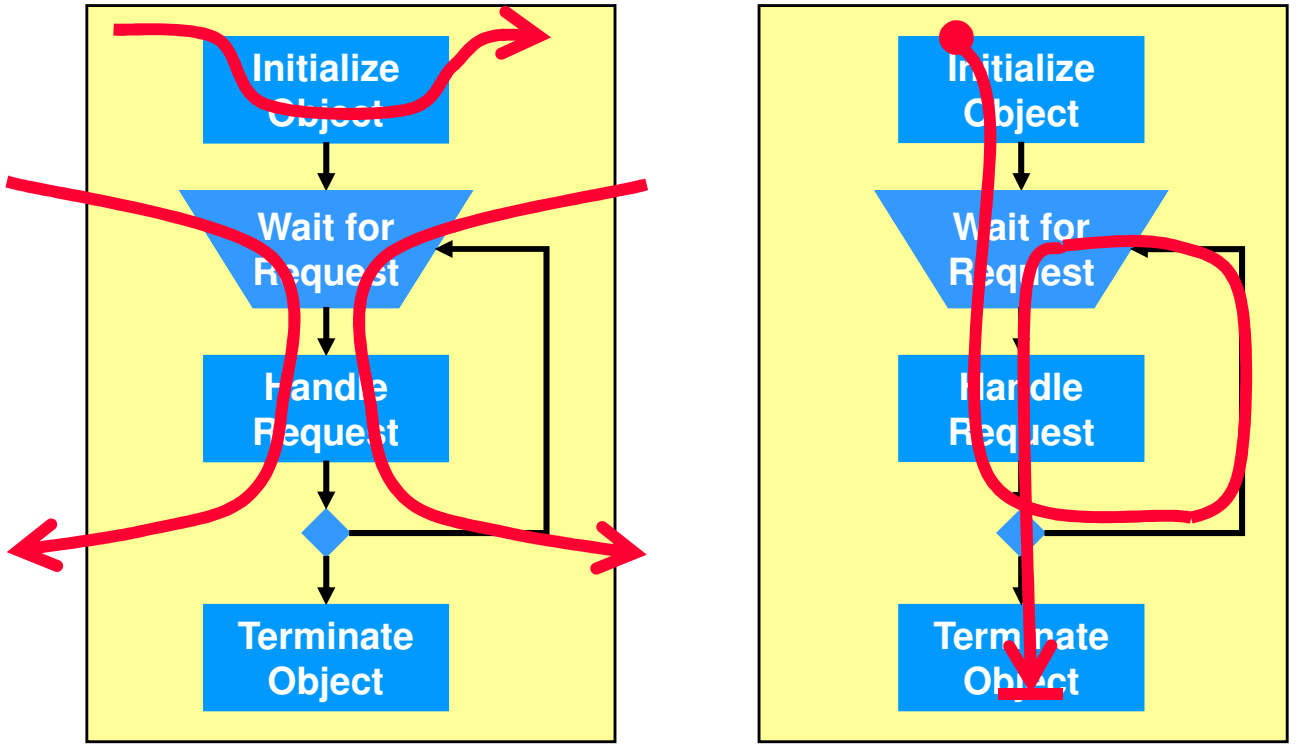
Object Behavior and State Machines

- Direct mapping:

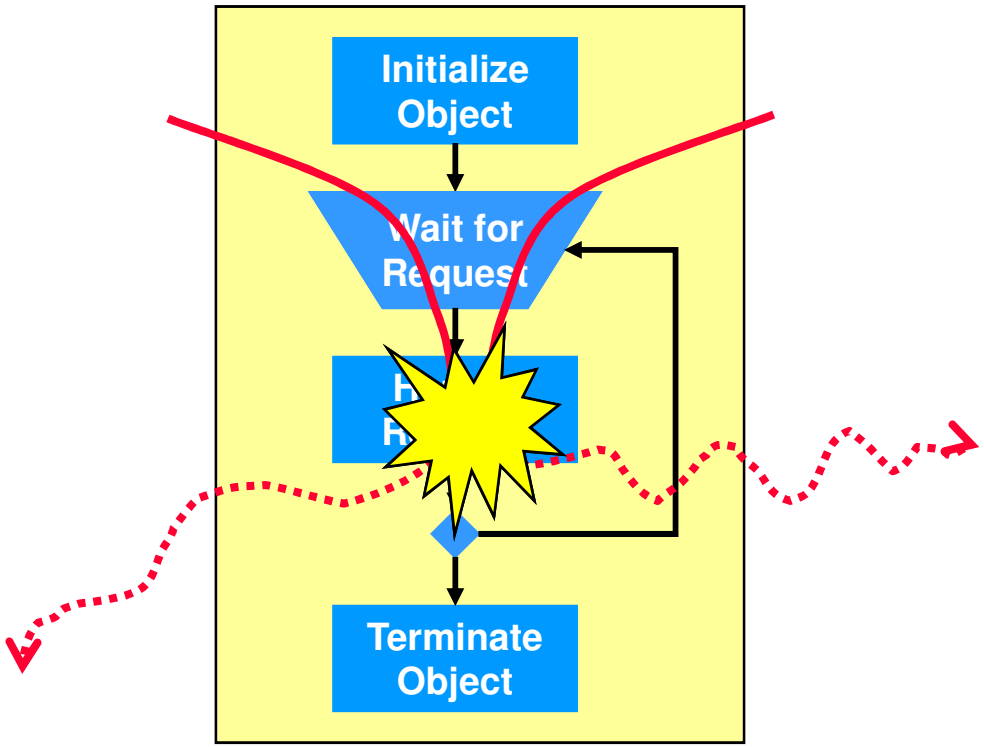


Object and Threads

- **Passive objects:** depend on external power (thread of execution)
- **Active objects:** self-powered (own thread of execution)



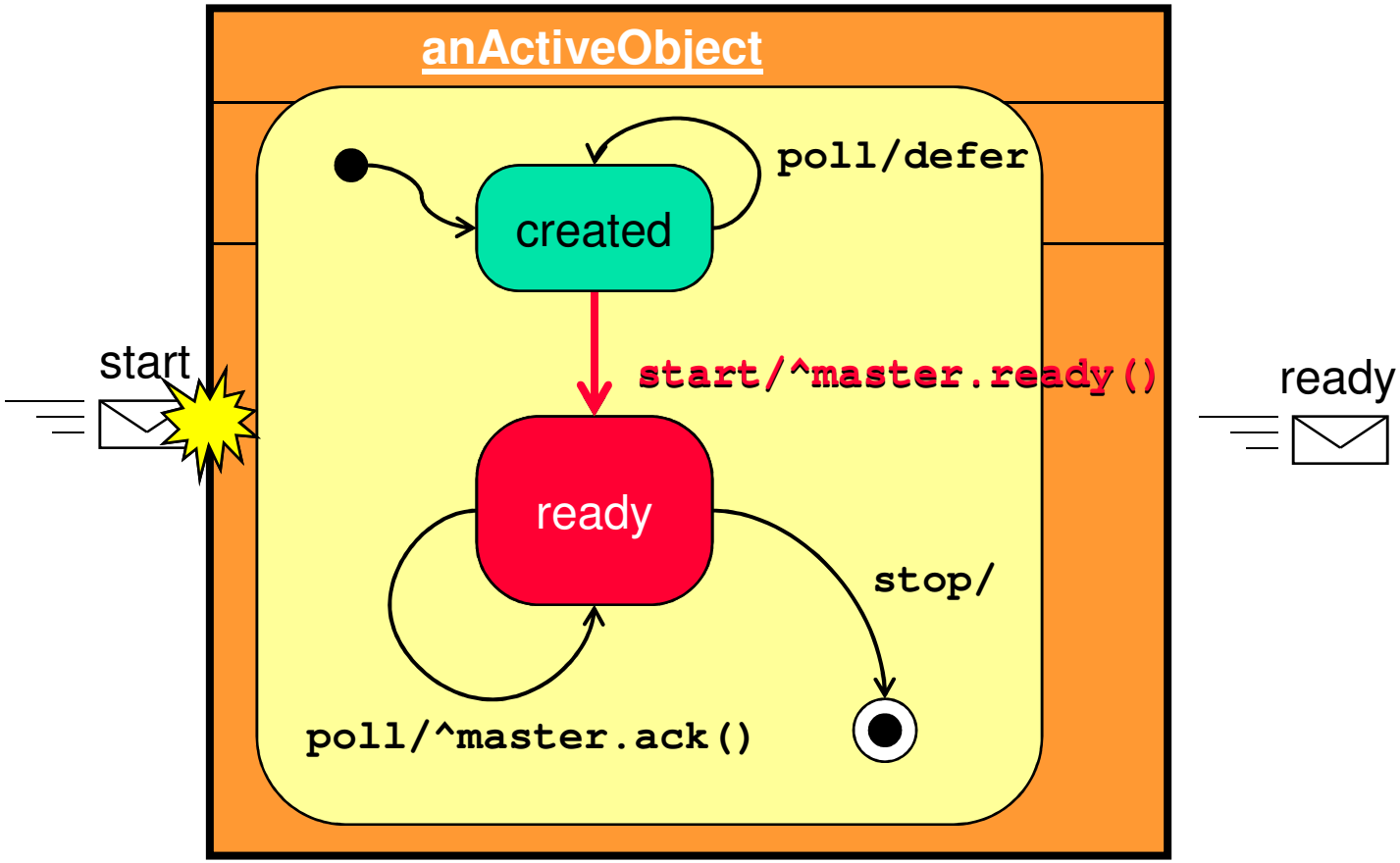
Passive Objects: Dynamic Semantics



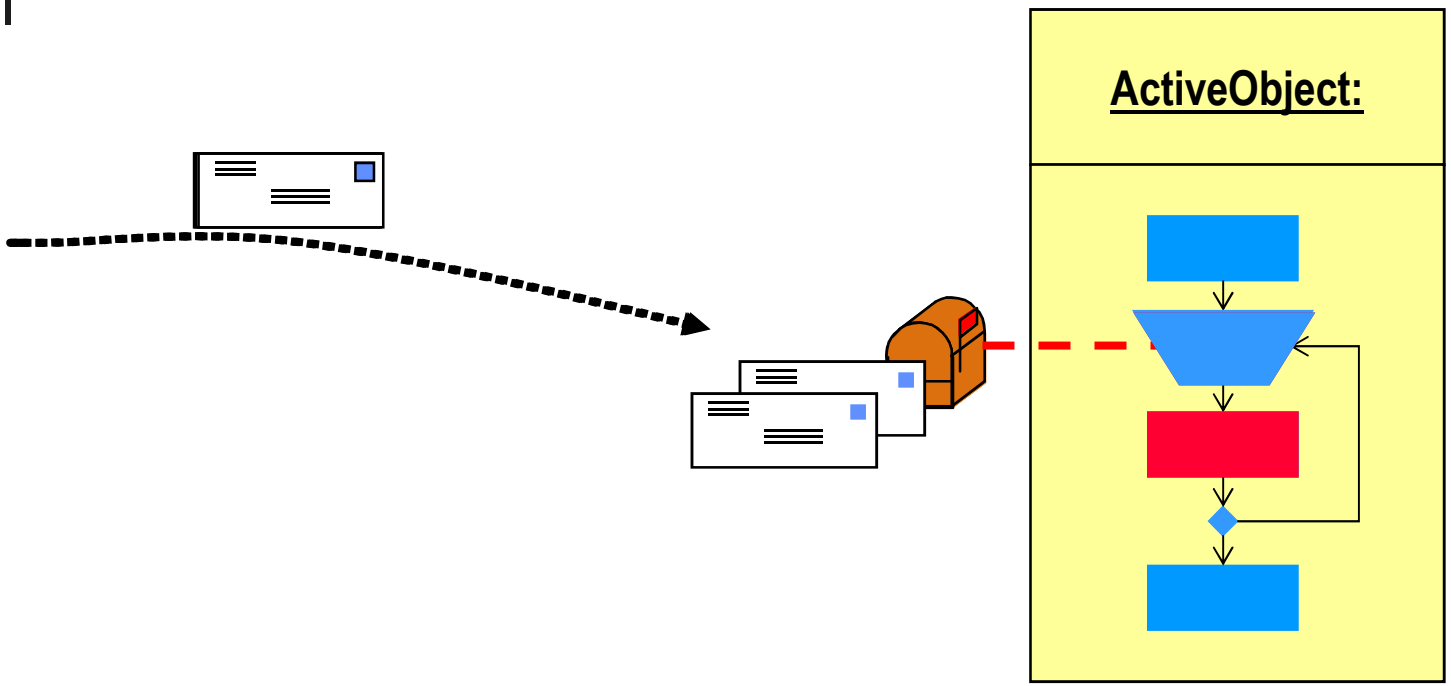
- Encapsulation does not protect the object from concurrency conflicts!
- Explicit synchronization is still required

Active Objects and State Machines

- Objects that encapsulate own thread of execution



Active Objects: Dynamic Semantics

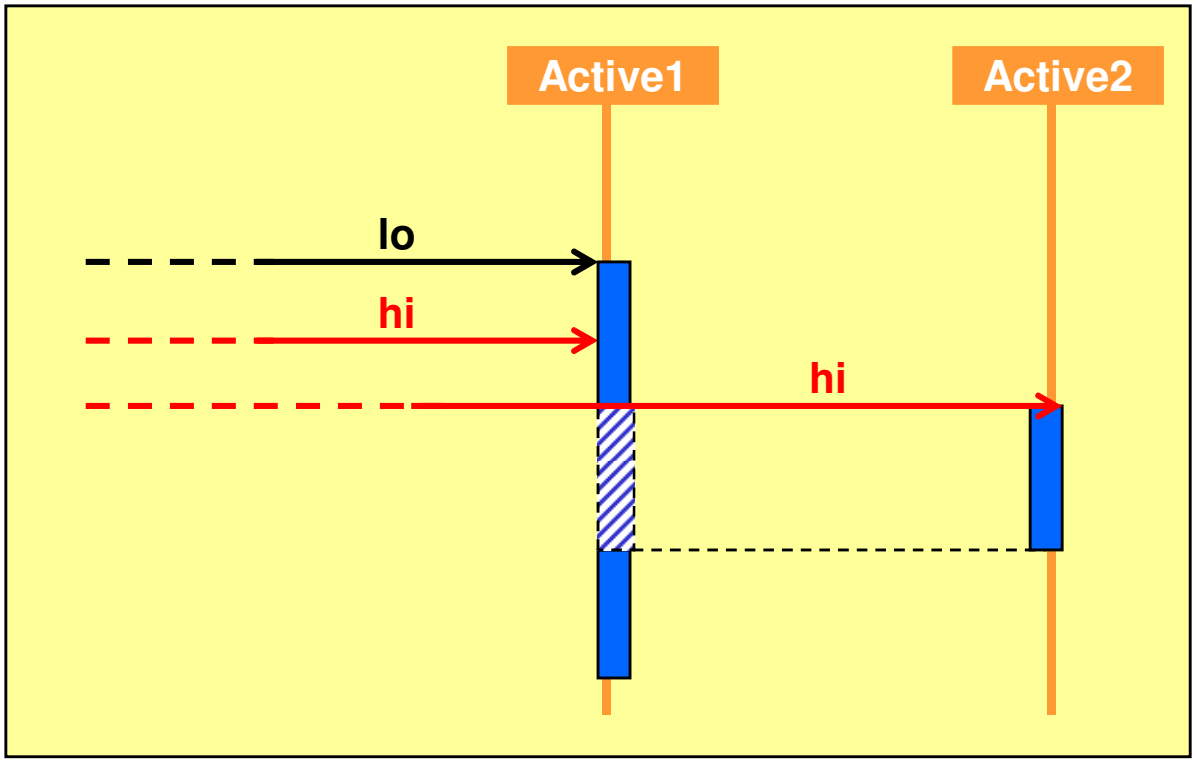


Run-to-completion model:

- serialized event handling
- eliminates internal concurrency
- minimal context switching overhead

The Run-to-Completion Model

- A high priority event for (another) active object will preempt an active object that is handling a low-priority event



Basic State Machine Concepts

Statecharts and Objects

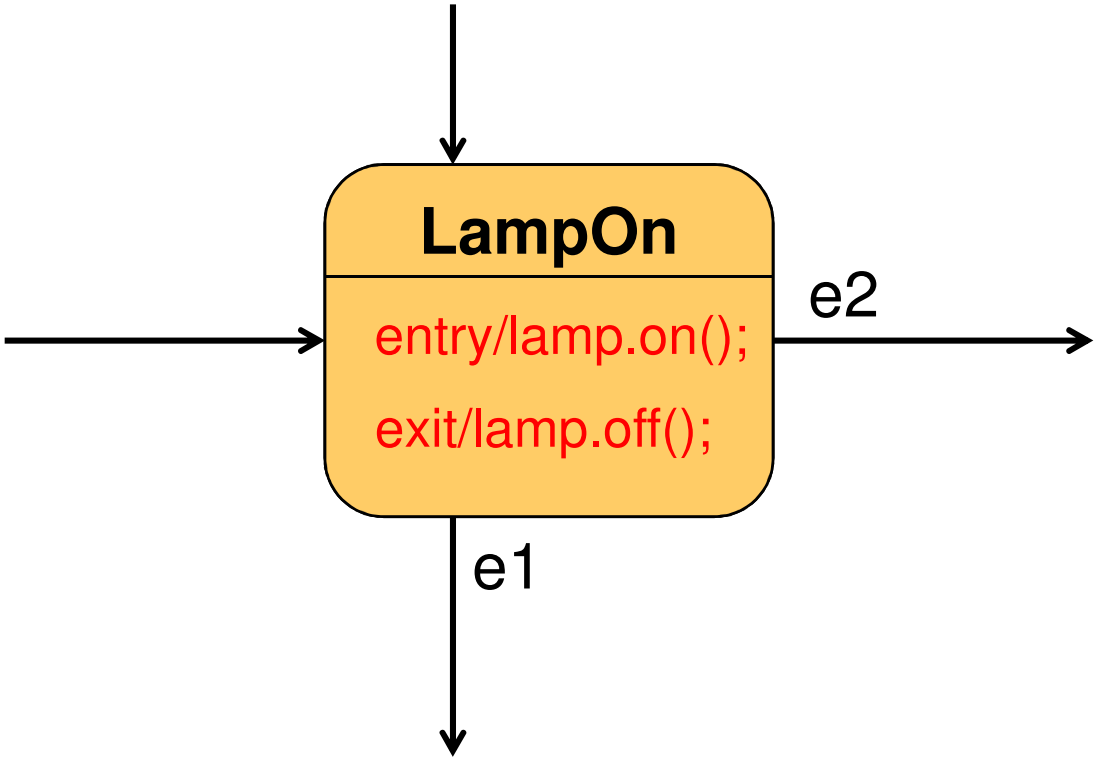
Advanced Modeling Concepts

Case Study

Wrap Up

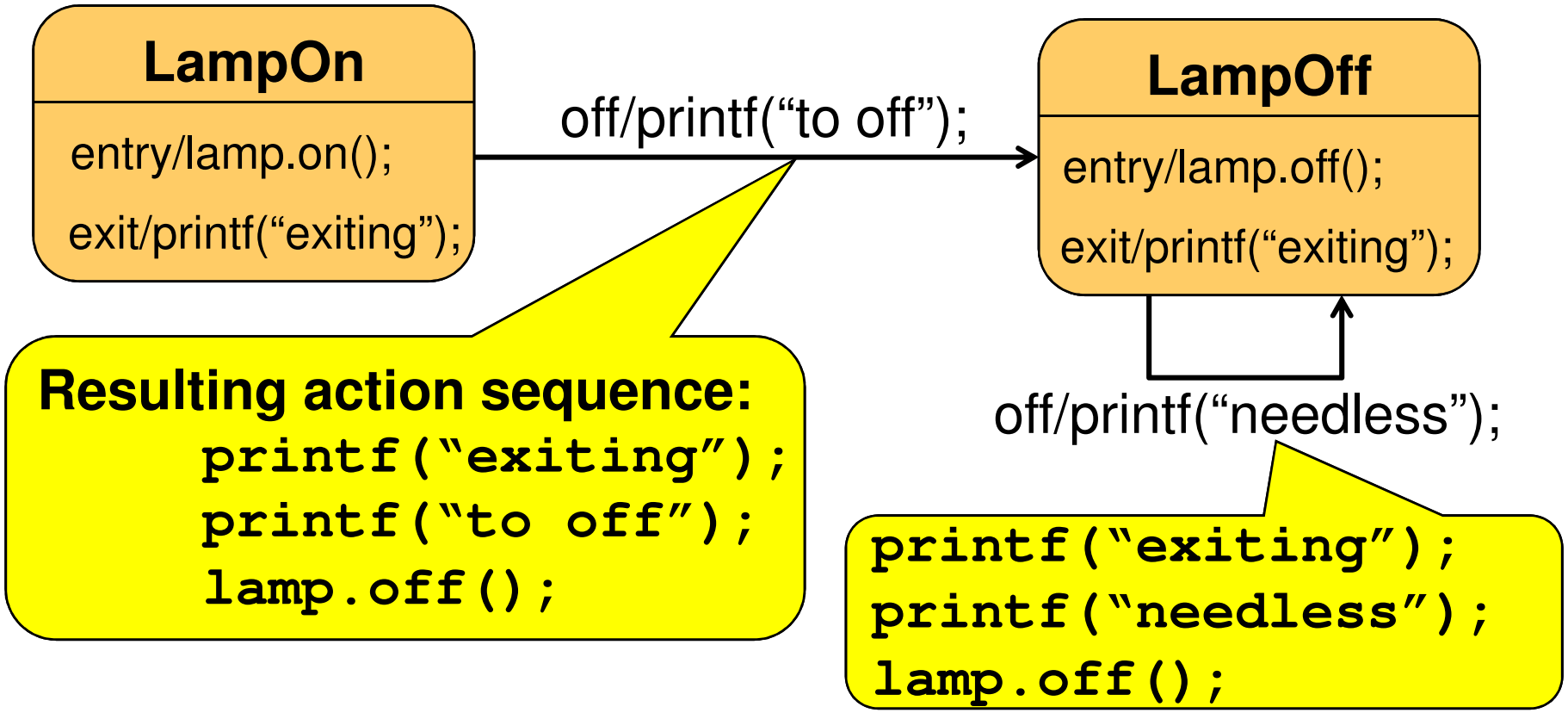
State Entry and Exit Actions

- A dynamic assertion mechanism



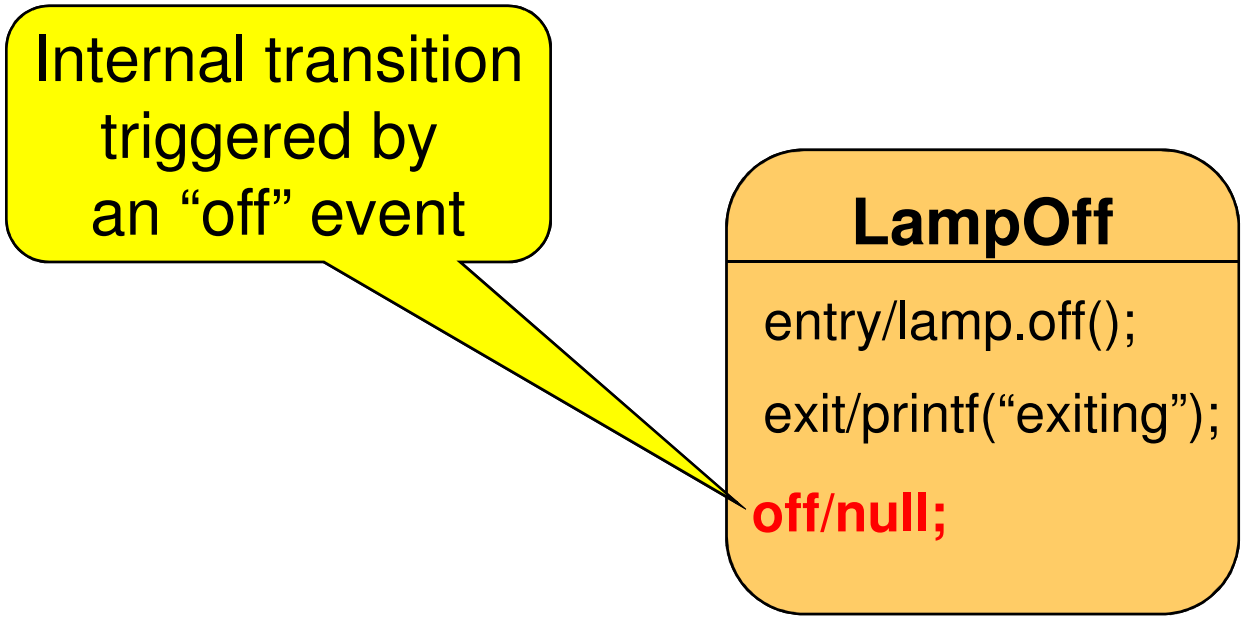
Order of Actions: Simple Case

- Exit actions prefix transition actions
- Entry action postfix transition actions



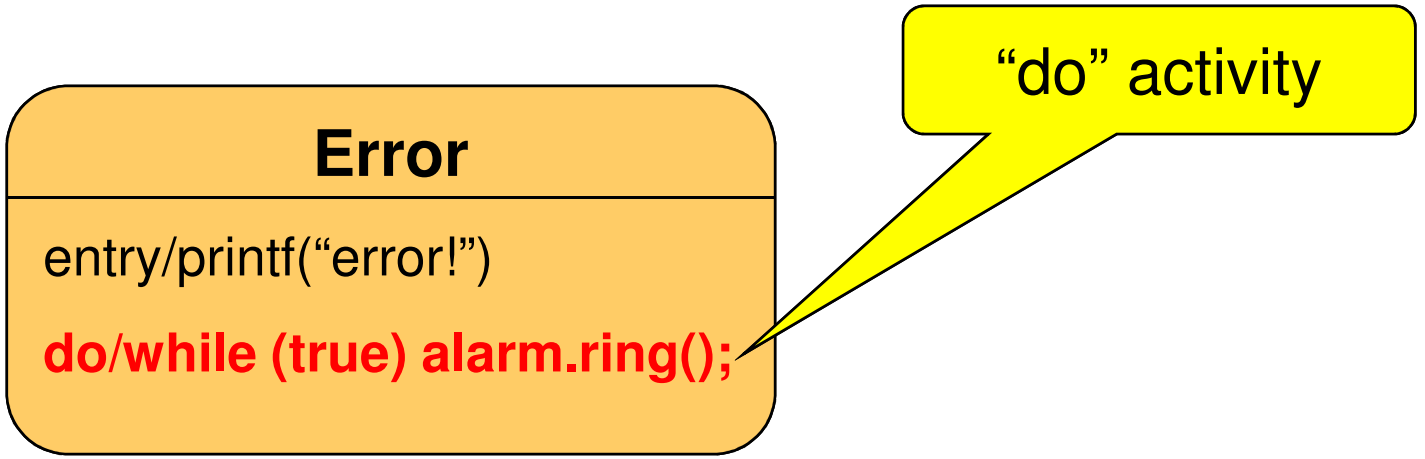
Internal Transitions

- Self-transitions that bypass entry and exit actions



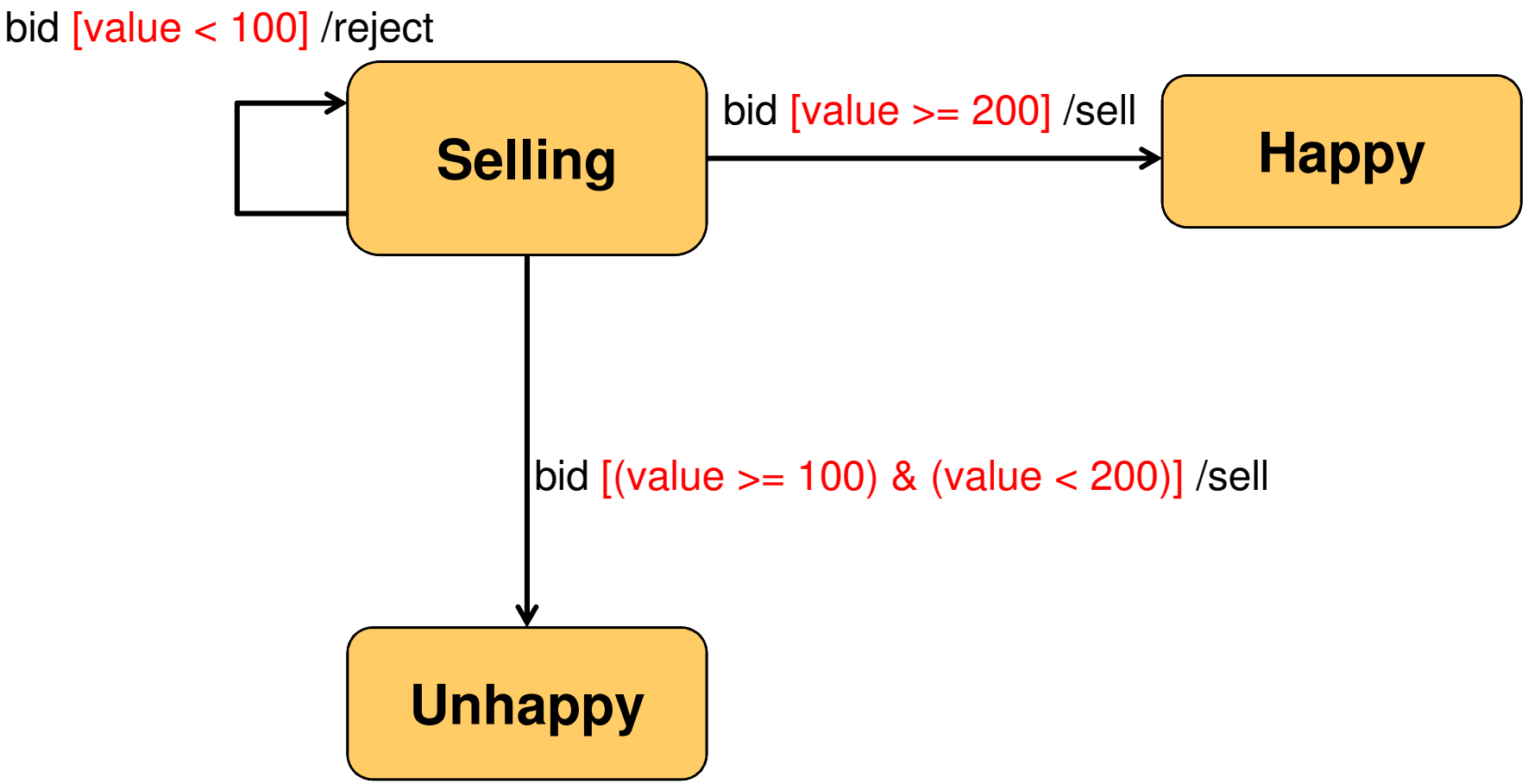
State ("Do") Activities

- Forks a concurrent thread that executes until:
  - the action completes or
  - the state is exited through an outgoing transition



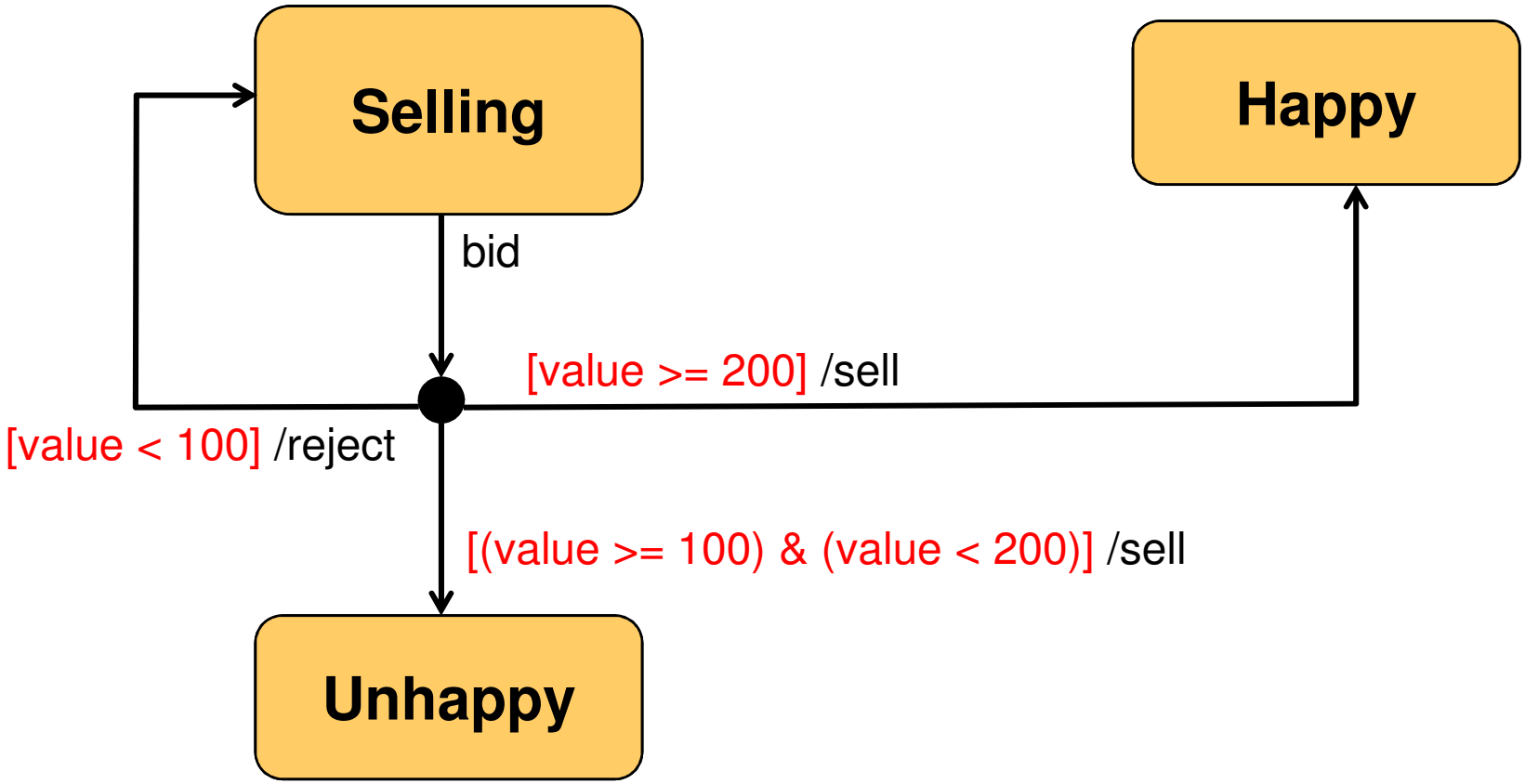
Guards

- Conditional execution of transitions
  - guards (Boolean predicates) must be side-effect free



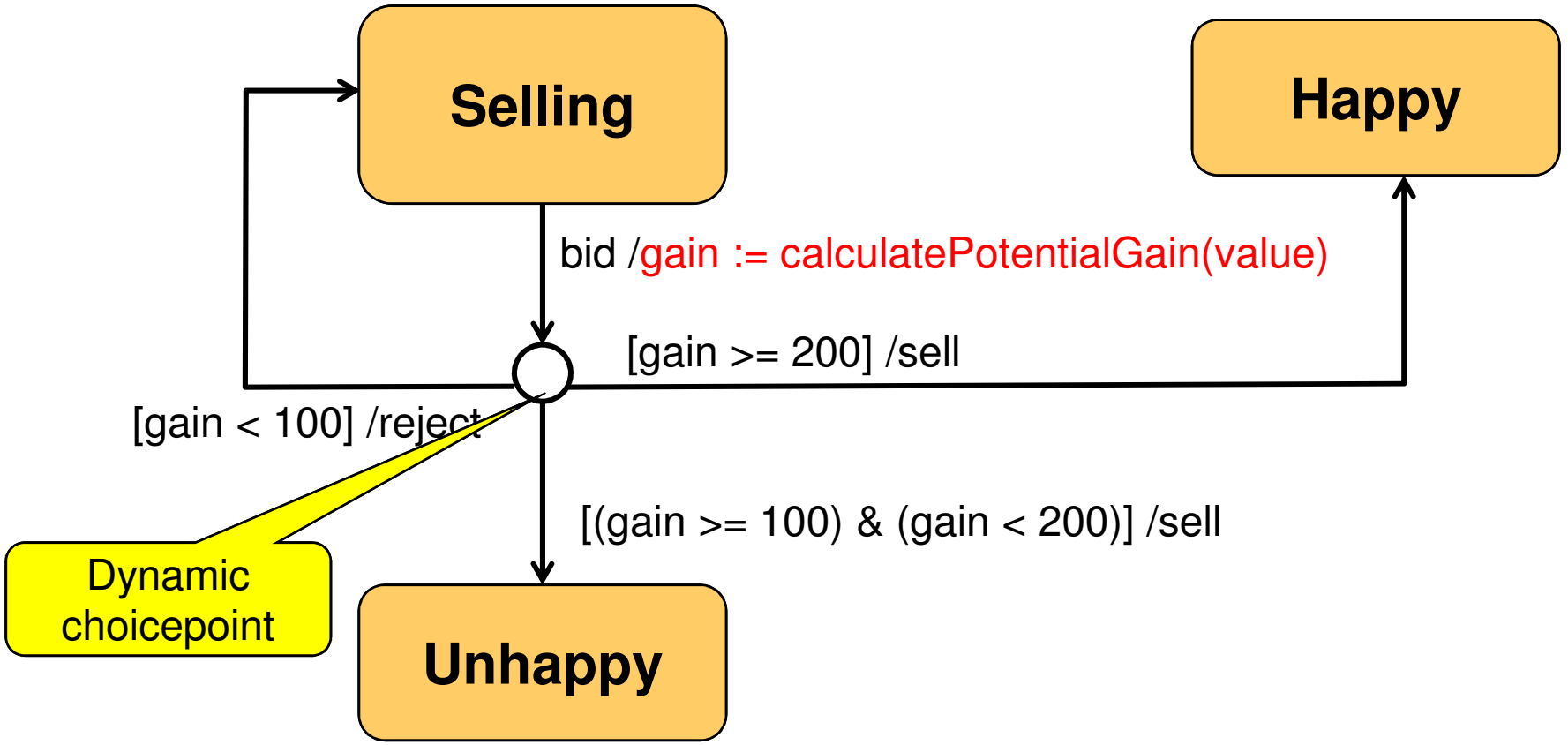
Static Conditional Branching

- Merely a graphical shortcut for convenient rendering of decision trees



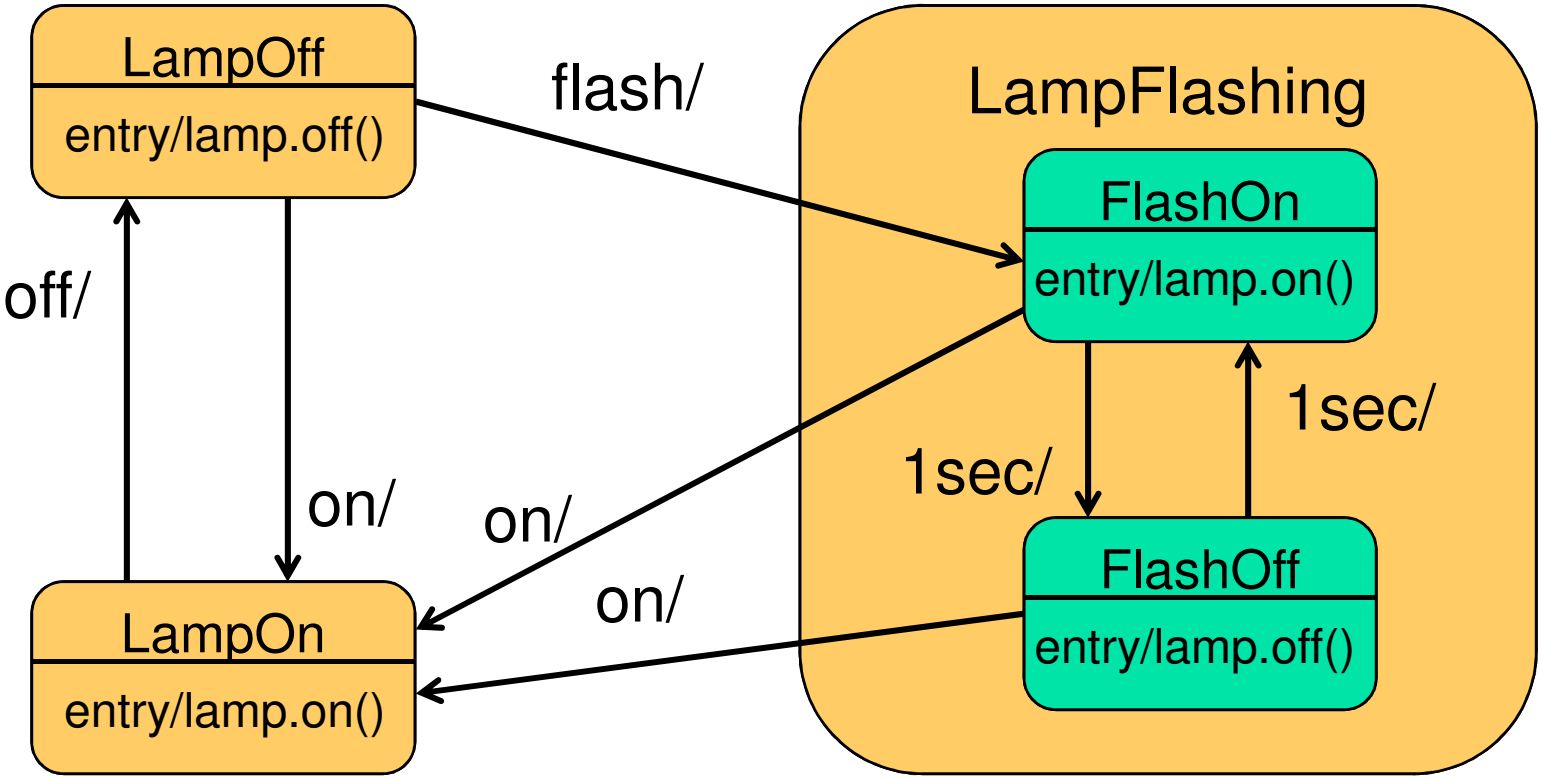
Dynamic Conditional Branching

- Choice pseudostate: guards are evaluated at the instant when the decision point is reached



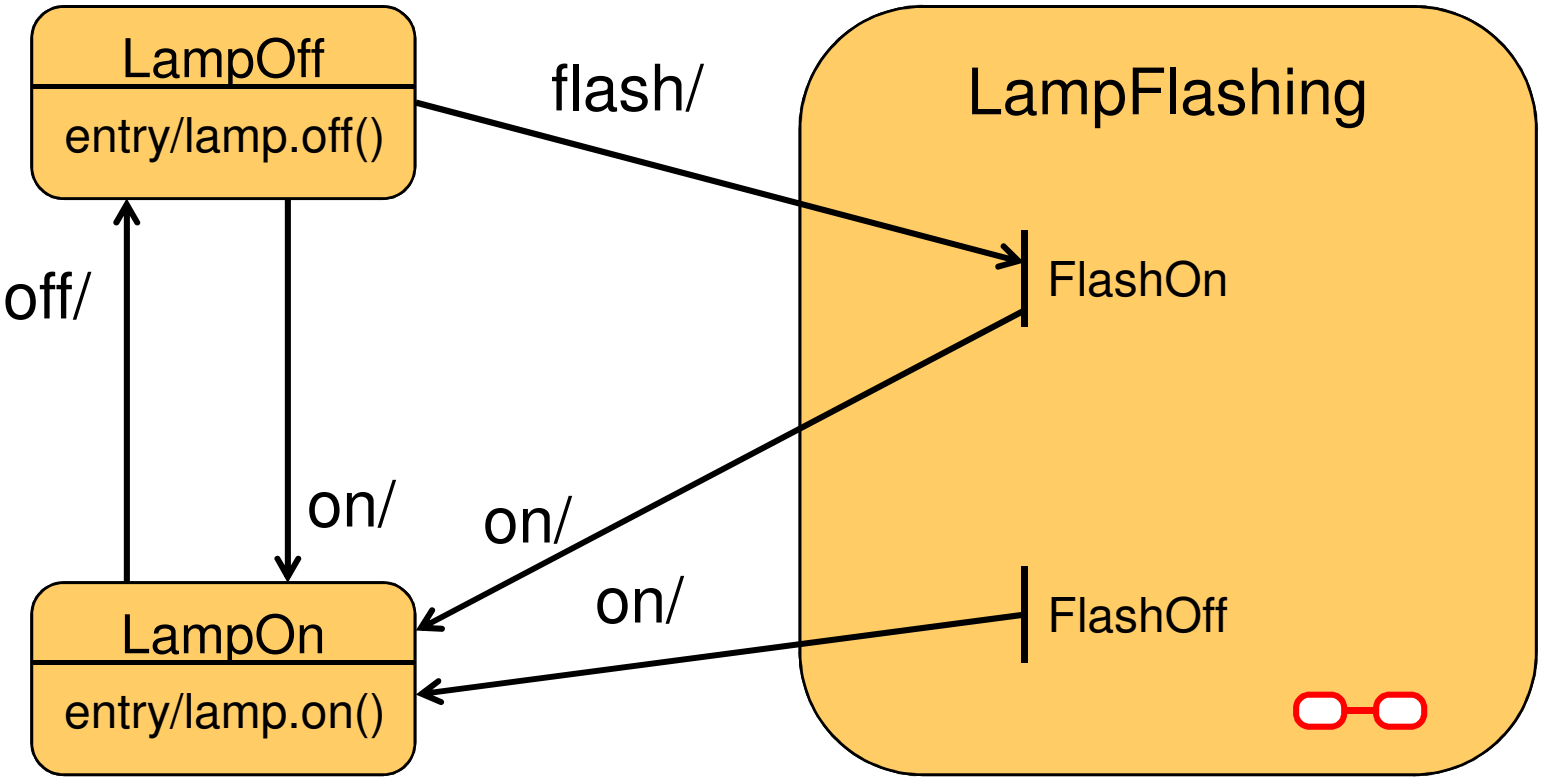
Hierarchical State Machines

- Graduated attack on complexity
  - states decomposed into state machines



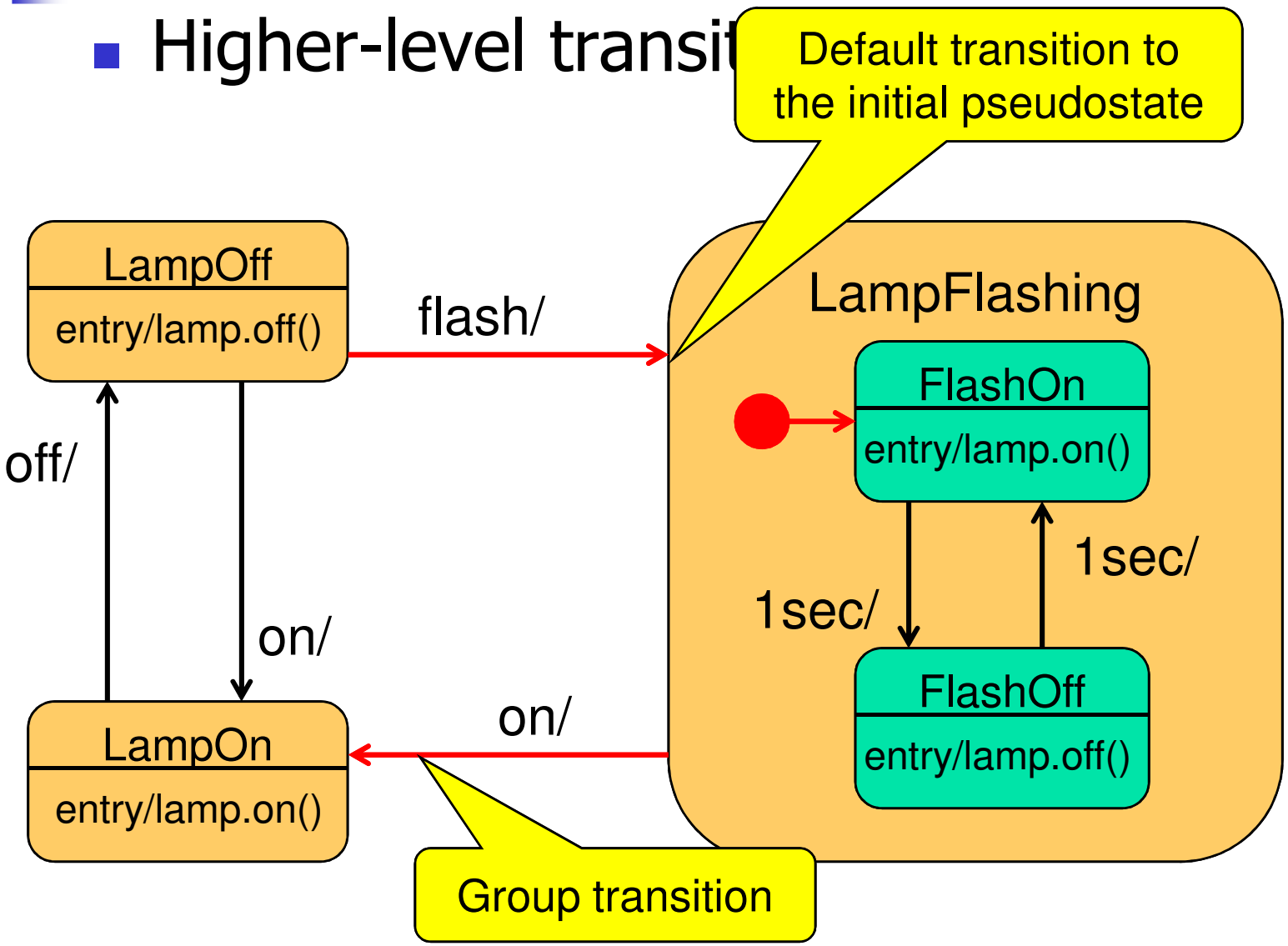
"Stub" Notation

- Notational shortcut: no semantic significance



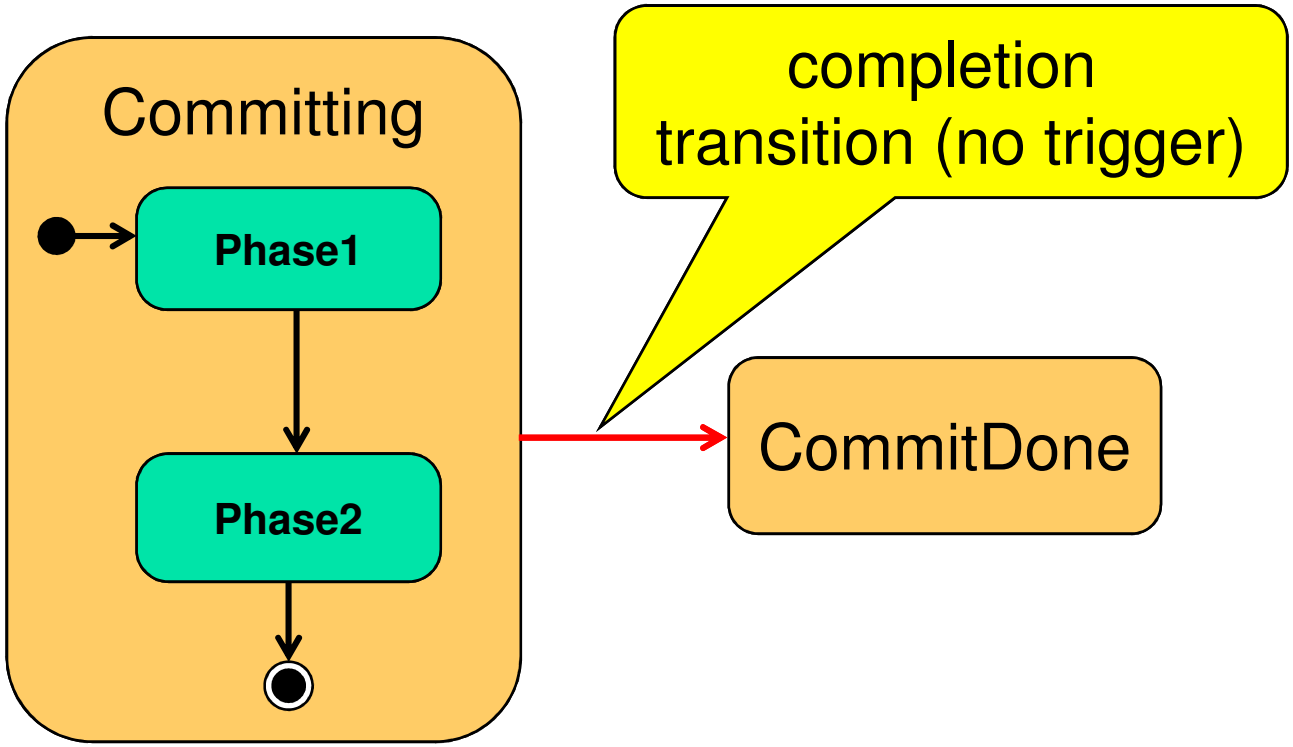
Group Transitions

- Higher-level transit



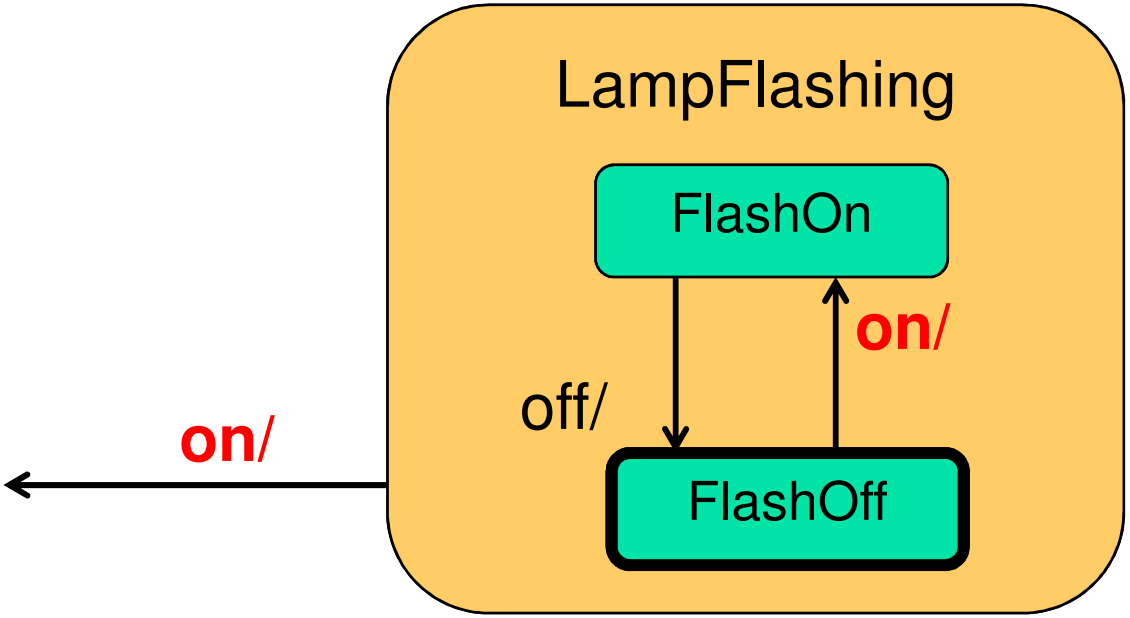
Completion Transitions

- Triggered by a completion event
  - generated automatically when an immediately nested state machine terminates



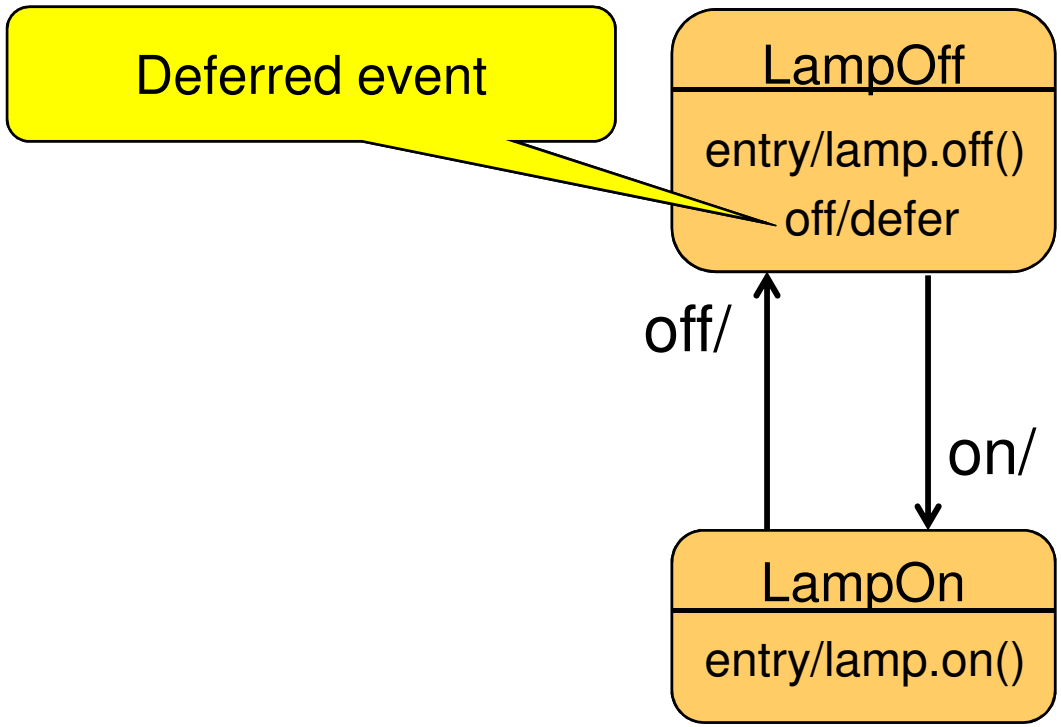
Triggering Rules

- Two or more transitions may have the same event trigger
  - innermost transition takes precedence
  - event is discarded whether or not it triggers a transition



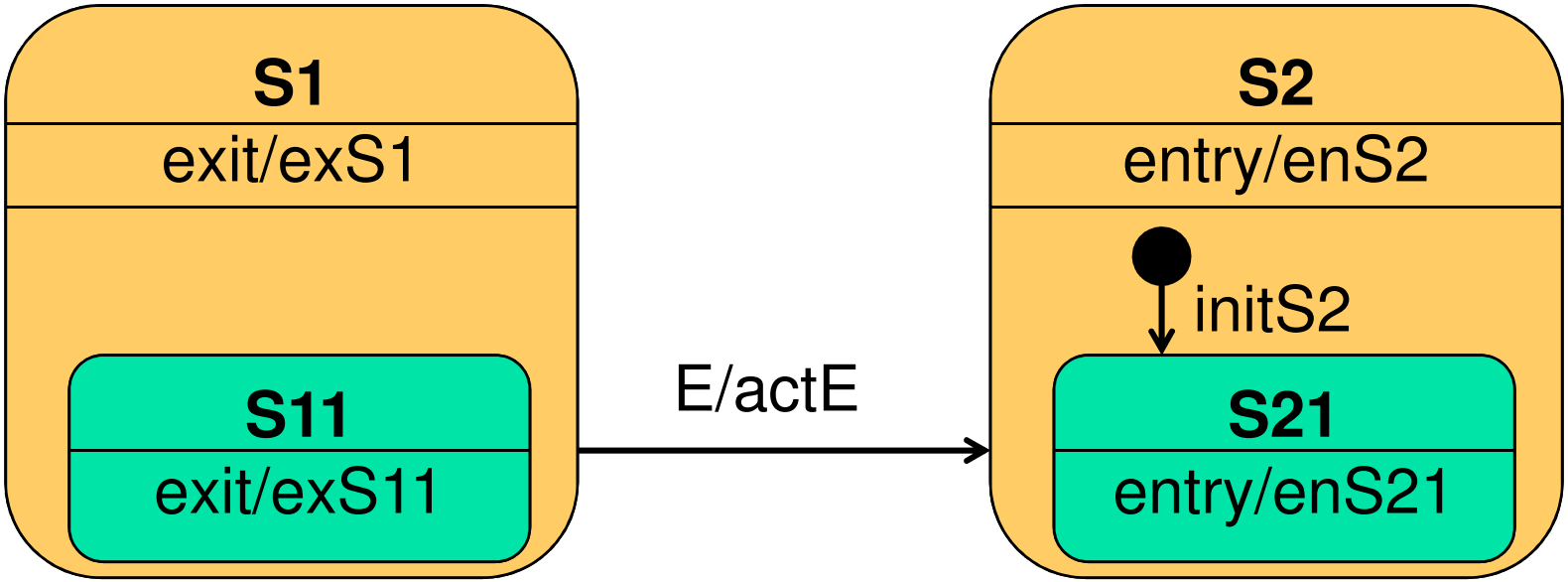
Deferred Events

- Events can be retained if they do not trigger a transition



Order of Actions: Complex Case

- Same approach as for the simple case

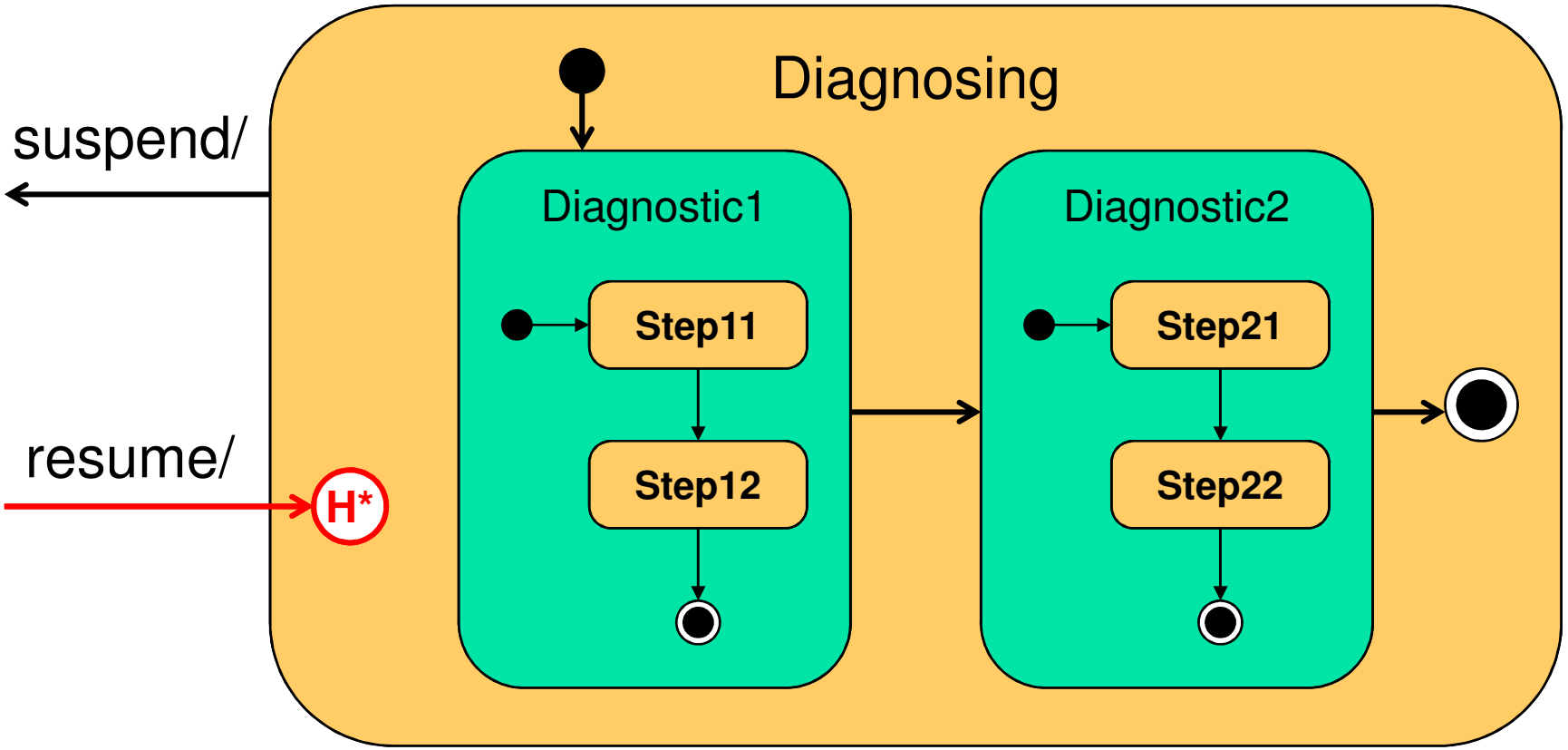


Actions execution sequence:

exS11 ⇨ exS1 ⇨ actE ⇨ enS2 ⇨ initS2 ⇨ enS21

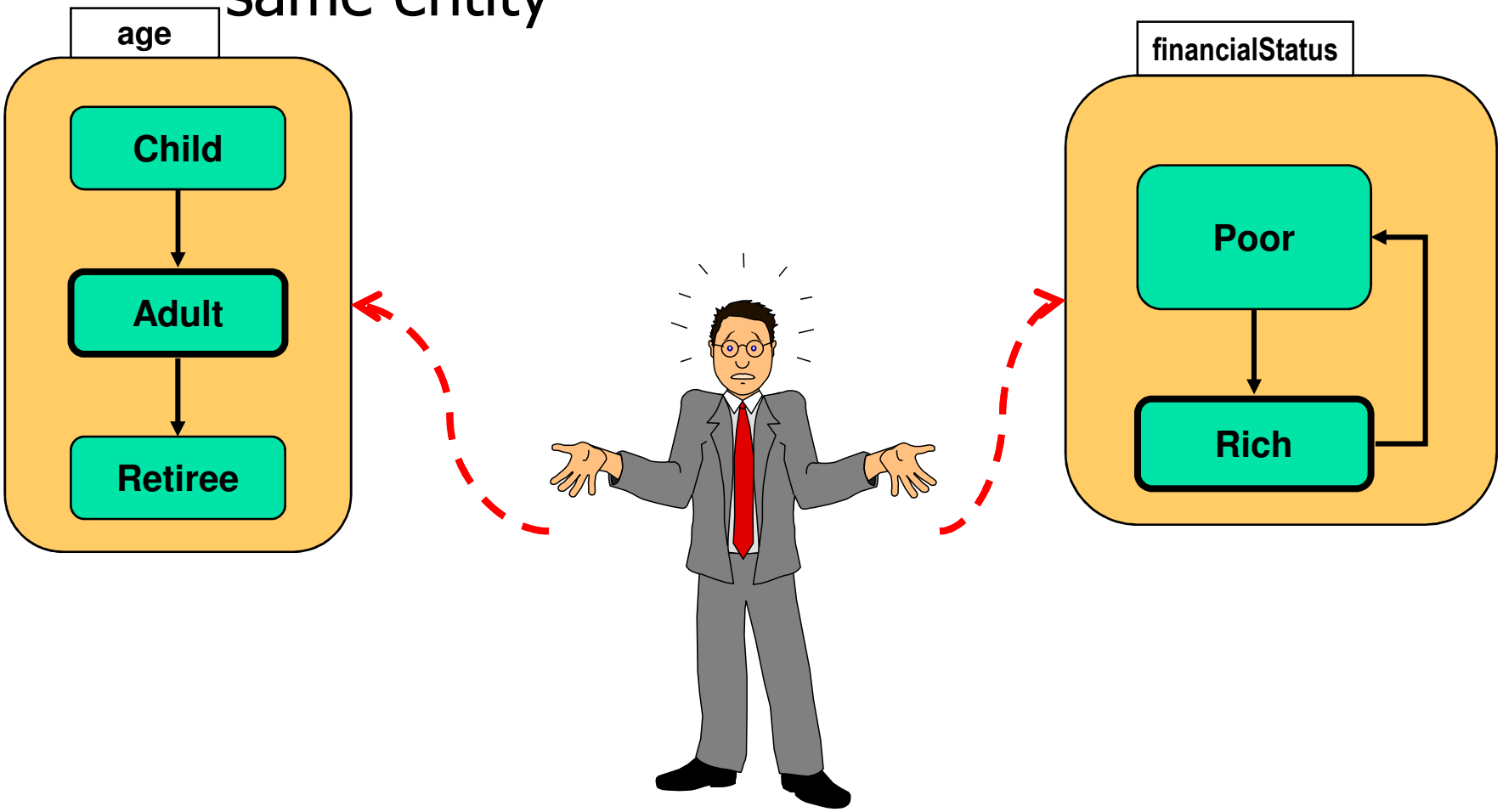
History

- Return to a previously visited hierarchical state
  - deep and shallow history options



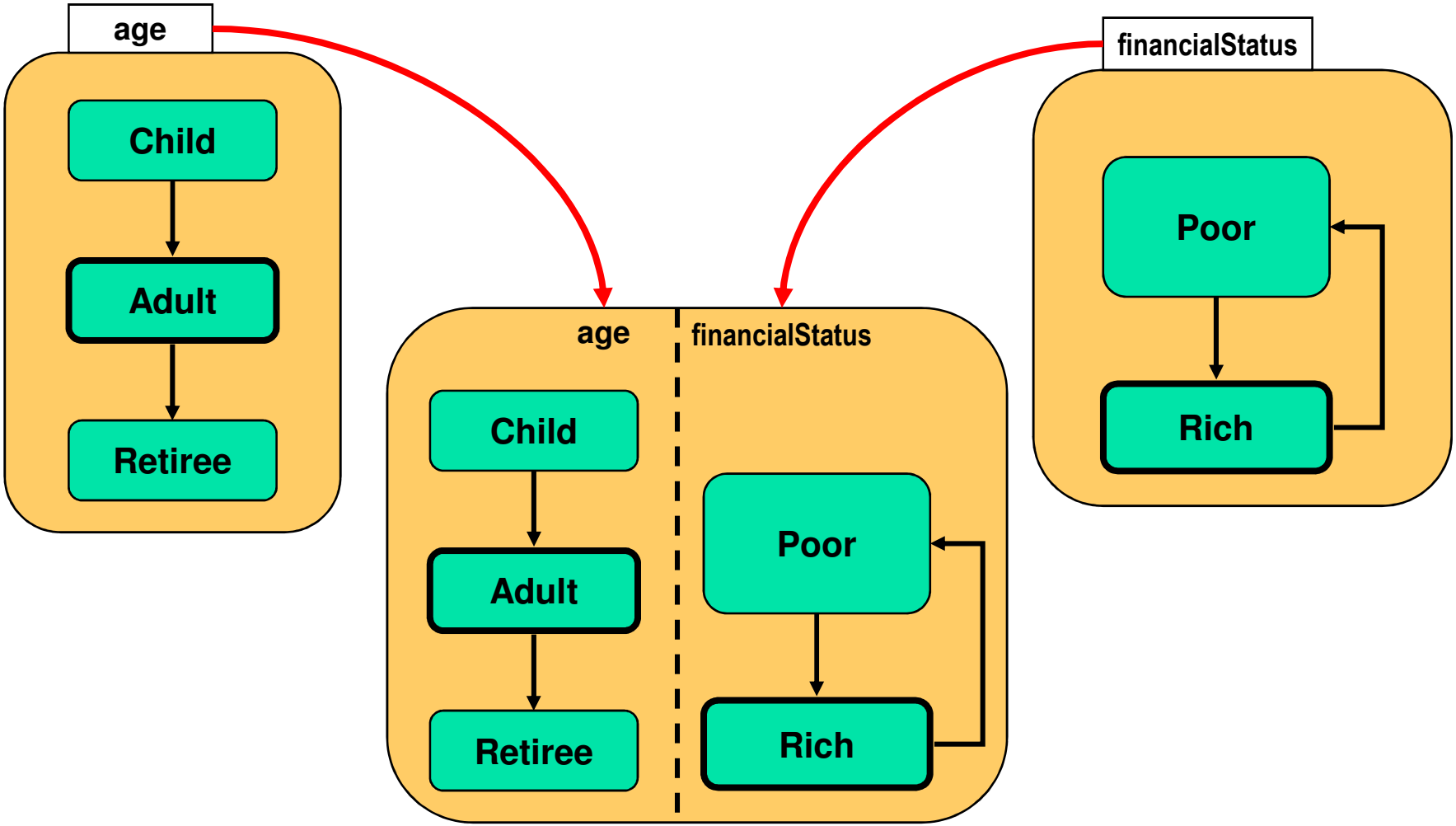
Orthogonality

- Multiple simultaneous perspectives on the same entity



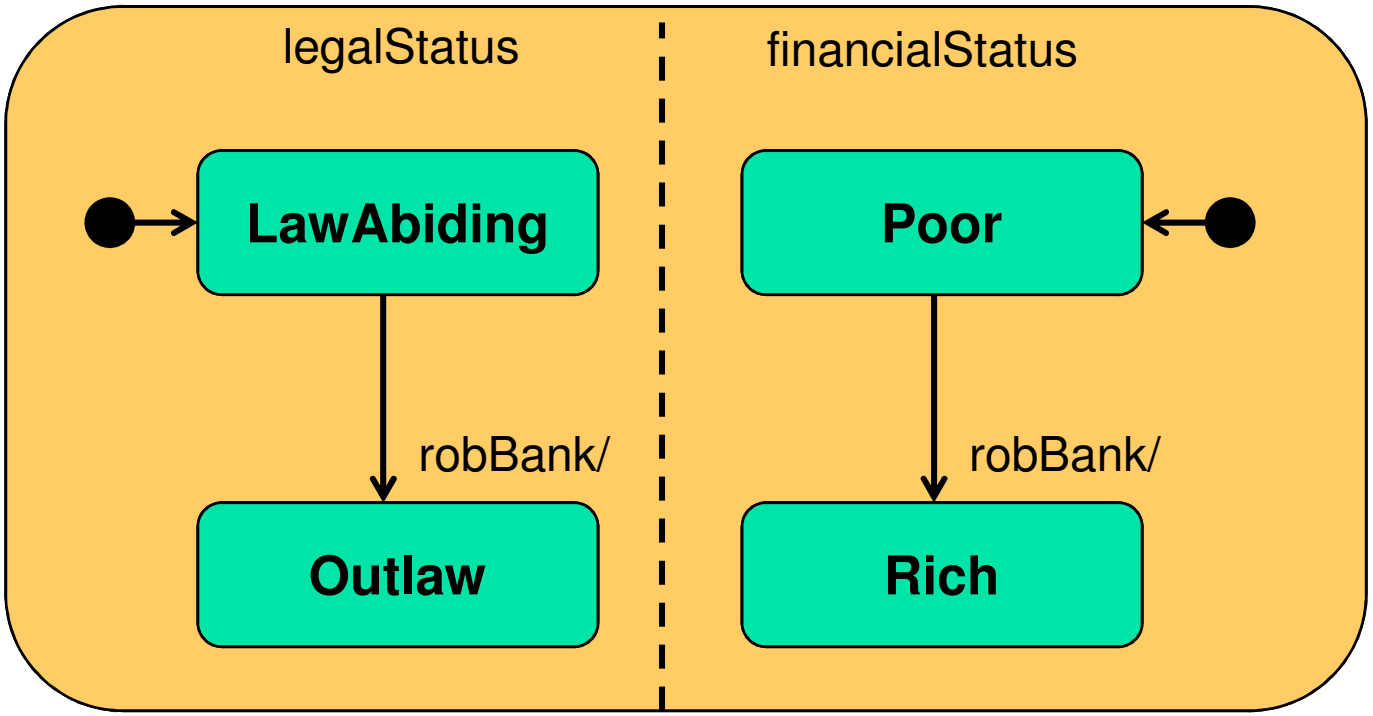
Orthogonal Regions

- Combine multiple simultaneous descriptions



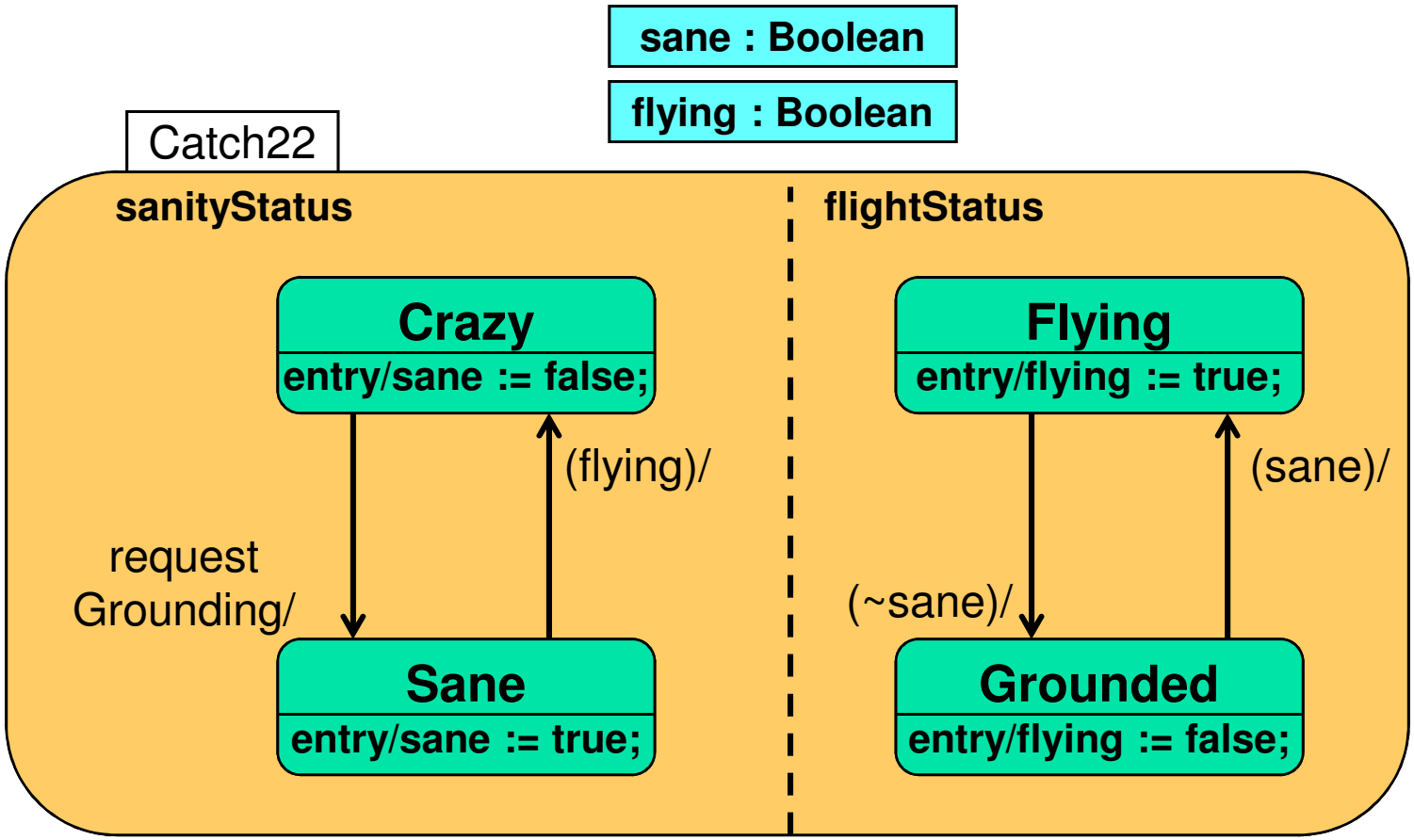
Orthogonal Regions - Semantics

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



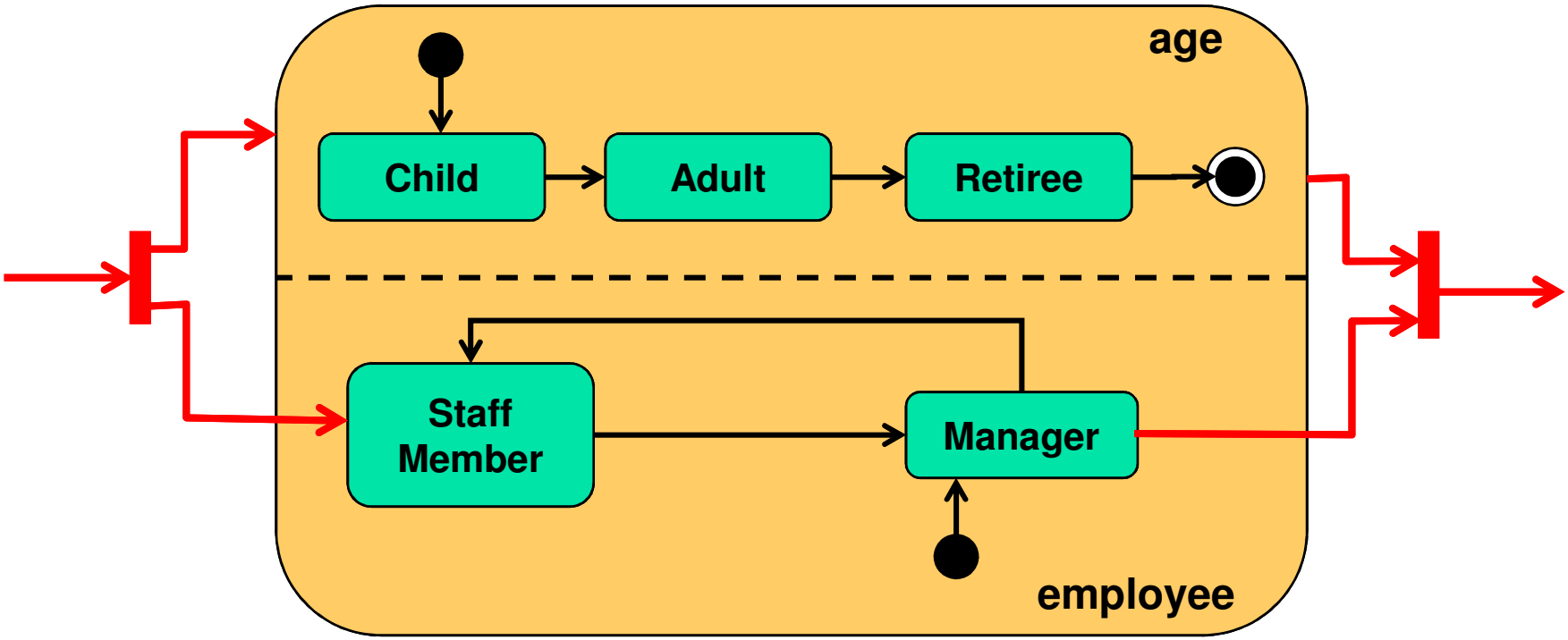
Interactions Between Regions

- Typically through shared variables or awareness of other regions' state changes



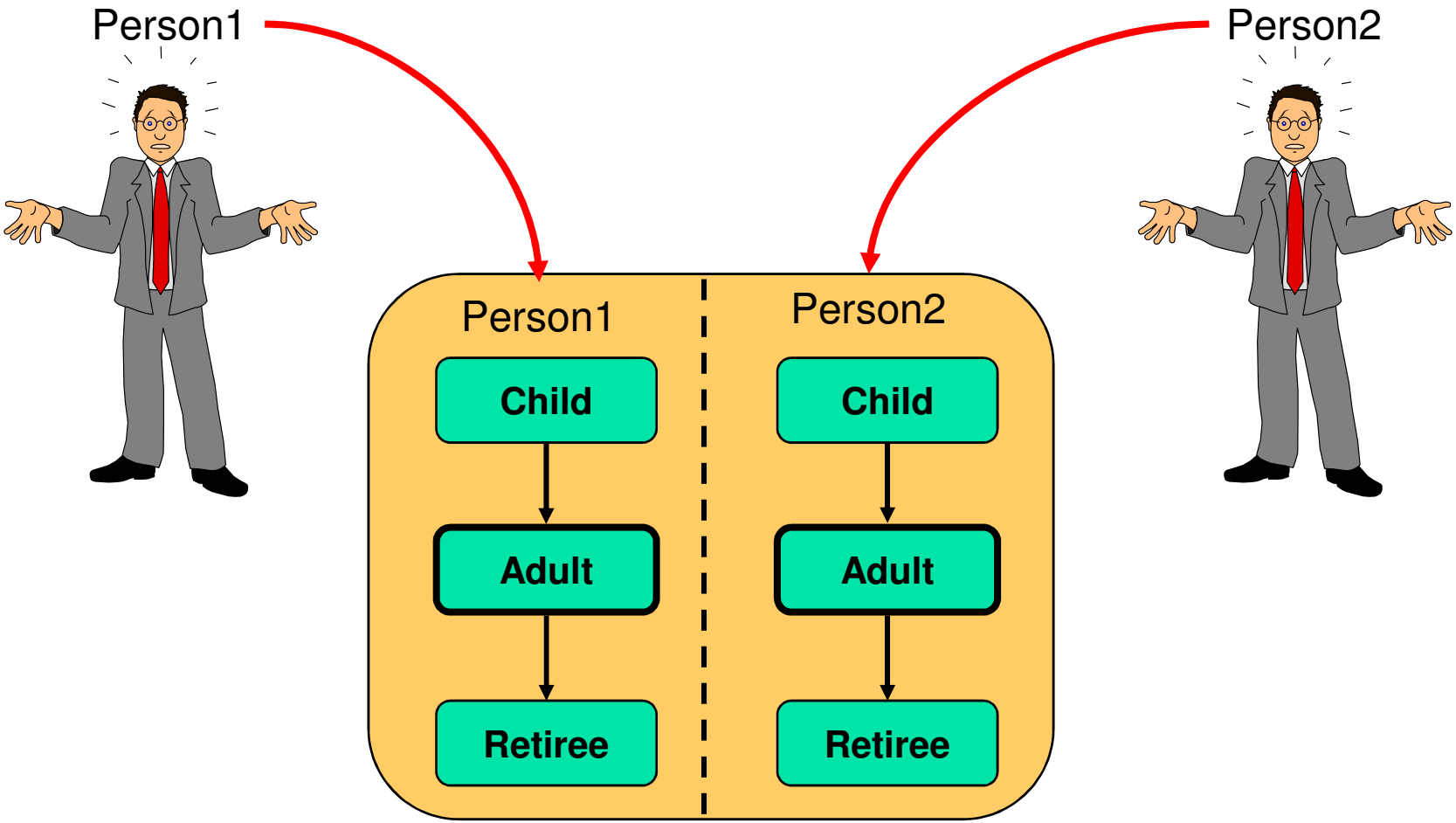
Transition Forks and Joins

- For transitions into/out of orthogonal regions:



Common Misuse of Orthogonality

- Using regions to model independent objects



Basic State Machine Concepts

Statecharts and Objects

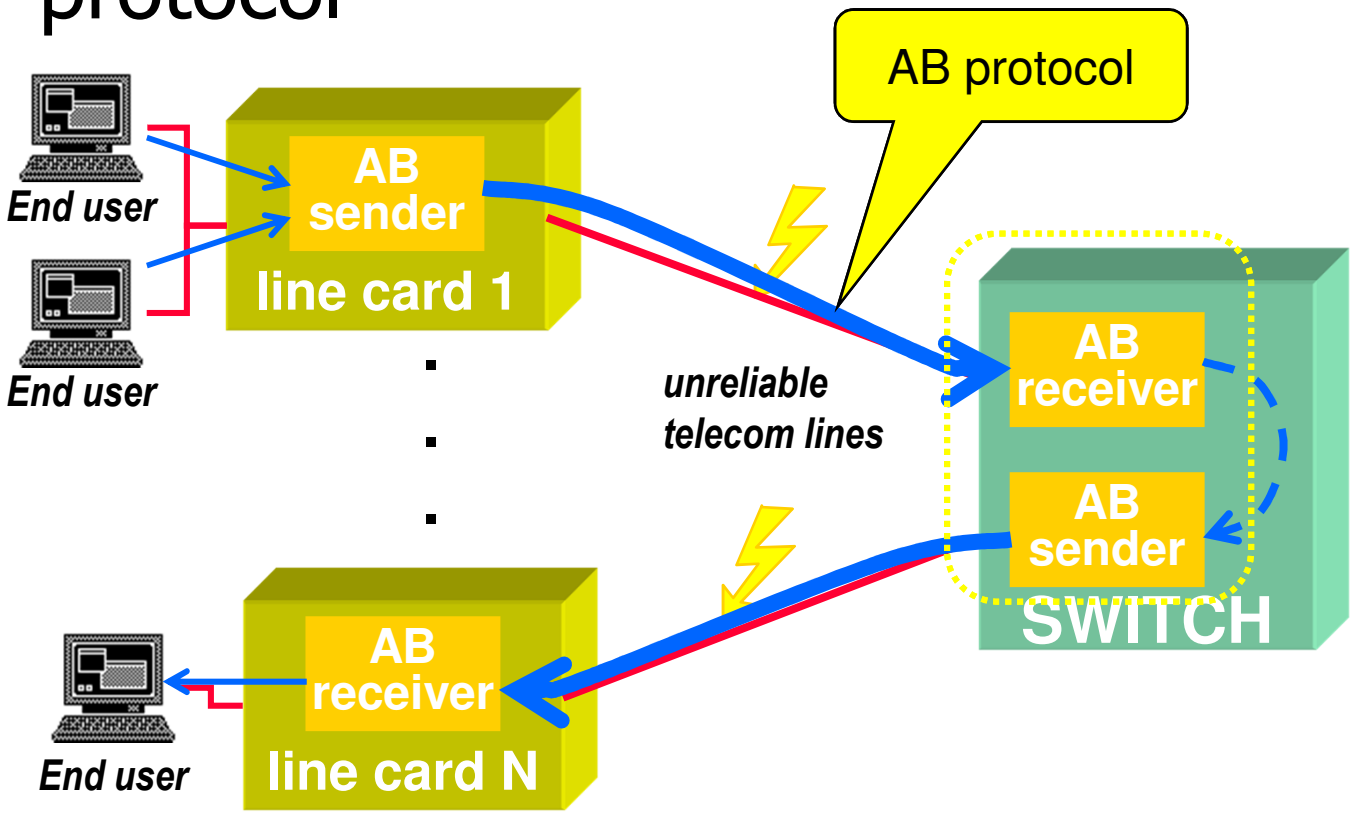
Advanced Modeling Concepts

Case Study

Wrap Up

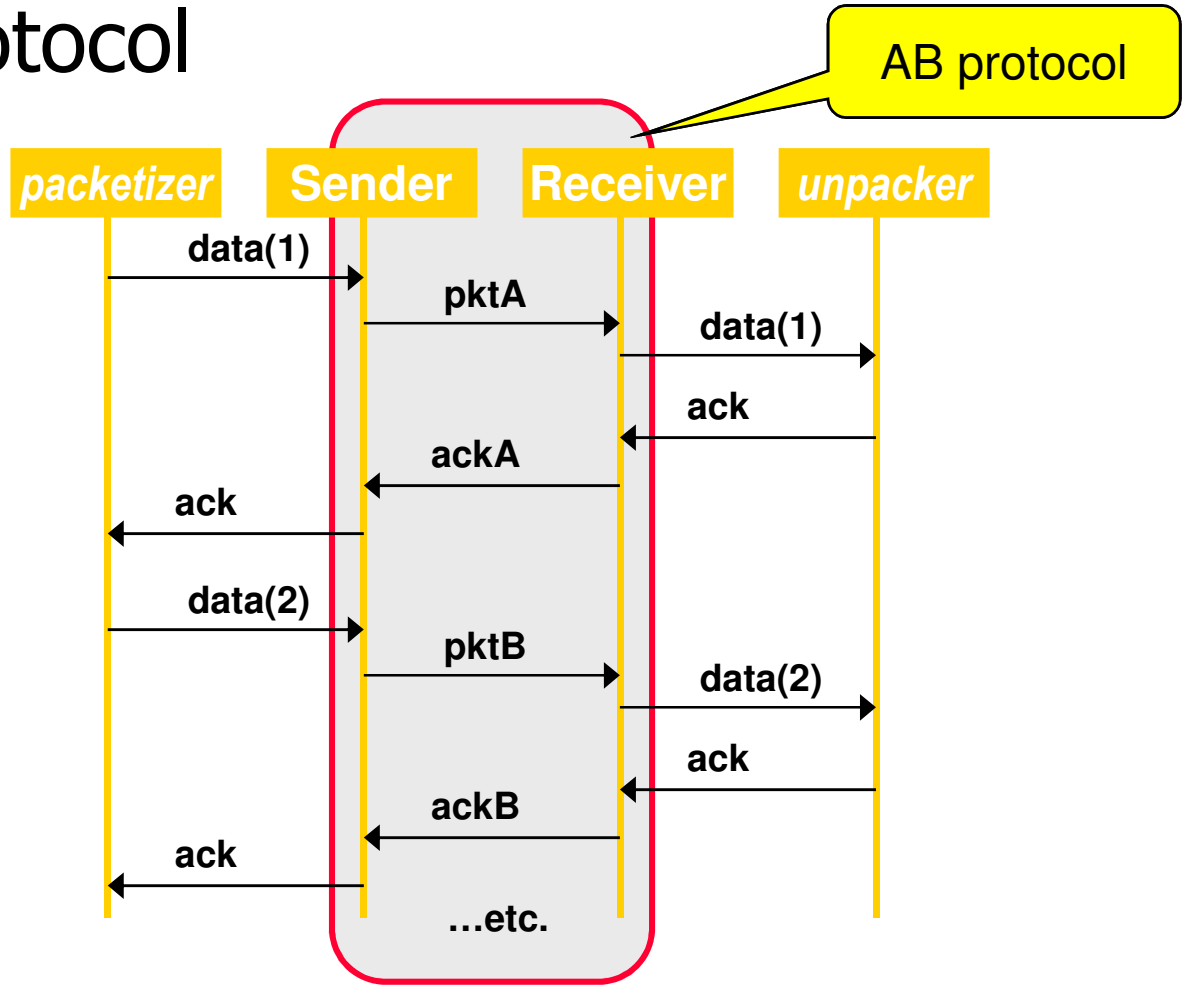
Case Study: Protocol Handler

- A multi-line packet switch that uses the alternating-bit protocol as its link protocol



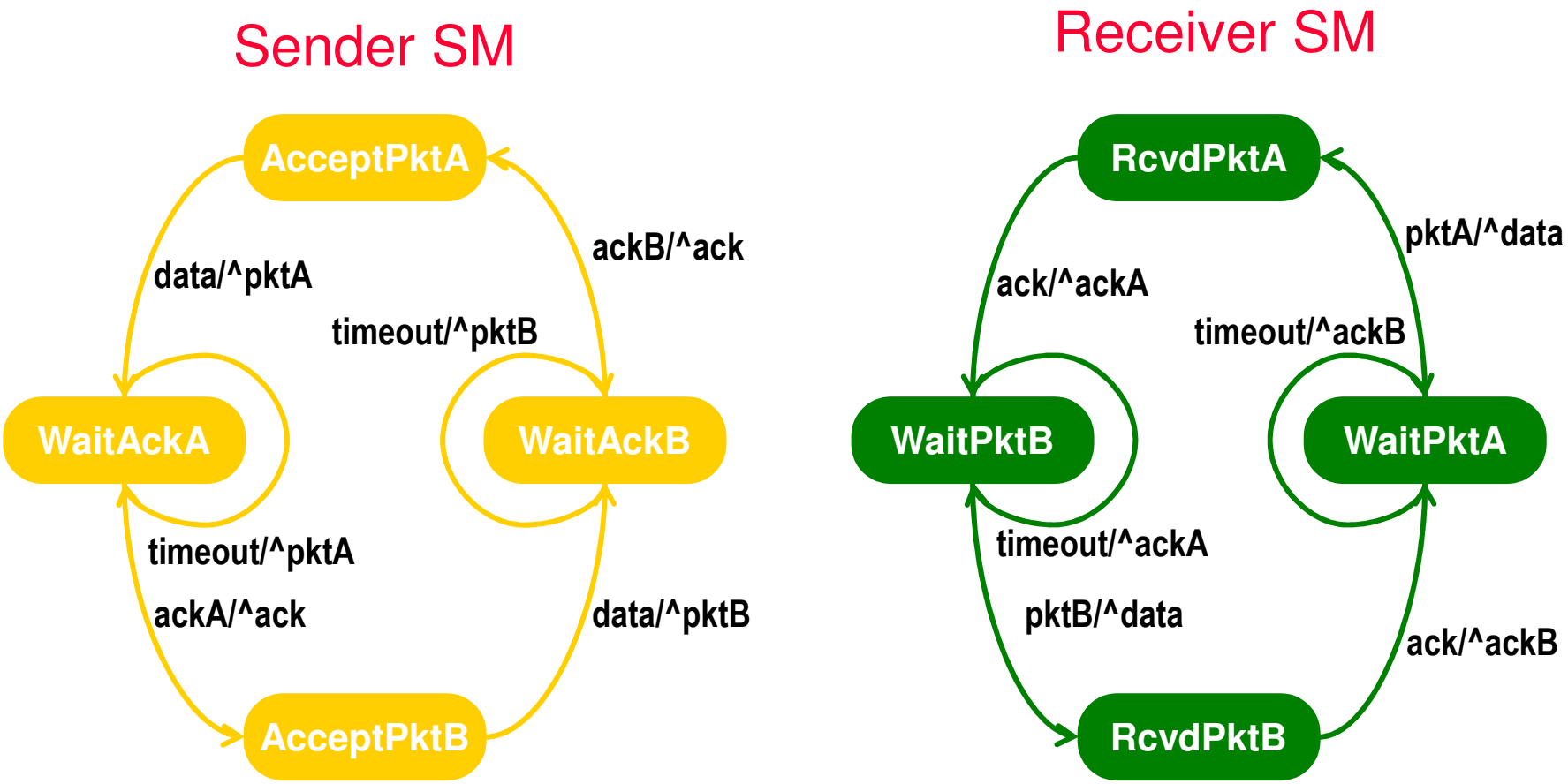
Alternating Bit Protocol (1)

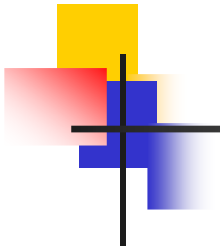
- A simple one-way point-to-point packet protocol



Alternating Bit Protocol (2)

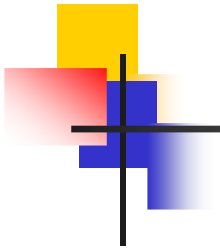
- State machine specification





Wrap Up: Statecharts

- UML uses an object-oriented variant of Harel's statecharts
  - adjusted to software modeling needs
- Used to model event-driven (reactive) behavior
  - well-suited to the server model inherent in the object paradigm
- Primary use for modeling the behavior of active event-driven objects
  - systems modeled as networks of collaborating state machines
  - run-to-completion paradigm significantly simplifies concurrency management



Wrap Up: Statecharts (cont'd)

- Includes a number of sophisticated features that realize common state-machine usage patterns:
  - entry/exit actions
  - state activities
  - dynamic and static conditional branching
- Also, provides hierarchical modeling for dealing with very complex systems
  - hierarchical states
  - hierarchical transitions
  - orthogonality