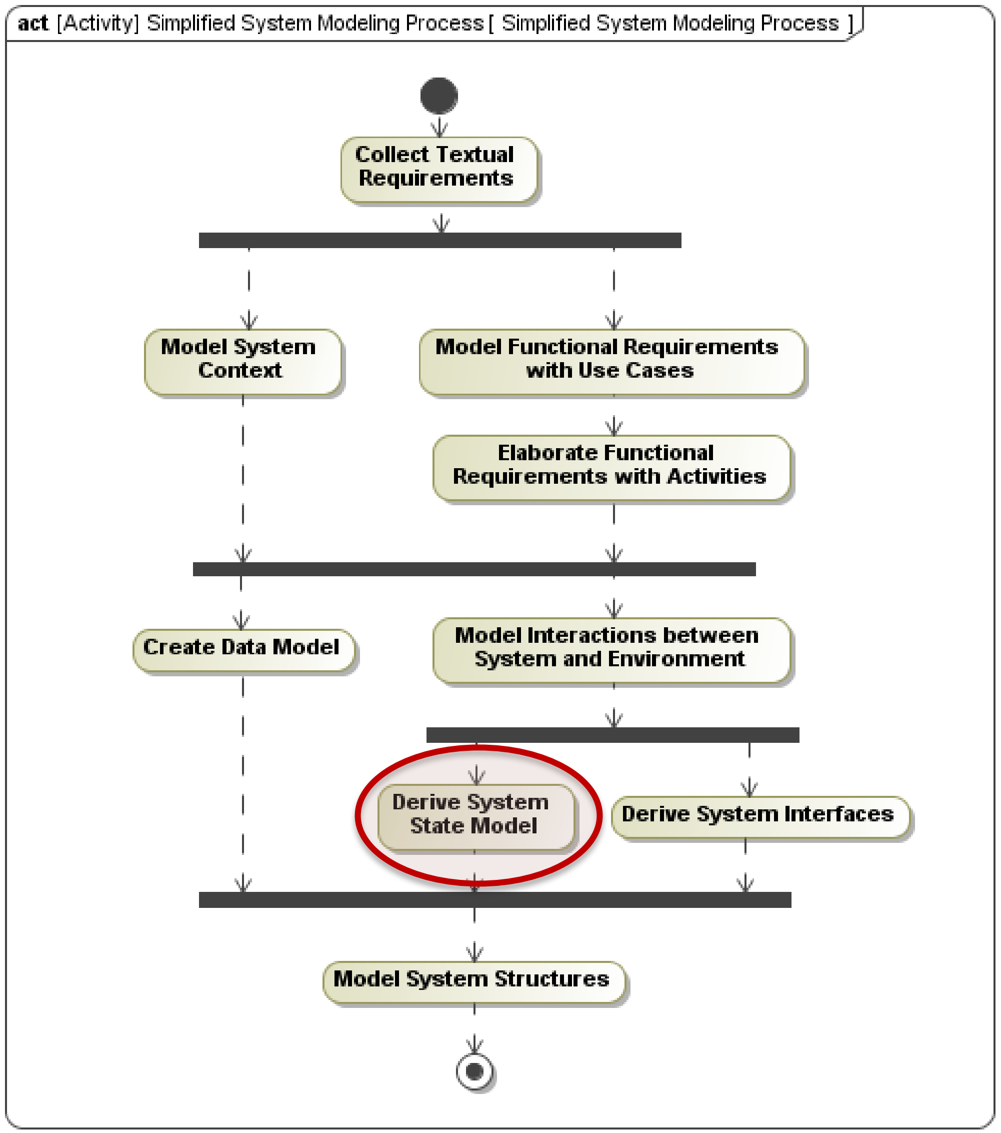
Modeling Event-Based Behavior with State Machines

Critical Embedded Systems

Gábor Guta, PhD

Prepared by
Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Dept. of Measurement and Information Systems

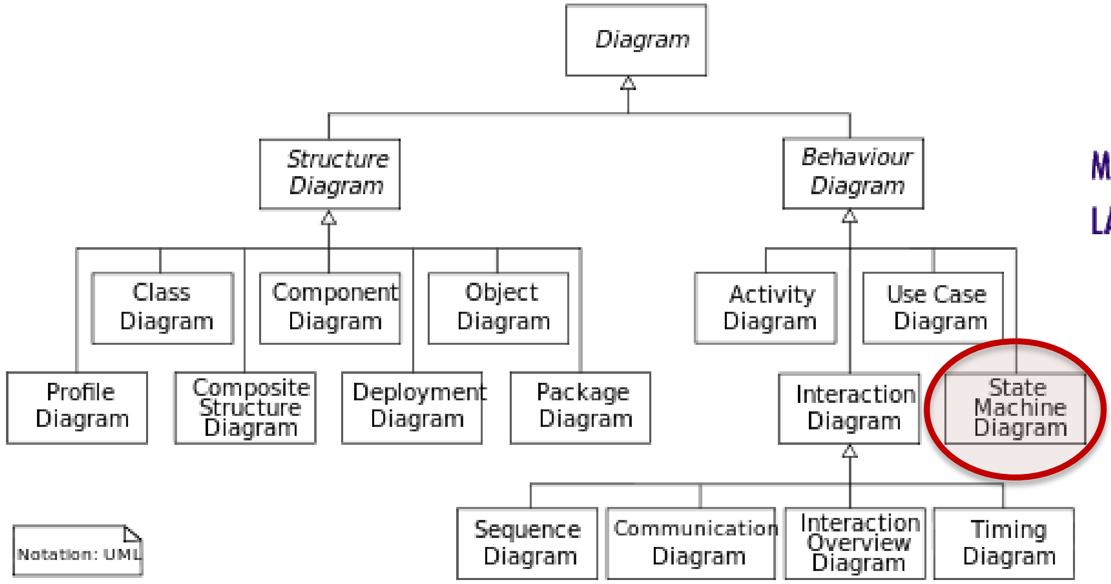
System Modeling Process

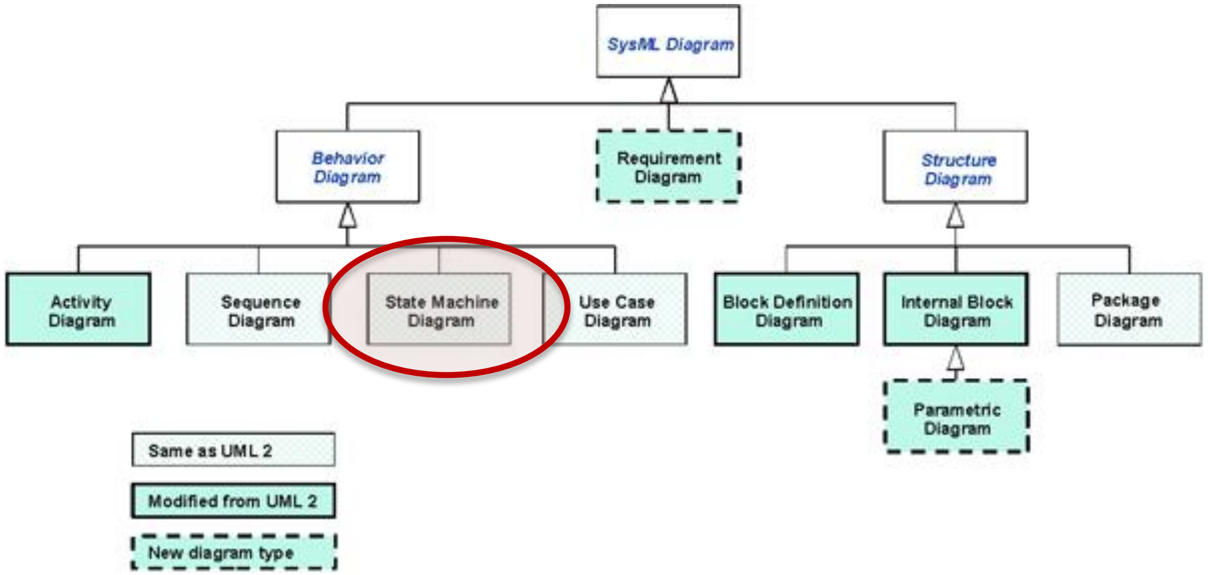

act [Activity] Simplified System Modeling Process [Simplified System Modeling Process]
Collect Textual Requirements
Model System Context
Model Functional Requirements with Use Cases
Elaborate Functional Requirements with Activities
Create Data Model
Model Interactions between System and Environment
Derive System State Model
Derive System Interfaces
Model System Structures

What is it about?

Context of the Modeling Aspect

State Machine Diagram





Modeling Aspect

What are the states of the selected component?

How it reacts to events (how it changes states)?

What are the building blocks?

Modeling Elements & Notation

Atoms of dynamic modeling

- **Event:**
  - *Asynchronous* occurrence/*happening* with parameters e.g. mouse click and its place and which button
  - *Full-fledged* object, instance of the Event class inheritance: extension of its attributes
  - Life-cycle:
    - Initialization, notification of target objects
    - Event-queues and selection
    - Processing
  - Reactive objects: react to events

Atoms of dynamic modeling II.

- **Operation**:
  - Services provided by the classes (methods)
    - client-server relation
    - can have return values
  - *Part of the class definition*
  - *Synchronous or asynchronous communication* between objects
    - e.g., method invocation
      `result = server->operation(p1, p2, ..., pn)`
- **Signal reception**
  - *Asynchronous communication* between objects

Atoms of dynamic modeling III.

- **State**:
  - The state of an object
  - Defined by:
    - value of its attributes (e.g., x<3)
    - conditions are met (e.g., operation can be executed)
- **Transition**:
  - Change of state
  - Triggered either by the incoming event or completion
- **Action**:
  - The operations to be executed by the object

Dynamic Modeling

with State Machines

State Machines

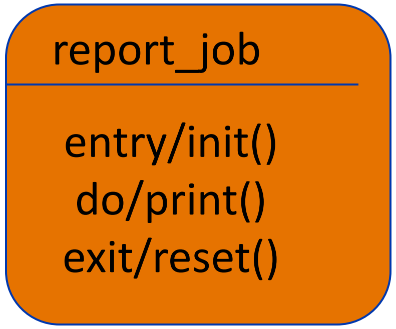
- Describes the states and state transitions of the system, of a subsystem, or of one specific object.
  - hierarchical and concurrent systems
- States
  - Concrete state:
    - Combination of possible values of attributes
    - Can be infinite
  - Abstract states: (like in State Machines)
    - Predicates over concrete states
    - One abstract state ← many concrete states
    - Hierarchical states:
      - Frequent in embedded apps (e.g. control of car brake)
- Transitions
  - Triggering Event
  - Guard
  - Action

State Machine - introduction

- For defining reactive behavior of objects
  - Responds to events:
    state transitions and actions
  - Traditional approach: state machine
- UML State Machine: extension to state machine
  - <u>State hierarchy</u>: refinement of states
  - <u>Concurrent behavior</u>: parallel threads
  - <u>Memory</u>: last active state configuration

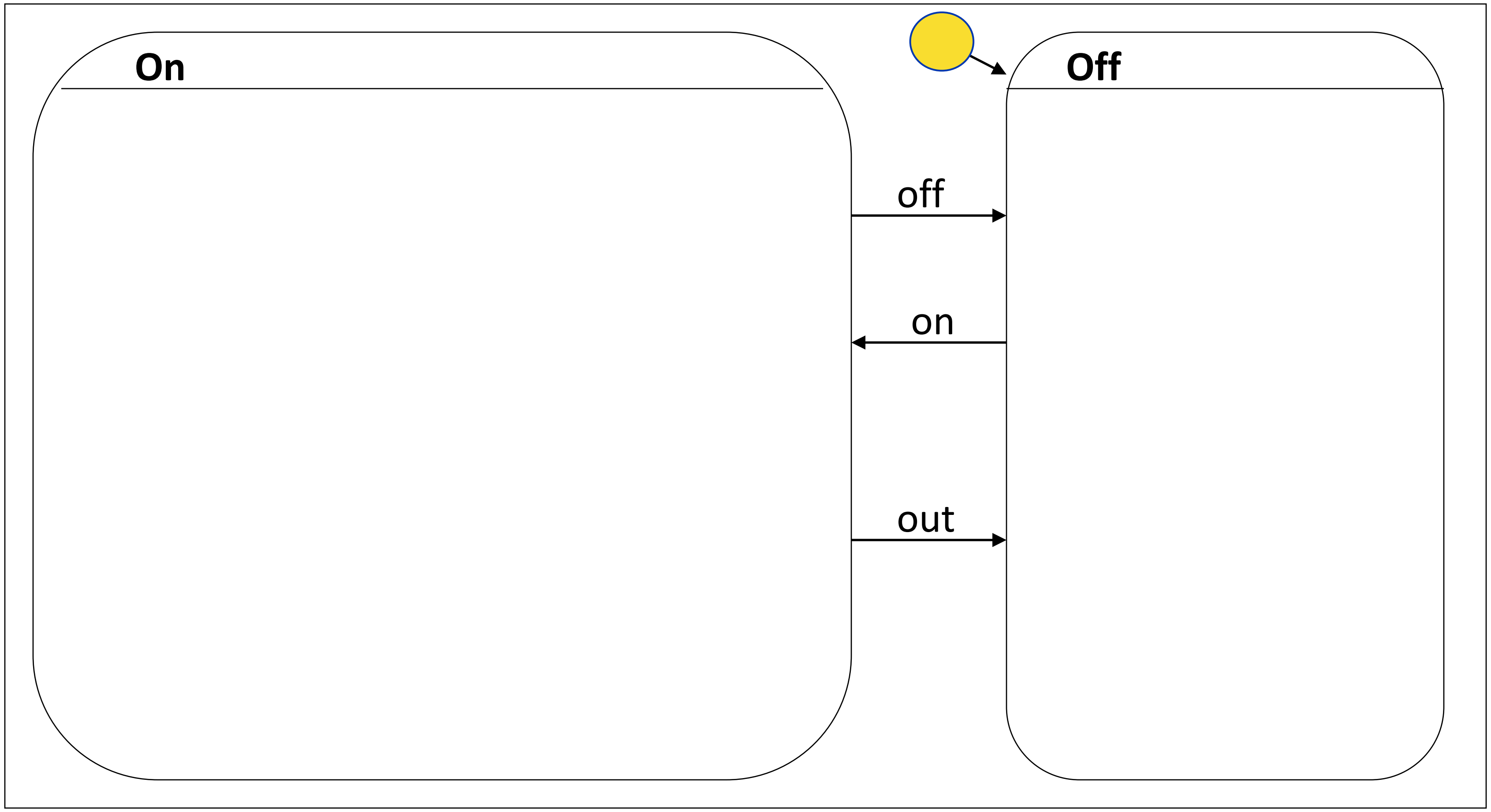
States I.

- Attributes:
  - entry action
  - do action
  - exit action

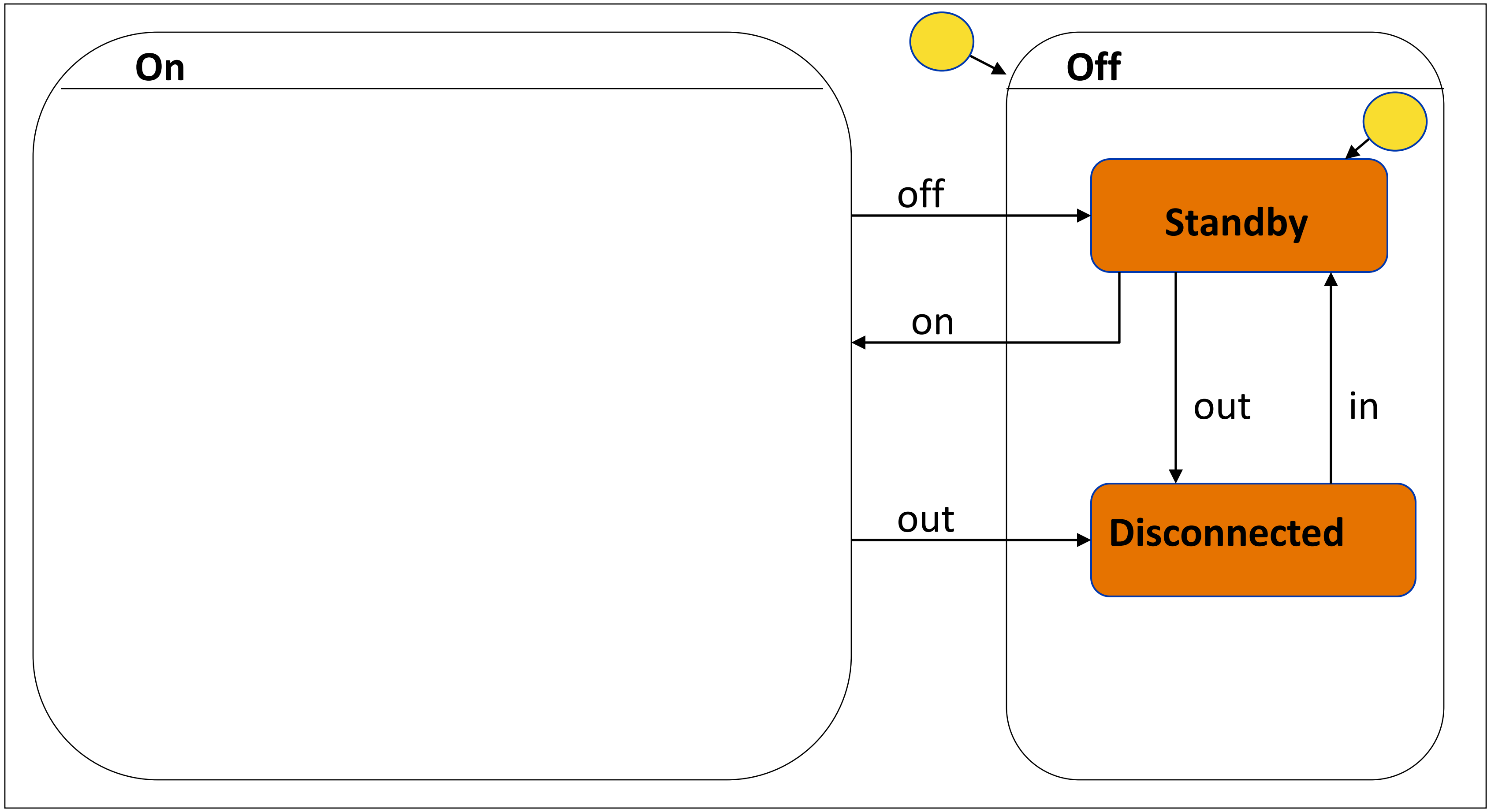


- State refinement
  - *Simple state*
  - *OR refinement*: auxiliary state machine, only one active state
  - *AND refinement*: concurrent regions (state machines), all regions are active in parallel

Example: State refinement I.

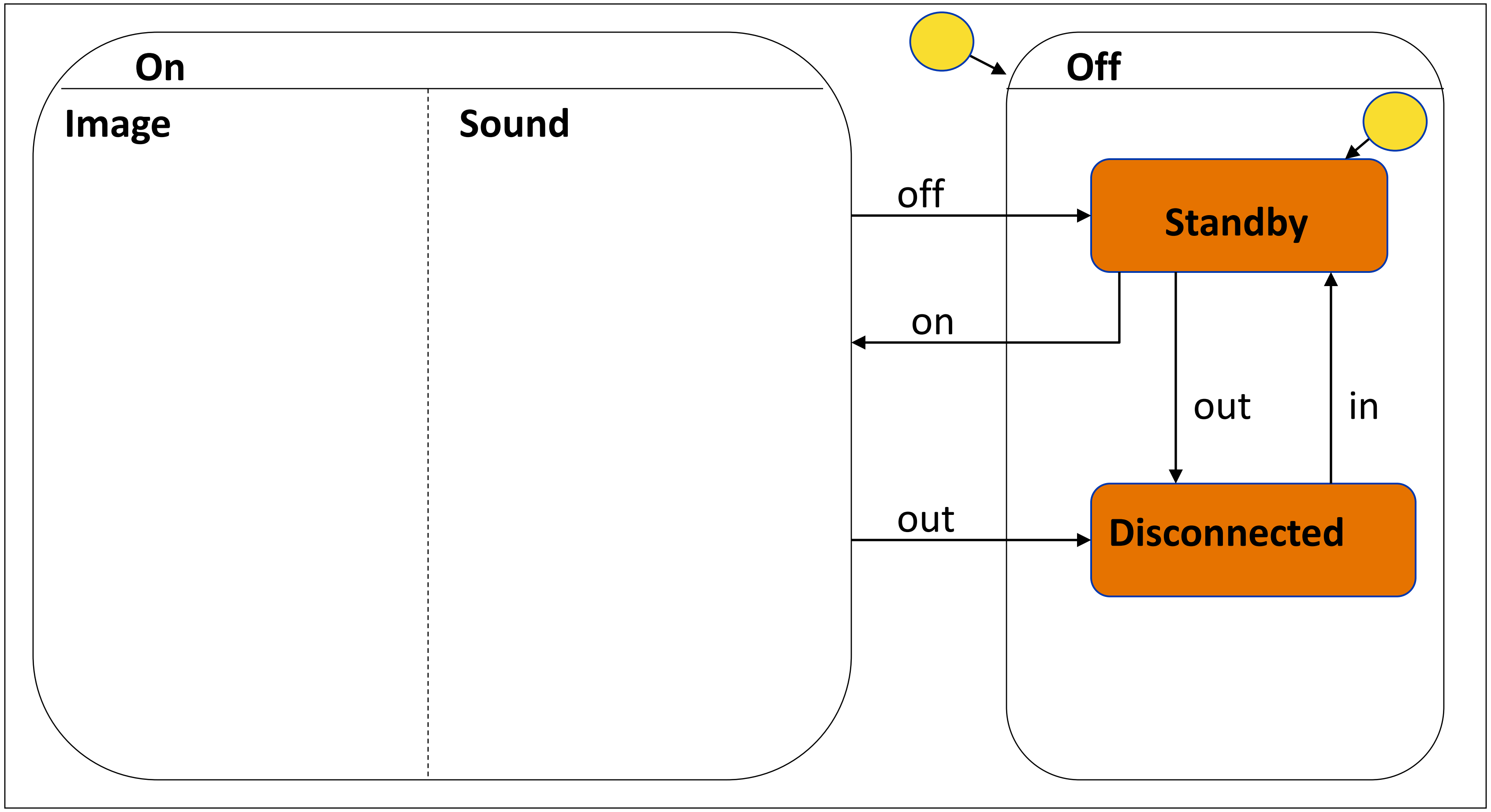


Example: State refinement II.



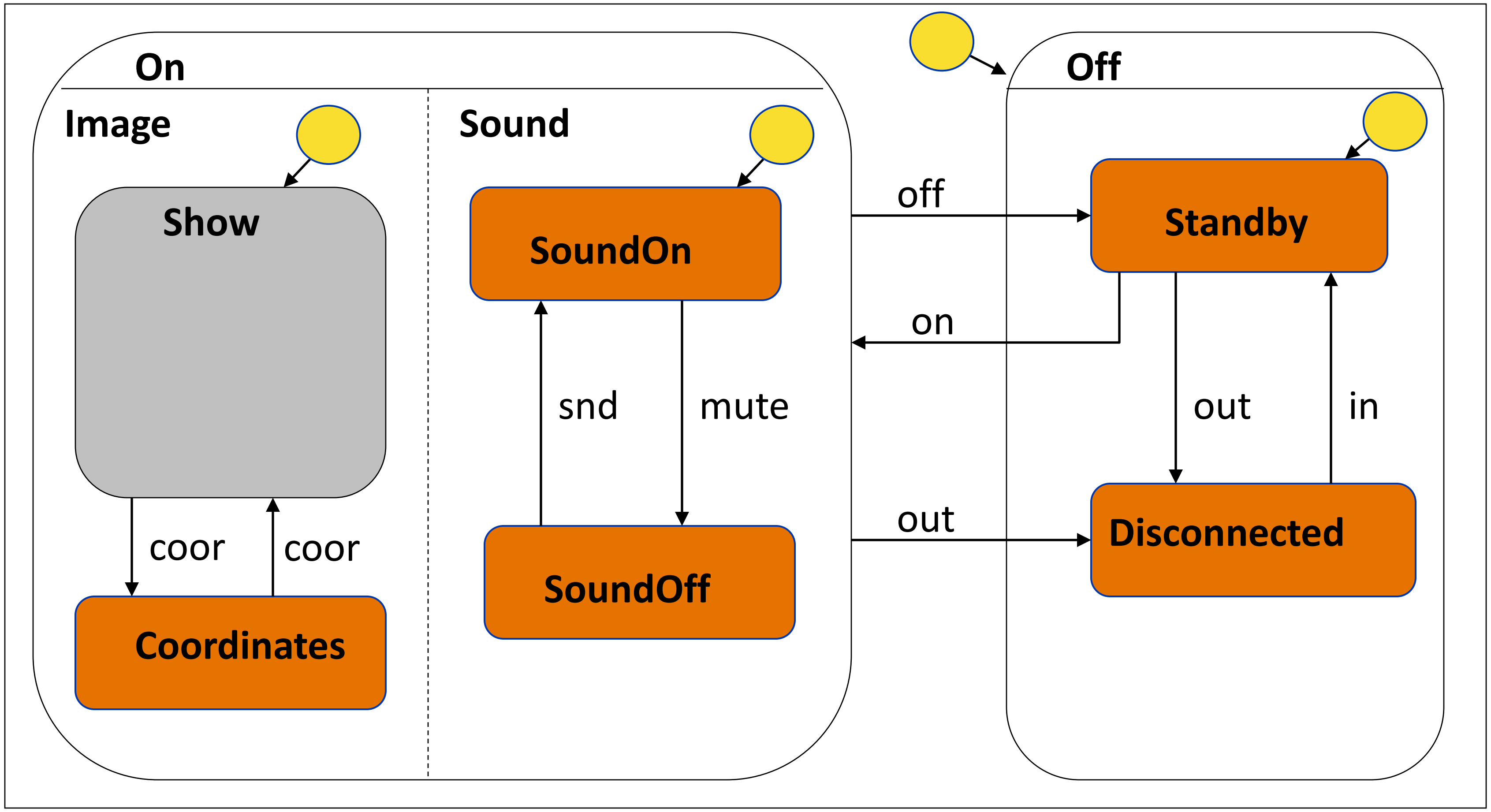
OR refinement

Example: State refinement III.

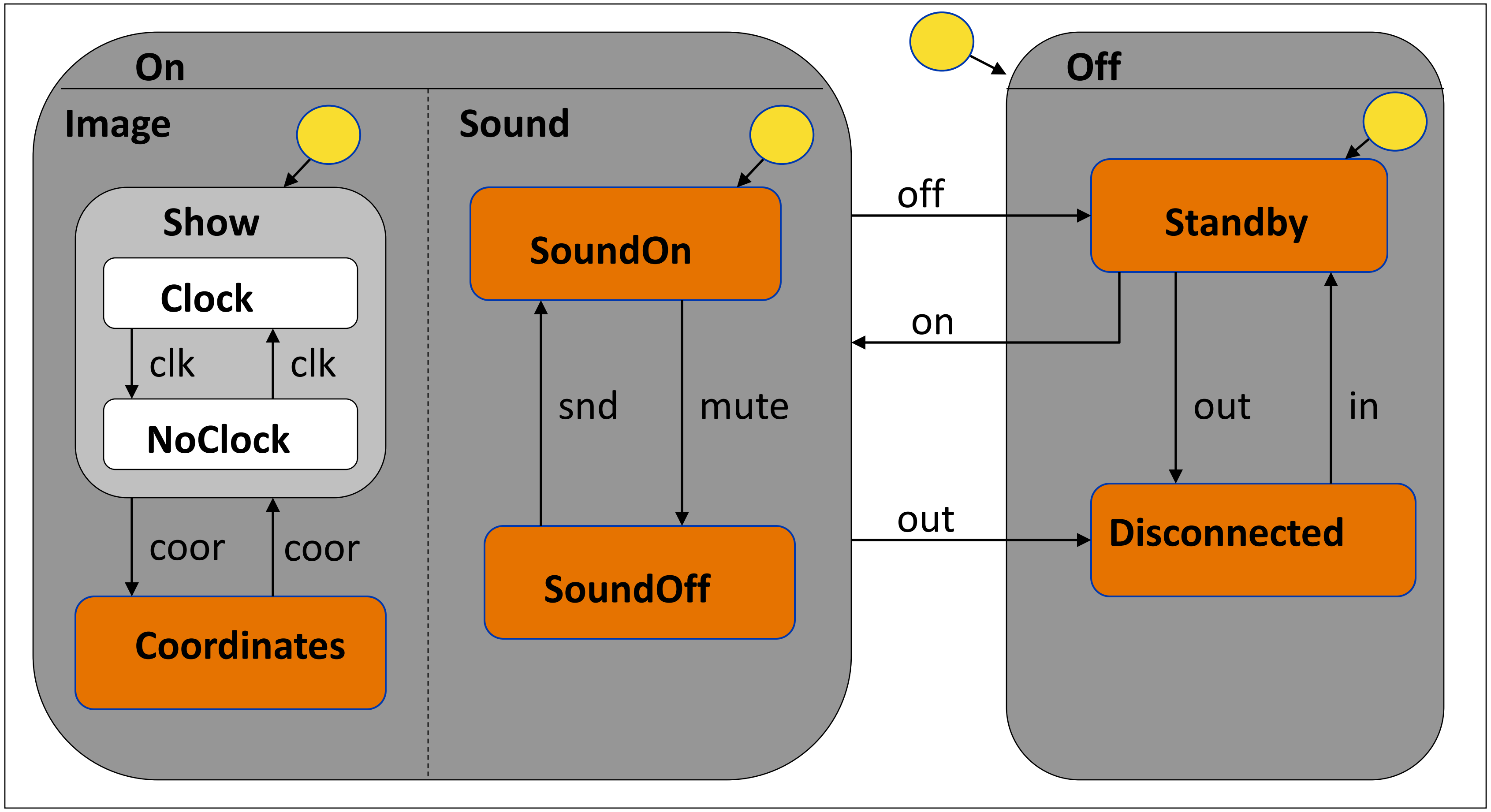


AND refinement

Example: State refinement IV.



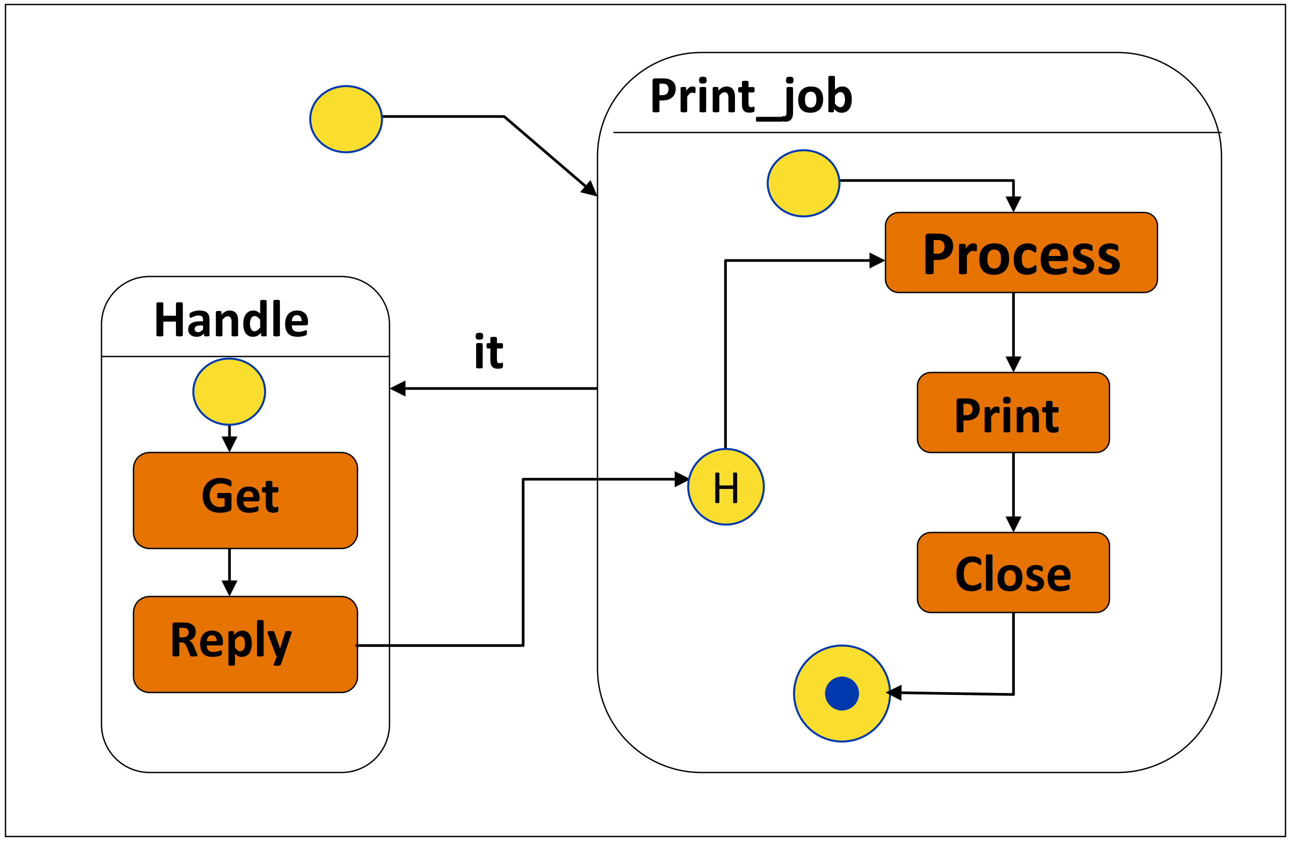
Example: State refinement V.



State II.

- History state
  - Stores the last active state configuration
  - Input transition: it sets the object to the saved state configuration
  - Output transition: defines the default state, if there were no active state since
  - Deep history state: saves the complete state hierarchy (down to the lowest substates)
- Initial state: becomes active when entered to the region
  - One in each OR refinement
  - One in each AND region
- Final state: state machine terminates

Example: History State



Transition I.

- Defining state changes
- Syntax:

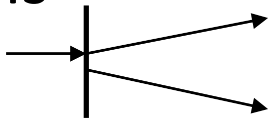
  trigger [guard] / action

  - trigger: event, triggered operation or time-out
  - guard: transition condition
    - Logic formula over the attributes of the objects and events
    - referring to a state: IS_IN(state) macro
    - Without trigger: if becomes true the transition is active
  - action: operations $\Rightarrow$ action semantics

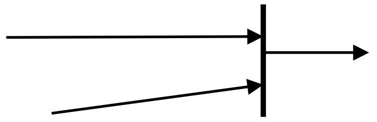
Transition II.

- Time-out trigger:
  - becomes active if the object stays in he source state for the predefined interval

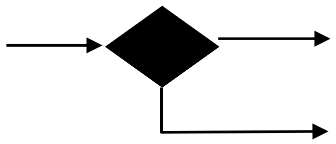
    e.g., tm(50), based on system time
- Complex transitions
  - Fork



  - Join

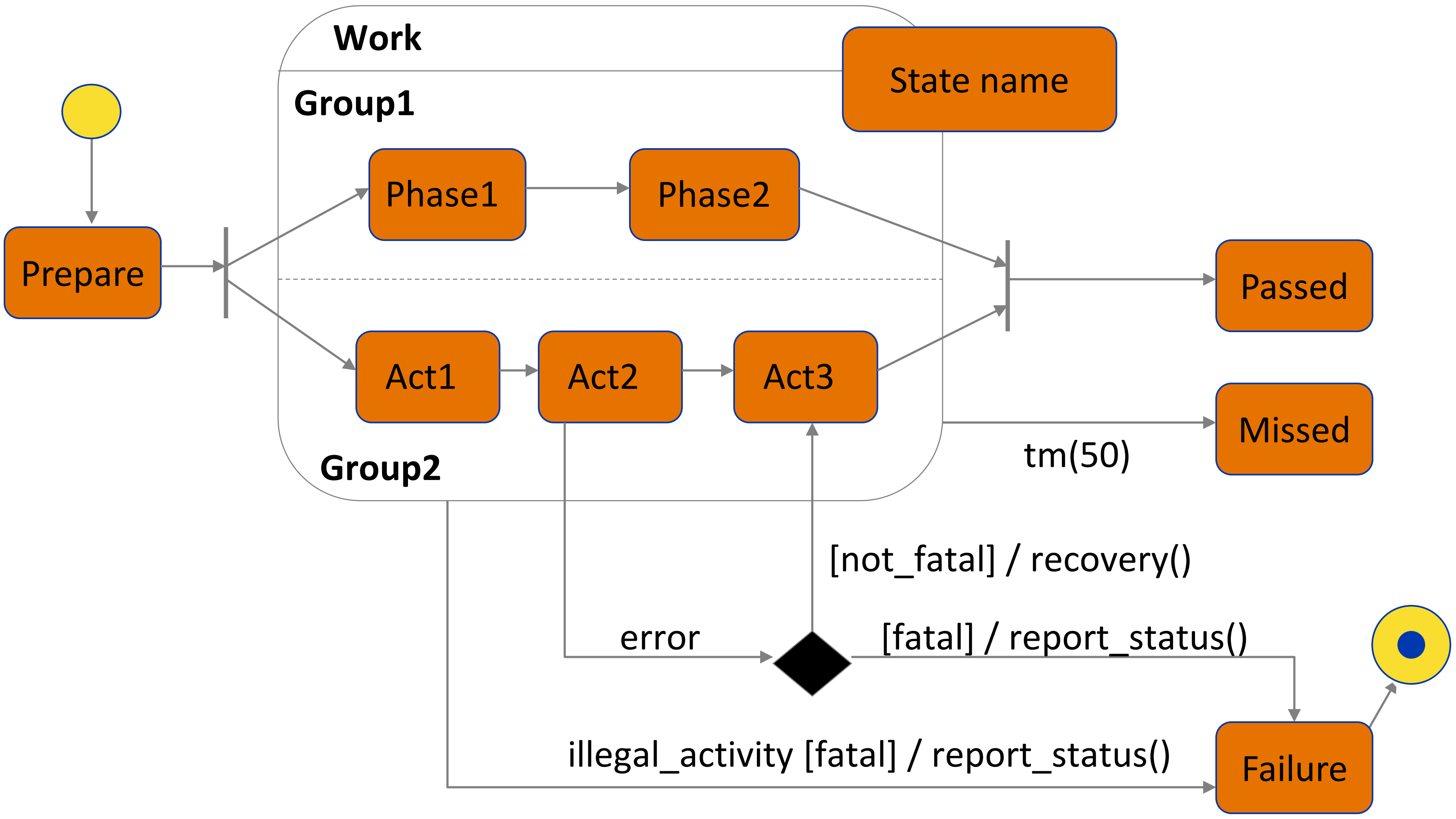


  - Condition



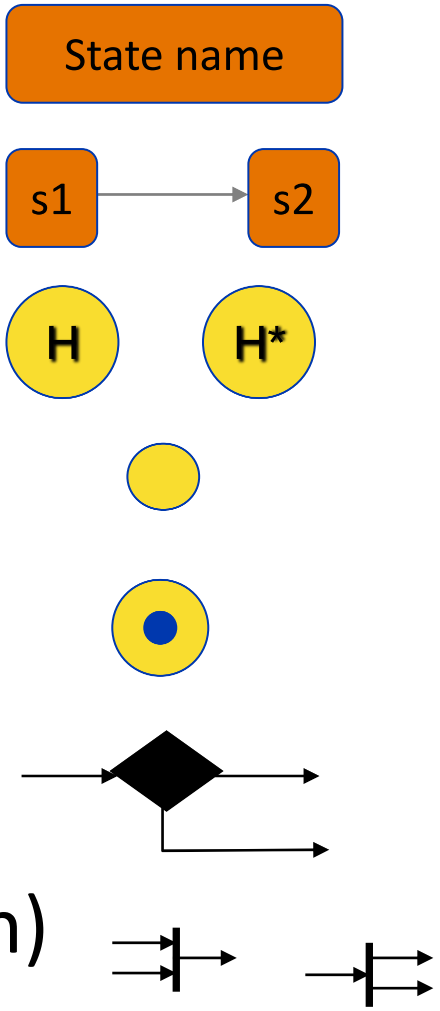
  - (Internal)
    - executes without exiting or re-entering the state in which it is defined
- Transitions between different hierarchy levels

Transition example



(Basic) State Machine elements

- State
- (Transition)
- History state
- Initial State
- Final State
- Conditional transition
- Synchronization(fork/join)



How is the model interpreted?

Semantics of the Model

Semantics: How does it work?

- Basics:
  - Hierarchical state machine (state chart)
  - Event queue + scheduler
- Semantics defines:
  Behavior in case an event occurs
  $\rightarrow$ one step of the state chart
  - (concurrent) transitions fire
  - State configuration changes
    in all region in the active state and also one substate in the OR refinement (recursively)

Semantics of State Transitions

- Separately processed events:
  - Scheduler only triggers the next event if the previous one is completely processed
    stable configuration: there is no state change without an event
- Complete processing of events:
  - The largest set of possible fireable transitions
    (all enabled transition fires, if they are not in conflict)

How does it work?: Steps of the event processing

Steps of event processing I.

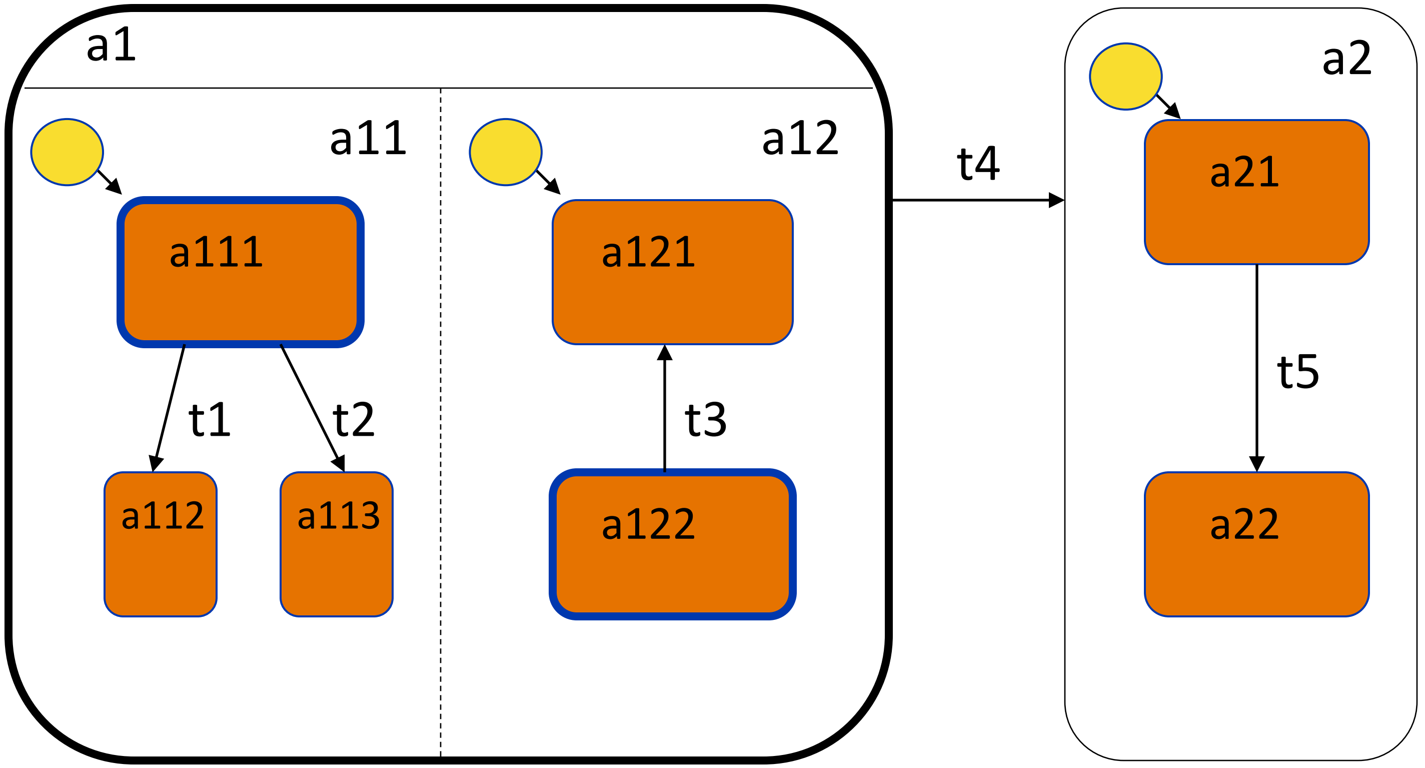
- Scheduler triggers an event for the State Machine in a stable state configuration
- Enabled transitions:
  - Source state is active
  - The event is their trigger
  - Guards are evaluated to true

Based on the number of fireable transitions

  - Only one: fire!
  - None: do nothing
  - More than one: select transitions to fire?

Example: Conflict

All transitions are triggered by the same **e** event: Which should fire?



Disabled (cannot fire): t5

Cannot fire together : (t1,t2); (t1,t4); (t2,t4); (t3,t4)

Steps of event processing II.

- Selection of fireable transitions:
  - Fireable = Enabled + Max, priority
  - Conflict: Has the same source state
    - Formally: the intersection of their left (exit) states is not empty

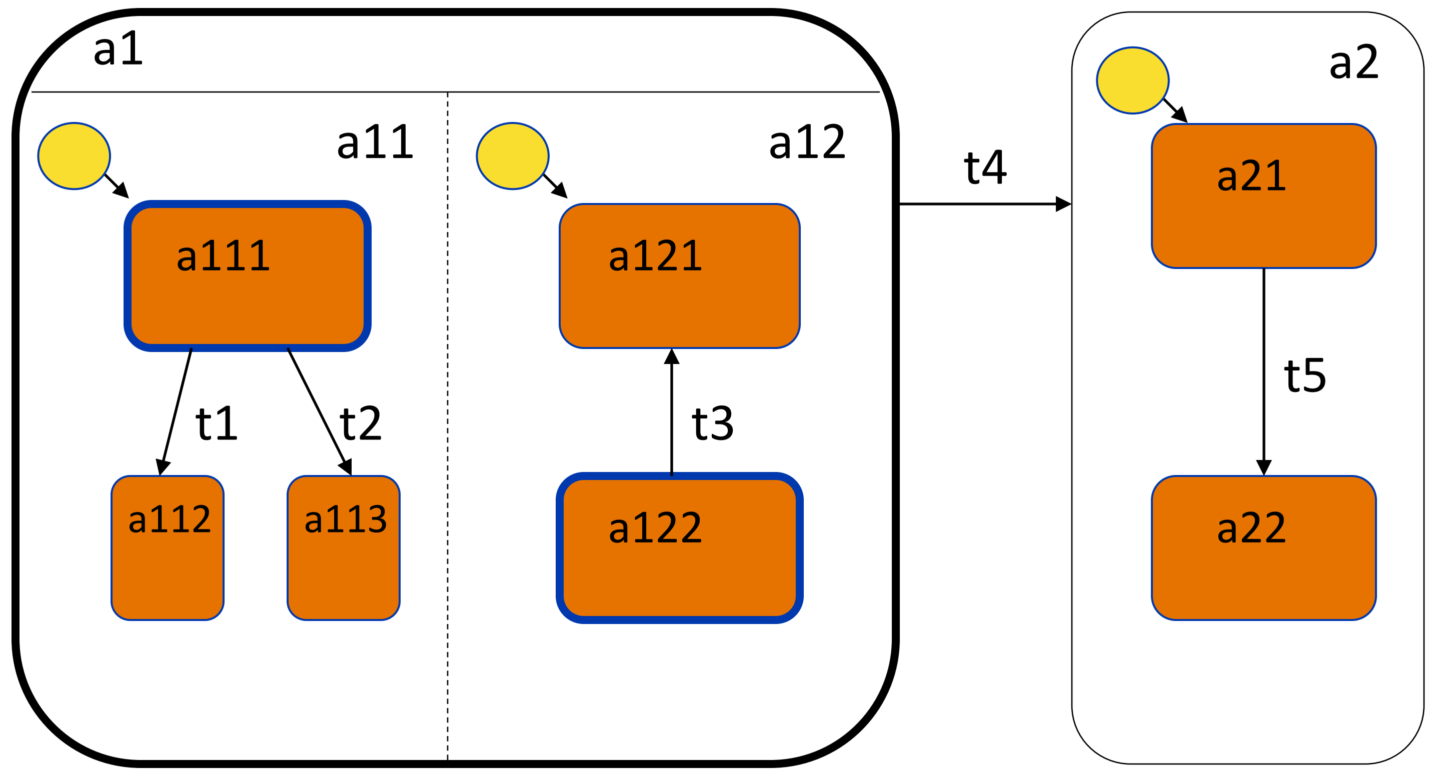
$\rightarrow$Conflict resolution → <u>priority</u>:

- Defined between two transitions (t_1 and t_2)
- $t_1 > t_2$, if and only if the source state of t_1 is a substate within the state hierarchy of t_2 („lower level”)

Steps of event processing III.

- Selection of transitions to fire:
  - Set of transitions to fire: parallel execution of concurrent transitions:
    - Maximum number of fireable transitions (= cannot be extended any further)
    - There is no conflict between any two transitions
  - Selection of this set:
    - <u>Random!</u>

Conflict resolution



Fireable: (t1,t3) or (t2,t3)

Steps of event processing IV.

- Selected transitions fire:

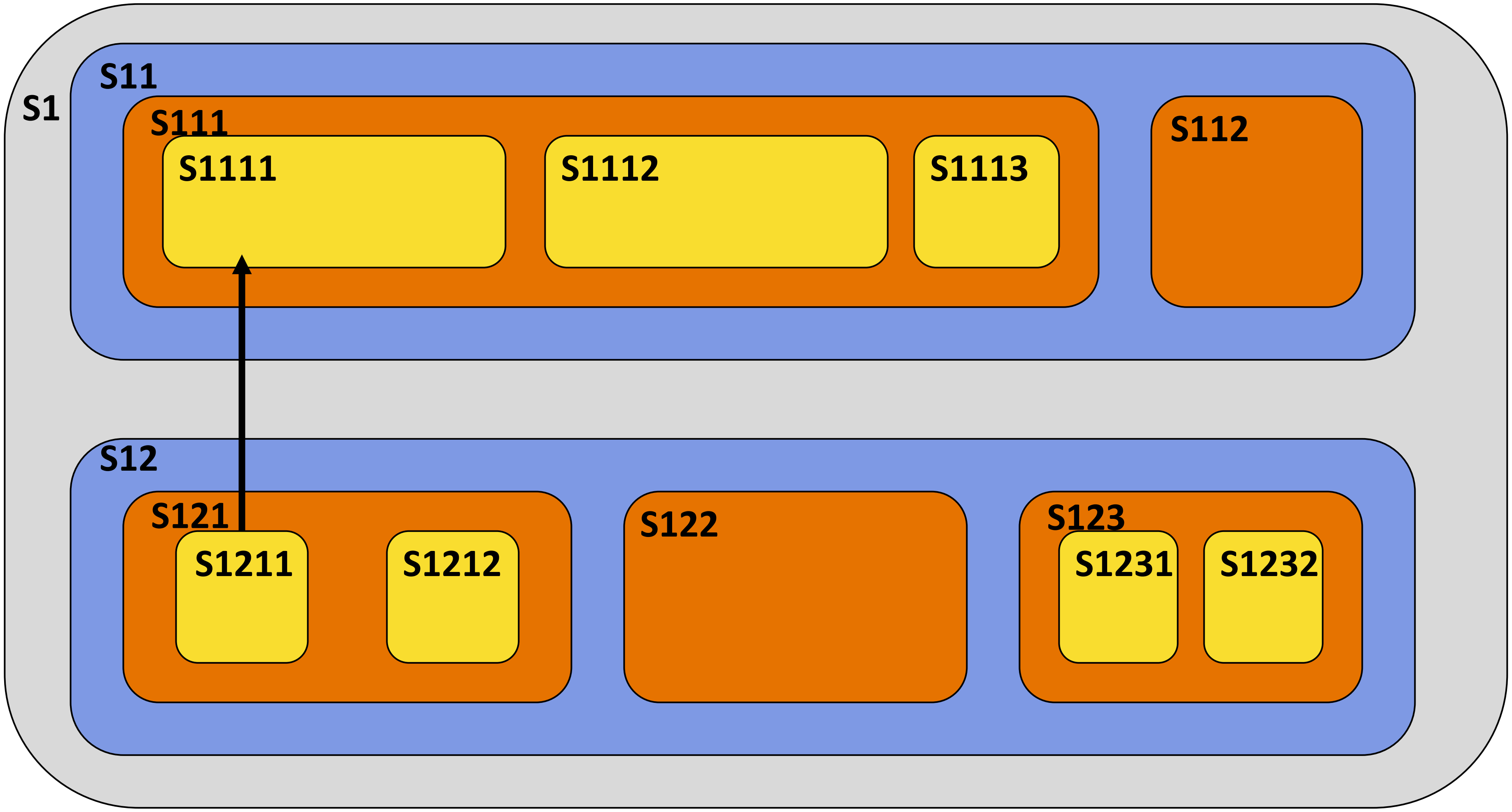
  in random order

- Firing one transition:
  - Leaving the source states from the bottom to top and execute all their *exit* operations
  - Execute the action of the transition
  - Entering the target states from top to bottom and execute the *entry* actions $\rightarrow$ new state configuration

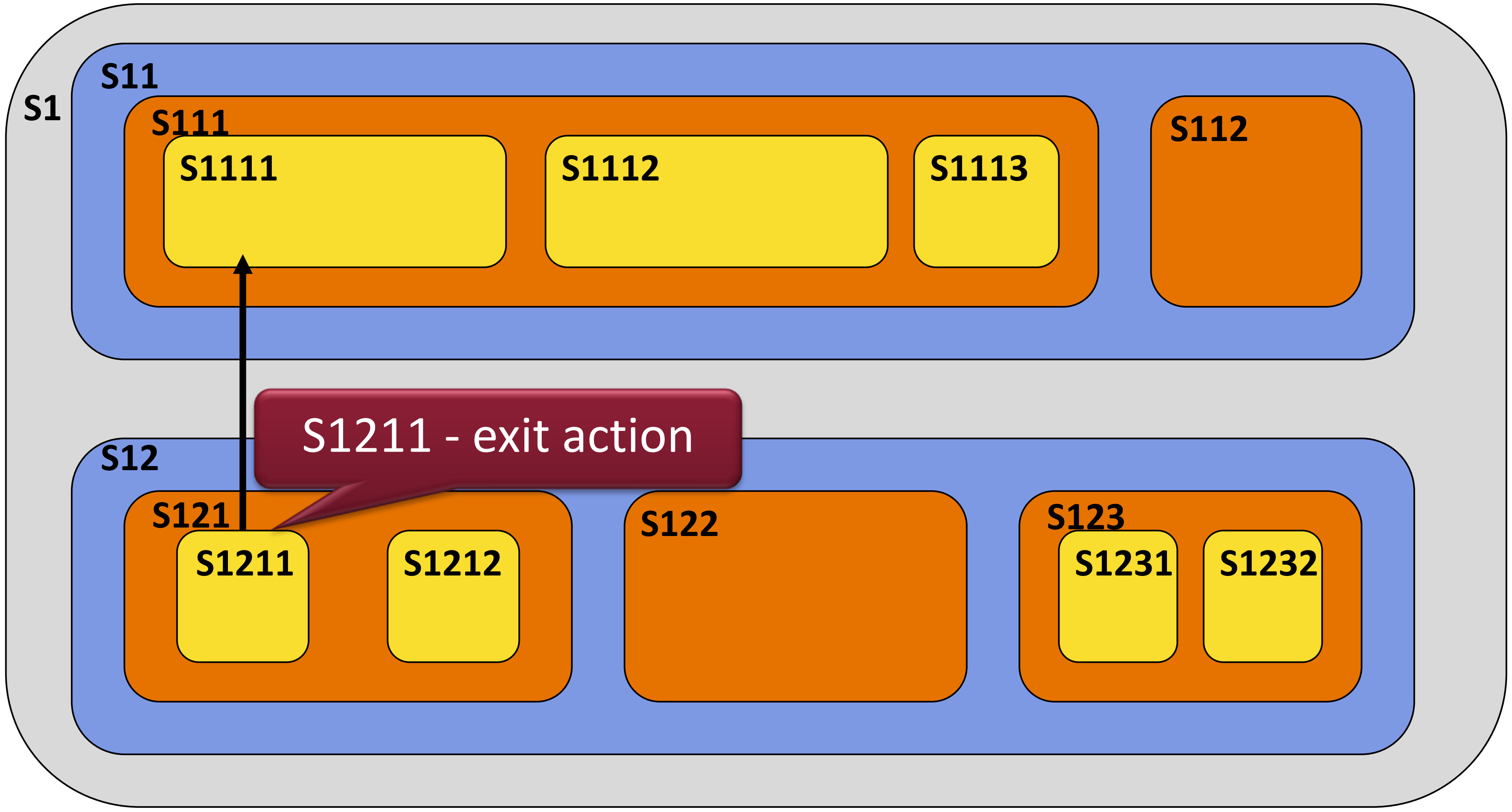
Steps of event processing V.

- Entering a new state configuration:
  - Simple target state: part of the state configuration
  - Non-concurrent superstate: direct target of one of its substate or its initial state
  - Concurrent target state: all of its regions have to have an active state either as direct target state or with initial state
  - History state : the last active state configuration
    if there is none: the target state of the history state

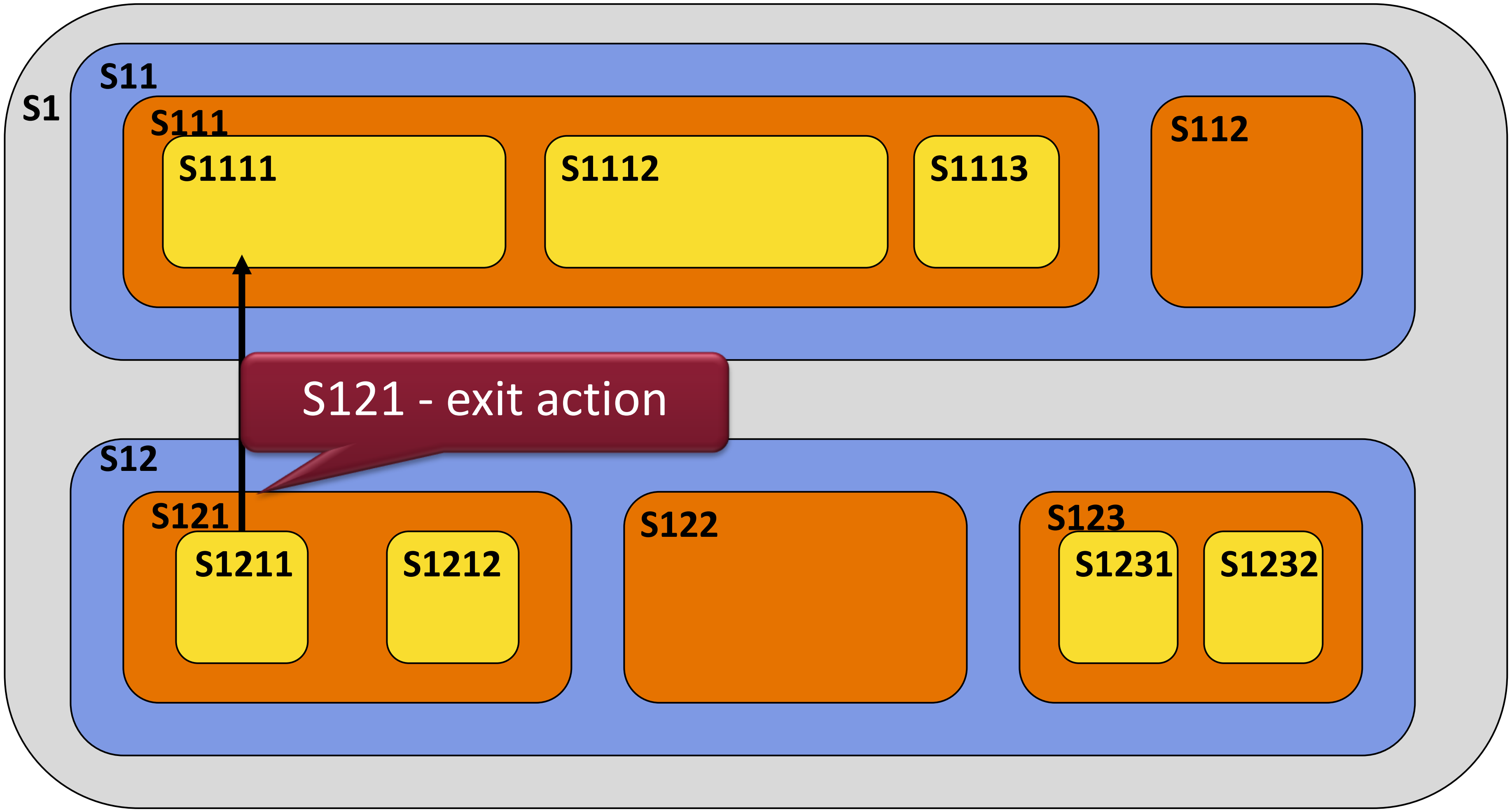
State transition example



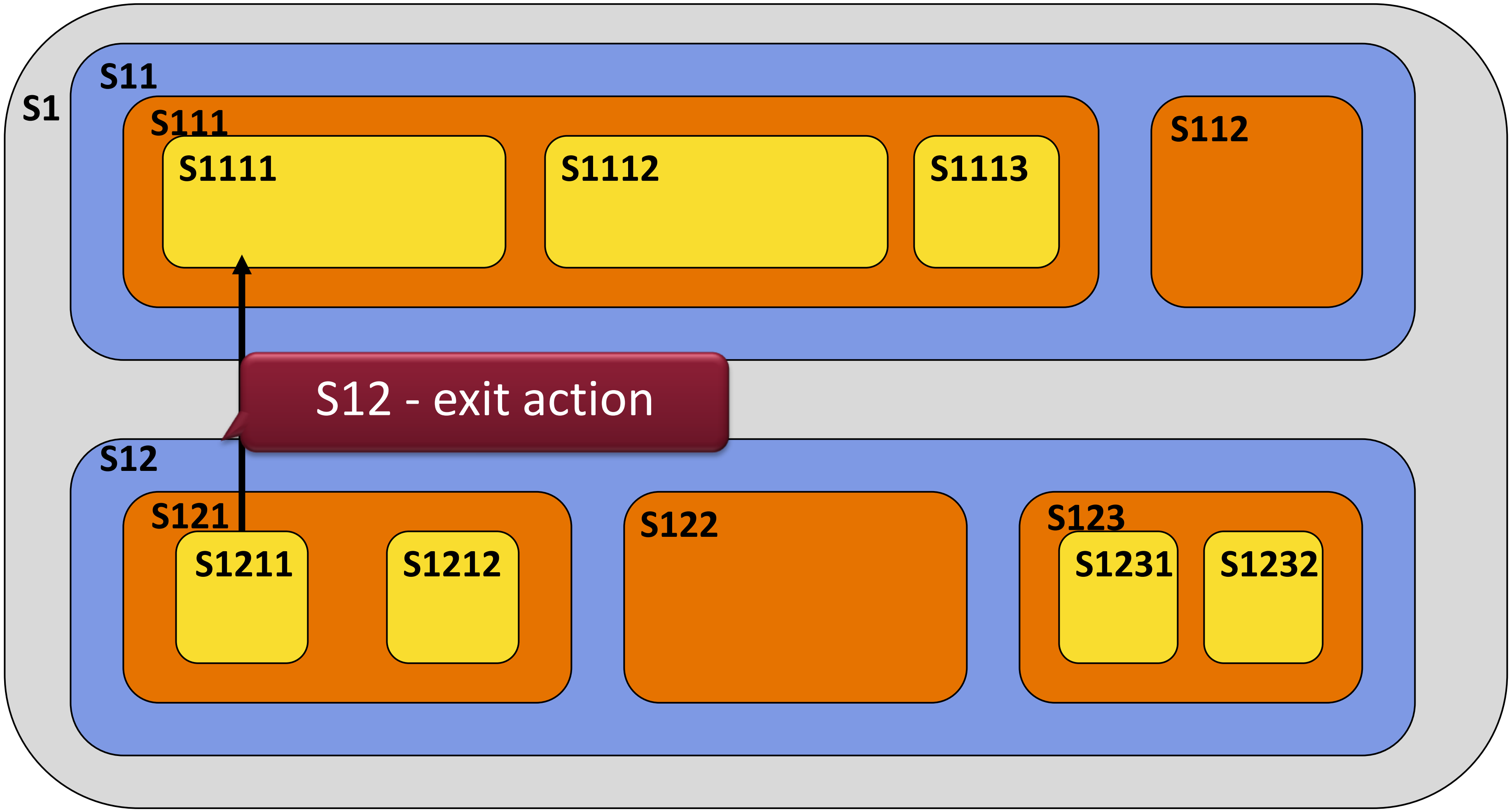
State transition example



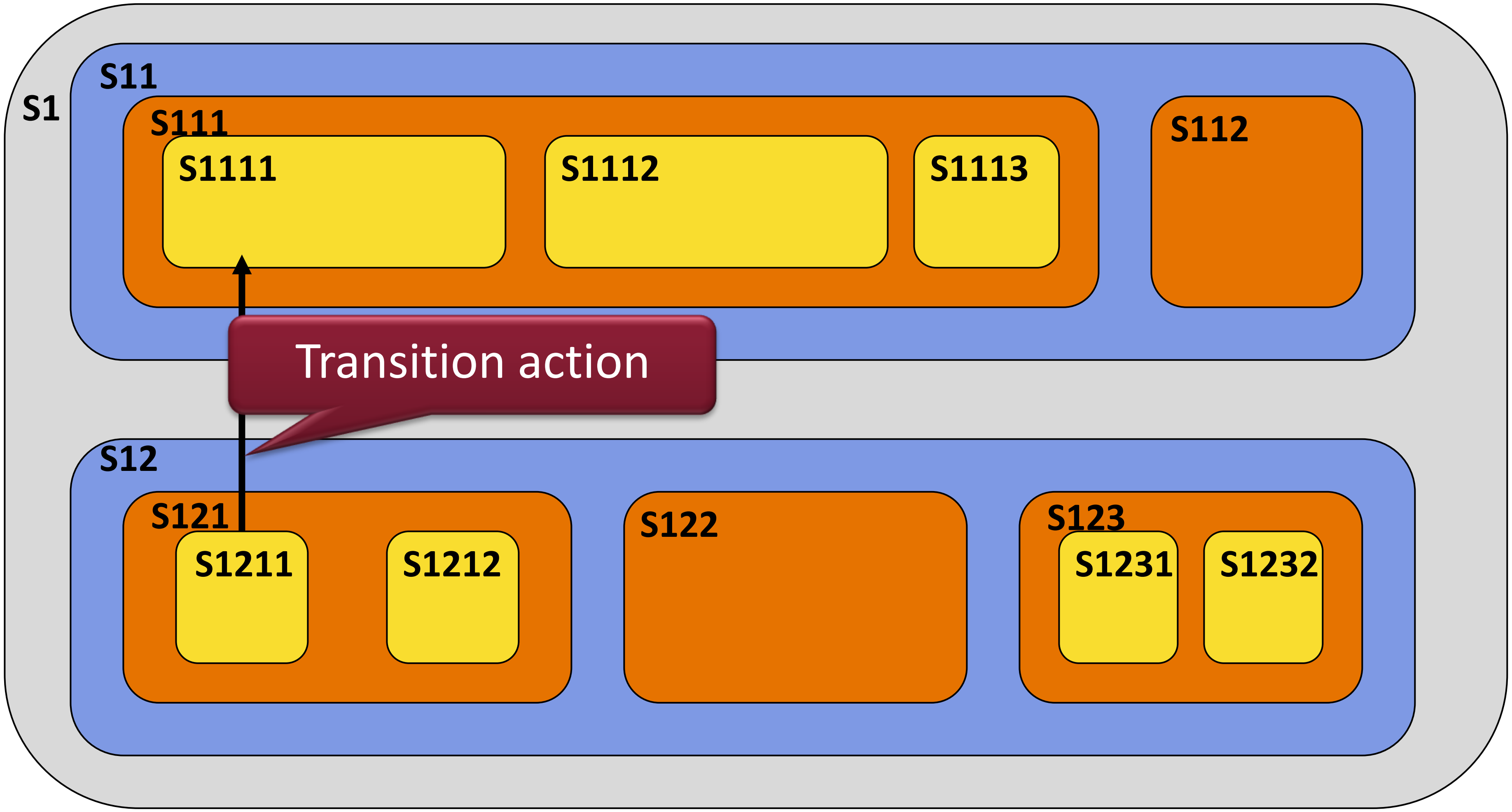
State transition example



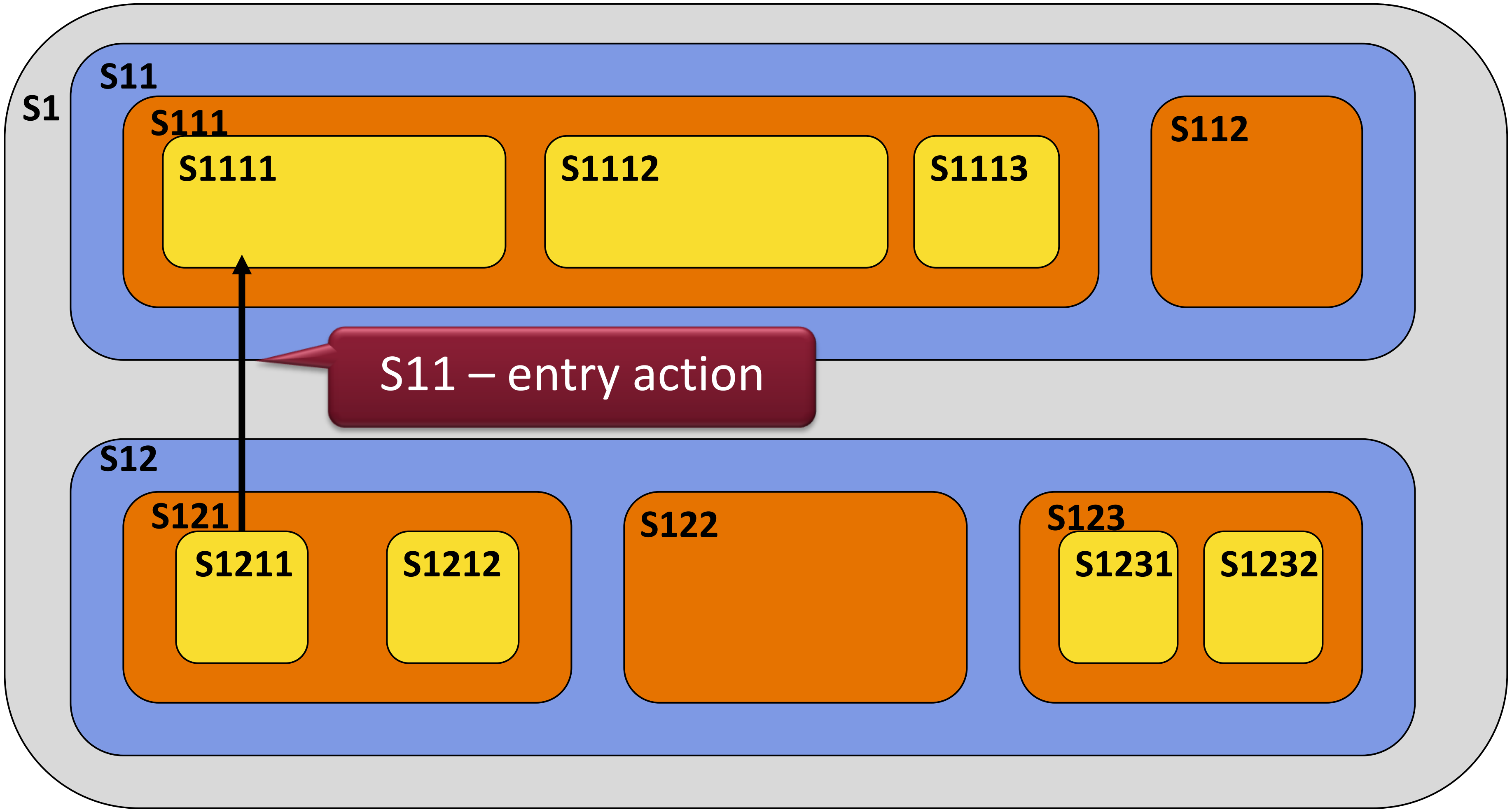
State transition example



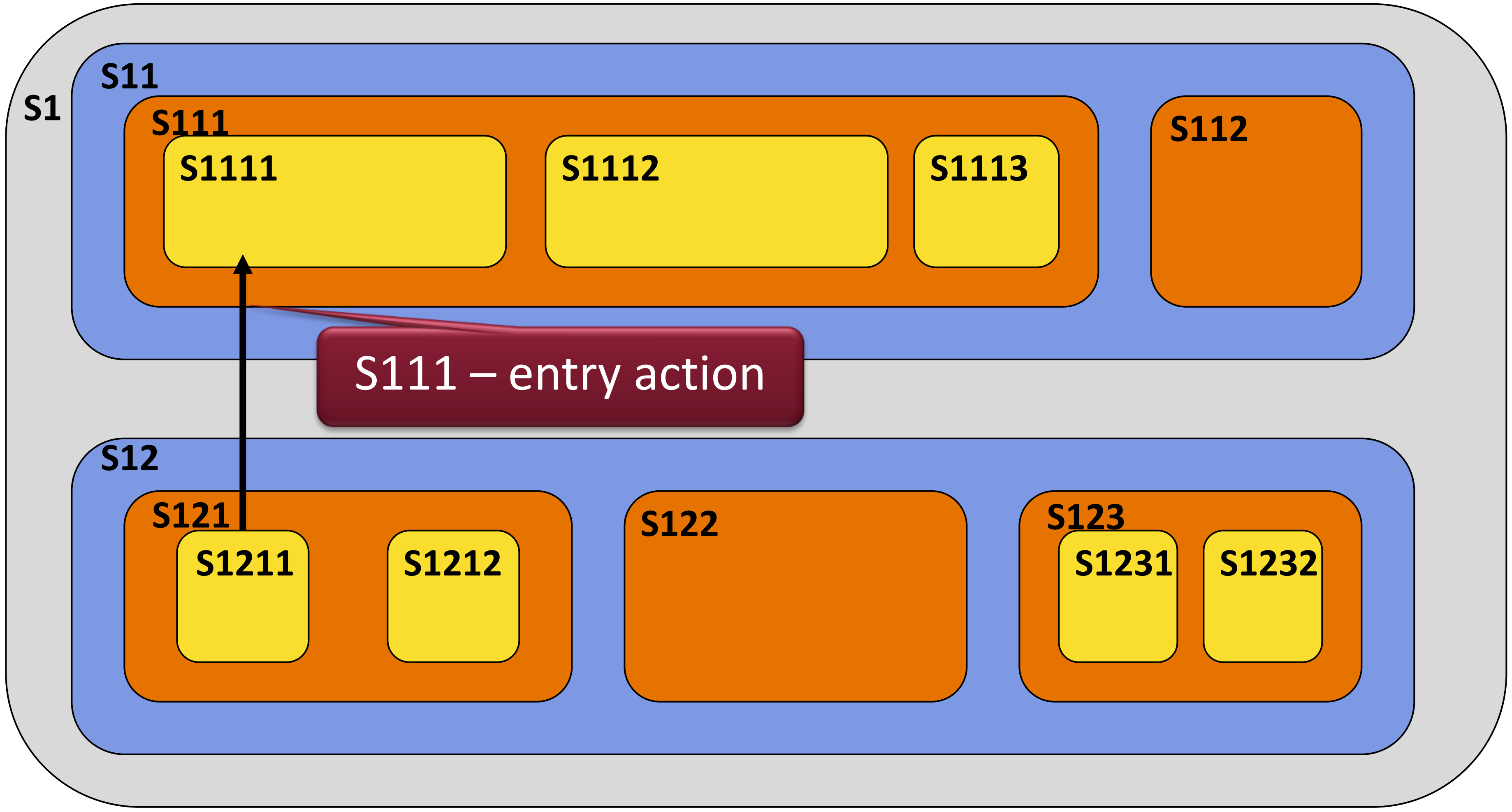
State transition example



State transition example

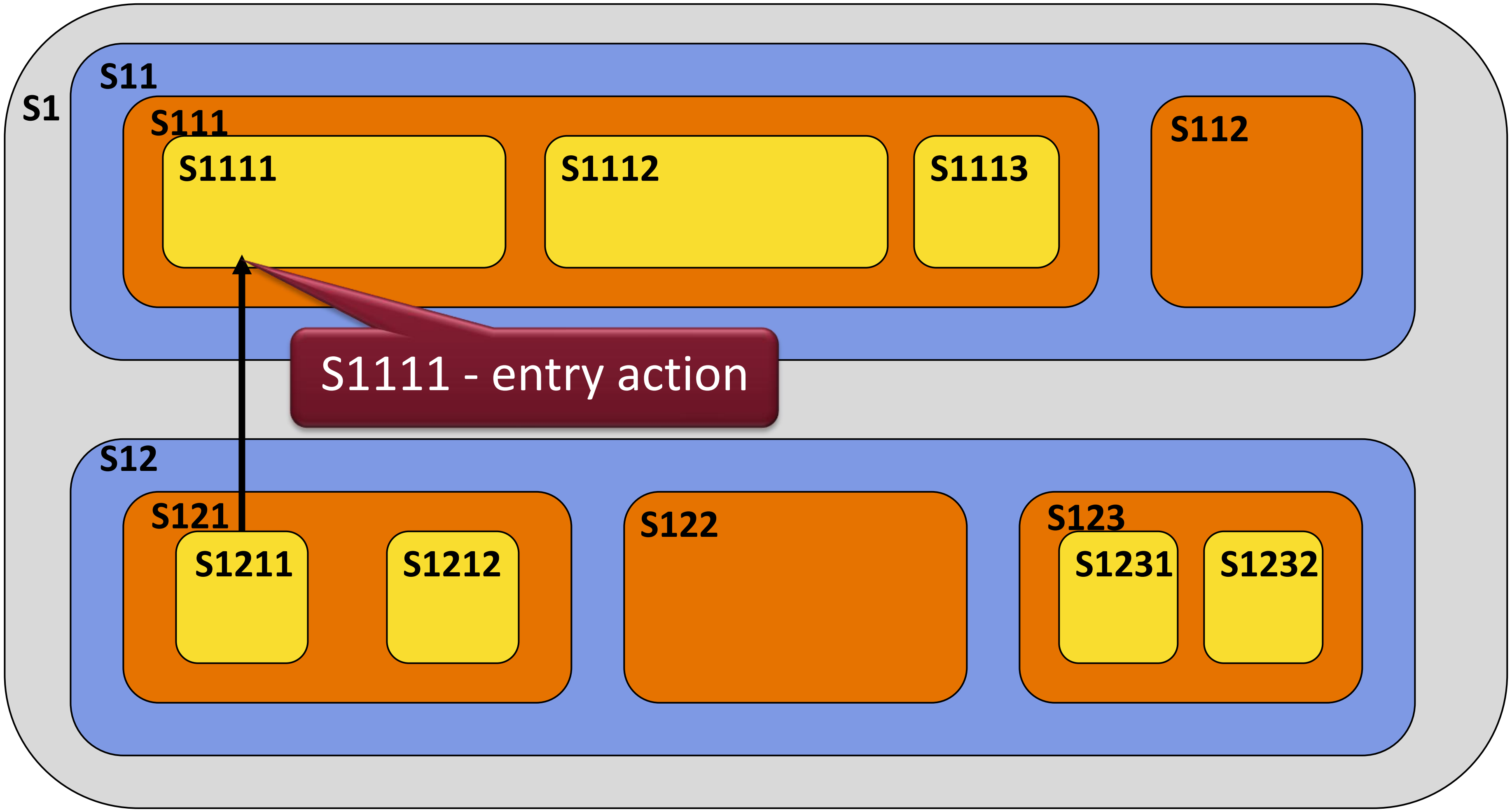


State transition example



S111 – entry action

State transition example



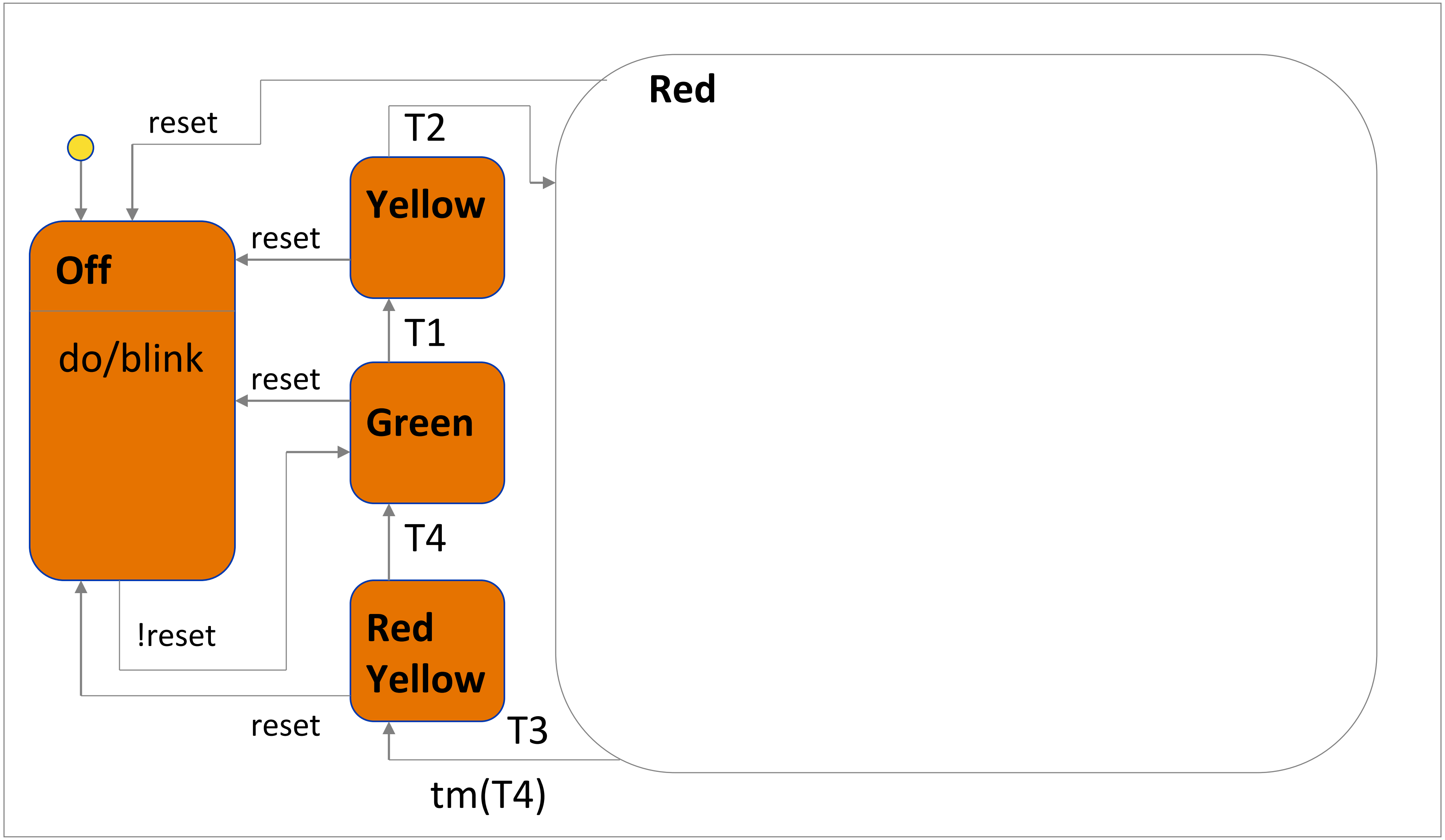
Summary

- Effective technique to model certain dynamic systems
- Hierarchic refinement allows iterative development
- Already used in many application domain
    - Avionics, automotive, control, etc.

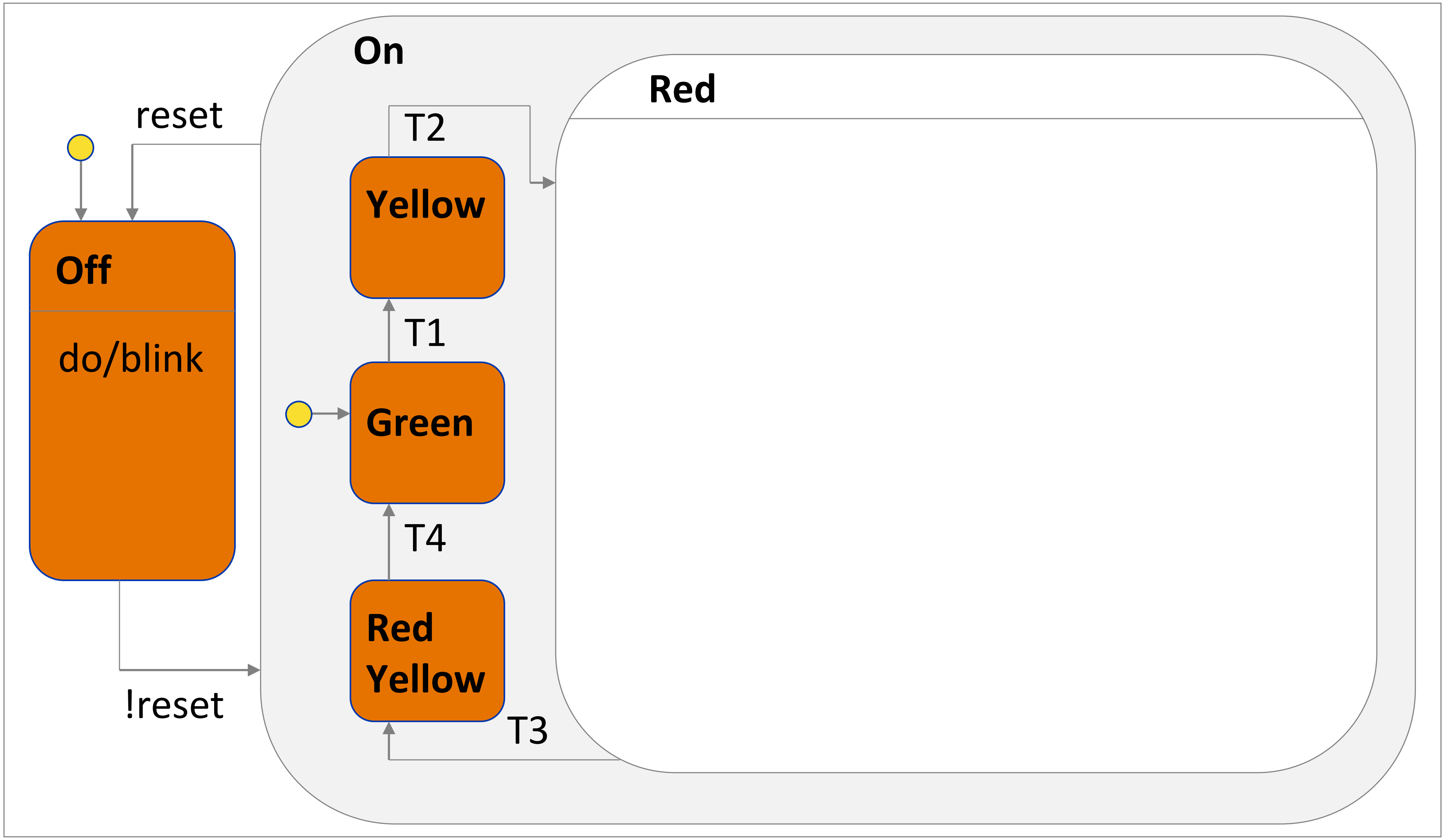
Complex Example

- Traffic light for an intersection with a prioritized road
  - Off: (blinking yellow)
  - On: green for the priority road
  - Green, yellow, red etc. Different timerange (timer)
  - 3 waiting vehicle on priority road: green light despite the timer's ticks
  - Automatically take photos of vehicles crossing the priority road on red light. Manual on/off for this feature.

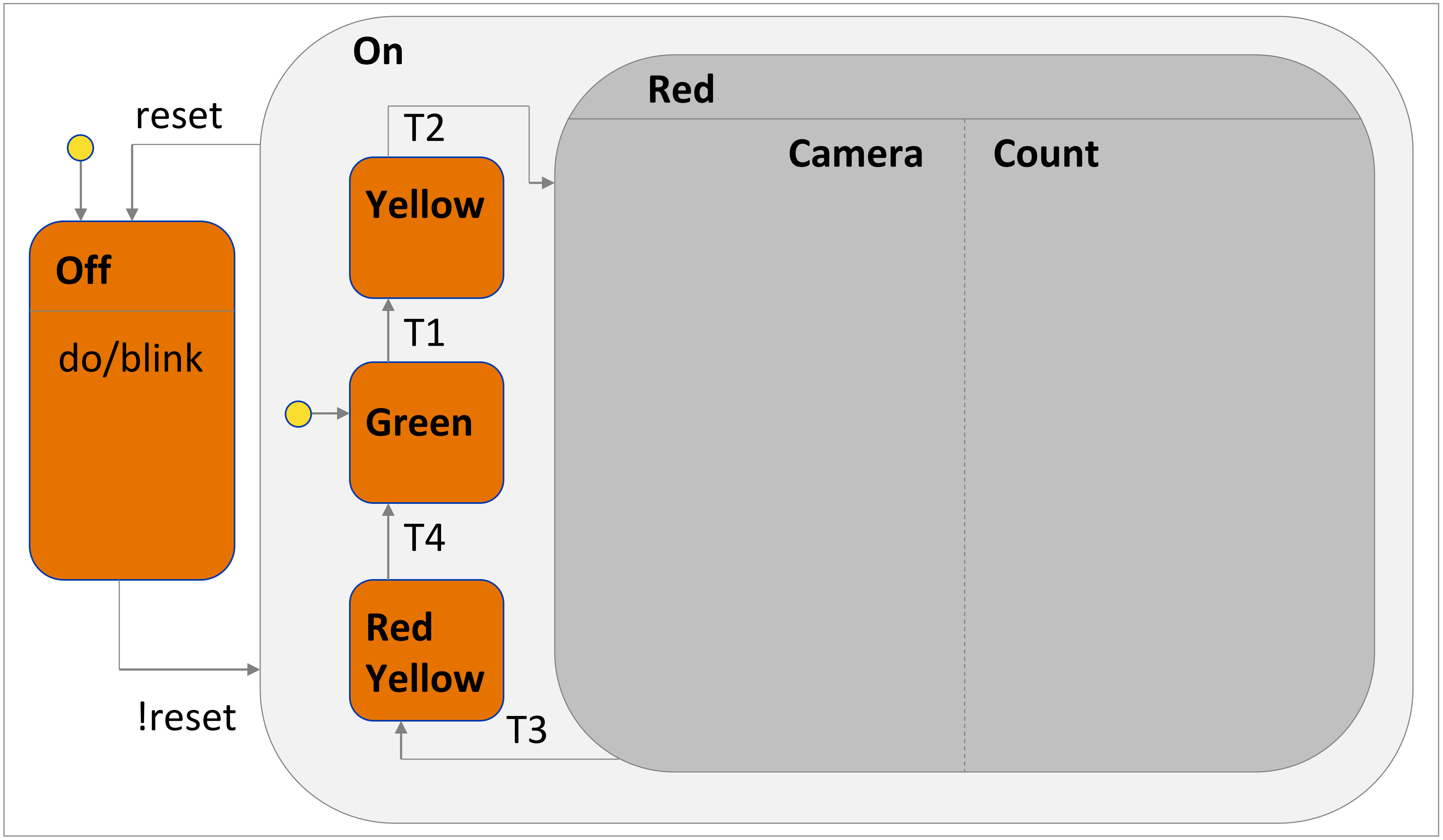
1. Basic state machines



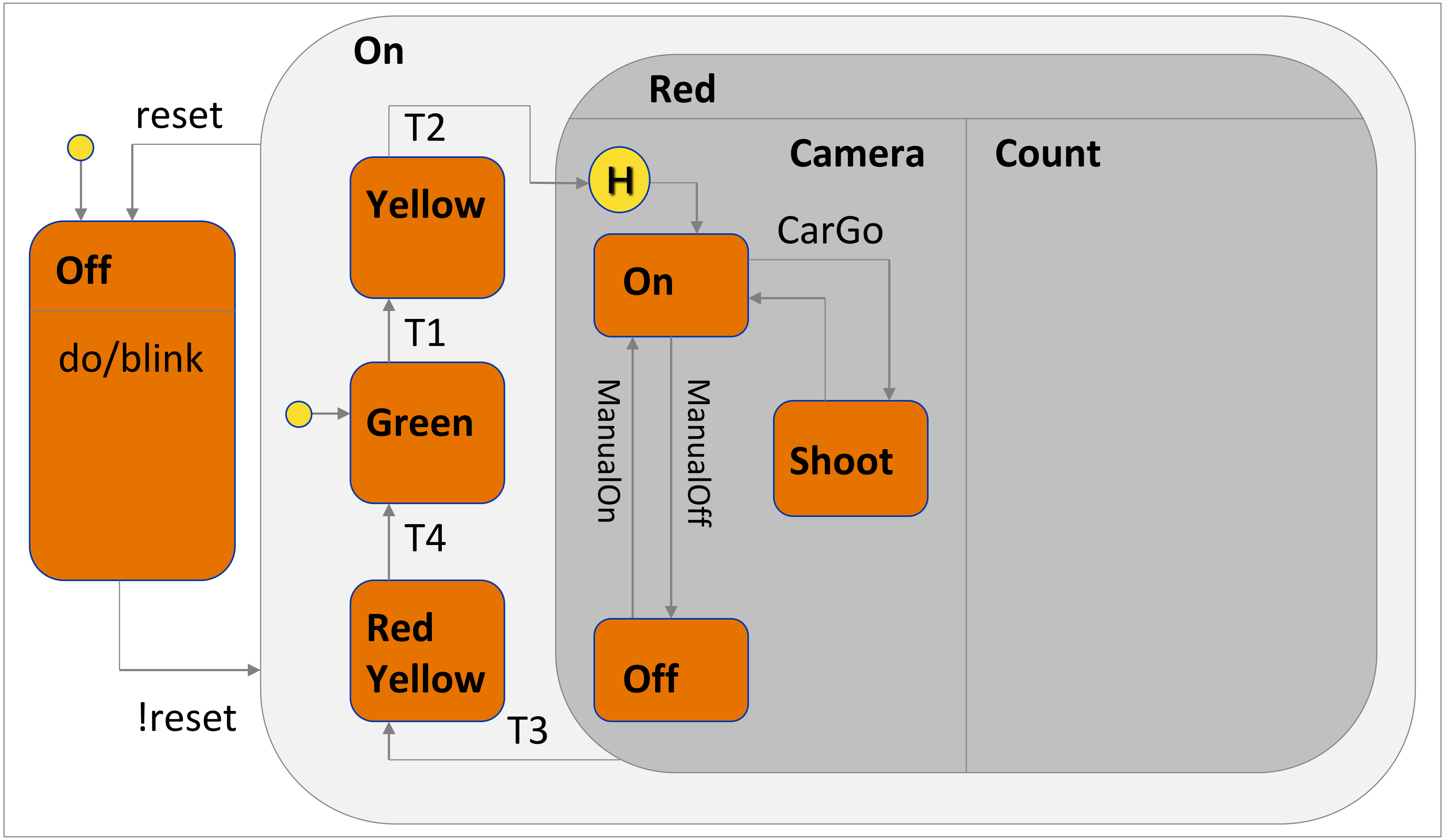
2. Hierarchy


On
Red
reset
T2
Yellow
Off
do/blink
T1
Green
T4
Red Yellow
!reset
T3

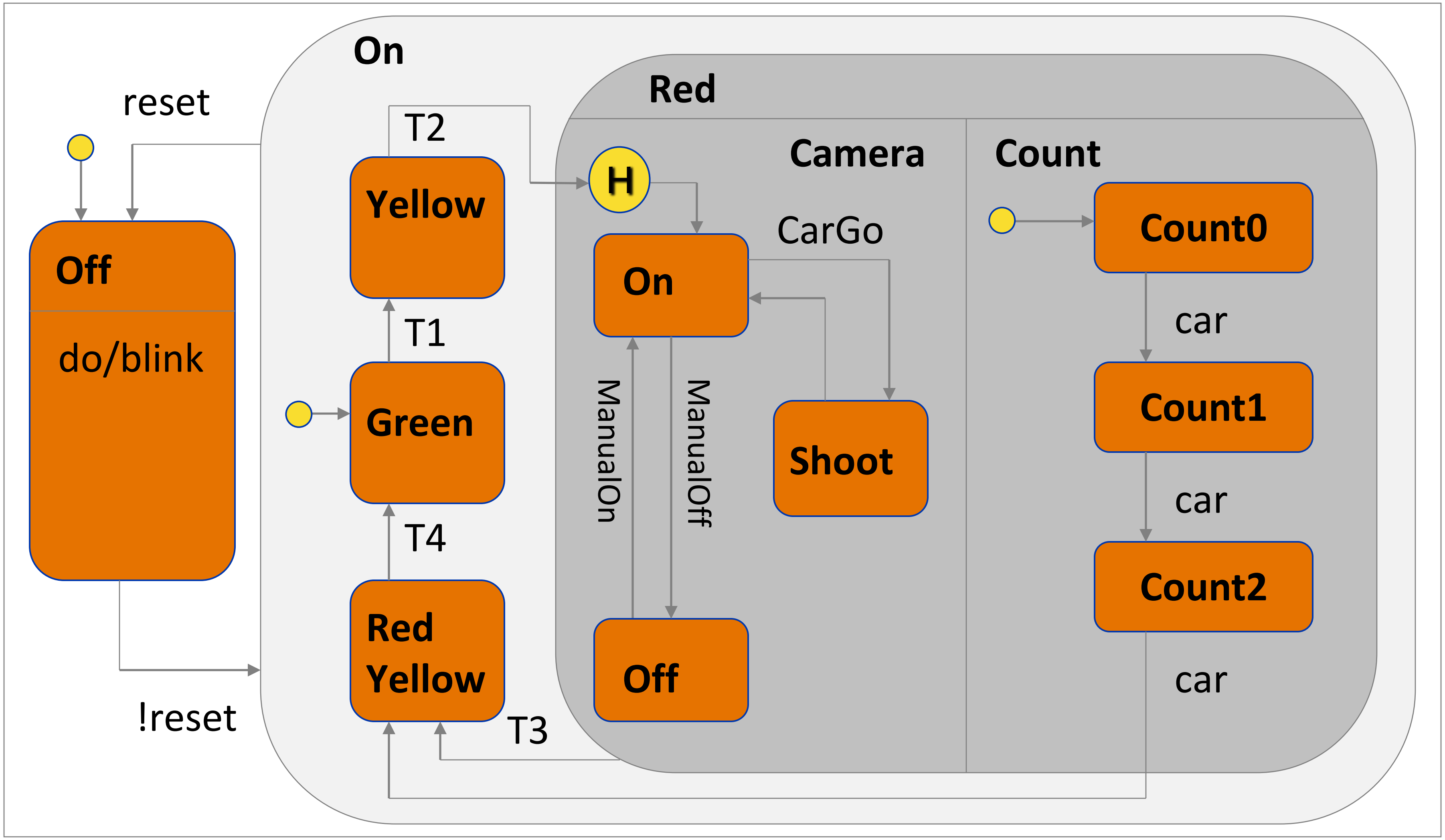
3. Concurrent states



4. History States



Complete System



On
reset
Off
do/blink
!reset
T2
Yellow
T1
Green
T4
Red
Yellow
T3
Red
H
Camera
CarGo
On
Shoot
ManualOn
ManualOff
Off
Count
Count0
car
Count1
car
Count2
car