# Statecharts and OCL

Ákos Horváth and Dániel Varró
With Contributions from
István Majzik and Gergely Pintér

Model Driven Software Development
Lecture 4

# Dynamic Modeling

## Statecharts

# Statecharts

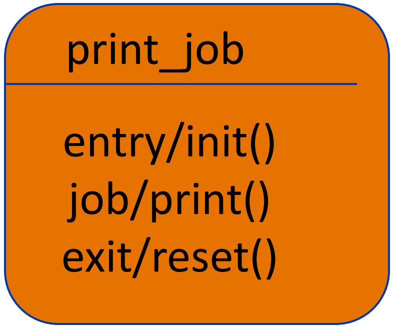
- Describes the states and state transitions of the system, of a subsystem, or of one specific object.
  - hierarchical and concurrent systems
- States
  - Concrete state:
    - Combination of possible values of attributes
    - Can be infinite
  - Abstract states: (like in Statecharts)
    - Predicates 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

# Statechart - introduction

- For defining reactive behavior of objects
  - Responds to events:
    state transitions and actions
  - Traditional approach: state machine
- Statechart: extension to state machine
  - State hierarchy: refinement of states
  - Concurent behavior: parallel threads
  - Memory: last active state configuration

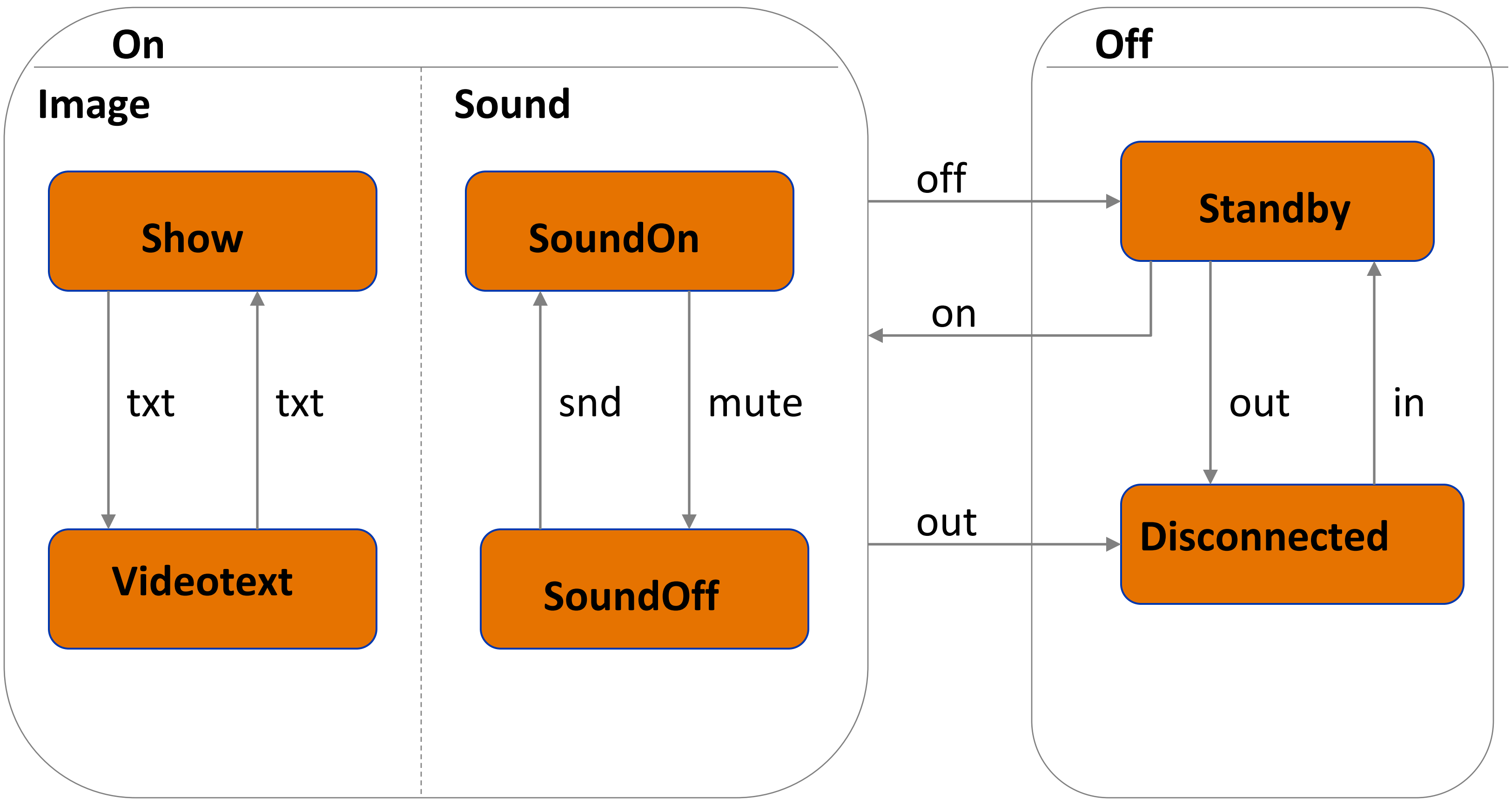
# States I.

- Attributes:
  - entry action
  - exit action
  - static reaction




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

# Example for state refinement: TV

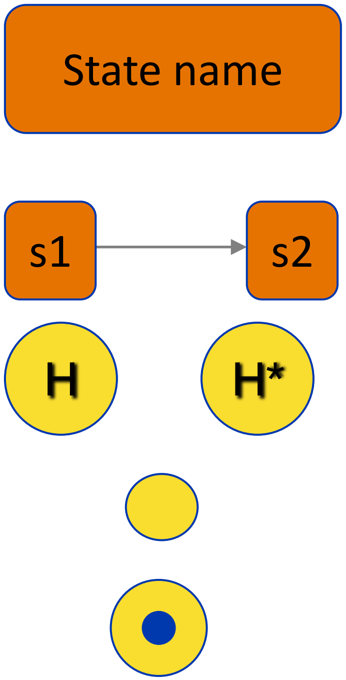


# State II.

- History state
  - Stores the last active state configuration
  - Input transition: it sets the object to the saved state configuration objektum
  - Output transition: defines the default state, if there were no active state since
- Inital state: becomes active when entered to the region
  - One in each OR refinement
  - One in each AND region
- Végállapot jelzés: állapotgép terminálás

# Statechart elements

- State
- (Transition)
- History state
- Initial State
- Final State



# Transition I.

- Defining state changes
- Syntax:

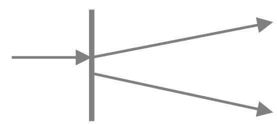
  **trigger [guard] / action**

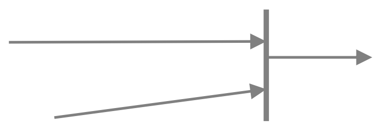
  - trigger: event, triggered operation or time-out
  - guard: transition condition
    - Logic formula over the attributes of the objects and events
    - referring to a state: IS_IN(state) macro
    - Without trigger: if becomes true the transition is active
  - action: operations $\Rightarrow$ action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

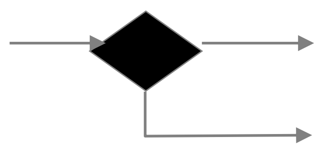
# Transition II.

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

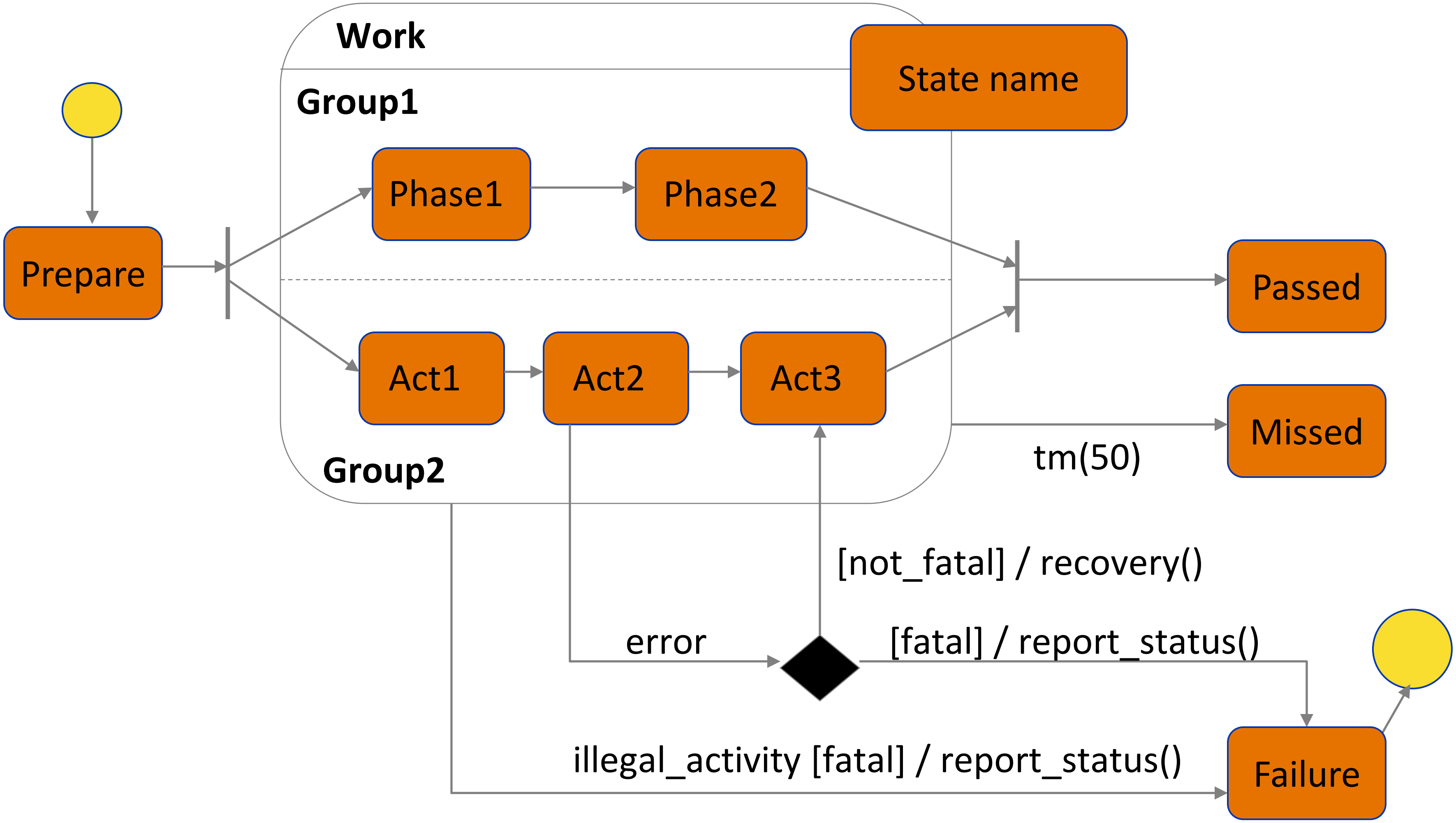
    e.g., tm(50), based on system time
- Complex transitions
  - <u>Fork</u>

    
  - <u>Join</u>

    
  - <u>Condition</u>

    
- Transitions between different hierarchy levels

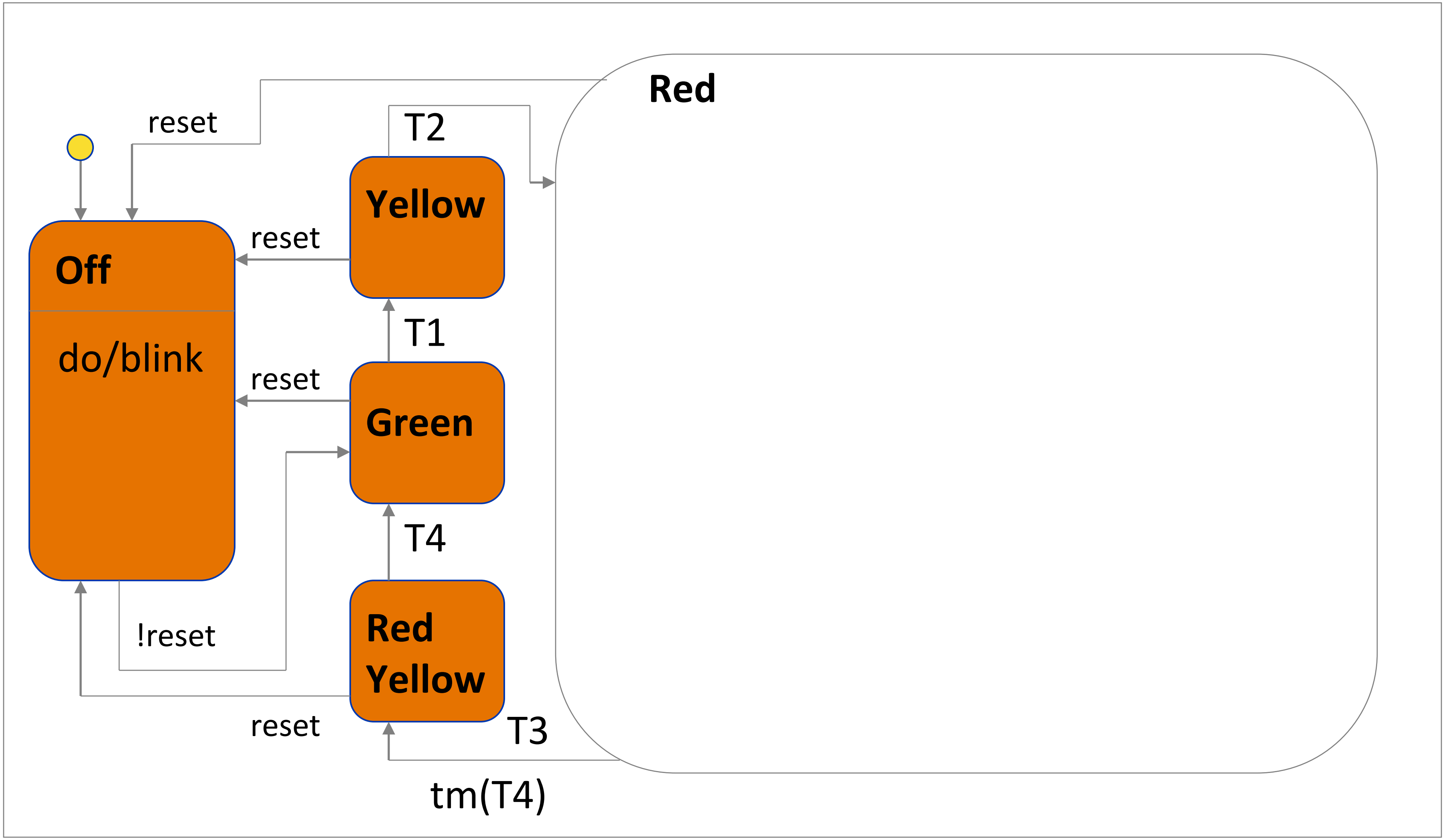
# Transition example



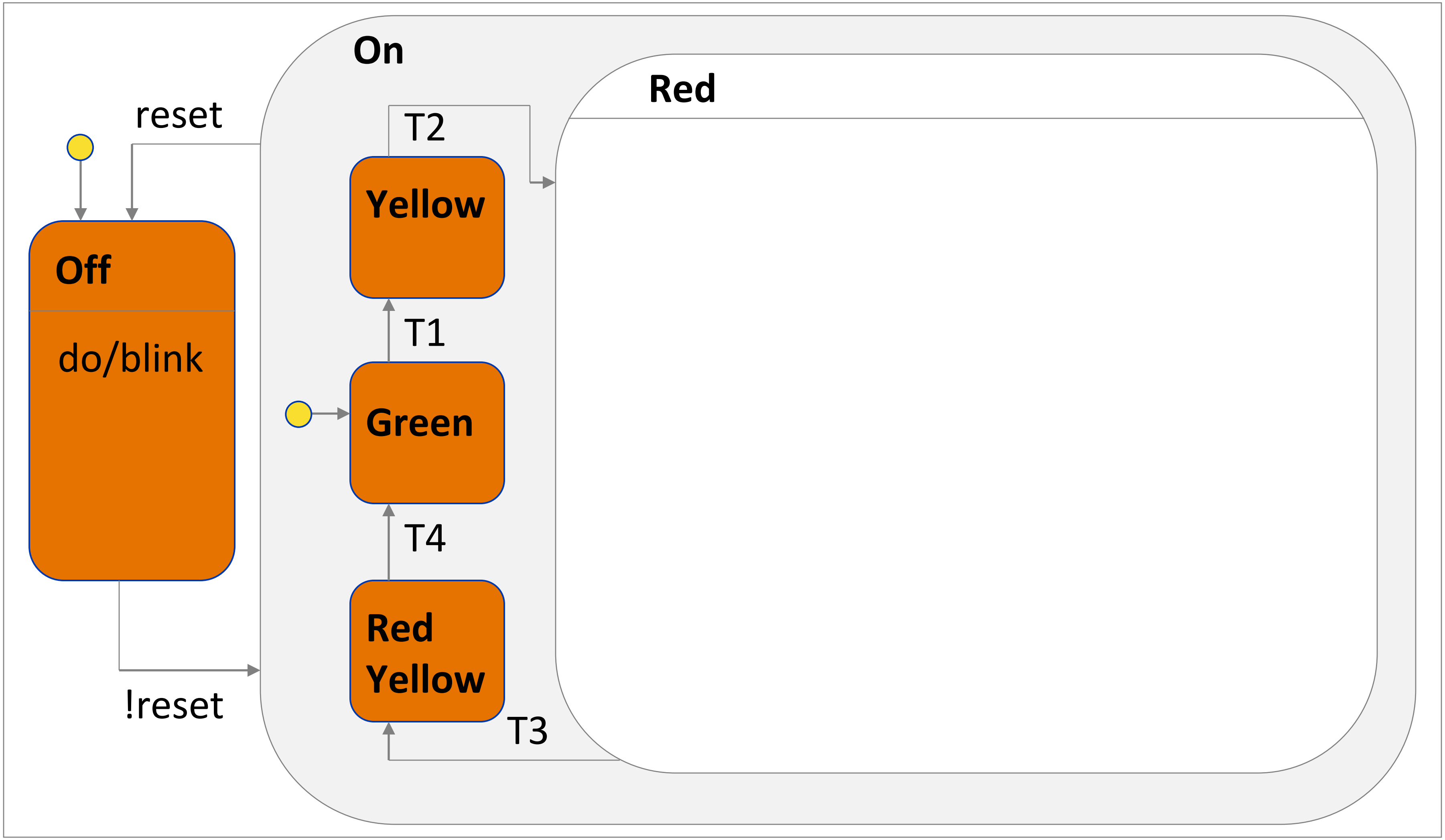
# Complex Example

- 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 piority road on red light. Manual on/off for this feature.

# 1. Basic state machines

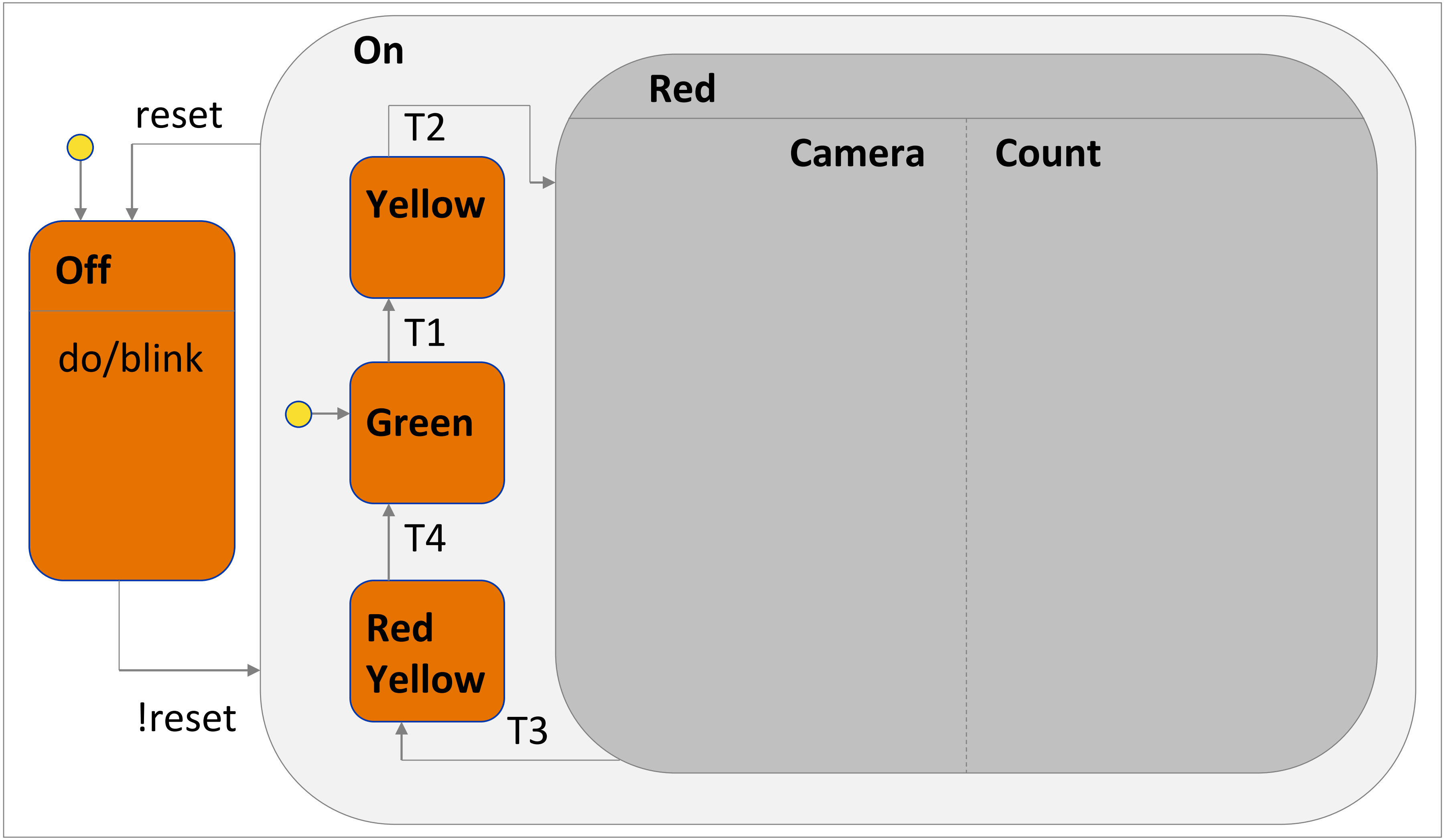


# 2. Hierarchy

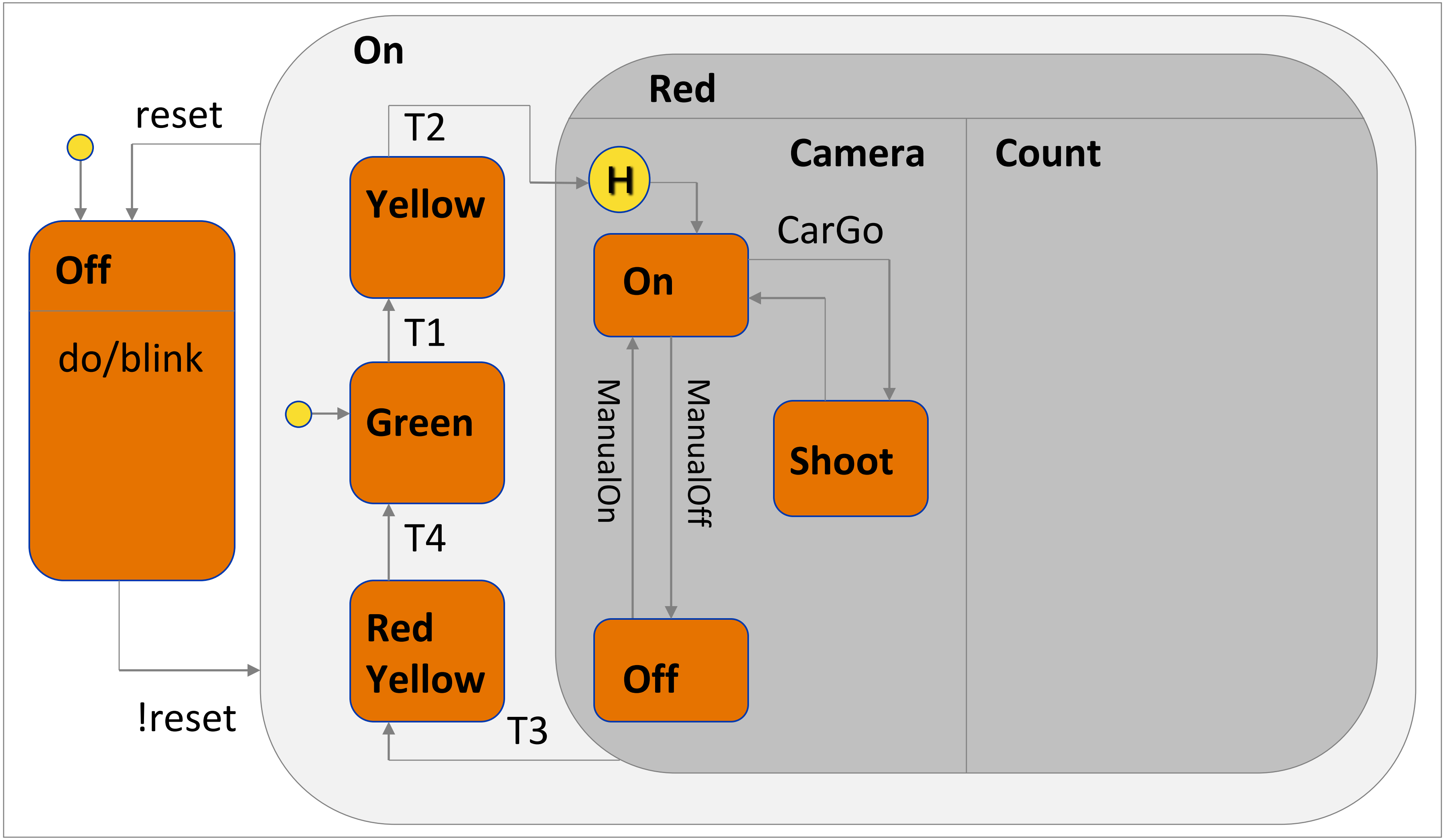


On
Red
reset
T2
Yellow
Off
do/blink
T1
Green
T4
Red
Yellow
!reset
T3

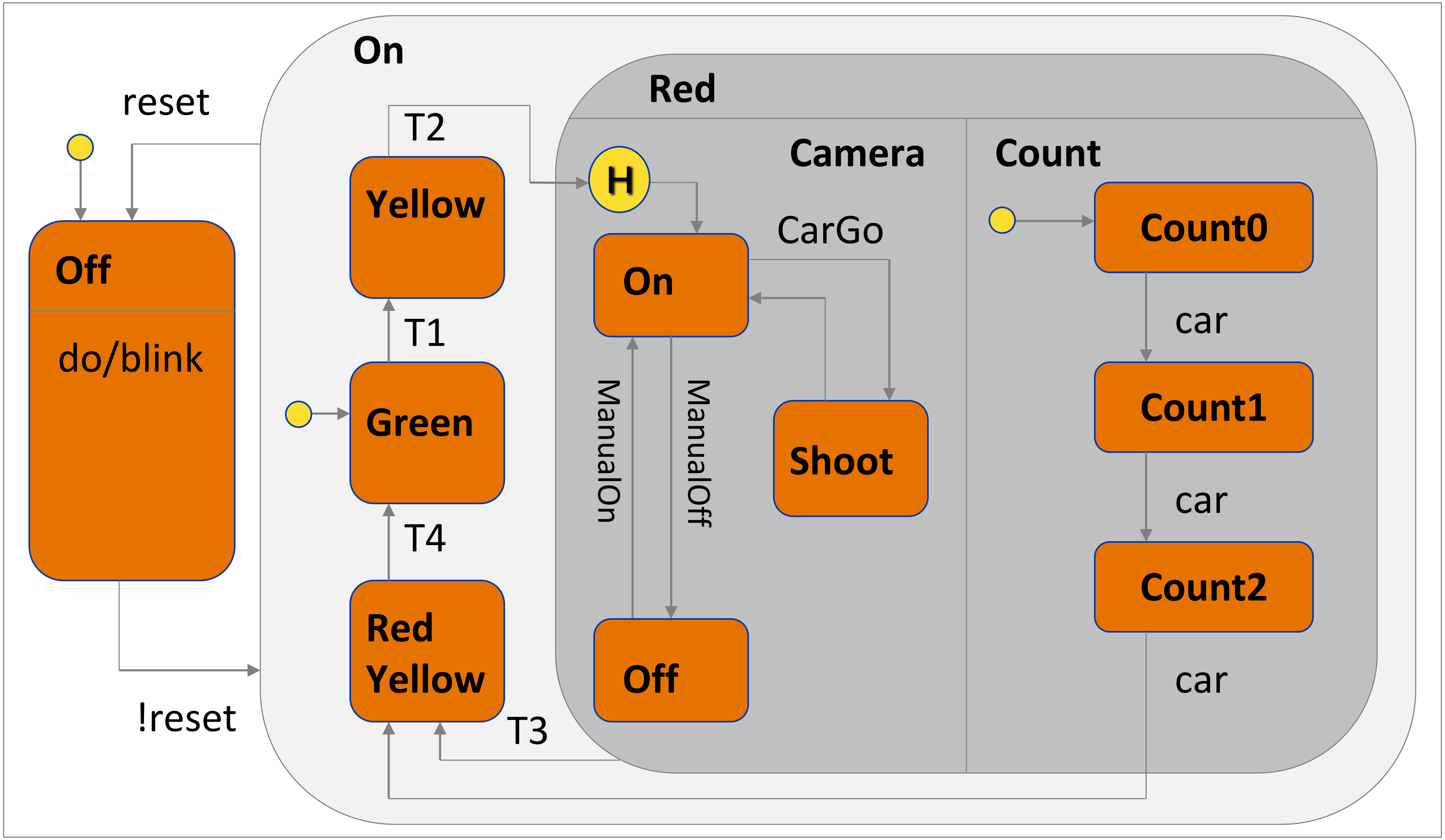
# 3. Concurrent states



# 4. History States



# Complete System



# Semantics: How does it work?

- Basics:
  - Hierarchical state machine (state chart)
  - Event queue + scheduler
- Semantics defines:
  Behavior in case an event occurs
  → 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 statechart 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?

# 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
  
  → 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:
    - Random!

# Steps of event processing IV.

- Selected transitions fire:

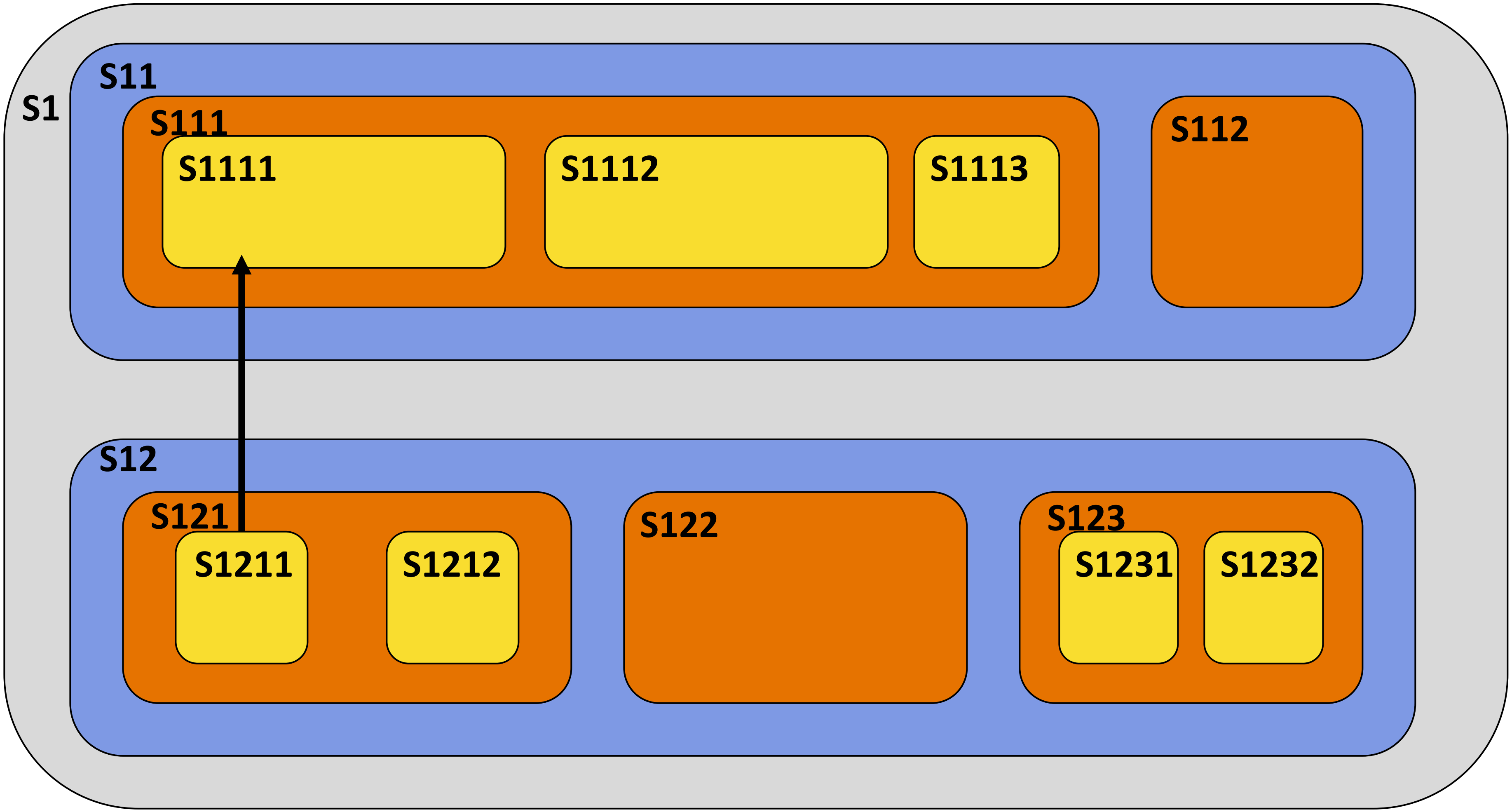
  in random order

- 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

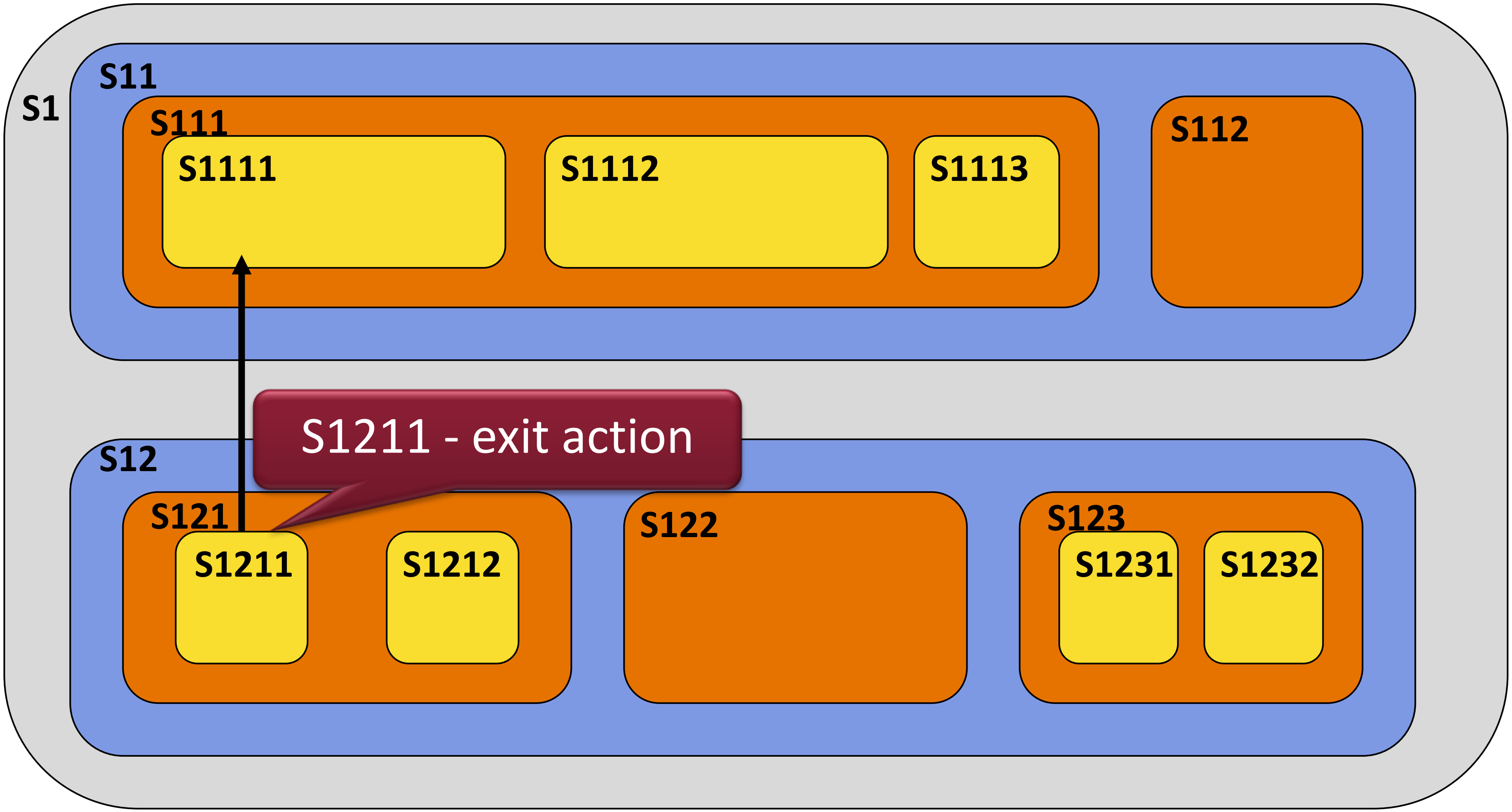
# Steps of event processing V.

- 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 of with initial state
  - History state : the last active state configuration if there is none: the target state of the history state

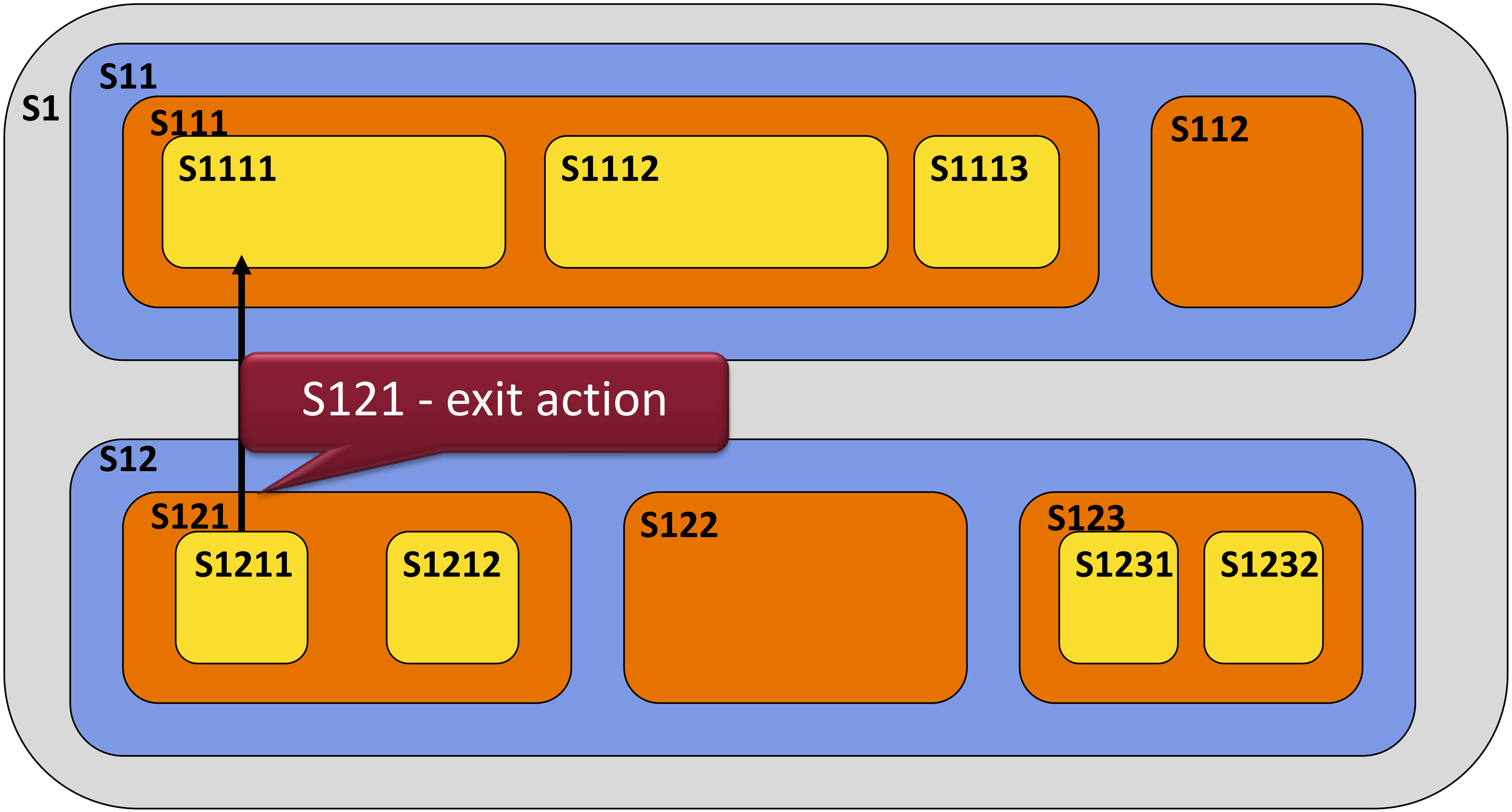
# State transition example



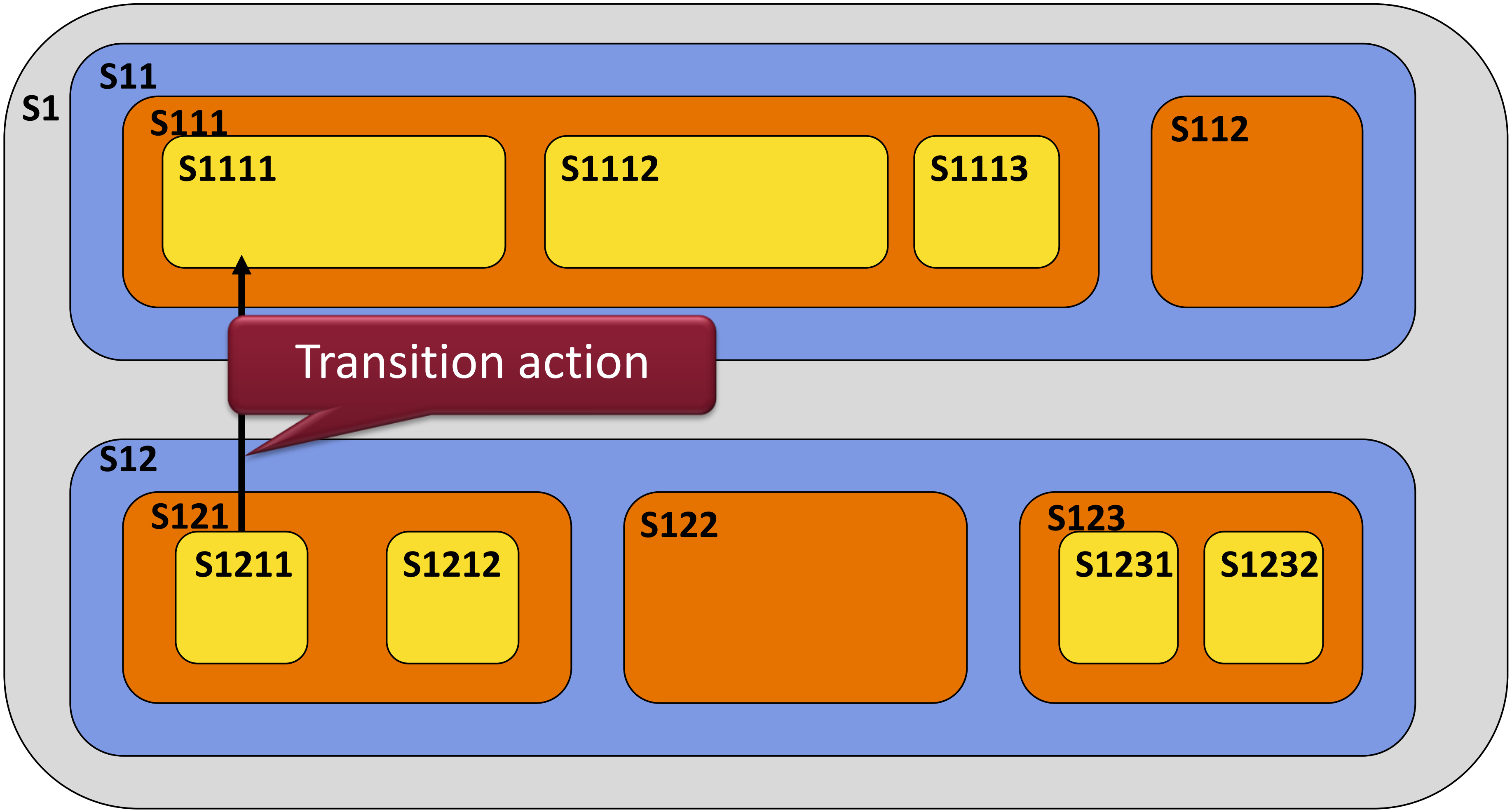
# State transition example



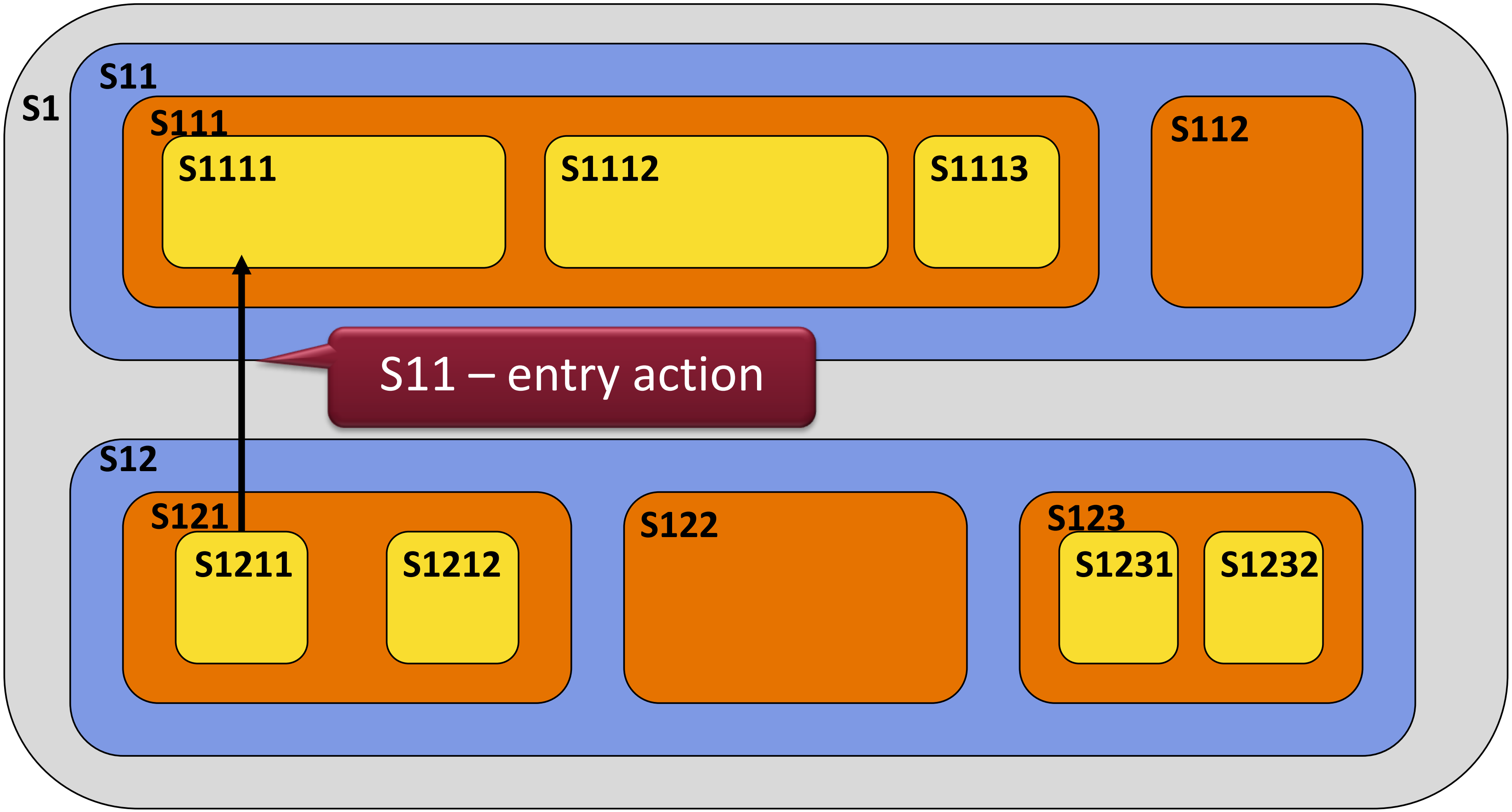
# State transition example



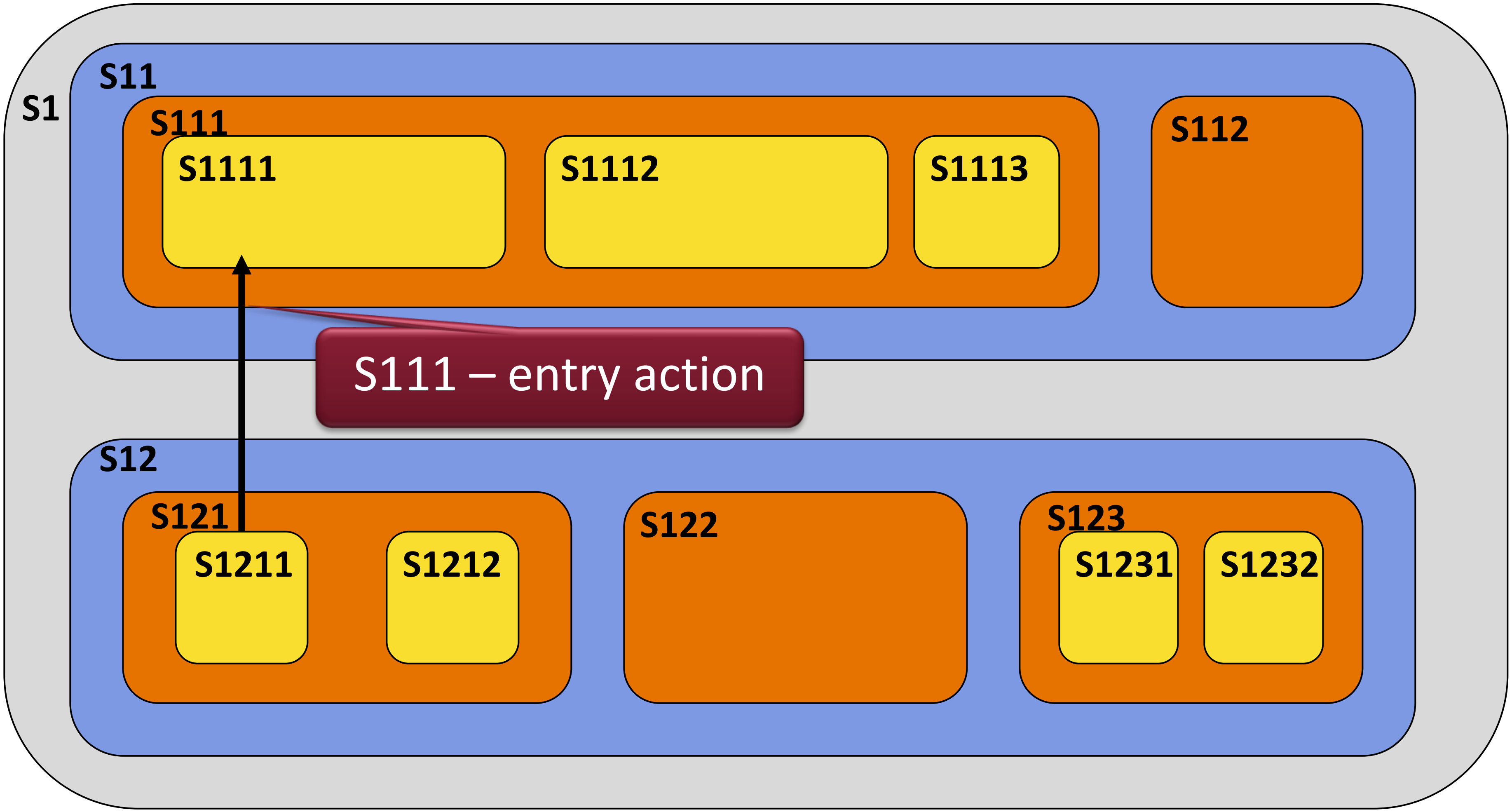
# State transition example



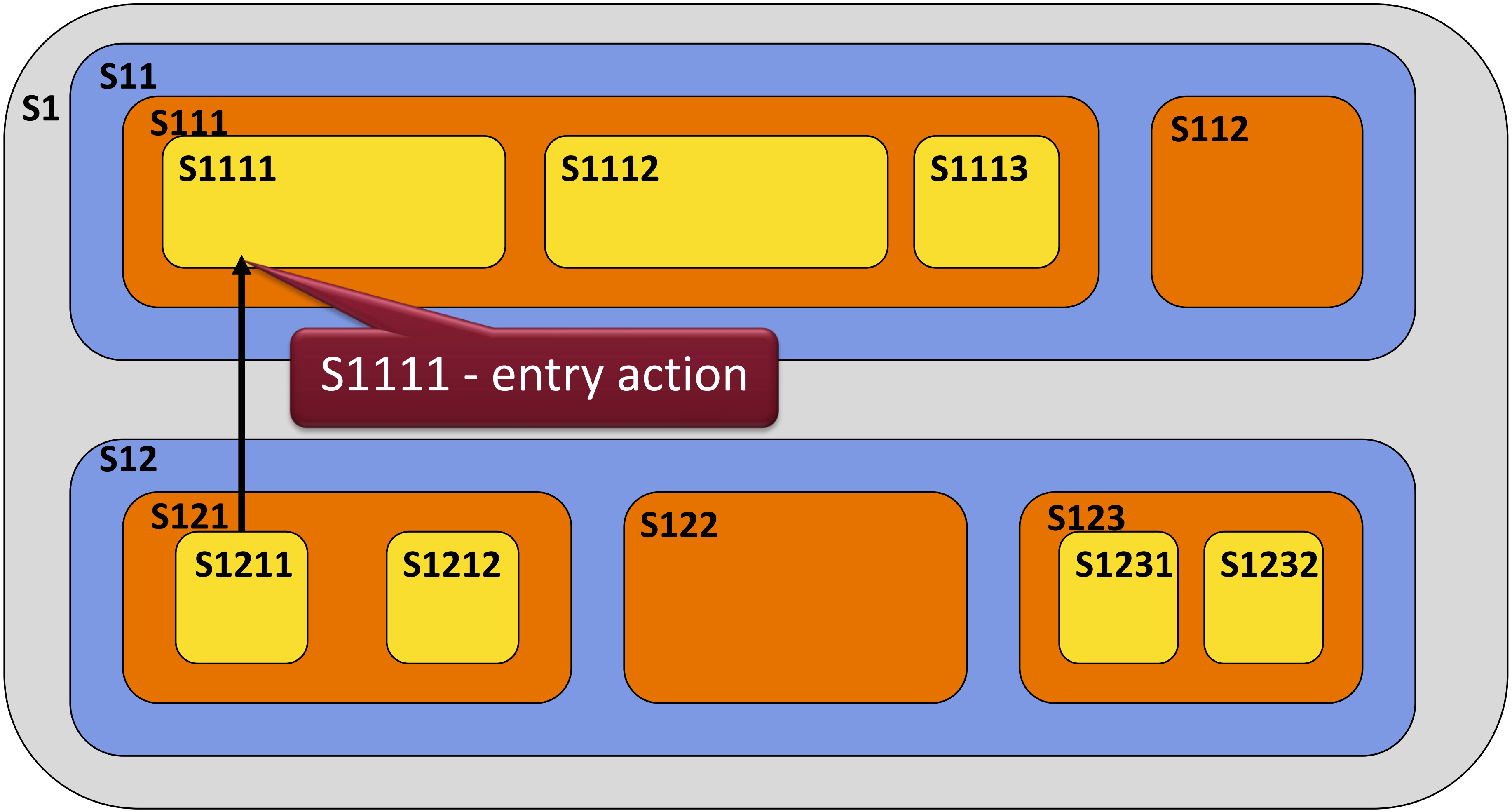
# State transition example



# State transition example



# State transition example



# Summary

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