CHRISTIAN HUEMER
MARION SCHOLZ

Object-Oriented Modeling with UML
Part III – State Machine Diagram

State Machine Diagram

The State Machine Diagram

Christian Huemer und Marion Scholz

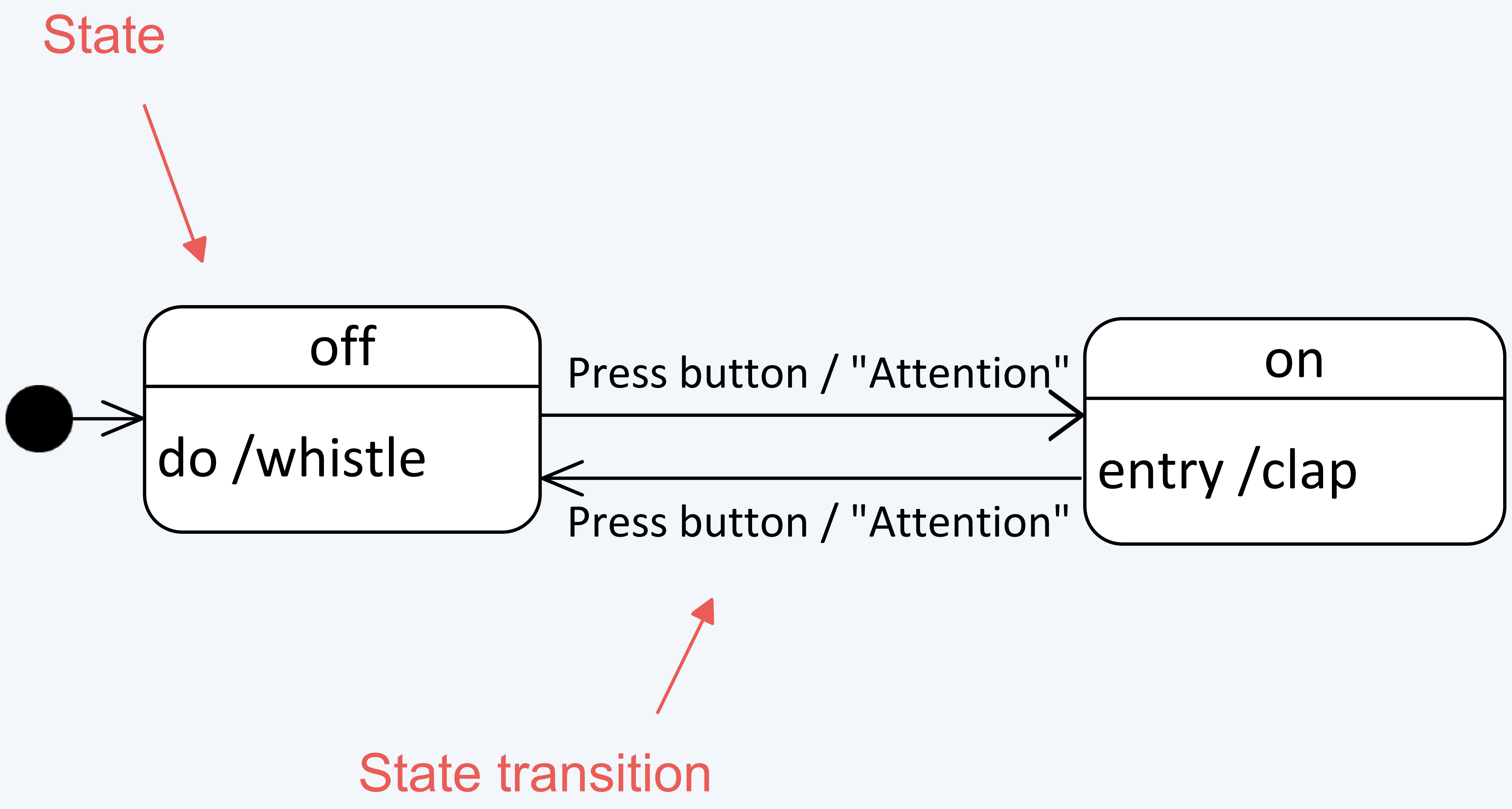
Presented by Nicholas Bzowski

Introduction

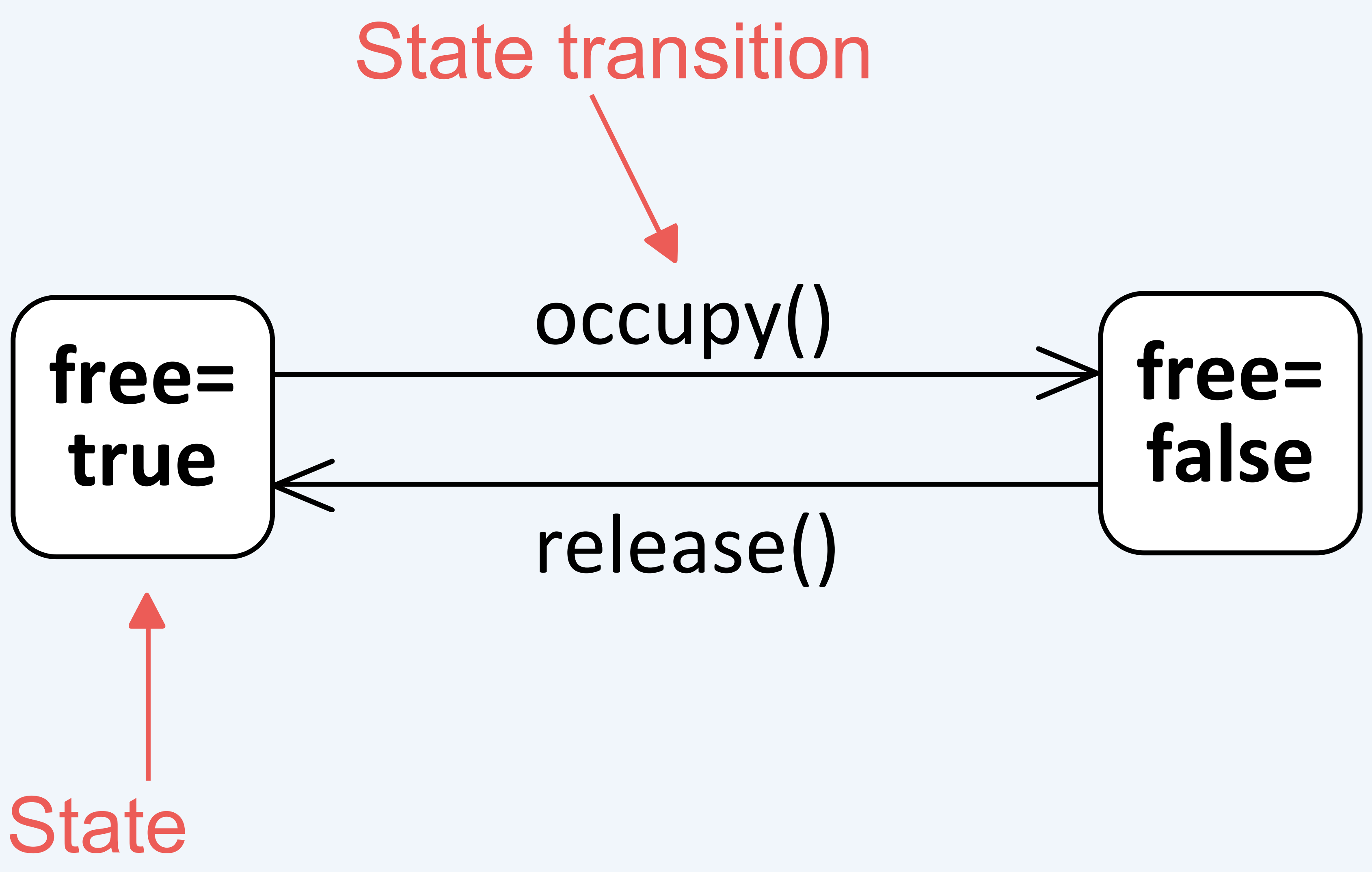
- A state diagram describes the possible **sequences of states** of a **model element**, typically for objects of a certain class
  - During its **lifecycle** (from creation to deletion)
  - During the **execution** of an **operation** or **interaction**

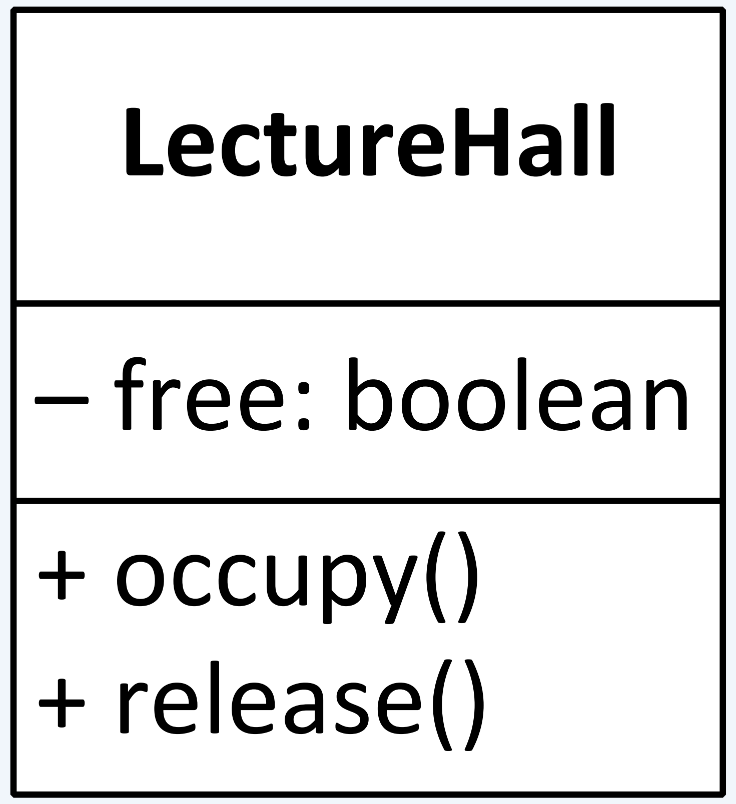
- The following are modeled
  - The **states** in which the objects of a class can be
  - The possible **state transitions** from one state to another
  - The **events** that trigger transitions
  - **Activities** that are executed in states or in the process of transitions

Example: Lamp



Example: Lecture Hall

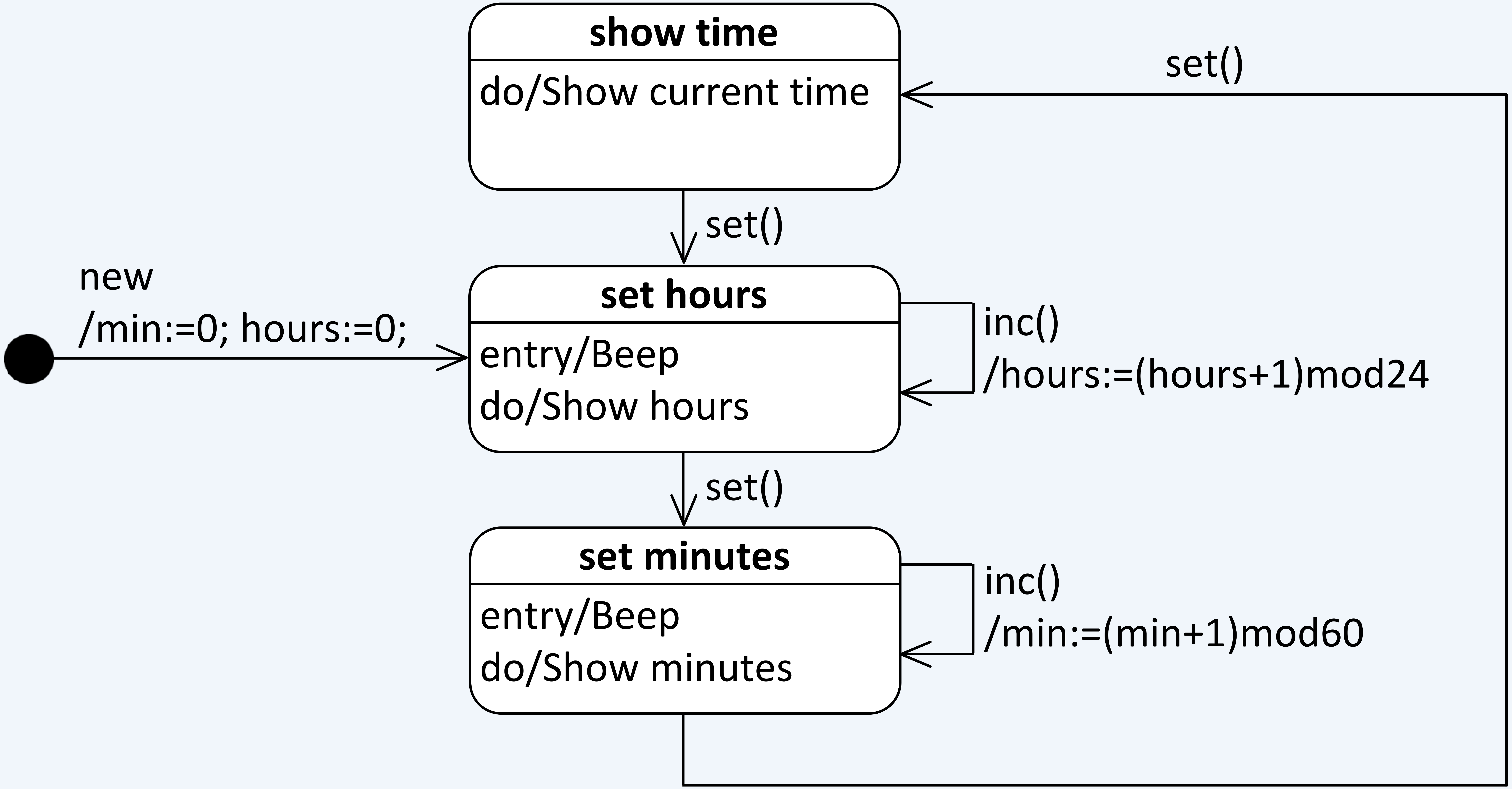


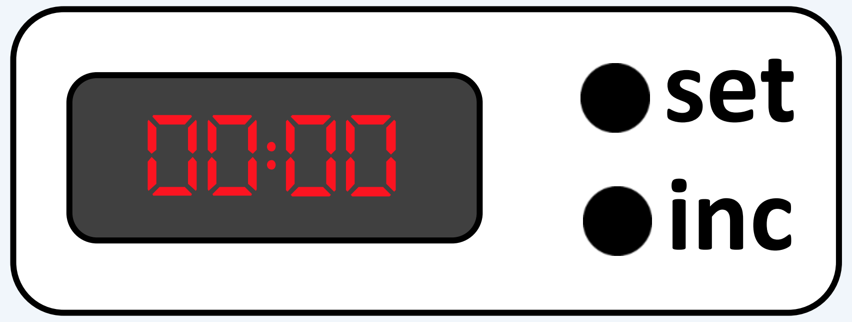


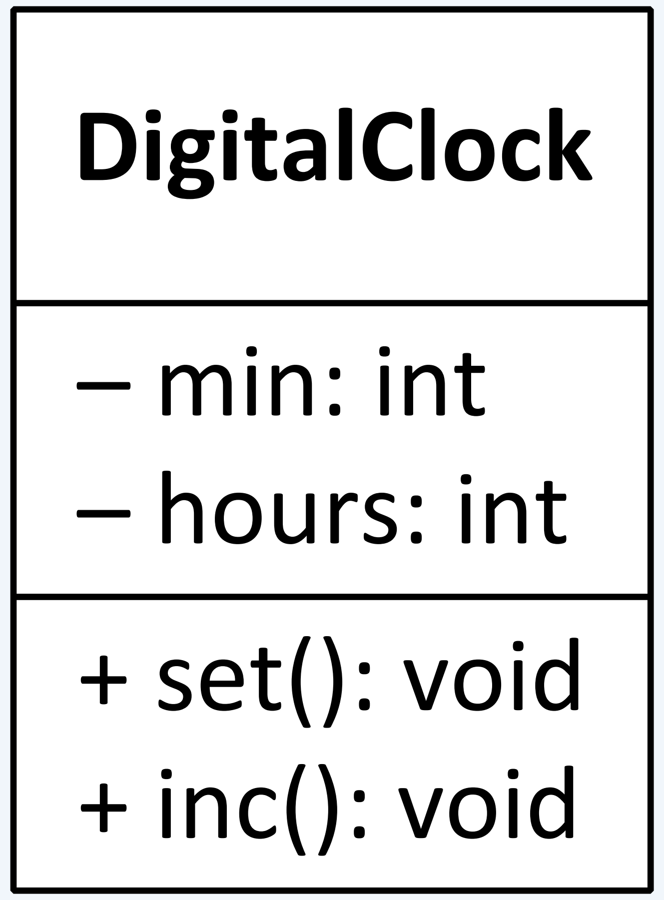

```
class LectureHall {
  private boolean free;

  public void occupy() {
     free=false;
  }
  public void release() {
     free=true;
  }
}
```

Example: Digital Clock

- The states that a digital clock can be in when setting the clock are modeled.



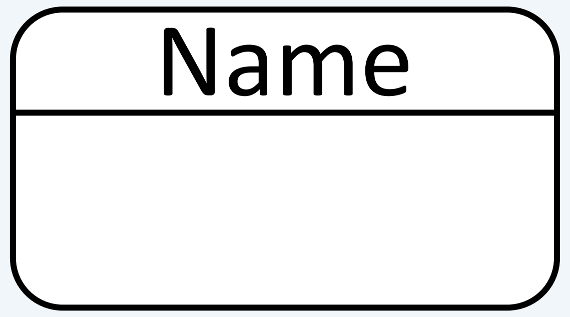




State Machine Diagram
The State

Christian Huemer und Marion Scholz
Presented by Nicholas Bzowski

State



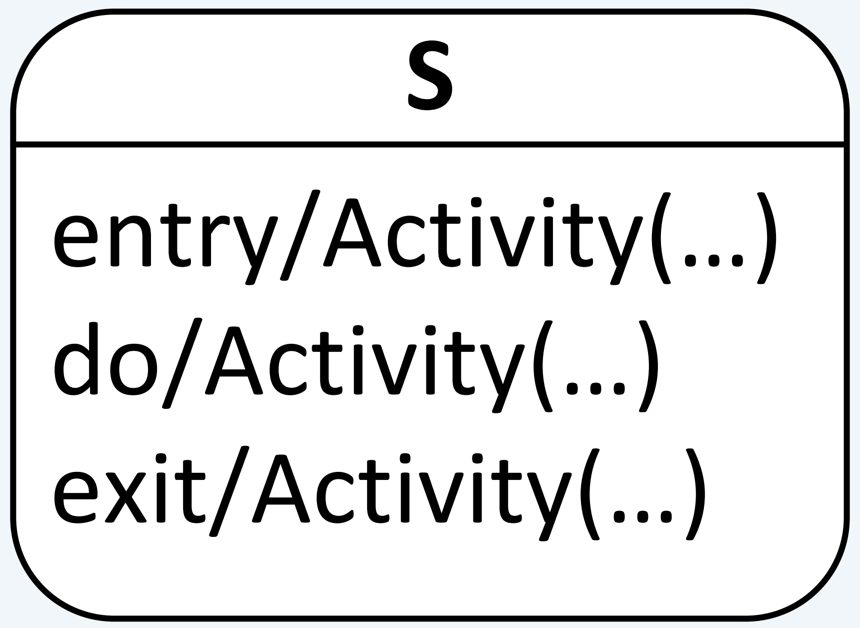
- State
  - System can be in a permanent state
  - “Real" state
  - Final state

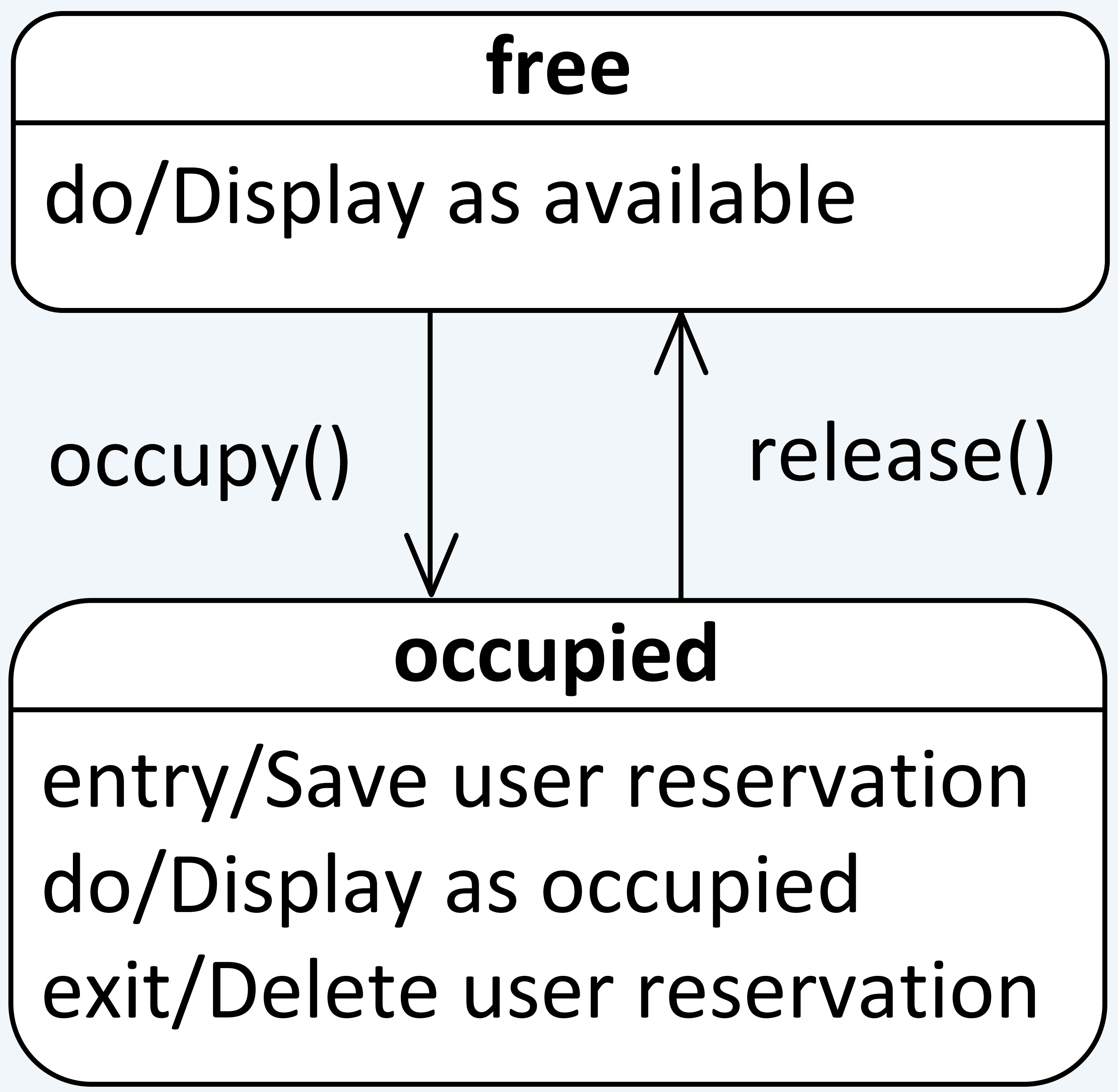
- Pseudostate

  System cannot be permanently in this state
  - Initial state
  - Shallow/deep history state
  - Parallelization nodes & synchronization nodes
  - Terminate node
  - Decision node

Activities within a State

- `entry` / Activity
  - Executed when entering the state
- `exit` / Activity
  - Executed when leaving the state
- `do` / Activity
  - Executed when the system is in the state
  - Parameters are allowed
- `event` / Activity
  Activity handles event within the state
  - Executed when the system is in the state and the event occurs





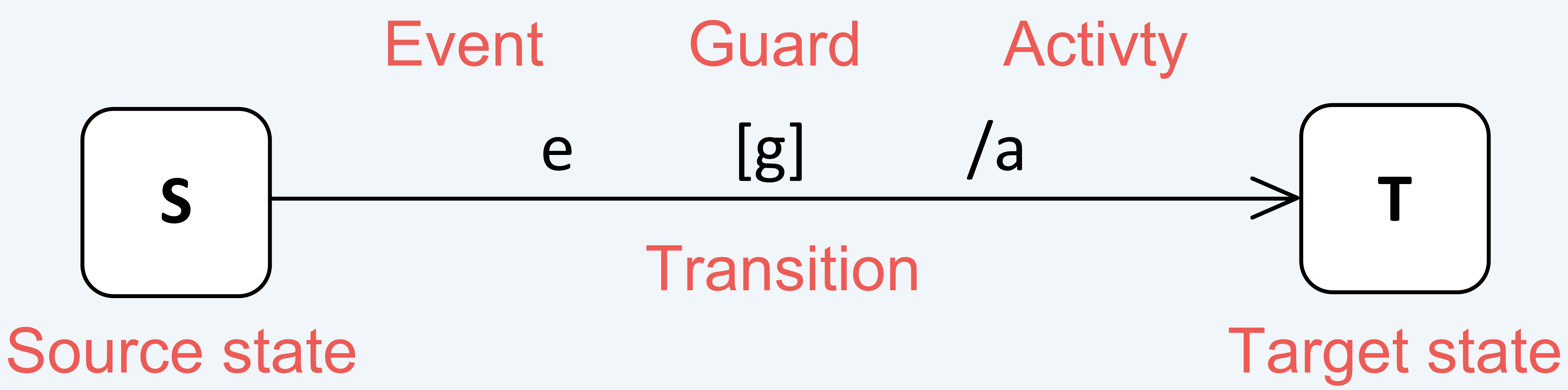
State Machine Diagram
The State Transitions

Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

State Transition (1/2)

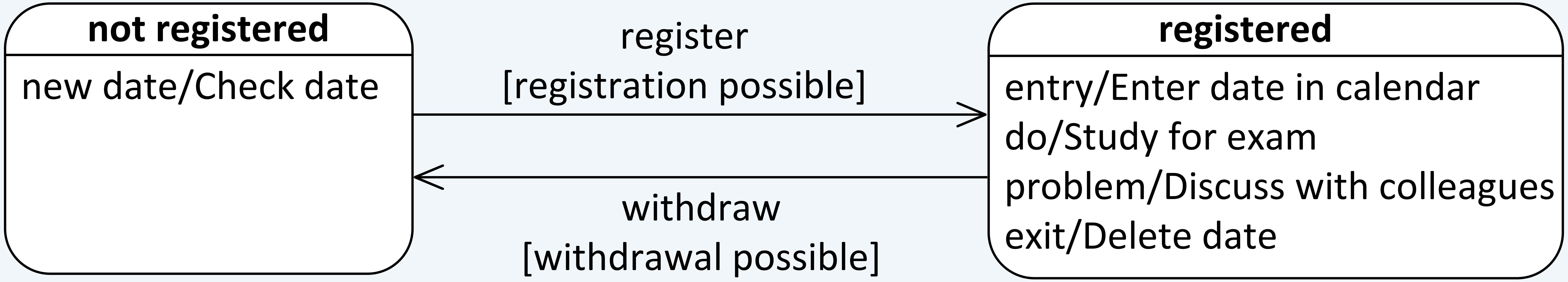
- Transition from the source state to the target state
- Occurs when the event occurs and the condition (if any) is fulfilled



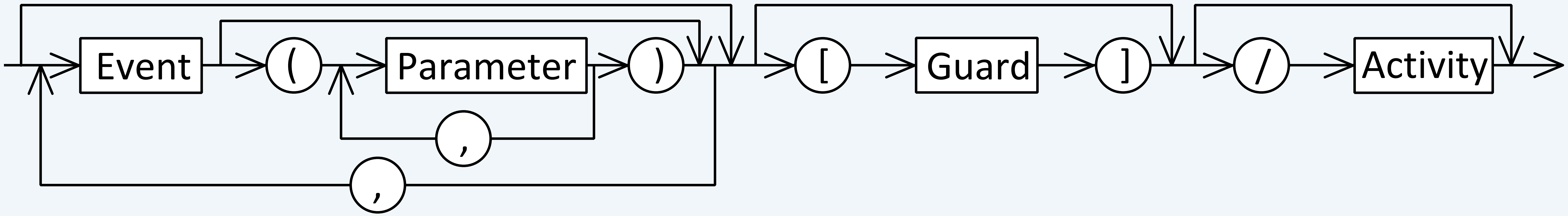
State Transition (2/2)

- Event, Trigger
  - External stimulus
  - Can trigger state transition
- Guard (Condition)
  - Boolean expression
  - Evaluated when the associated event occurs
  - Condition true: Activities in the current state are canceled, exit activities are executed and the state transition takes place
  - Condition false: System remains in the current state, the event is lost
- Activity (Effect)
  - Executed during the transition
  - Can include a number of actions

Example: Registration status for an exam



Syntax of Transitions



- The activity can consist of several actions

- Example:

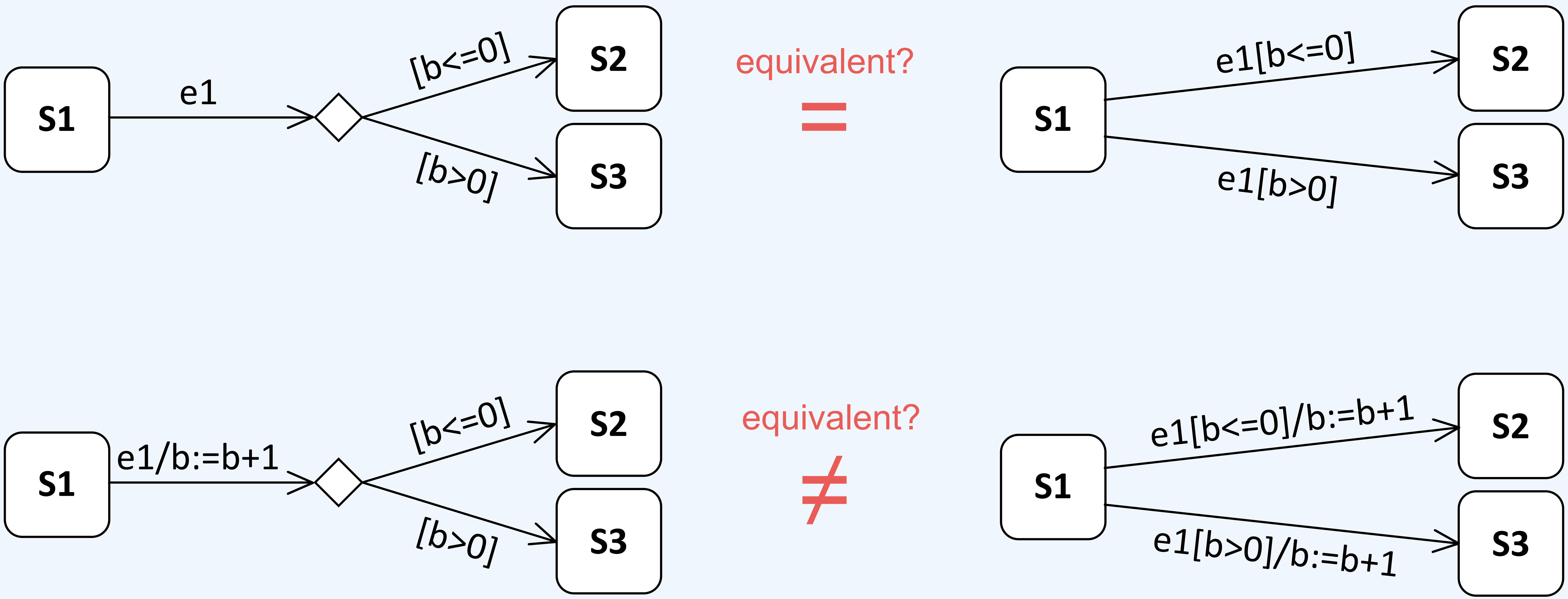
Event   Guard

```
right-mouse-button-down (loc) [loc in window]
/ obj:= pick-obj (loc); send obj.highlight()
```

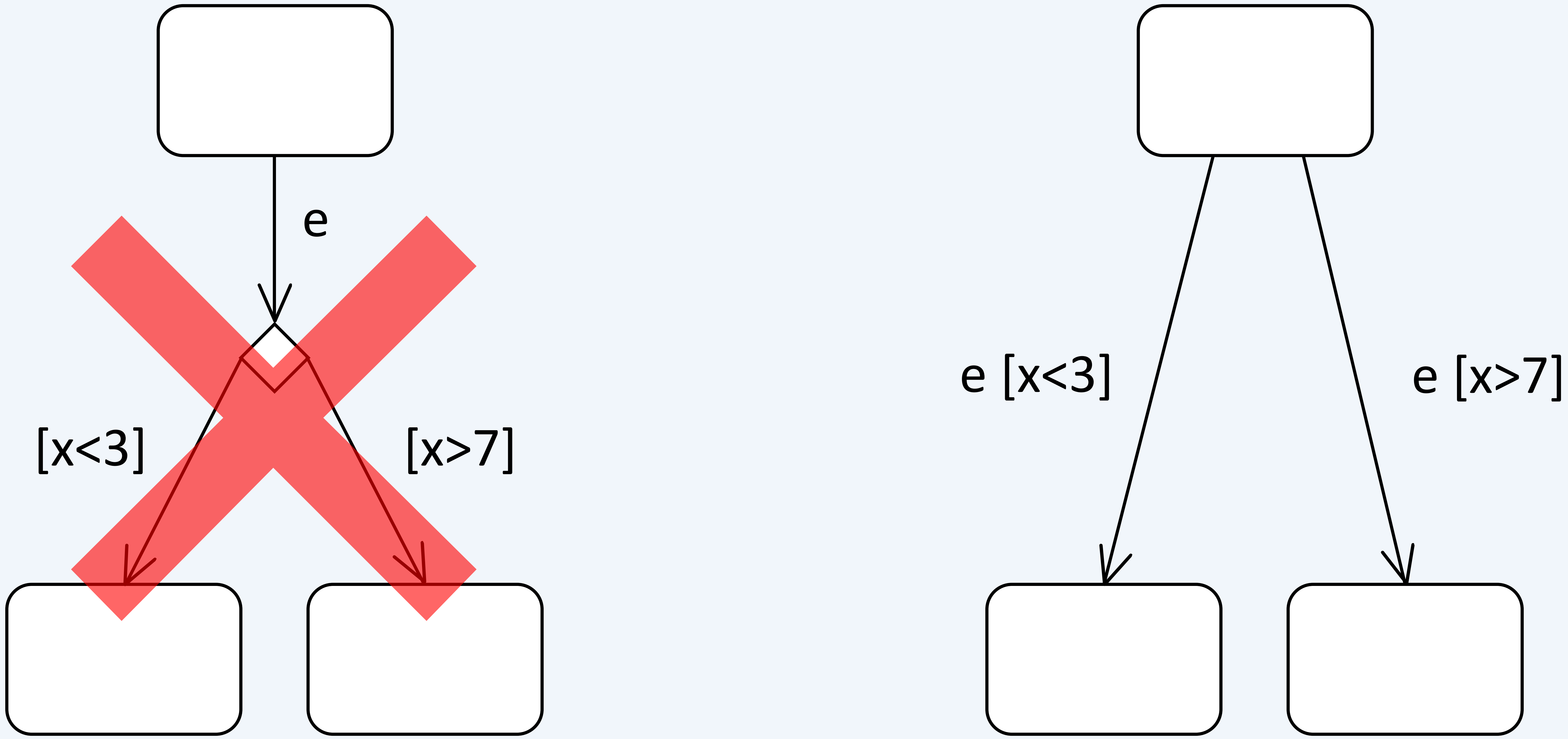
Action 1   Action 2

Modeling with Decision Nodes (1/2)

- Pseudostate
- Can be used to model alternative transitions



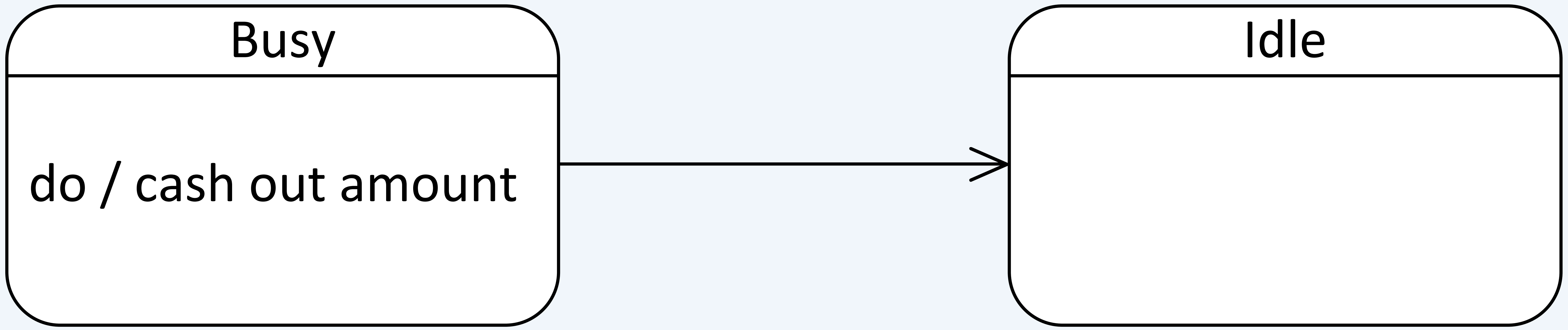
Modeling with Decision Nodes (2/2)



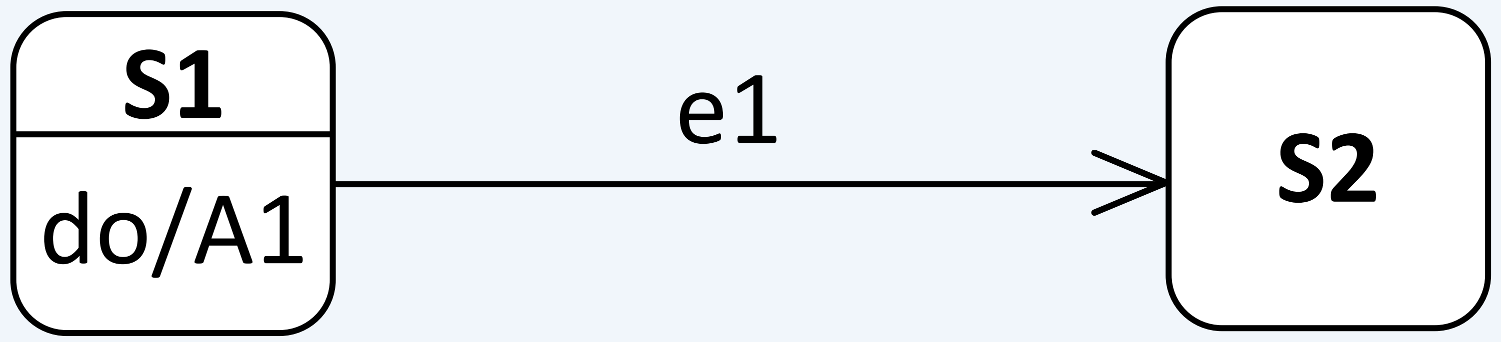
Default Values for State Transitions

- Default values
  - Missing event corresponds to the "Activity is complete" event
  - Missing condition corresponds to the condition `[true]`

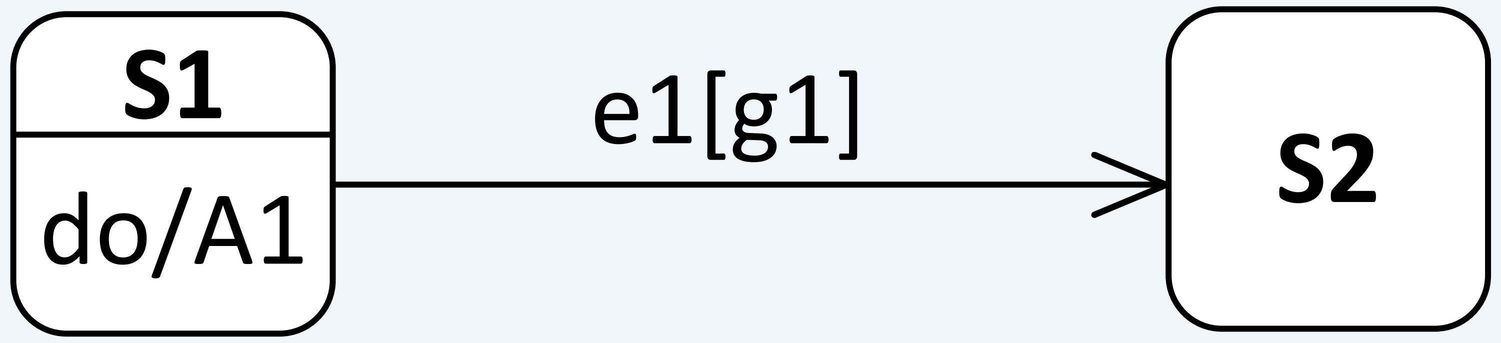
- Example: ATM



Types of Transitions

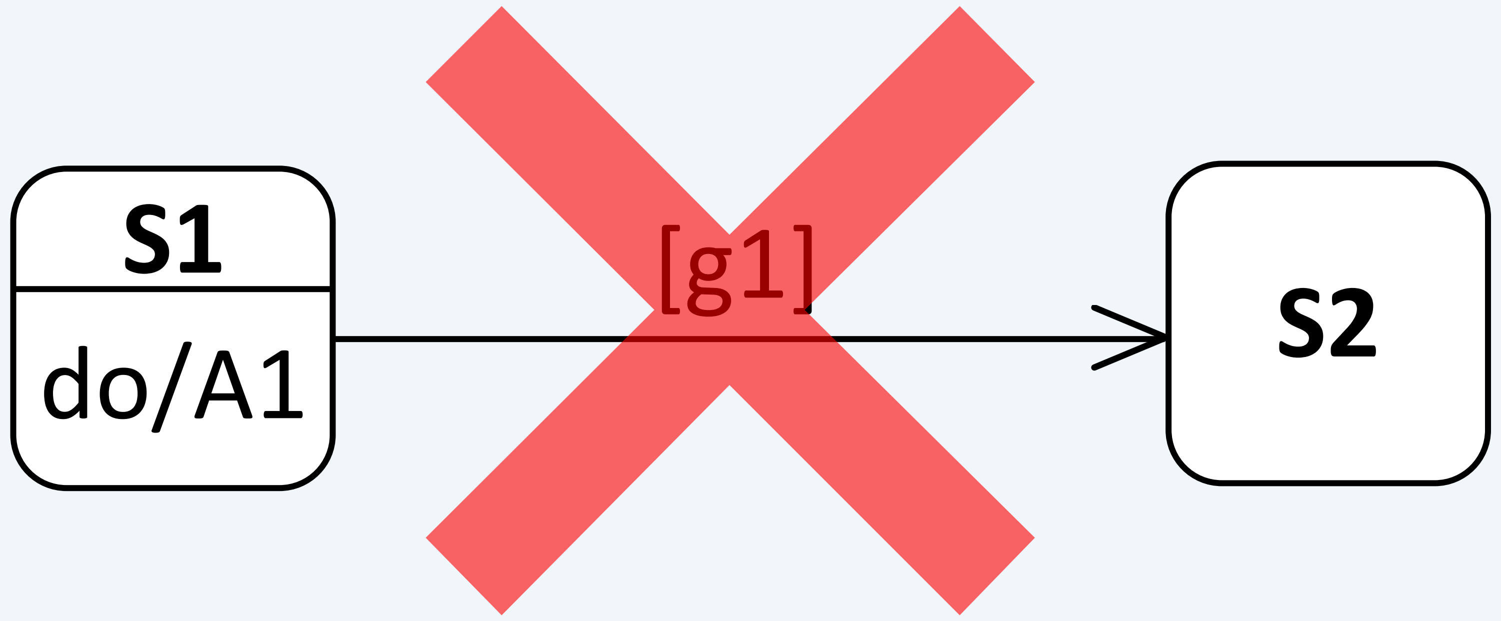


If `e1` occurs, `A1` is canceled and the object changes to the `S2` state



If `e1` occurs and `g1` is true, `A1` is canceled and the object changes to state `S2`

As soon as `A1` is completed, a completion event is generated that triggers the transition to `S2`



rarely models the desired situation

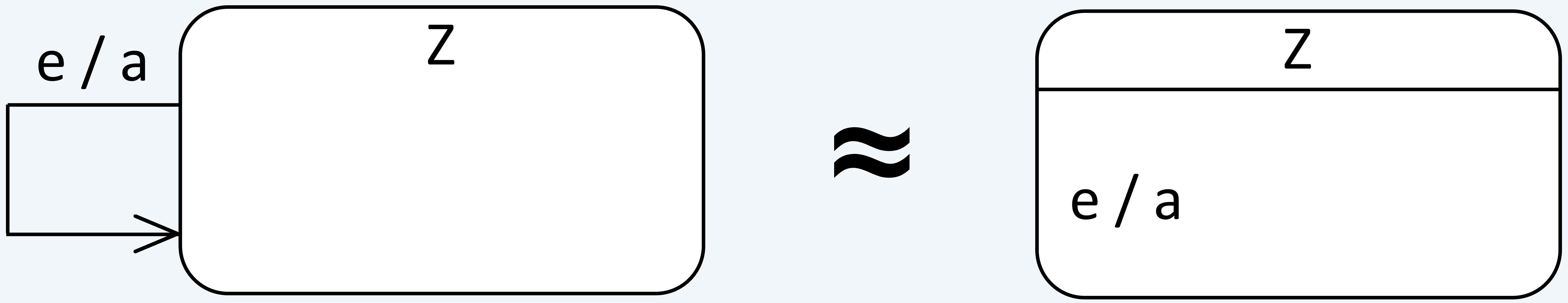
As soon as `A1` is completed, a completion event is generated and `g1` is evaluated. If `g1` is true, the transition to `S2` occurs. If `g1` is false, the transition can never take place

State Machine Diagram
The Internal Transitions

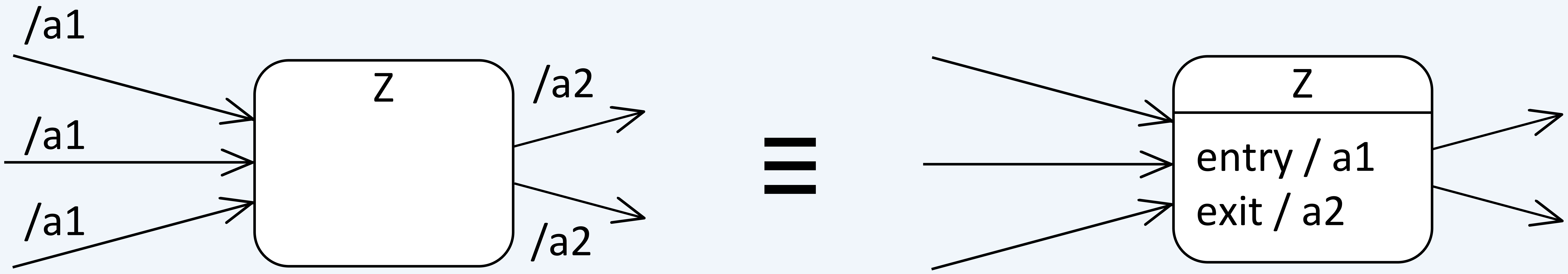
Christian Huemer und Marion Scholz
Presented by Nicholas Bzowski

State Transition: Internal Transitions (1/2)

- Are triggered by events like "external" transitions, but **do not leave the current state**
- Equivalent to self-transition if no **`entry`** /**`exit`** activities are available

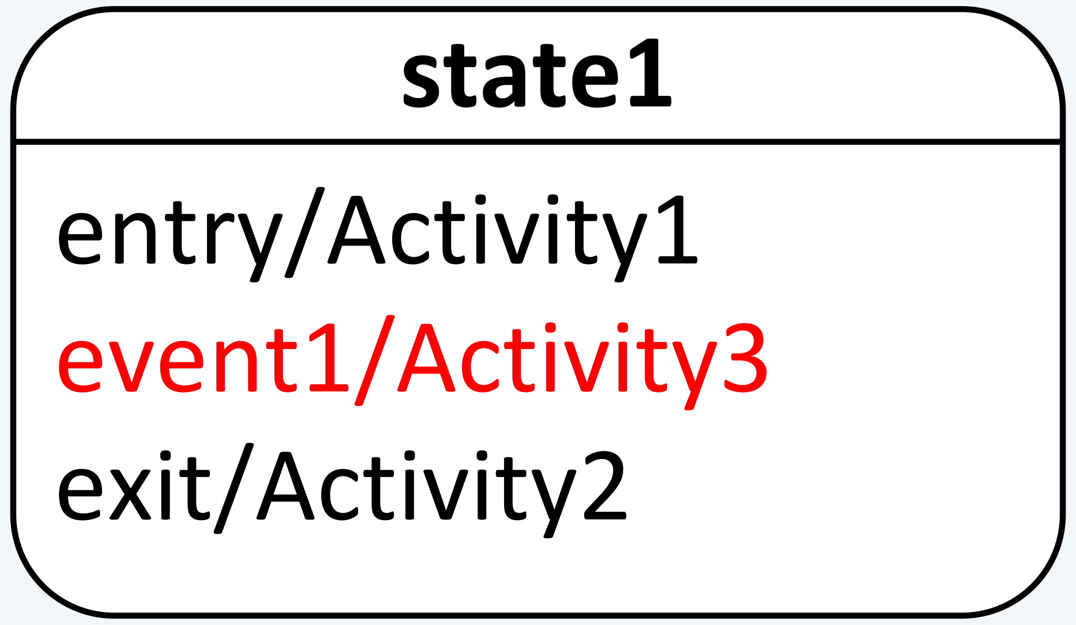


- Identical activities can be moved into the state:

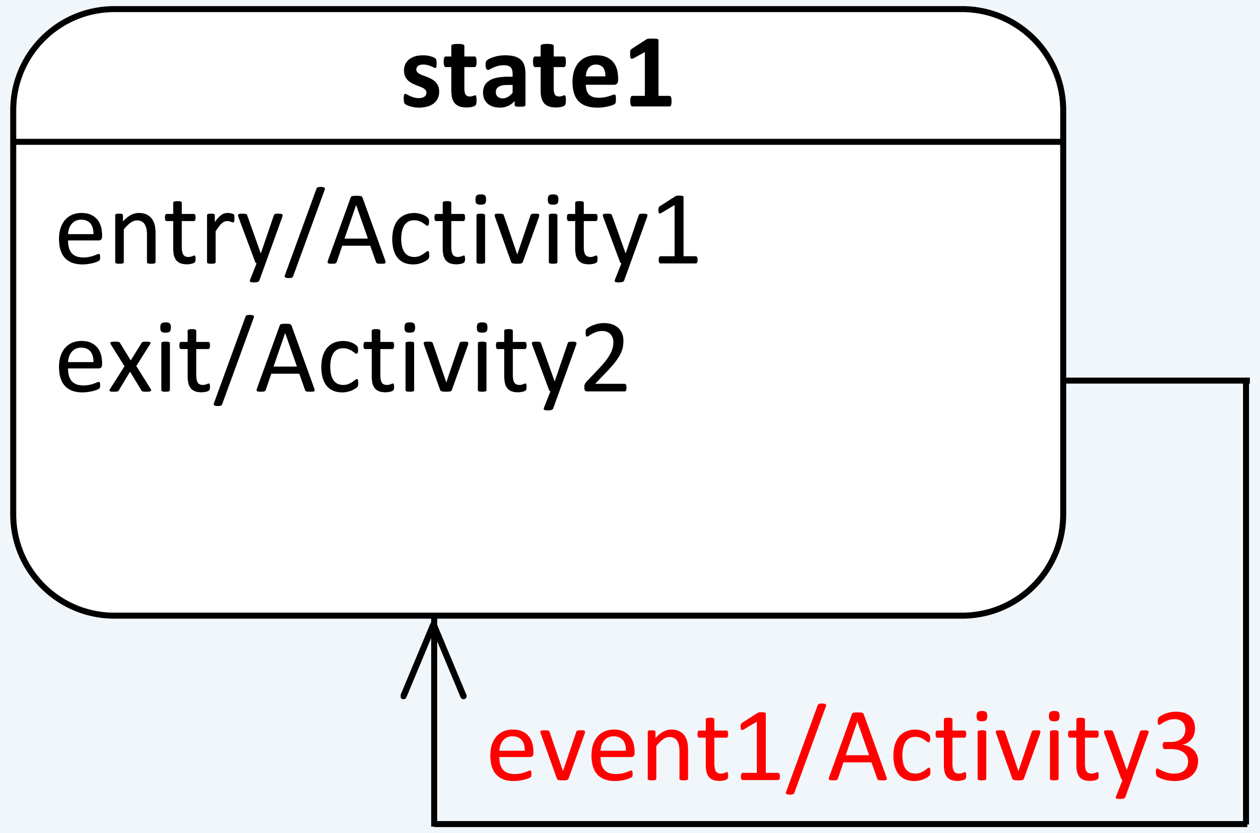


State Transition: Internal Transitions (2/2)

Interal Transition



Self-transition

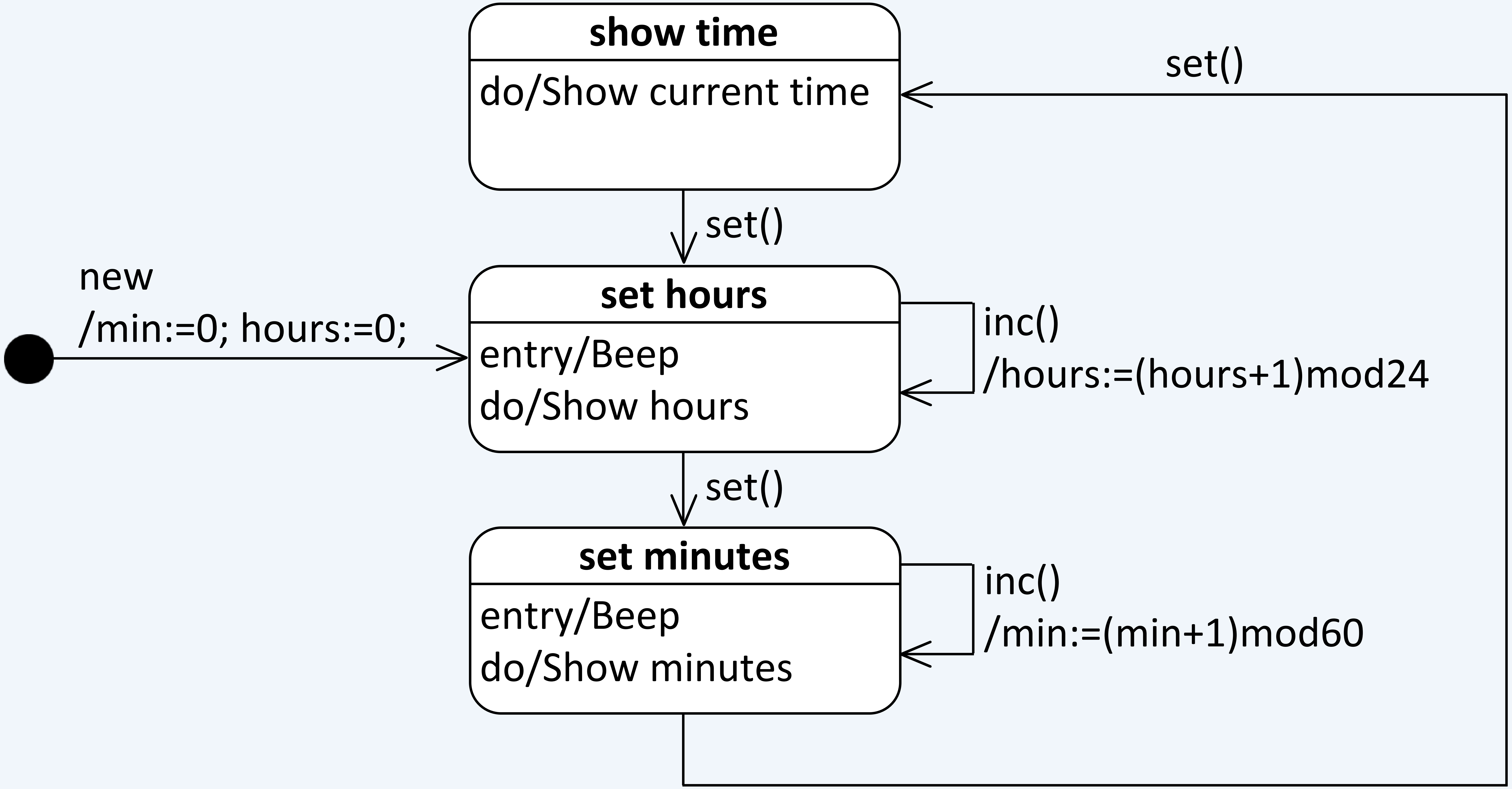


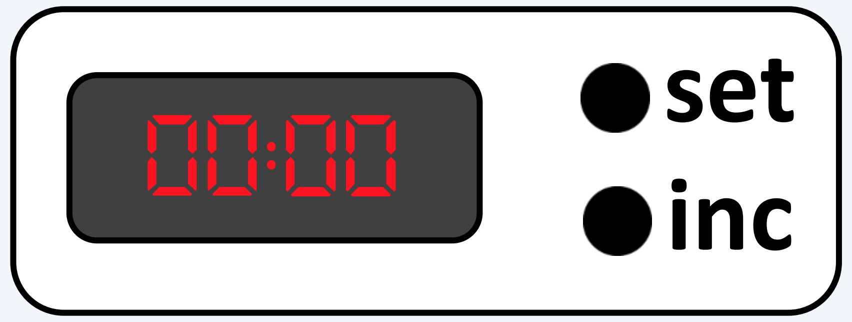
- If **`event1`** occurs
  - Object remains in **`state1`**
  - **`Activity3`** is executed

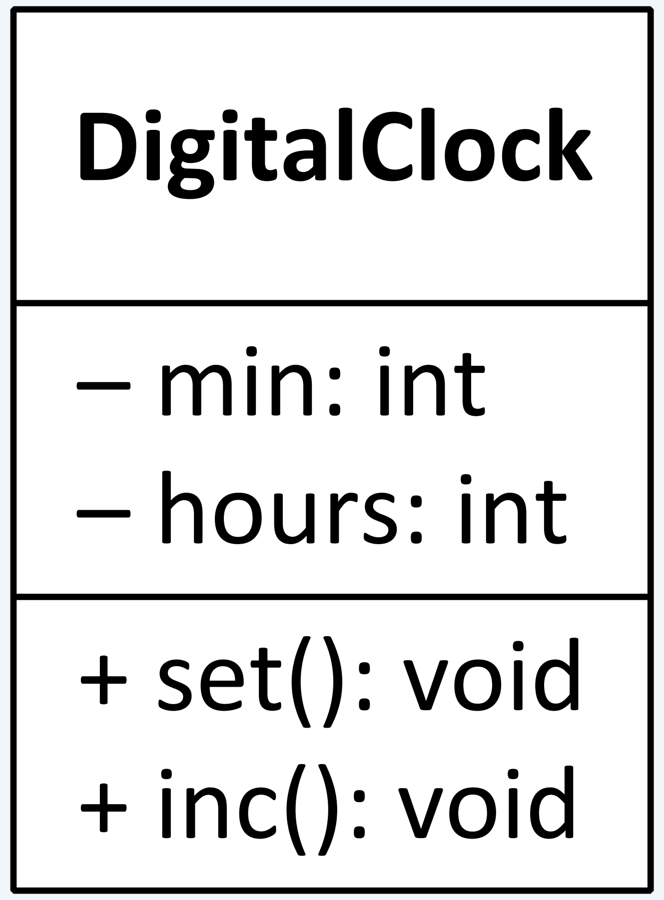
- If **`event1`** occurs
  - Object exits **`state1`** and **`Activity2`** is executed
  - **`Activity3`** is executed
  - Object enters **`state1`** and **`Activity1`** is executed

Example: Digital Clock

- The states that a digital clock can assume when setting the clock are modeled.

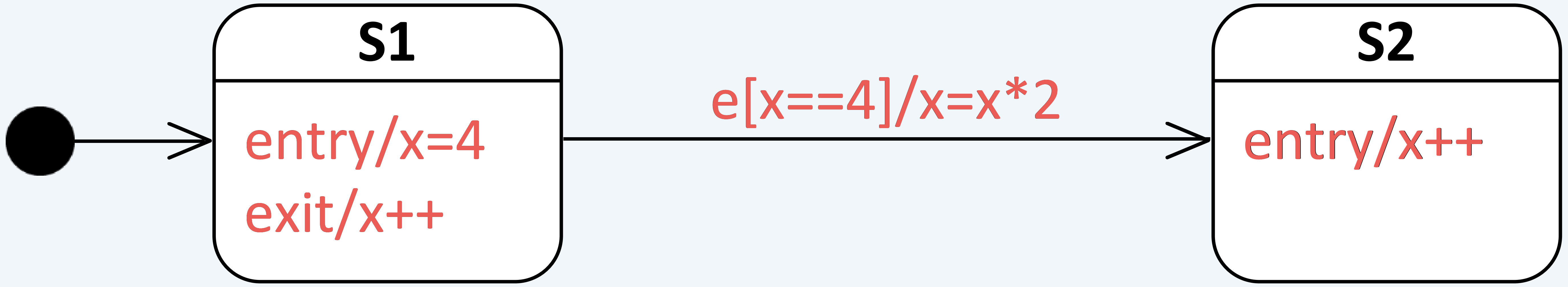






Execution Sequence of Activities - Example

- Assuming **S1** is active
  … what value does **x** have after the occurence of **e** ?



S1 becomes active, **x** is assigned the value **4**

e occurs, the condition is evaluated and is true

S1 is exited, **x** is assigned the value **5**

The state transition takes place, **x** is assigned the value **10**

S2 is entered, **x** is increased by **1** and now has the value **11**

State Machine Diagram
The Event Types

Christian Huemer und Marion Scholz
Presented by Nicholas Bzowski

StateTransition: Event Types (1/2)

- CallEvent
  Receipt of a message (operation call)
  - e.g.: `occupy()`,
    `register()`
- SignalEvent
  Receipt of a signal
  - e.g.: `rightmousedown`,
    `receiveSMS`
- TimeEvent
  - Relative: related to the time of entry into the currently active state
    - e.g.: `after(5 seconds)`
  - Absolute
    - e.g.: `when(time==16:00)`,
      `when(date==20250101)`

StateTransition: Event Types (2/2)

- **ChangeEvent**

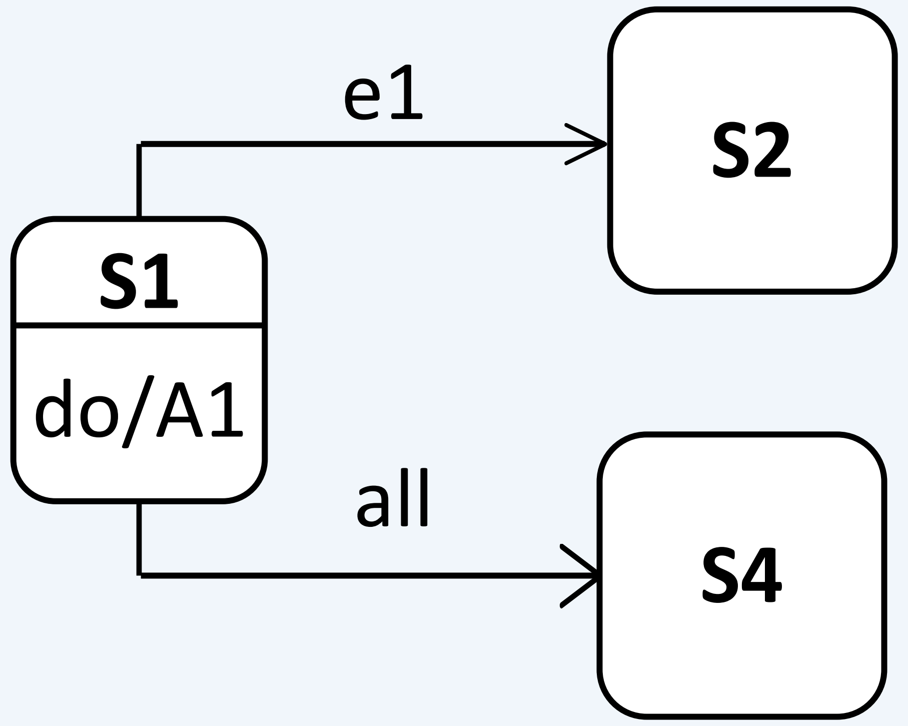
  The fulfillment of a condition is continuously monitored

  - Ex.: `when(x > y)`

- **Completion event**

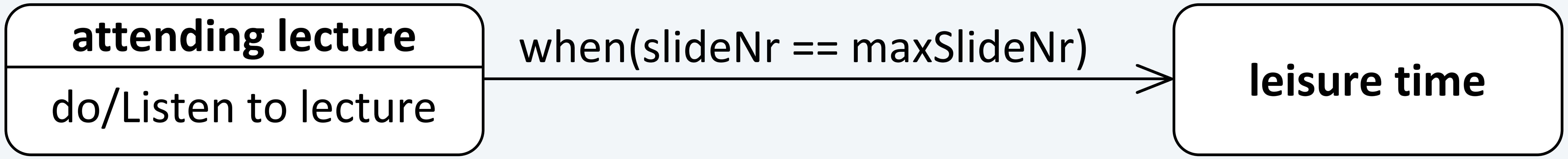
  Is triggered when all `do` activities of the current state have been completed

- **Any receive event**
  - Occurs when any event occurs that does not trigger another transition in the active state
  - A type of "else-transition"
  - Keyword `all`

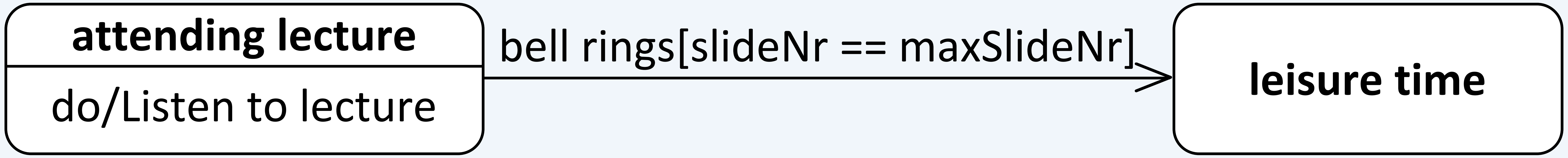


Difference Between ChangeEvent and Guard

- **ChangeEvent**:
  - Condition is continually checked
  - If the condition is true, the associated state transition can be triggered (if not blocked by the associated guard condition)



- **Guard**:
  - Only checked if assigned event occurs
  - Cannot trigger a state transition itself



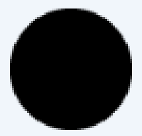
What happens when the last slide is reached before the bell rings?

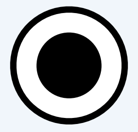
What happens if the bell rings before the last slide is reached?

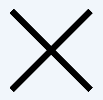
State Machine Diagram
The Lifecycle of an Object

Christian Huemer und Marion Scholz
Presented by Nicholas Bzowski

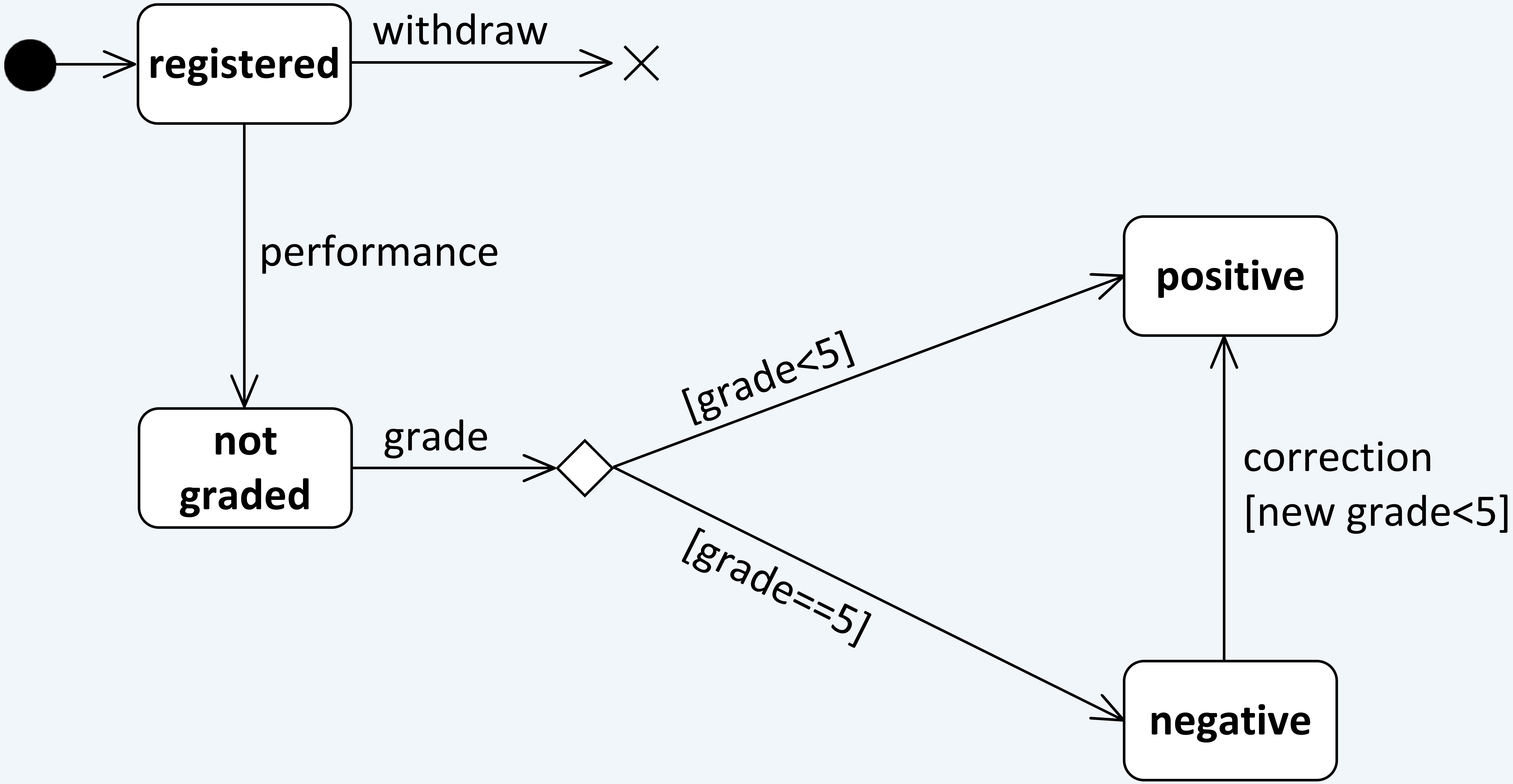
Initial and End State, Terminate Node

- **Initial State**
  
  - "Start" of the state machine diagram
  - Exactly one outgoing transition
    - Is triggered immediately when the system is in the initial state
    - No conditions and events (exception: event for creating the object under consideration)
    - Specification of activities is allowed

- **Final State**
  
  - No outgoing transitions
  - Not a pseudostate!

- **Terminate Node**
  
  - Object whose behavior is modeled ceases to exist

Example: Exam Attempt



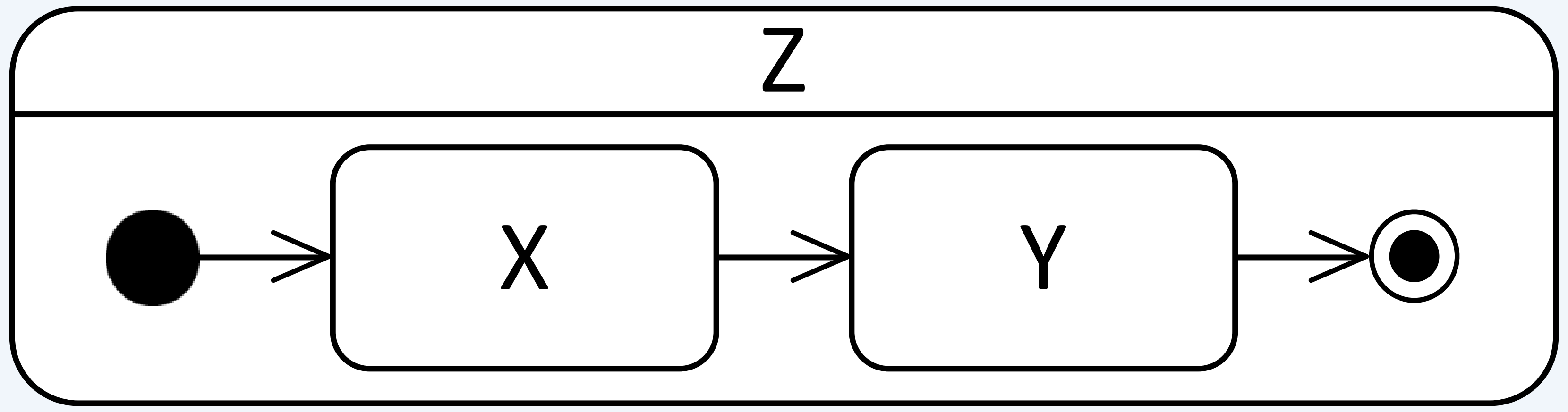
State Machine Diagram
The Composite State

Christian Huemer und Marion Scholz
Presented by Nicholas Bzowski

Composite States

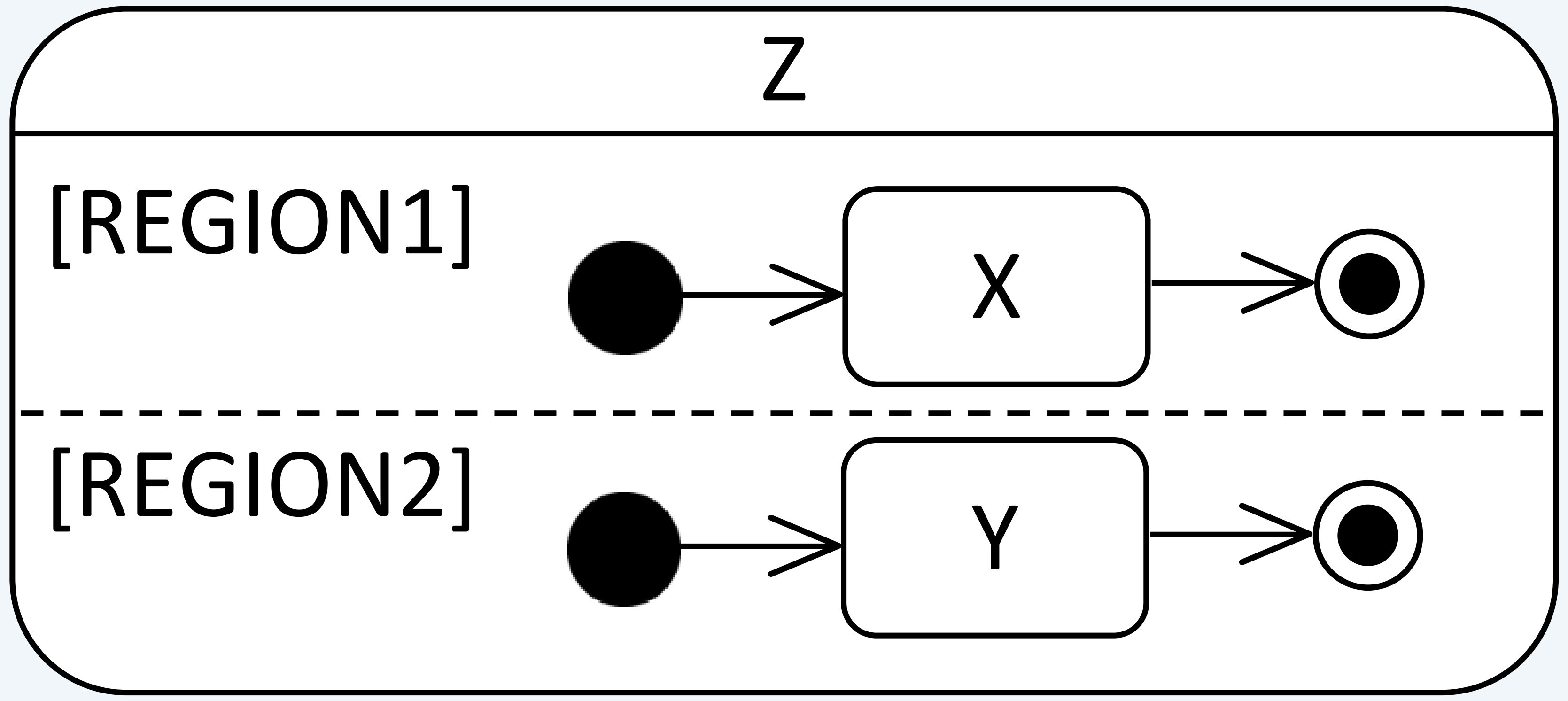
- Are composed of multiple substates

  → Nested state machine diagram
- The substates are disjoint



Only **X** OR **Y** can be active at any given time!

- Division of the superstate into several regions
  - → the substates are parallel, simultaneously active
  - **Z** = "orthogonal state"

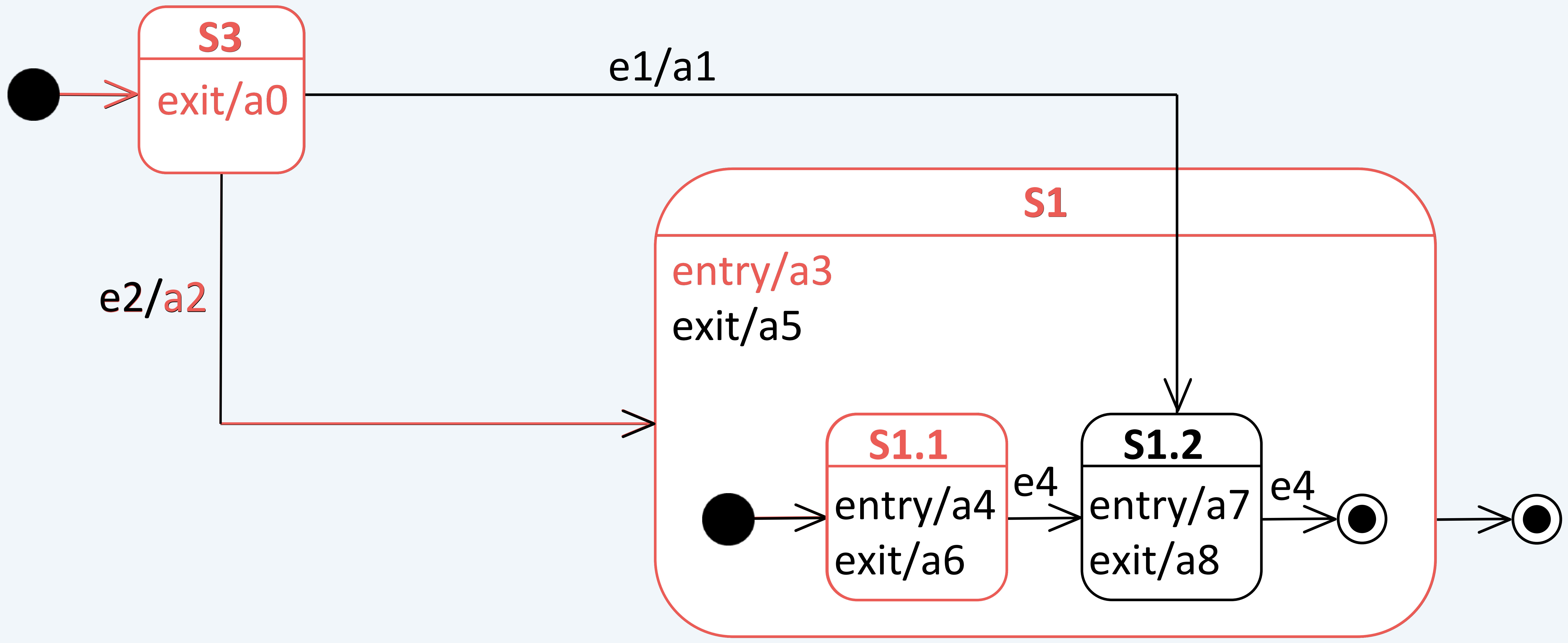


X AND **Y** are active at the same time!

Entering a Composite State (1/2)

- Transition to the edge of a composite state: Initial state is activated

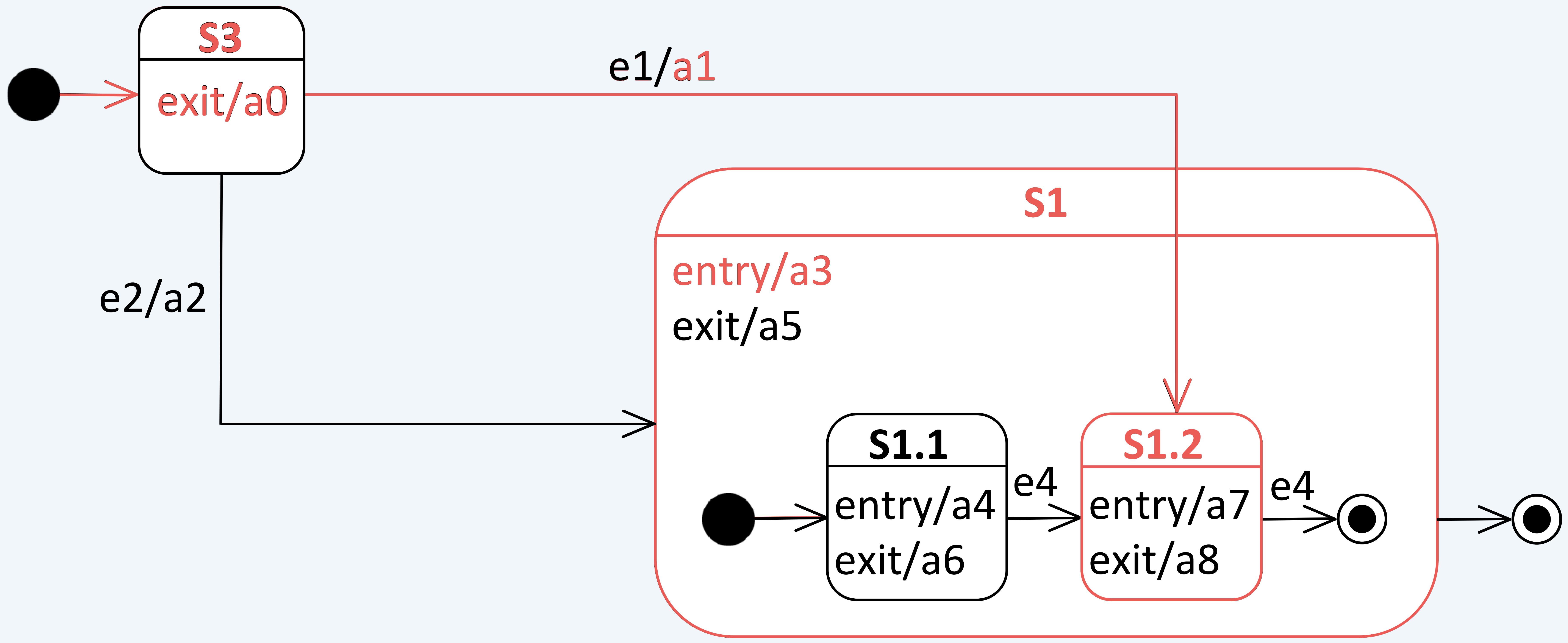
Event	State	Activities Performed
"Start"	S3	
e2	S1/S1.1	a0-a2-a3-a4



Entering a Composite State (2/2)

- Transition to a substate:
  Substate is activated

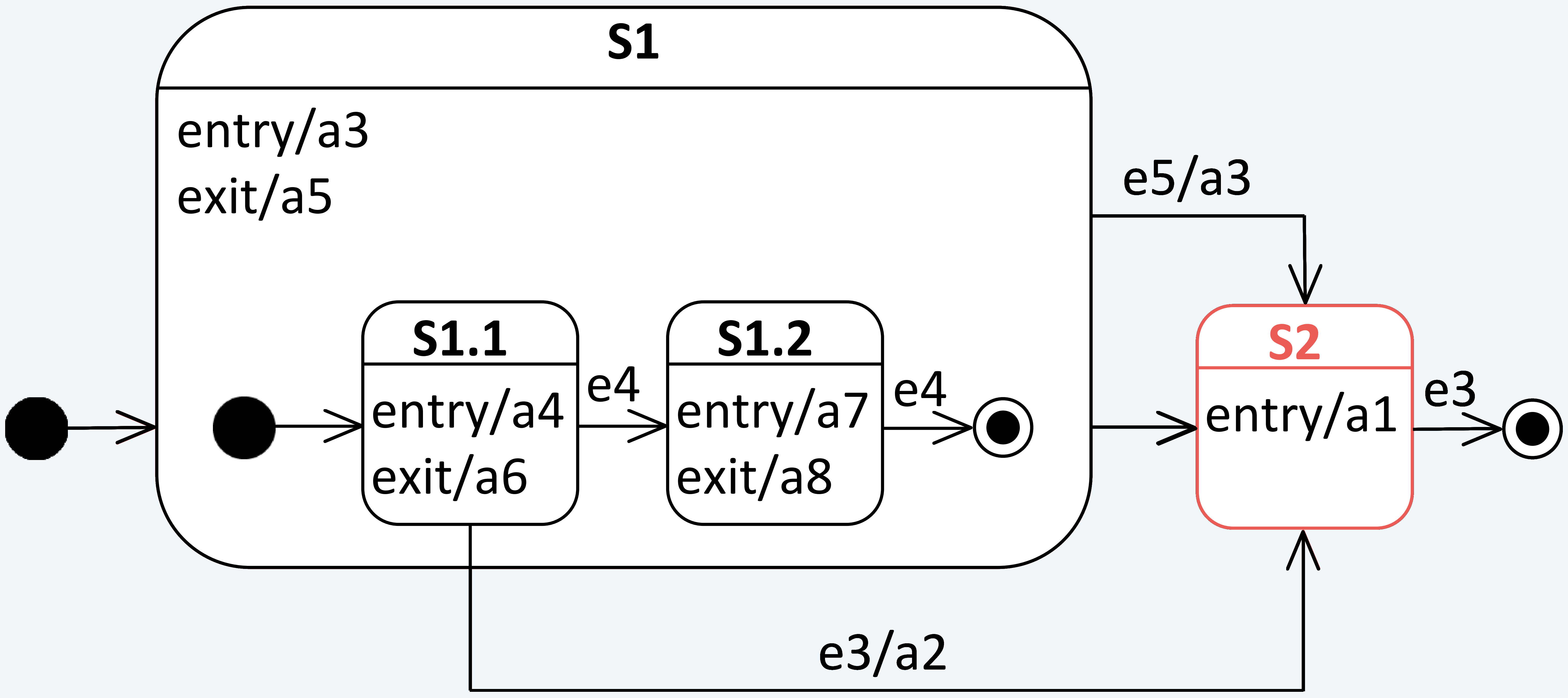
Event	State	Activities Performed
“Start”	S3	
e1	S1/S1.2	a0-a1-a3-a7



Exiting a Composite State (1/3)

- Transition from a substate

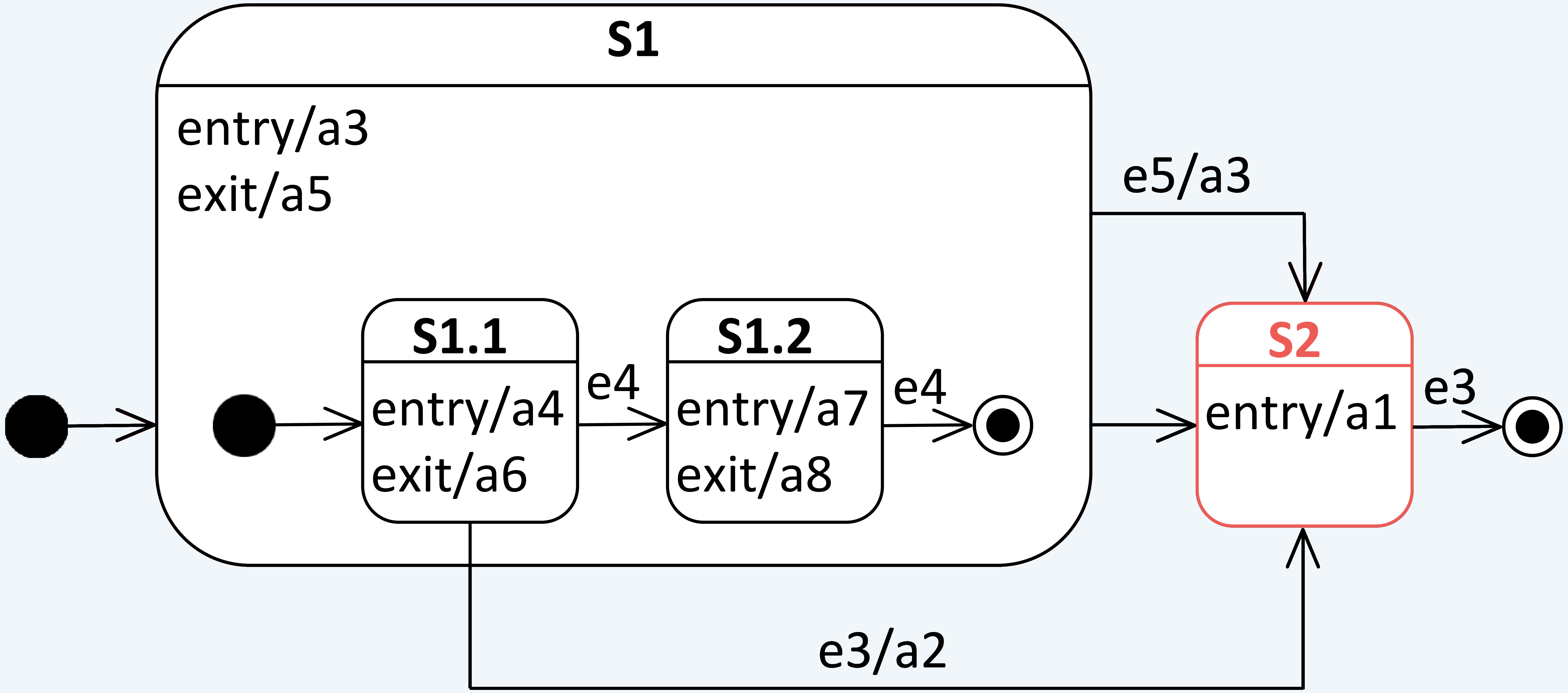
Event	State	Activities Performed
“Start”	S1/S1.1	a3-a4
e3	S2	a6-a5-a2-a1



Exiting a Composite State (2/3)

- Transition from the edge of a composite state

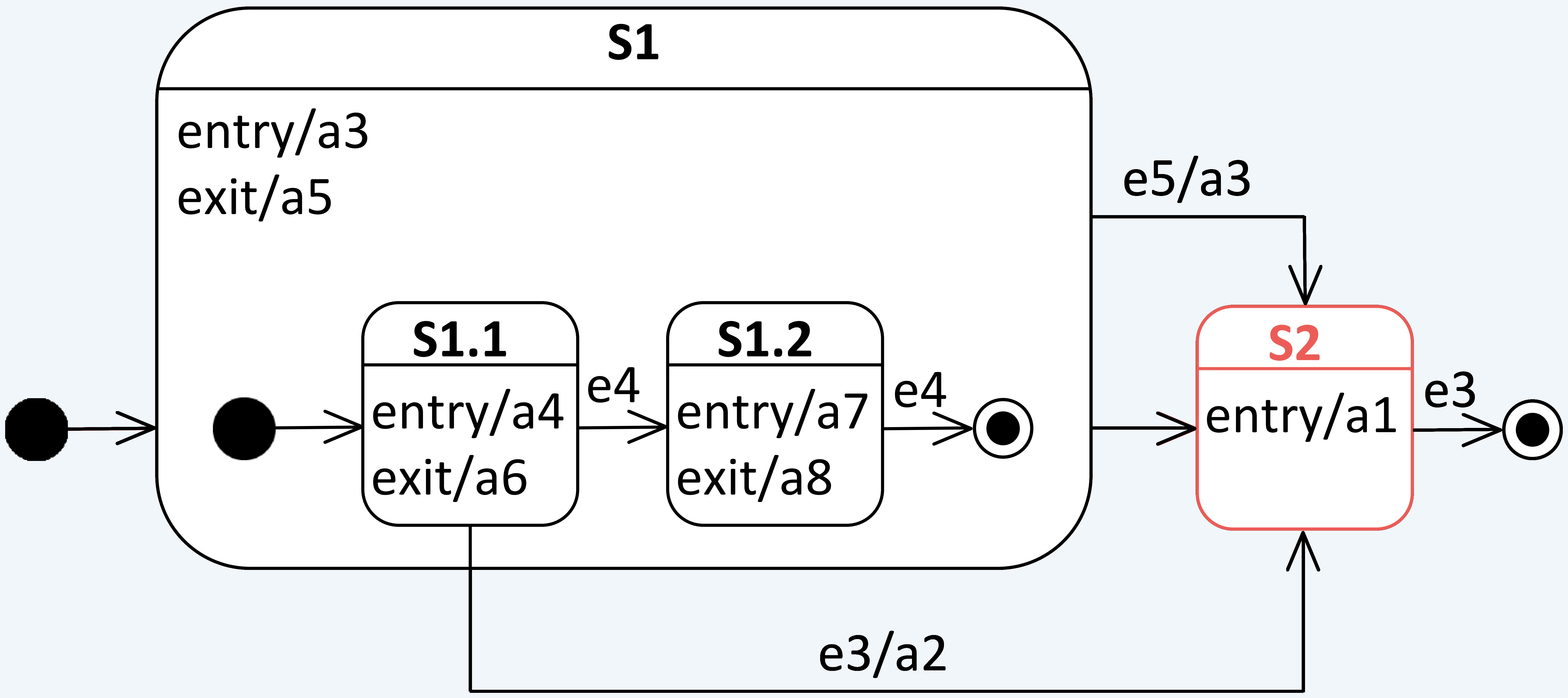
Event	State	Activities Performed
“Start”	S1/S1.1	a3-a4
e5	S2	a6-a5-a3-a1



Exiting a Composite State (3/3)

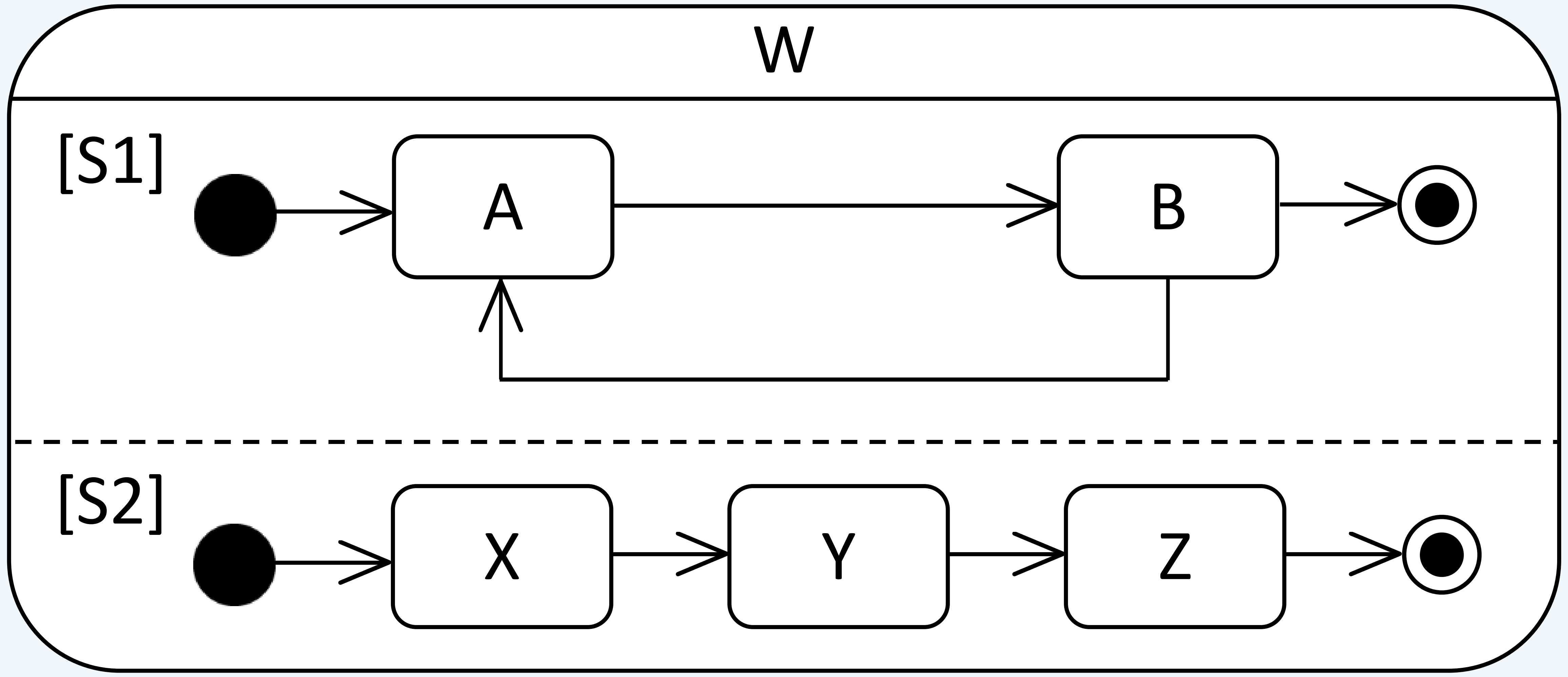
- Transition by ending the substate sequence

Event	State	Activities Performed
“Start”	S1/S1.1	a3-a4
e4	S1/S1.2	a6-a7
e4	S2	a8-a5-a1



Orthogonal (Concurrent) States

- Ex.:



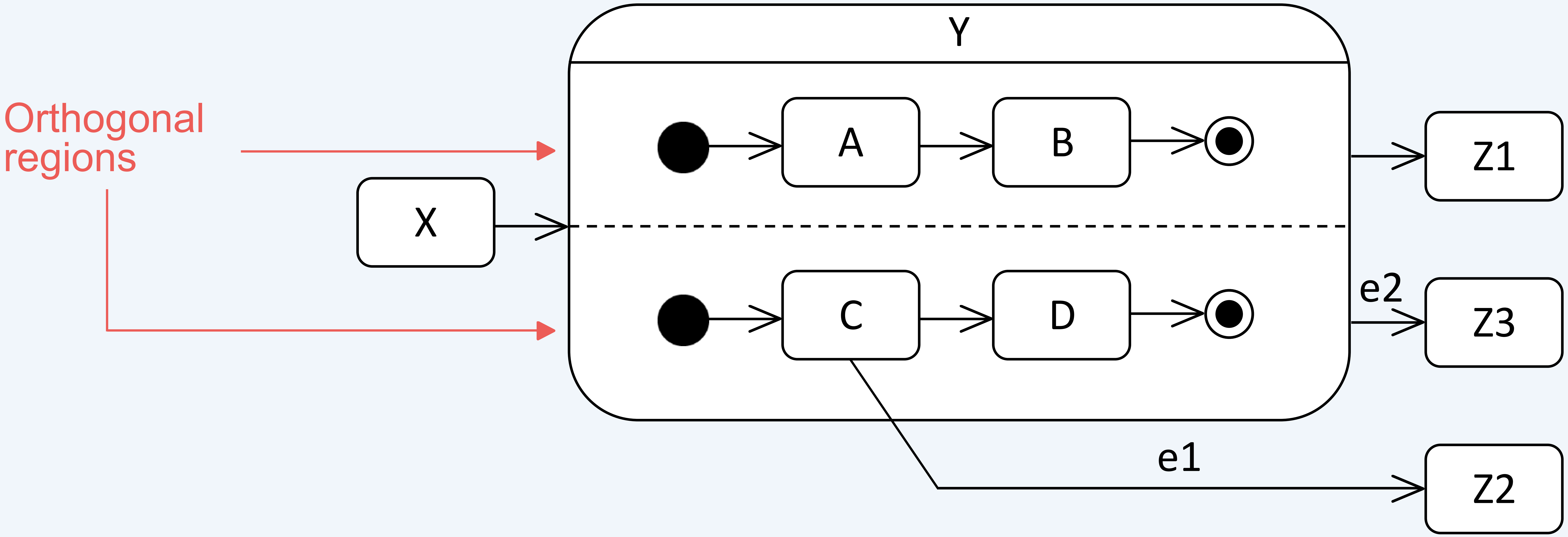
At any given time, one substate of each of the two orthogonal (=parallel) regions of **`W`** is active!

- Possible combinations of simultaneously active states:

  `A` & **`X`** or **`A`** & **`Y`** or **`A`** & **`Z`** or **`A`** & Final State of **`[S2]`**
  `B` & **`X`** or **`B`** & **`Y`** or **`B`** & **`Z`** or **`B`** & Final State of **`[S2]`**

  or Final State of **`[S1]`** & **`X`** or Final State of **`[S1]`** & **`Y`** or Final State of **`[S1]`** & **`Z`** or Final State of **`[S1]`** & Final State of **`[S2]`**

Exiting Orthogonal States

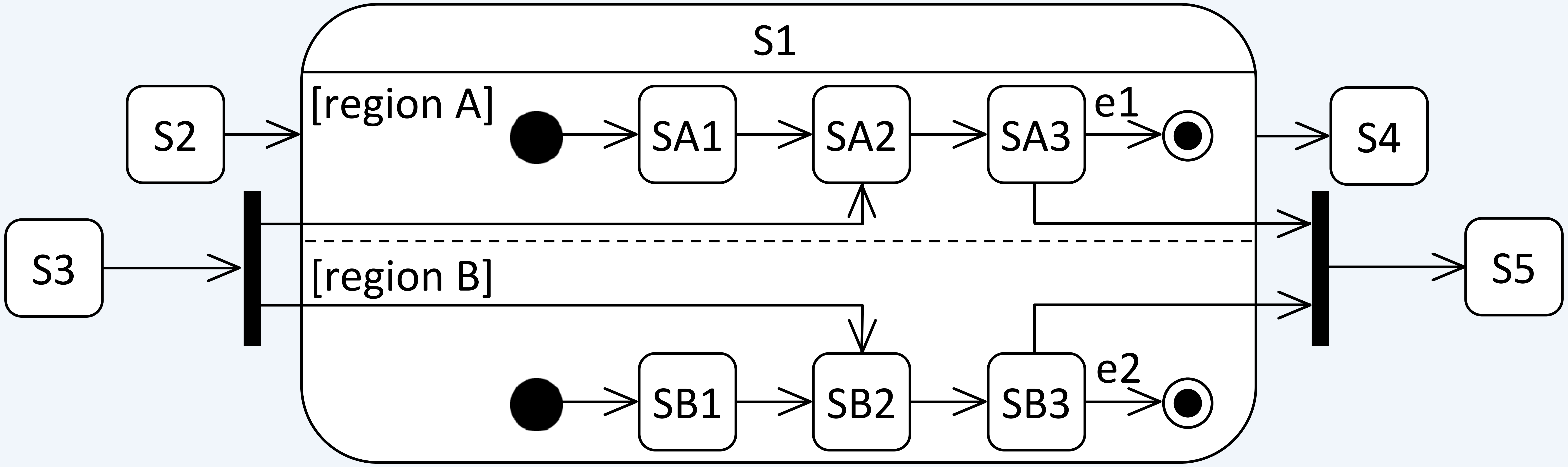


- The composite state **Y** is exited when
  - **B** and **D** have been exited (subsequent state **Z1**)
    [= the substate sequences have ended]
  - event **e2** occurs in any substate (subsequent state **Z3**)
  - event **e1** occurs in state **C** (subsequent state **Z2**)

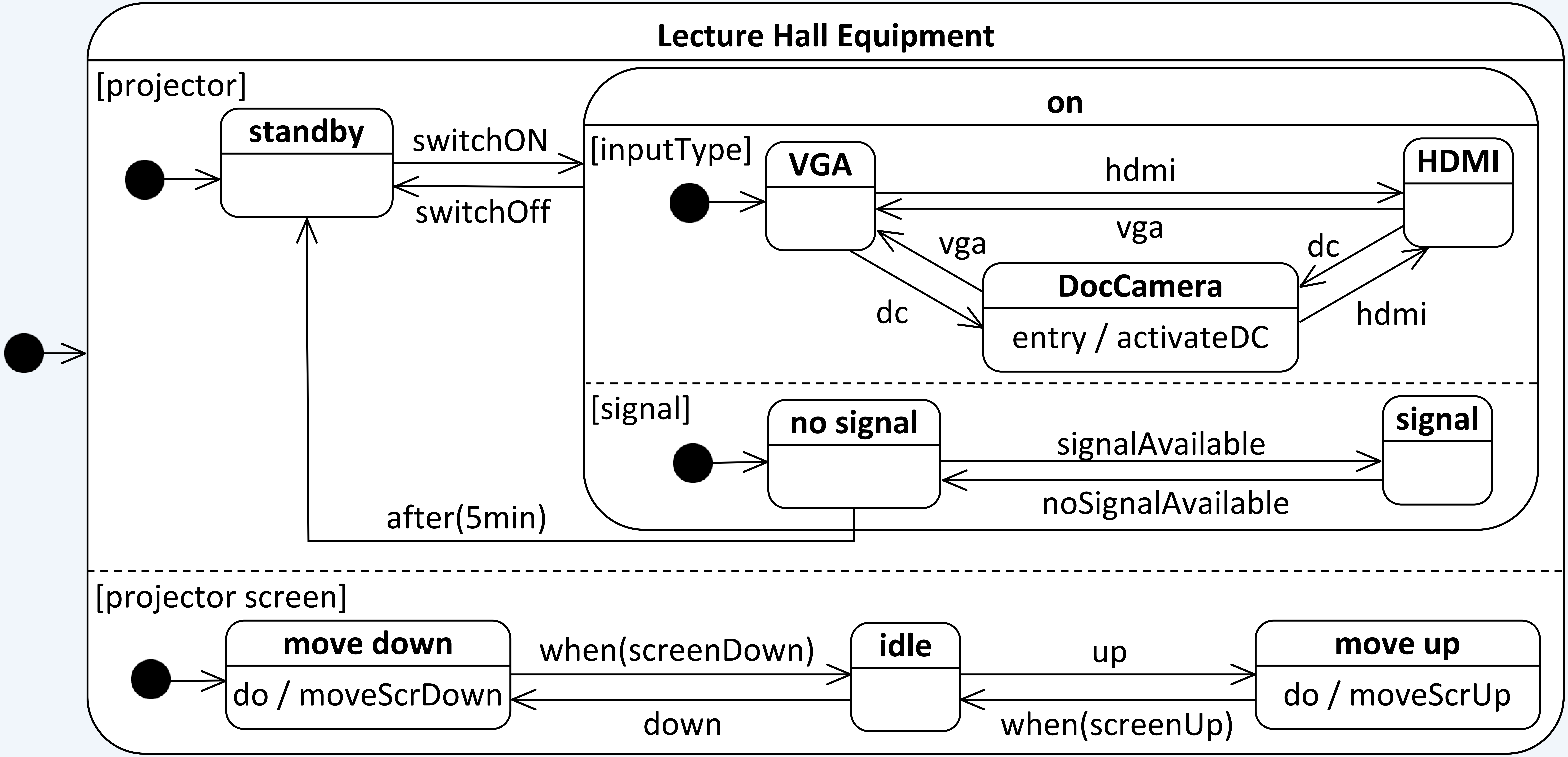
Complex Transition for Orthogonal States

- If an orthogonal state is activated, **all** initial nodes of is **adjacent regions are activated**
- However, if you want to start the control flow at other positions, you use parallelization or synchronization nodes
- Parallelization node
  - Target states must be in different regions
  - Source state must be outside the orthogonal state

  (Rules for synchronization nodes in reverse)

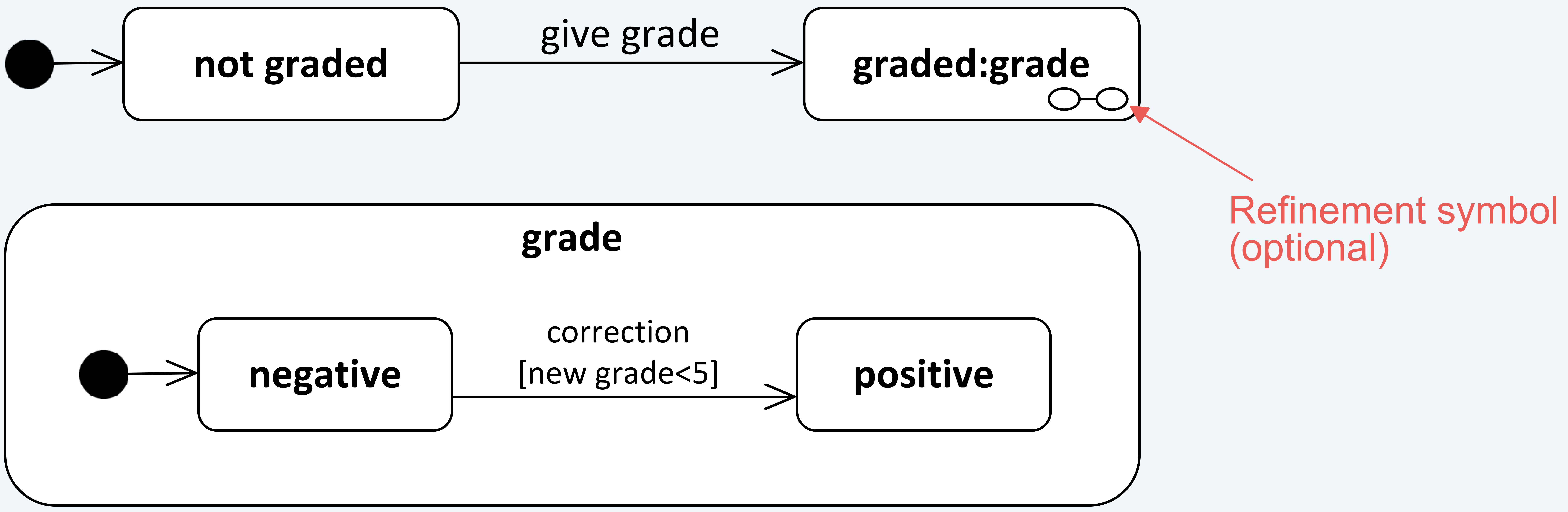


Example: Lecture Hall Equipment



Submachines

- To reuse parts of a state machine diagram in other state machine diagrams
- Notation: `State:SubmachineState`
- As soon as the submachine state is activated, the initial state of the referenced submachine is activated
  - Similar to subroutine calls in programming languages



State Machine Diagram
The History State

Christian Huemer und Marion Scholz
Presented by Nicholas Bzowski

History State

- Historical states remember the last **internal** state in a composite state when leaving this composite state.

- It is possible to **return** to this state at a later time
  - All **`entry`** activities are executed again

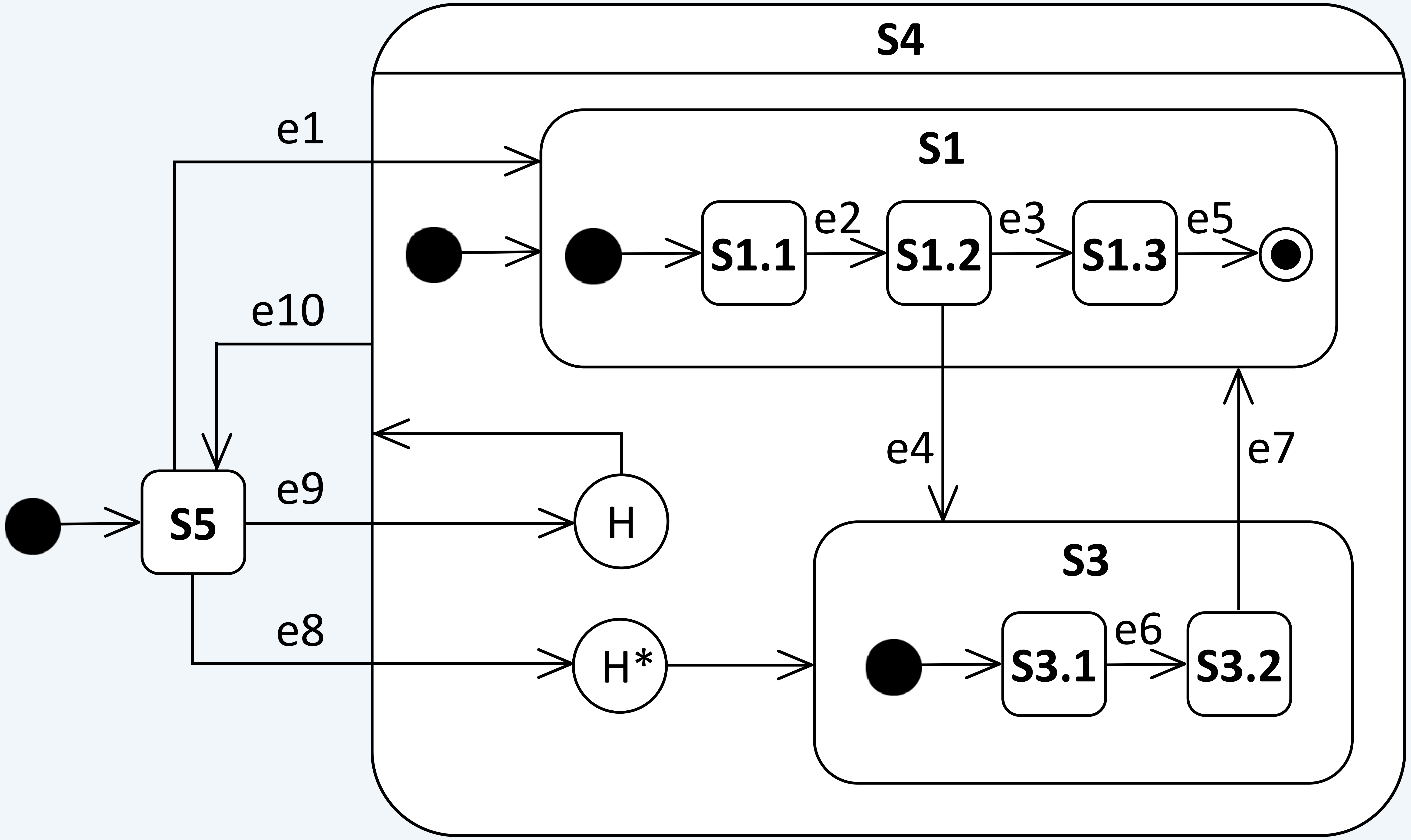
- **Shallow history state** remembers **one level**

H

- All substates are retained over **the entire nesting depth** via a **deep history state "H*"**

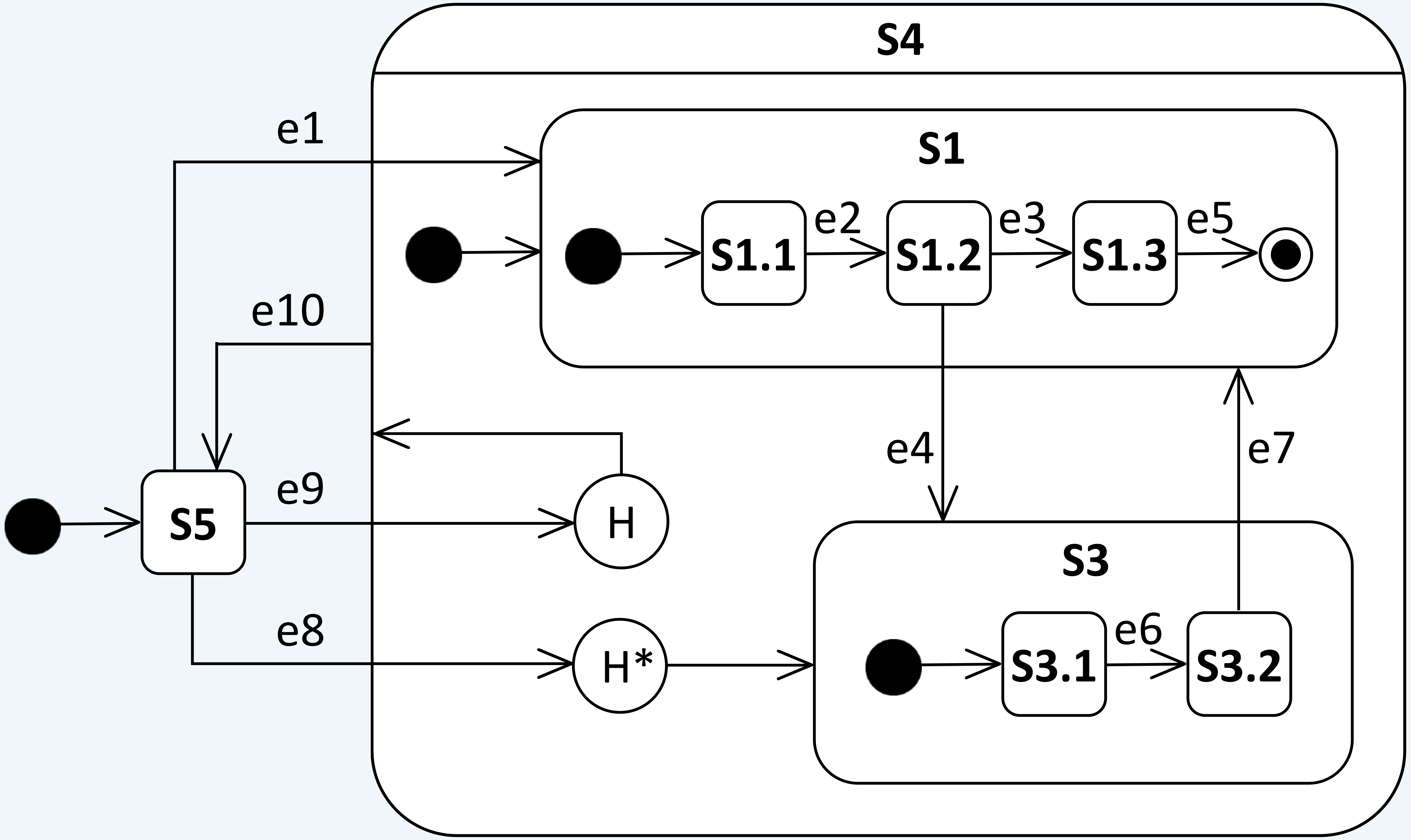
H*

Example: History State (1/4)



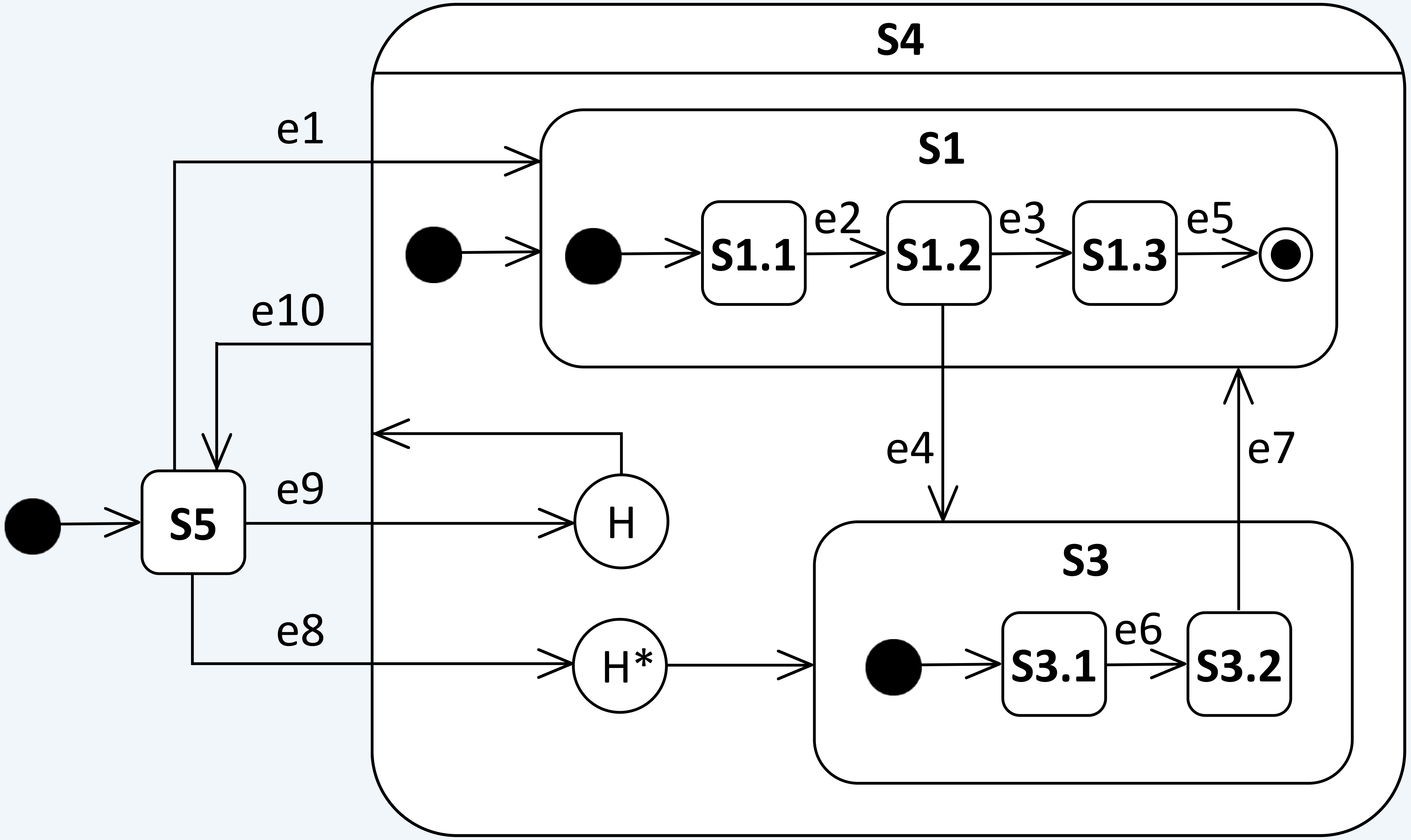
Event	State
“Start”	S5
e1	S4/S1/S1.1
e2	S4/S1/S1.2
e10	S5
e9	(H→) S4/S1/S1.1

Example: History State (2/4)



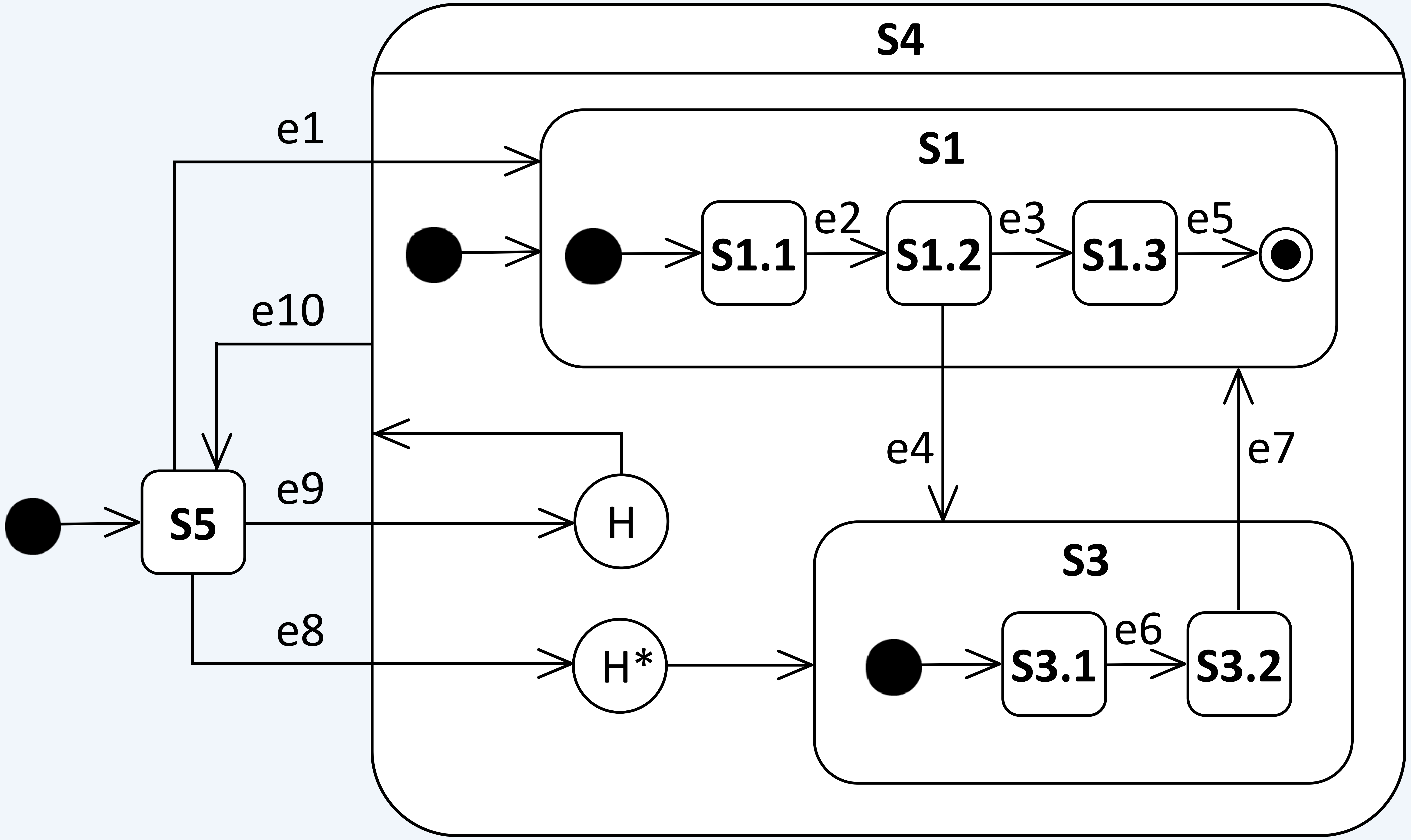
Event	State
“Start”	S5
e1	S4/S1/S1.1
e2	S4/S1/S1.2
e10	S5
e8	(H*→) S4/S1/S1.2

Example: History State (3/4)



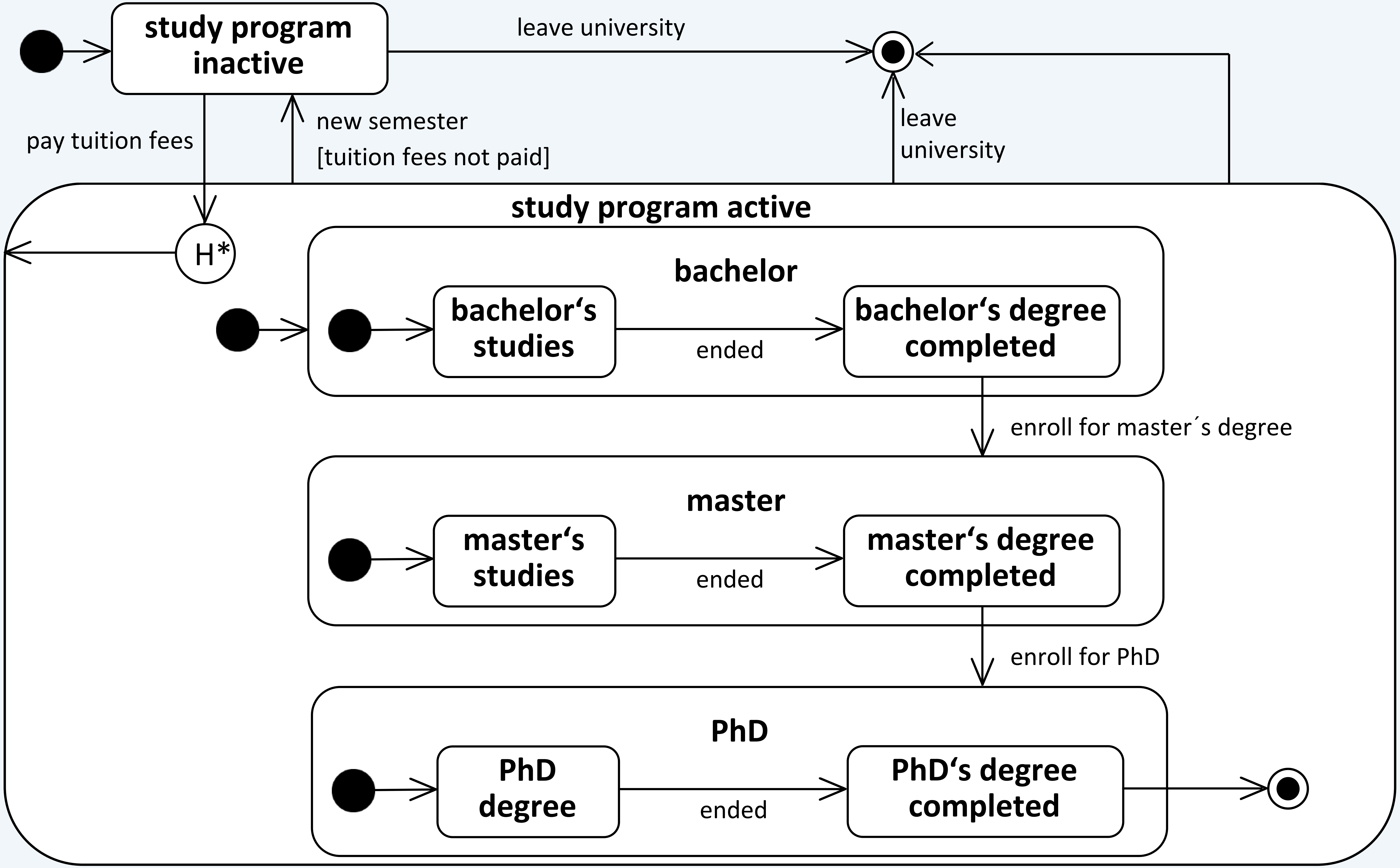
Event	State
"Start"	S5
e9	(H→) S4/S1/S1.1

Example: History State (4/4)



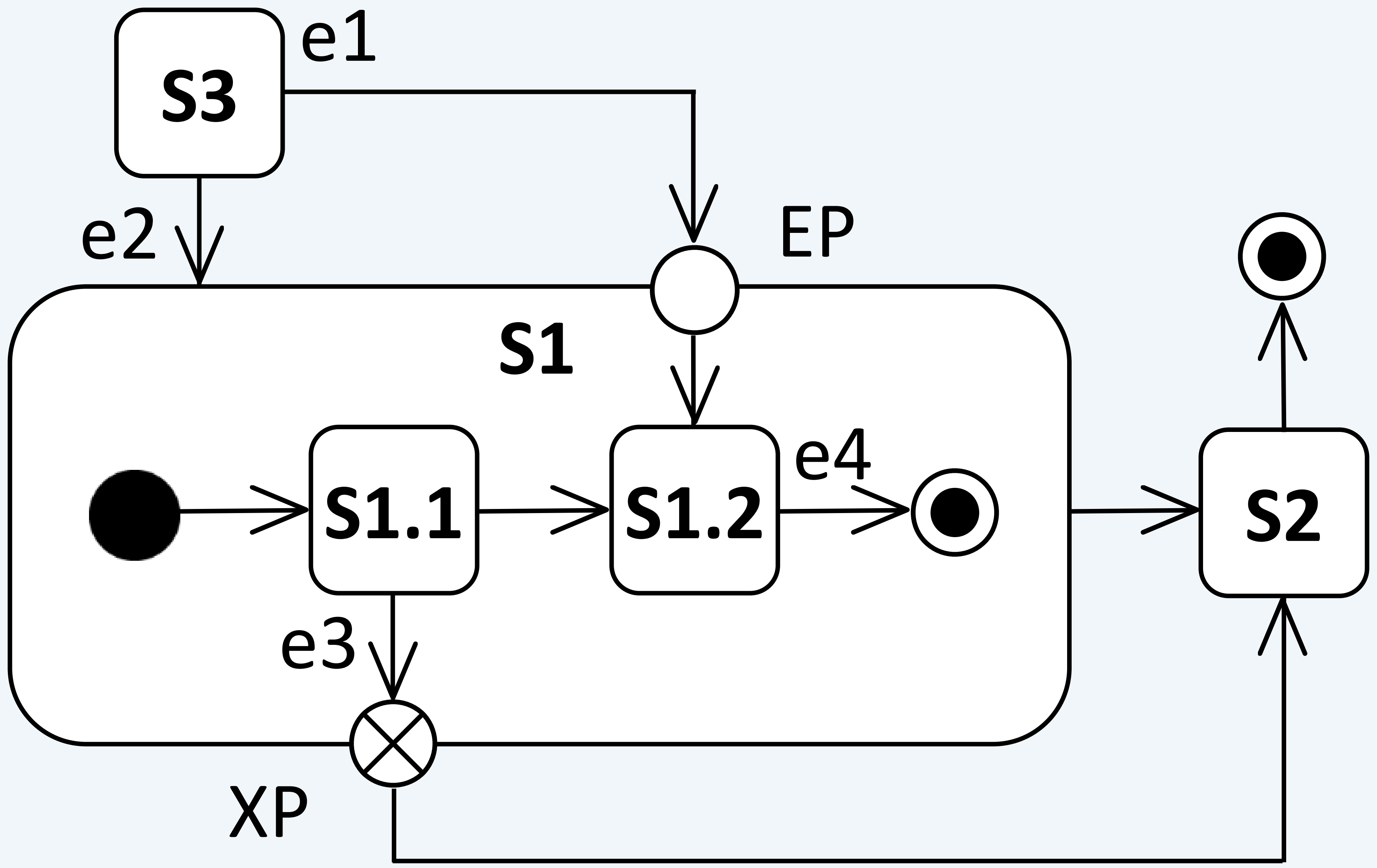
Event	State
“Start”	S5
e8	(H*→) S4/S3/S3.1

Example: States of an academic education



Entry and Exit Points

- Encapsulation mechanism
  - Transition to a specific substate of a composite state without the external transition having to know the structure of the composite state
  - Similar transition out of a composite state



Example: Entry and Exit Points