UML State Machines (Basics)

Software Design (40007) – 2023/2024

Justus Bogner
Ivano Malavolta

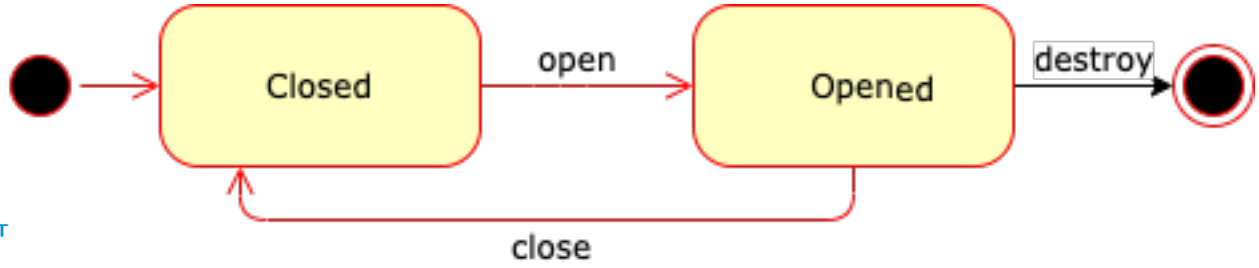
Software and Sustainability research group (S2)
Department of Computer Science, Faculty of Sciences

What is the state of an object?

- Concrete state

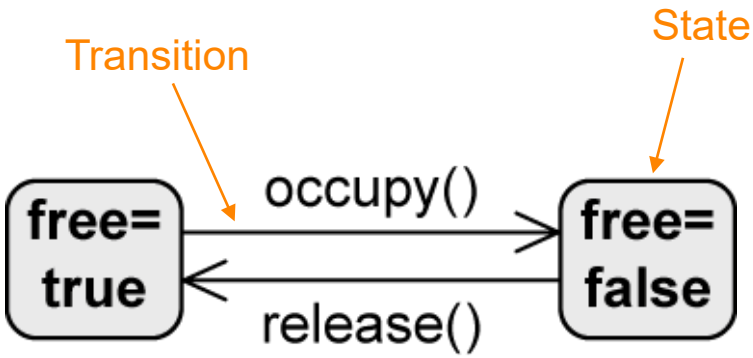
```
1 public class Door {
2     private int height = 0;
3     private int width = 0;
4     private Color = Color.BLUE;
5     private orientation = Orientation.SOUTHWEST;
6     private bool closed = false;
7 }
```

- Abstract state
  - Arbitrarily-defined set of **logical** states
  - Goal: to define the abstract state space of objects to better reason about the system and arrive at a suitable solution

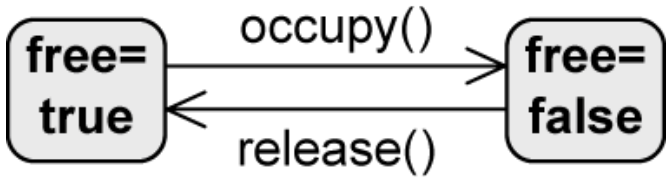


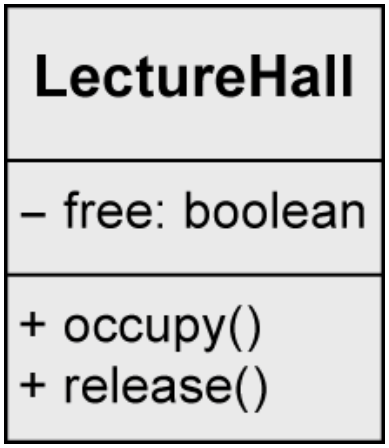
State machines

- Every domain object has a **finite** set of states during its life
- State machines are used to
  - reason about the **complexity** of the domain and the system
  - avoid overlooking important execution paths
- State machine diagrams
  - model the possible states of a system or object
  - show how state transitions occur as a consequence of events
  - show what behavior the system or object exhibits in each state
- Example: high-level behavior of a lecture hall



Example: lecture hall with details

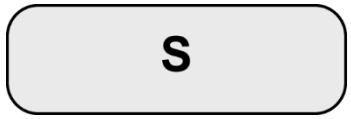


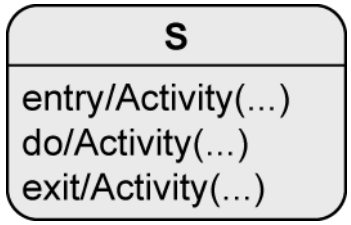



```
class LectureHall {
   private boolean free;

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

States

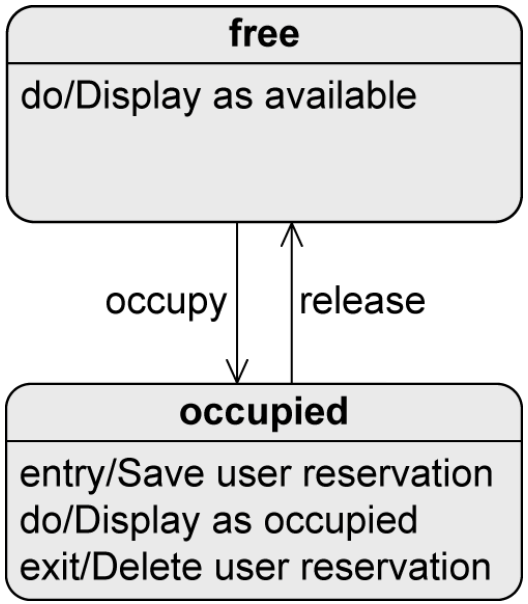




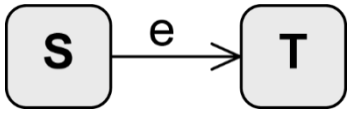
- States = nodes of the state machine
- When a state is active
  - The object is in that state
  - All **internal activities** specified in this state can be executed

Internal activities:

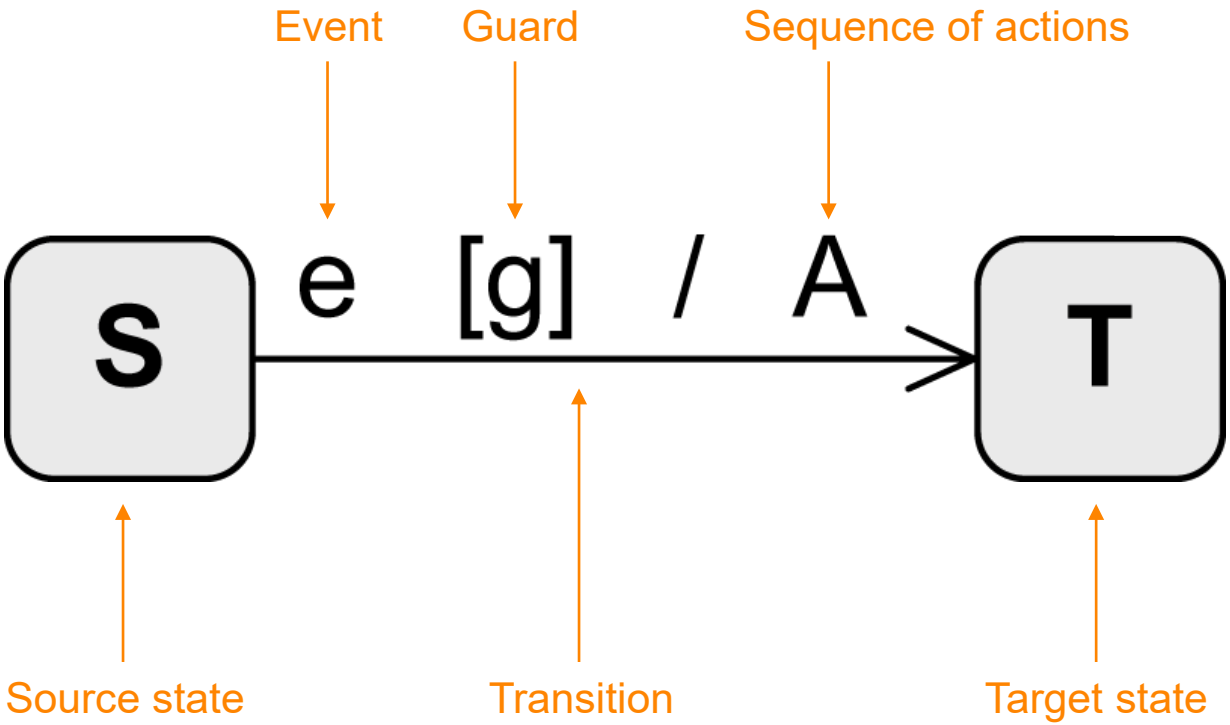
- entry / <activity>
  - Executed when the object enters the state
- exit / <activity>
  - Executed when the object leaves the state
- do / <activity>
  - Executed while the object remains in this state



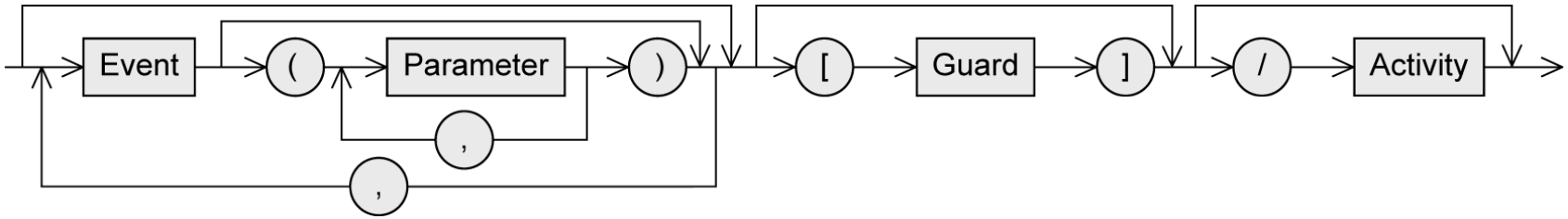
Transitions



Changing from one state to another



Transition – syntax



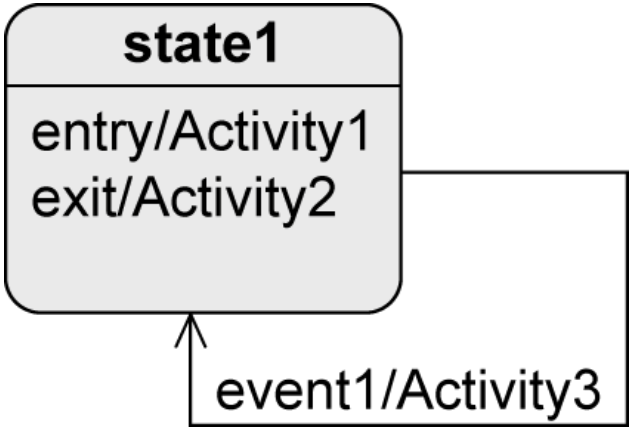
- Event(s) → trigger
  - Trigger a state transition
  - Can have parameters (like an operation)
- Guard → condition
  - Boolean expression to check if the event occurs
  - If the guard is `true`
    1. All activities in the current state are terminated
    2. All `exit` activities are executed
    3. The transition takes place
  - If the guard is `false`, no transition happens (event is **discarded**)
- Activity → effect
  - Sequence of actions executed during the state transition

Transition – types (1/2)

Internal transition

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

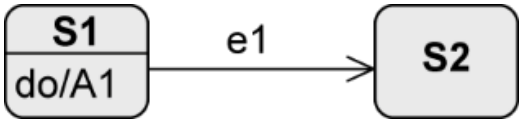
External transition



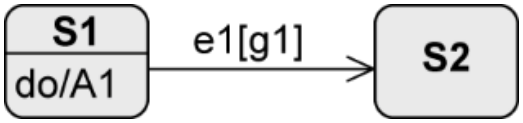
- If **`event1`** occurs
  - Object leaves **`state1`** and **`Activity2`** is executed
  - **`Activity3`** is executed
  - Object (re-)enters **`state1`** and **`Activity1`** is executed

Transition – types (2/2)

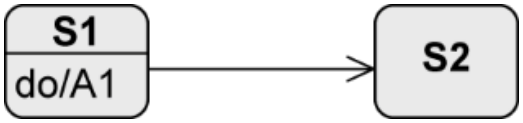
- When do the following transitions take place?



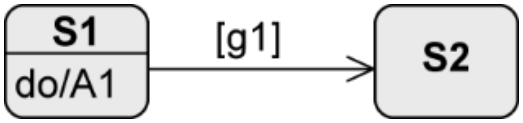
If **`e1`** occurs, **`A1`** is aborted and the object changes to **`S2`**



If **`e1`** occurs and **`g1`** evaluates to `true`, **`A1`** is aborted and the object changes to **`S2`**



As soon as the execution of **`A1`** is finished, a completion event is generated that initiates the transition to **`S2`**

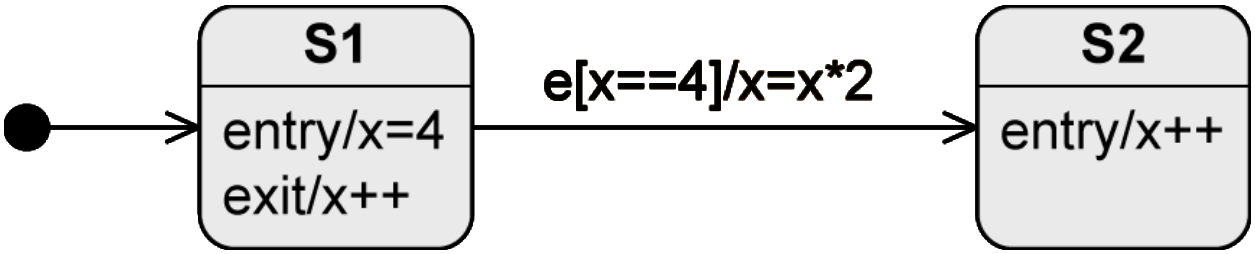


As soon as the execution of **`A1`** is finished, a completion event is generated;
if **`g1`** evaluates to `true`, the transition takes place; if not, this transition never happens

Typically overlooked

Transition – sequence of activity executions

Assume below state machine started and **e** occurred after some time. What is the final value of **x**?



S1 becomes active, **x** is set to the value **4**

e occurs, the guard is checked and evaluates to true

S1 is left, **x** is set to **5**

The transition takes place, **x** is set to **10**

S2 is entered, **x** is set to **11**

Event – types (1/2)

- **Signal event**

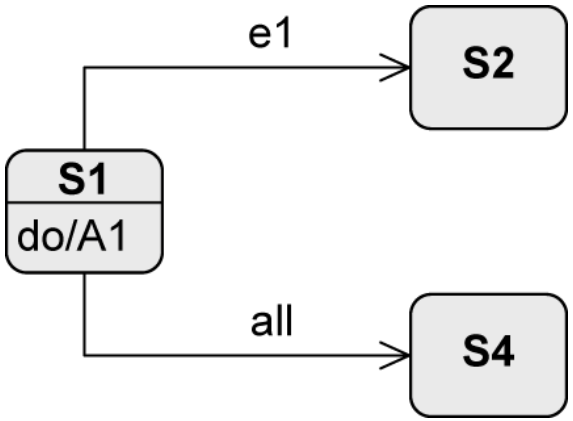
  Receipt of a signal
  - E.g., `rightMouseDown`, `sendSMS(message)`
- **Call event**

  Operation call
  - E.g., `occupy(user, lectureHall)`, `register(exam)`
- **Time event**

  Time-based state transition
  - Relative: based on the time of the occurrence of the event
    - E.g., `after(5 seconds)`
  - Absolute
    - E.g., `when(time==16:00)`
      `when(date==2015-01-01)`

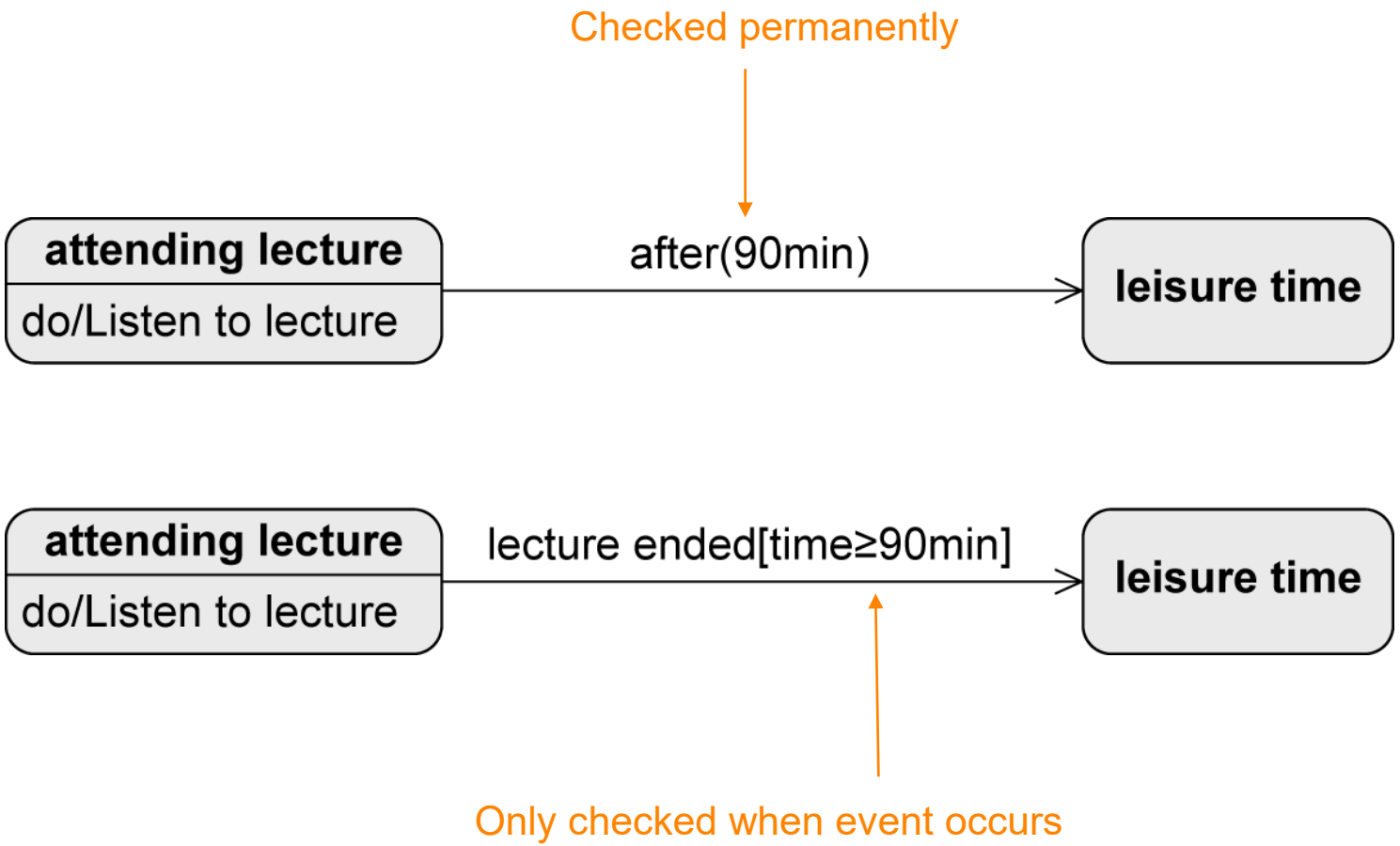
Event – types (2/2)

- **"Any receive" event**
  - Occurs when any event occurs that does not trigger another transition from the active state
  - Keyword **`all`**



- **Completion event**
  - Generated automatically when everything to be done in the current state is completed (it is implicit)
- **Change event**
  - Permanently checking whether a condition becomes true
  - E.g., **`when(x > y)`**

Change event vs. guard



Question: What if the lecture is shorter than 90min?

Initial state 

- "Start" of a state machine diagram
- Pseudo state
  - Transient, i.e., system cannot remain in that state
  - Rather a control structure than a real state
- No incoming edges
- If >1 outgoing edges
  - Guards must be mutually exclusive and cover all possible cases to ensure that exactly one target state is reached
- If initial state becomes active, the object immediately switches to the next state
  - No events allowed on the outgoing edges

Final state and terminate node

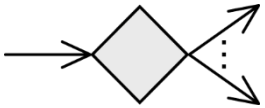
Final state ◉

- Real state
- Marks the end of the sequence of states
- Object can remain in a final state forever

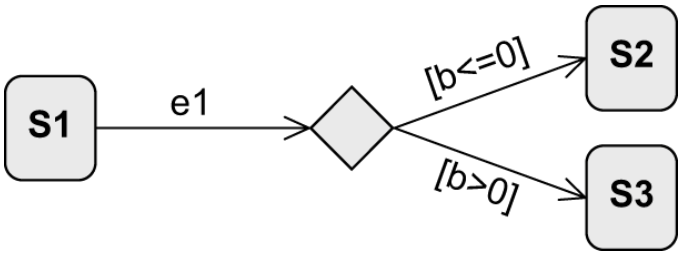
Terminate node ⨯

- Pseudo state
- Terminates the state machine
- The modeled object ceases to exist (= is deleted)

Decision node

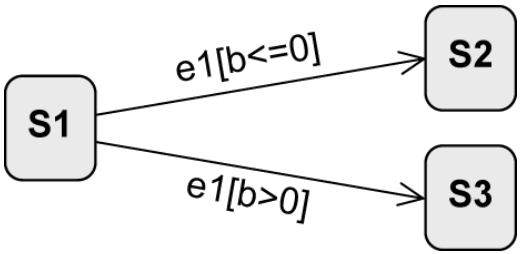


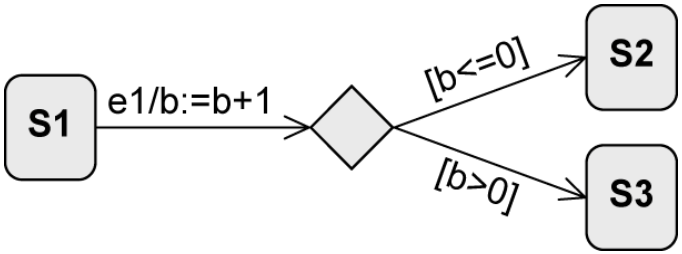
- Pseudo state
- Used to model alternative transitions



equivalent?

=





equivalent?

≠