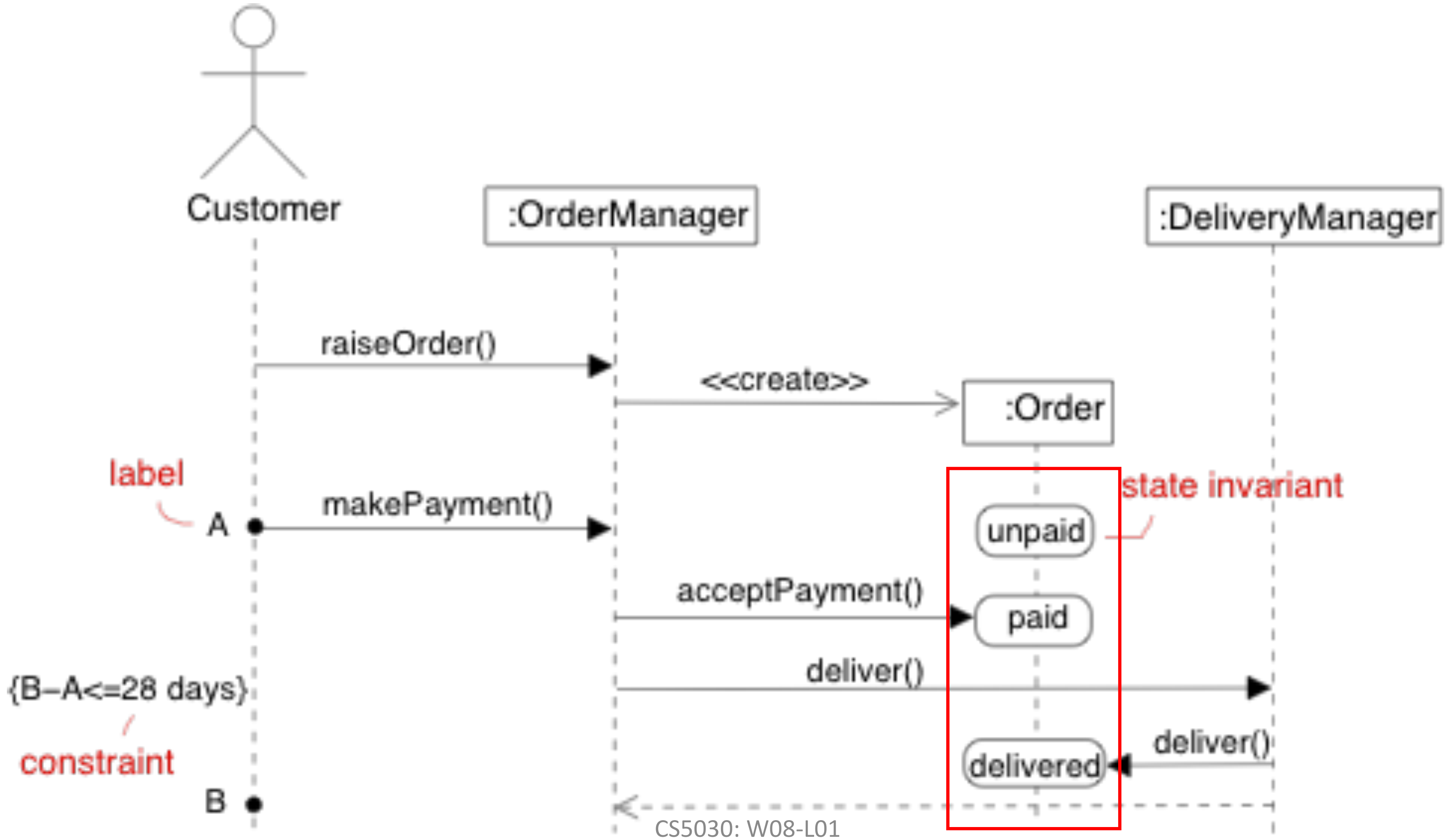
CS5030

Modelling State Transitions

Learning objectives

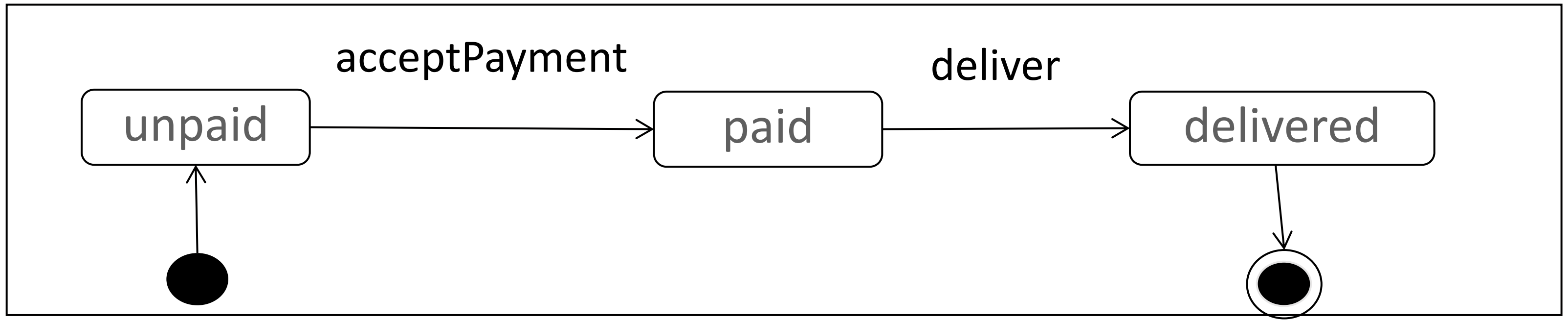
- On completing this lecture and associated reading, you should
  - Understand the purpose of state machines
  - Know the key elements of state machines
  - Be able to create state machines for given scenarios

Previously …



Customer
:OrderManager
:DeliveryManager
raiseOrder()
<<create>>
:Order
label
A
makePayment()
state invariant
unpaid
acceptPayment()
paid
deliver()
{B–A<=28 days}
constraint
deliver()
delivered
B

Next …



Event-driven modelling

- Event-driven modelling shows how a system responds to external and internal events

Assumptions

- A system has a finite number of states
- Events (stimuli) may cause a transition from one state to another

State machine model

- State machine models show the states of an object (or a system) and the events that cause state changes
    - Can model behaviours and protocols

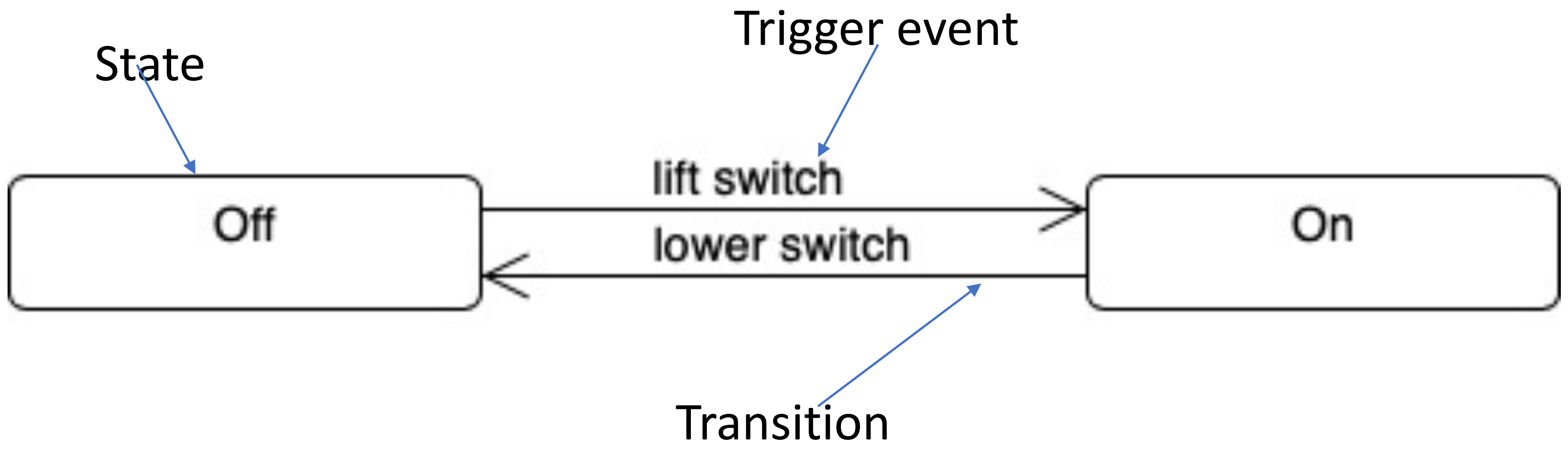
- Used for modelling
    - Real-time and mission-critical systems
    - Devices whose behaviour is defined in terms of state
        - For eg, ATMS
    - Some types of games

State machine

- A state machine models the life cycle of a single object (or a system) as a finite state machine

- Three key elements
  - State
  - Events
  - Transitions

State machine notation (1)



State machine notation (2)

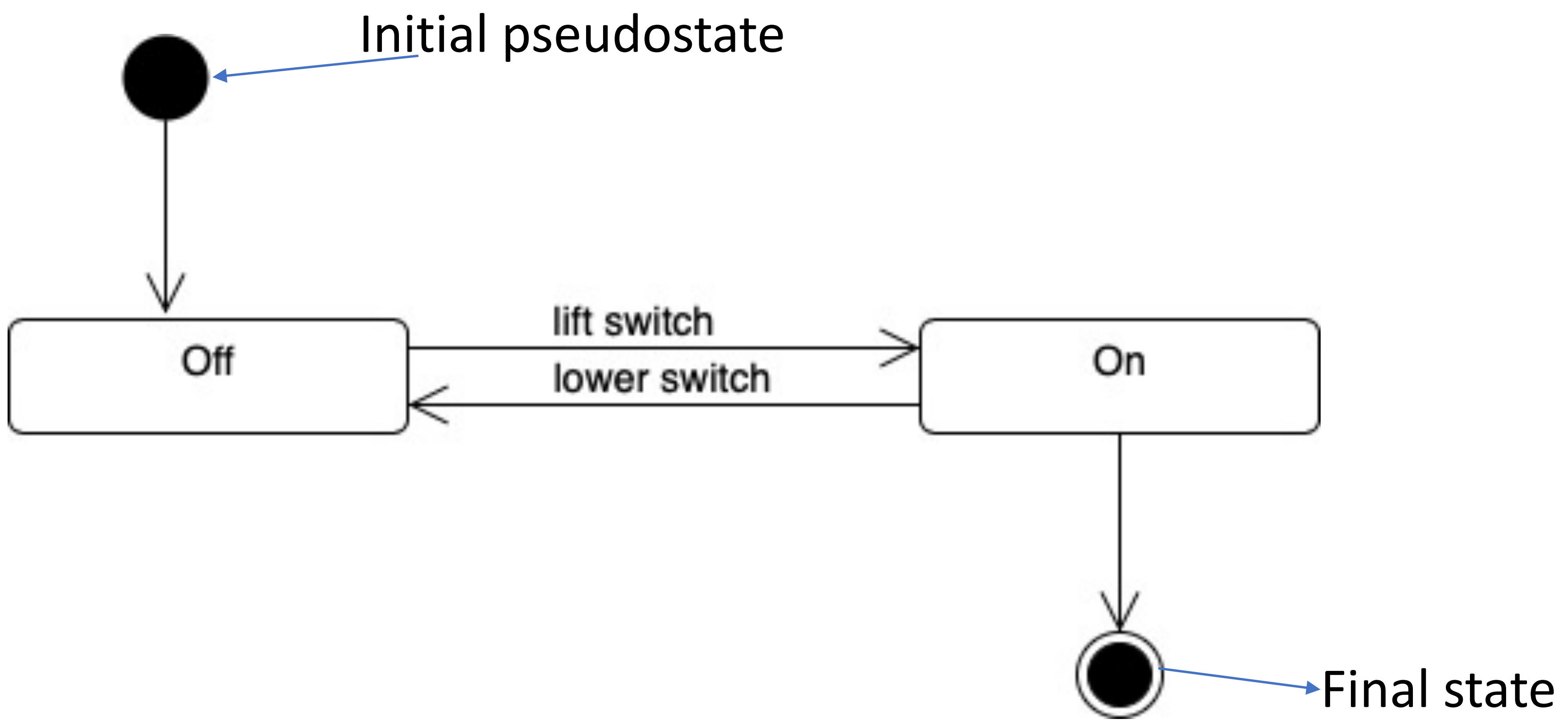


Table view of state machines

- A table view can be helpful as an intermediate step in creating state machines

State / Trigger	Light switch lifted	Light switch lowered
Off	On	-
On	-	Off

- (dash) means transition cannot happen

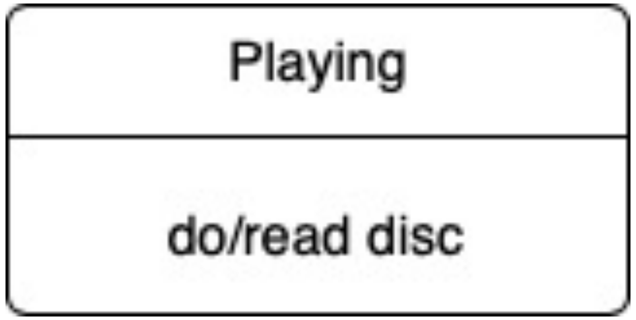
State (1)

- State is a condition of being at a certain time
- The state of an object varies over time, but at a particular point of time it is determined by
  - The object attribute values
  - The relationships it has to other objects
  - The operation it is performing
- Over time, objects send messages to one another, and these messages can cause changes in object state

State (2)

- A state is
  - Active when entered through a transition
  - Inactive when exited through a transition

- ‘Doing’ states are used when an object is doing something
  - This behaviour can be shown as part of the state



Pseudostates

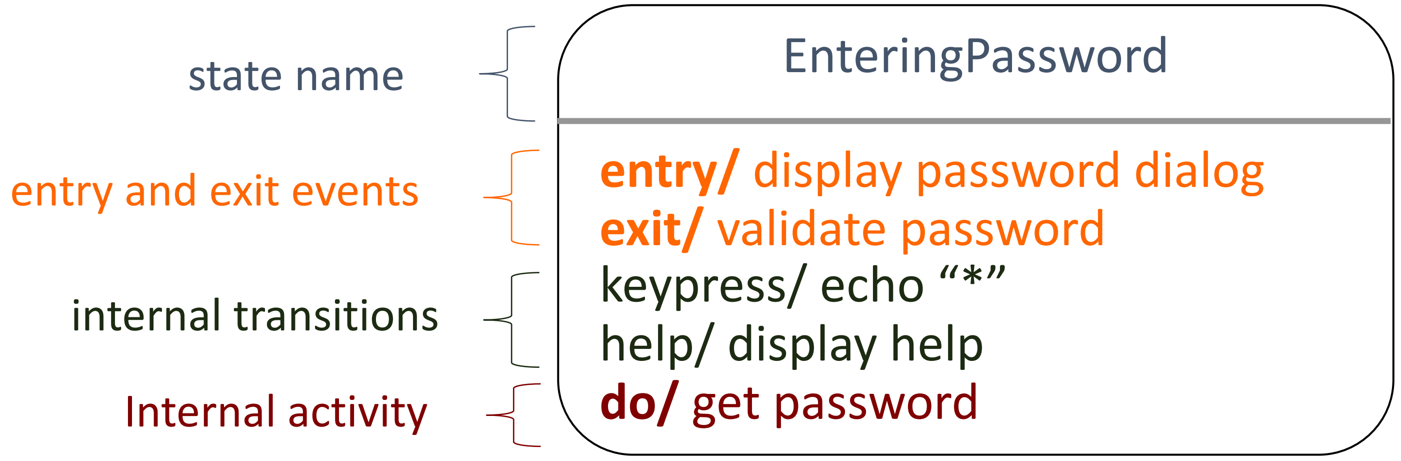
- Special markers that direct the flow in a state machine diagram

- Already seen initial pseudostate

- Others to be seen later

Advanced state behaviour

- Each state in a state machine can show additional details
  - Entry and exit events
  - Internal transitions
  - Internal activities (behaviour)



Transition (1)

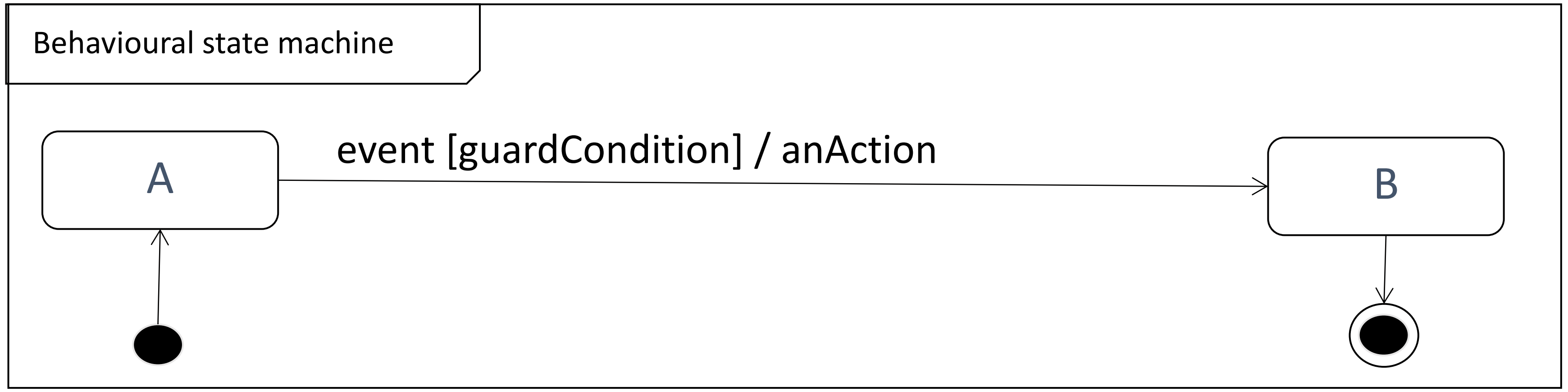
- A transition represents a change in states from a source state to a target state

- Simple transition description
  - Lists trigger

- Complete notation
  - *trigger[guard] / behaviour*
  - All elements are optional

Transition (2)

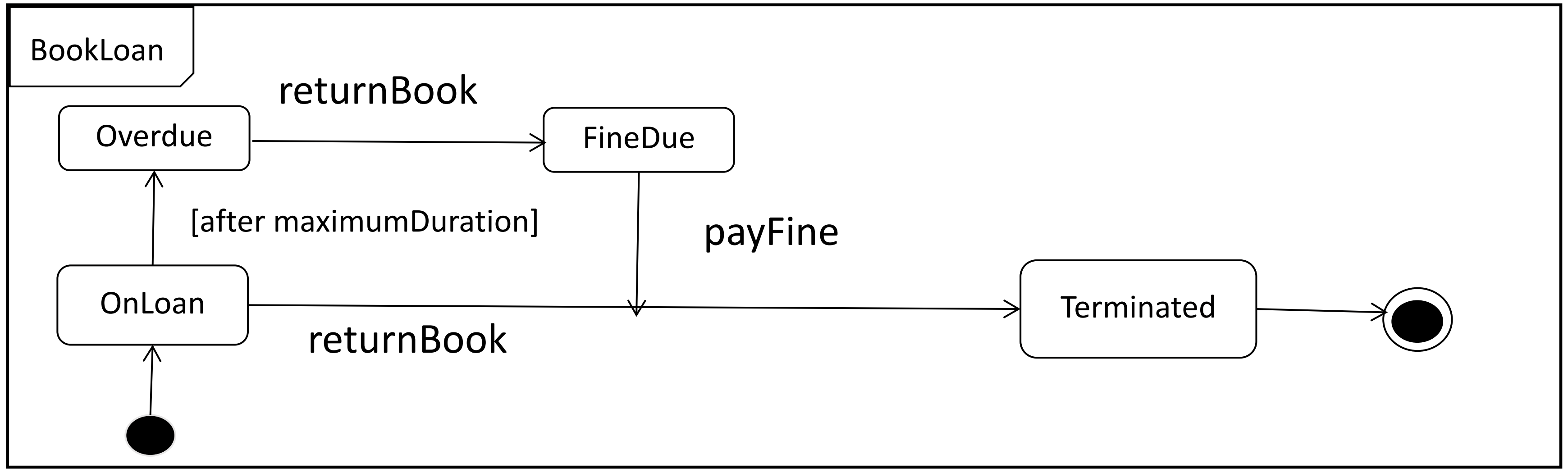
- From state A
  - on (event) if (guardCondition is true) then perform anAction and enter state B



Transition - example

- Example:
  - Assume each object in BookLoan can have four states: onloan, overdue, fineDue, and terminated

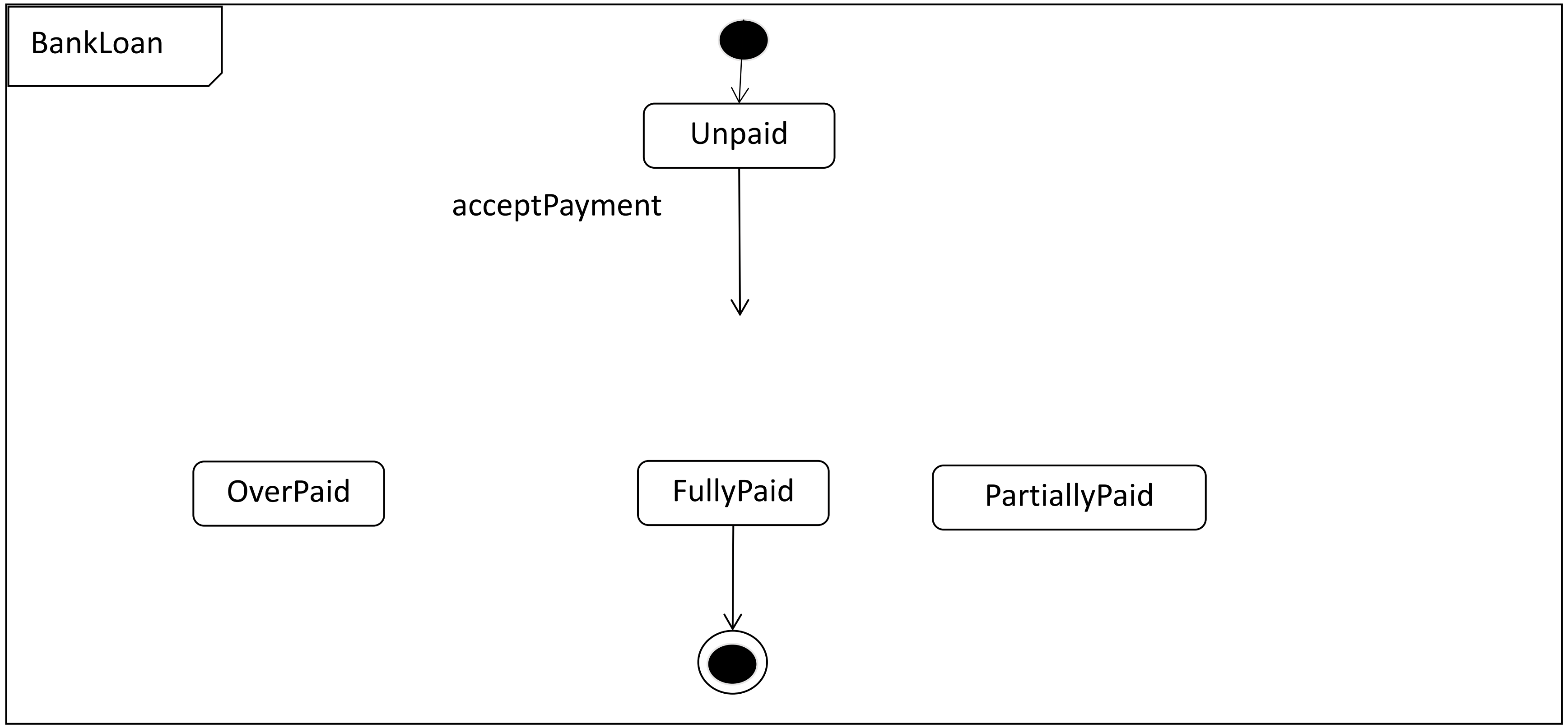
We will refine this diagram later



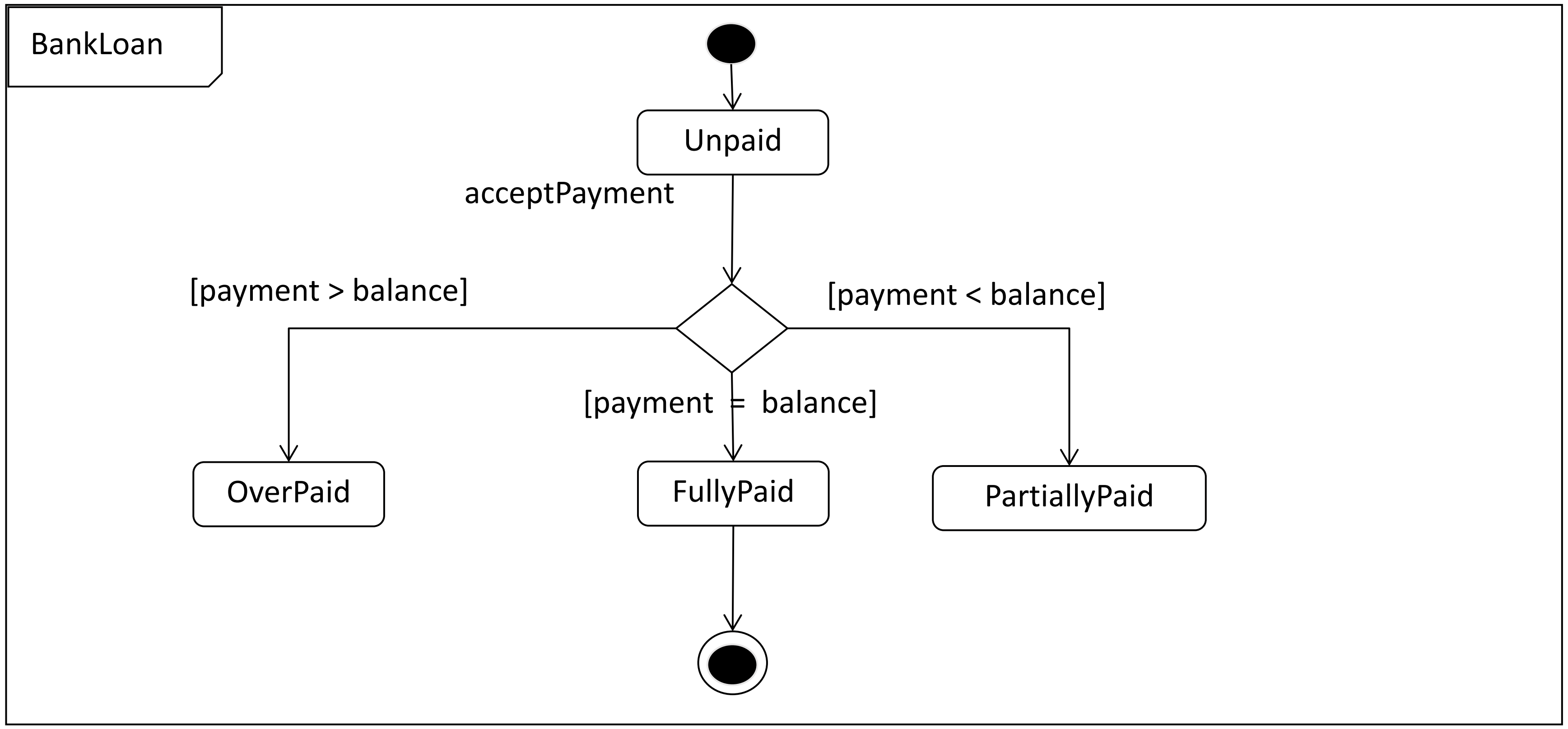
Advanced transitions

- Using pseudostates
- Connecting transitions
  - Transitions can be connected by junction pseudostates
- Branching transitions
  - Choice pseudostate
  - Direct the flow through the state machine according to conditions

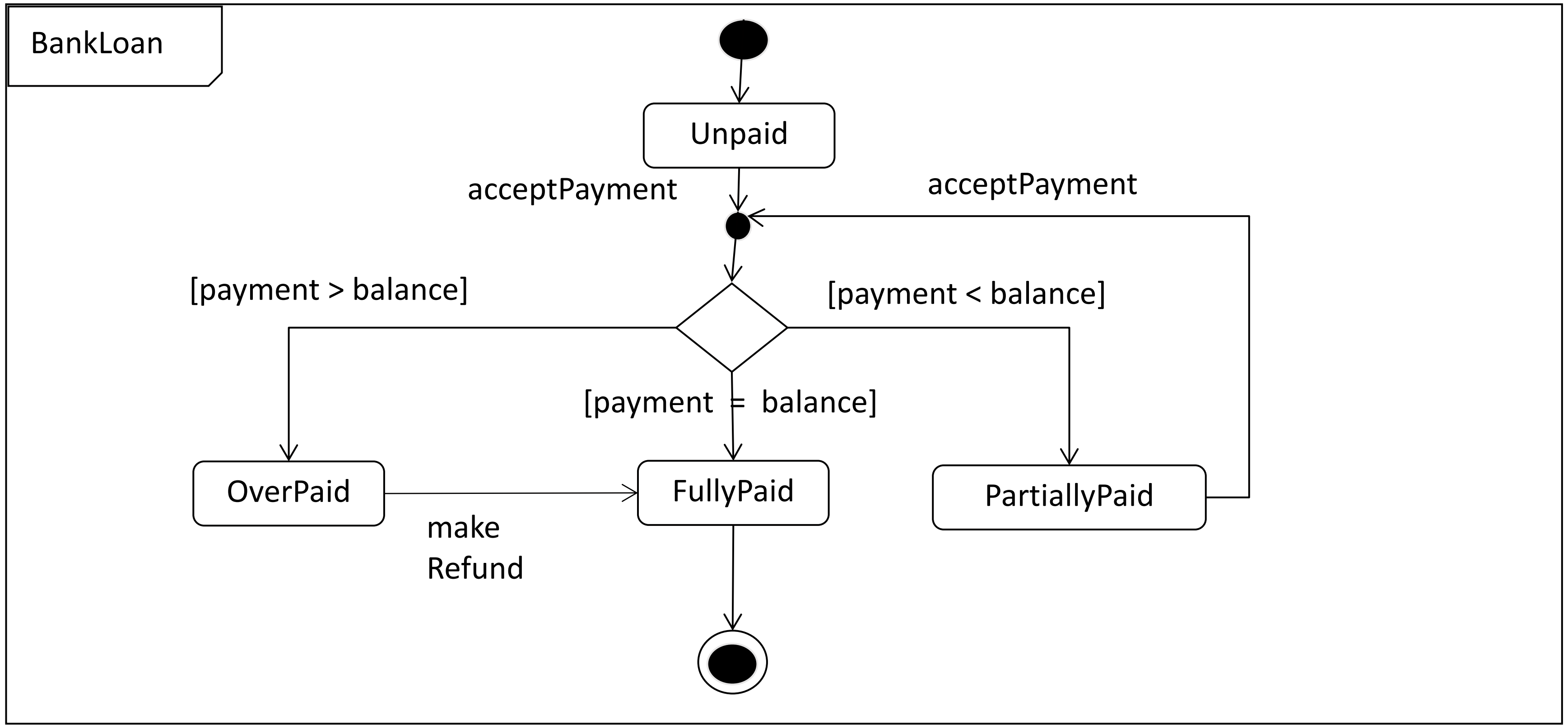
Branching transitions



Branching transitions

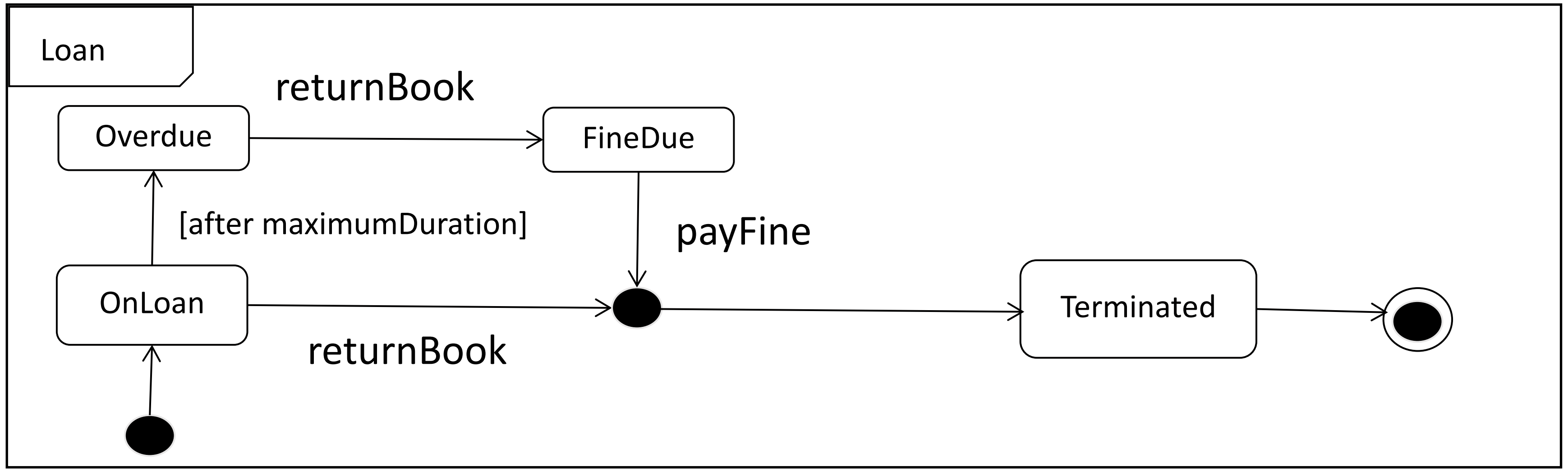


Branching transitions



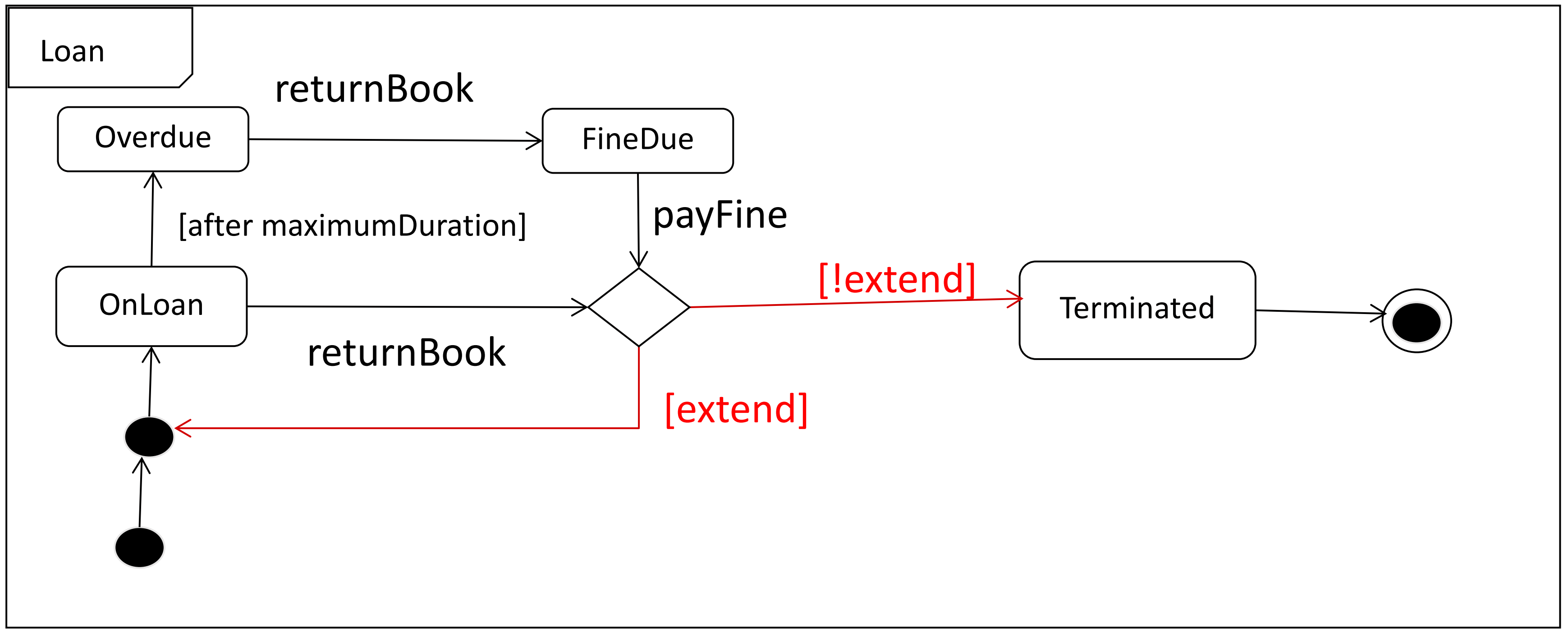
Connecting transitions

- Transitions can be connected by junction pseudostates



Connecting transitions

- If there exist more than one output transitions, each output transition must be protected by a mutually exclusive guard condition



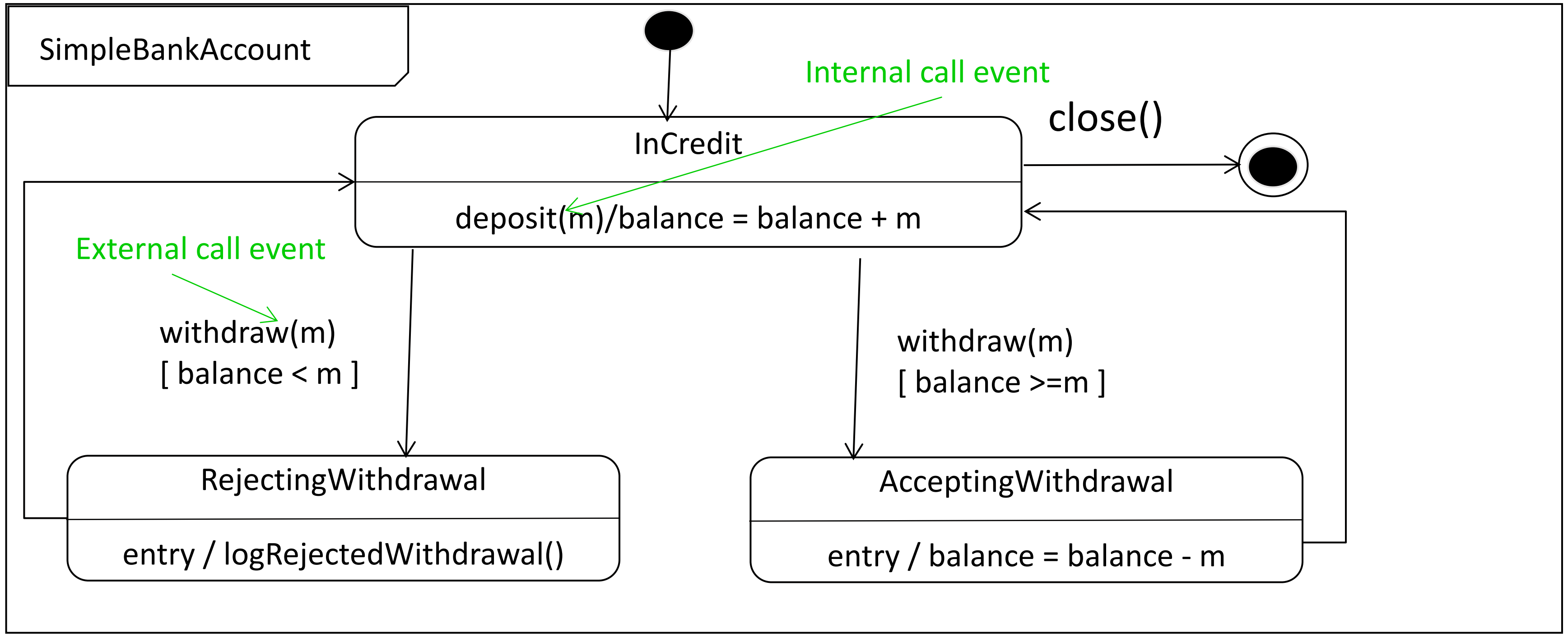
Event

- An event is defined as “the specification of a noteworthy occurrence that has location in time and space”

- Four types of events
  - Call event
    - Arrival of a call to an operation
  - Signal event
    - Arrival of an asynchronous message or signal
  - Change event
    - When a specified condition has been met
  - Time event
    - After a specified time has elapsed

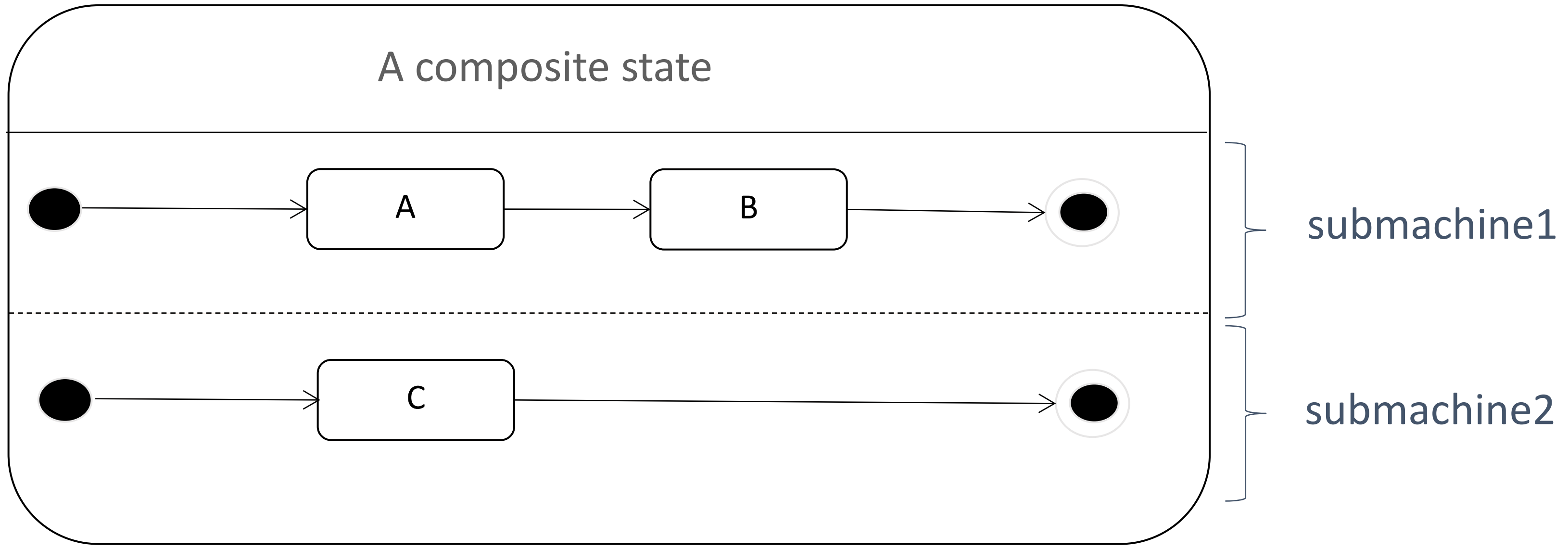
Call event

- A call event is a request for a specific operation to be invoked on an instance of the context class
- Receipt of a call event is a trigger for the operation to execute

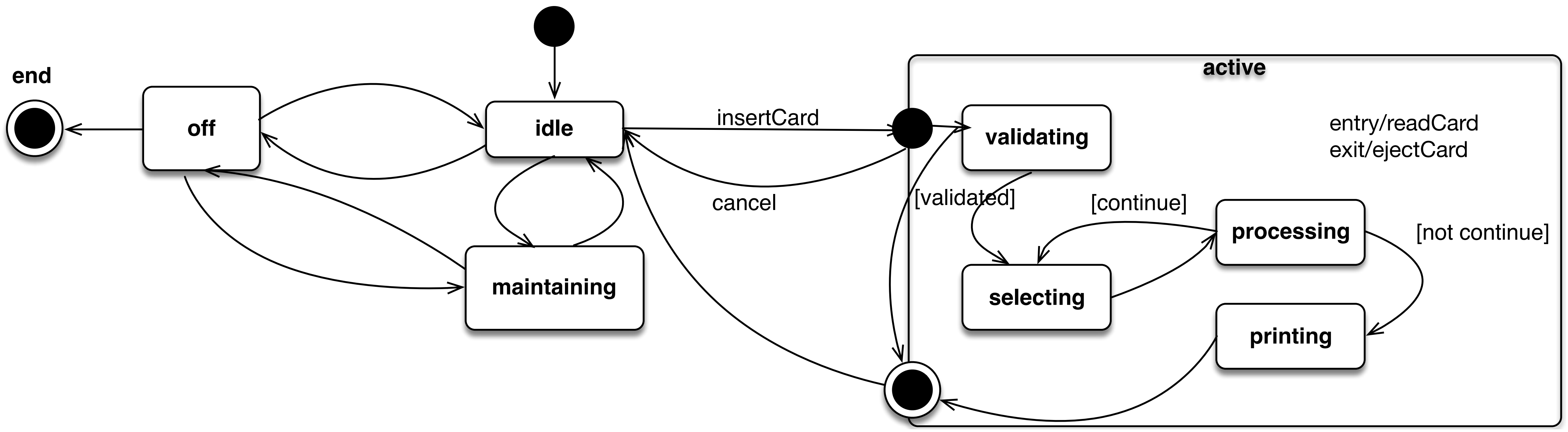


Composite state

- A composite state is a state that contains nested states
- The nested states are organised into one or more state machine, called submachines



Composite state example - printer



Creating state machines

- Represent the sequence of states that describes the normal behaviour of an instance, together with the transitions that are associated with it
- Progressively add the transitions that correspond to the "alternative" or error sequences
- Complete the actions on the transitions and activities on the states
- Structure the diagram into substates and use advanced notation (entry, exit, and so on) if it becomes too complex

Testing state machines

- Normally perform a manual walkthrough and see how the state machine reacts under different circumstances

- The best way to test state machines is to simulate them
  - Execute the state machine to see how it behaves
  - Such tools may be excessive for usual business modelling, but essential for real-time embedded systems where objects may have complex behaviours and life cycles

Example: actions in a microwave

- Assume a simple microwave has a switch to select full or half power, a numeric keypad to input the cooking time, a start/stop button, and an alphanumeric display
  - Example from Sommerville, 2016

- The sequence of actions includes:
  - Select the power level (either half power or full power)
  - Input the cooking time using a numeric keypad
  - Press Start and the food is cooked for the given time

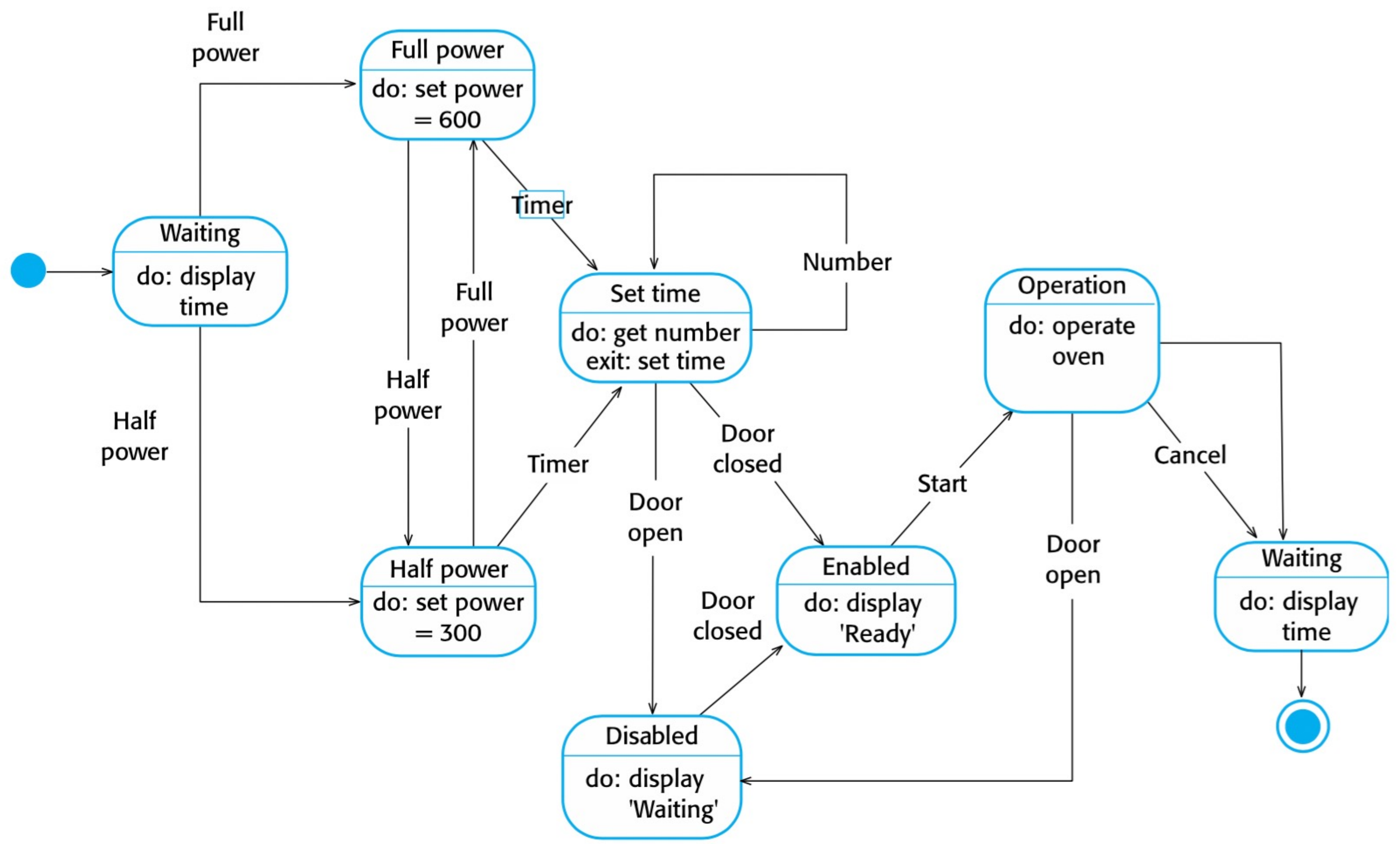
Example: stimuli for microwave

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

Example: stimuli for microwave

Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

Example: state machine



[Sommerville, 2016]

Key points

- State machine diagrams allow the behaviour of objects and systems to be modelled in terms of states and transitions between states
- This behaviour can be in response to internal and external events
- As always, UML allows different levels of detail to be captured as part of these diagrams