CHAPTER 4:
UML DYNAMIC MODEL

Software Engineering

Computer Science School

DSIC-UPV

Contents

Introduction

- The behavioral model consists of:
  - Interaction Diagrams: Communication within the system.
  - State Transition Diagrams (STD): To describe lifecycles of objects.

Interaction

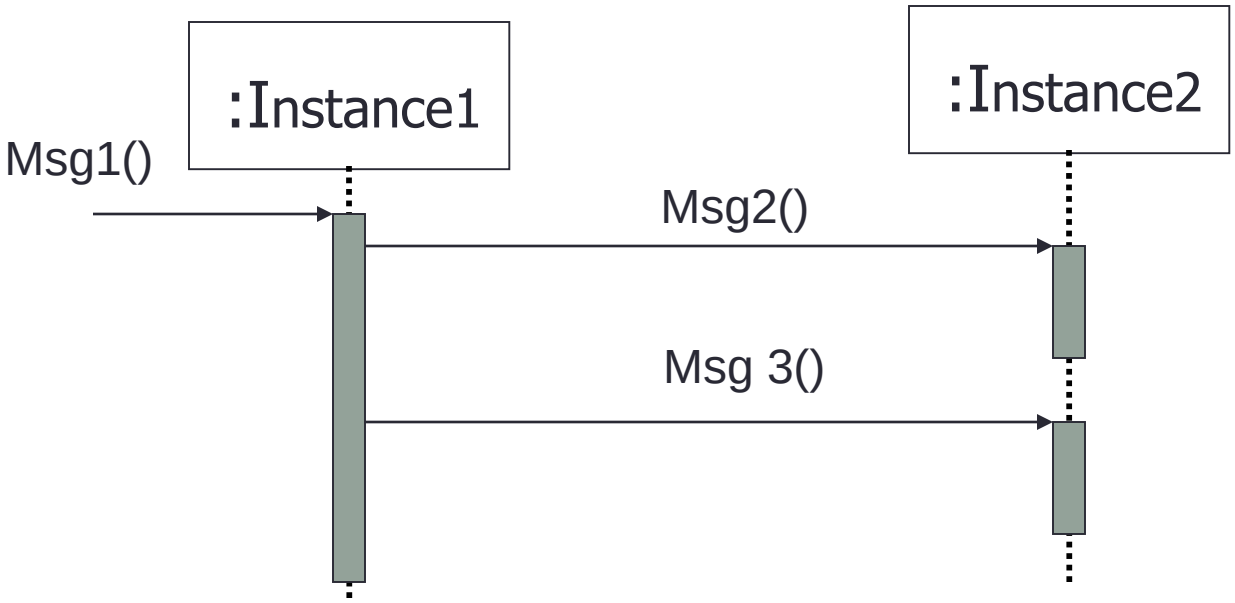
- Objects interact to perform collectivelly the services offered by applications. Interaction diagrams show how objects communicate in an interaction.
- There are two types of interaction diagrams: Sequence diagrams and collaboration diagrams.

Sequence Diagram

- It shows the sequence of messages between objects in a concrete scenario
- Each object is represented with a vertical line
- Time evolves downwards
- The return of a message is optional, it can be used to show the return of values.
- Activation Box: it shows the activation of the methods in the execution stack

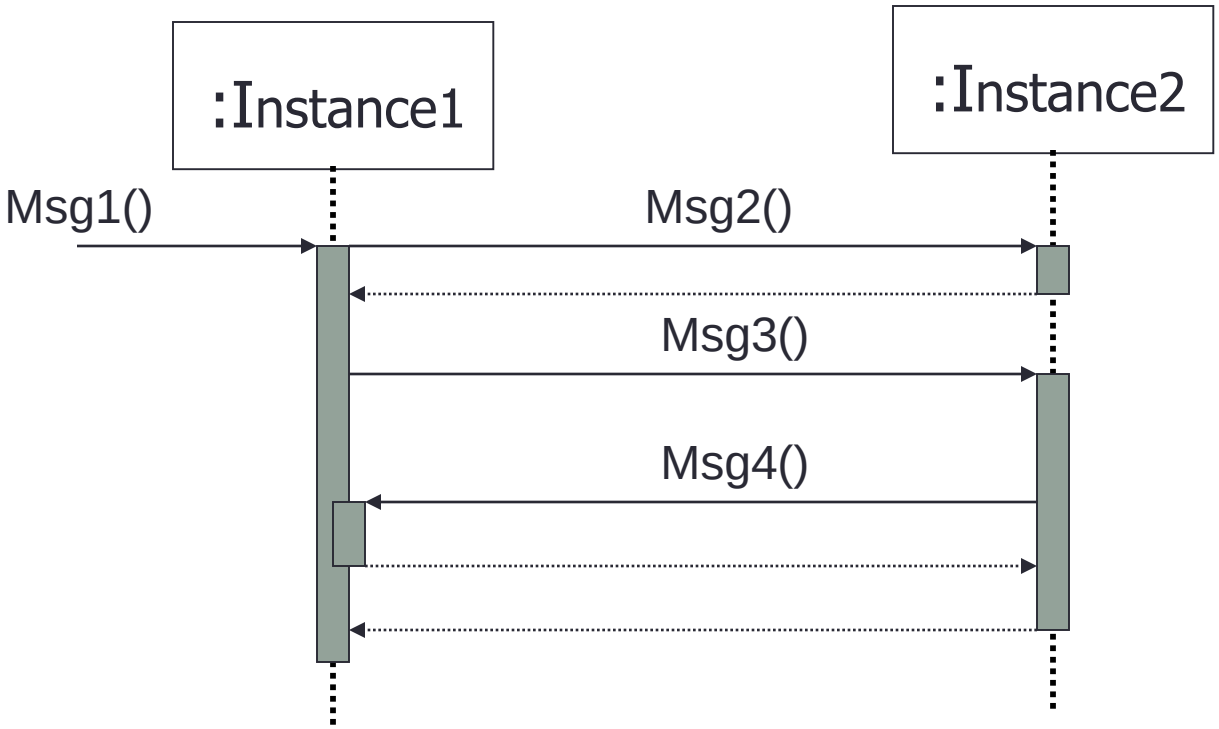
Sequence Diagram

- Communication between objects



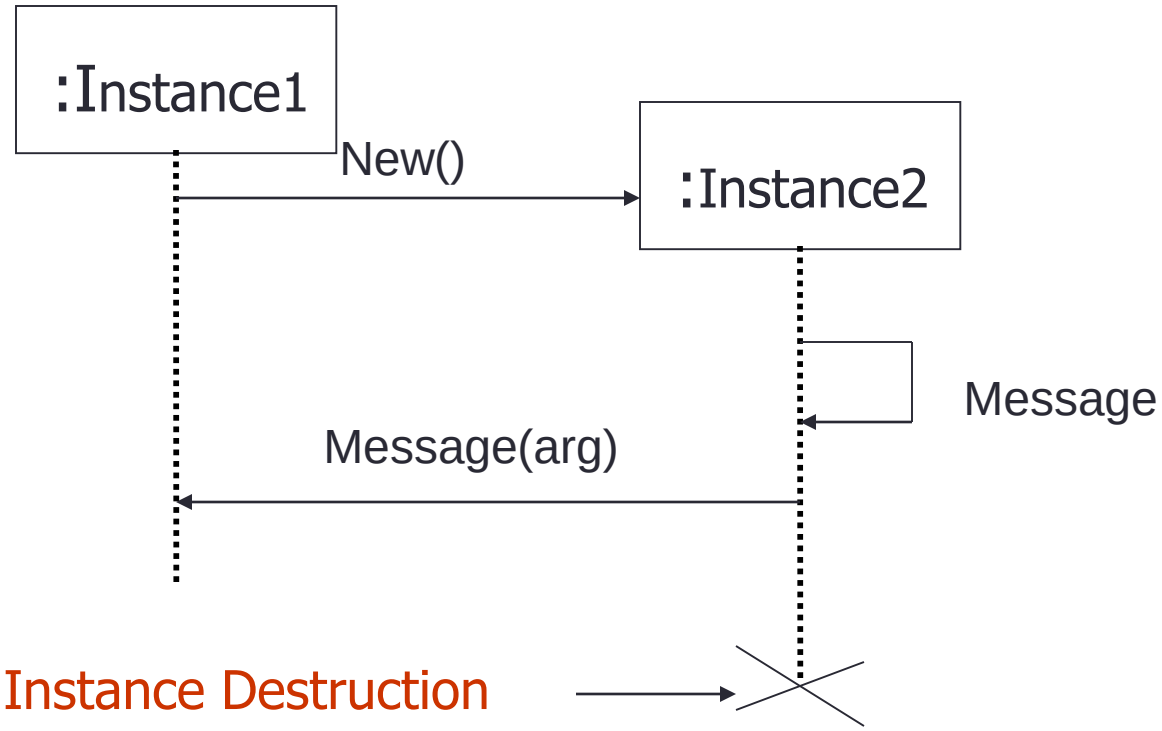
Sequence Diagram

- Indicating the return of an invocation



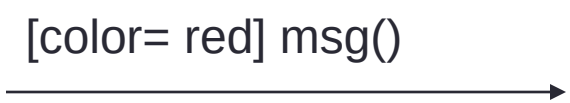
Sequence Diagram

- Creation and destruction

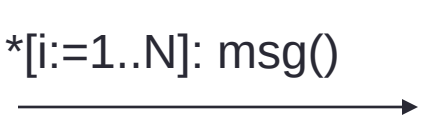


Conditions and iteration

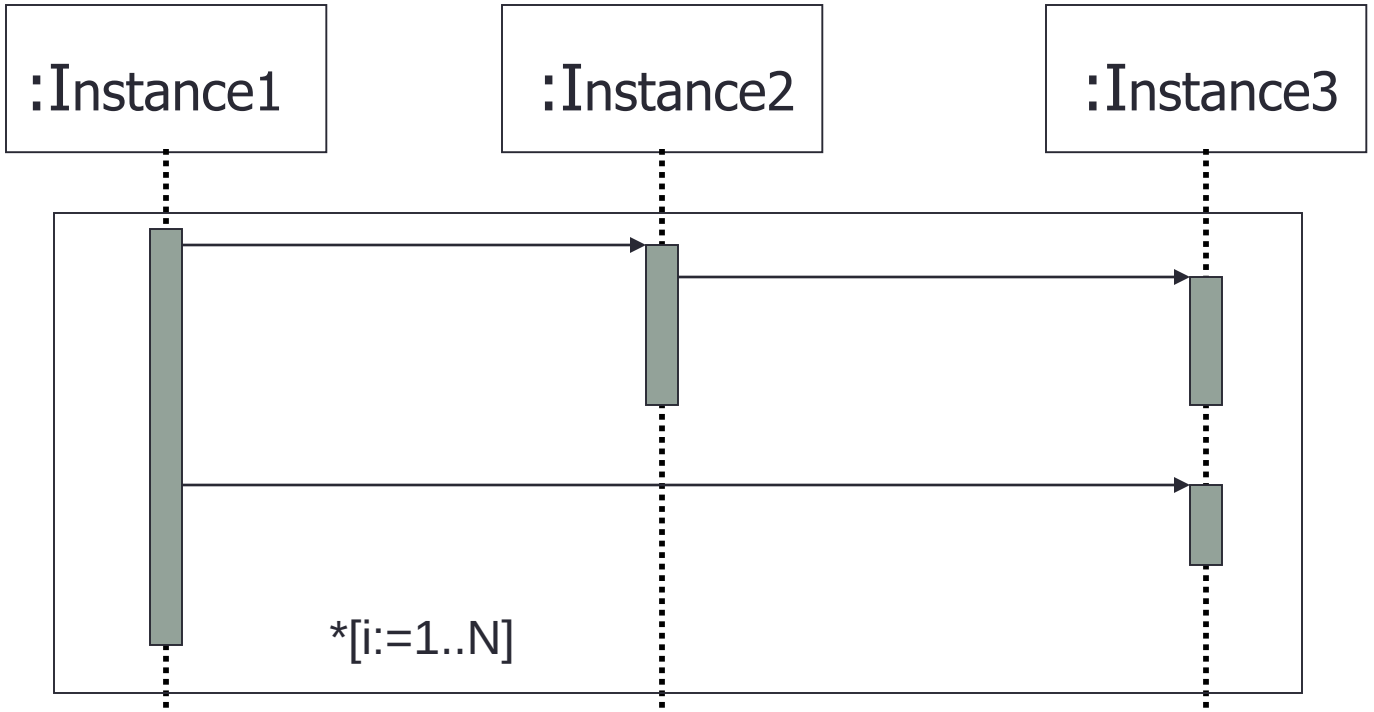
- For simple messages:
  - Condition:



  - Iteration:



Iteration for groups of messages

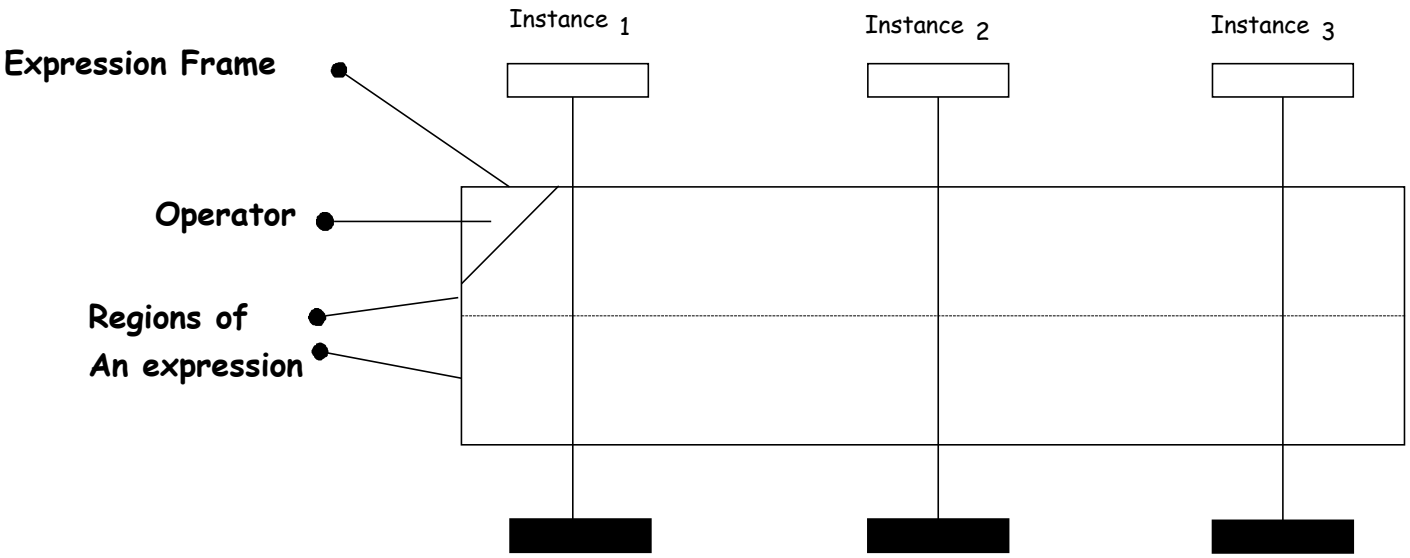


Diagrams UML 2.0

- UML 2.0 sequence diagrams use the notation of MSC (Message Sequence Charts).
- Graphical notation for:
  - Conditions.
  - Loops.
  - Alternatives.
  - Optional parts.
  - Inclusions of diagrams.

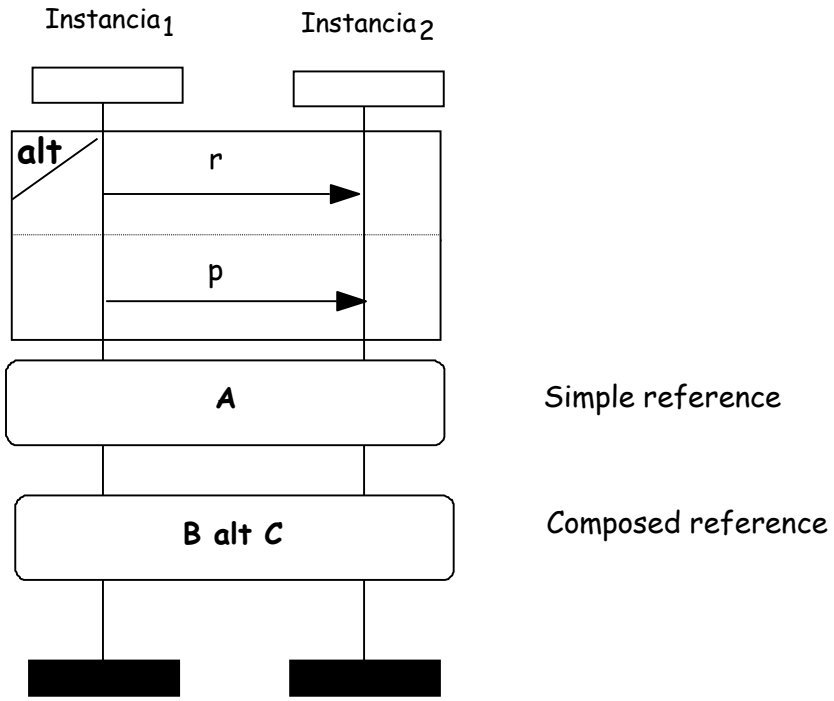
UML 2.0: Inline expressions

- Alternative (alt).
- Parallel composition (par).
- Iteration (loop).
- Optional composition (opt).
- Break (break).

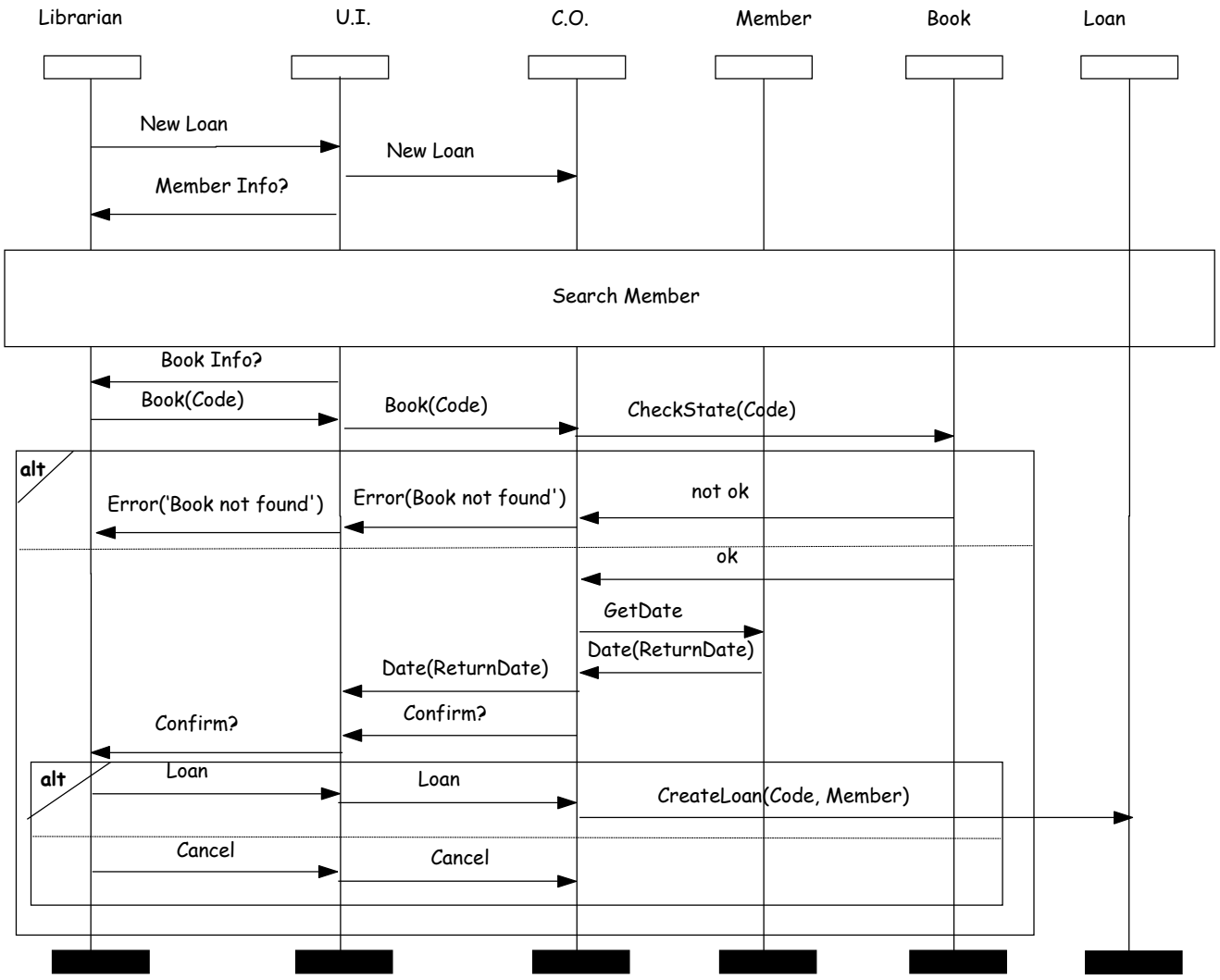


UML 2.0: References to diagrams

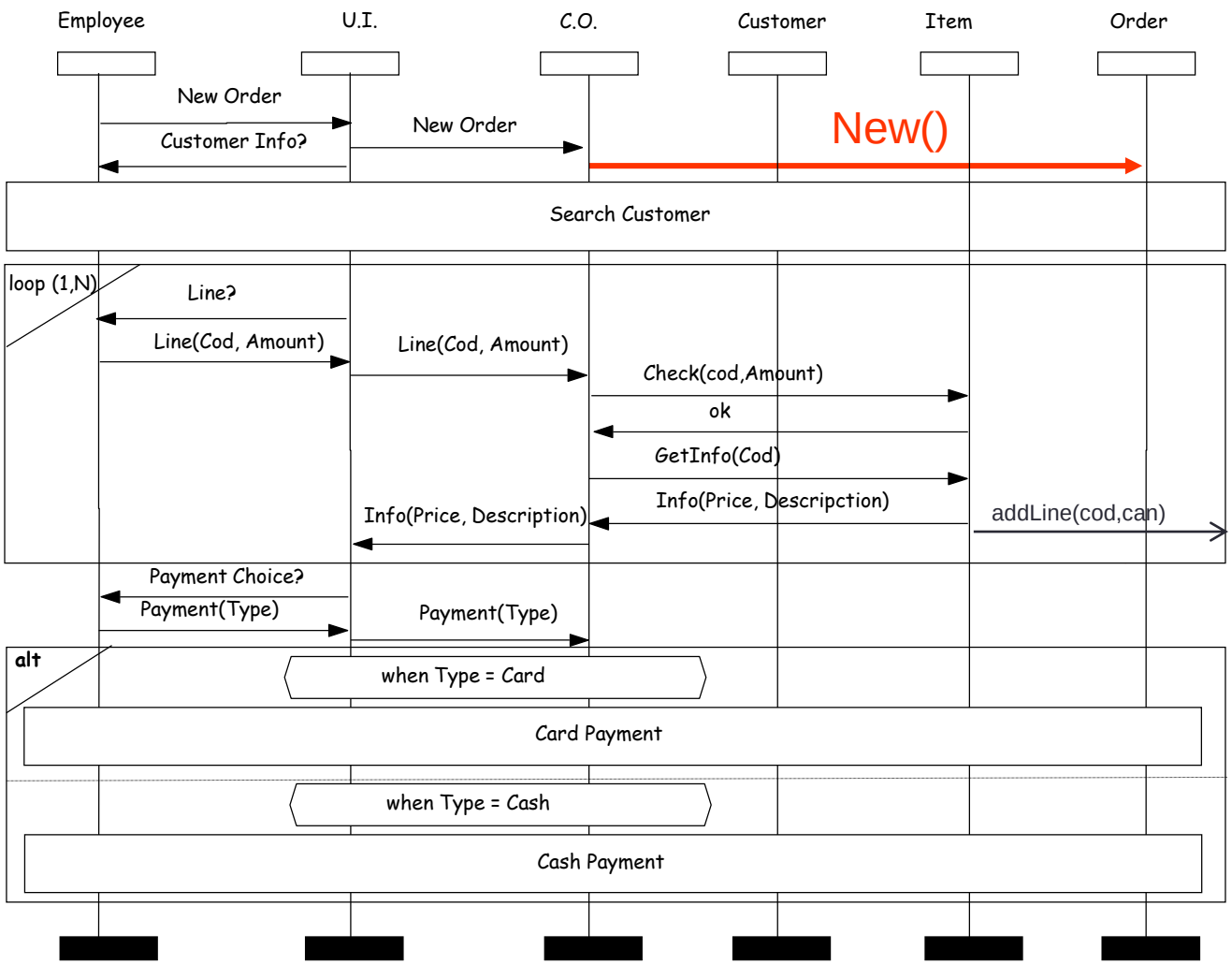
- Expressions in an MSC that allow references to another diagram:
  - Simples (Flat references)
  - Composed (Flat references + operators alt, par, loop, etc.)



UML 2.0: Example

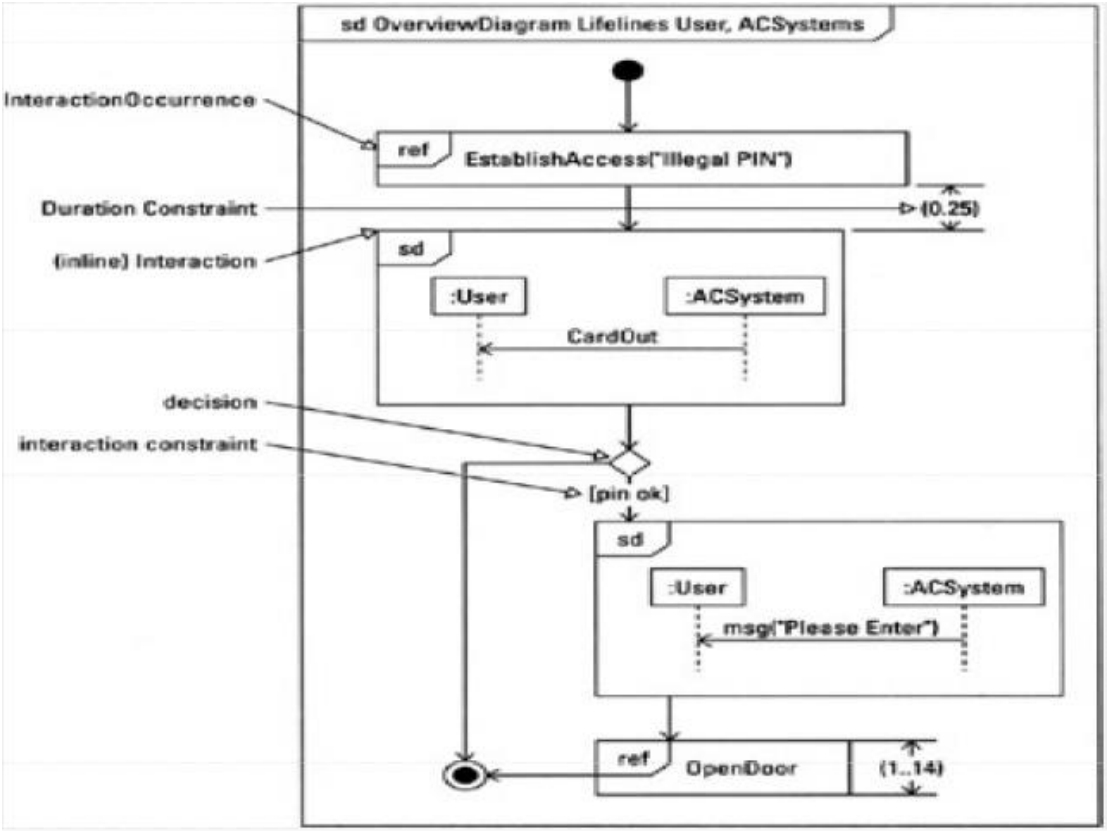


UML 2.0: Example



General Interaction Diagrams

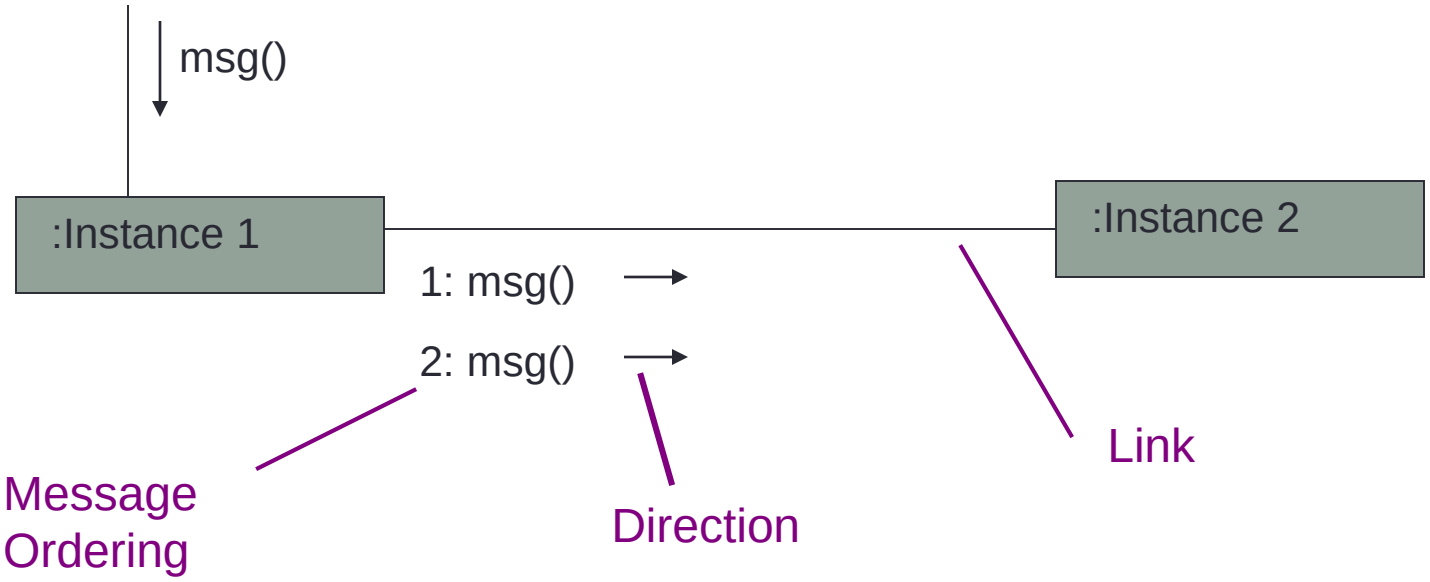
- Include several scenarios or execution alternatives



Collaboration Diagrams

- They represent the same information as in sequence diagrams:
  - Both are exchangable.
- In collaboration diagrams the control flow is not so well described.

Collaboration Diagrams: Notation



Collaboration Diagrams

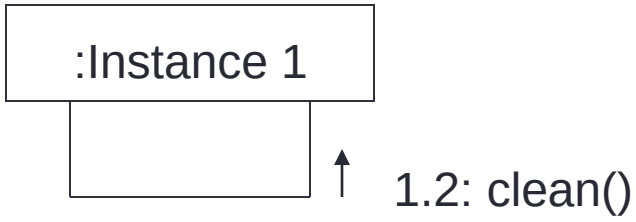
- The links show the exchange of messages between objects:
  - Messages can be exchanged in both directions.
  - Several messages may be exchanged in the same direction.
- Notation for messages:
  - value:= message(paramet:type):returntype

Collaboration Diagrams

- Types information can be removed if it is not important.
  - Description:= GetDescrProduct(id)
  - Description:= GetDescrProduct(id:ItemID)
  - Description:= GetDescrProduct(id:ItemID):DescrProduct

Collaboration Diagrams

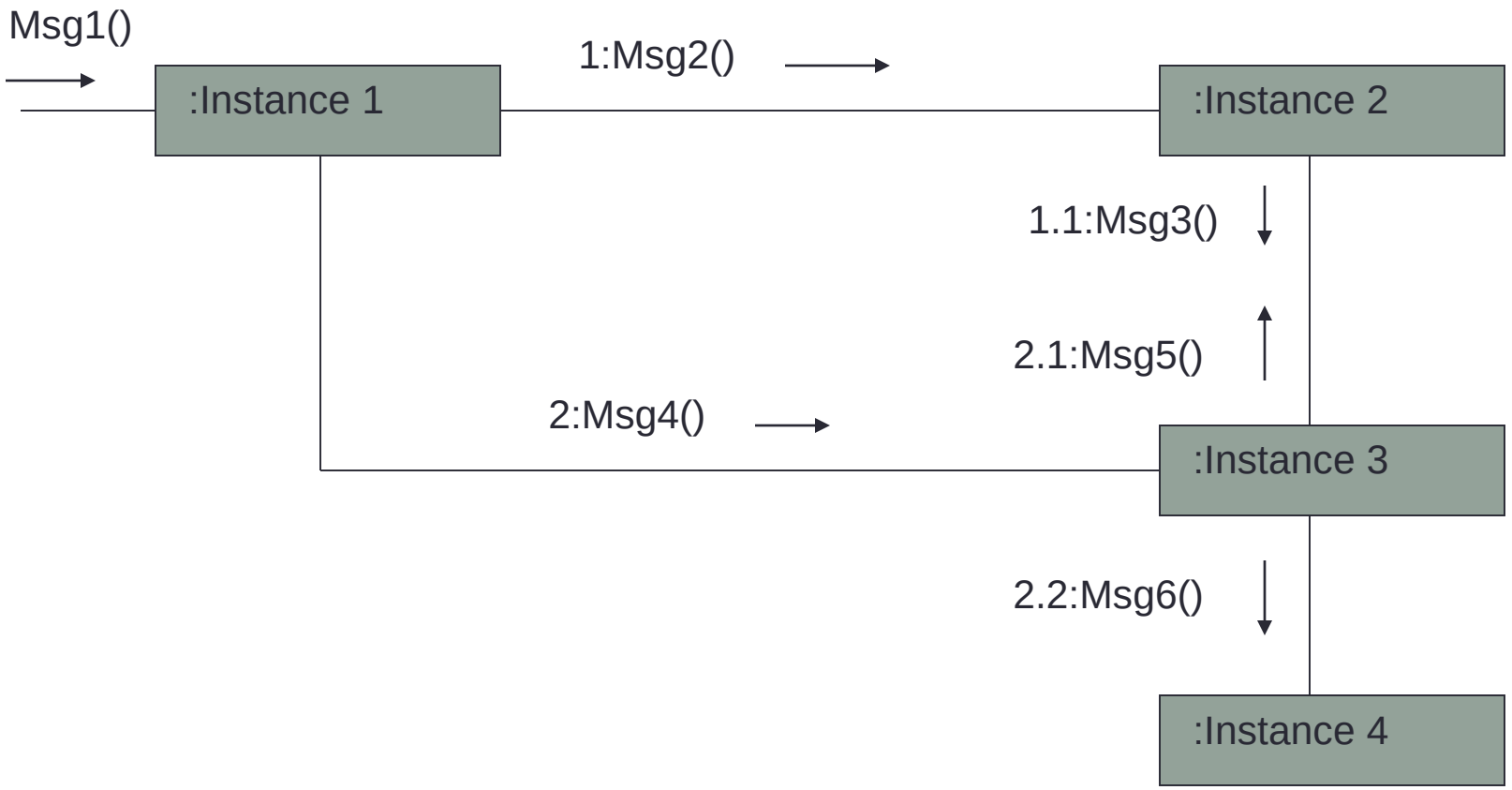
- The arrow indicates the direction of the message
- The numbering sequence describes the ordering
  - 1, 2.1, 3.4.1
- Messages may also target the current instance (self,this).



Collaboration Diagrams : Messages numbering

- The first message is not numbered
  - It represents an action/event that triggers the collaboration
- I a message m is received with numbering x.y.z, then the messages that are sent during the processing of m are numbered with: x.y.z.1, x.y.z.2, etc.

Collaboration Diagrams : Messages numbering



State Transition Diagrams

- A STD relates events and states.
- They help to identify the changes that objects exhibit during their lifetime and they show the sequences of operations produced in response to one or more events.
- A state change produced by an event is called **transition**.

State Transition Diagrams

- All STDs must be deterministic.

- A STD, as it occurs with classes, defines a pattern; it describes a sequence, eventually infinite, of events.

Events & States

- The stimuli sent from an object to another object are called **events.** They represent flows of information between objects or between actors and the system.

- An **event** is something that occurs in the environment of the system and has no duration.

Events & States

- Some events are just simple signals to indicate that something has occurred but most of them ahve attributes that indicate some transferred information.

Mouse button clicked (button, position)
string inserted(text)
Disk unit not ready
Aborted transaction
Timeout.
After(20 seconds).

Examples of events and attributes

Events & States

- The response to an event depends on the state of the object that receives it. This response may be a change of state and/or the emission of another event.

- A state is an abstraction of the values of the atributes of an object. Different sets of values are grouped together to form a state.

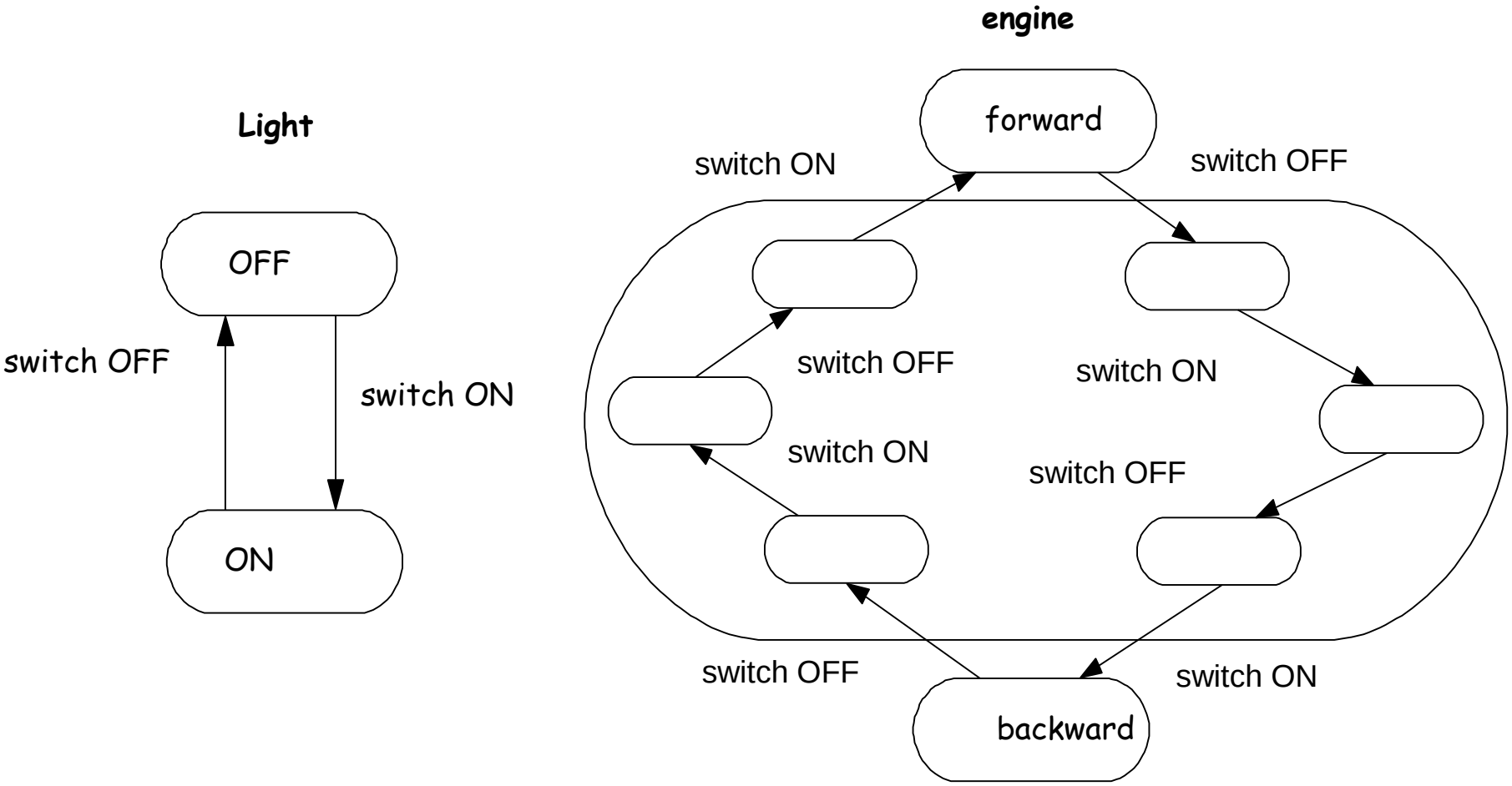
Events & States

Object type: Door
  attributes: OpeningAngle, colour, etc
States:
  closed: 0 = OpeningAngle
  open: 45<= OpeningAngle <=180.
  half-closed: 0< OpeningAngle <45.

Object type: Bank Account
  attributes: balance, creation date, etc.
States:
  ok 0<=balance
  broken: balance<0

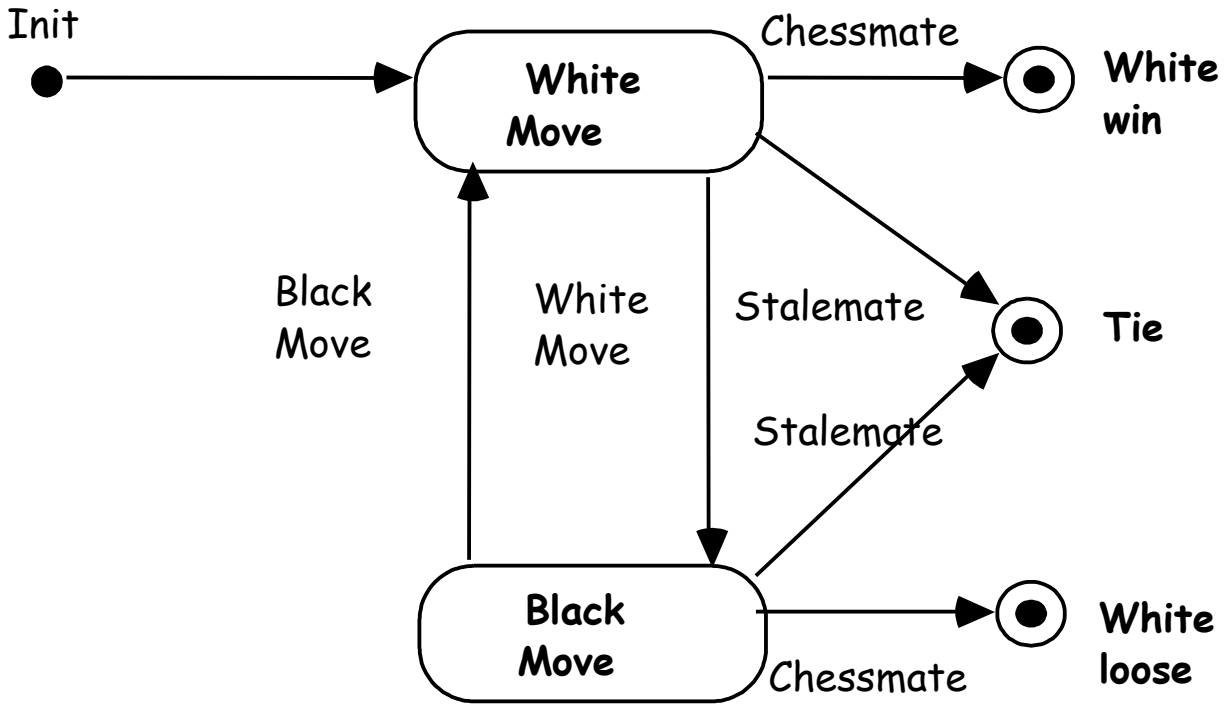
Relevant states

State Transition Diagrams



State Transition Diagrams

Diagrams with final states represent objects with finite lives. The initial state is reached when an object is created; entering a final state results in the object being destroyed.



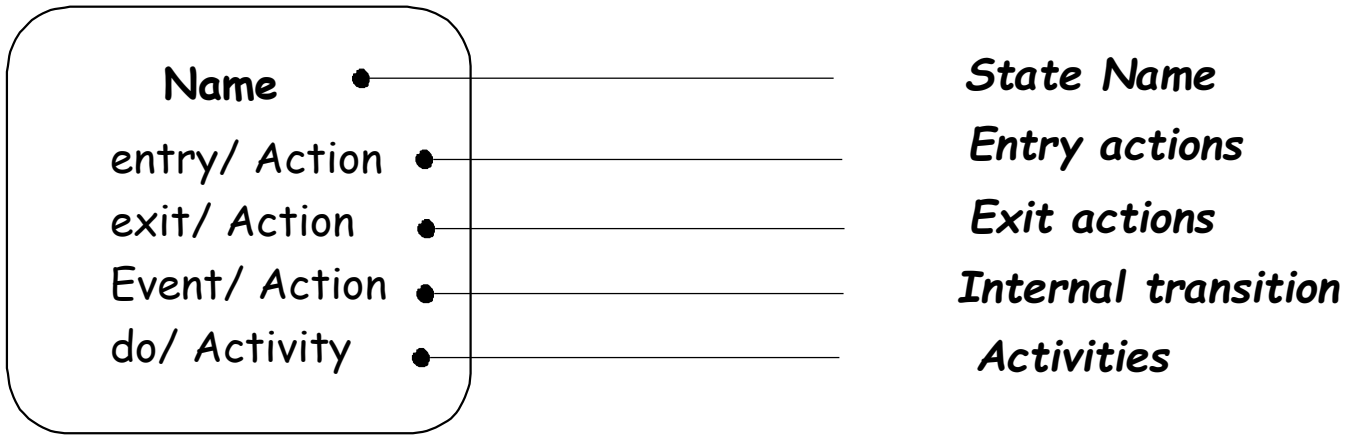
Chess game state transition diagram

Components of a state and a transition

- A state consists of:
  - Name: This is an optional feature.
  - Actions (Entry/Exit) : Operations that are executed when entering/exiting a state.
  - Activities: Operations that are continuously executed in the state.

Components of a state and a transition

- Internal transitions: Transitions that occur without changing to another state.

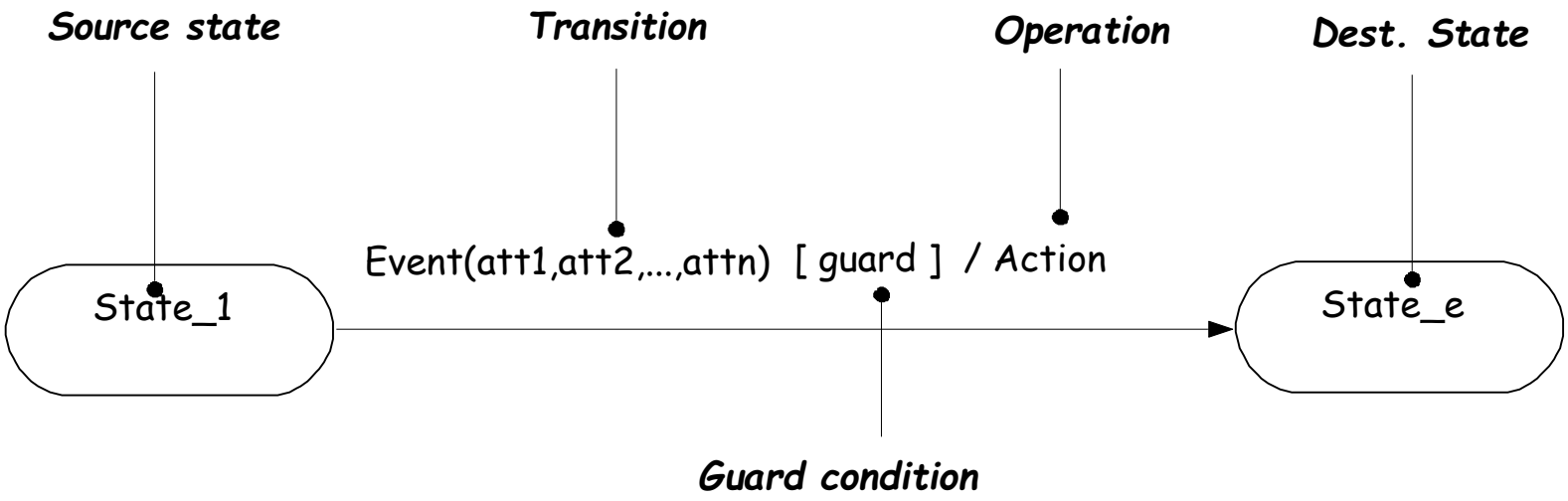


Components of a state and a transition

- A transition has the following components:
  - Source state: State affected by a transition.
  - Triggering event: Event that, if received, causes the state change.
  - Guard condition: Boolean expression that enables state transitions. If an event is received and the guard condition holds then the transition takes place.
  - Action: Operation that may act on the object associated to the diagram or on other objects.

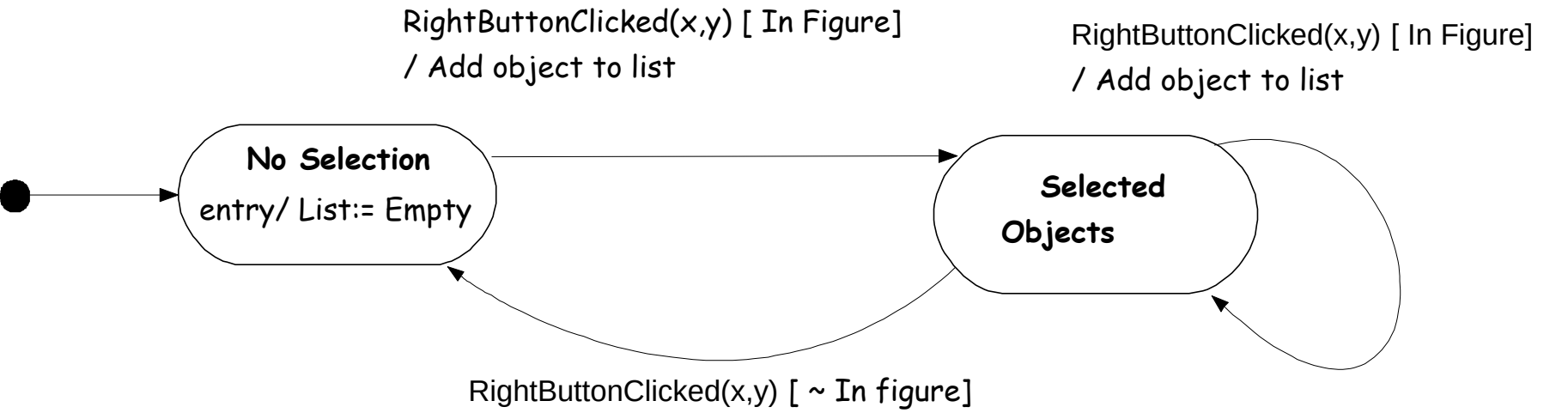
Components of a state and a transition

- Destination state: State that is reached when the transition is completed.



Guard conditions and actions in transitions

- Guard conditions are boolean fucntions that are applied on objects to enable transitions.
- Actions in a transition are operations that are executed when the transition is processed.



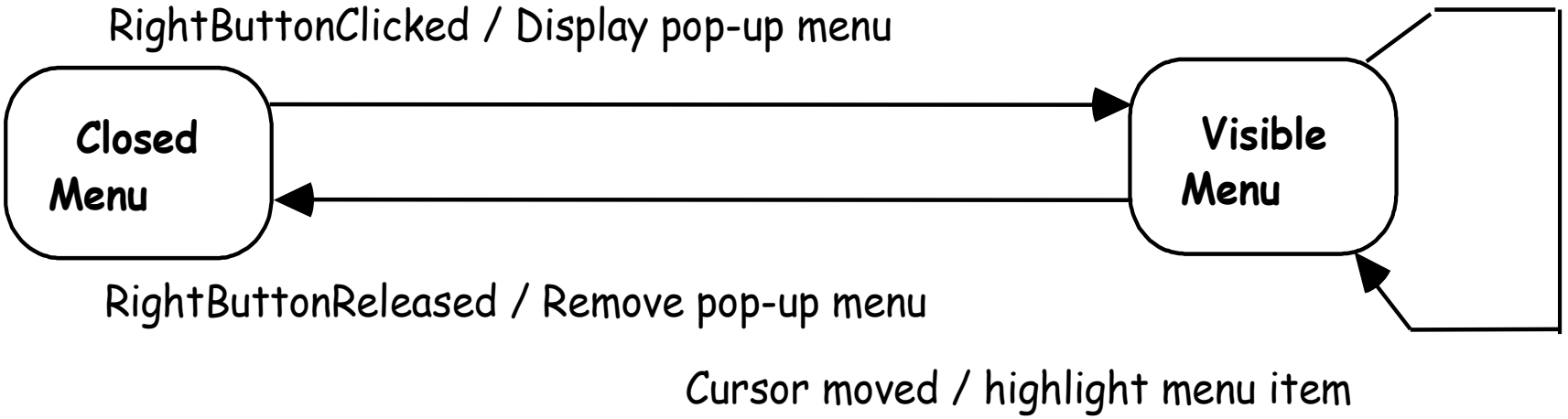
Operations in Diagrams

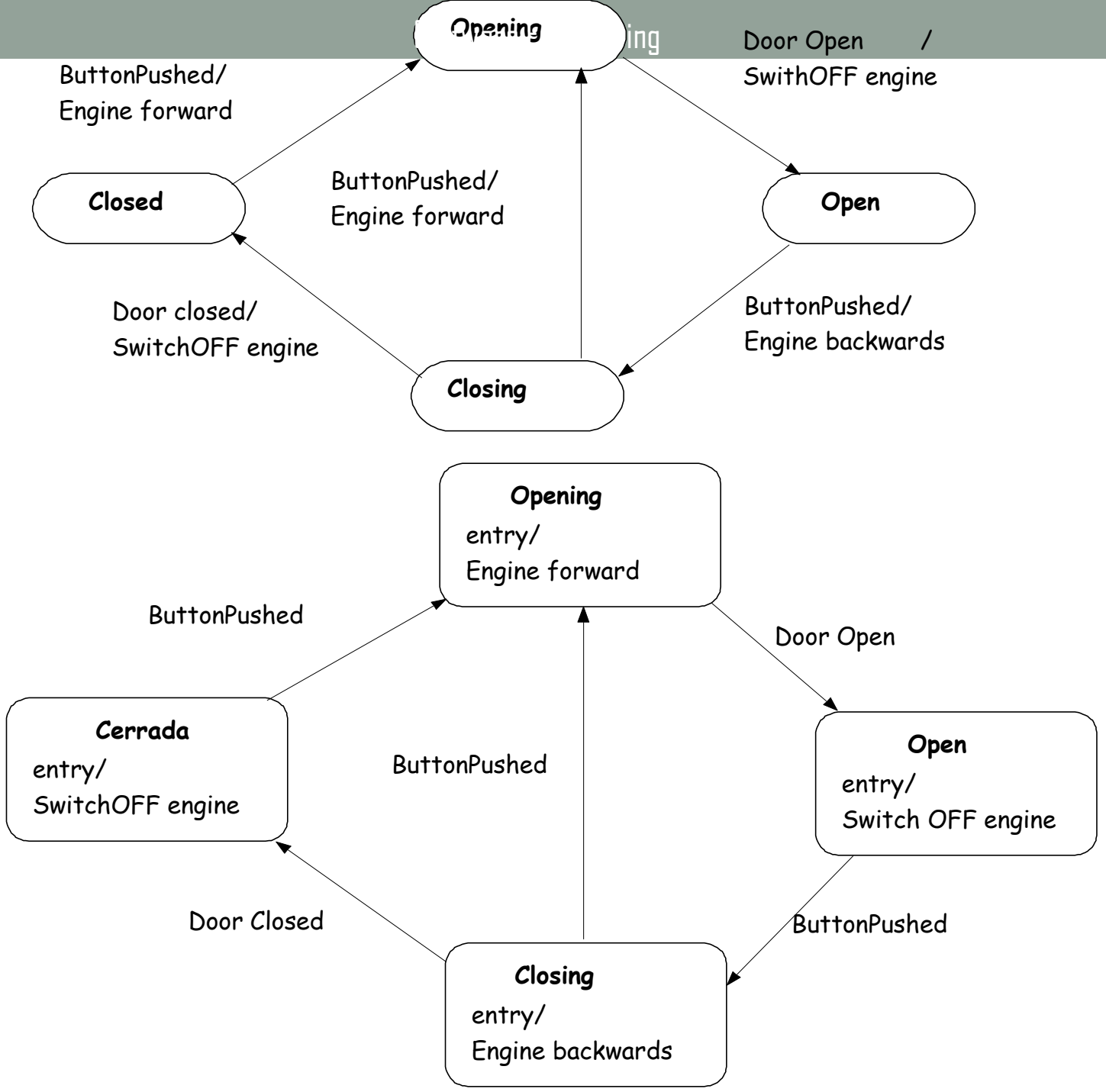
- An operation may be associated to a state or a transition and operations are carried out in response to events or state entries/exits.
- An activity is a non instantaneous operation that is associated to a state. Activities are continuous operations.
- A state may control a continuous activity that lasts until an event triggers a transition and the state is left.

Operations in diagrams

- The notation *"do: A"* or "do/A" in a state indicates that the activity A is started when the state is entered and it finishes when the state is left.

- Actions can also be associated to entries and exits of states.

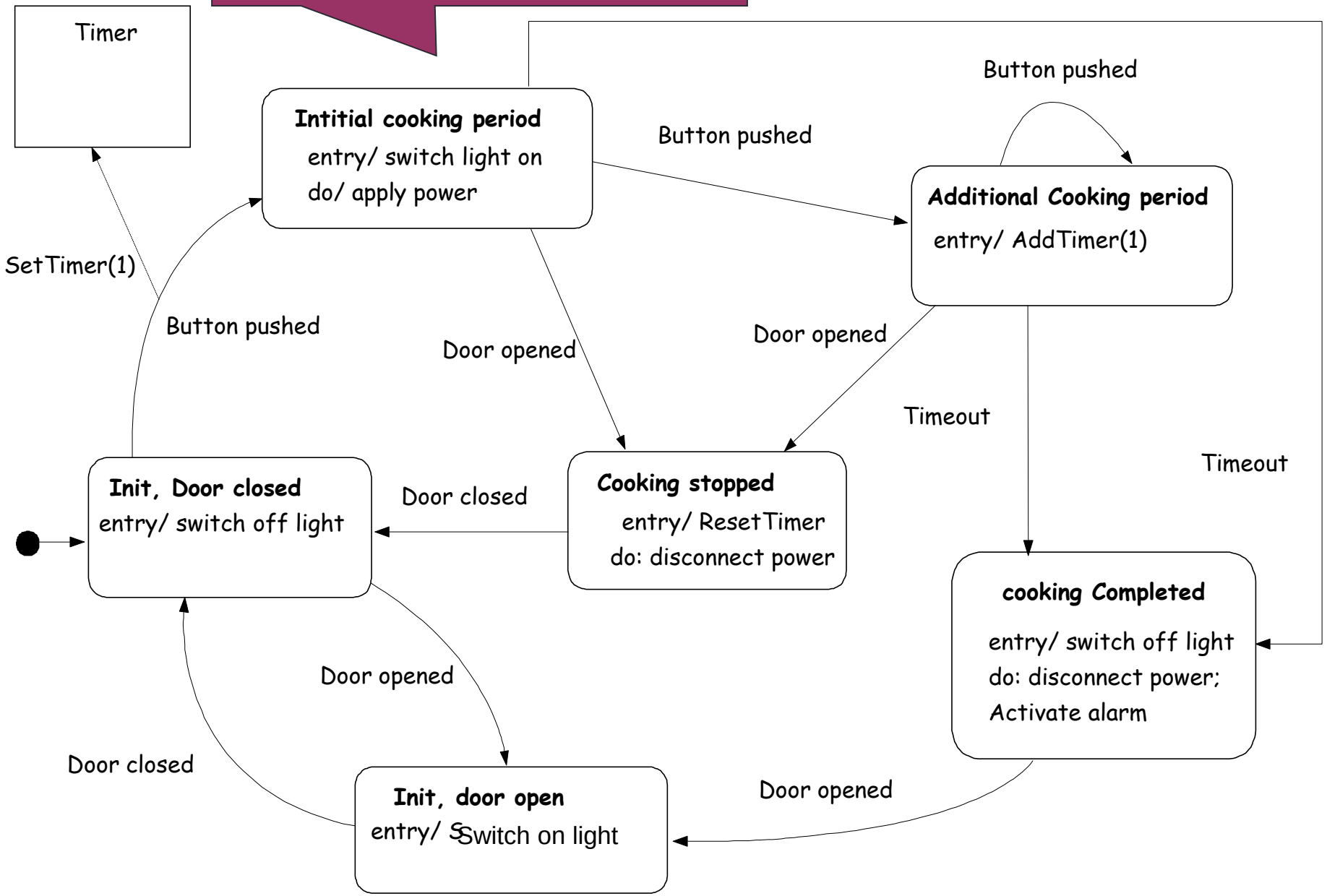



Opening
Door Open /
SwithOFF engine
ButtonPushed/
Engine forward
Closed
ButtonPushed/
Engine forward
Open
Door closed/
SwitchOFF engine
ButtonPushed/
Engine backwards
Closing
Opening
entry/
Engine forward
ButtonPushed
Door Open
Cerrada
entry/
SwitchOFF engine
ButtonPushed
Open
entry/
Switch OFF engine
Door Closed
ButtonPushed
Closing
entry/
Engine backwards

Operations in diagrams

- When a state is entered the entry action is executed. An entry action is executed for all incoming transitions. If an incoming transition is processed its associated action is executed first.
- When exiting a state the exit action is executed.
- Activities are interrupted when leaving a state but actions (entry/exit) are always completed.

Simplified Microwave State Transition Diagram

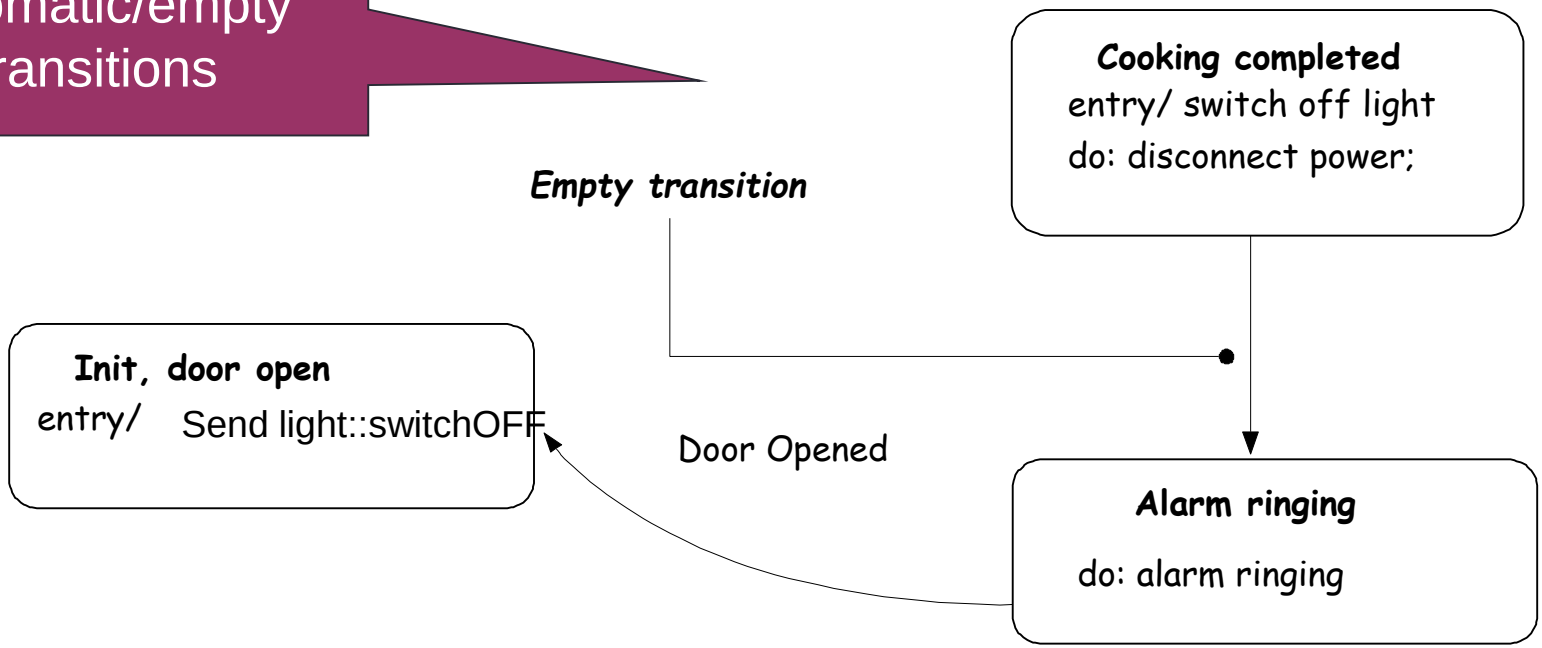


Operations in diagrams

- Internal actions do not cause state transitions.
- An action may be sent to another object. The action *"send E(attributes)"* sends an event E and its attributes to the receiving objects.
- A transition without an event name is an automatic transition and is triggered when the activity associated with the state is completed.

Operations in diagrams

State transition diagram with automatic/empty transitions

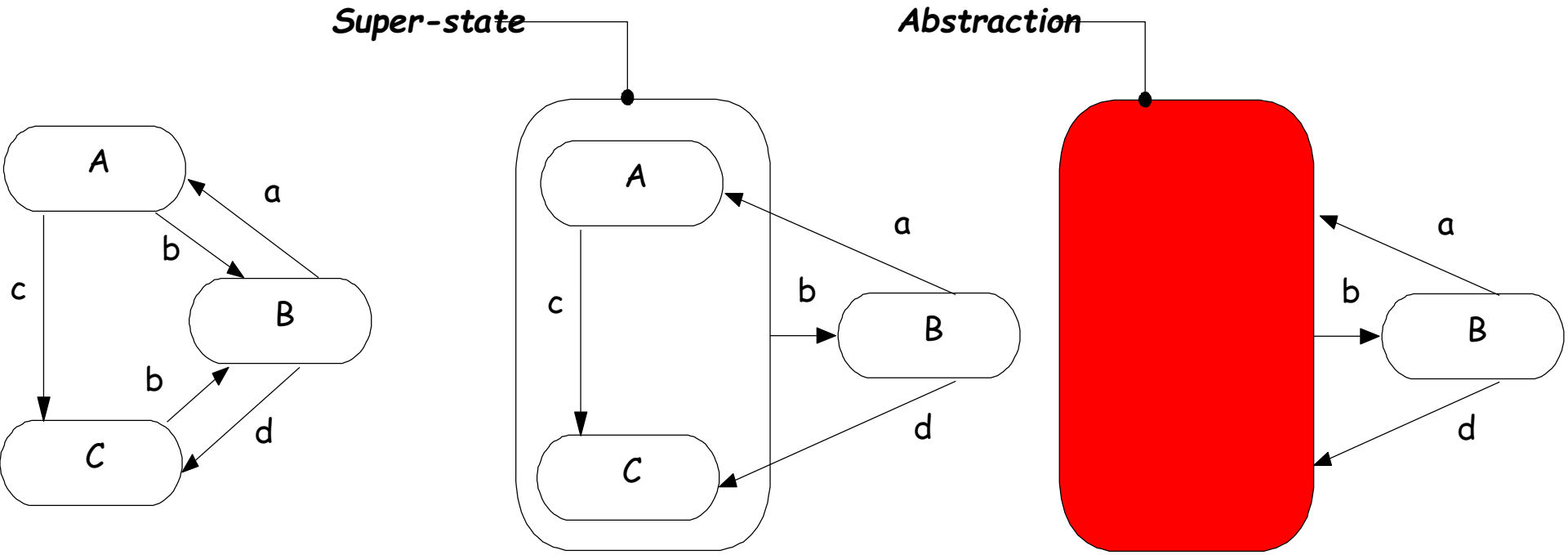


Super-states

- Super-states are used to reduce the number of transitions in a diagram.

- A super-state includes a state transition diagram or another super-state.

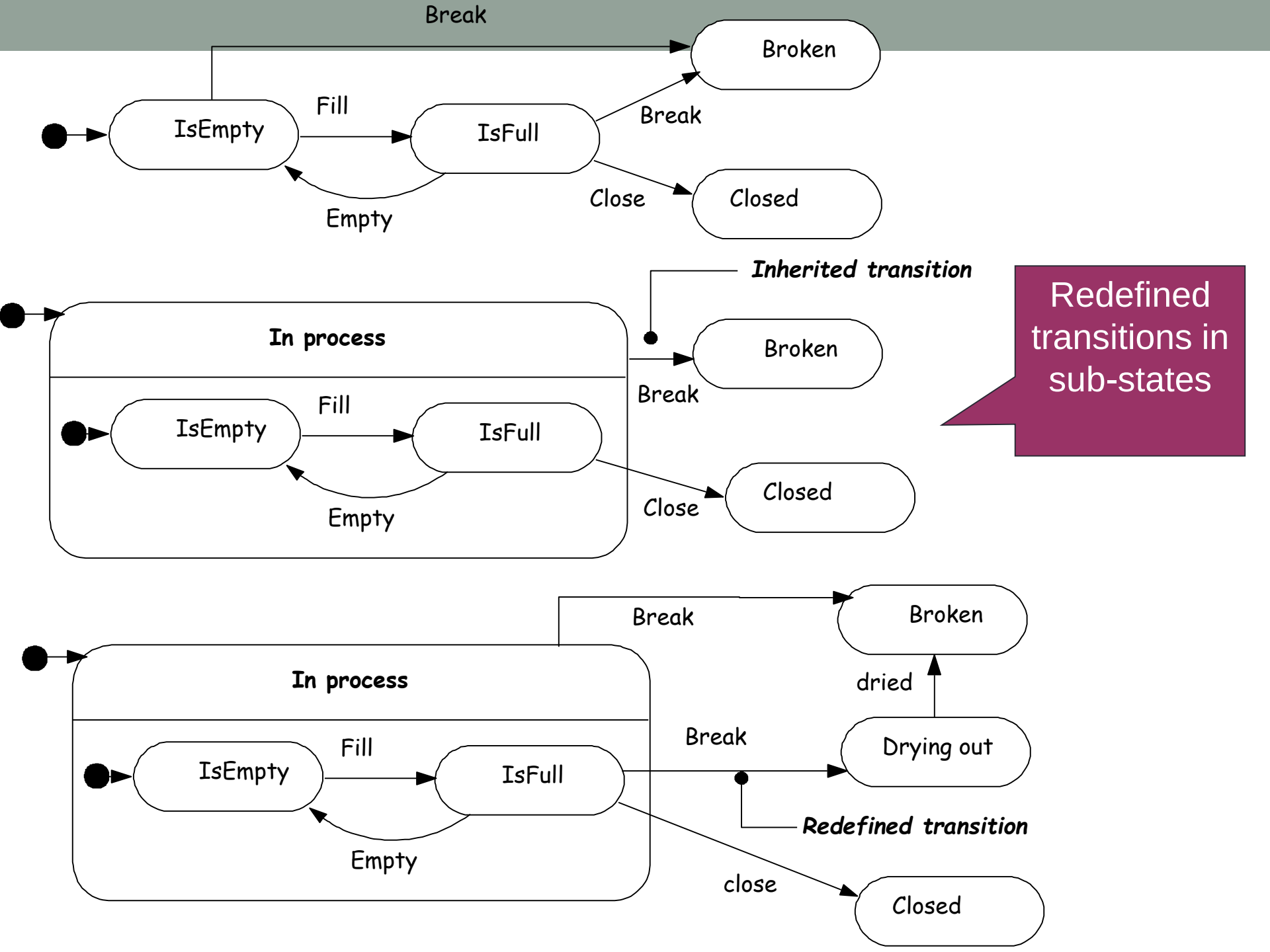
Super-states



Super-states

- A super-state is considered as another normal state: actions, activities, etc.

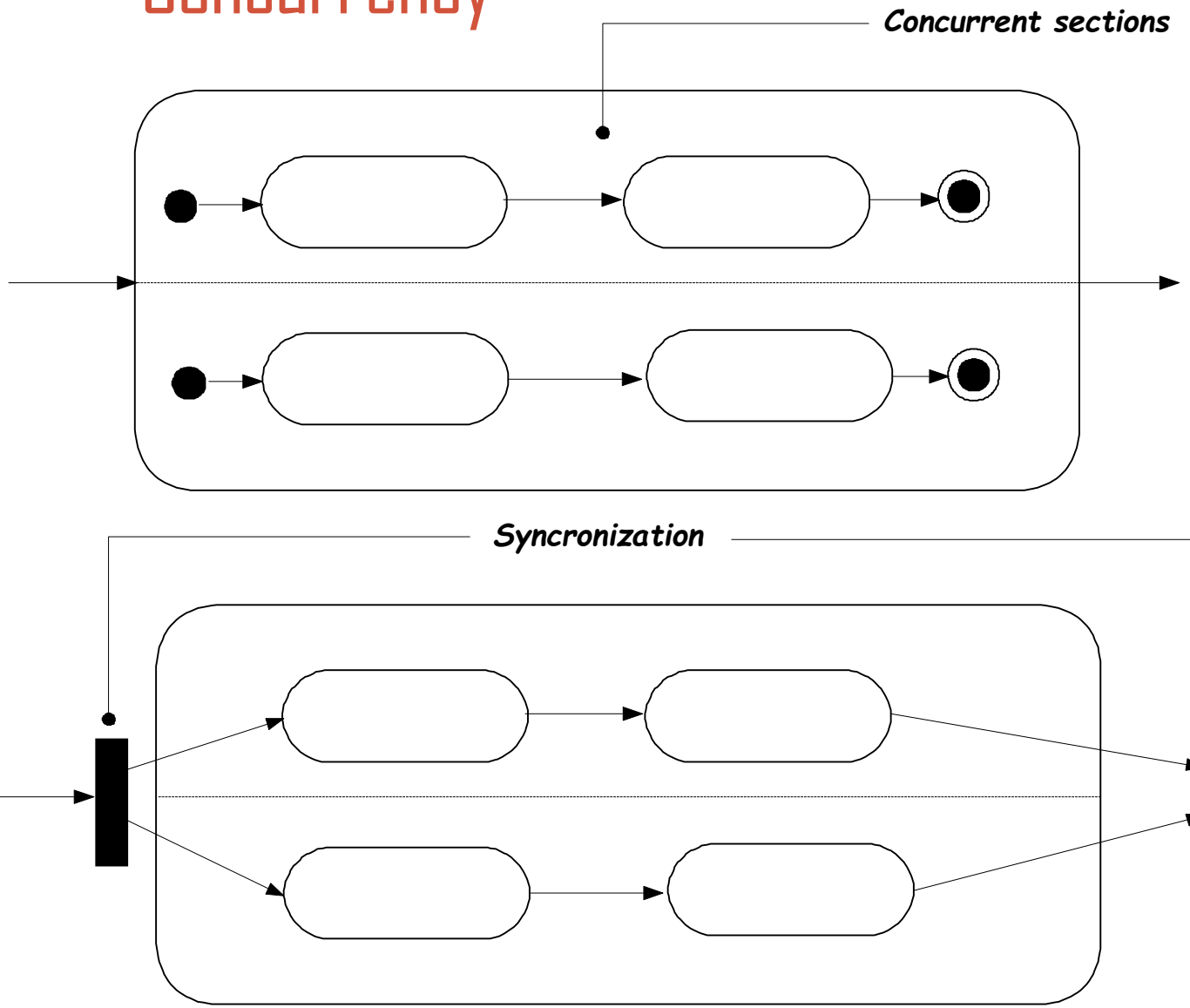
- Its substates inherit all the outgoing transitions of the super-state but they can be redefined.


Break
Broken
IsEmpty
Fill
IsFull
Break
Close
Closed
Empty
Inherited transition
In process
Broken
Break
IsEmpty
Fill
IsFull
Closed
Close
Empty
Redefined transitions in sub-states
Break
Broken
In process
dried
Break
Drying out
IsEmpty
Fill
IsFull
Redefined transition
Empty
close
Closed

Concurrency

Concurrent state transition diagrams are used when an object contains independent behaviors or independent execution regions.

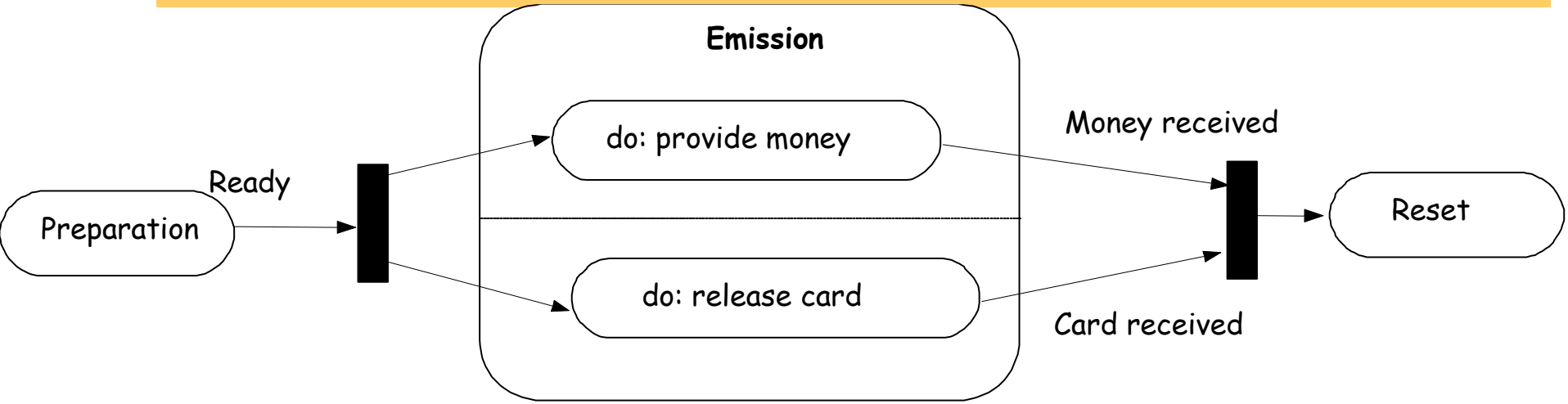
Concurrency



Notation for concurrency within an object

Concurrency

The initial diagram may be divided into independent ones.



Concurrency within a super-state