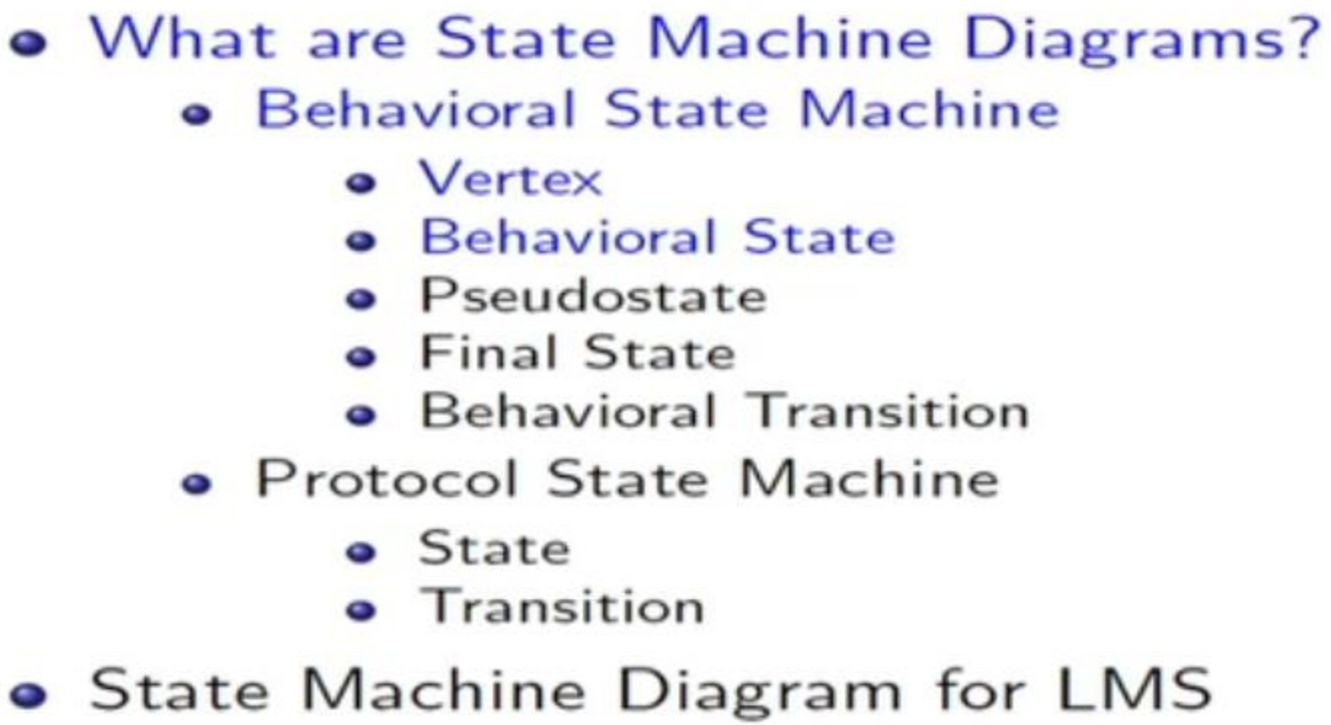
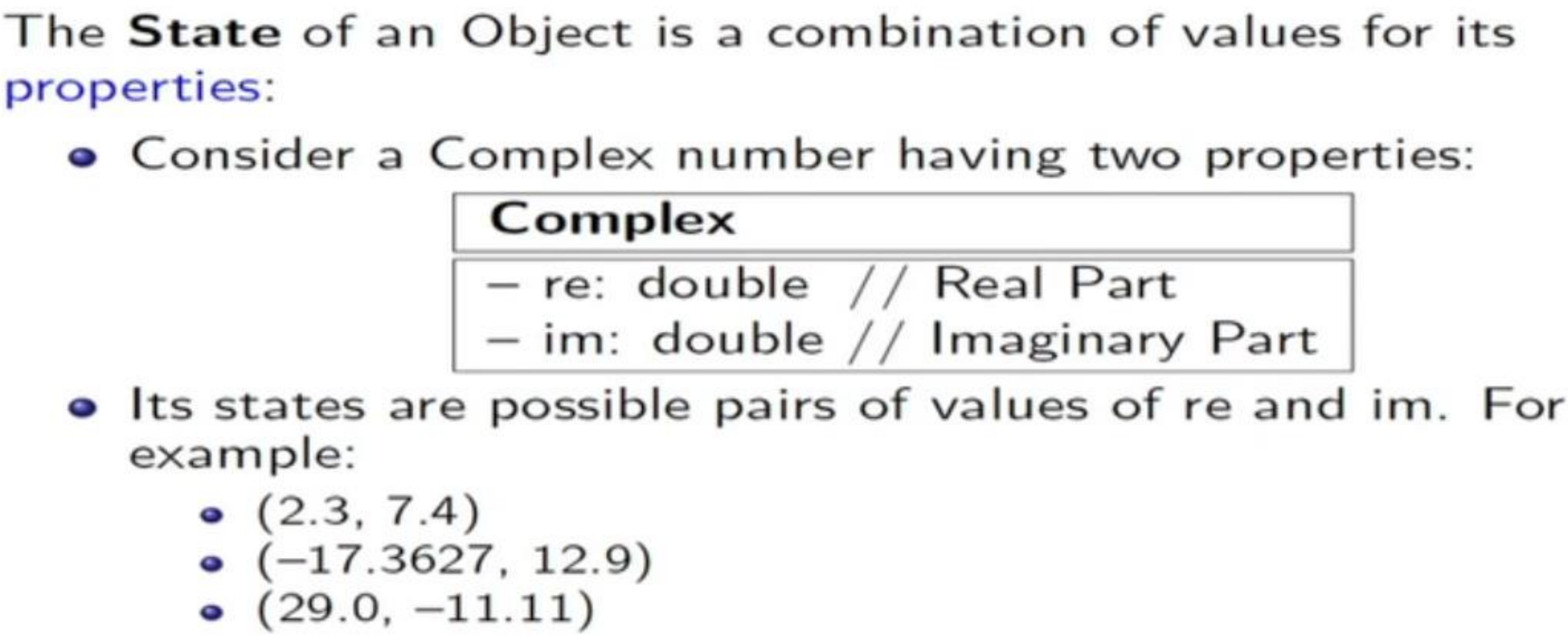
State Machine Diagram

Overview



- What are State Machine Diagrams?
  - Behavioral State Machine
    - Vertex
    - Behavioral State
    - Pseudostate
    - Final State
    - Behavioral Transition
  - Protocol State Machine
    - State
    - Transition
- State Machine Diagram for LMS

State of an Object



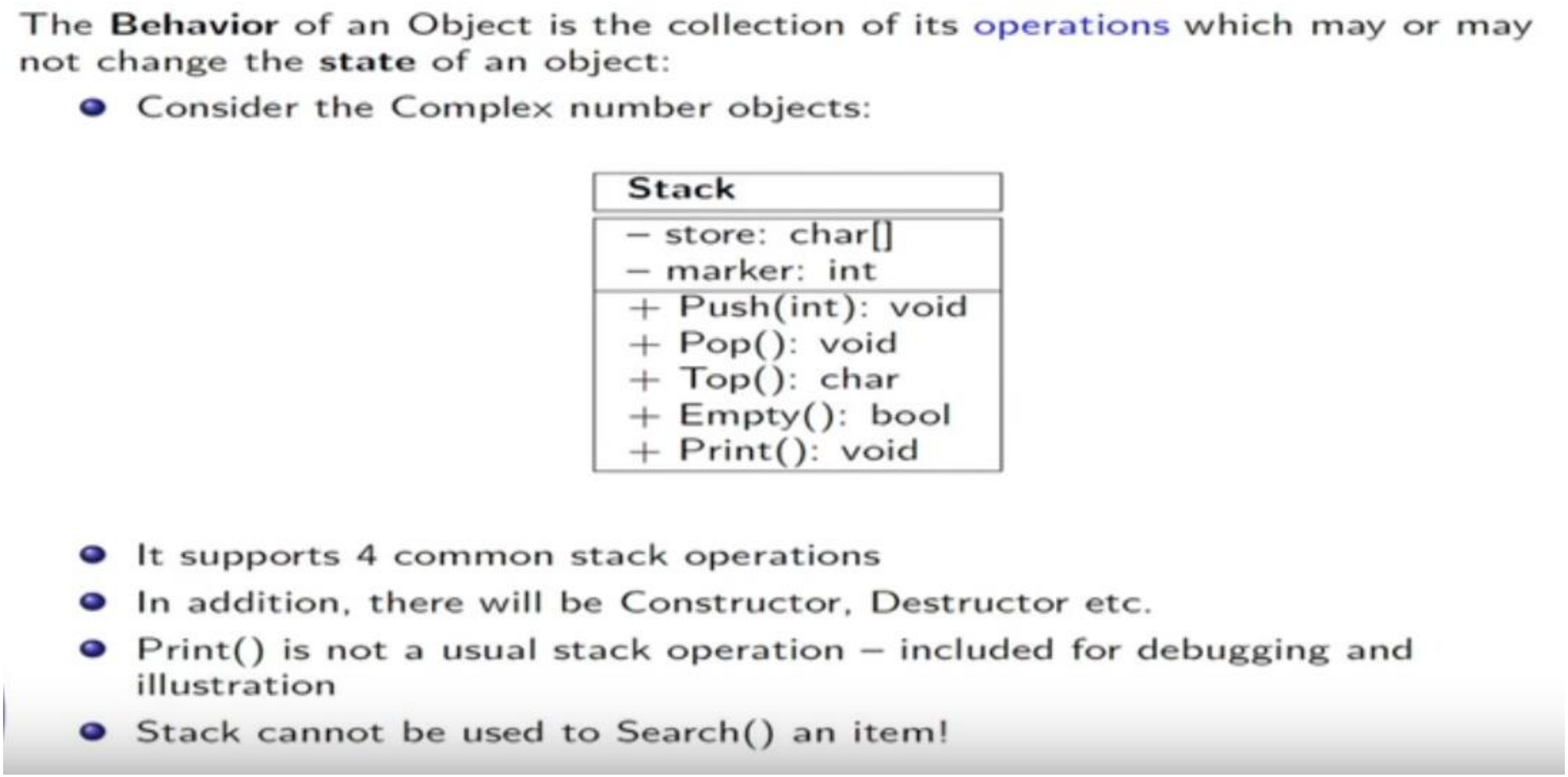
The **State** of an Object is a combination of values for its properties:

- Consider a Complex number having two properties:

Complex
– re: double // Real Part – im: double // Imaginary Part

- Its states are possible pairs of values of re and im. For example:
  - (2.3, 7.4)
  - (–17.3627, 12.9)
  - (29.0, –11.11)

Behaviour of an Object



The **Behavior** of an Object is the collection of its operations which may or may not change the **state** of an object:

- Consider the Complex number objects:

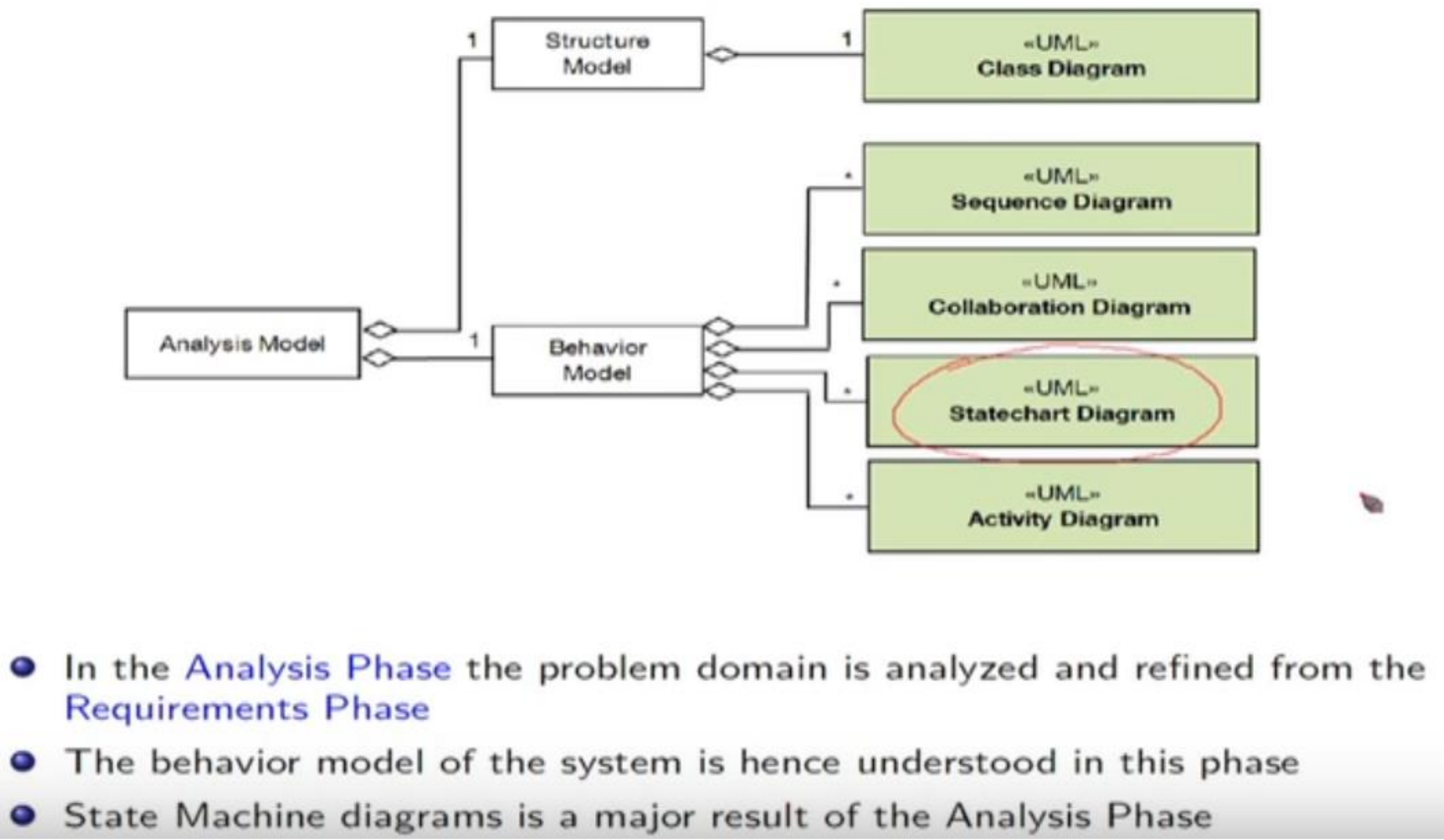
Stack

– store: char[]
– marker: int

+ Push(int): void
+ Pop(): void
+ Top(): char
+ Empty(): bool
+ Print(): void

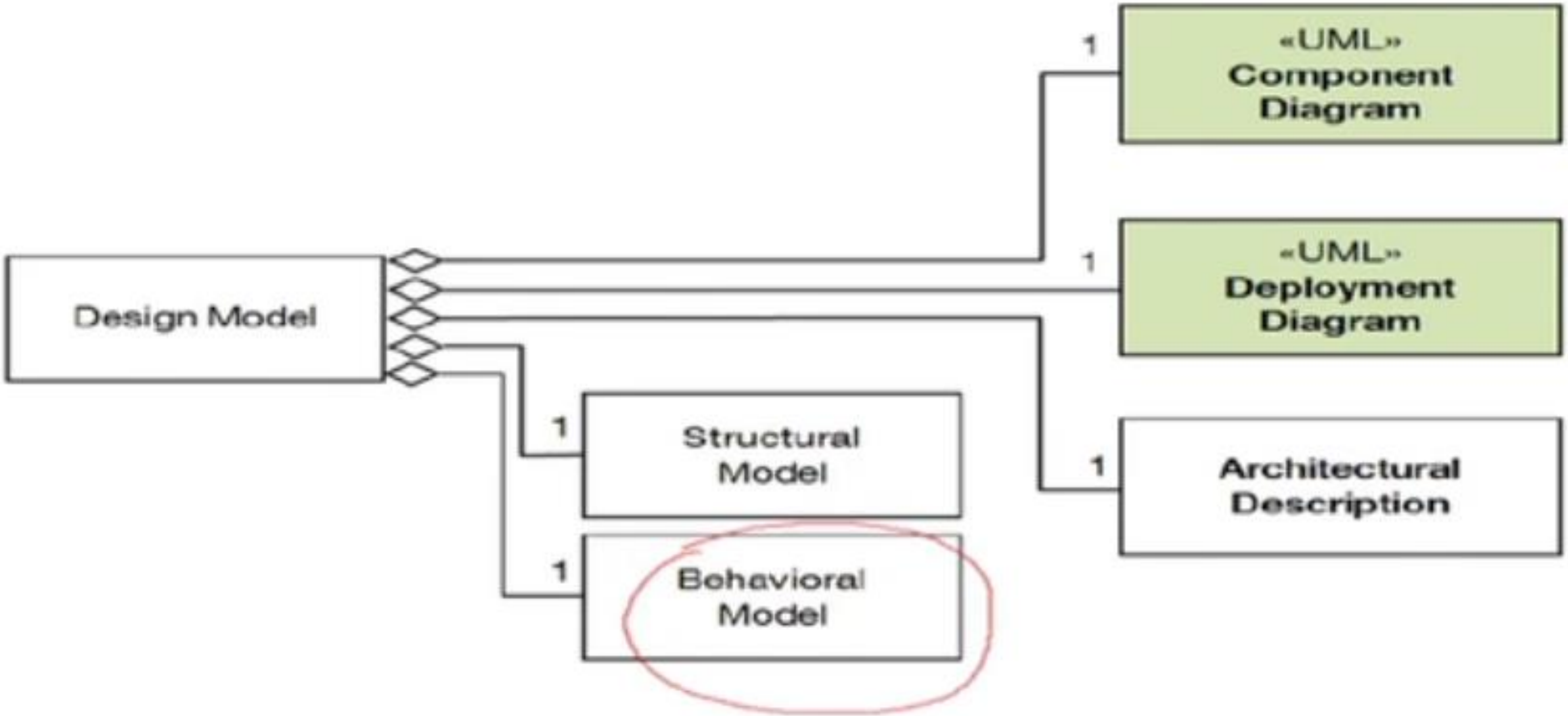
- It supports 4 common stack operations
- In addition, there will be Constructor, Destructor etc.
- Print() is not a usual stack operation – included for debugging and illustration
- Stack cannot be used to Search() an item!

State Machine Diagram



- In the Analysis Phase the problem domain is analyzed and refined from the Requirements Phase
- The behavior model of the system is hence understood in this phase
- State Machine diagrams is a major result of the Analysis Phase

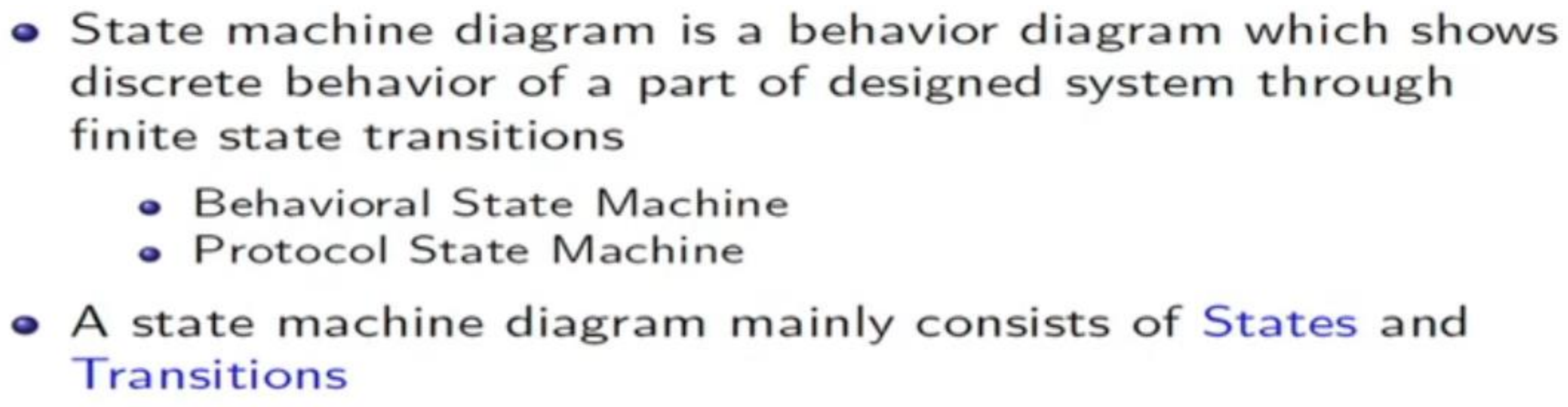
State Machine Diagram



- State Machine diagram is included in the Behavioral Model
- It is further refined in the Design Phase

State Machine Diagram

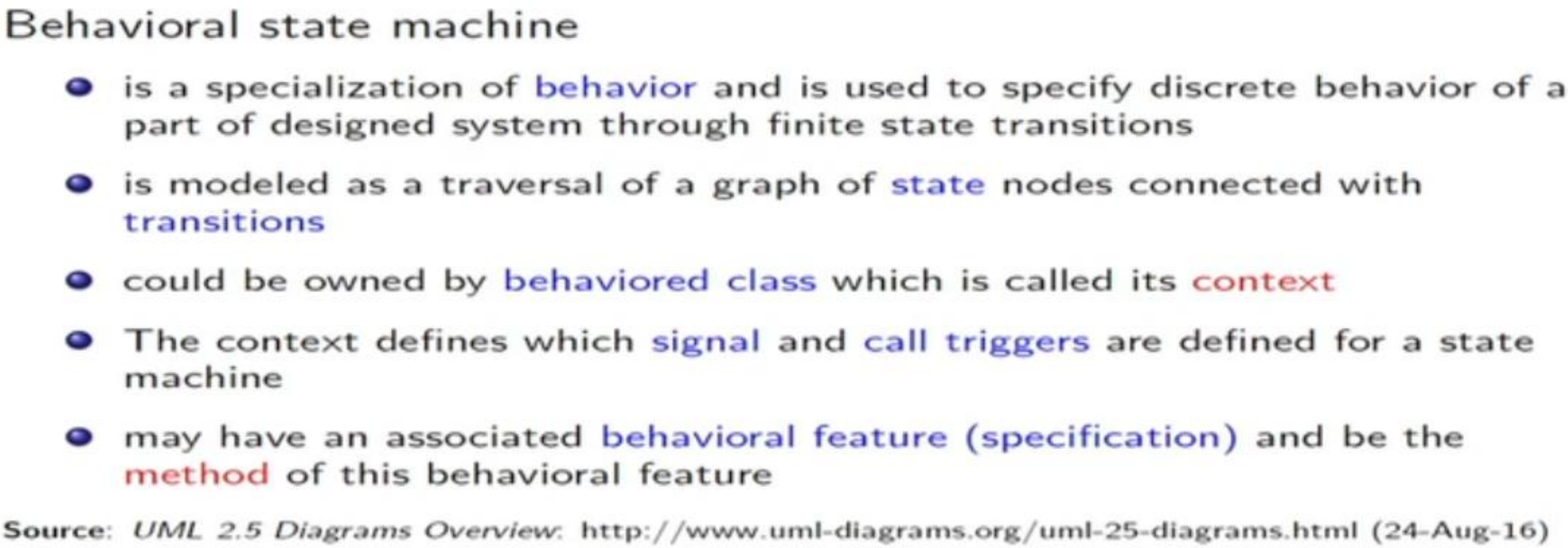
What are State Machine Diagrams



- State machine diagram is a behavior diagram which shows discrete behavior of a part of designed system through finite state transitions
  - Behavioral State Machine
  - Protocol State Machine
- A state machine diagram mainly consists of States and Transitions

State Machine Diagram

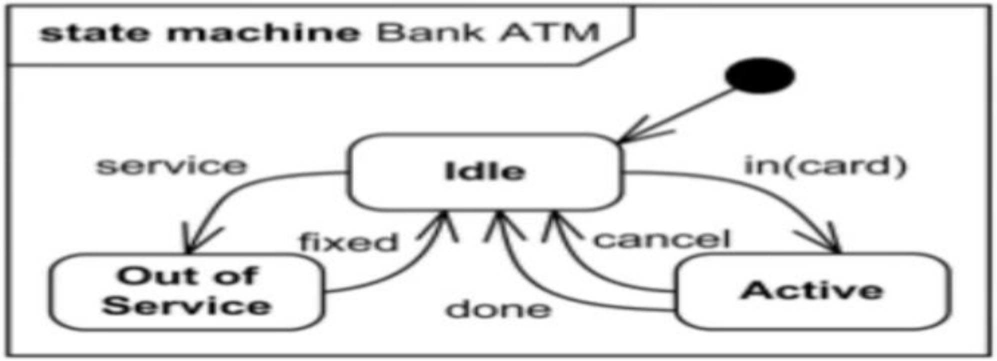
Behavioral State Machine



Behavioral state machine

- is a specialization of behavior and is used to specify discrete behavior of a part of designed system through finite state transitions
- is modeled as a traversal of a graph of state nodes connected with transitions
- could be owned by behaviored class which is called its context
- The context defines which signal and call triggers are defined for a state machine
- may have an associated behavioral feature (specification) and be the method of this behavioral feature

Source: *UML 2.5 Diagrams Overview*. http://www.uml-diagrams.org/uml-25-diagrams.html (24-Aug-16)

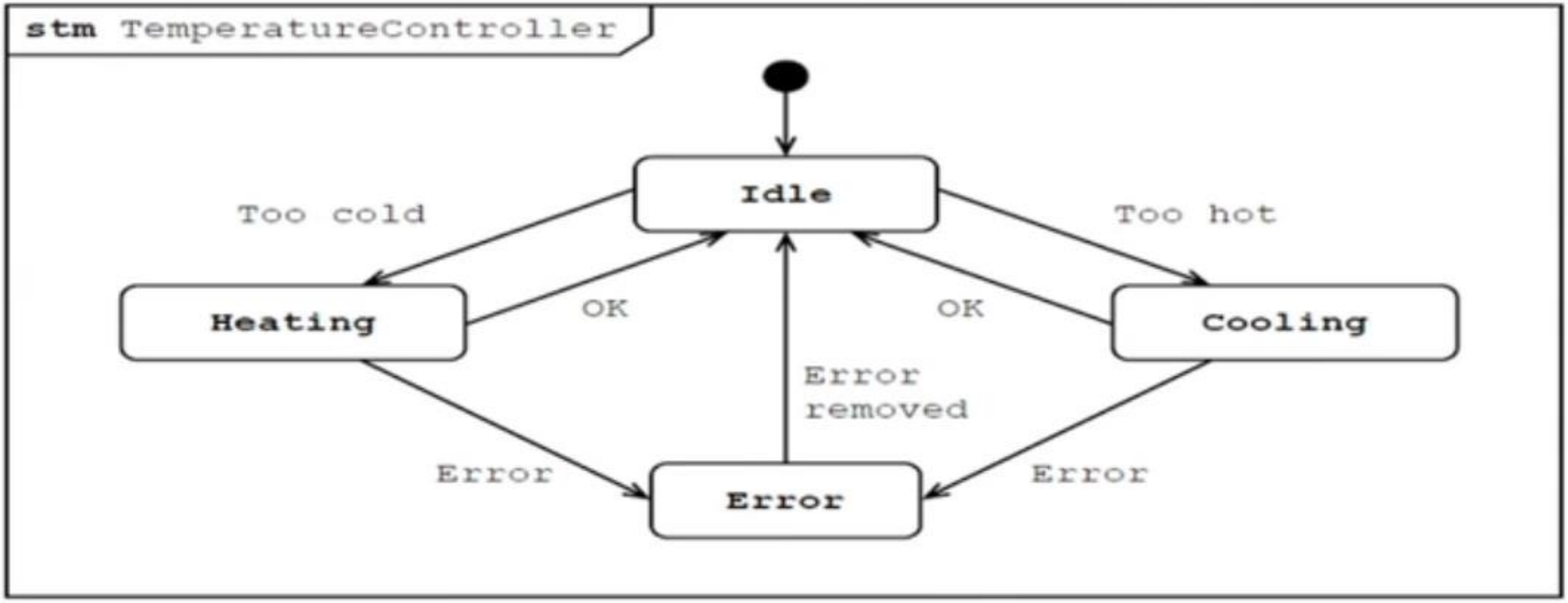


High level behavioral state machine for bank ATM

Source: *UML 2.5 Diagrams Overview*. http://www.uml-diagrams.org/uml-25-diagrams.html (24-Aug-16)

State Machine Diagram

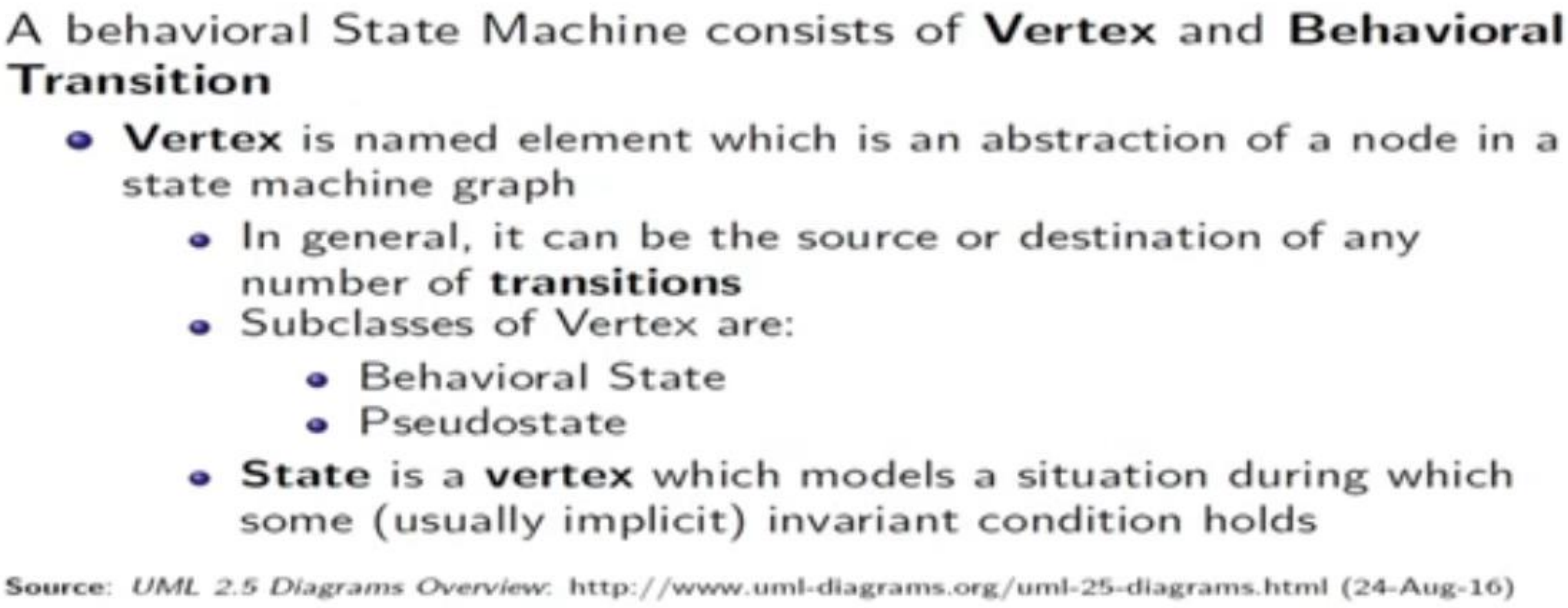
Behavioral State Machine



Behavioral state machine for Temperature Controller

State Machine Diagram

Vertex



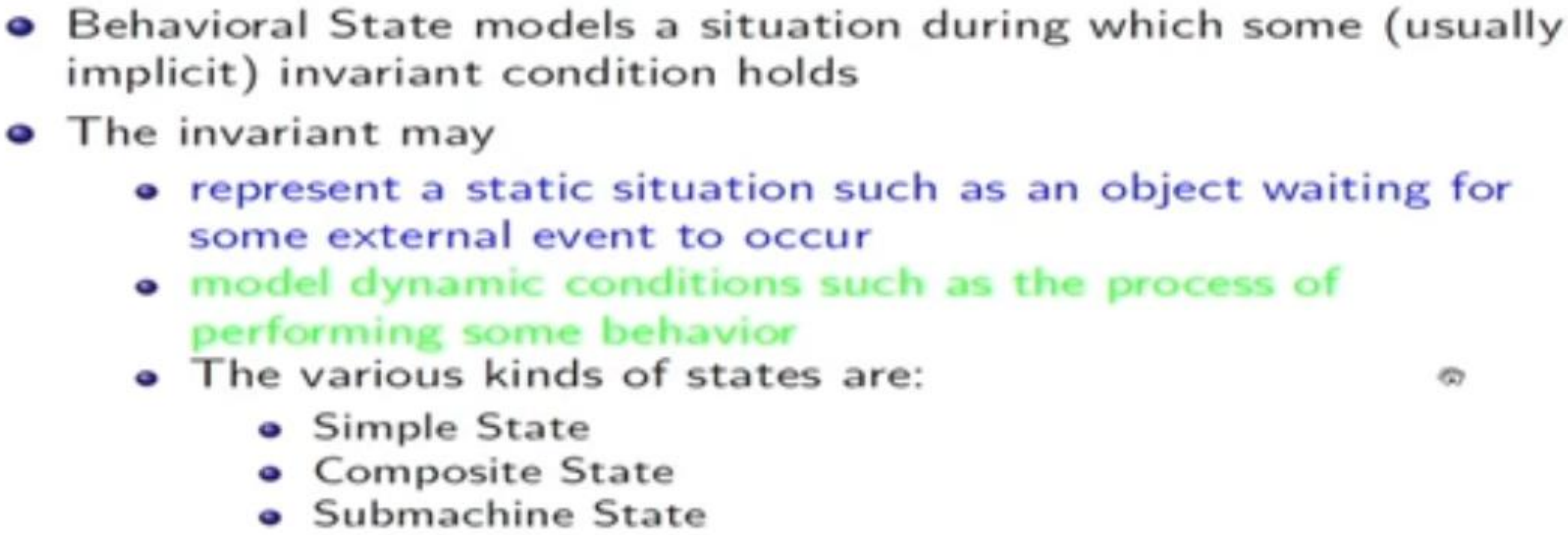
A behavioral State Machine consists of **Vertex** and **Behavioral Transition**

- **Vertex** is named element which is an abstraction of a node in a state machine graph
  - In general, it can be the source or destination of any number of **transitions**
  - Subclasses of Vertex are:
    - Behavioral State
    - Pseudostate
  - **State** is a **vertex** which models a situation during which some (usually implicit) invariant condition holds

Source: *UML 2.5 Diagrams Overview*: http://www.uml-diagrams.org/uml-25-diagrams.html (24-Aug-16)

State Machine Diagram

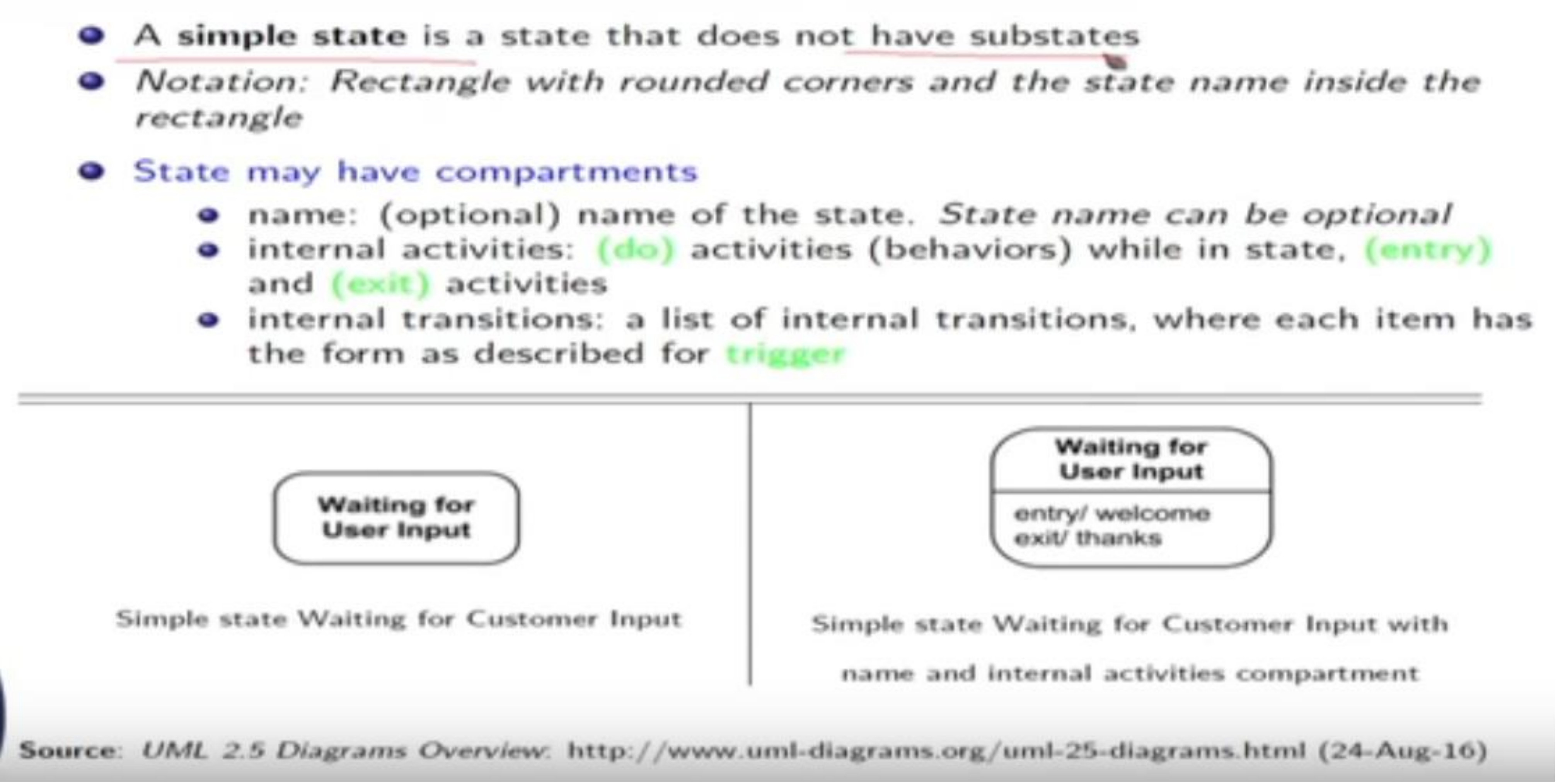
Vertex: Behavioral State



- Behavioral State models a situation during which some (usually implicit) invariant condition holds
- The invariant may
  - represent a static situation such as an object waiting for some external event to occur
  - model dynamic conditions such as the process of performing some behavior
  - The various kinds of states are:
    - Simple State
    - Composite State
    - Submachine State

State Machine Diagram

Behavioral State: Simple State



- A **simple state** is a state that does not have substates
- *Notation: Rectangle with rounded corners and the state name inside the rectangle*
- State may have compartments
  - name: (optional) name of the state. *State name can be optional*
  - internal activities: (do) activities (behaviors) while in state, (entry) and (exit) activities
  - internal transitions: a list of internal transitions, where each item has the form as described for trigger

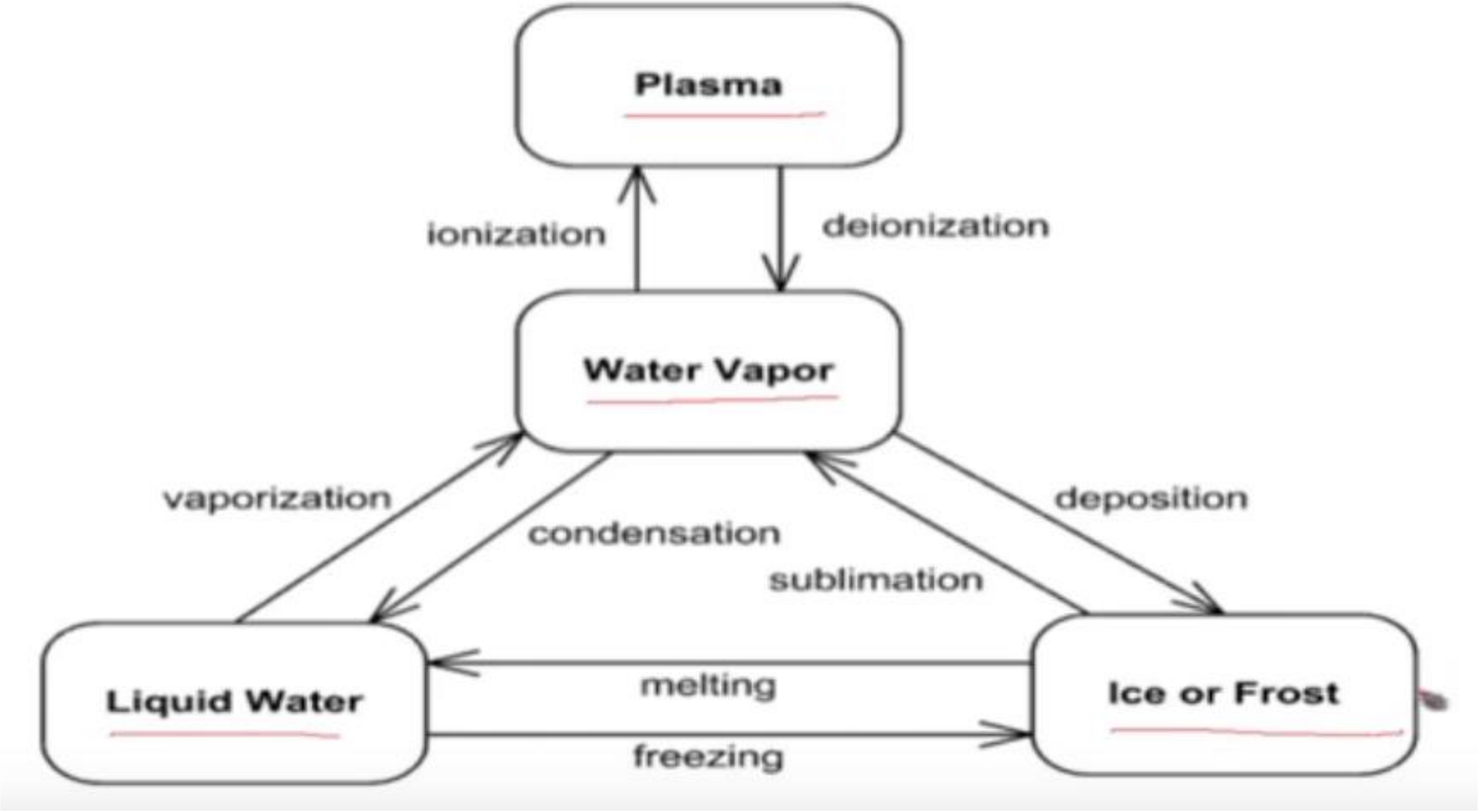
Simple state Waiting for Customer Input

Simple state Waiting for Customer Input with name and internal activities compartment

Source: *UML 2.5 Diagrams Overview*: http://www.uml-diagrams.org/uml-25-diagrams.html (24-Aug-16)

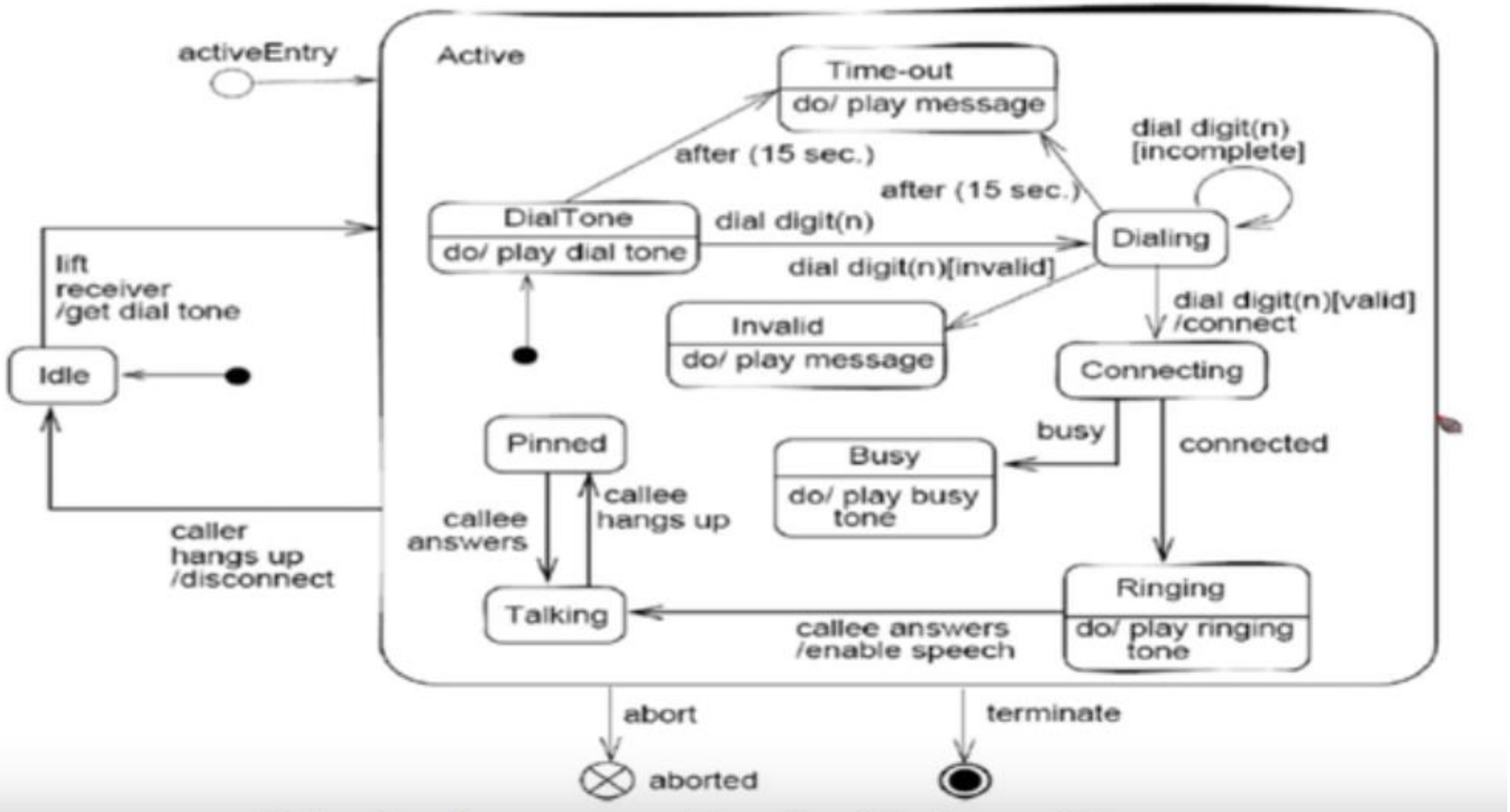
State Machine Diagram

Water Phase Management



State Machine Diagram

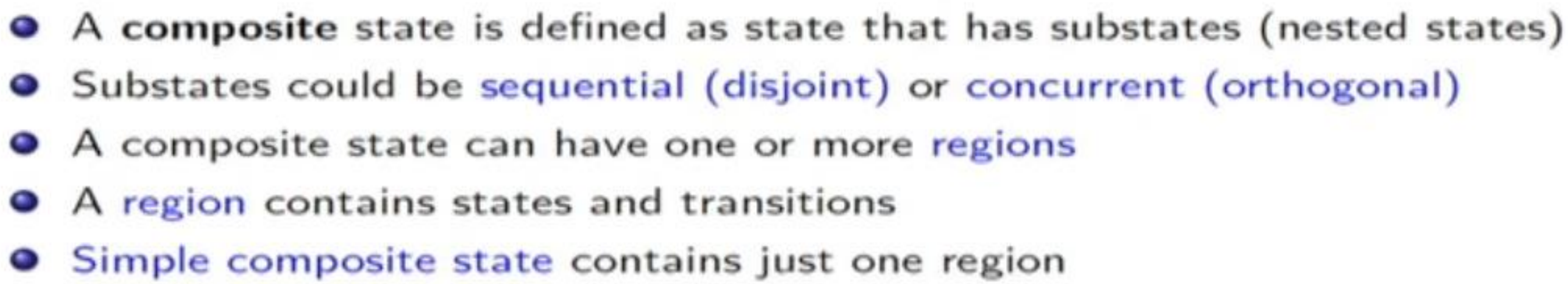
Behavioral State Machine for Dialing a Phone



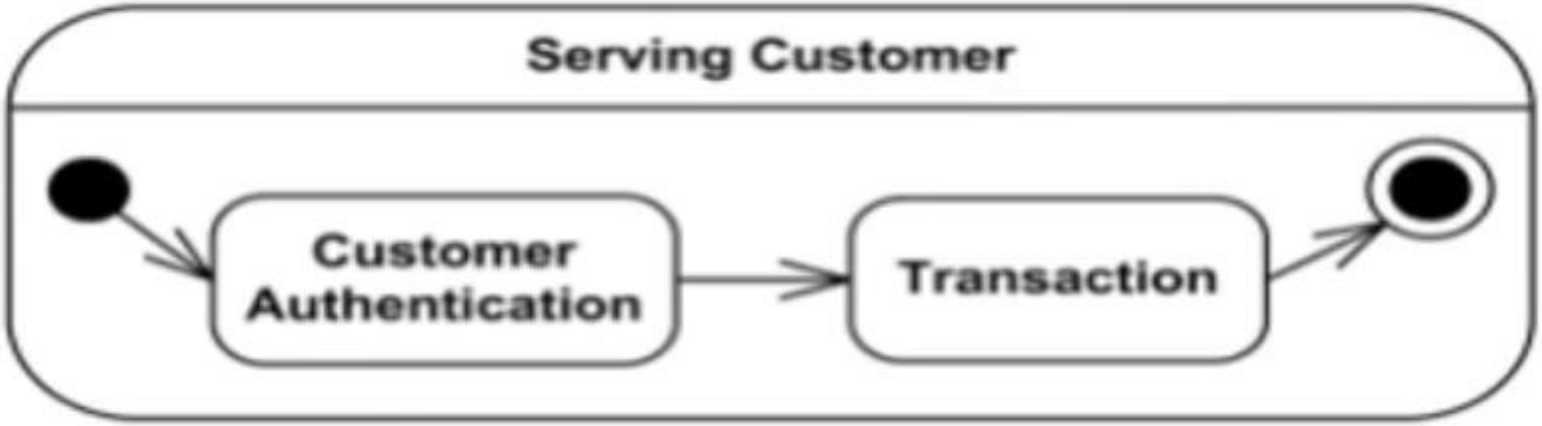
Behavioral state machine for Dialing a Phone

State Machine Diagram

Behavioral State- Composite State



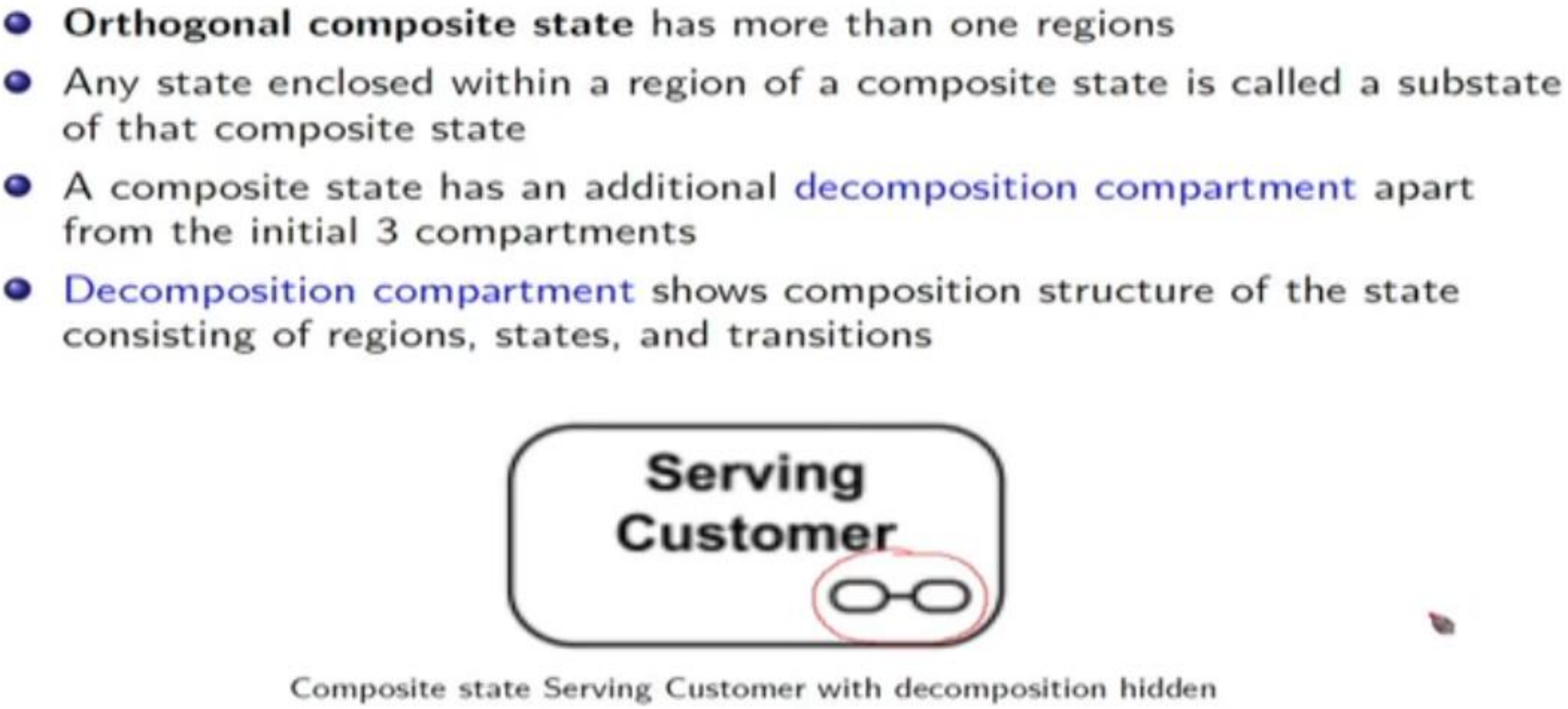
- A **composite** state is defined as state that has substates (nested states)
- Substates could be sequential (disjoint) or concurrent (orthogonal)
- A composite state can have one or more regions
- A region contains states and transitions
- Simple composite state contains just one region



Simple composite state Serving Customer has two substates

State Machine Diagram

Behavioral State- Composite State

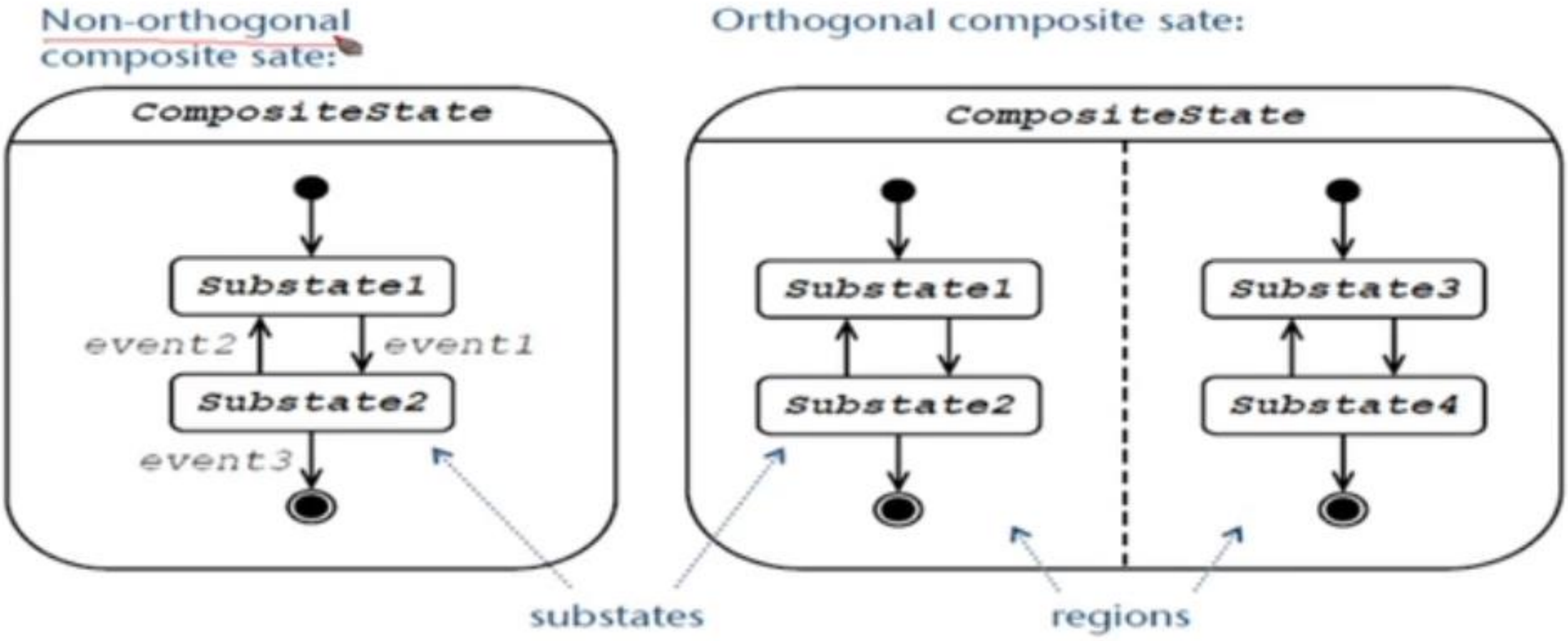


- **Orthogonal composite state** has more than one regions
- Any state enclosed within a region of a composite state is called a substate of that composite state
- A composite state has an additional decomposition compartment apart from the initial 3 compartments
- Decomposition compartment shows composition structure of the state consisting of regions, states, and transitions

Composite state Serving Customer with decomposition hidden

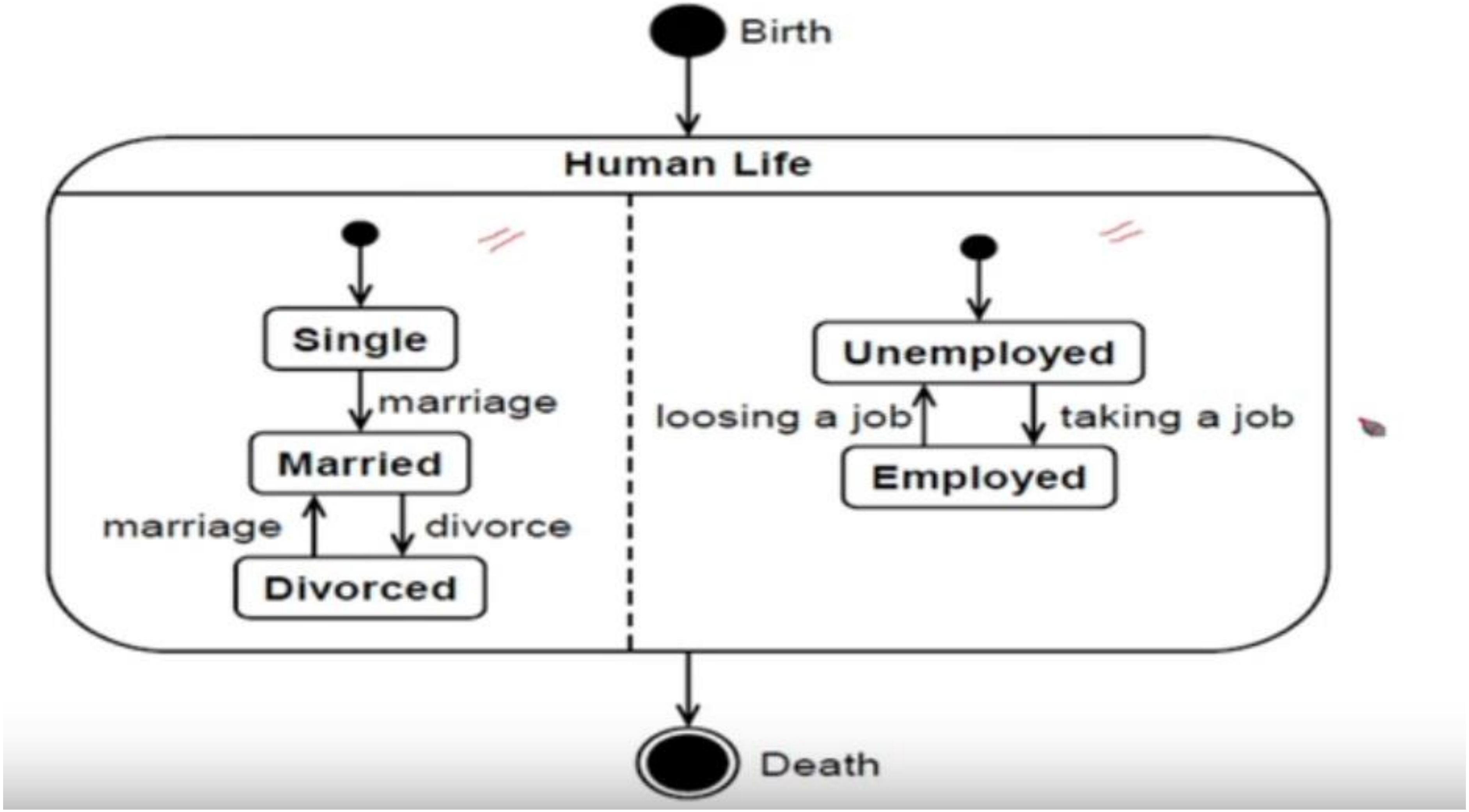
State Machine Diagram

Behavioral State- Orthogonal Composite State



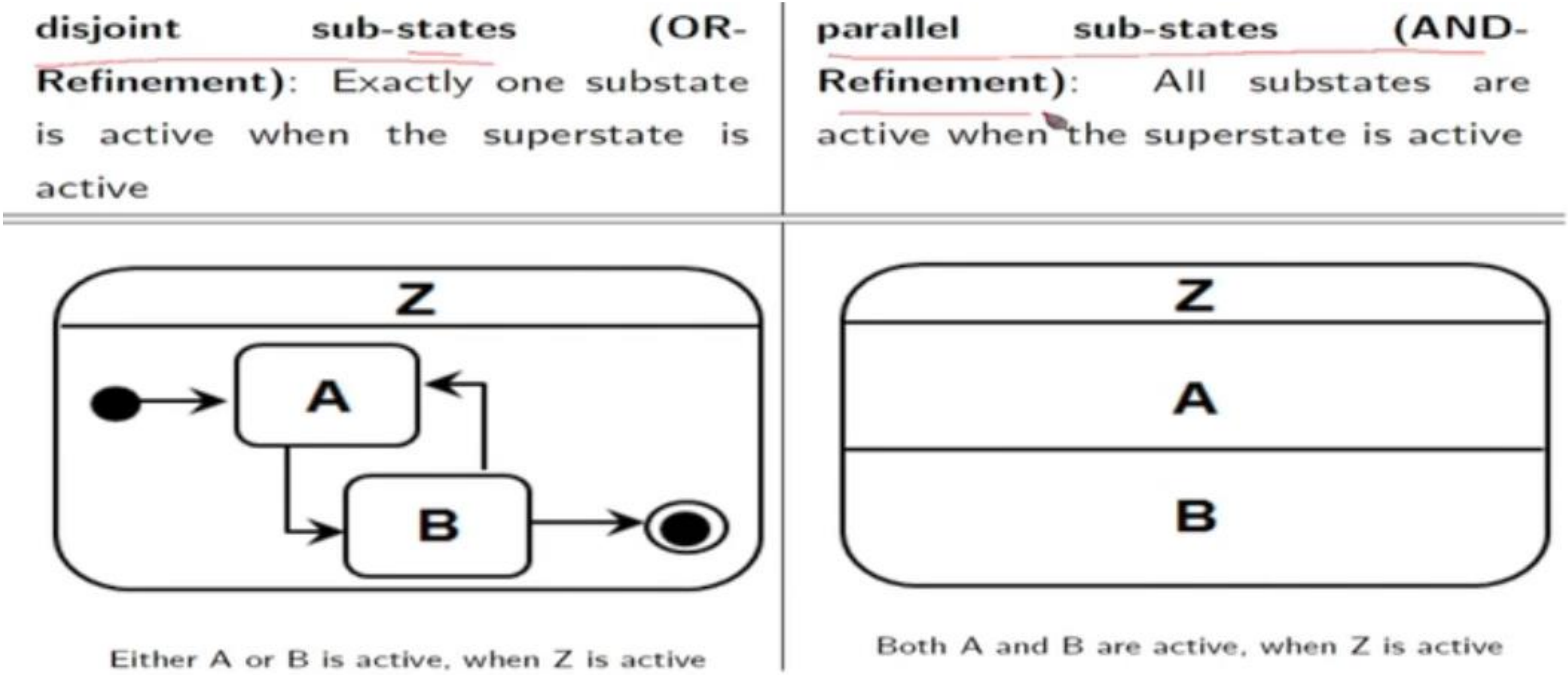
State Machine Diagram

Behavioral State- Orthogonal Composite State



State Machine Diagram

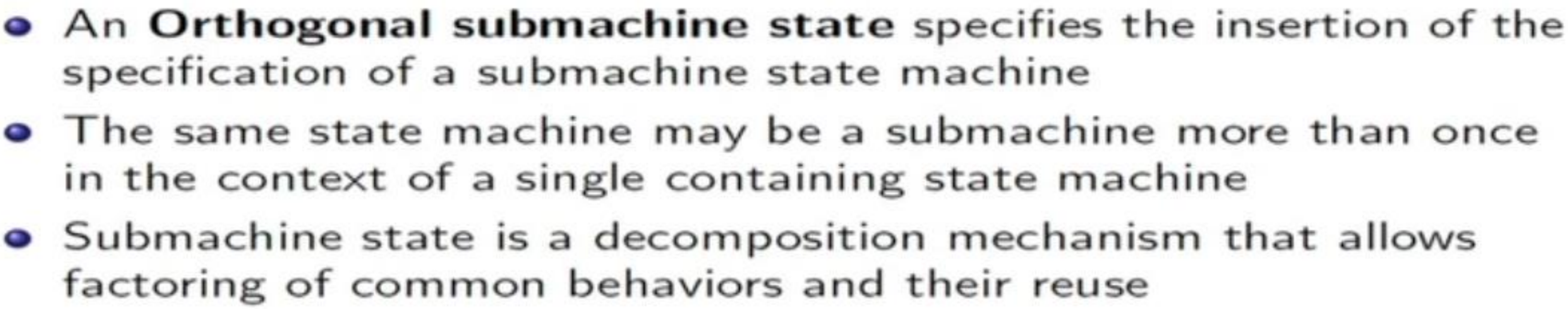
Behavioral State- Super-State and Sub-State



disjoint sub-states (OR-Refinement): Exactly one substate is active when the superstate is active	**parallel sub-states (AND-Refinement)**: All substates are active when the superstate is active
Either A or B is active, when Z is active	Both A and B are active, when Z is active

State Machine Diagram

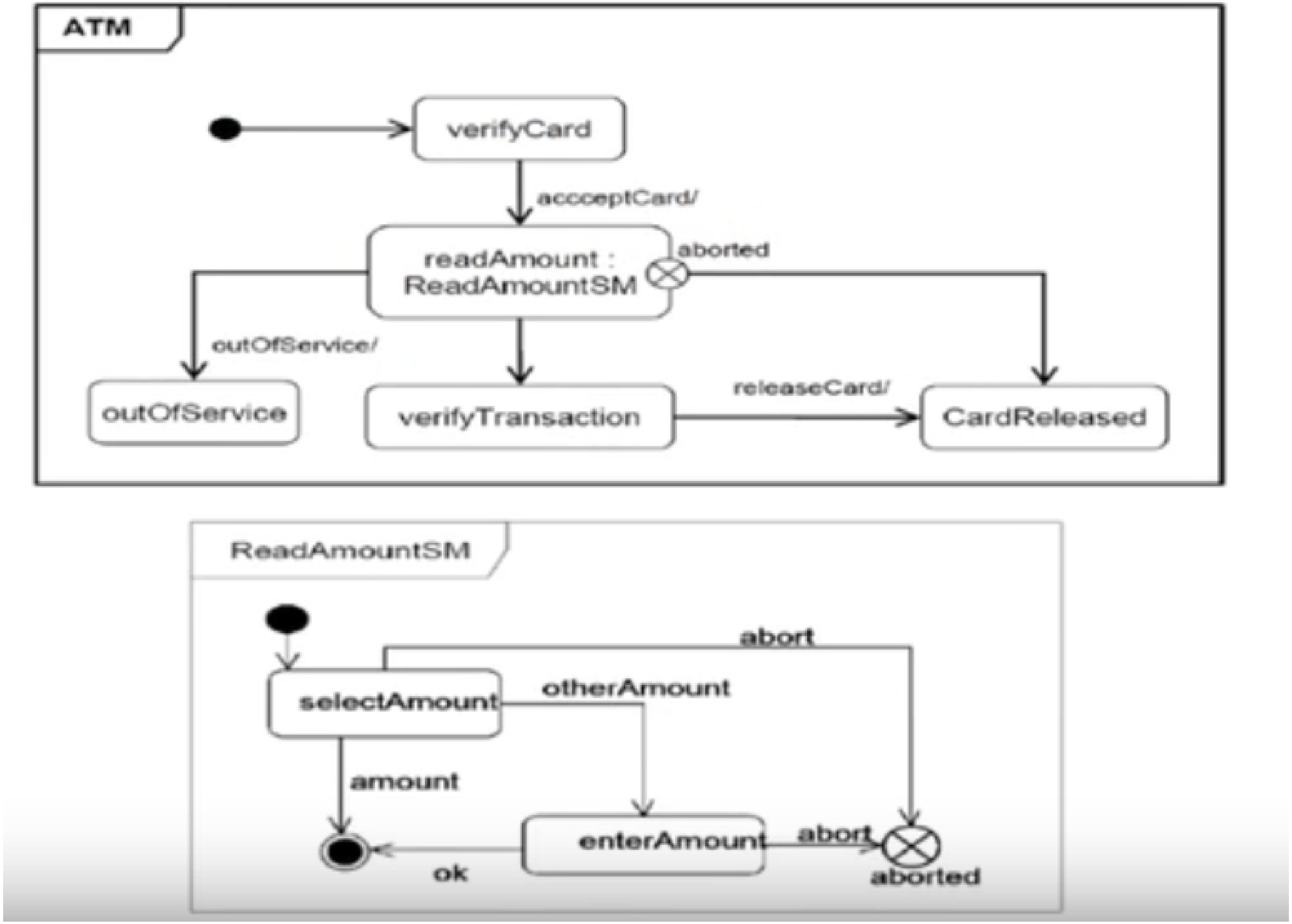
Behavioral State- submachine State



- An **Orthogonal submachine state** specifies the insertion of the specification of a submachine state machine
- The same state machine may be a submachine more than once in the context of a single containing state machine
- Submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse

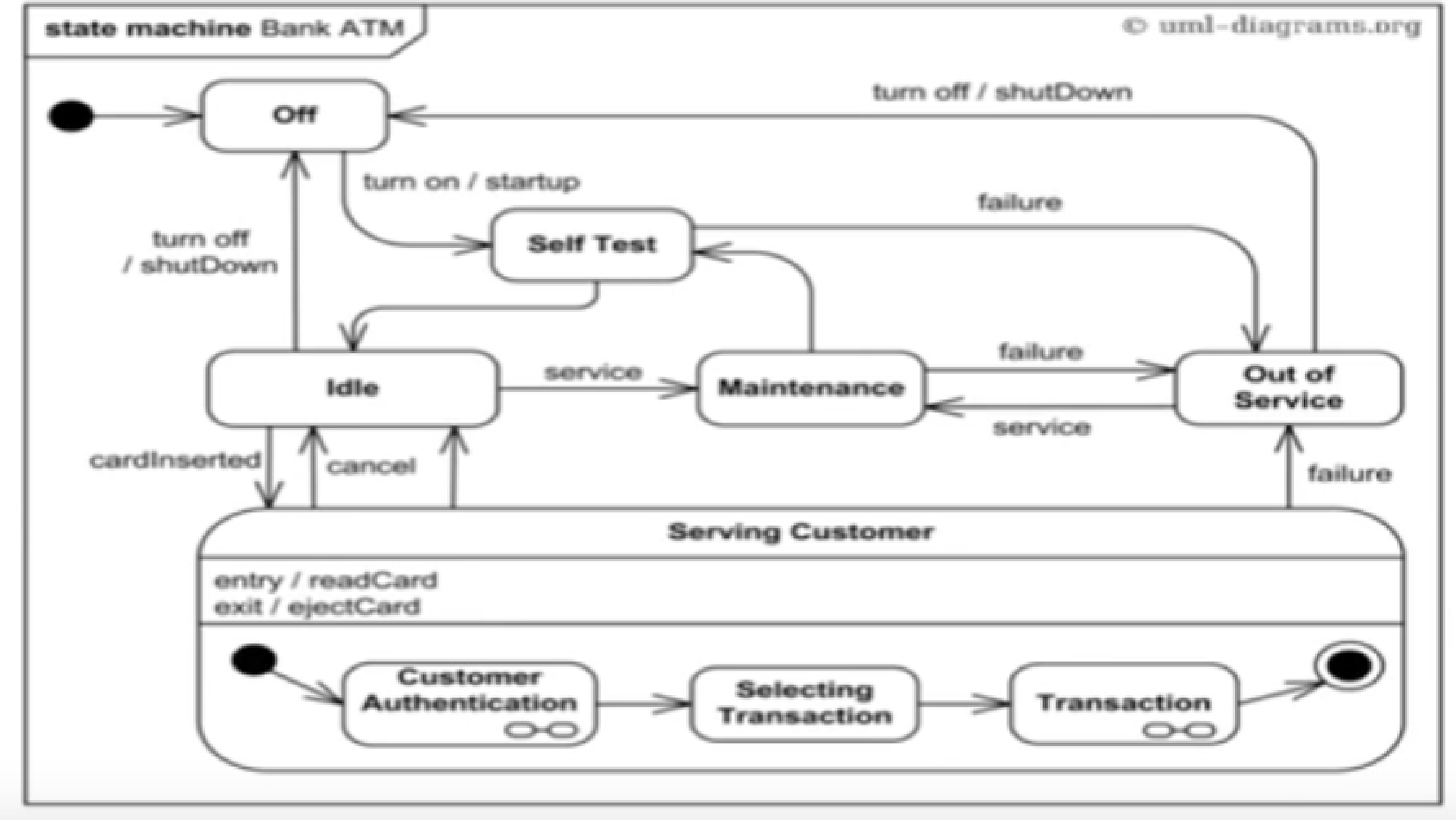
State Machine Diagram

Behavioral State- submachine State

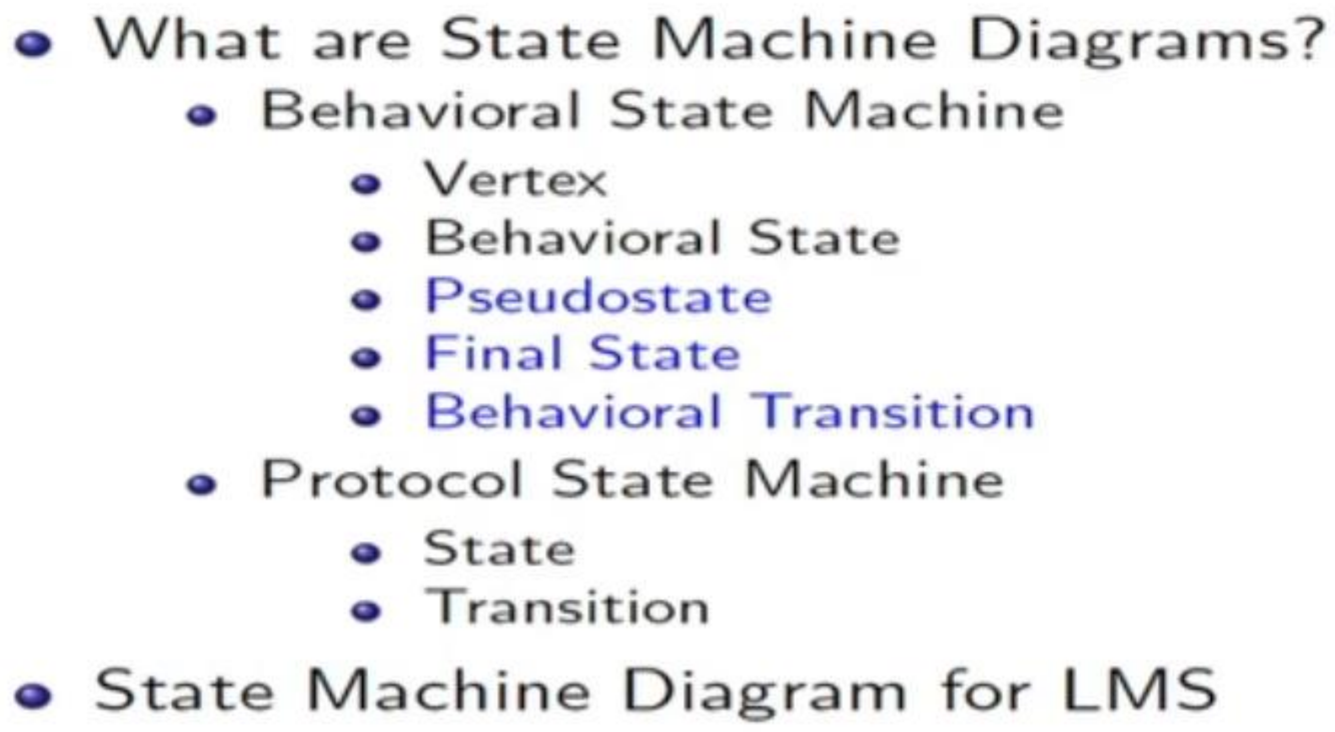


State Machine Diagram

Behavioral State Machine Diagram for Bank ATM



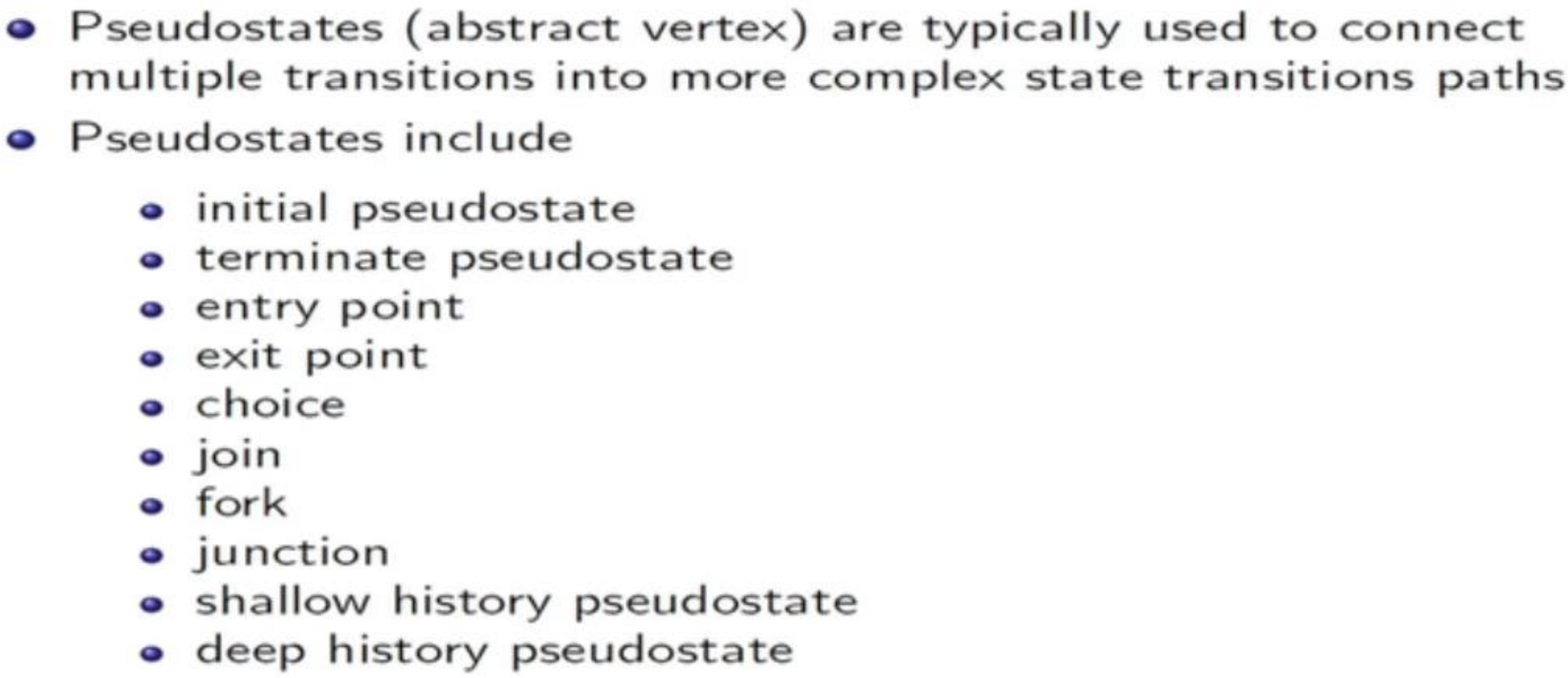
State Machine Diagram



- What are State Machine Diagrams?
  - Behavioral State Machine
    - Vertex
    - Behavioral State
    - Pseudostate
    - Final State
    - Behavioral Transition
  - Protocol State Machine
    - State
    - Transition
- State Machine Diagram for LMS

State Machine Diagram

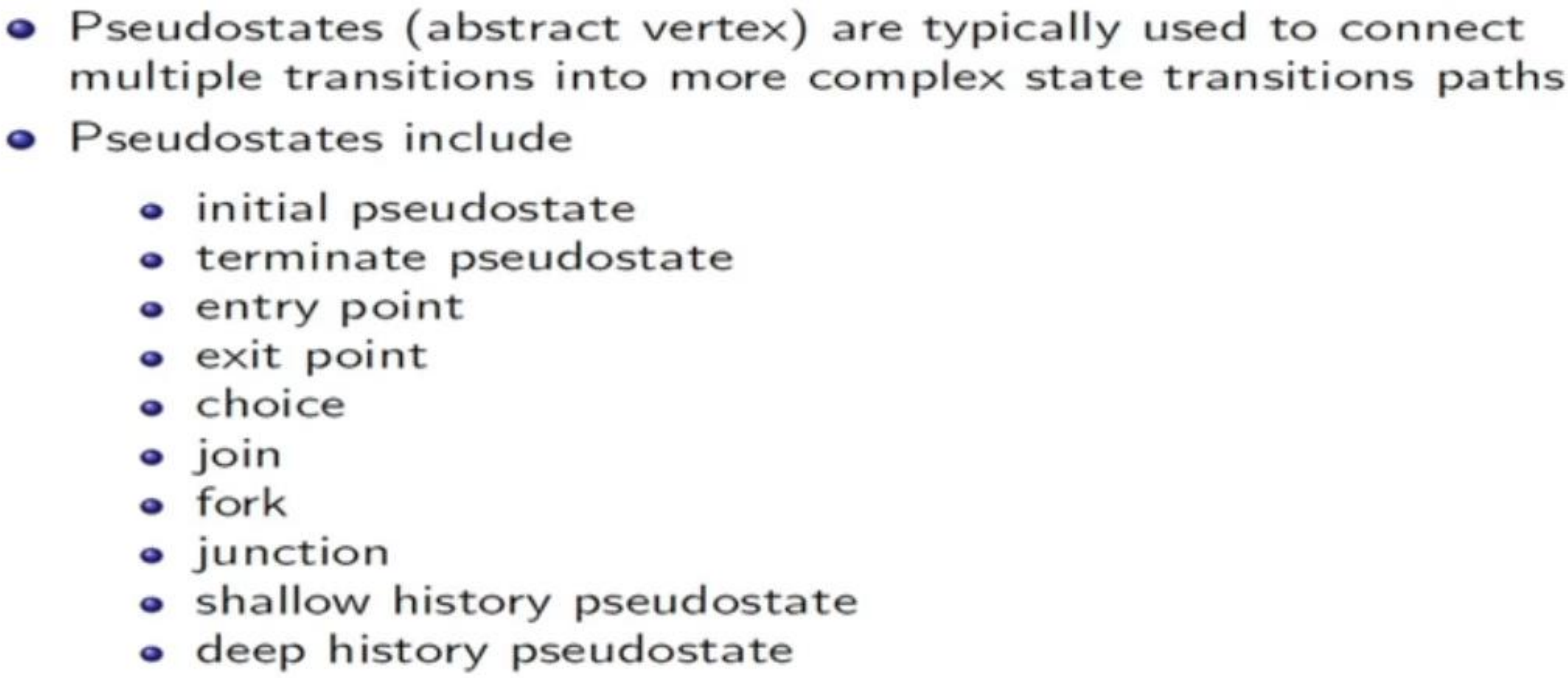
Pseudo state Machine



- Pseudostates (abstract vertex) are typically used to connect multiple transitions into more complex state transitions paths
- Pseudostates include
  - initial pseudostate
  - terminate pseudostate
  - entry point
  - exit point
  - choice
  - join
  - fork
  - junction
  - shallow history pseudostate
  - deep history pseudostate

State Machine Diagram

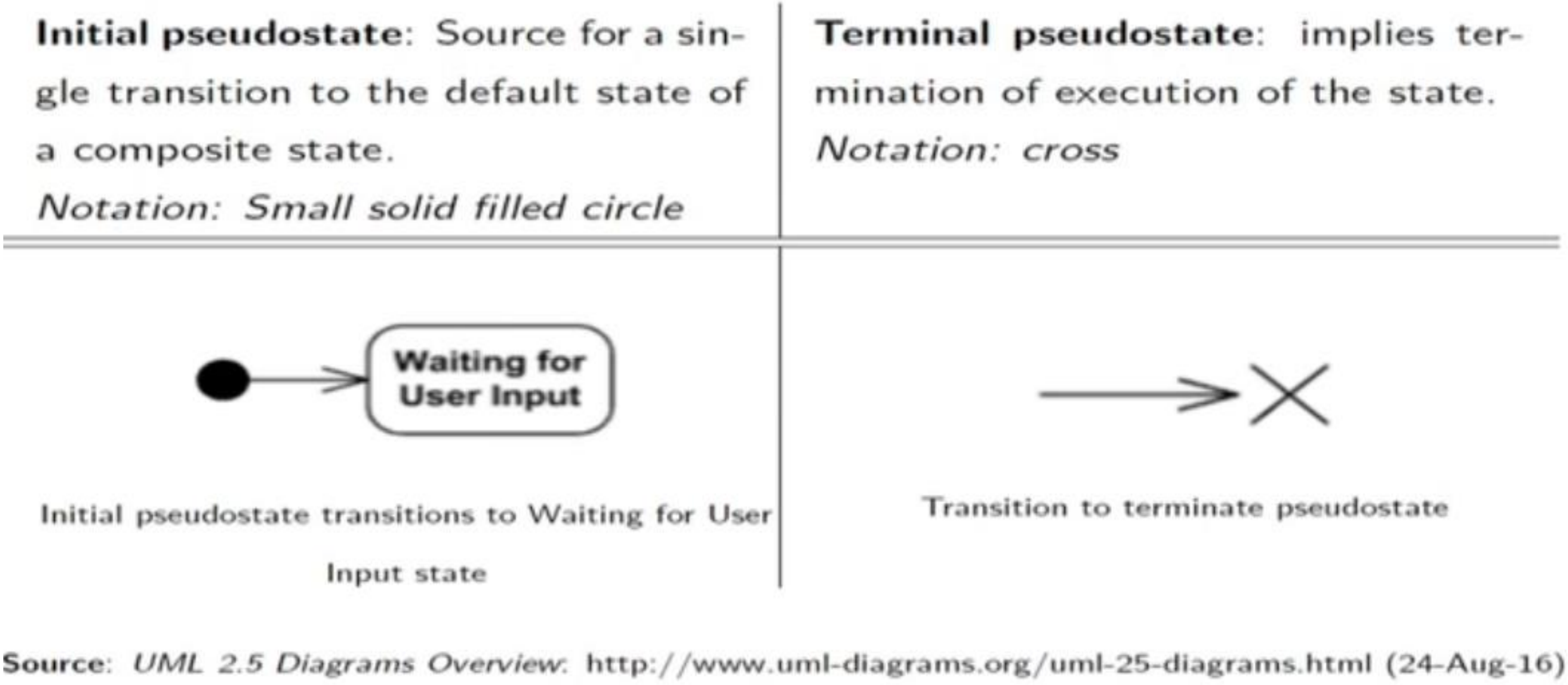
Pseudo state Machine



- Pseudostates (abstract vertex) are typically used to connect multiple transitions into more complex state transitions paths
- Pseudostates include
  - initial pseudostate
  - terminate pseudostate
  - entry point
  - exit point
  - choice
  - join
  - fork
  - junction
  - shallow history pseudostate
  - deep history pseudostate

State Machine Diagram

Pseudo State Machine



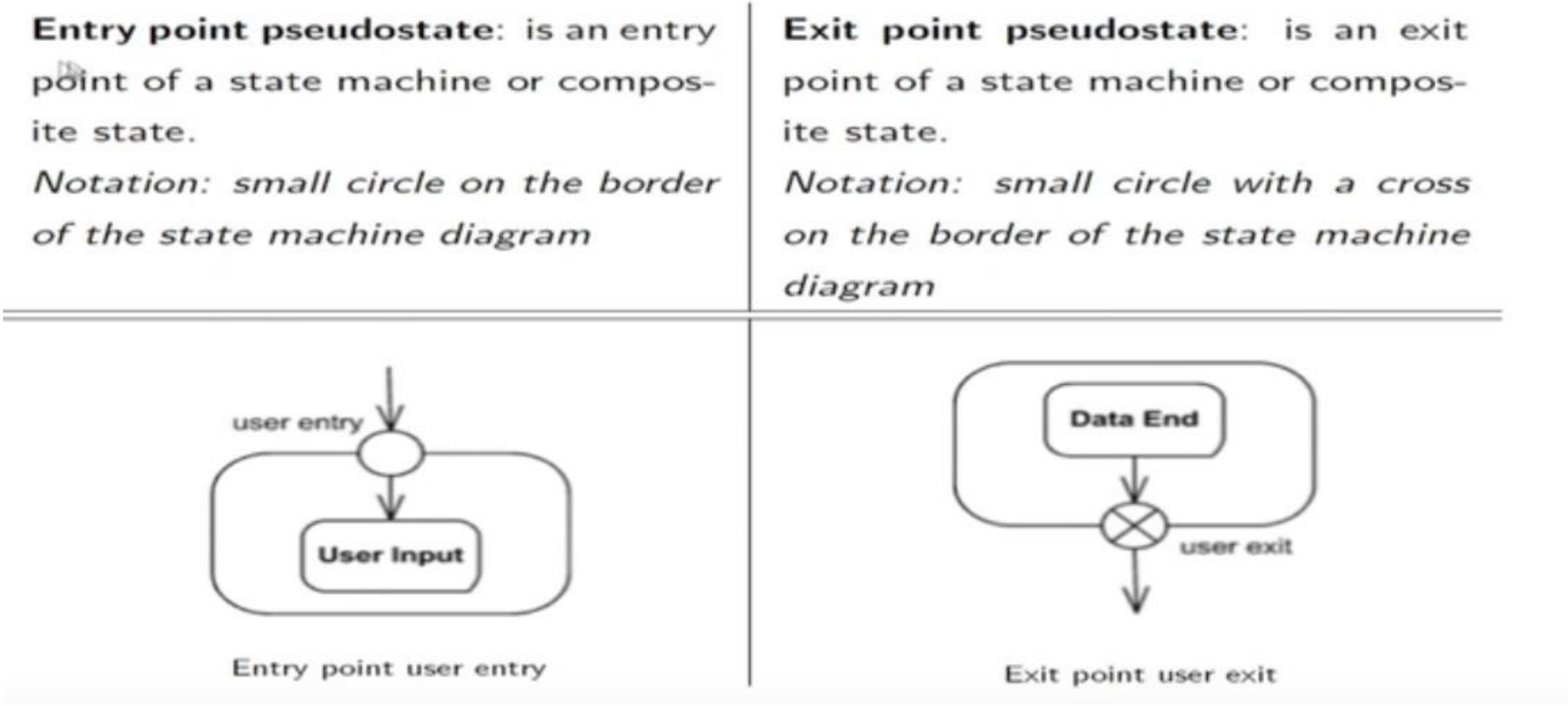
Initial pseudostate: Source for a single transition to the default state of a composite state. *Notation: Small solid filled circle*	**Terminal pseudostate**: implies termination of execution of the state. *Notation: cross*
Initial pseudostate transitions to Waiting for User Input state	Transition to terminate pseudostate

Waiting for User Input

Source: *UML 2.5 Diagrams Overview*: http://www.uml-diagrams.org/uml-25-diagrams.html (24-Aug-16)

State Machine Diagram

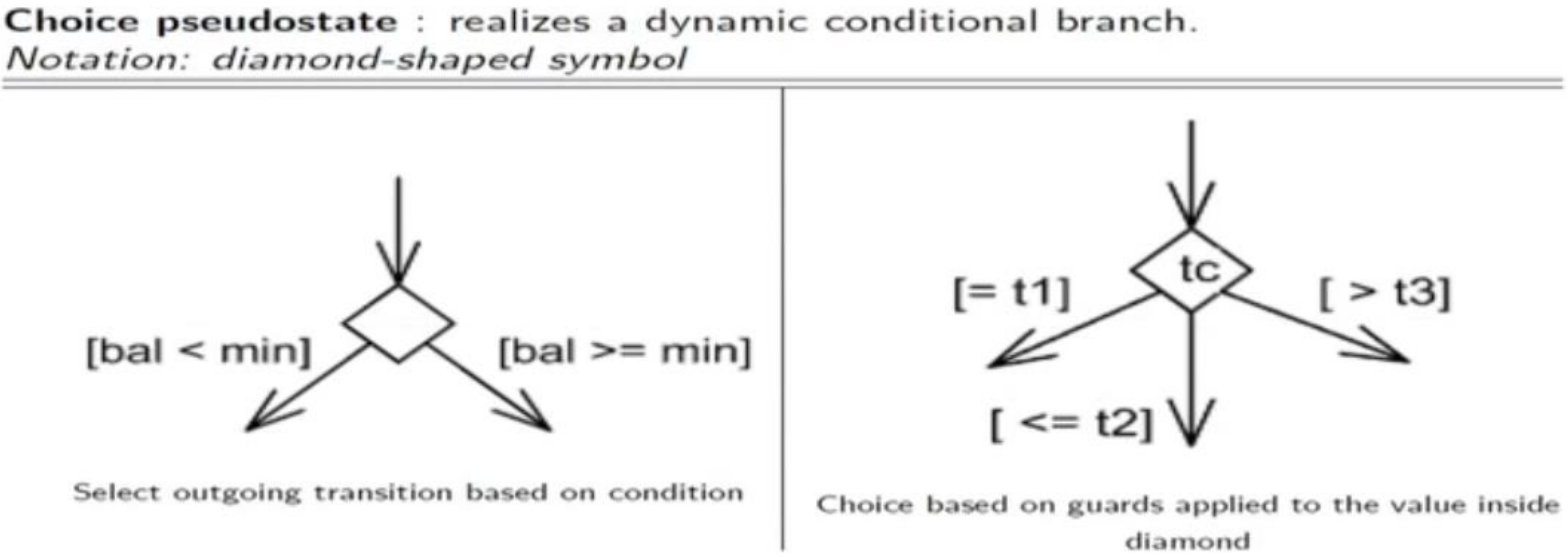
Pseudo State Machine



Entry point pseudostate: is an entry point of a state machine or composite state. *Notation: small circle on the border of the state machine diagram*	**Exit point pseudostate:** is an exit point of a state machine or composite state. *Notation: small circle with a cross on the border of the state machine diagram*
Entry point user entry	Exit point user exit

State Machine Diagram

Pseudo State Machine



Choice pseudostate : realizes a dynamic conditional branch.

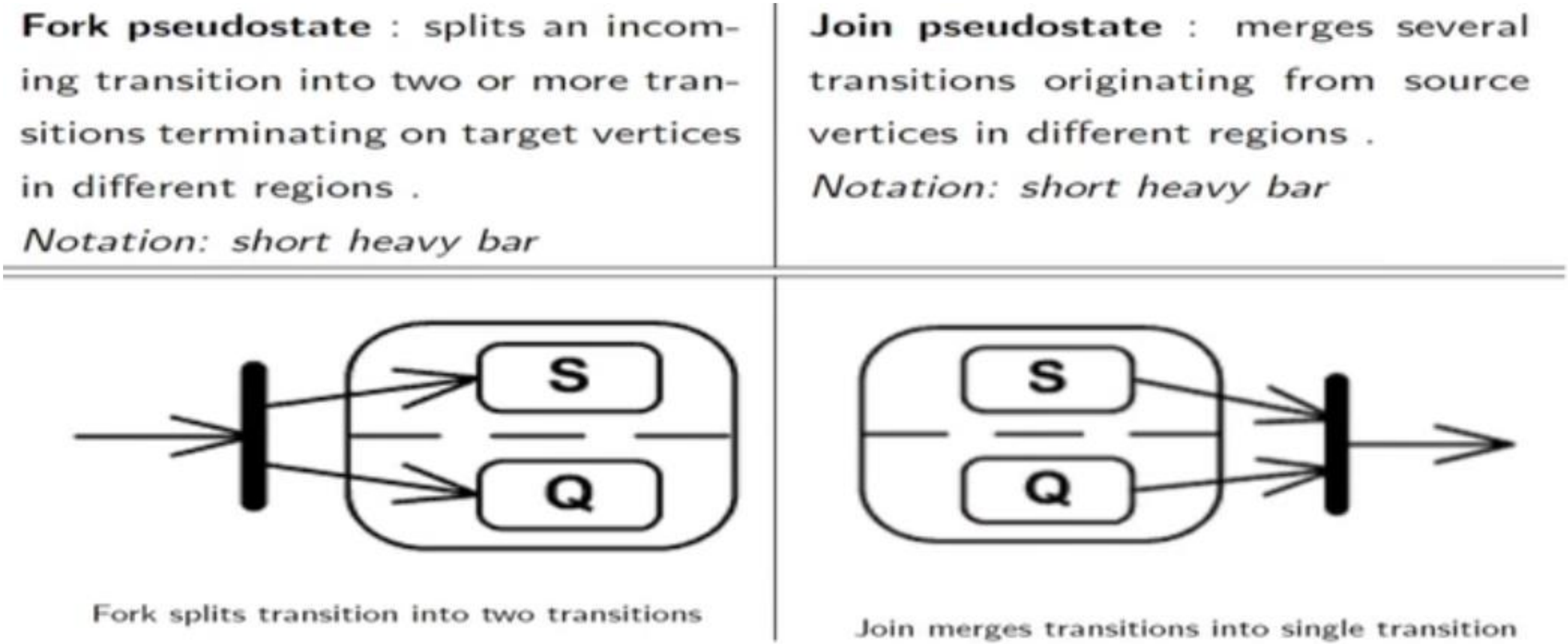
Notation: diamond-shaped symbol

Select outgoing transition based on condition

Choice based on guards applied to the value inside diamond

State Machine Diagram

Pseudo State Machine



Fork pseudostate : splits an incoming transition into two or more transitions terminating on target vertices in different regions .
Notation: short heavy bar

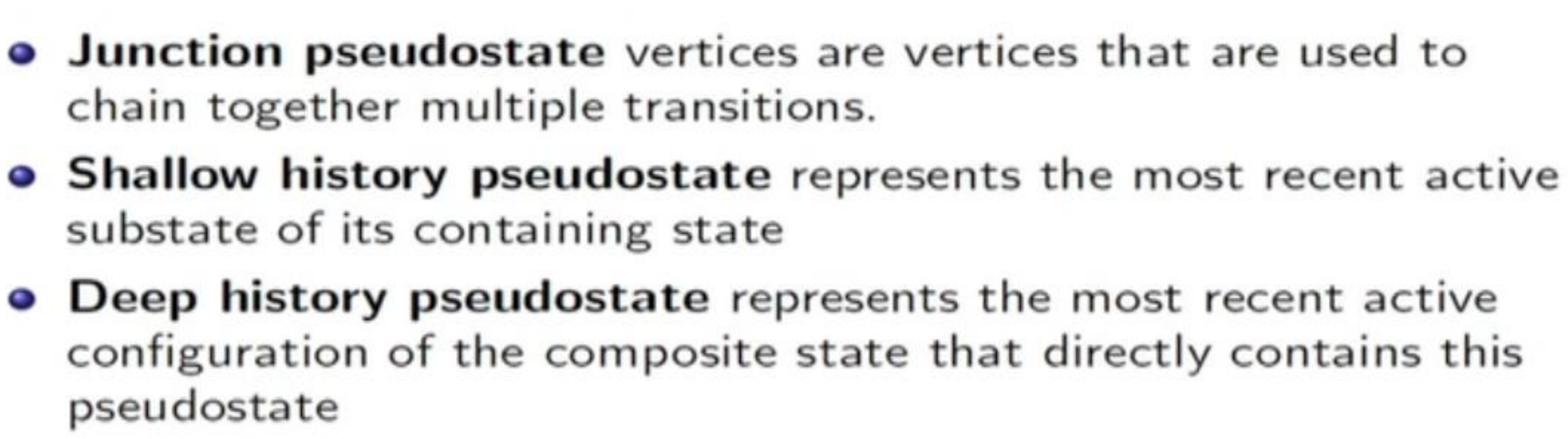
Join pseudostate : merges several transitions originating from source vertices in different regions .
Notation: short heavy bar

Fork splits transition into two transitions

Join merges transitions into single transition

State Machine Diagram

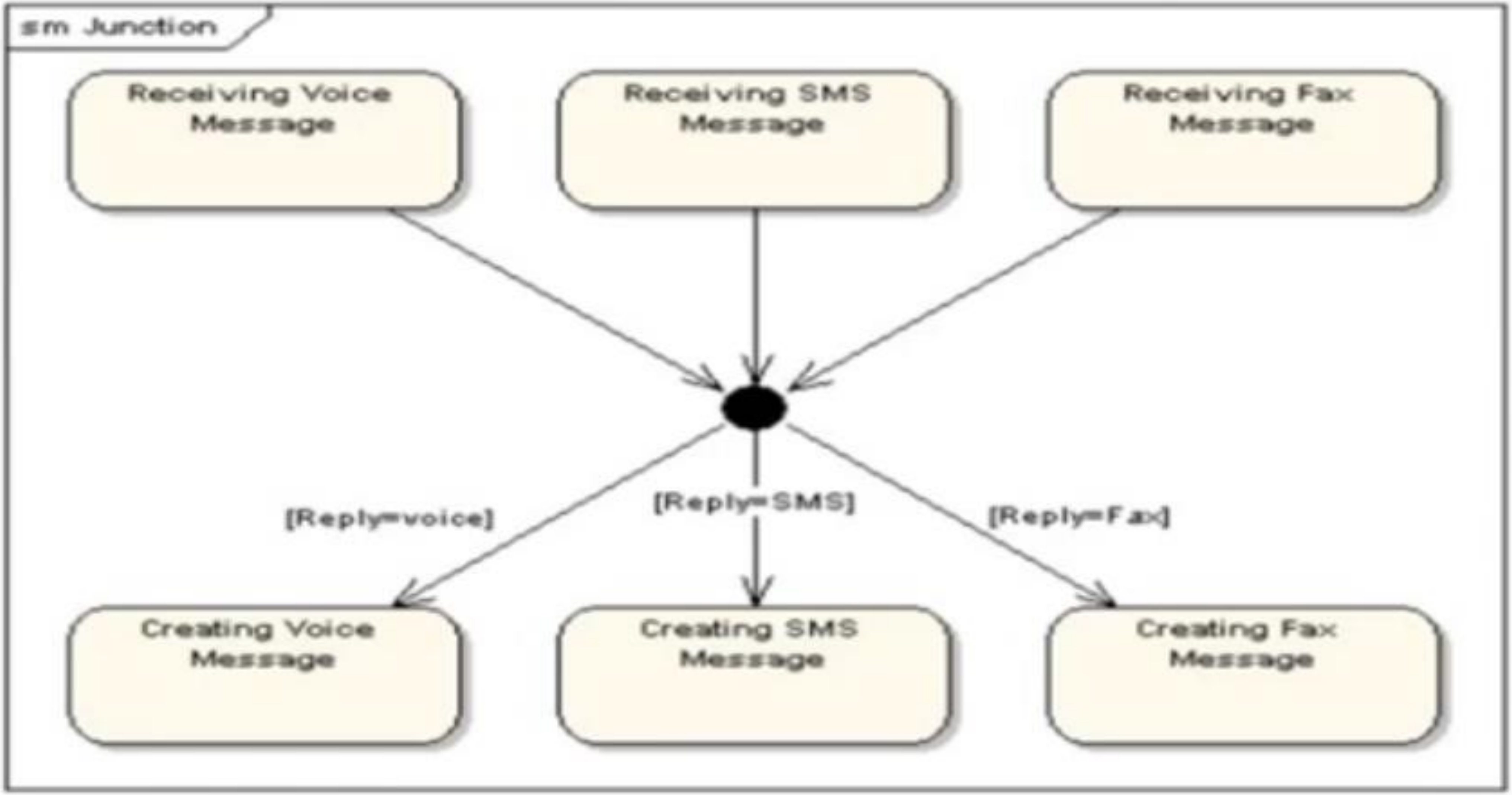
Pseudo State Machine



- **Junction pseudostate** vertices are vertices that are used to chain together multiple transitions.
- **Shallow history pseudostate** represents the most recent active substate of its containing state
- **Deep history pseudostate** represents the most recent active configuration of the composite state that directly contains this pseudostate

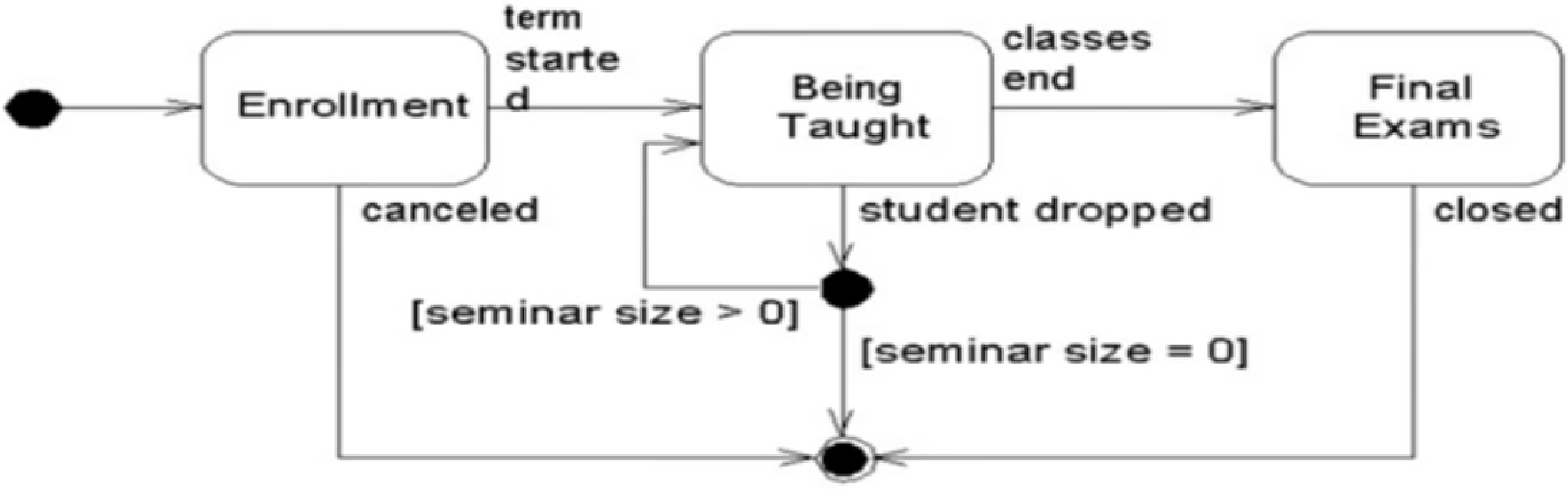
State Machine Diagram

Pseudo State Machine – Junction Pseudo State



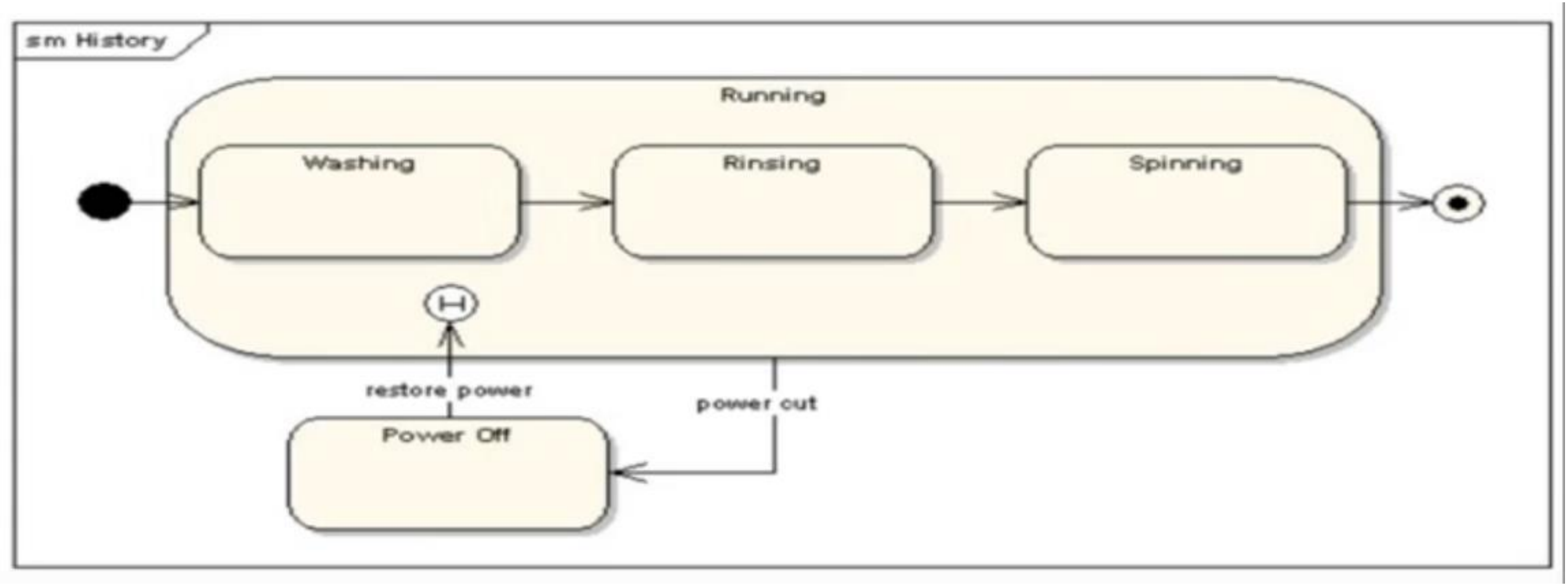
State Machine Diagram

Pseudo State Machine – Junction Pseudo State



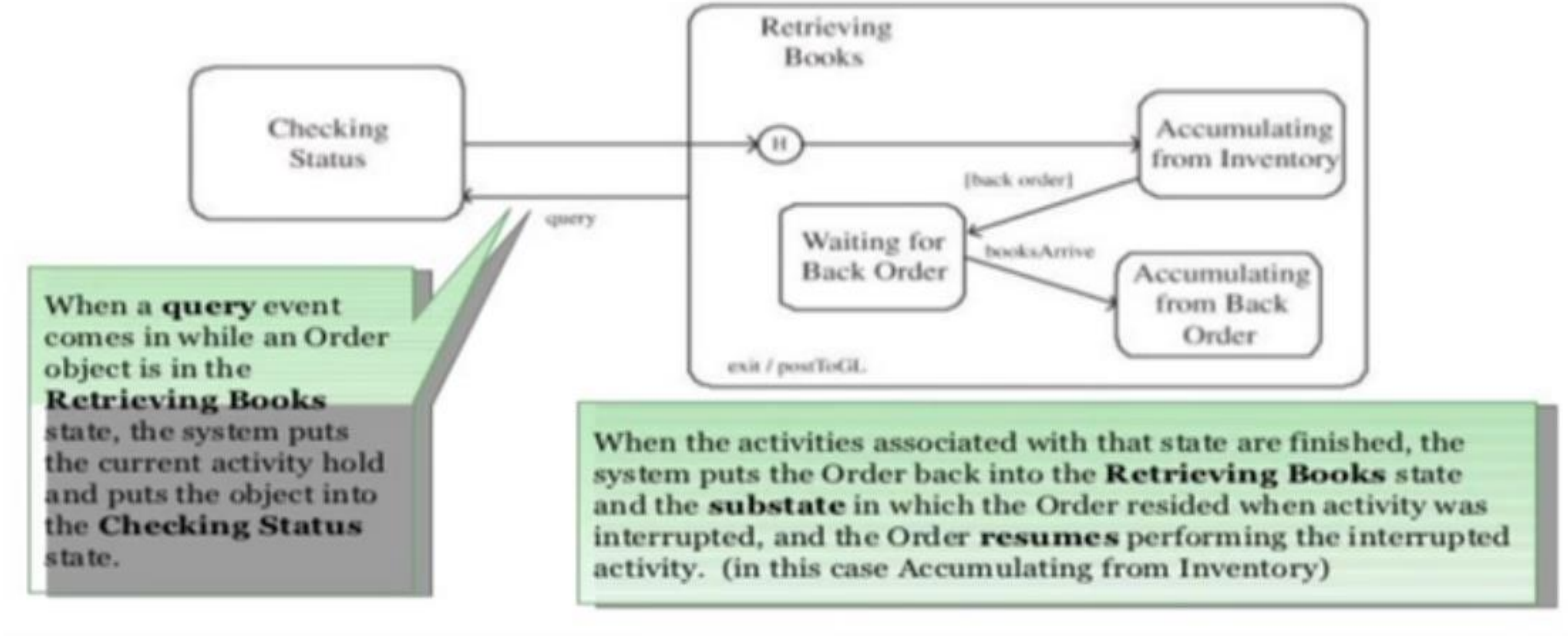
State Machine Diagram

Pseudo State Machine – Shallow History Pseudo State



State Machine Diagram

Pseudo State Machine – Shallow History Pseudo State

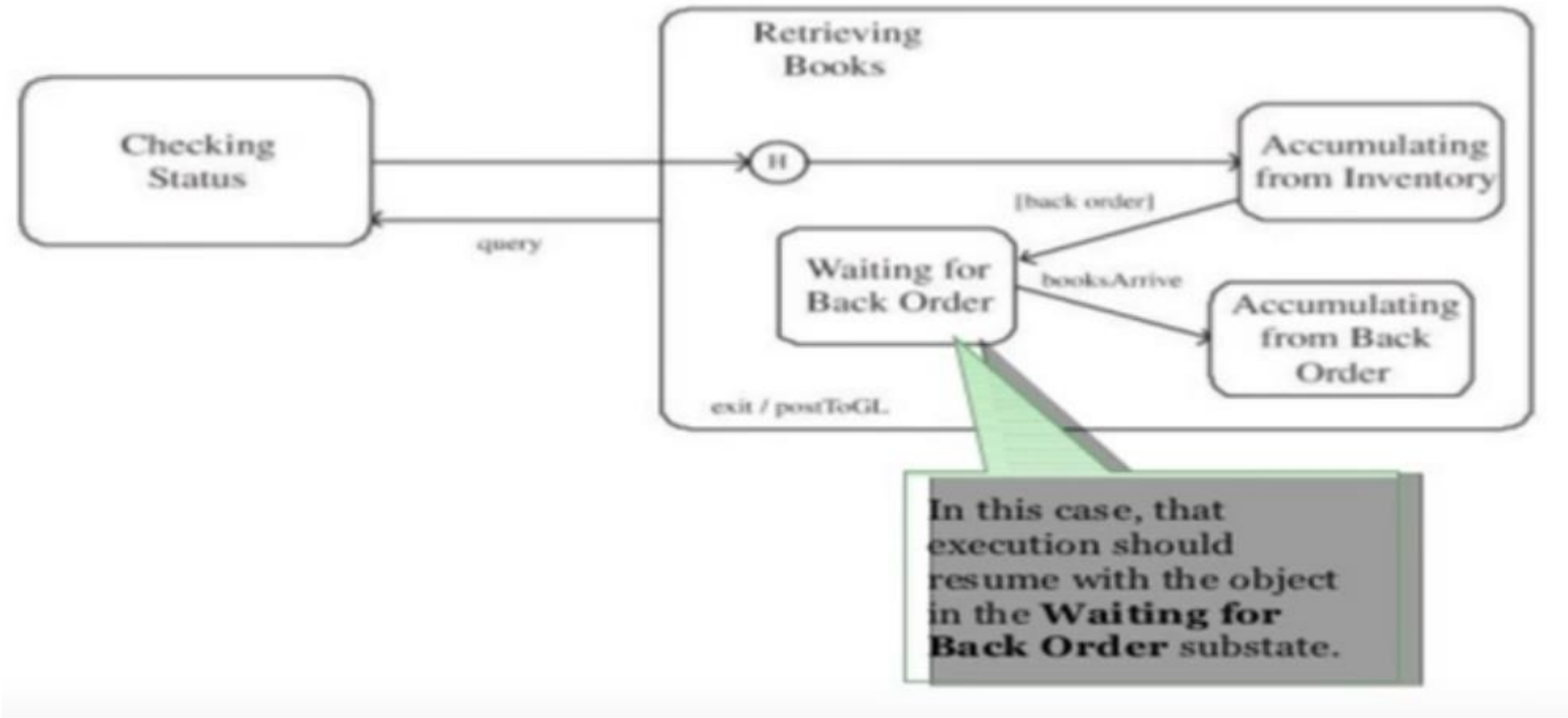


When a **query** event comes in while an Order object is in the **Retrieving Books** state, the system puts the current activity hold and puts the object into the **Checking Status** state.

When the activities associated with that state are finished, the system puts the Order back into the **Retrieving Books** state and the **substate** in which the Order resided when activity was interrupted, and the Order **resumes** performing the interrupted activity. (in this case Accumulating from Inventory)

State Machine Diagram

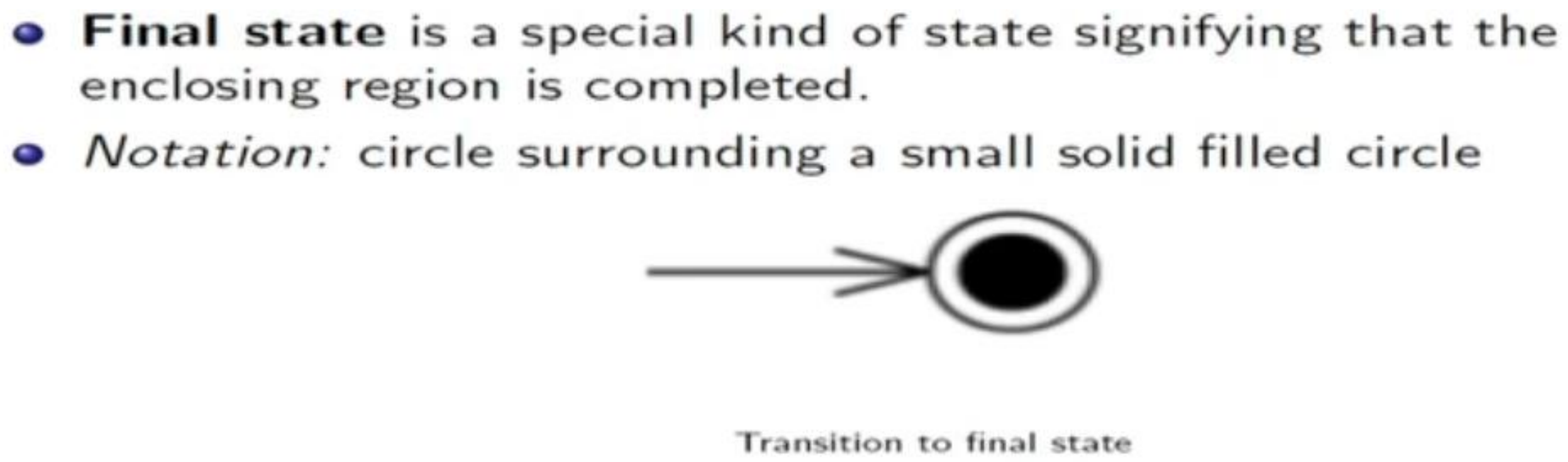
Pseudo State Machine – Deep History Pseudo State



In this case, that execution should resume with the object in the **Waiting for Back Order** substate.

State Machine Diagram

Final State

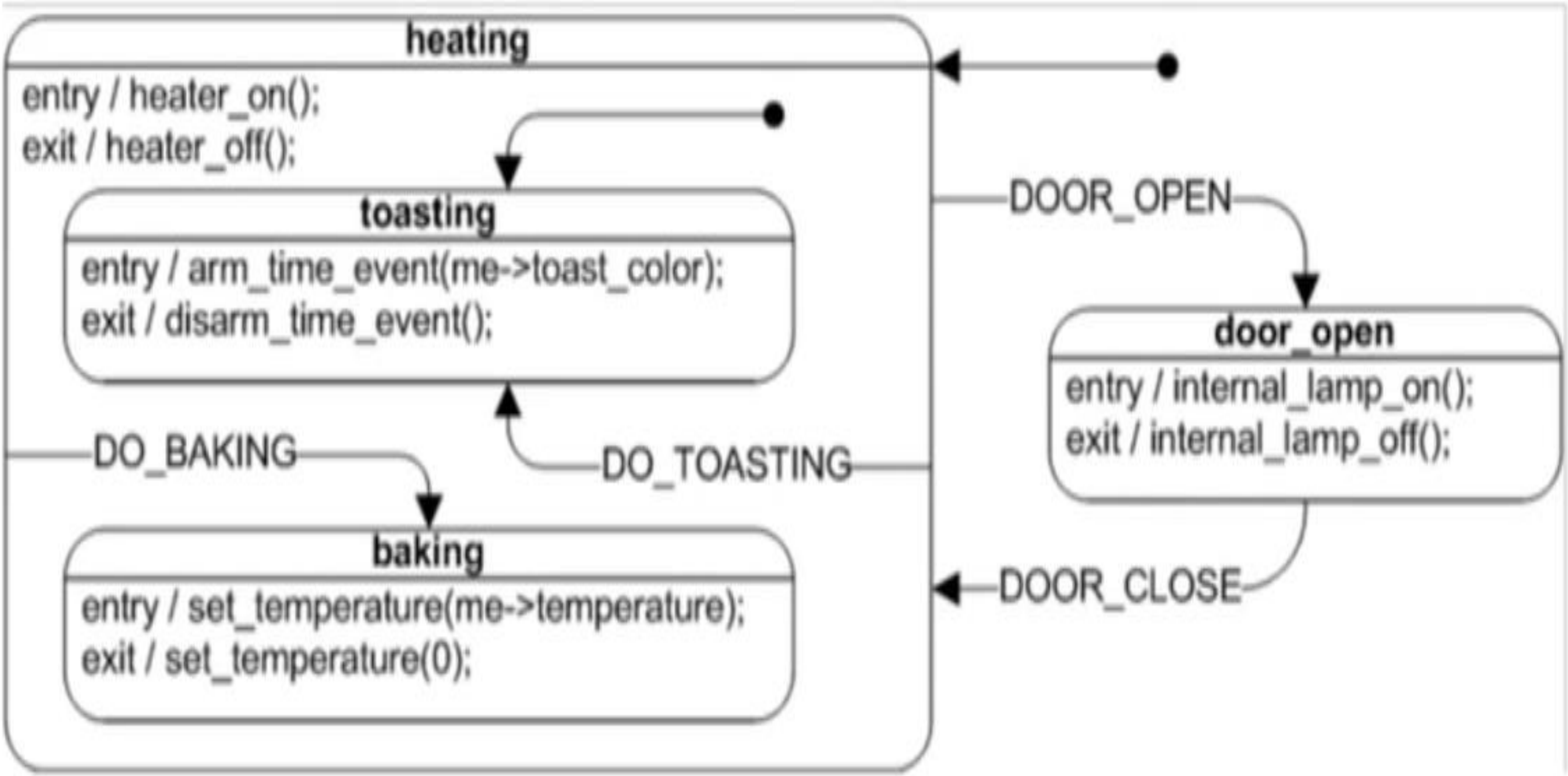


- **Final state** is a special kind of state signifying that the enclosing region is completed.
- *Notation:* circle surrounding a small solid filled circle

Transition to final state

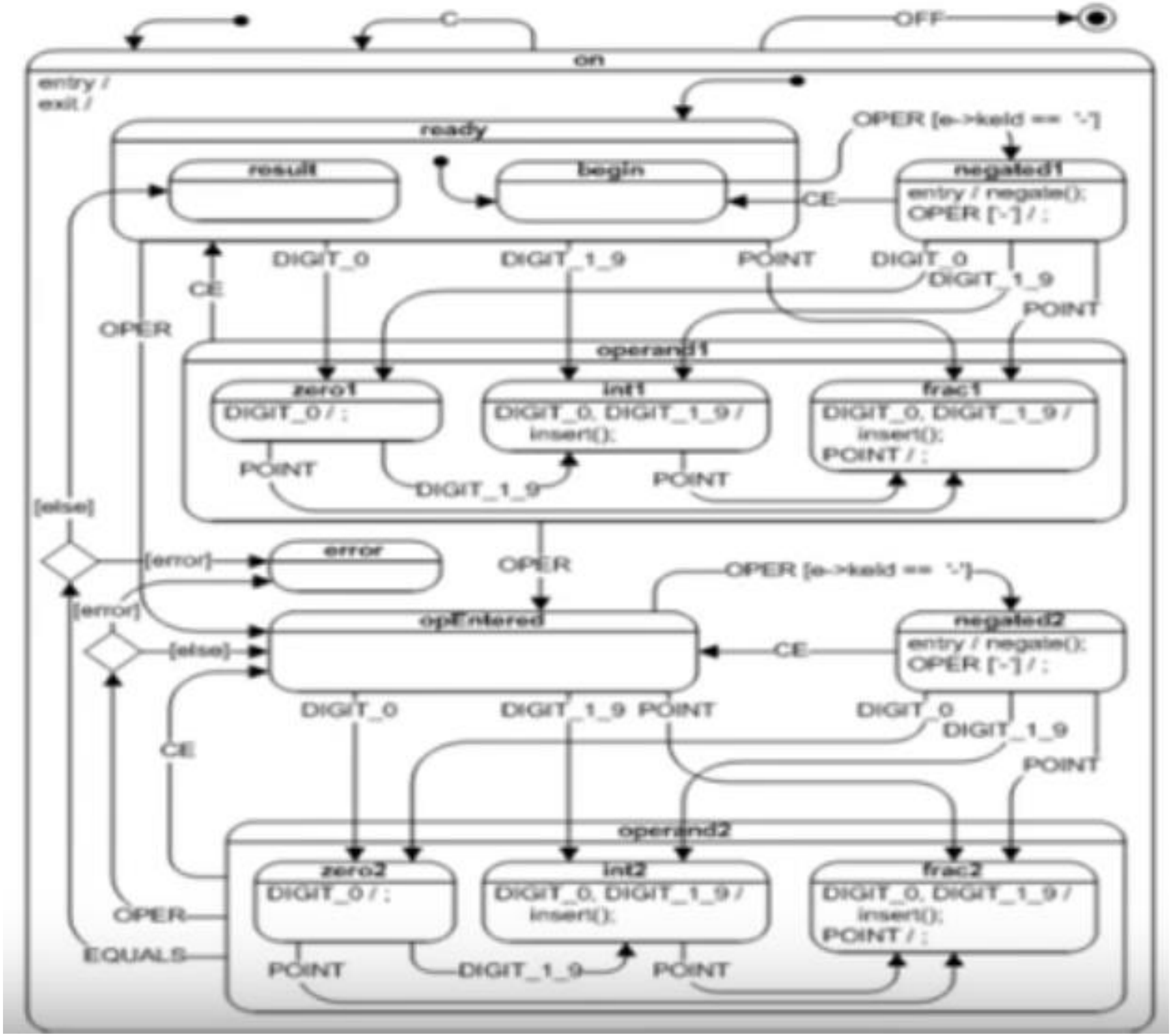
State Machine Diagram

Toaster- Oven States


heating
entry / heater_on();
exit / heater_off();
toasting
entry / arm_time_event(me->toast_color);
exit / disarm_time_event();
DO_BAKING
DO_TOASTING
baking
entry / set_temperature(me->temperature);
exit / set_temperature(0);
DOOR_OPEN
door_open
entry / internal_lamp_on();
exit / internal_lamp_off();
DOOR_CLOSE

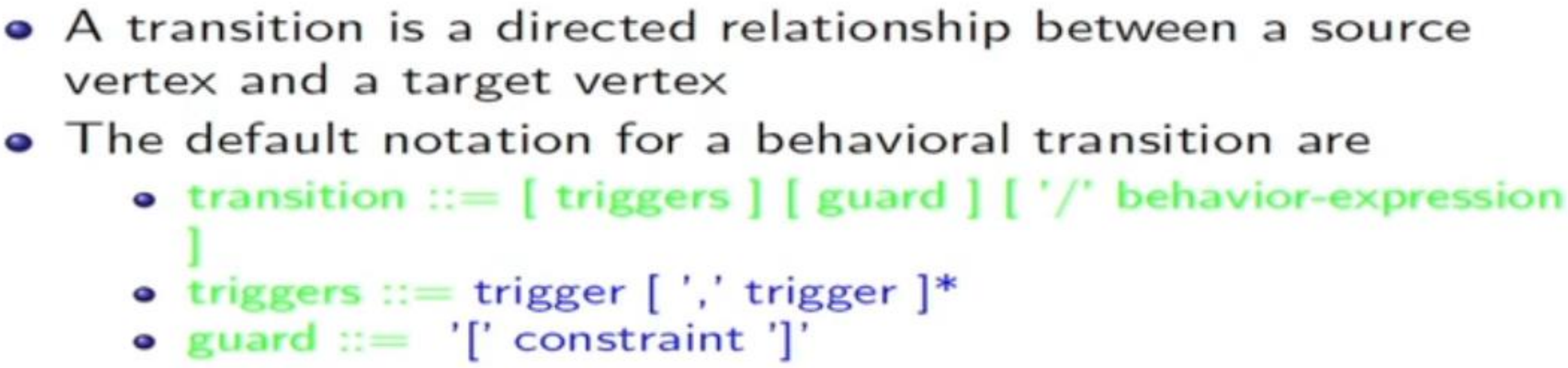
State Machine Diagram

Calculator



State Machine Diagram

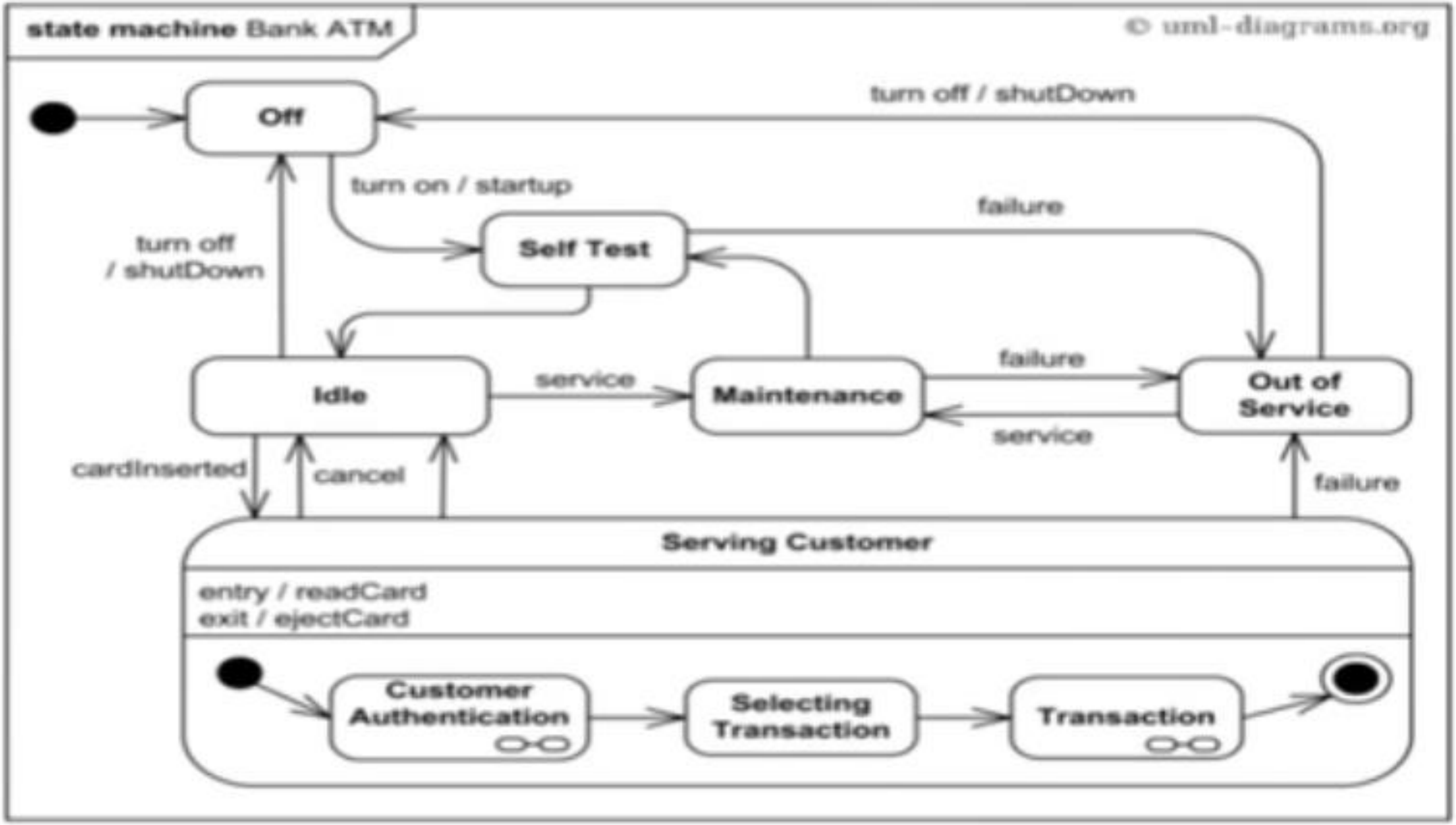
Behavioral Transition



- A transition is a directed relationship between a source vertex and a target vertex
- The default notation for a behavioral transition are
  - transition ::= [triggers] [guard] ['/' behavior-expression]
  - triggers ::= trigger [',' trigger]*
  - guard ::= '[' constraint ']'

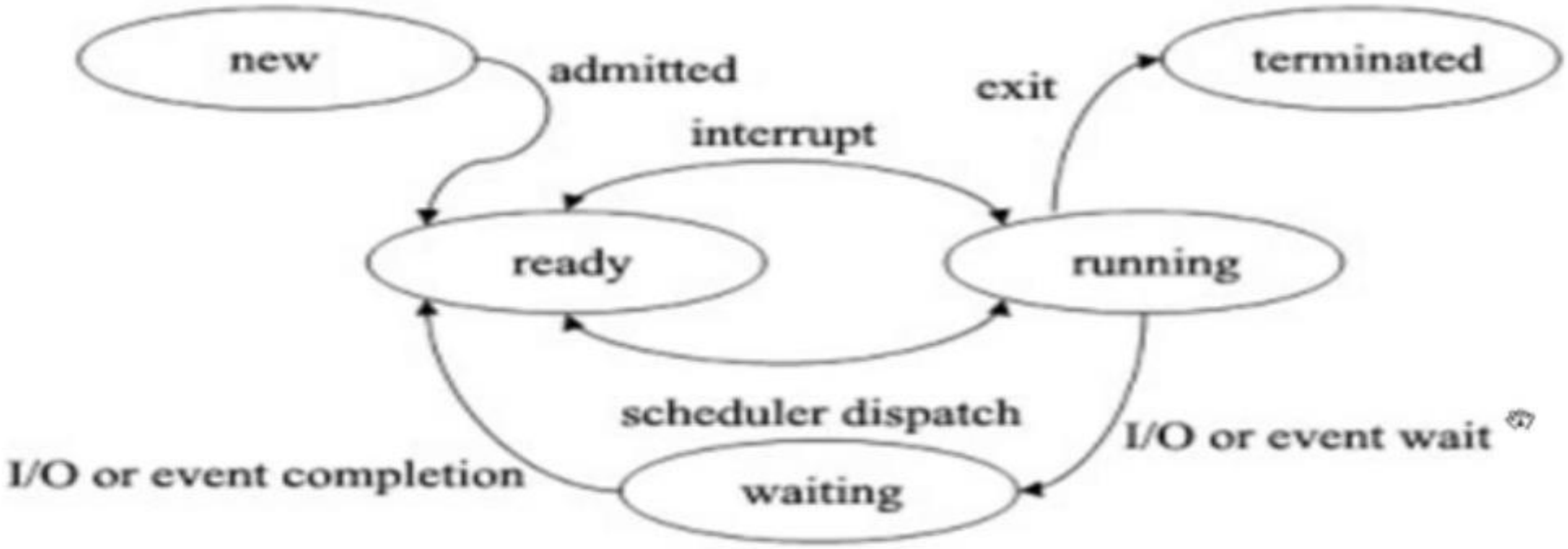
State Machine Diagram

Behavioral State Diagram – Bank ATM

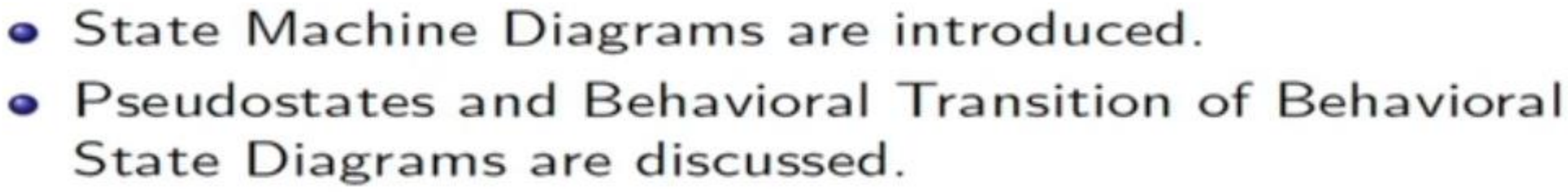


State Machine Diagram

Behavioral State Diagram – OS Process

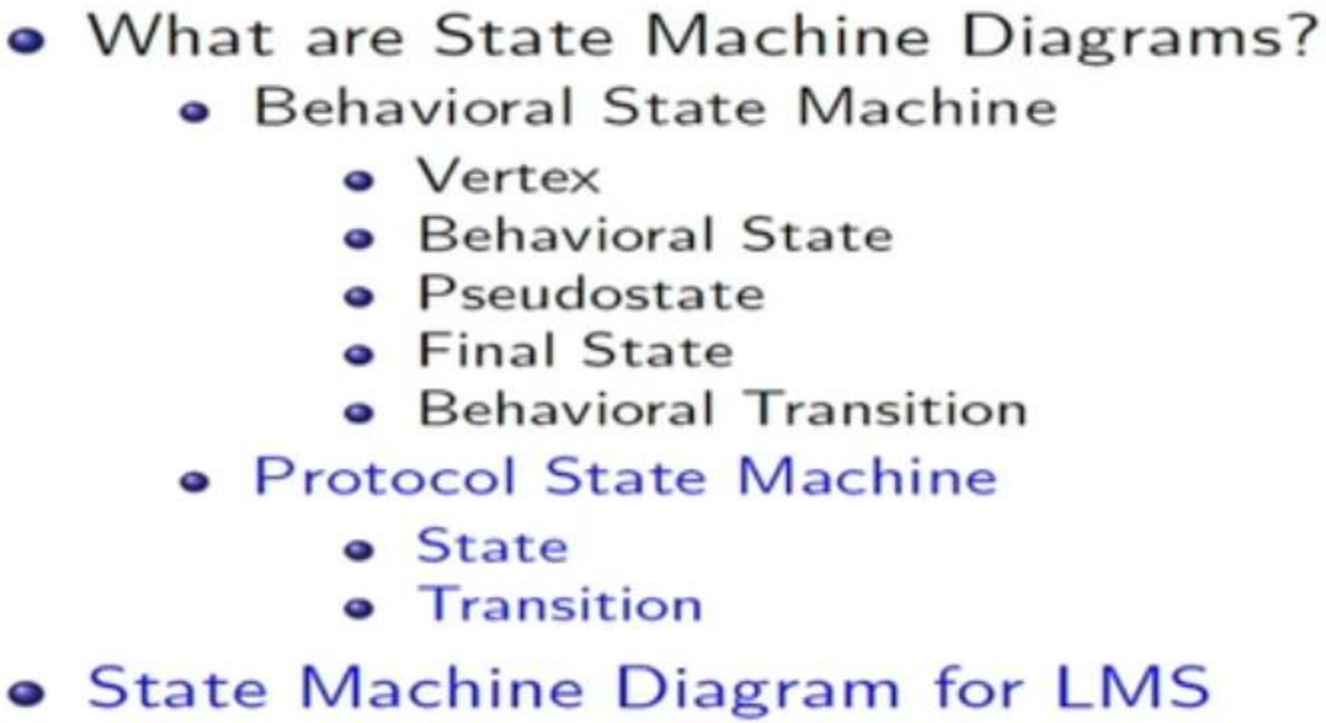


State Machine Diagram



- State Machine Diagrams are introduced.
- Pseudostates and Behavioral Transition of Behavioral State Diagrams are discussed.

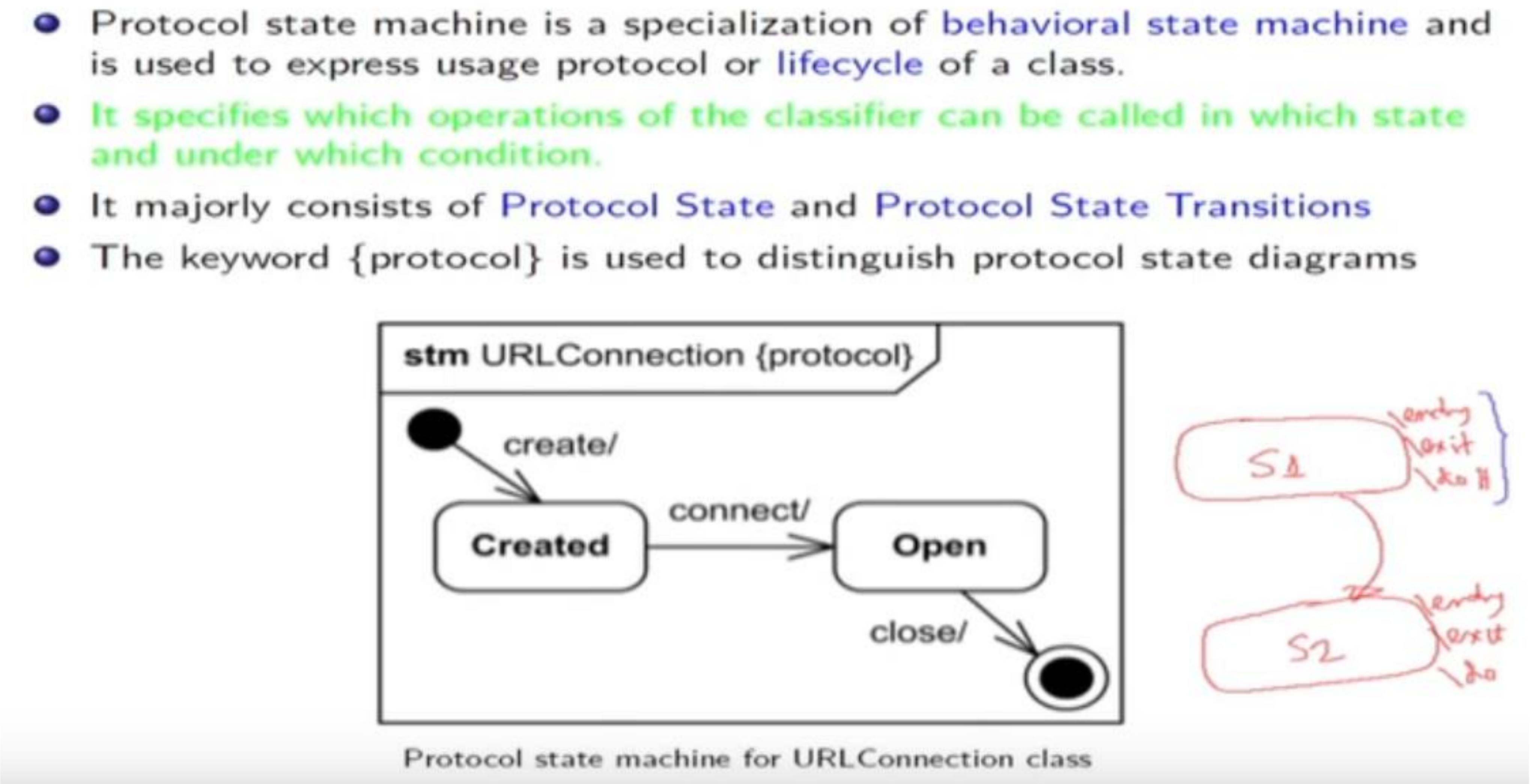
State Machine Diagram



- What are State Machine Diagrams?
    - Behavioral State Machine
        - Vertex
        - Behavioral State
        - Pseudostate
        - Final State
        - Behavioral Transition
    - Protocol State Machine
        - State
        - Transition
- State Machine Diagram for LMS

State Machine Diagram

Protocol State Machine – URL Connection Class

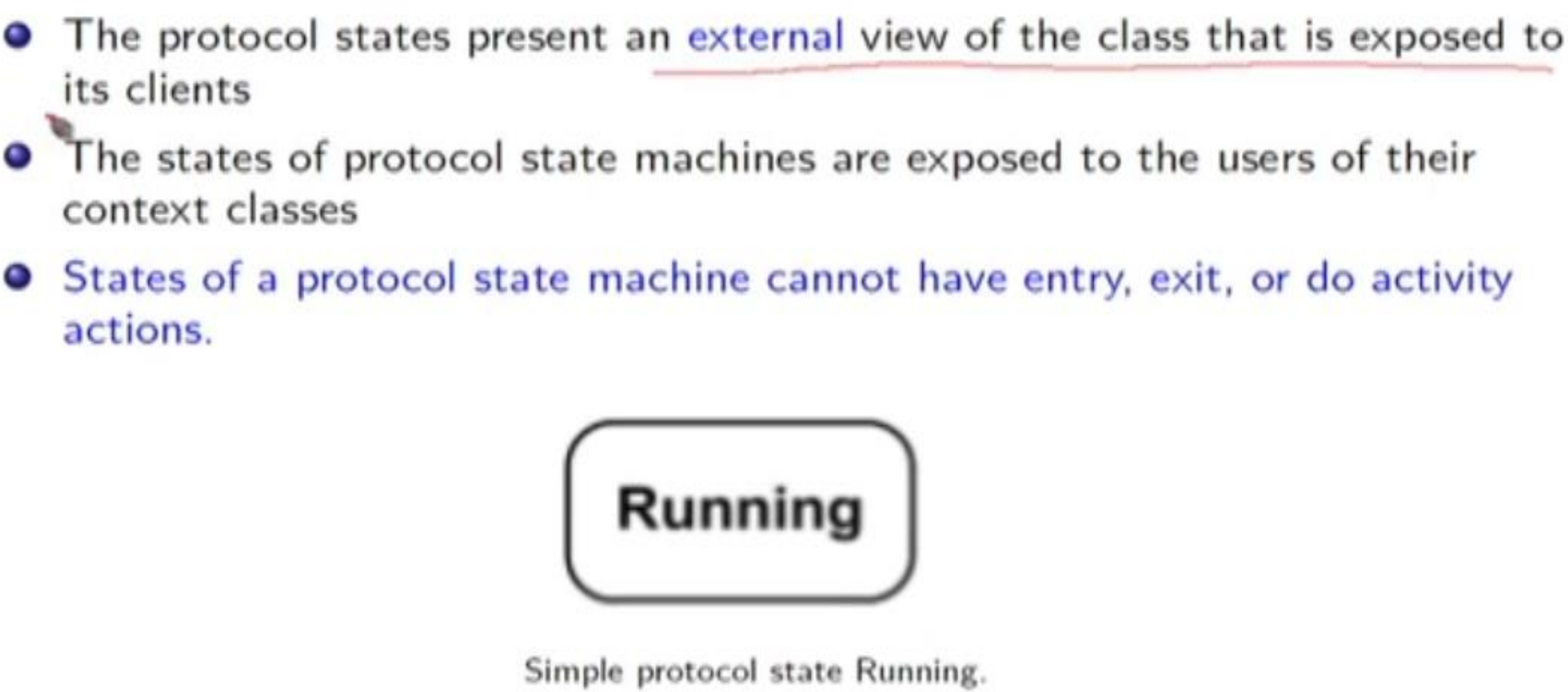


- Protocol state machine is a specialization of behavioral state machine and is used to express usage protocol or lifecycle of a class.
- It specifies which operations of the classifier can be called in which state and under which condition.
- It majorly consists of Protocol State and Protocol State Transitions
- The keyword {protocol} is used to distinguish protocol state diagrams

Protocol state machine for URLConnection class

State Machine Diagram

Protocol State Machine – Protocol State

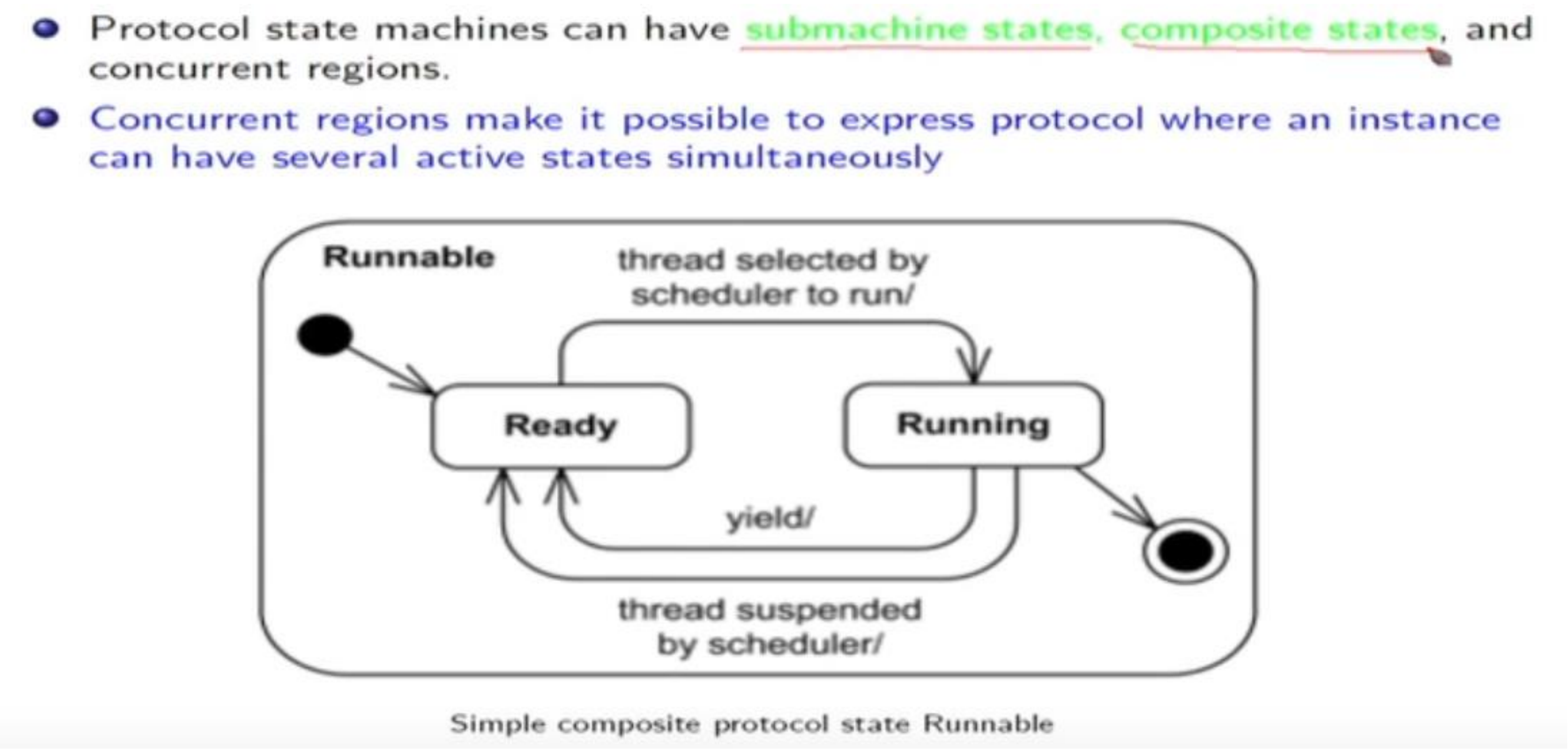


- The protocol states present an external view of the class that is exposed to its clients
- The states of protocol state machines are exposed to the users of their context classes
- States of a protocol state machine cannot have entry, exit, or do activity actions.

Simple protocol state Running.

State Machine Diagram

Protocol State Machine – Protocol State



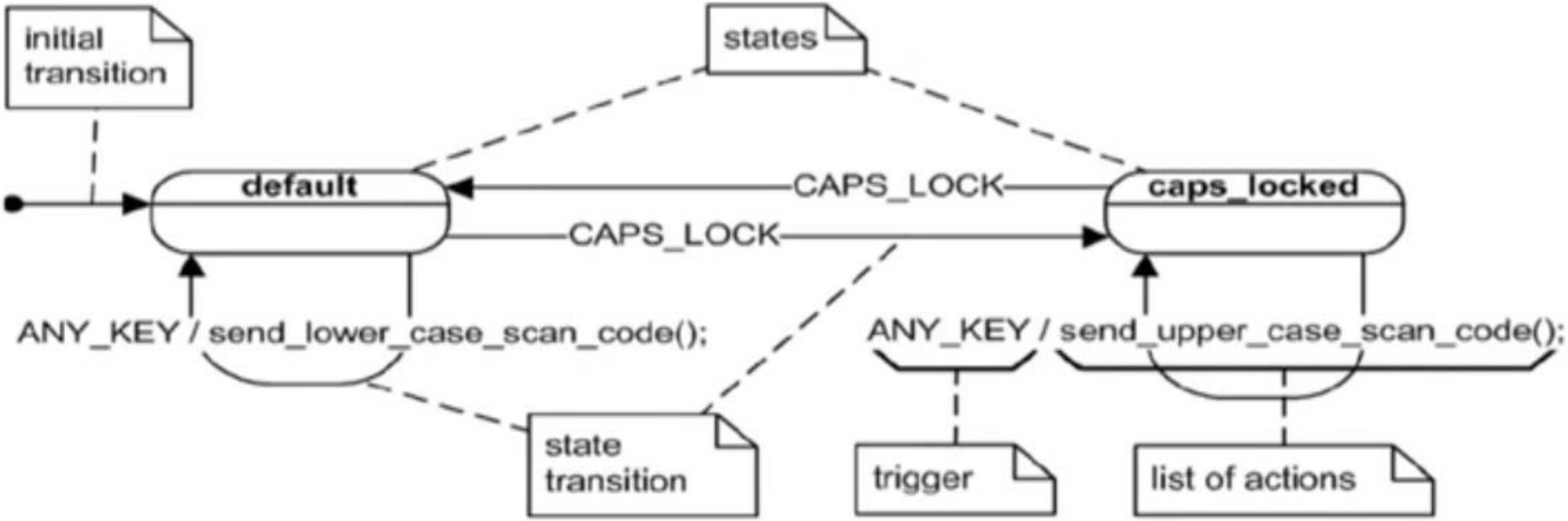
- Protocol state machines can have submachine states, composite states, and concurrent regions.
- Concurrent regions make it possible to express protocol where an instance can have several active states simultaneously

Runnable
thread selected by scheduler to run/
Ready
Running
yield/
thread suspended by scheduler/

Simple composite protocol state Runnable

State Machine Diagram

Protocol State Machine – Protocol State – Keyboard Operation

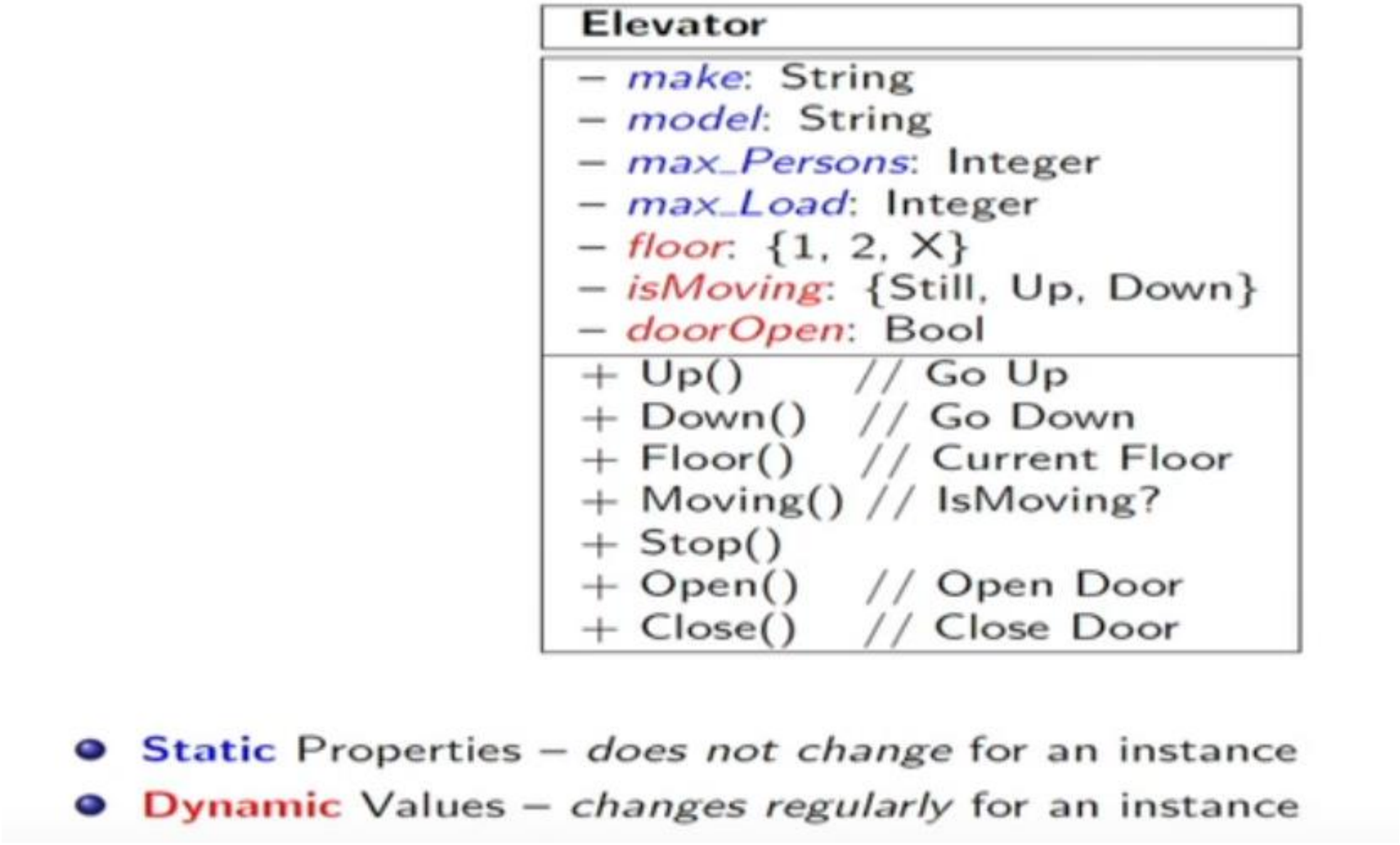


Source: *url:*

https://en.wikipedia.org/wiki/UML_state_machine#/media/File:UML_state_machine_Fig1.png(24-Aug-16)

State Machine Diagram

Protocol State Machine – Protocol State -An Elevator



Elevator

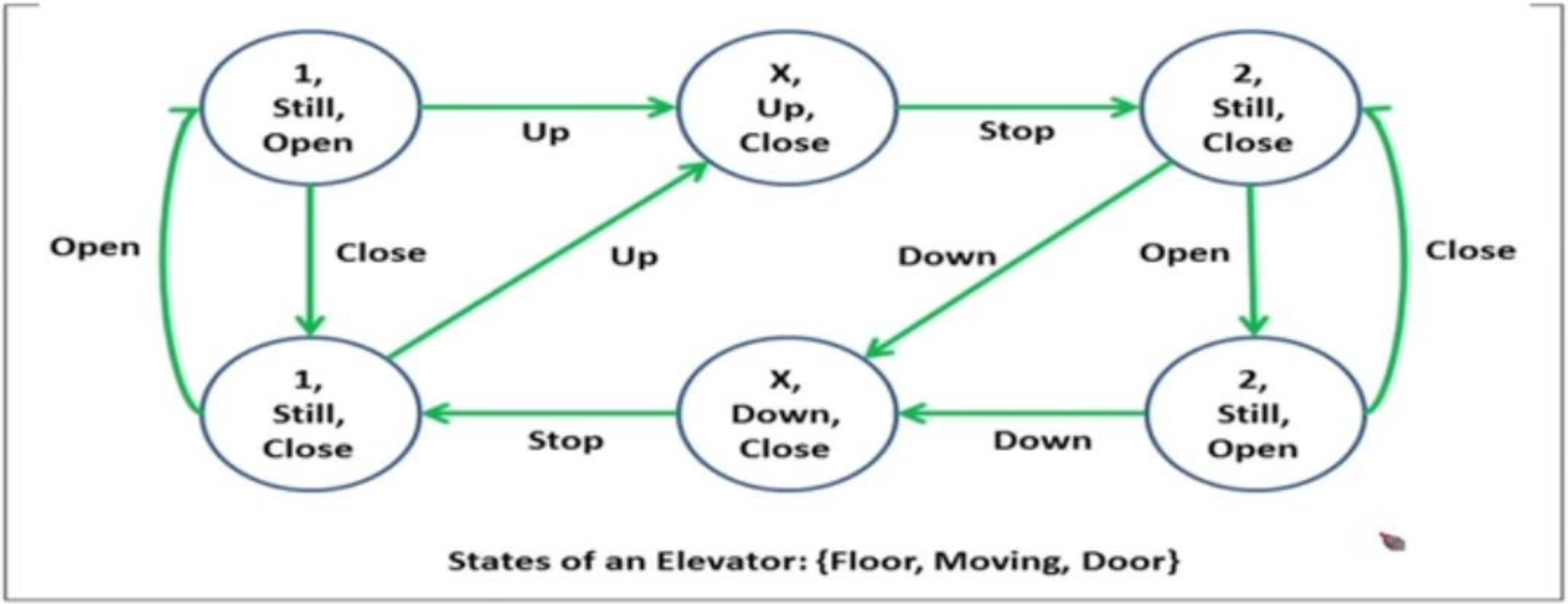
− *make*: String
− *model*: String
− *max_Persons*: Integer
− *max_Load*: Integer
− *floor*: {1, 2, X}
− *isMoving*: {Still, Up, Down}
− *doorOpen*: Bool

+ Up() // Go Up
+ Down() // Go Down
+ Floor() // Current Floor
+ Moving() // IsMoving?
+ Stop()
+ Open() // Open Door
+ Close() // Close Door

- Static Properties – *does not change* for an instance
- Dynamic Values – *changes regularly* for an instance

State Machine Diagram

Protocol State Machine – Protocol State -An Elevator States

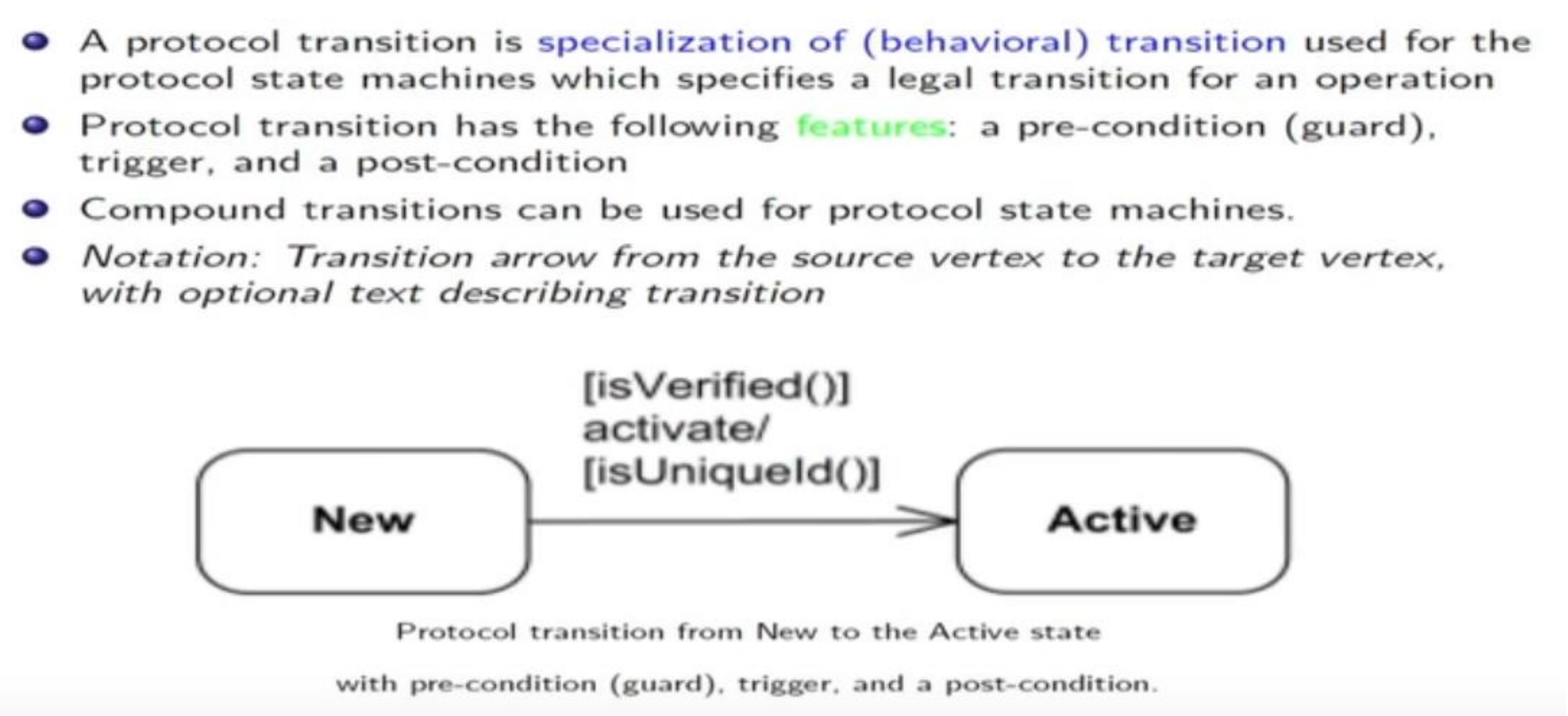


States of an Elevator: {Floor, Moving, Door}

The elevator moves through these states as it operates

State Machine Diagram

Protocol State Machine - Protocol Transition



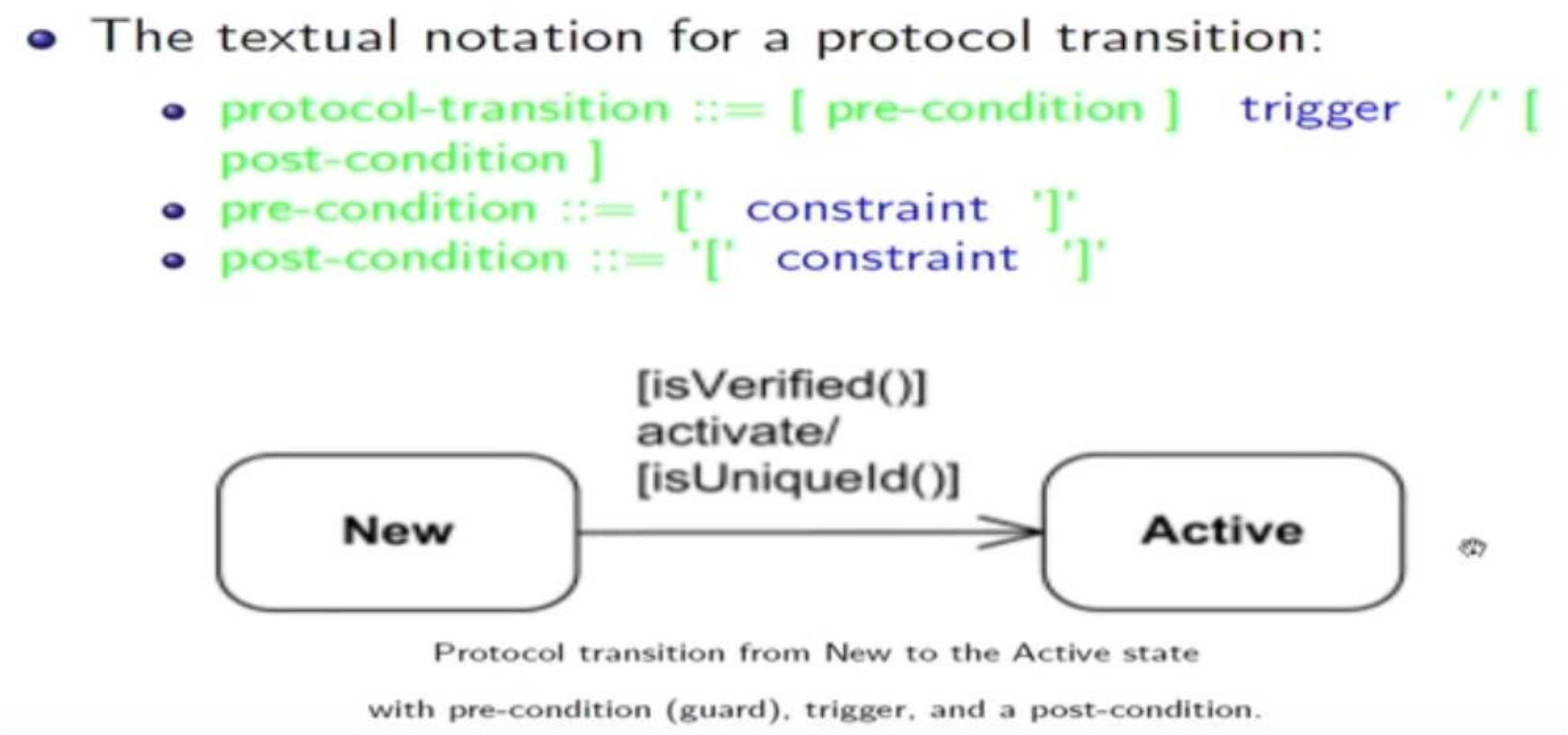
- A protocol transition is specialization of (behavioral) transition used for the protocol state machines which specifies a legal transition for an operation
- Protocol transition has the following features: a pre-condition (guard), trigger, and a post-condition
- Compound transitions can be used for protocol state machines.
- *Notation: Transition arrow from the source vertex to the target vertex, with optional text describing transition*

Protocol transition from New to the Active state

with pre-condition (guard), trigger, and a post-condition.

State Machine Diagram

Protocol State Machine - Protocol Transition



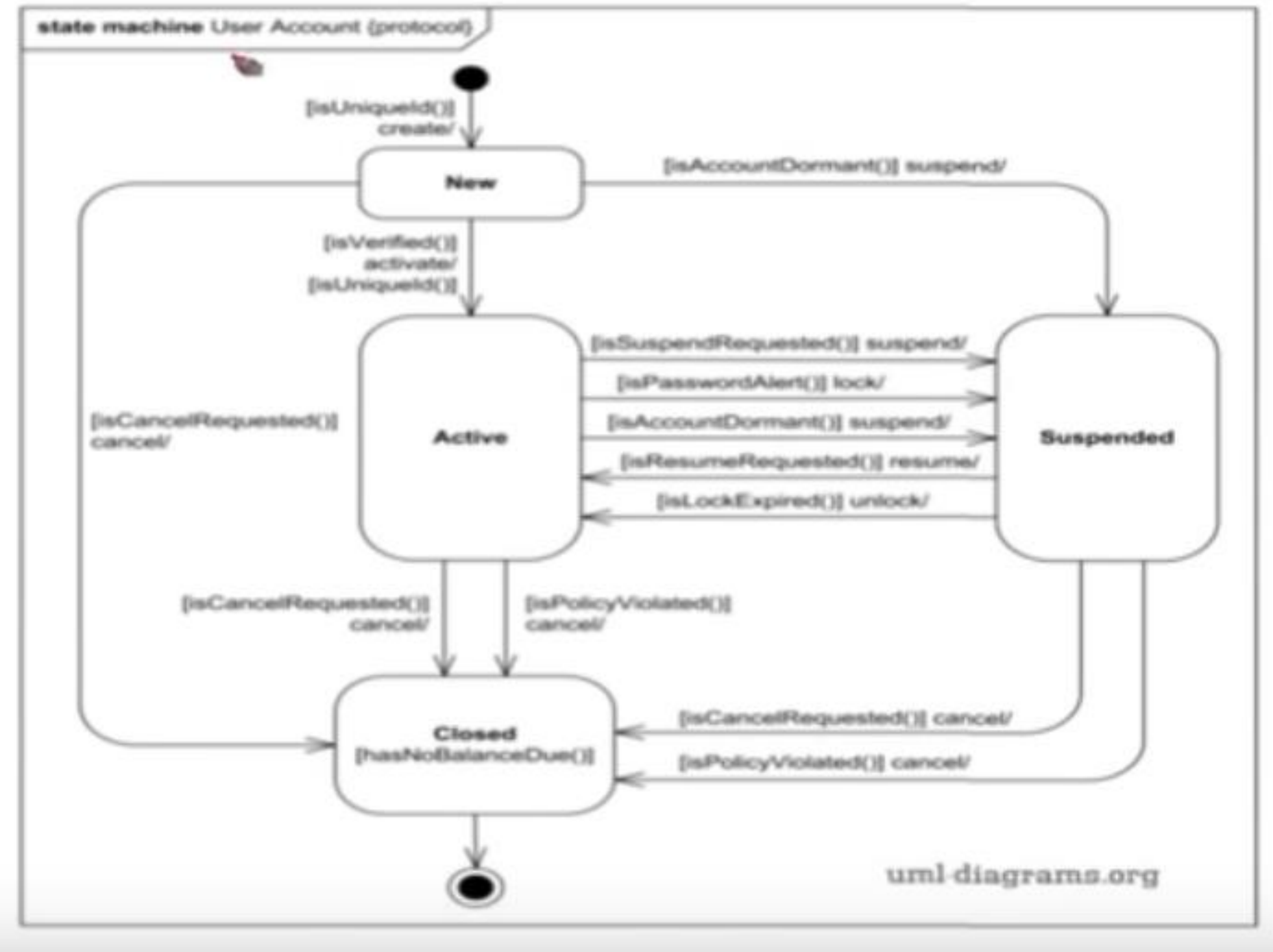
- The textual notation for a protocol transition:
  - protocol-transition ::= [pre-condition] trigger '/' [post-condition]
  - pre-condition ::= '[' constraint ']'
  - post-condition ::= '[' constraint ']'

Protocol transition from New to the Active state

with pre-condition (guard), trigger, and a post-condition.

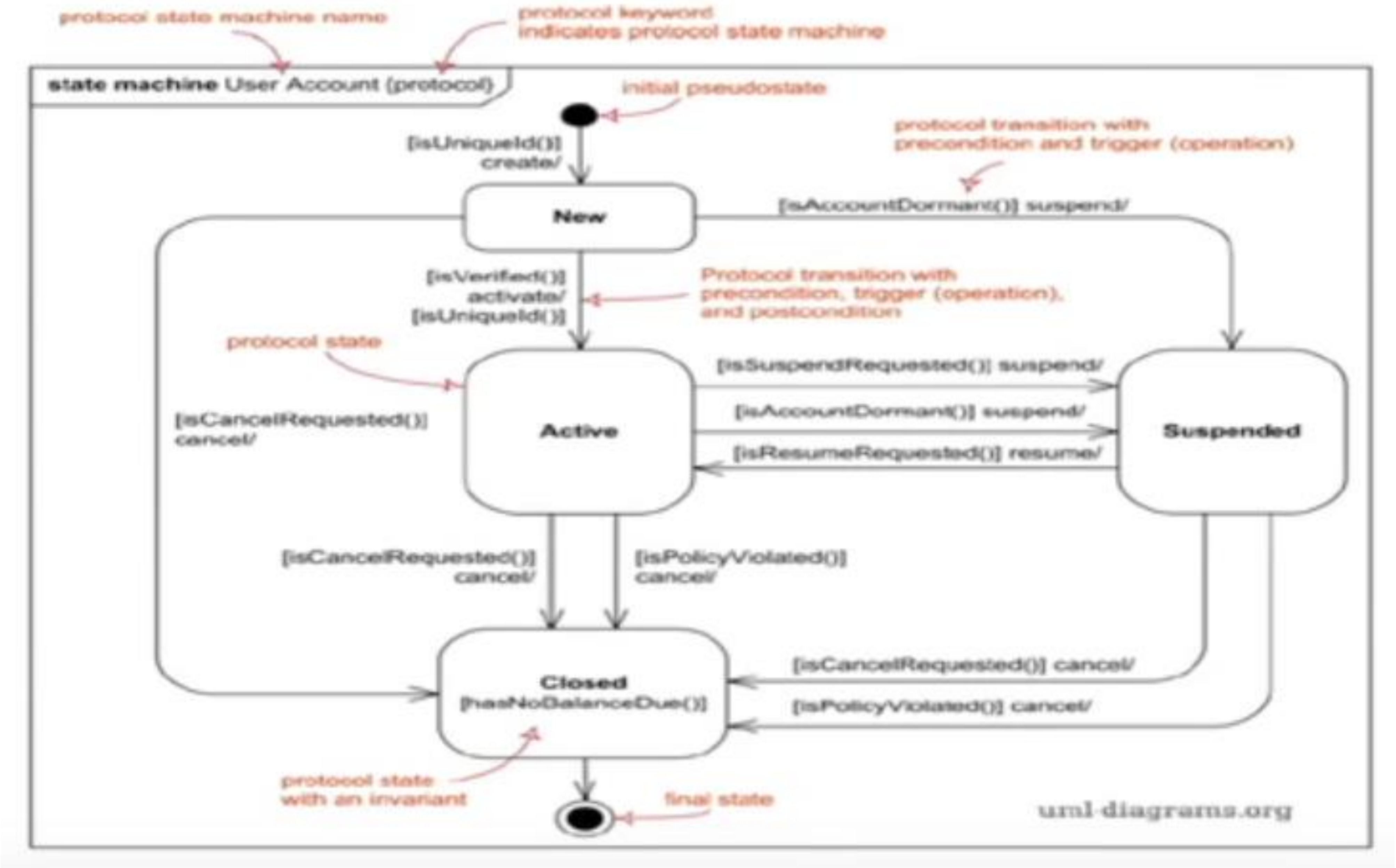
State Machine Diagram

Protocol State Machine - Protocol Transition – Online Shopping User Account


state machine User Account {protocol}
[isUniqueId()]
create/
New
[isAccountDormant()] suspend/
[isVerified()]
activate/
[isUniqueId()]
Active
Suspended
[isSuspendRequested()] suspend/
[isPasswordAlert()] lock/
[isAccountDormant()] suspend/
[isResumeRequested()] resume/
[isLockExpired()] unlock/
[isCancelRequested()]
cancel/
[isCancelRequested()]
cancel/
[isPolicyViolated()]
cancel/
Closed
[hasNoBalanceDue()]
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
uml-diagrams.org

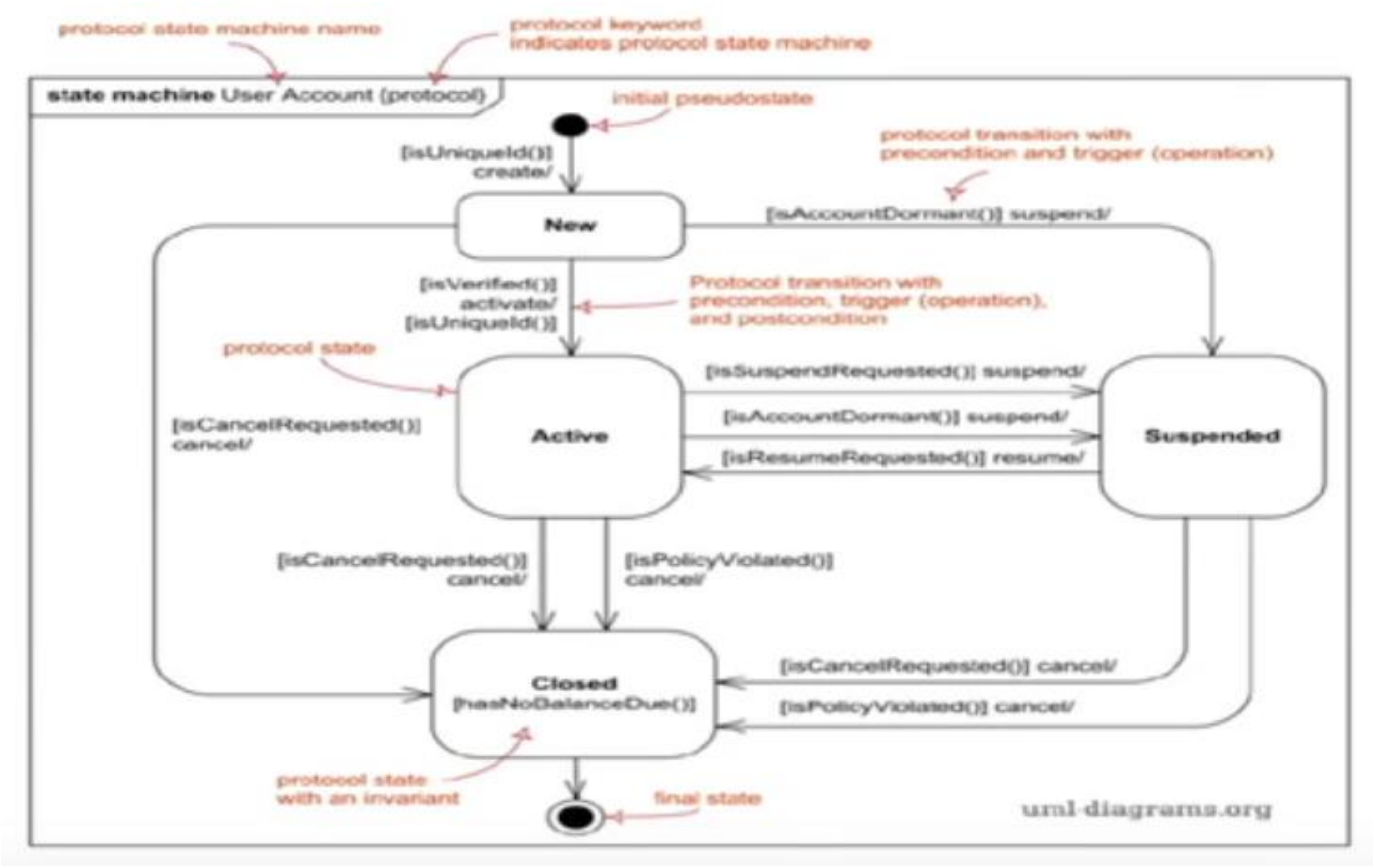
State Machine Diagram

Protocol State Machine - Protocol Transition – Online Shopping User Account

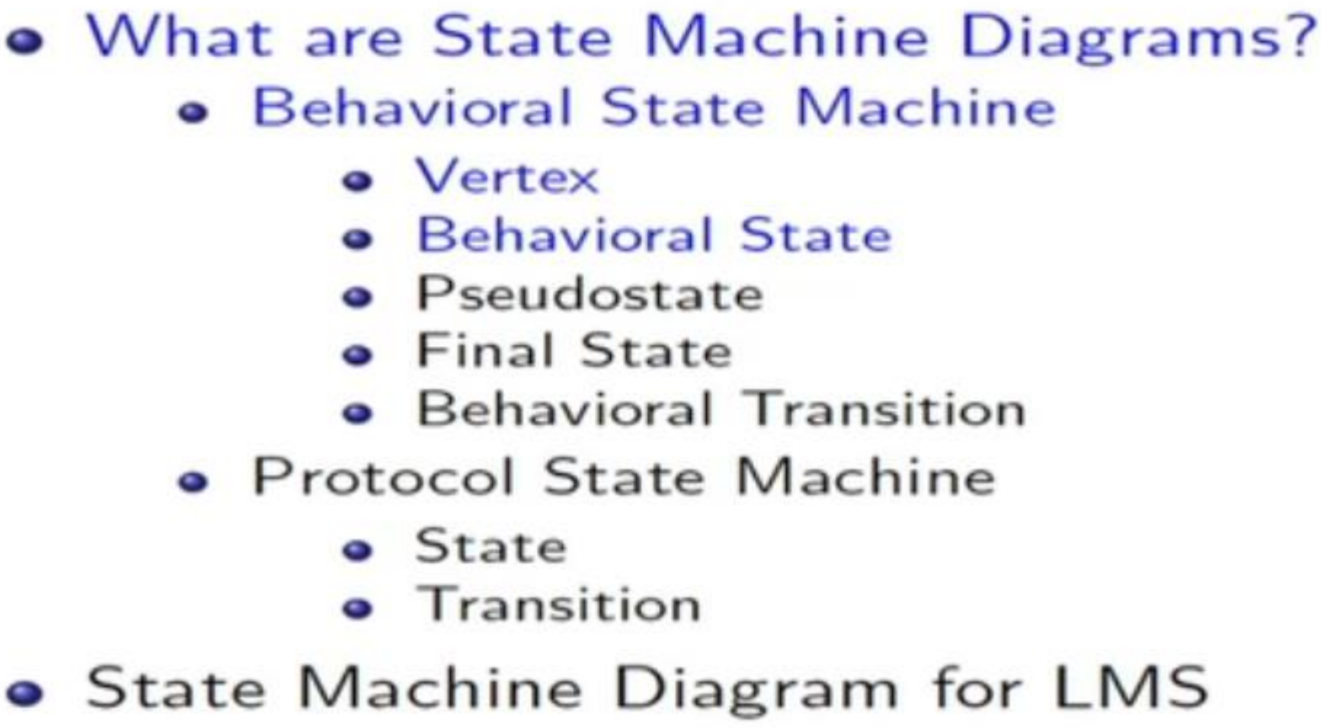


State Machine Diagram

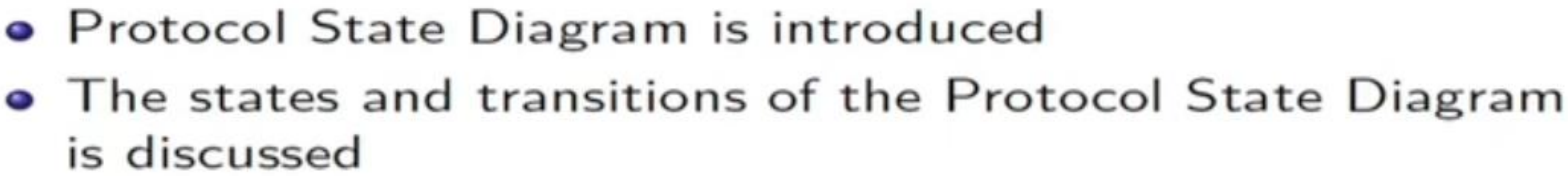
Protocol State Machine - Protocol Transition – Online Shopping User Account



State Machine Diagram Summary



- What are State Machine Diagrams?
  - Behavioral State Machine
    - Vertex
    - Behavioral State
    - Pseudostate
    - Final State
    - Behavioral Transition
  - Protocol State Machine
    - State
    - Transition
- State Machine Diagram for LMS



- Protocol State Diagram is introduced
- The states and transitions of the Protocol State Diagram is discussed

Reference

Source: NPTEL - Object-Oriented Analysis and Design, IIT Kharagpur Prof. Partha Pratim Das Prof. Samiran Chattopadhyay Prof. Kausik Datta

https://nptel.ac.in/courses/106105153