Université Nice Sophia Antipolis

Conception Orientée Objets

Machines états/transitions

Frédéric Mallet

http://deptinfo.unice.fr/~fmallet/

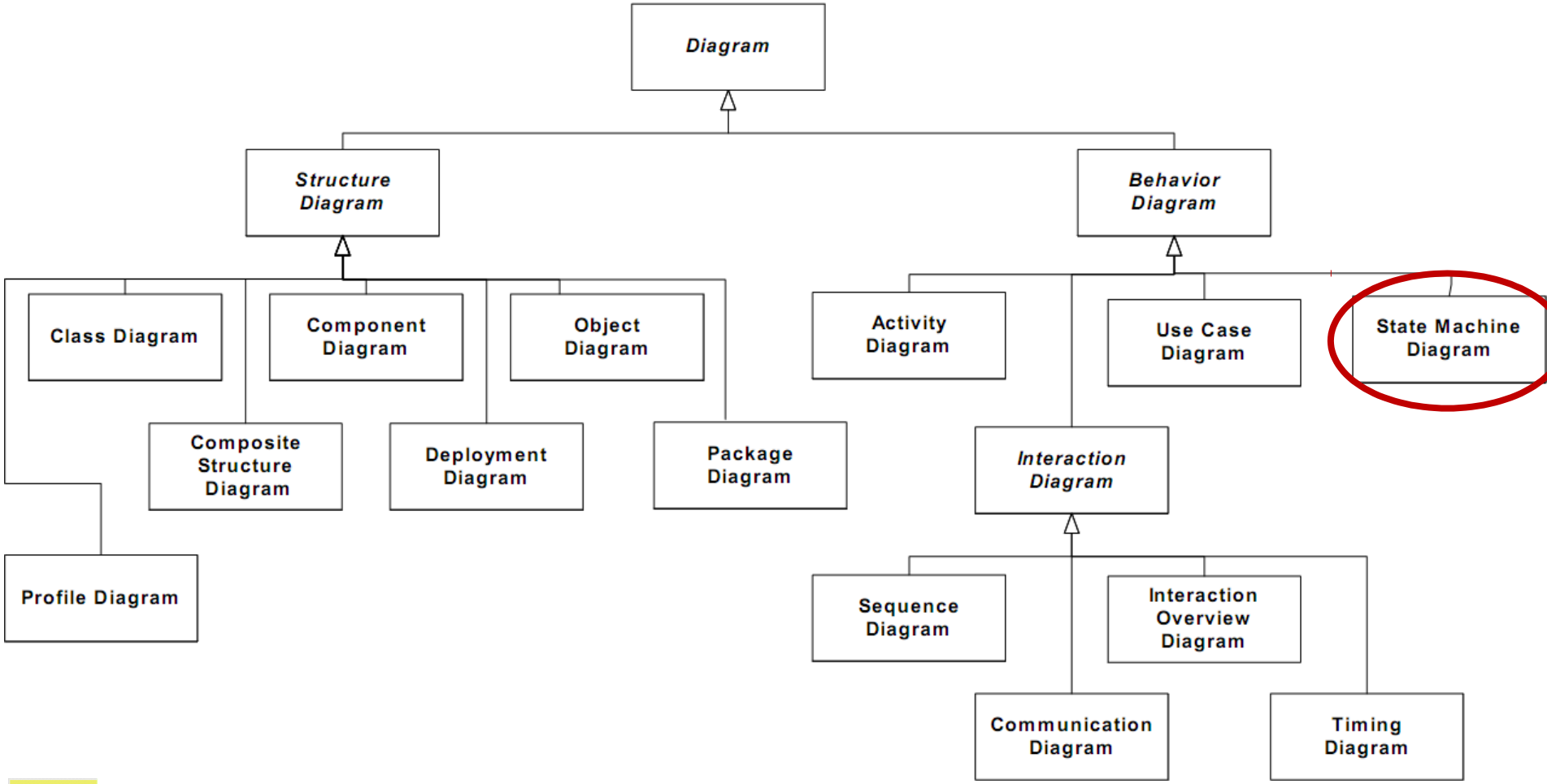
Université Nice Sophia Antipolis

UML2 – *State Machines*

- Objectifs
  - Behavioral State machines
    - Décrire le comportement attendu d'un *Use Case*
    - Décrire le comportement d'une méthode
    - Décrire le comportement **interne** d'une classe
  - Protocol State Machines
    - Décrire le comportement **externe** d'une classe
    - Décrire le comportement d'un ensemble de classes

Université Nice Sophia Antipolis

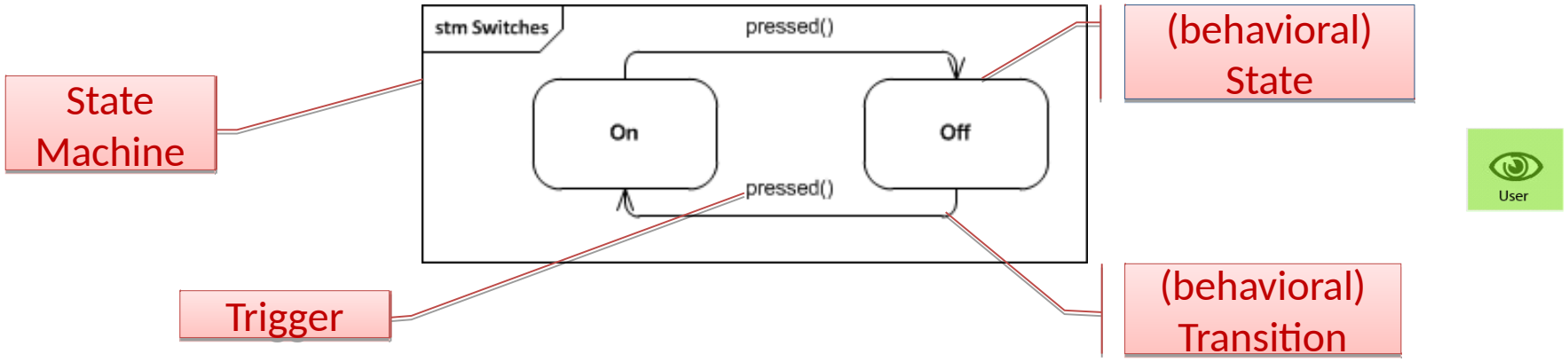
14 diagrams



UML

UML Behavioral State Machines

- Object-based variant of Harel's statecharts
- Specialization of Behavior
  - Behavior of individual entities (e.g., class instances, operations, actions, use cases)
    - Discrete behavior through finite state transitions
  - Traversal of graph of states connected with transitions
  - Transitions are triggered by event (occurrence)s
  - During the traversal, the SM executes a series of activities



Université Nice Sophia Antipolis

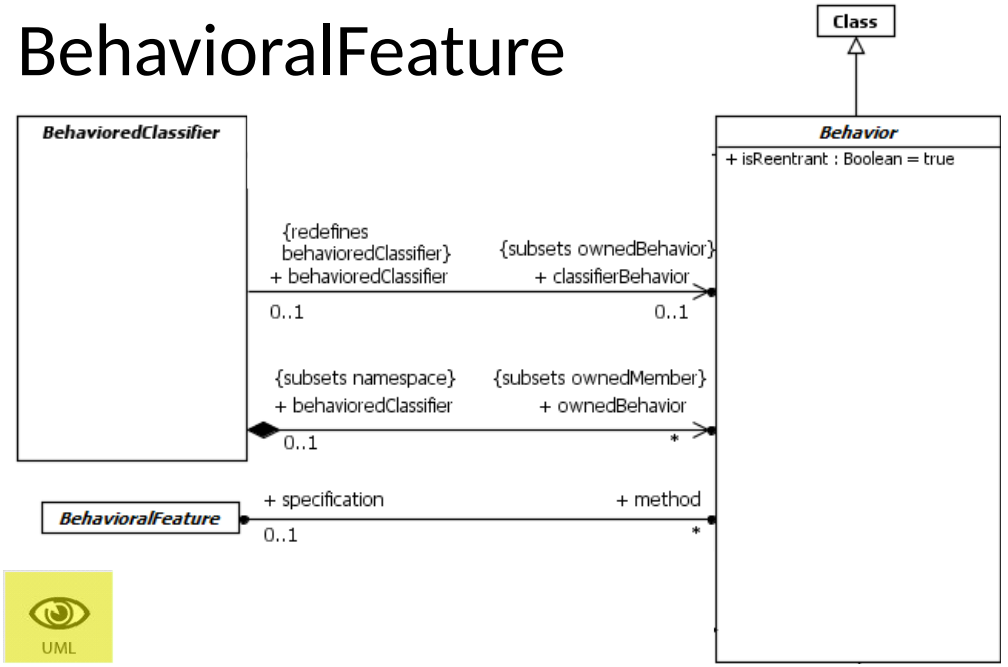
UML Behavioral State Machines

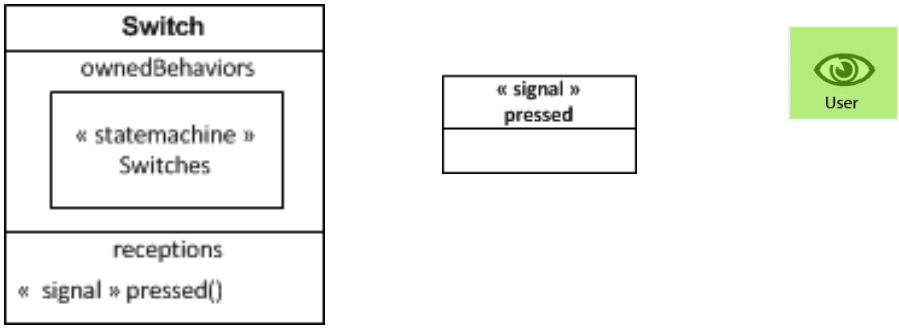
- Specialization of Behavior
- Context
  - Usually a *behaviored* classifier
  - Defines the signal and call **triggers** available
  - Defines the attribute and operations available in activities [trigger]
- Behavioral features and methods [effect]
  - A SM can be the method associated with a behavioral feature (operation, reception)
    - Parameters of behavioral feature => parameter of the state machine

Université Nice Sophia Antipolis

Context

- BehavioredClassifier or BehavioralFeature
  - BehavioredClassifier
    - Class/Component
    - UseCase/Actor
    - Node
  - Behavior
    - StateMachine
    - Activity
    - Interaction
  - BehavioralFeature
    - Operation
    - Reception

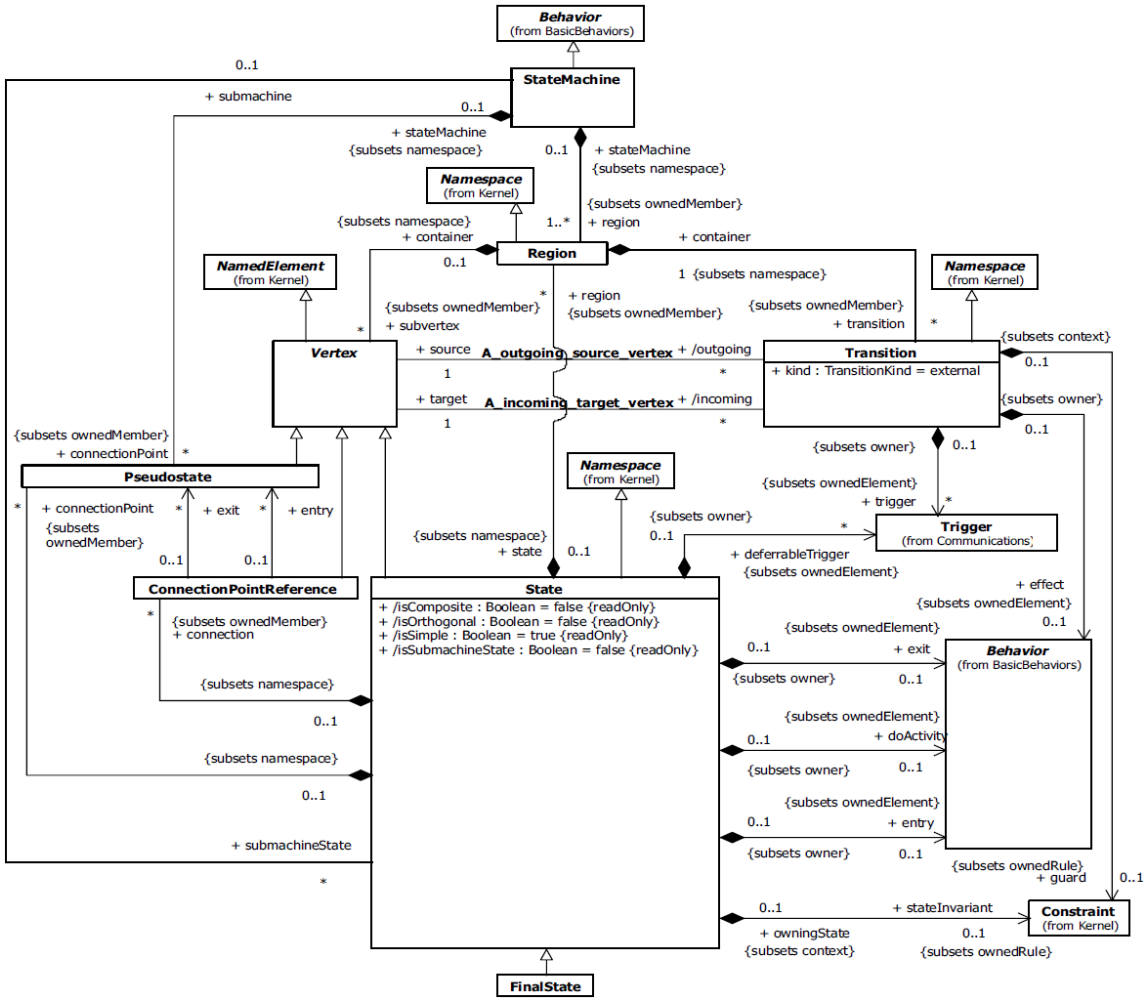




Université Nice Sophia Antipolis

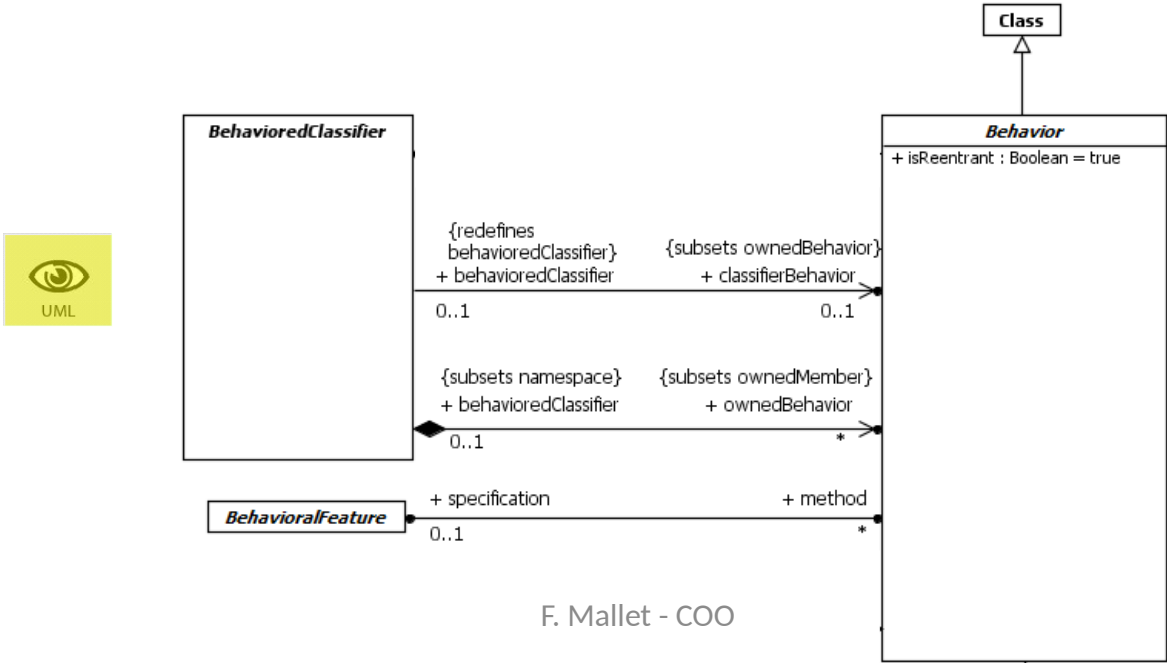
UML Behavioral State Machines

- Specialization of Behavior
  - Discrete behavior through finite state transitions



UML Behavioral State Machines

- States and Transitions of ONE classifier
  - Recommend: Make at least one state machine for each classifier that has an important dynamic behavior
- Usually one classifier owns several behaviors
  - All the state machines are executed concurrently
  - One behavior describes THE behavior



Université Nice Sophia Antipolis

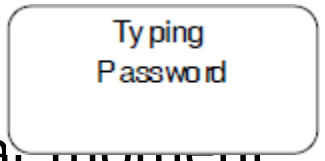
Regions and States

Regions

- A StateMachine contains one or more regions
- Regions contain vertices (states) and transitions
- Only one *active state* in each region

The word *state* may refer to

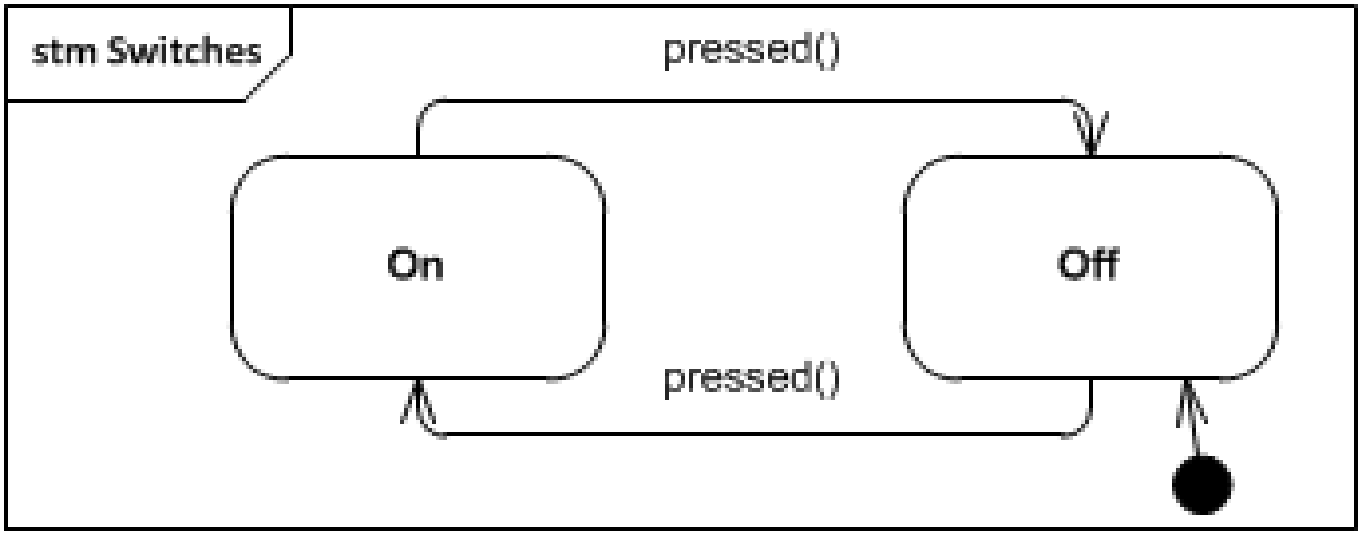
- One kind of vertex in a state machine
- A snapshot of the system at one particular moment
  - Set of all the active states
    - Card: the card is inserted
    - ATM: the user is typing the password
  - Set of all the values of each variable
    - Card: the user has three chances and it has already given one wrong password



Université Nice Sophia Antipolis

Initial (*pseudo*-)state

- Pseudo states are transient vertices
  - Markers: not allowed to remain in a pseudo-state
    - Initial [0..1]: default state of a composite, no trigger, no guard



Université Nice Sophia Antipolis

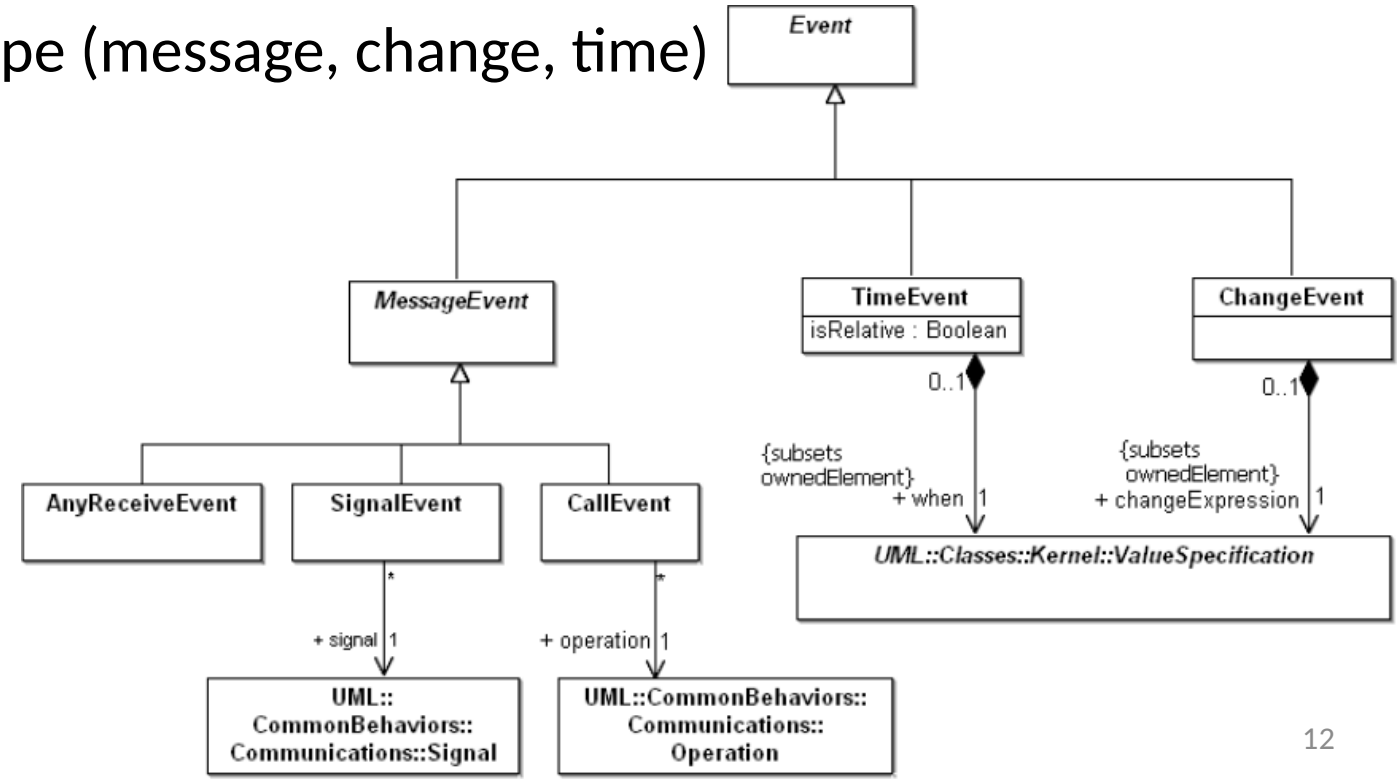
Final state

- Not a pseudo-state !
  - At most one per region
  - When a final state is reached, the enclosing region is completed
  - A state machine is completed when all its regions are completed
- Completion transitions (whose target is a final state)
  - Unlabeled transitions
- Notation:

Université Nice Sophia Antipolis

Events

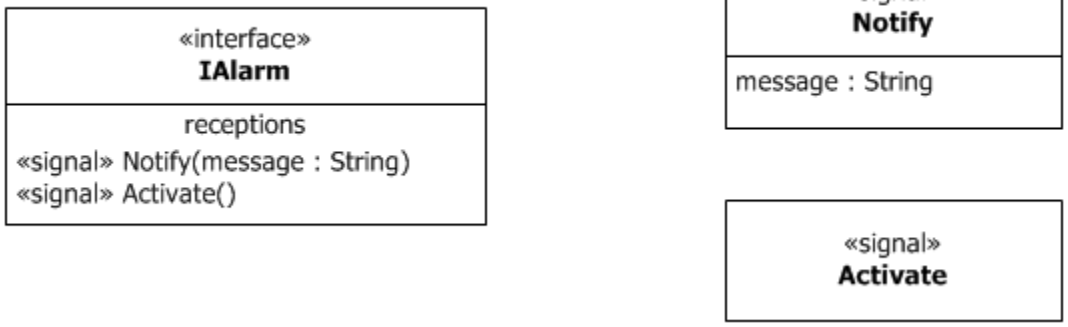
- An event
  - represents *things* that occur during the execution
    - E.g.: press the button, close the door
  - can have several occurrences
  - has NO duration
  - has a type (message, change, time)



Université Nice Sophia Antipolis

Signal, Reception, SignalEvent

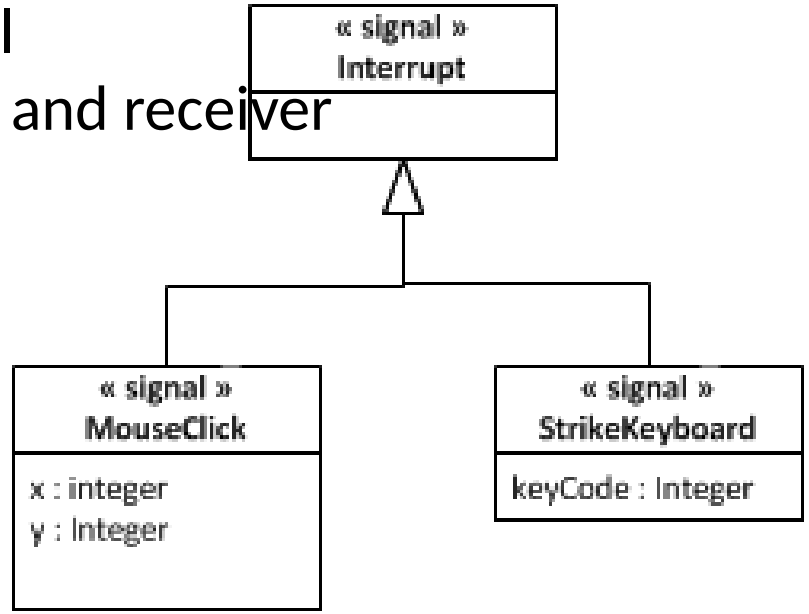
- Signal
  - Represents an asynchronous communication => no reply
  - Owns some attributes (whose values are carried by the message)
- Reception
  - Declares that an event may be received by a classifier
  - Parameters must match the attributes
- SignalEvent
  - Receipt of a signal
  - *<signal-event> ::= <name> [‘(‘ [<attr-name> [‘,’<attr-name>]*] ‘)’]*



Université Nice Sophia Antipolis

Signals

- A specific classifier
  - Asynchronous one-way communication
    - Explicit message
  - Owns some attributes
    - Information to be carried by the message
  - There is no operation on a signal
  - Can be at the same time sender and receiver



Université Nice Sophia Antipolis

Other message events

- CallEvent
  - Receipt, by an object, of a message invoking a call of an Operation

<call-event> ::= <name> [‘(‘ [<assigned-name> [‘,’<assigned-name>]] ‘)’]*

- AnyReceiveEvent
  - Triggered by any message not explicitely handled by a trigger

<any-receive-event> ::= ‘all’

Université Nice Sophia Antipolis

Change and time events

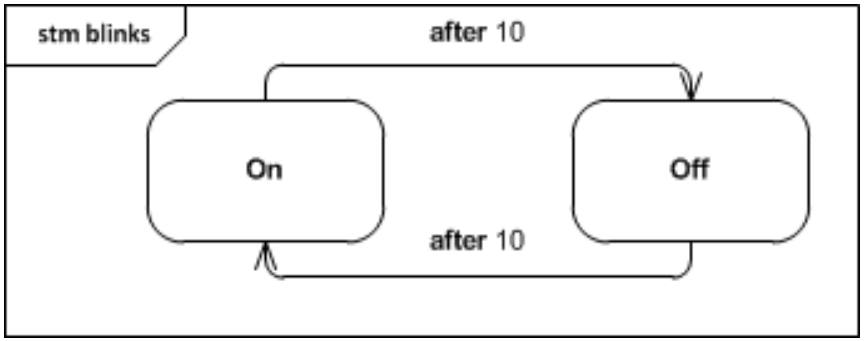
- Change events
  - A change in the system configuration that makes a condition true

<*change-event*> ::= 'when' <*value-specification*>

- Time events
  - Event that occurs at a specific point in time
  - Either relative or absolute

<*relative-time-event*> ::= 'after' <*time-expression*>

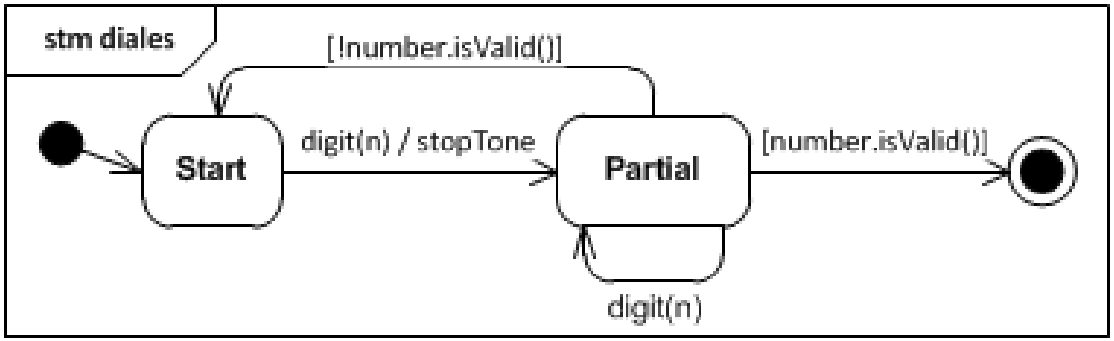
<*absolute-time-event*> ::= 'at' <*time-expression*>



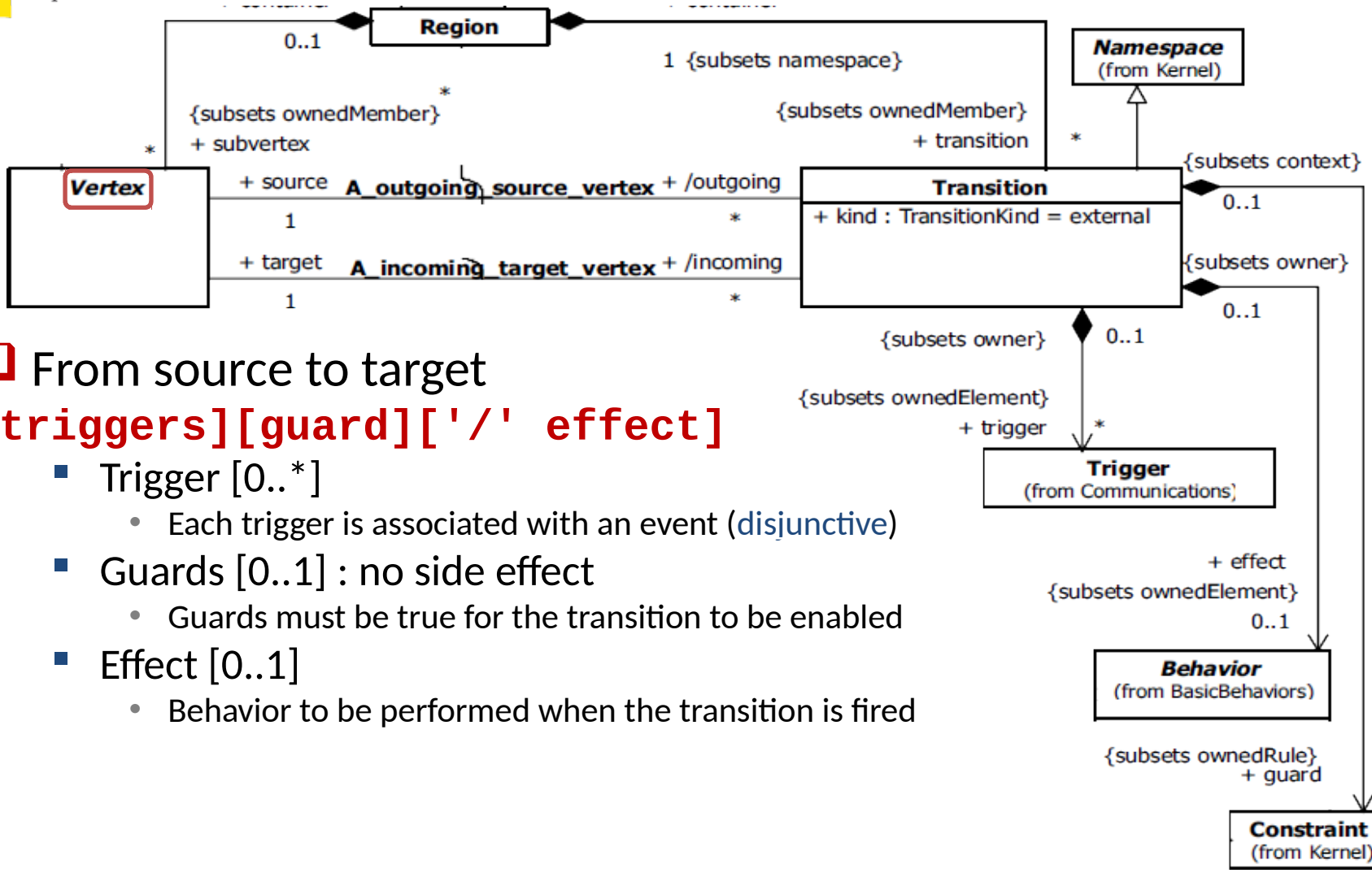
Université Nice Sophia Antipolis

Transitions

- A transition
  - captures the expected events at one given state
  - One single source and one single target (may be the same)
  - The duration is undefined
- Three statuses
  - Reached: source state is active
  - Traversed: being executed (including the effects)
  - Completed: after it has reached its target vertex



Behavioral transitions



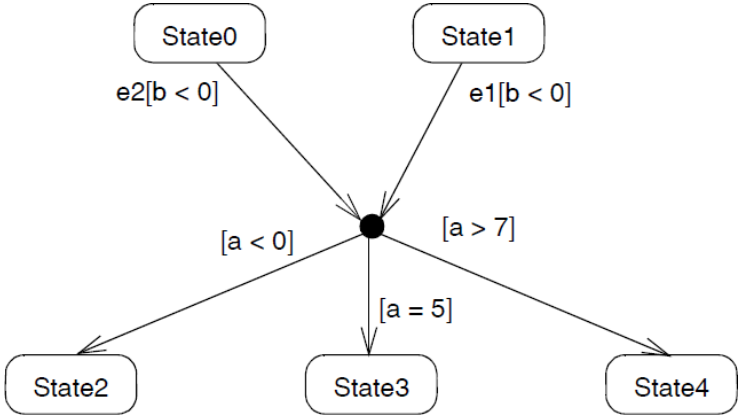
- From source to target

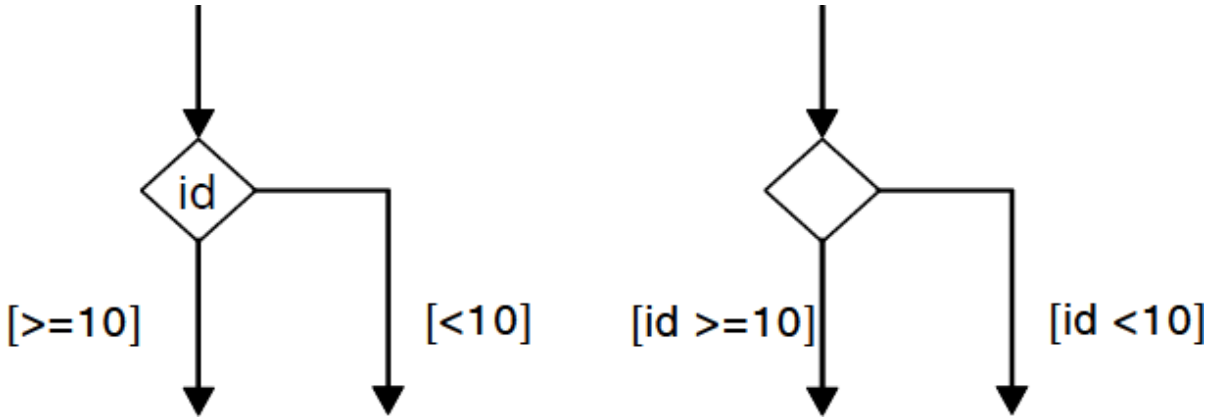
`[triggers][guard]['/' effect]`

  - Trigger [0..*]
    - Each trigger is associated with an event (disjunctive)
  - Guards [0..1] : no side effect
    - Guards must be true for the transition to be enabled
  - Effect [0..1]
    - Behavior to be performed when the transition is fired

Junction and choice Pseudo-states

- Pseudo states are transient vertices
  - Connect multiple transitions into more complex state transition paths
    - **junction**: semantic-free vertices
    - **choice** : dynamic conditional branc outgoing guards evaluated to true,

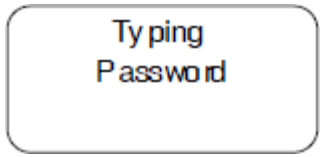


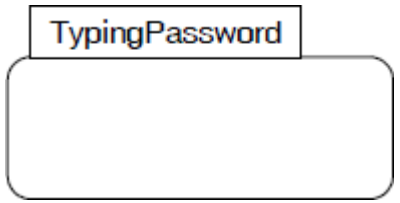


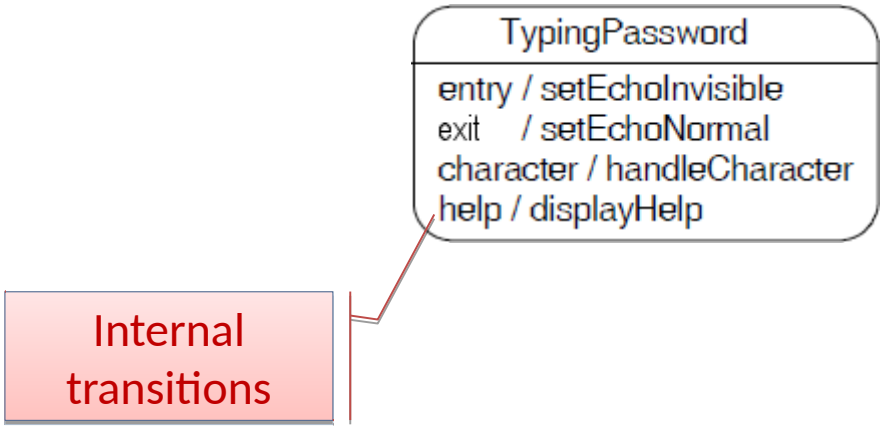
Université Nice Sophia Antipolis

Behavioral **Simple** States

- **Simple** states
  - Name (String)
  - Entry/do/Exit actions
- Composite states
- Submachine states

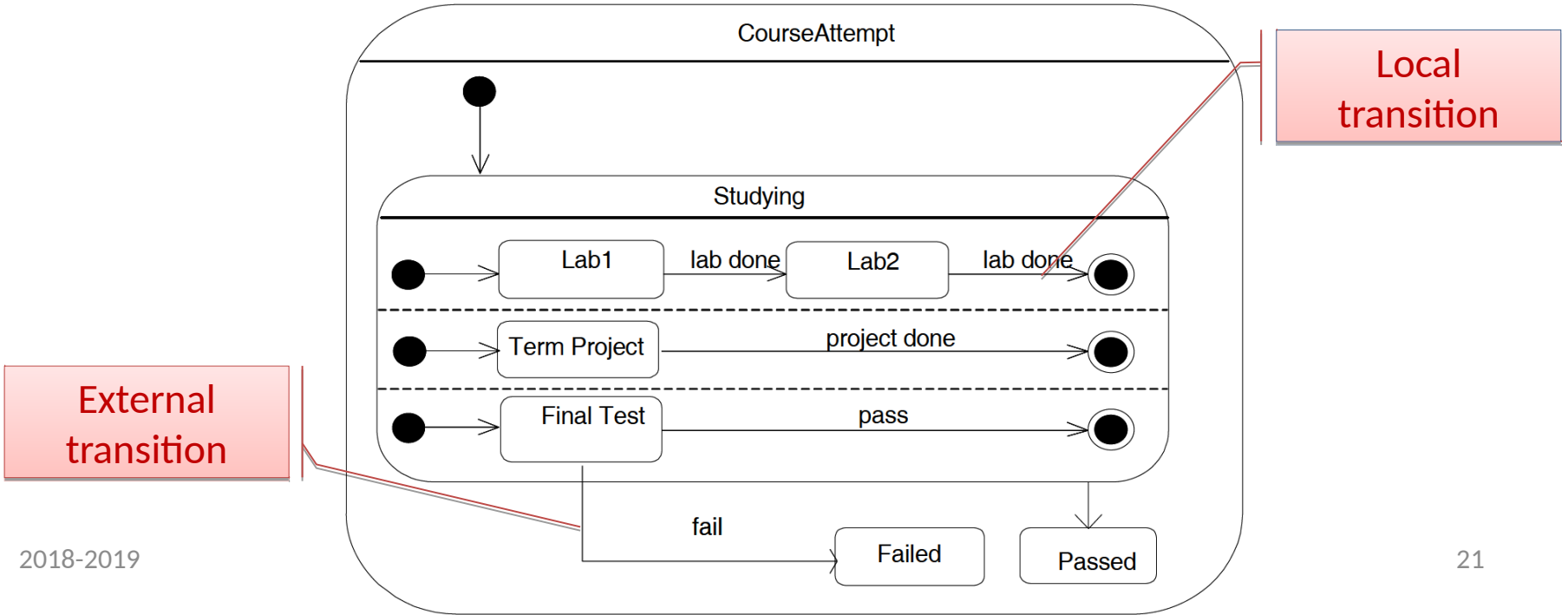






Behavioral **composite** states

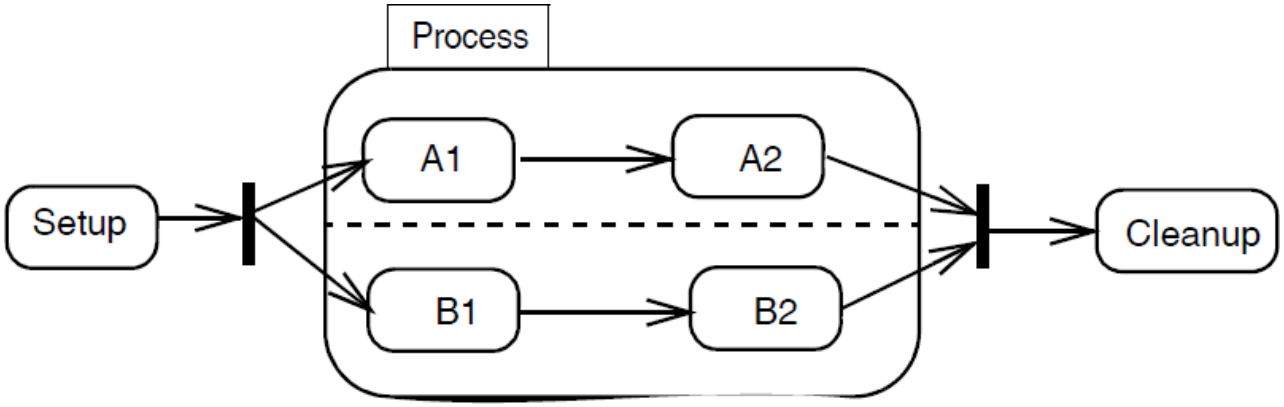
- Simple states
- Composite states
  - Either contains one region
  - Or decomposed into two or more orthogonal regions
- Submachine states



Université Nice Sophia Antipolis

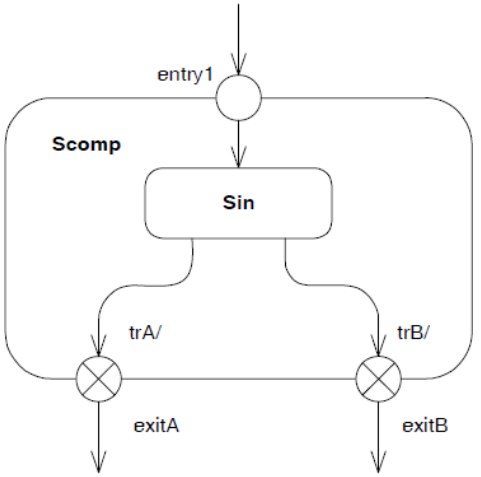
Join and Fork Pseudo-states

- Pseudo states are transient vertices
  - Connect multiple transitions into more complex state transition paths
    - join: join transitions from orthogonal regions, no trigger, no guard on the entering transitions
    - fork: split one transition into several ones, no trigger and no guard on the outgoing transitions



Entry/Exit Pseudo-states

- Pseudo states are transient vertices
  - Connect multiple transitions into more complex state transition paths
  - Entry point:
    - at most a single transition from the entry point to a vertex within the same region (allows for submachines)
    - Entry behavior is executed before the effect of the entry transition
    - If multiple regions => act as a fork
  - Exit point:
    - entering an exit point within any region implies the exit of the composite state or submachine state
    - Act as a join if multiple regions
  - Terminate: the execution is terminated without performing exit actions or exiting any state (DestroyObjectAction)



Université Nice Sophia Antipolis

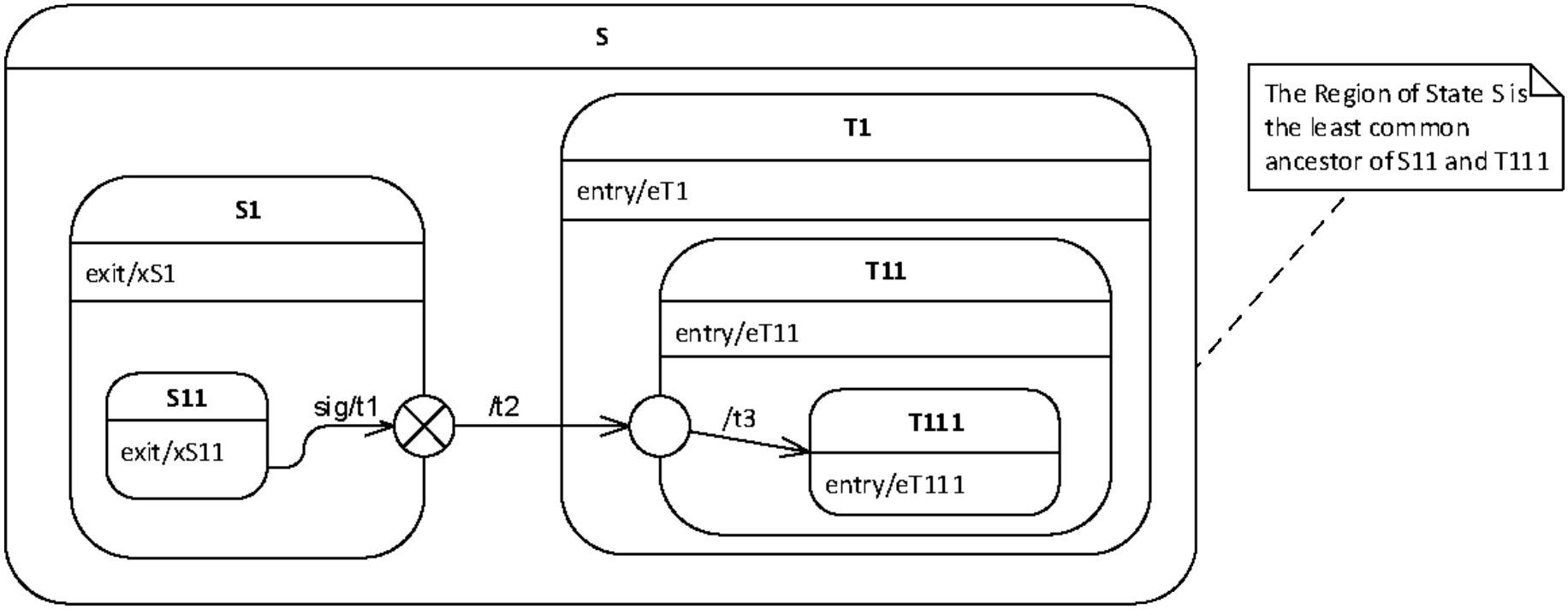
Transitions

- Three transition kinds
  - External [default]: exit the source, execute the exit behavior
  - Local: source is different from target, do not execute the exit behavior, only in composite states
  - Internal: self local transition (not exited, nor reentered), only for states
- Semantics
  - At each step, at most one enabled transition is fired
  - Conflicting transitions
    - An implicit priority is given depending on the state hierarchy
      - The lower in the hierarchy, the higher the priority
  - Transition selection : maximal set of transitions such that
    - All transitions are enabled
    - There is no conflicting transition within the set
    - There is no transition outside the set that has higher priority

Université Nice Sophia Antipolis

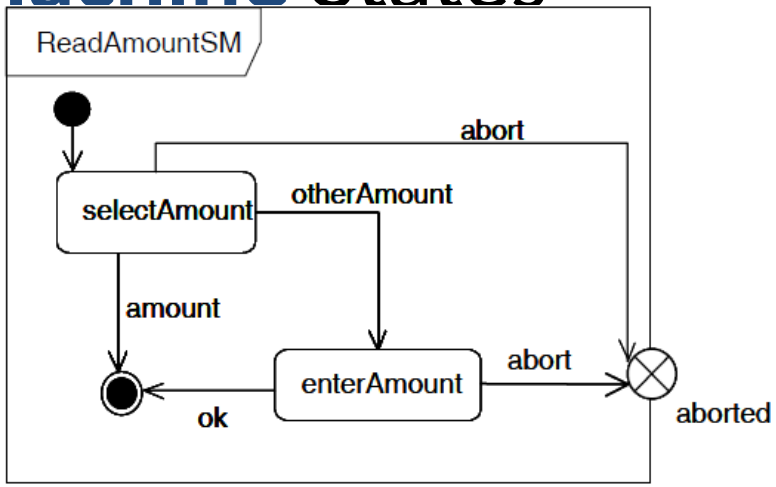
Compound Transitions

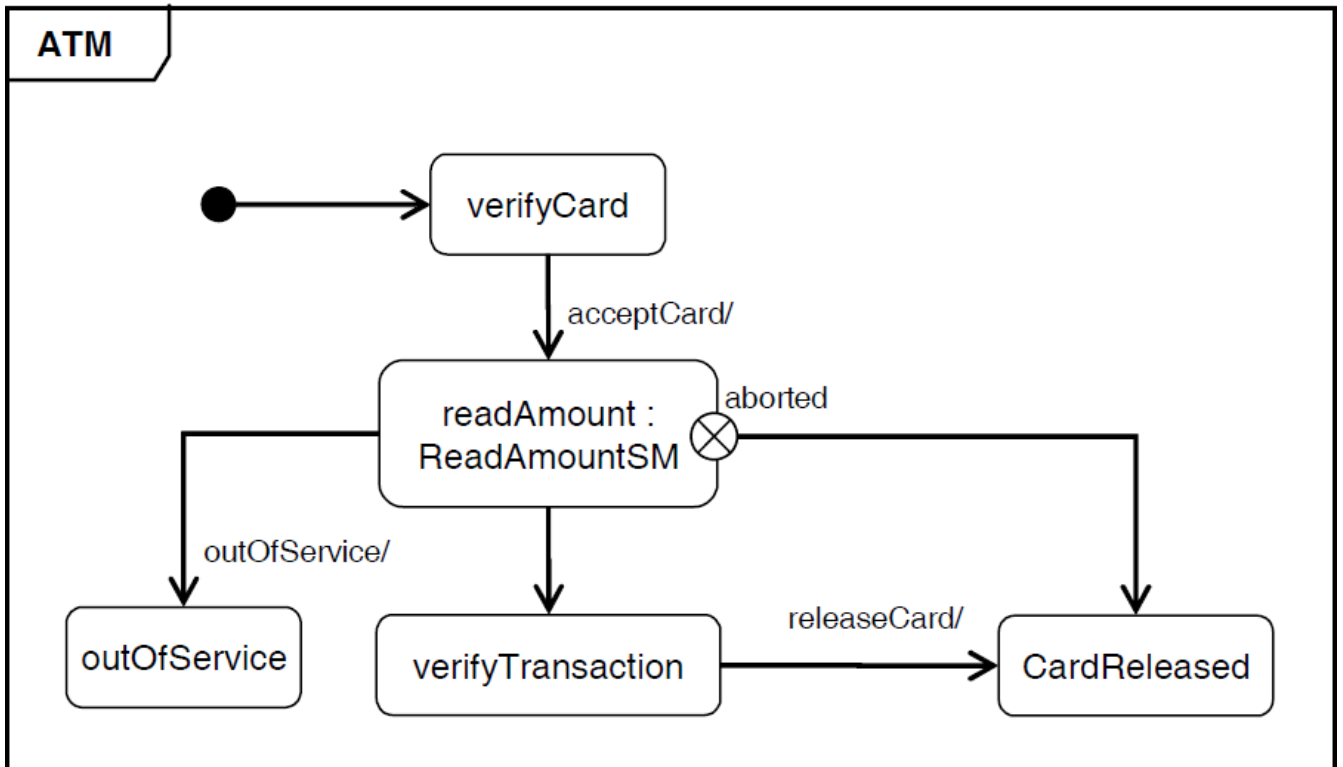
- Sequences of transitions with composite states



Behavioral **Submachine** States

- Simple states
- Composite states
- **Submachine** states

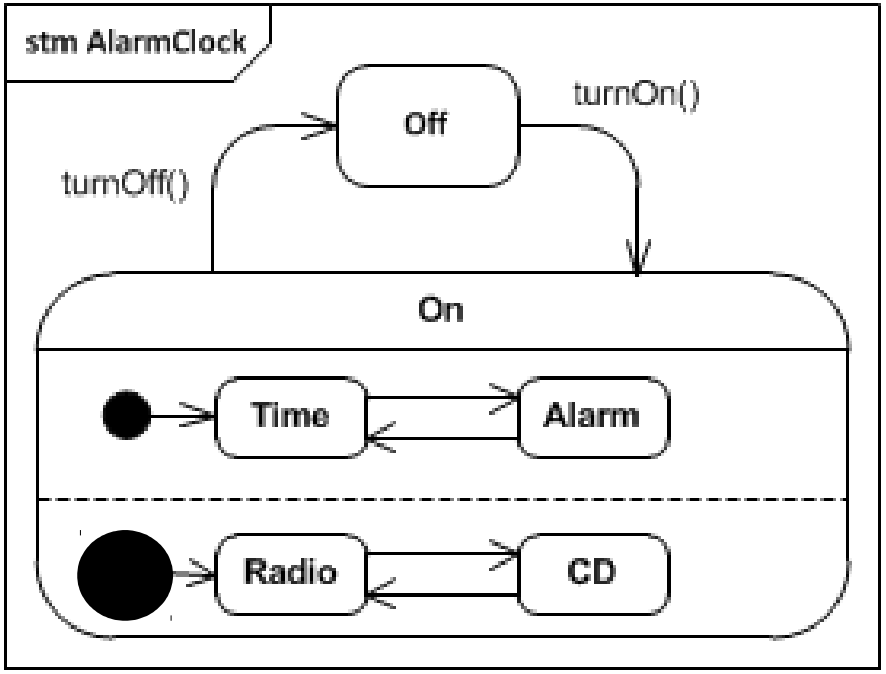




Université Nice Sophia Antipolis

History Pseudo-states

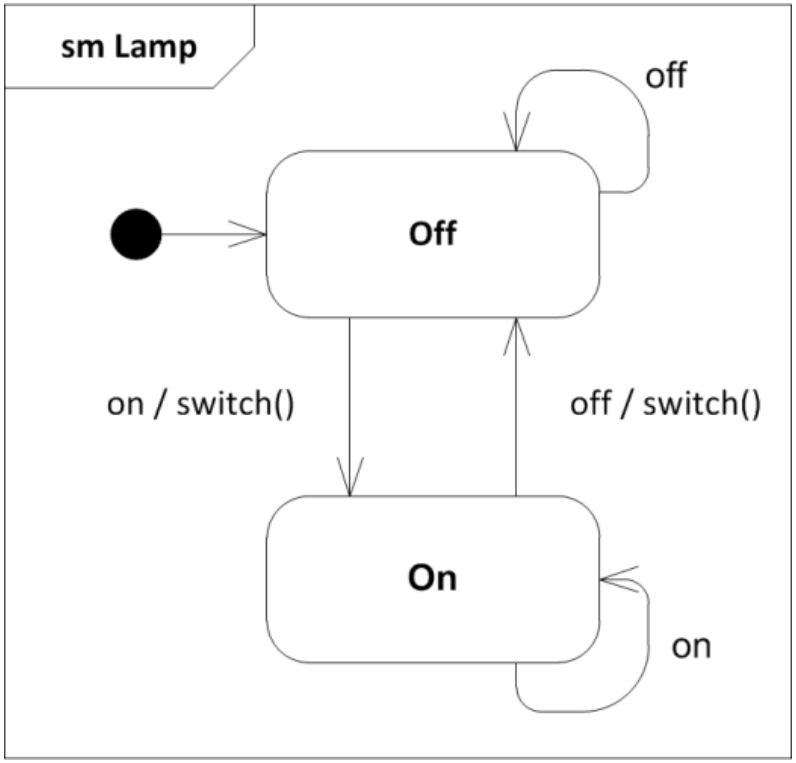
- Pseudo states are transient vertices
  - Connect multiple transitions into more complex state transition paths
    - deepHistory [0..1] (H*)
    - shallowHistory [0..1] (H)



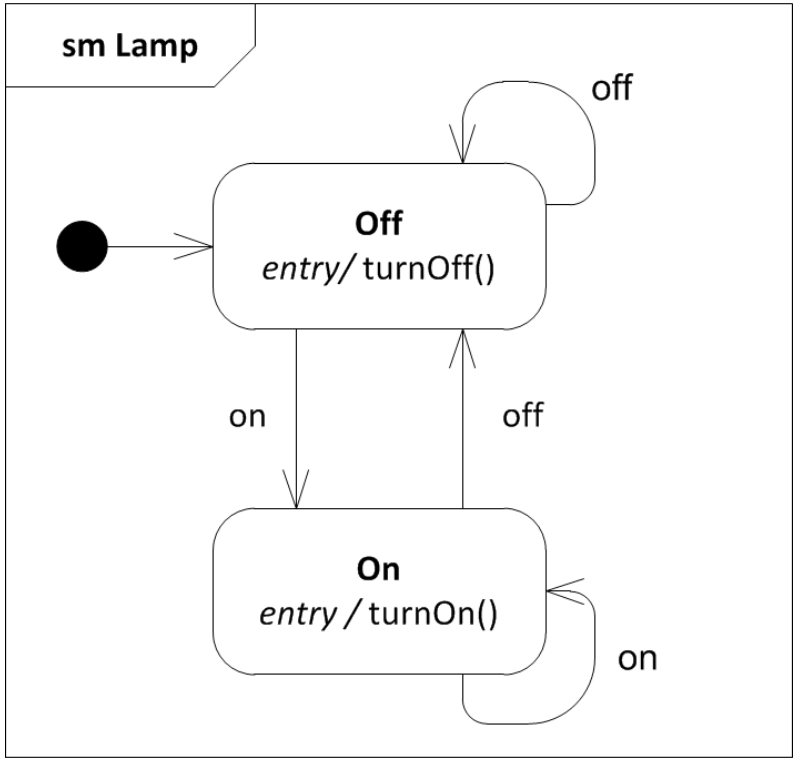
Université Nice Sophia Antipolis

Mealy vs. Moore Machines

Mealy machines



Moore Machines



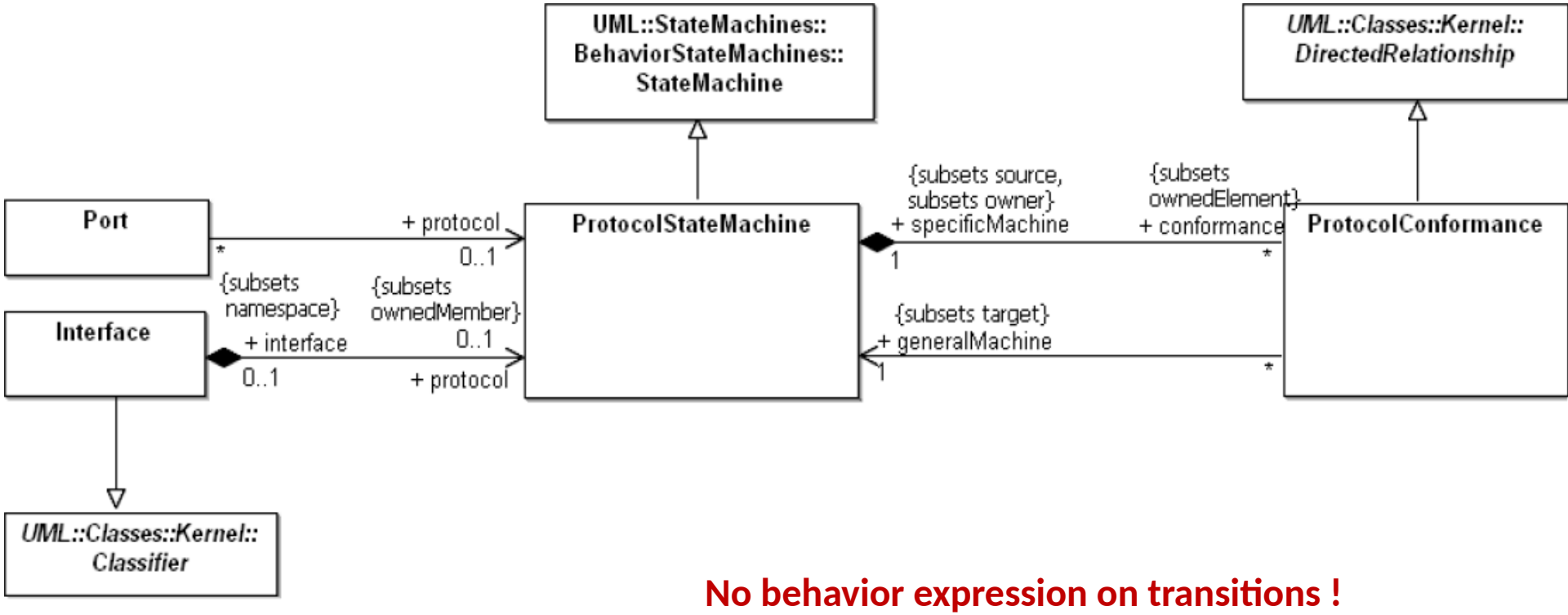
Université Nice Sophia Antipolis

UML State Machines

- **Behavioral** State Machines
  - Behavior of individual entities (e.g., class instances, operations, actions, use cases)
    - Associated with a classifier or a behavioral feature
  - Object-based variant of Harel statecharts
  - Behavioral states and behavioral transitions
- **Protocol** State Machines
  - Specialize Behavioral State Machines
  - Associated with a classifier (class, interface, port)
  - Usage Protocols
    - Legal transitions that a classifier can trigger, life cycle
    - Order of invocation of methods
  - Protocol states and protocol transitions

UML Protocol State Machines

- Specialization of StateMachine
  - Legal transitions than a classifier can trigger, life cycle

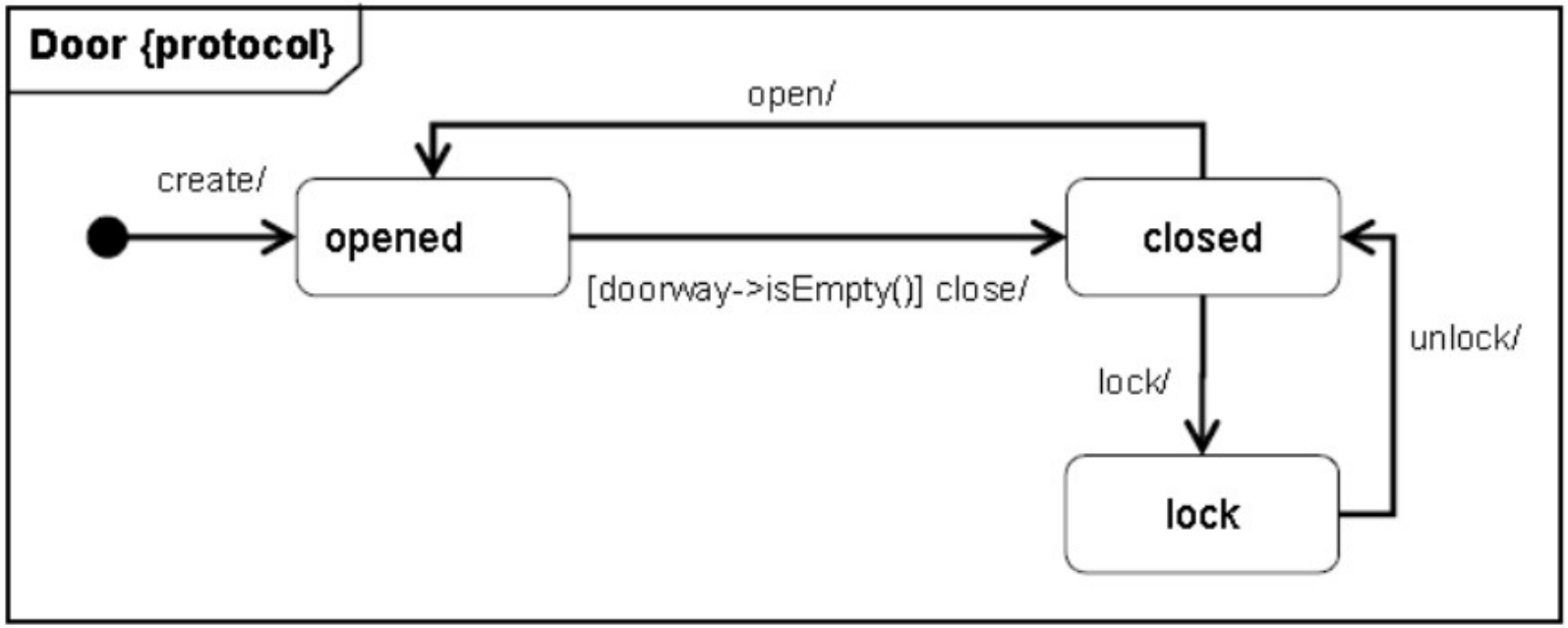


No behavior expression on transitions !
No entry/do/exit within states !
No deep/shallow history !

Université Nice Sophia Antipolis

Example protocol state machine

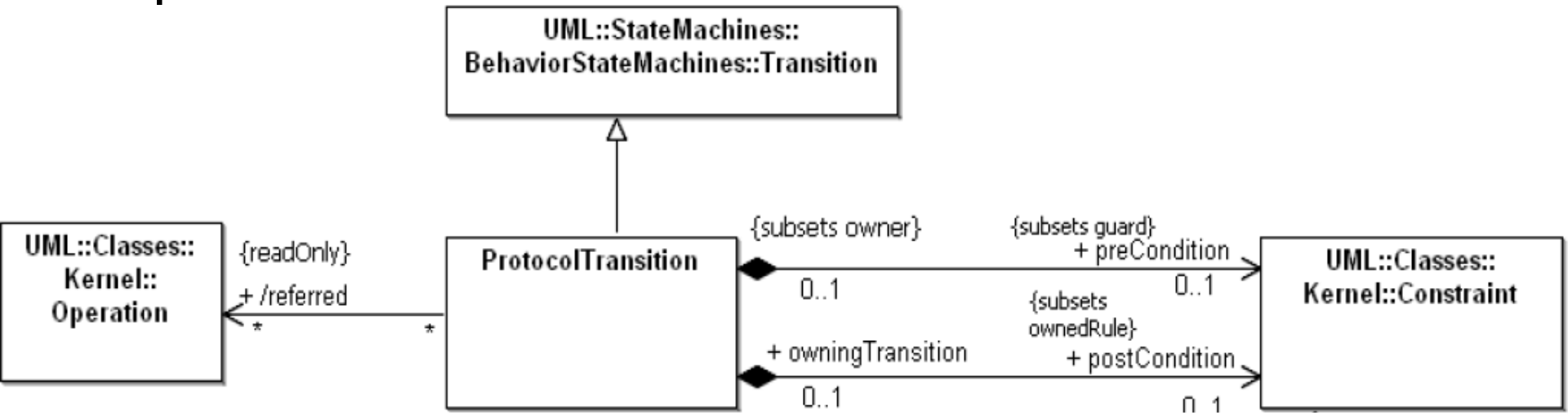
- Door



Université Nice Sophia Antipolis

Protocol transitions

- A protocol states only contains protocol states and protocol transitions



Université Nice Sophia Antipolis

Protocol Transition

- Transitions of protocol state machines

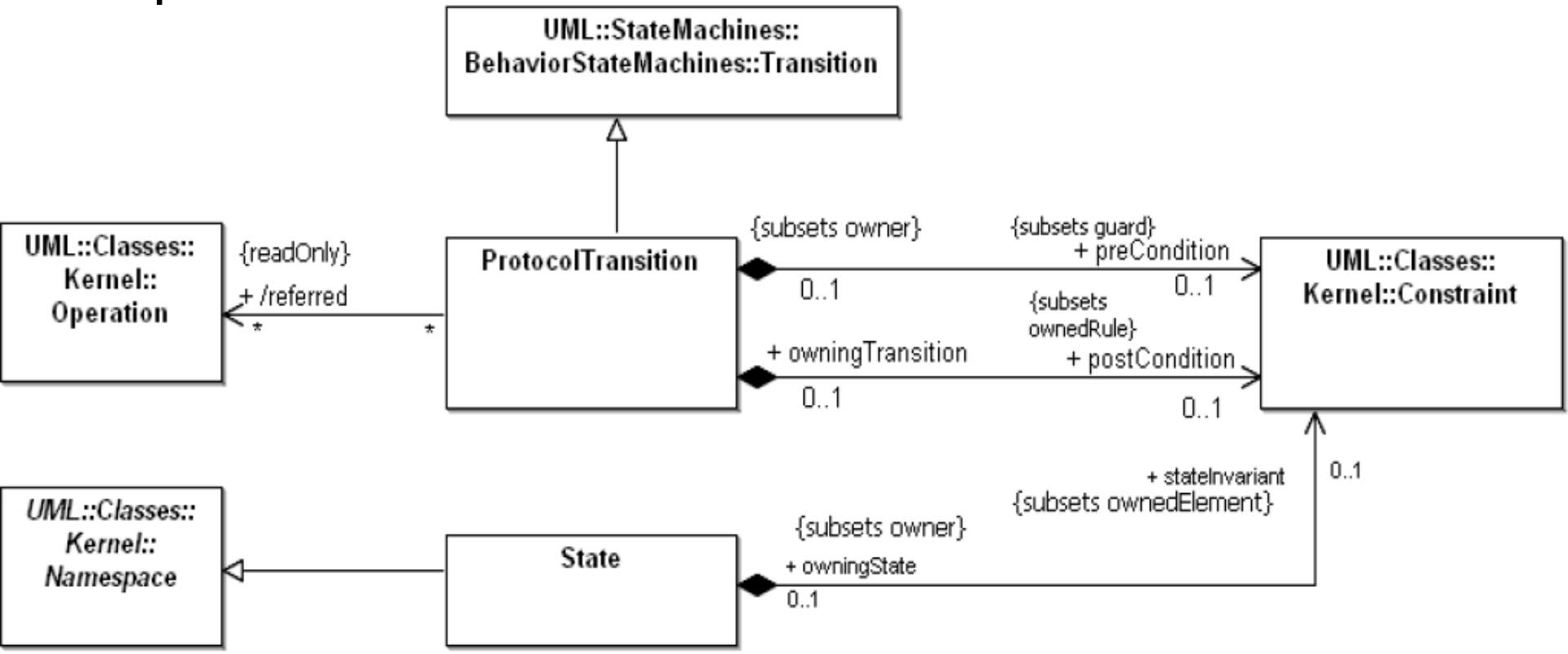
`[pre-condition] trigger / [post-condition]`

  - No effect action
    - When the trigger is a call action, the effect is the operation called
    - Otherwise, no effect
      - only specifies that a given event can be received under a specific state and pre-condition, and that a transition will lead to another state under a specific post-condition, whatever action is made
  - Unexpected event reception
    - Current state, state invariant, and pre-condition
      - **Pre-condition violation**: can be ignored, rejected or deferred
  - Unexpected behavior
    - Wrong final state, final state invariant or post-condition
      - **Error of the implementation**

Université Nice Sophia Antipolis

Protocol States

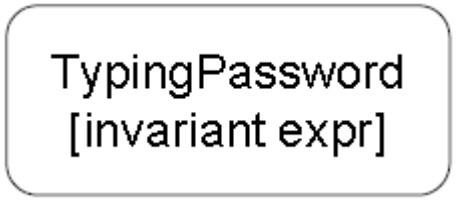
- A protocol states only contains protocol states and protocol transitions



Université Nice Sophia Antipolis

Protocol State

- Expose a stable condition of its context classifier
  - stateInvariant [0..1]
    - Specifies conditions that are always true when this state is the current state



State Machine redefinition

- State Machines can be extended