Métodos de Desenvolvimento de Software (MDS)
State Diagrams
2016/2017

State Diagrams: (a little bit of) history

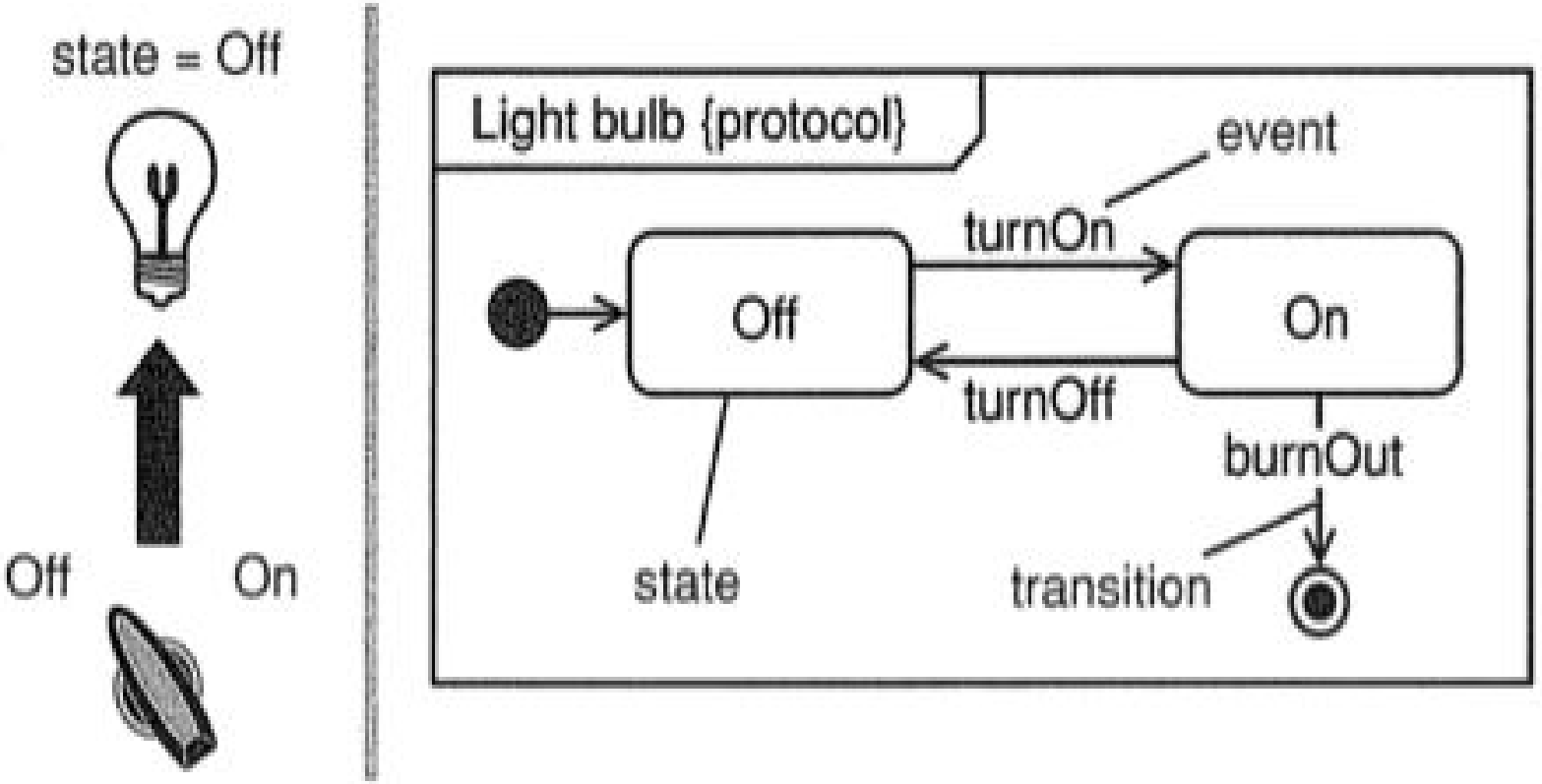
- Invented by David Harel (State Charts)
  - D. Harel. Statecharts: A Visual Formalism for Complex Systems. In Science of Computer Programming 8 (1987):231-274
- Became popular by James Rumbaugh (OMT)
- Most of the time professionals treat indifferently: *"statechart"*, *"state diagram"*

State Diagrams

- **Goal**: to model behavior (dynamic)
- Use for modeling objects (or system)with complex behavior that need more detail
  - When we still didn't understood the behavior (lifecycle)of the object (or system)
- Use if the control is deeply influenced by external events
- **Don't use**
  - When several objects are involved (in that case interaction diagrams are more adequate)

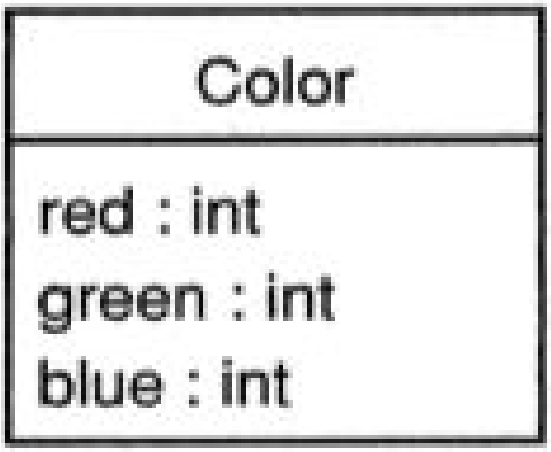
State Diagrams

- **States**: abstraction of the attributes and object relations
  - The gasoline deposit is too low when the level of gasoline is below x for more than n seconds
- **Events**: something happens at a certain time
  - The alarm switches off
- **Conditions**: “test” over attributes or state
  - The level of gasoline is low
  - The alarm is switched on
- **Transitions** between states: triggered by events
- **Actions and activities**: executed in transitions or states

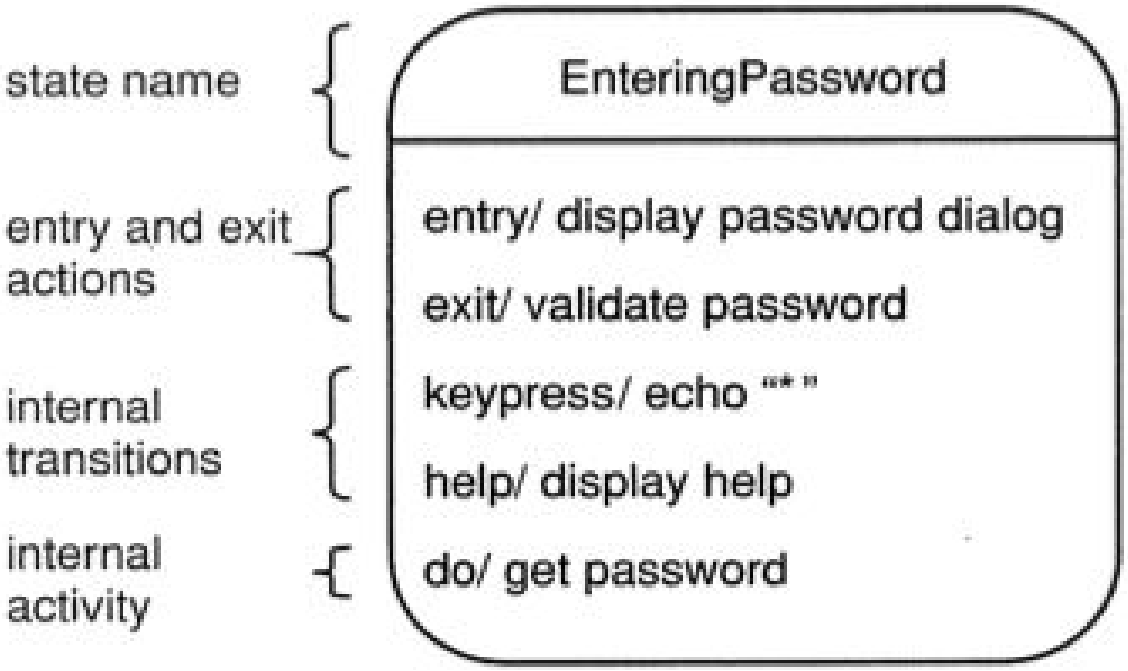

state = Off
Off
On
Light bulb {protocol}
event
turnOn
Off
On
turnOff
burnOut
state
transition

State

- A condition or situation during the lifetime of an object(or system) during which:
  - It satisfies some condition
  - Performs some activity
  - Or, waits for some event
- Consists of:
  - Name
  - Actions or activities
  - Internal transitions
  - Sub-states
  - Deferred events (to delay the answer to certain events)

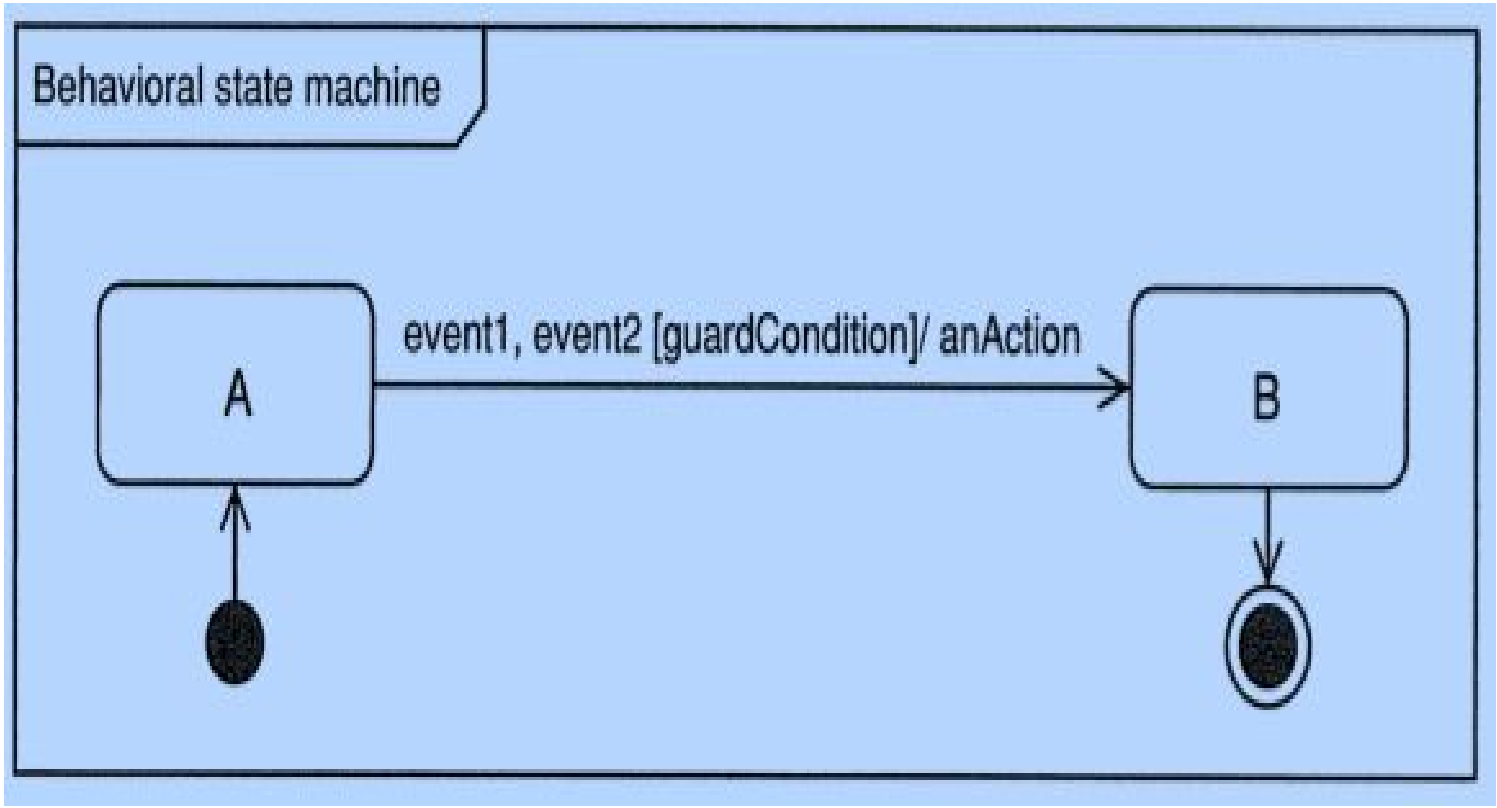


State



action syntax: eventName/ someAction

activity syntax: do/ someActivity


Behavioral state machine
A
event1, event2 [guardCondition]/ anAction
B

Transição

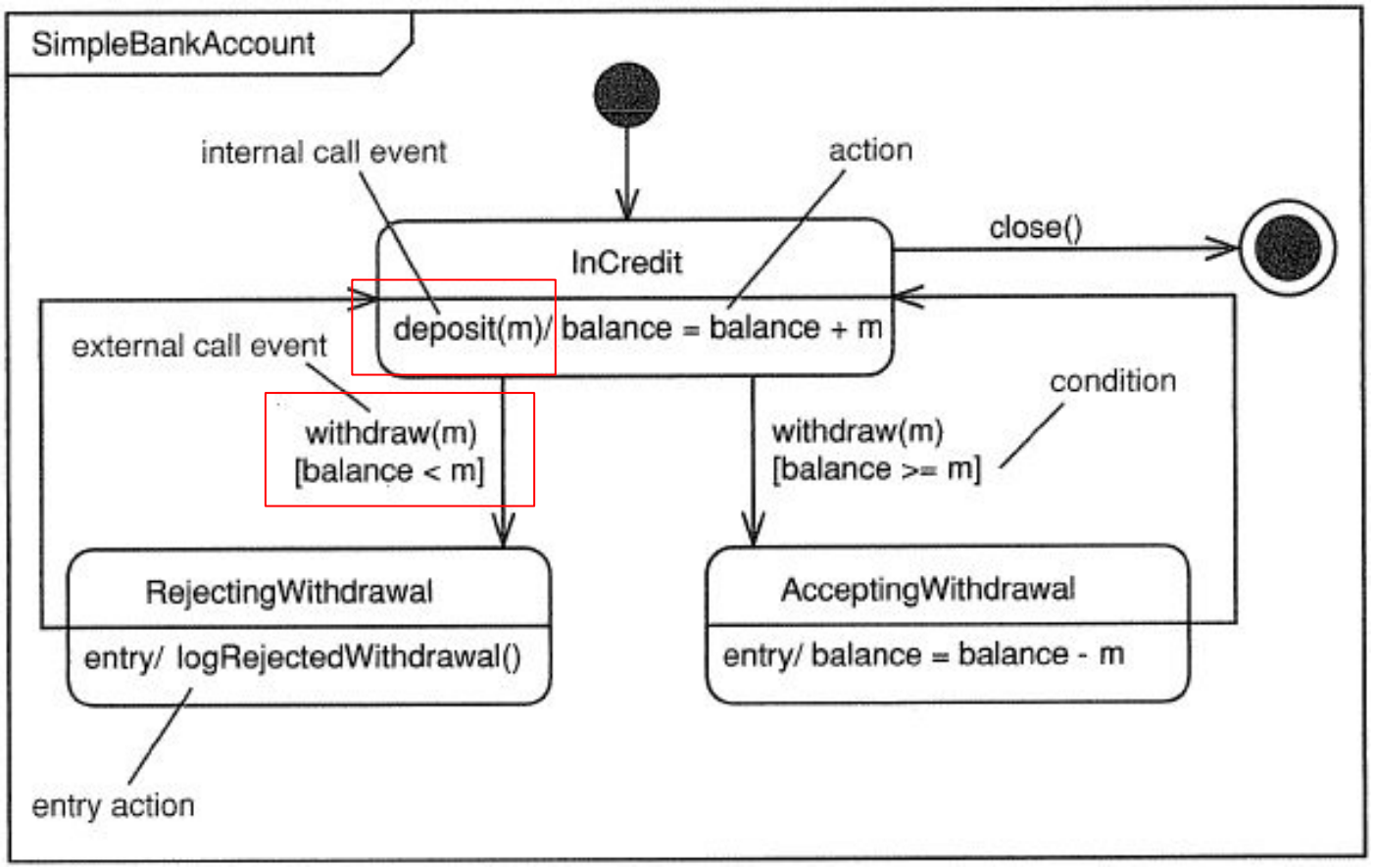
- Relação entre dois estados
- indicando que um objecto no **estado fonte** executará certas **acções** e passará para o **estado alvo**, quando um conjunto específico de **eventos** ocorrerem e **condições** forem satisfeitas
- Uma transição entre estados ocorre assim:
  - O objecto está num **estado fonte**
  - Um **evento** ocorre
  - Se uma **condição** for satisfeita (opcional)
  - Uma **acção** é executada
  - O objeto entra no **estado alvo**

Event

- Specify a significant occurrence
  - Corresponds to a stimuli that inits a transition of states
- Events can be:
  - **Internal**: between system objects.
    - Eg.: exception *overflow*
  - **External**: between the system and the actors
    - Eg.: press a button to cancel the transaction
- Types of events: signal, invoke/call, time and change

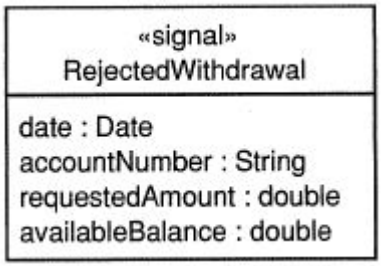
Call Event

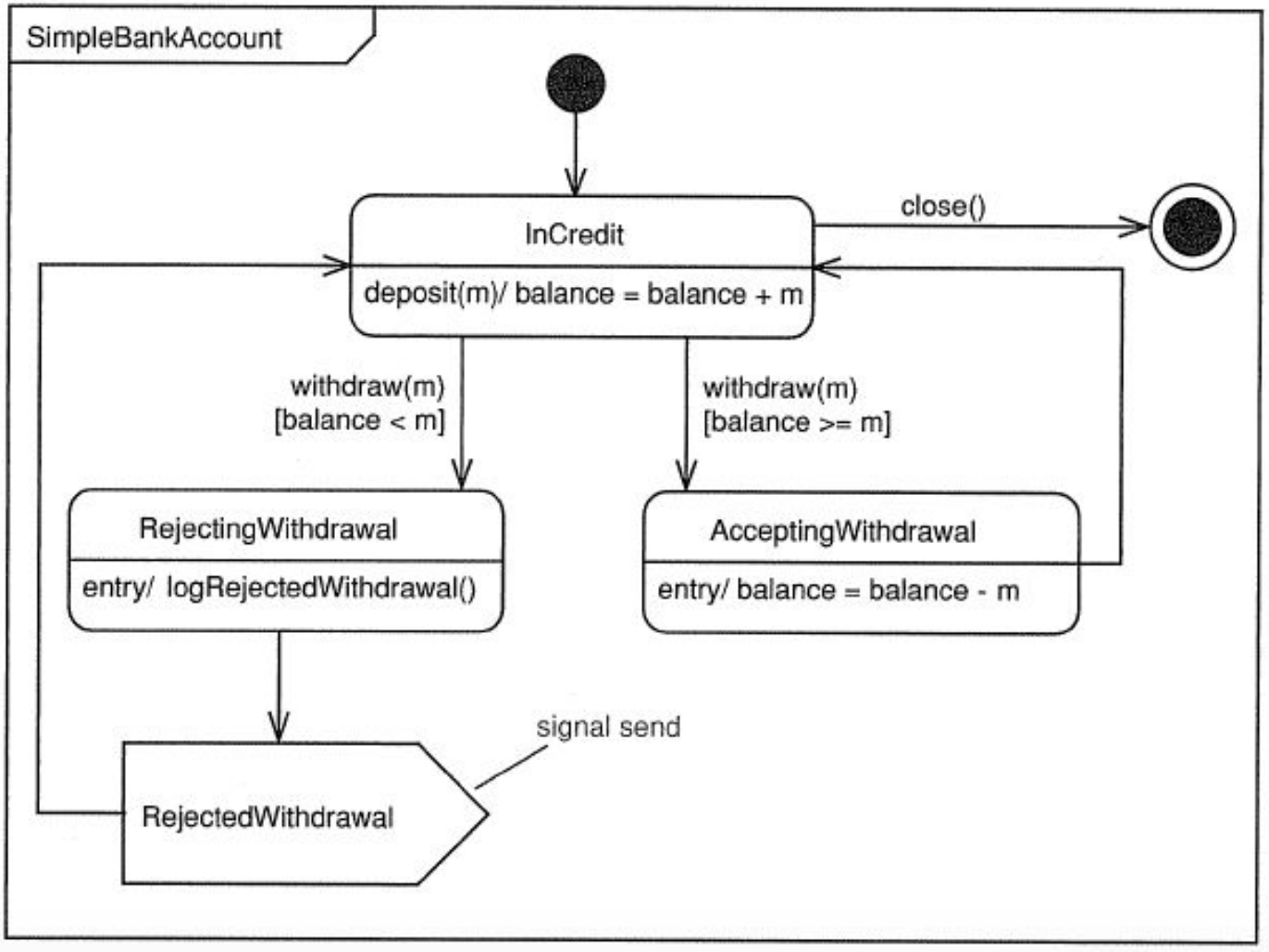
request for a specific operation to be invoked on an instance of the **context class (**should have the same signature)



Signal Event

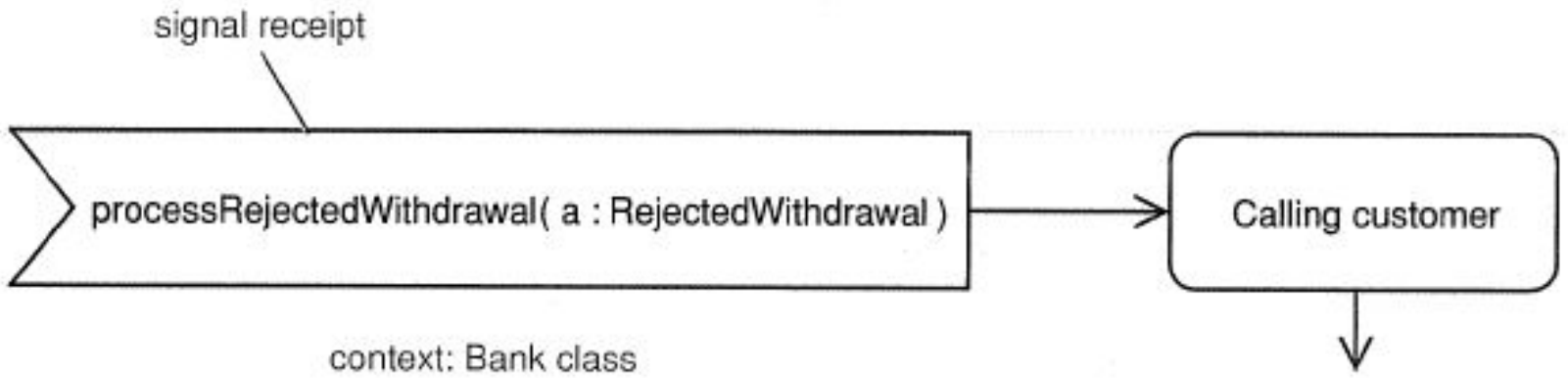
Information sent asynchronously between objects





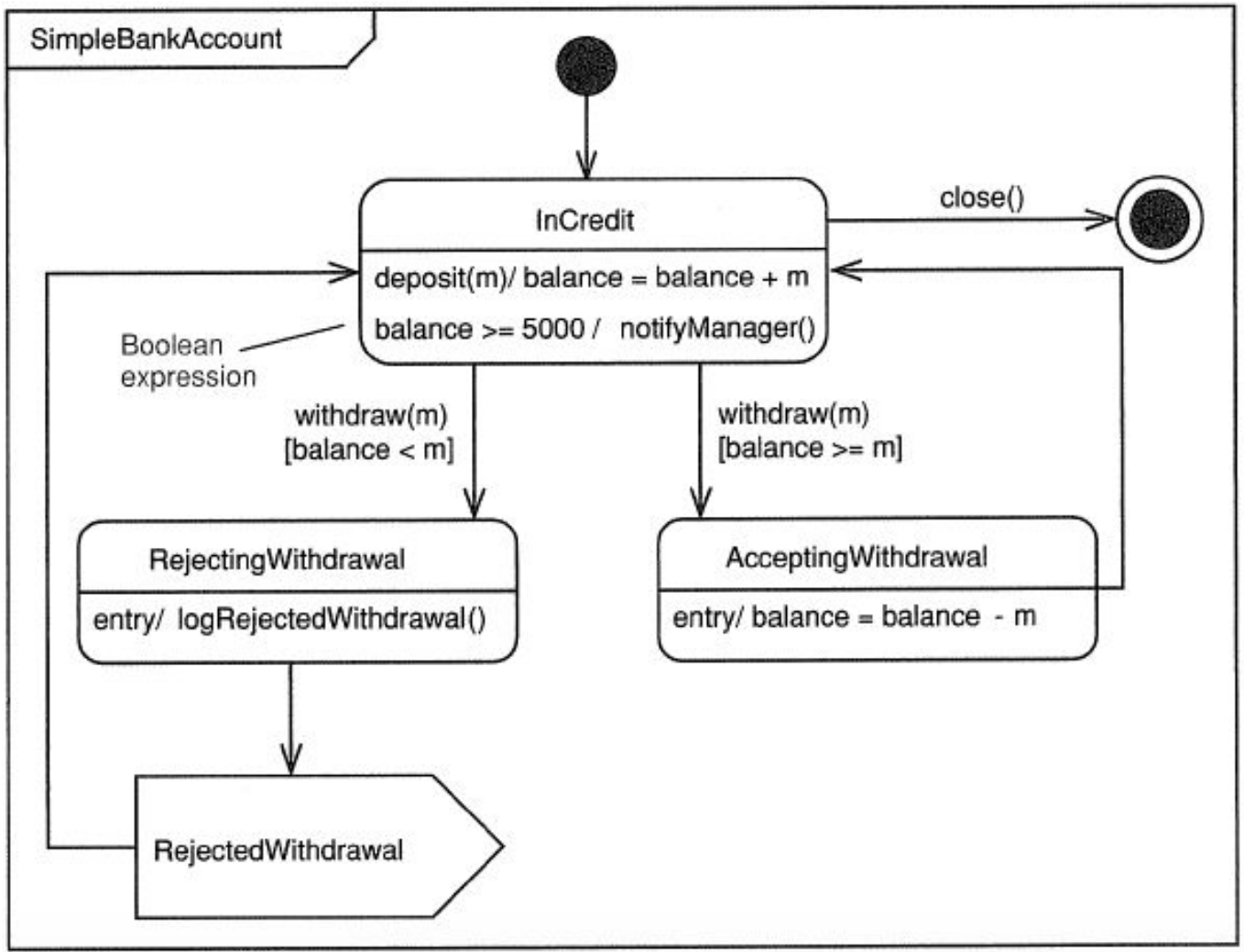
Signal Event (cont.)

operation of the context class



Change Event

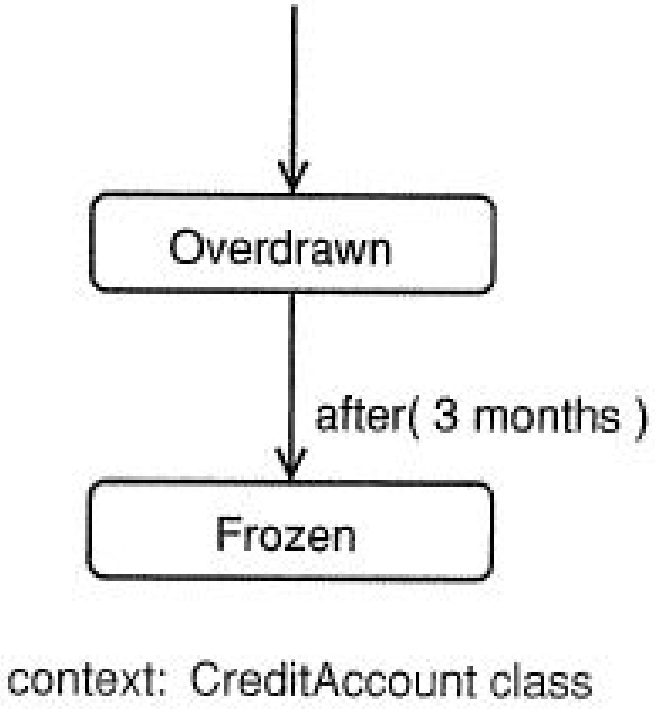
Boolean expressions that transition from false to true:



Time Events

keywords **when** and **after**

when(date=1/1/2020) after(3 months)



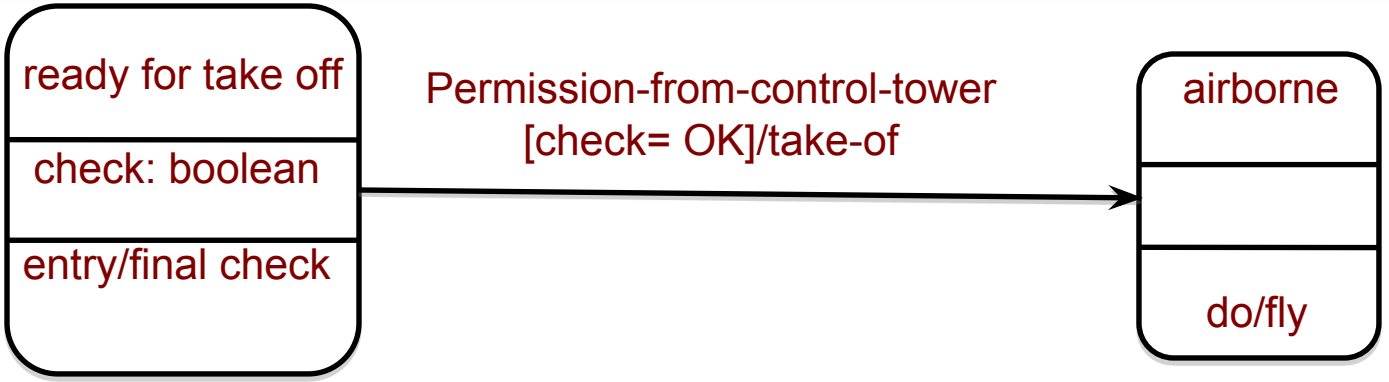
Action

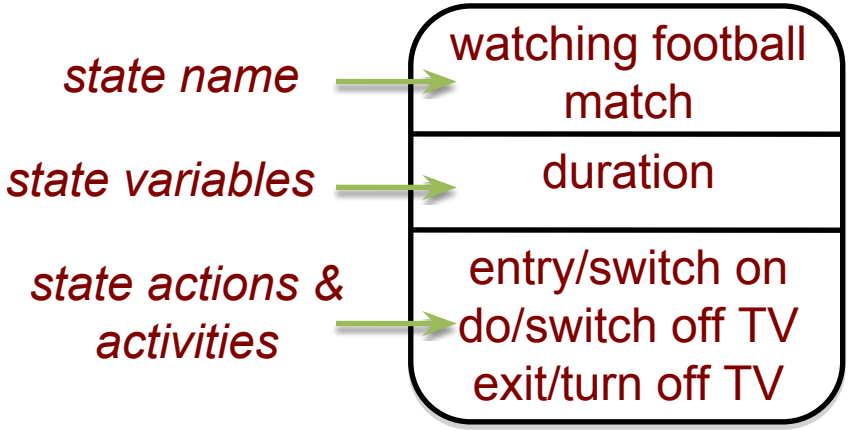
- Instantaneous, non-interruptible
- Where?
  - In the transition
  - While entering a state = action in all entry transitions
  - While exiting the state = action in all exit transitions
  - In events
- Examples:
  - Create or destroy an object in the system
  - Send a signal to other objects

Activity

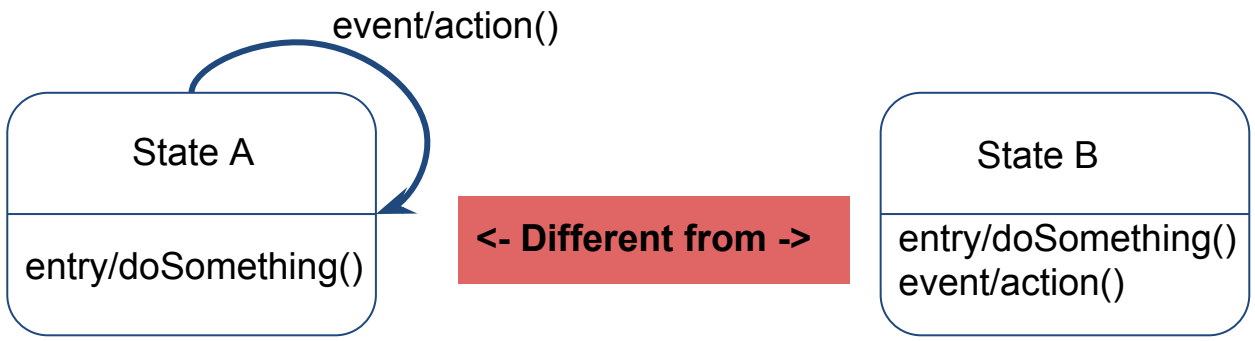
- Takes time, interruptible
- Where?
  - In the state
- Examples:
  - Calculate the IRS
  - Calculate the amount to pay for the parking

Advanced States





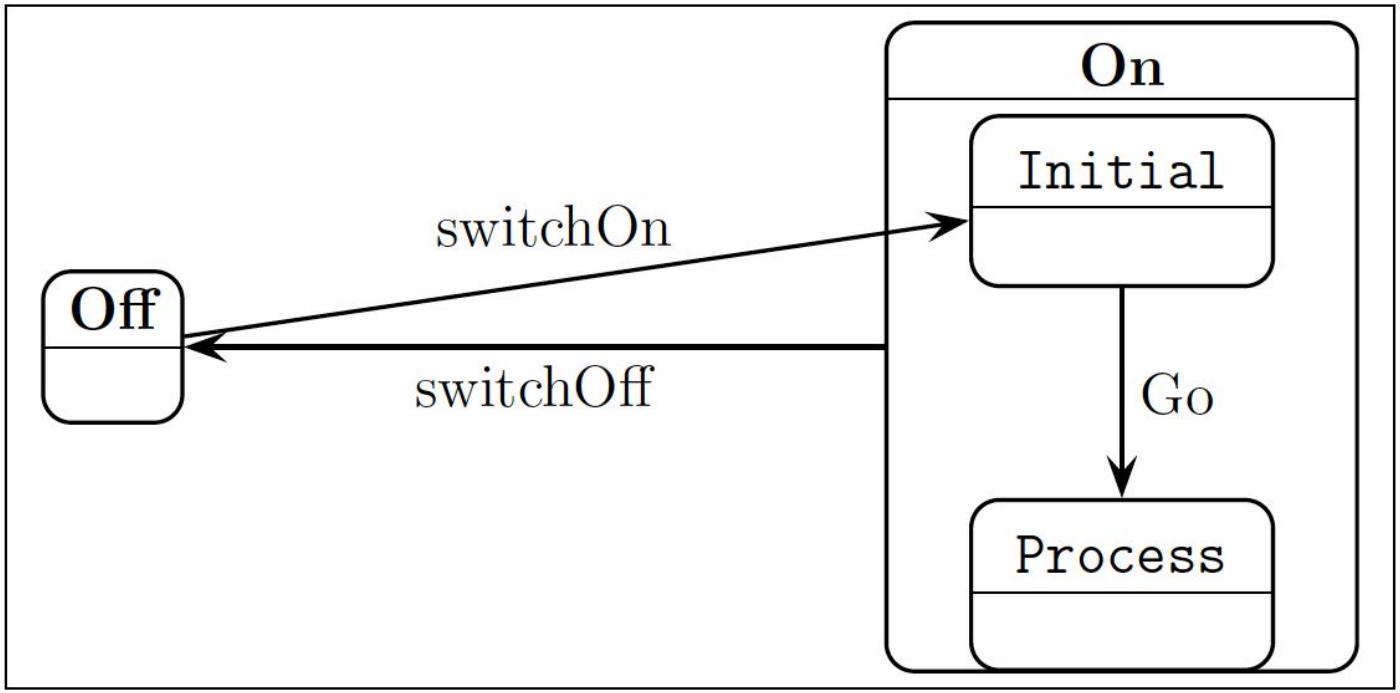
Difference between



Composed States/sub-states

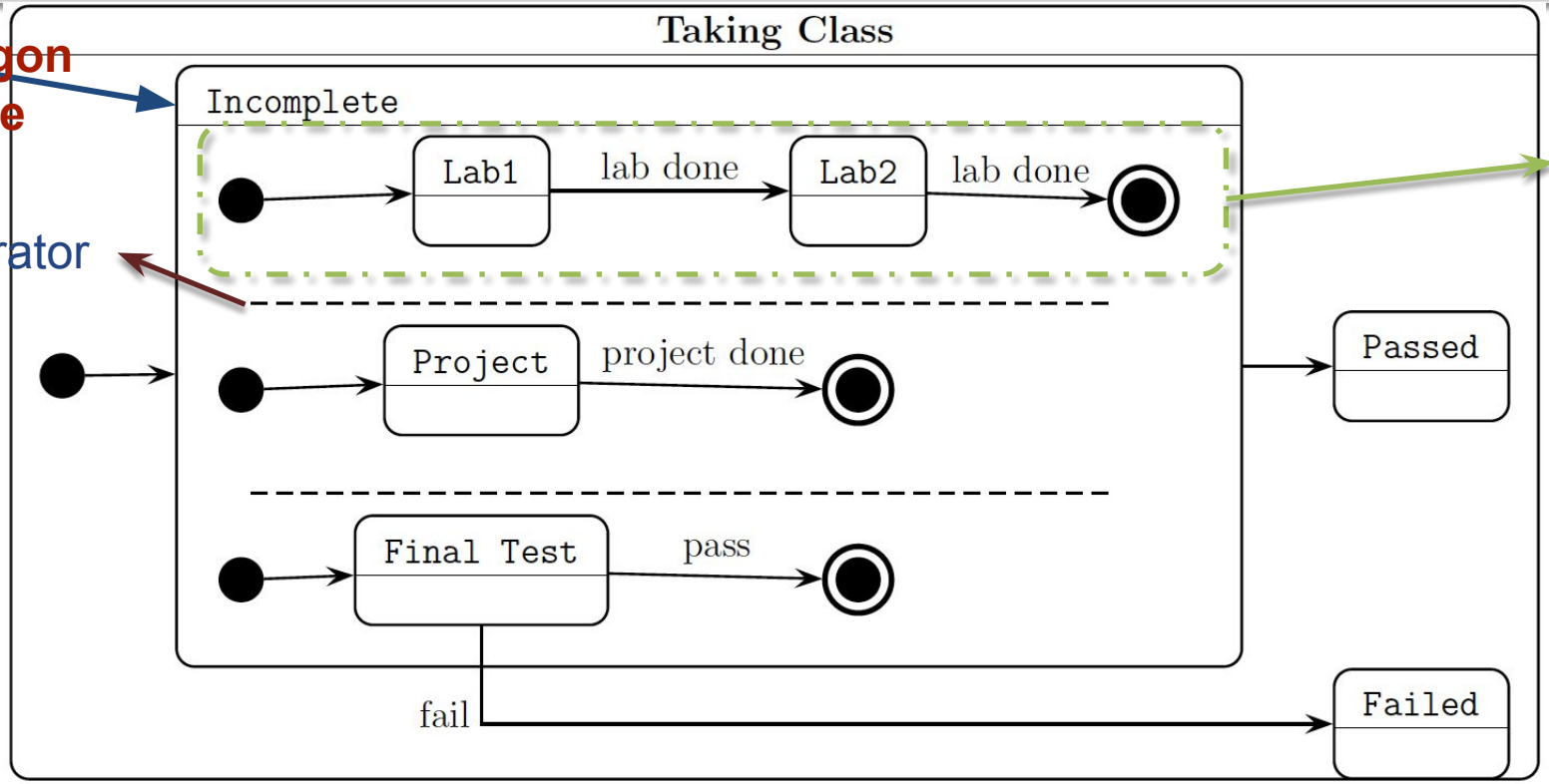
- Allow for modeling complex behavior
- Similar Sub-states are grouped together in a composed state (nesting)
- Composed States can be transitions, Entry and Exit actions, activities, ….
- Transitions, can connect states at different sub-levels
  - Sub-states “inherit” from a composed state
- Sub-states can be concurrent (orthogonal) or sequential (disjunct)

Composed States: example



Initial, **Process** are substates of **On**
Initial, **Process** “inherit” from transition **switchOff**

Composed concurrent states (parallel or Orthogonal)

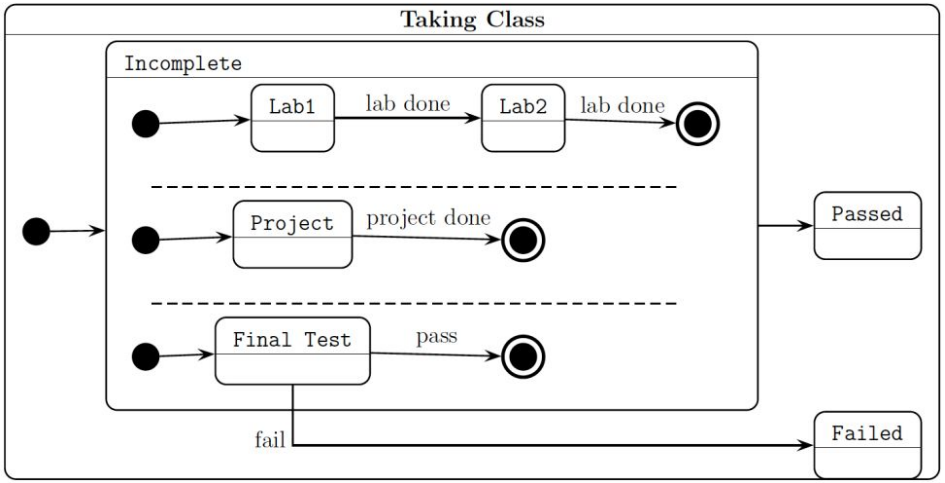


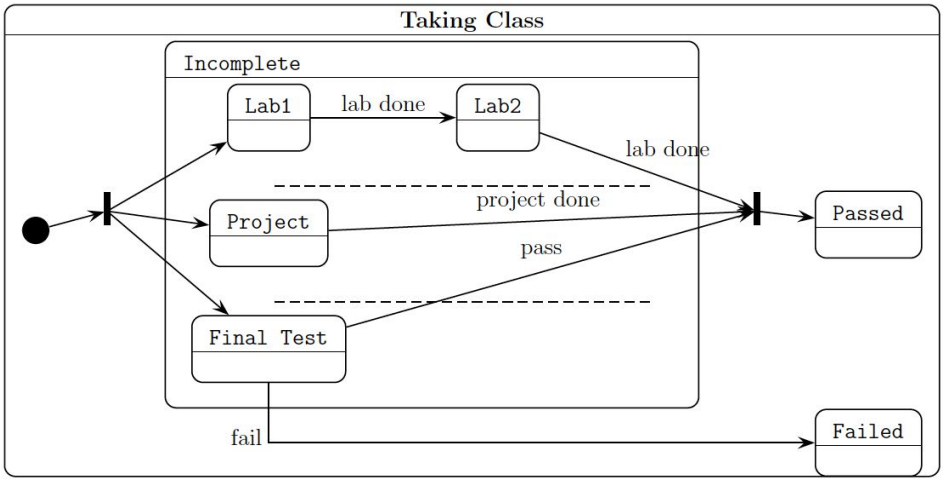
Regions

Independent concurrent parts of a composed state

Are activated synchronously (when the composed state is activated)

Composed States: Entering



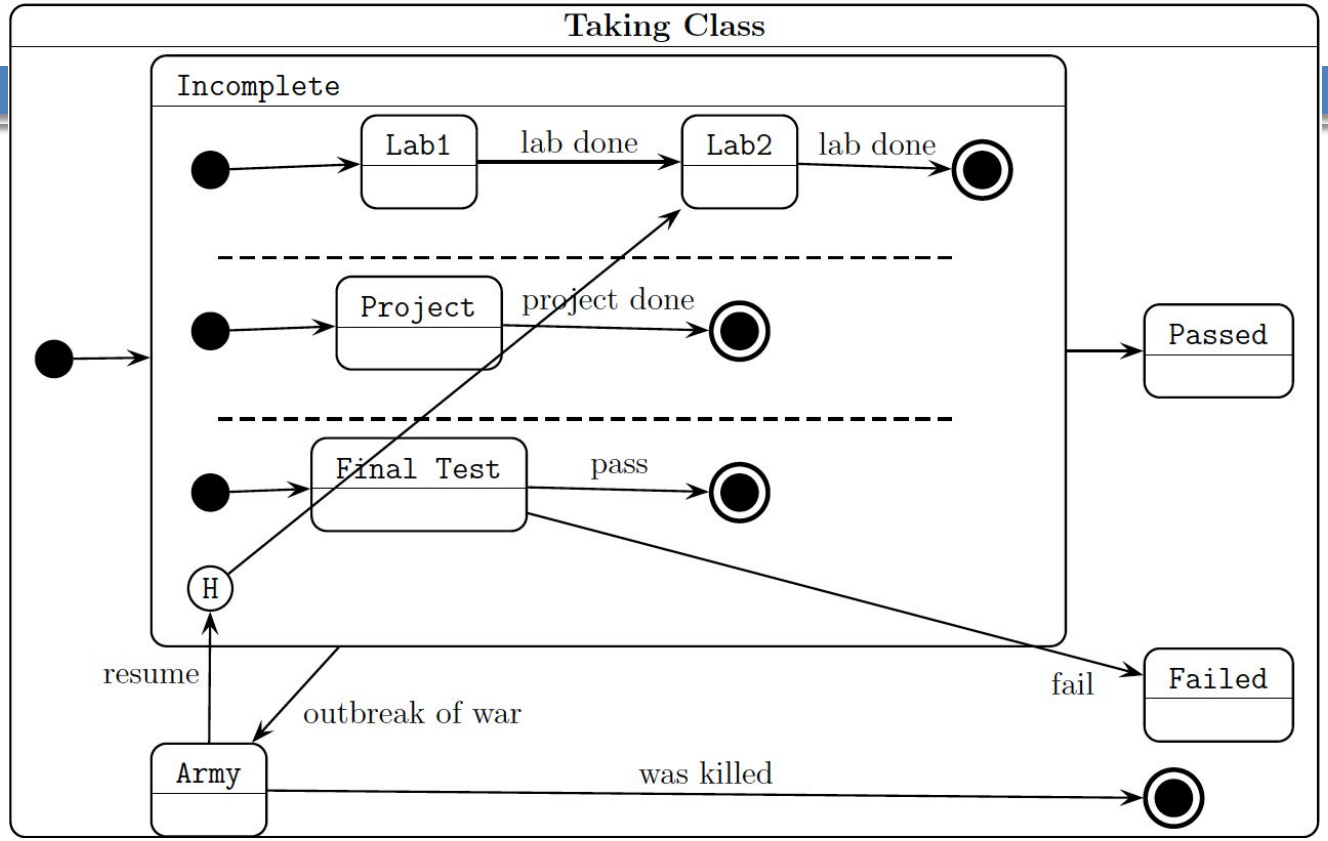


To Enter a composed state
There must be an initial sub-state in each region
To Enter a sub-state
There must be a sub-state in all other regions

Concurrent Transition
Alternative notation to get in a composed state

Use pseudo-states "**fork**" and "**join**"

History state



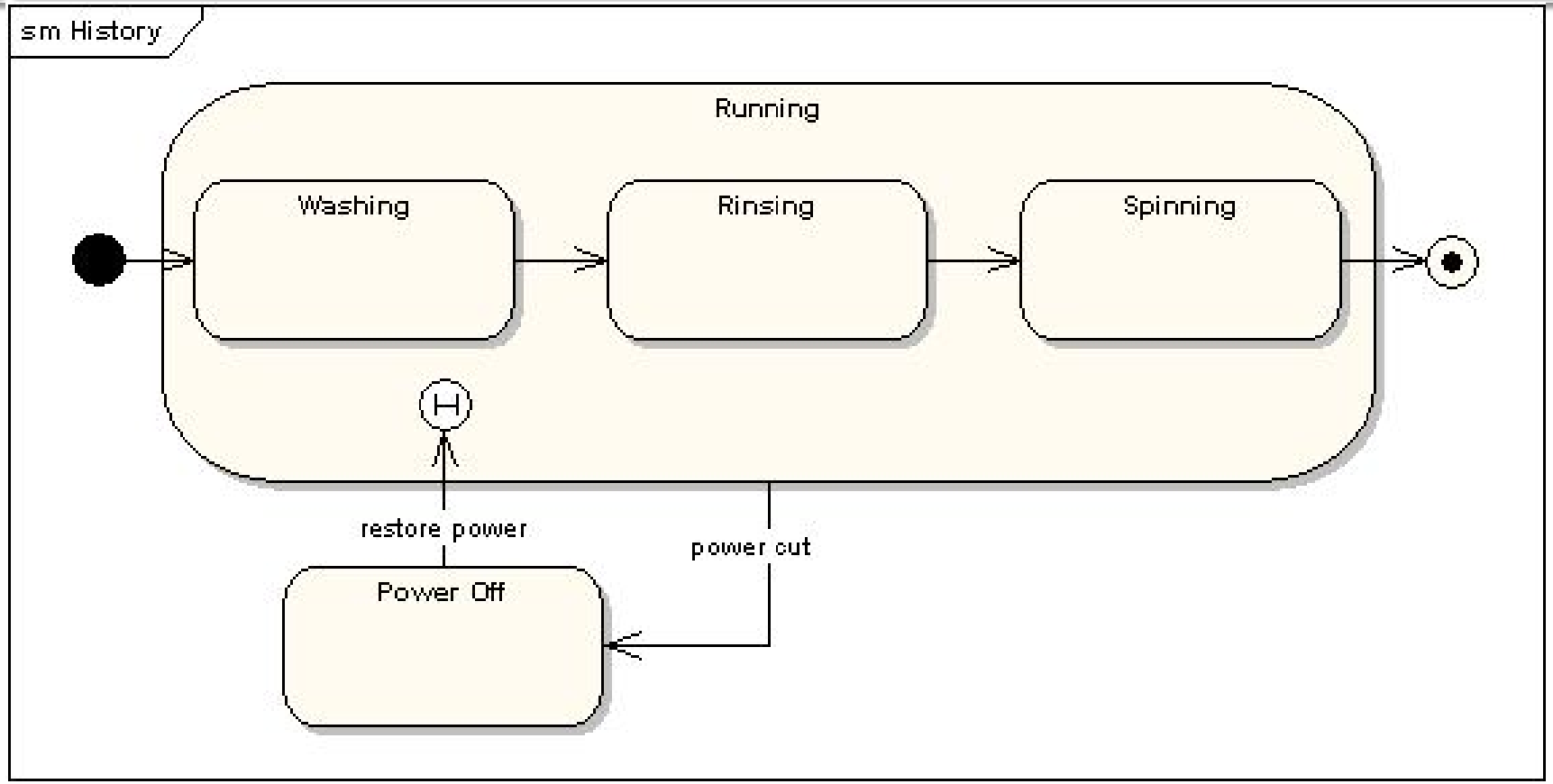
- **Shallow (H)**: remembers the history of a composed state (remembers the last active sub-state, but not from their sub-states)
- **Deep (H*)**: remembers also the nested history of the sub-states (remembers the configuration of active states before exiting from it)

Sub-machine state

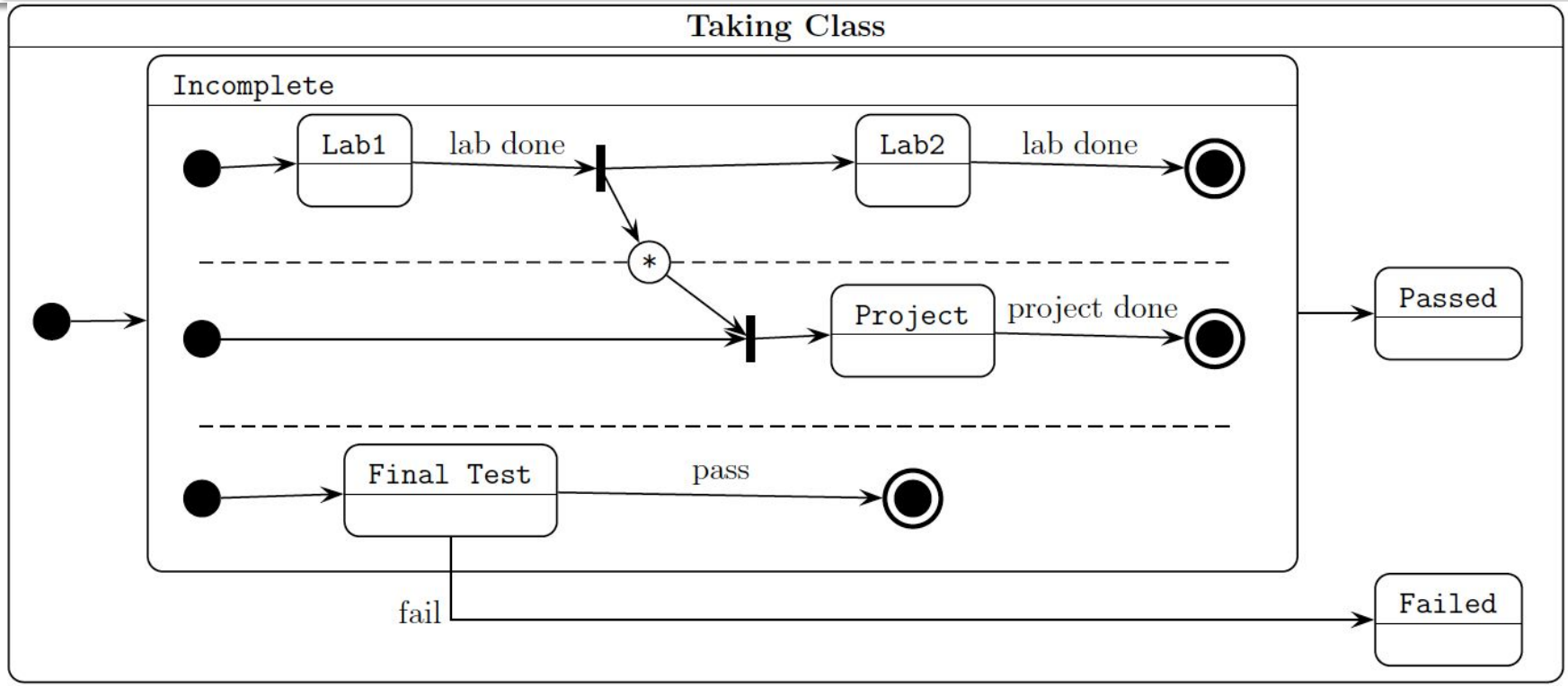
- **A state with a sub-machine is a state required if we want to re-use an existing state-machine**

<name of state> : < name of state machine>

History state

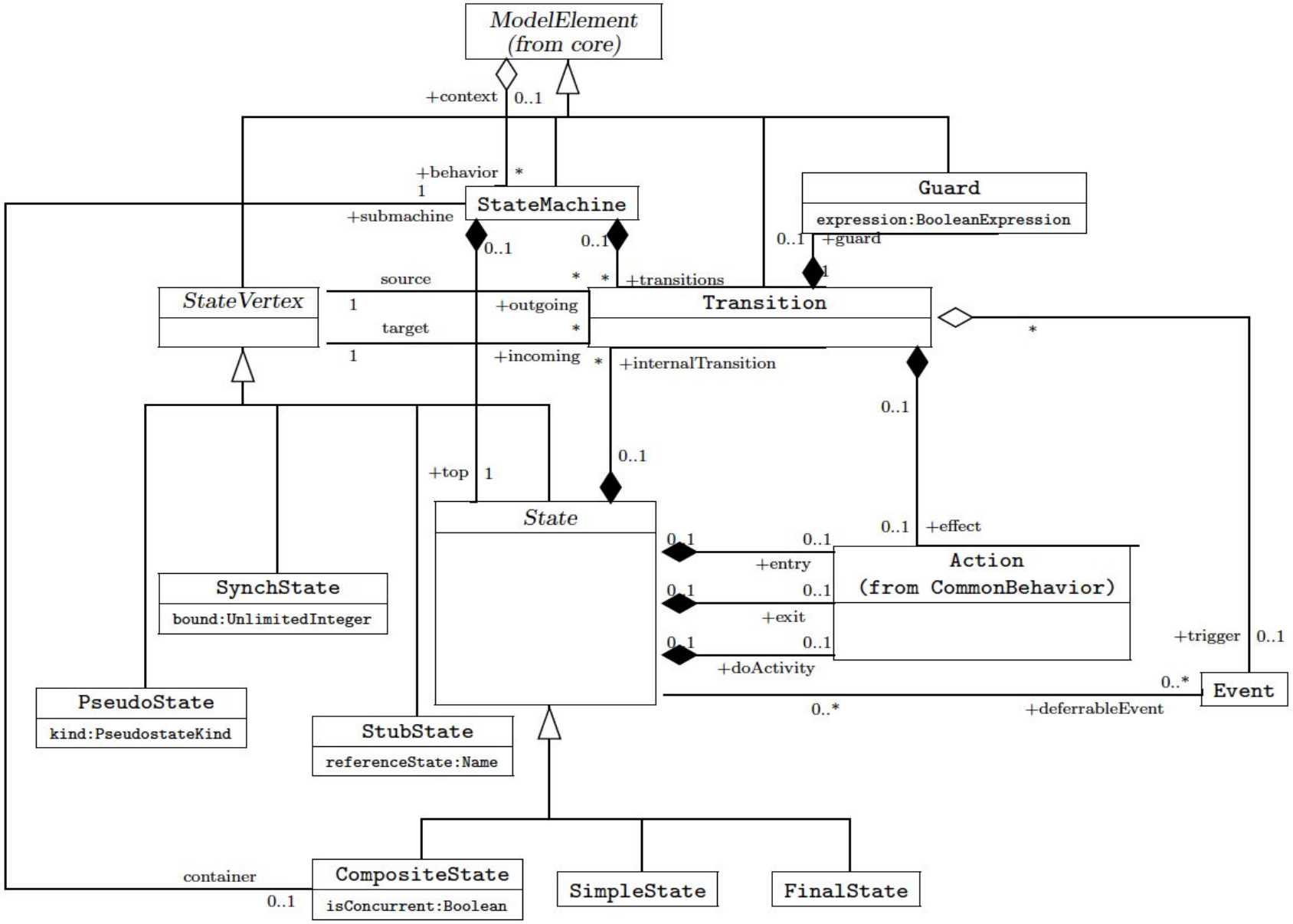


Synch state



- Allows for the synchronization in between regions
- Used in combination with **fork** and **join**

Metamodel


ModelElement
(from core)
+context
0..1
+behavior
*
1
StateMachine
+submachine
0..1
0..1
Guard
expression:BooleanExpression
0..1
+guard
1
source
*
*
+transitions
StateVertex
1
+outgoing
Transition
target
*
1
+incoming
*
+internalTransition
*
0..1
0..1
+top
1
State
0..1
+effect
0..1
0..1
+entry
Action
(from CommonBehavior)
0..1
0..1
+exit
0..1
0..1
+doActivity
+trigger
0..1
0..*
Event
0..*
+deferrableEvent
SynchState
bound:UnlimitedInteger
PseudoState
kind:PseudostateKind
StubState
referenceState:Name
container
0..1
CompositeState
isConcurrent:Boolean
SimpleState
FinalState

Exercise: a state diagram for the drink vending machine

- Consider:
  - Insert coins
  - Select product
    - May not be available
  - Verify amount
  - Give drink
  - Cancel operation

Drink Vending Machine

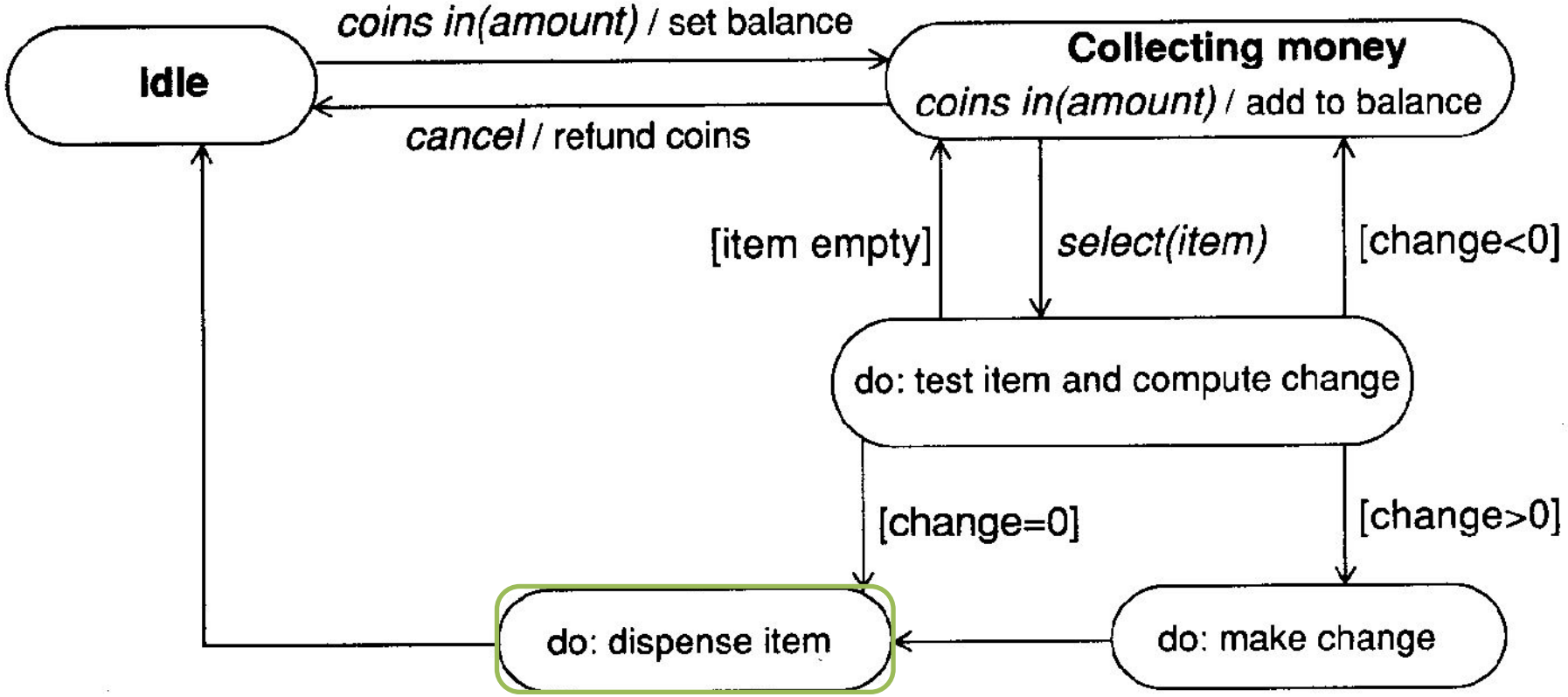


Figure 5.12 Vending machine model

Dispense item

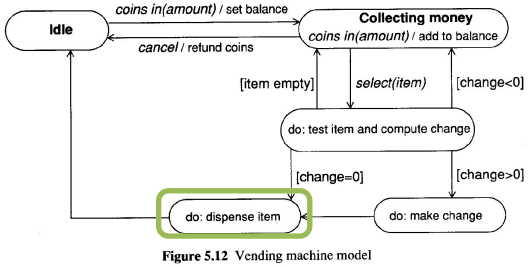


Figure 5.12 Vending machine model

Example: Microwave Oven

- Specify a state diagram based on the states and transitions described below

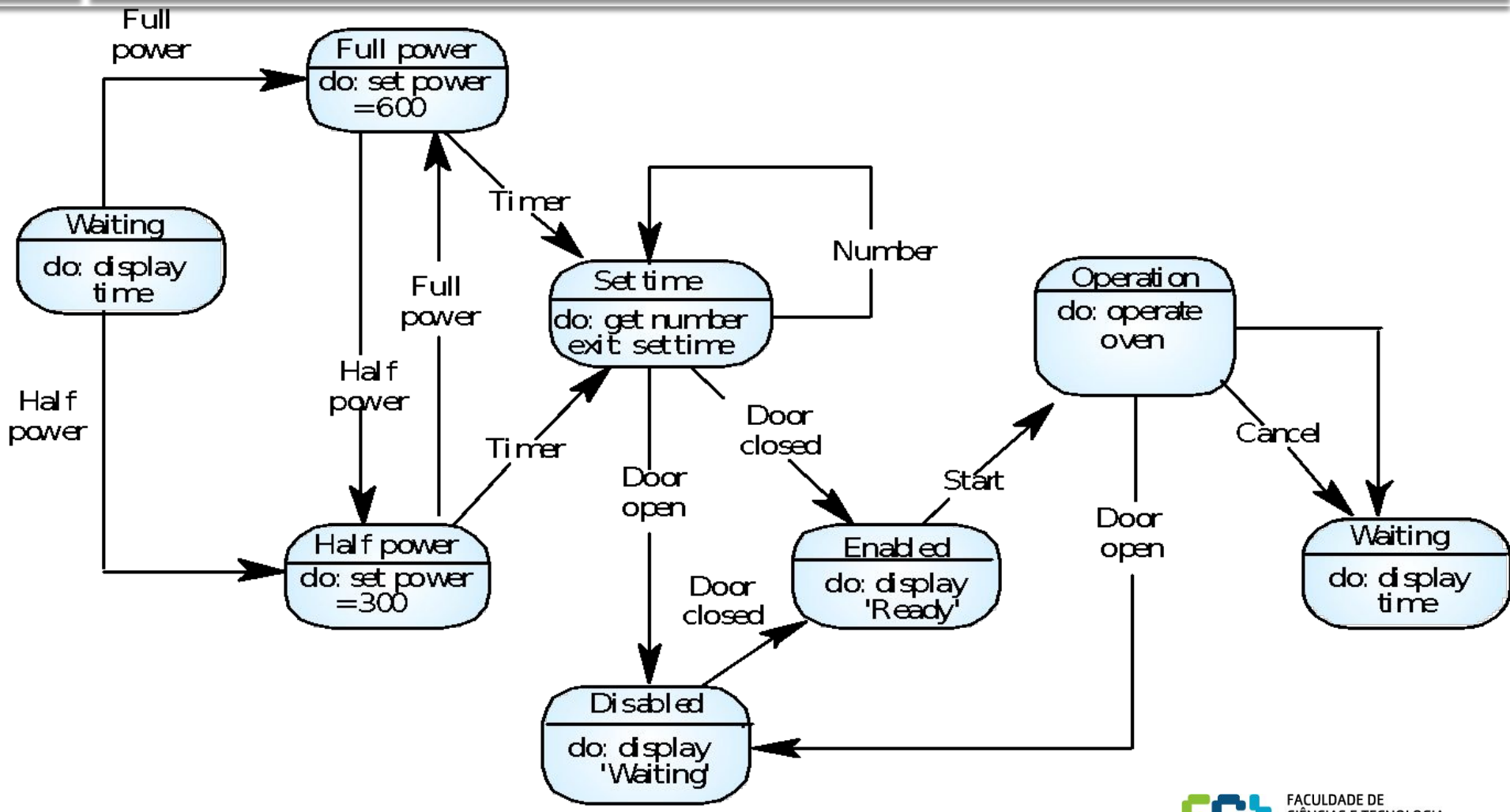
Descrição dos estados de um microondas

Estado	Descrição
Waiting	O forno está a espera de uma entrada. O display mostra a hora corrente.
Half power	A potência é determinada par 300 watts. O display mostra 'Half power'.
Full power	A potência é determinada para 600 watts. O display mostra 'Full power'.
Set time	O tempo de cozedura é introduzido pelo utilizador. O display mostra o tempo de cozedura seleccionado e sua actualização.
Disabled	A operação do forno é impossibilitada por segurança. A luz do forno é ligada. O display mostra 'Not ready'.
Enabled	A operação do forno é capacitada. A luz é desligada. O Display mostra 'Ready to cook'.
Operation	Forno em operação A luz é ligada.O Display mostra o timer sendo decrementado. Ao fim da cozedura o alarme é tocado por 5 segundos. O Display mostra 'Cooking complete' enquanto o alarme toca.

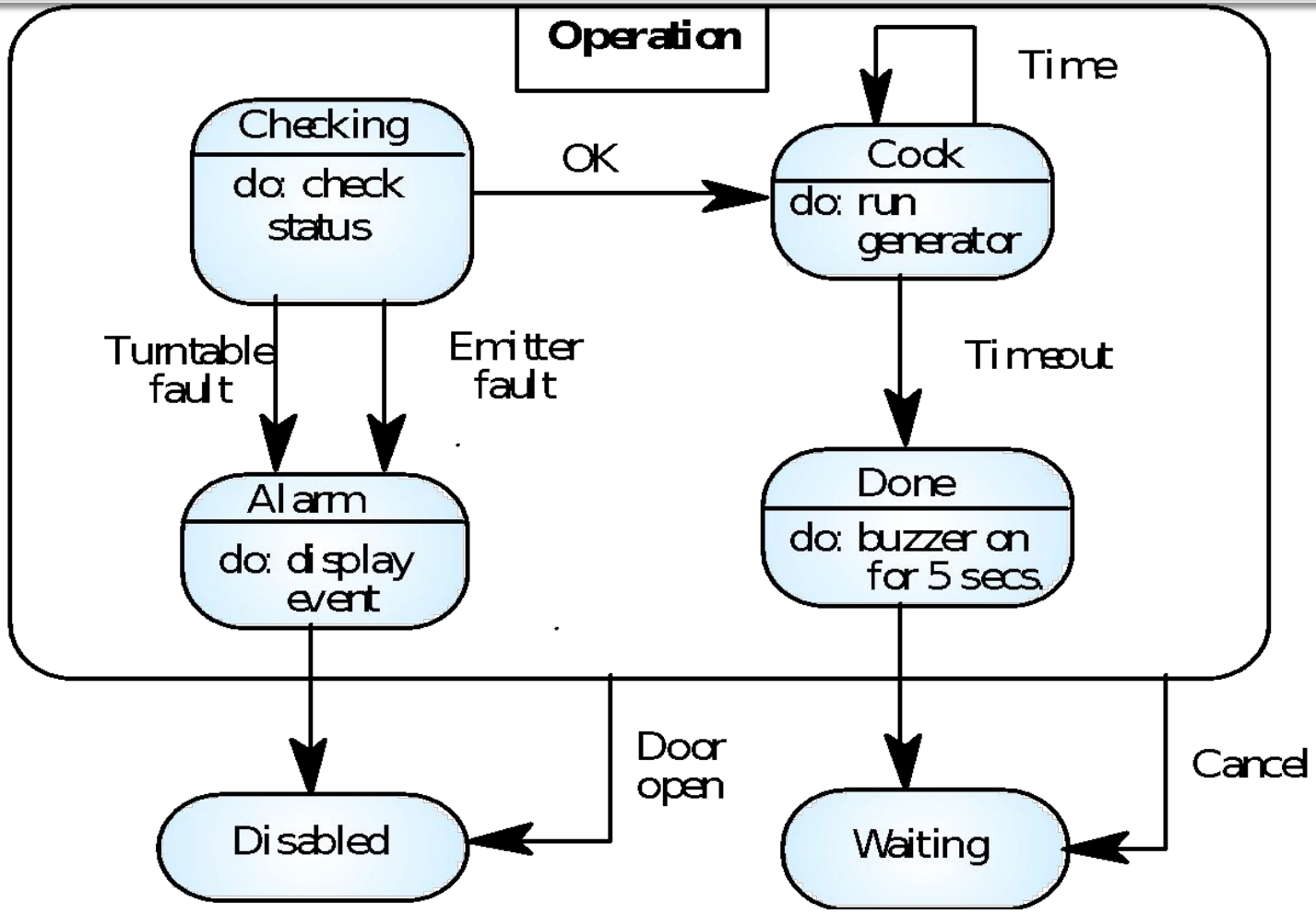
Microwave Stimuli

Estímulo	Descrição
Half power	O utilizador pressionou o botão half power
Full power	O utilizador pressionou o botão full power
Timer	O utilizador pressionou um dos botões do timer
Number	O utilizador pressionou uma chave numérica
Door open	A porta do microondas não está fechada
Door closed	A porta do microondas está fechada
Start	O utilizador pressionou o botão start
Cancel	O utilizador pressionou o botão cancel

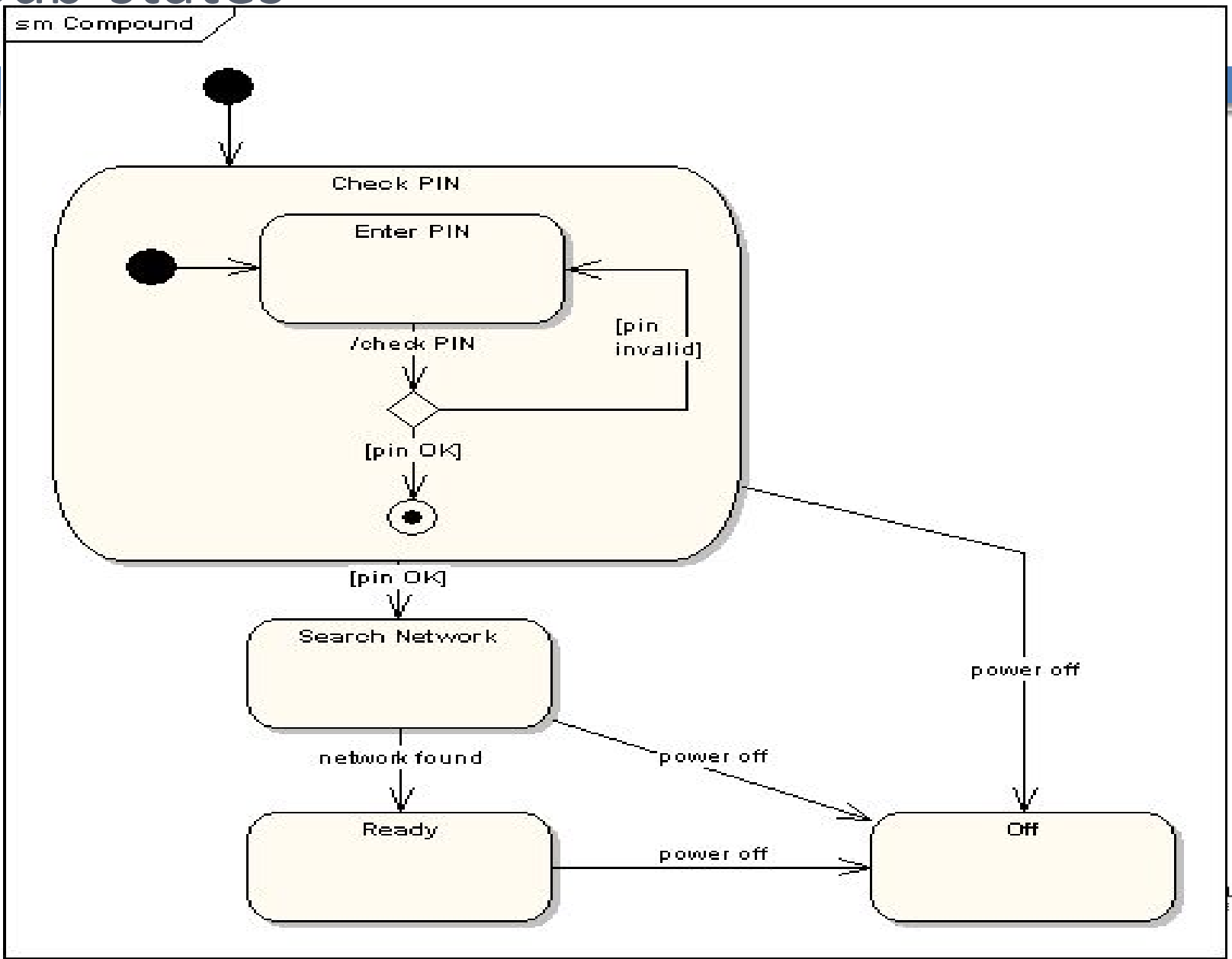
Modeling a Microwave



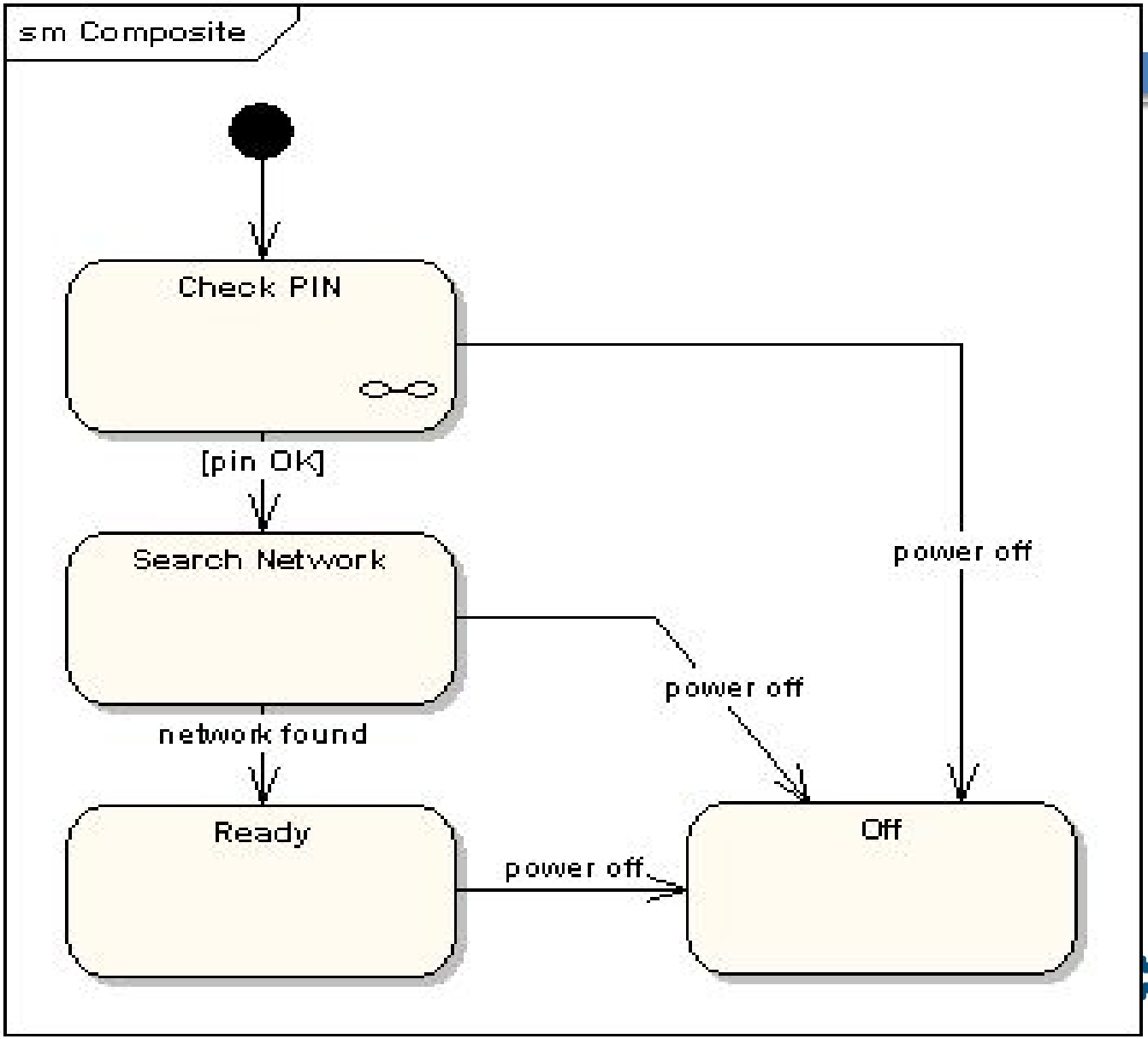
Microwave


Operation
Time
Checking
do: check status
OK
Cook
do: run generator
Turntable fault
Emitter fault
Timeout
Alarm
do: display event
Done
do: buzzer on for 5 secs.
Door open
Cancel
Disabled
Waiting

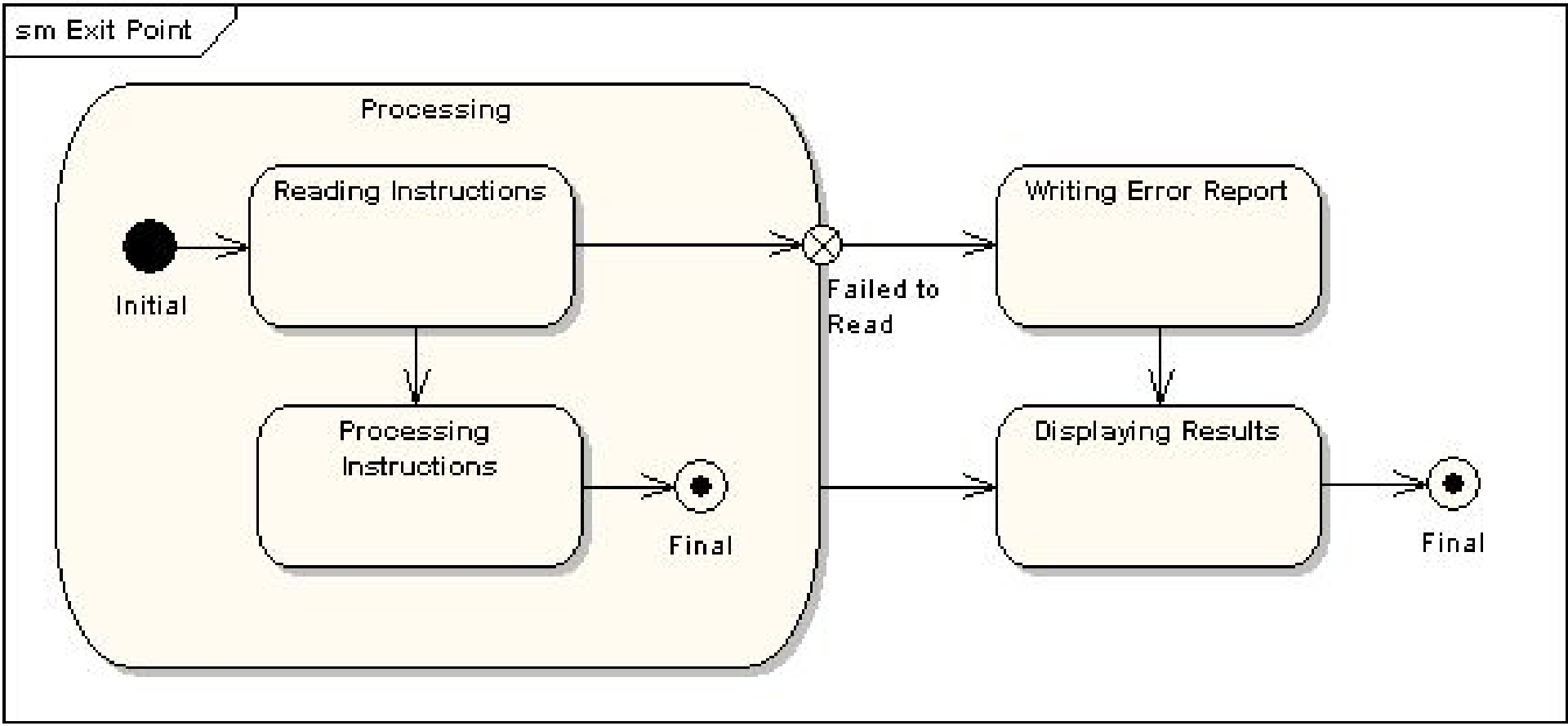
Sub-states


sm Compound
Check PIN
Enter PIN
/check PIN
[pin invalid]
[pin OK]
[pin OK]
Search Network
power off
network found
power off
Ready
power off
Off

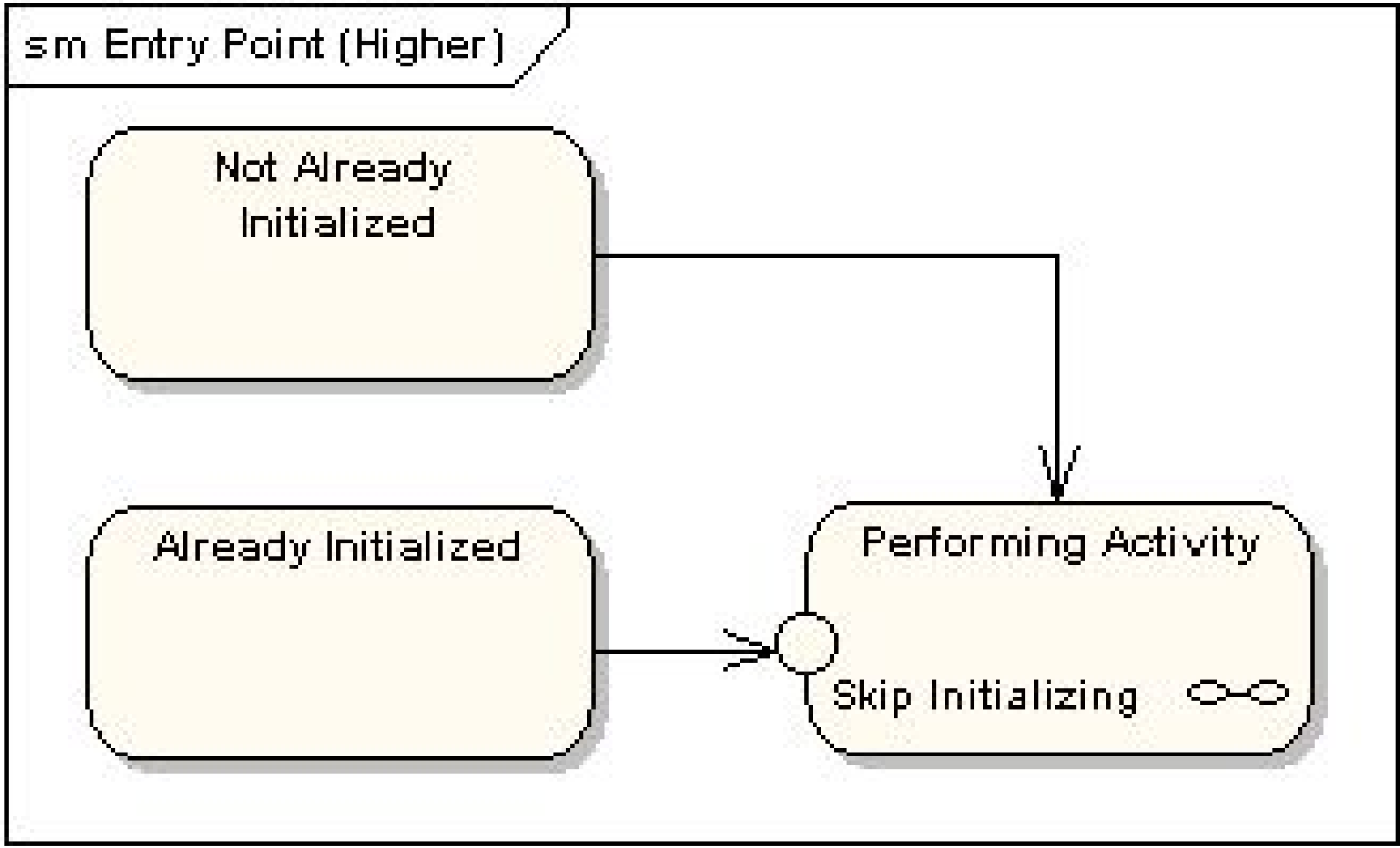
Sub-states


sm Composite
Check PIN
[pin OK]
Search Network
network found
Ready
power off
power off
power off
Off

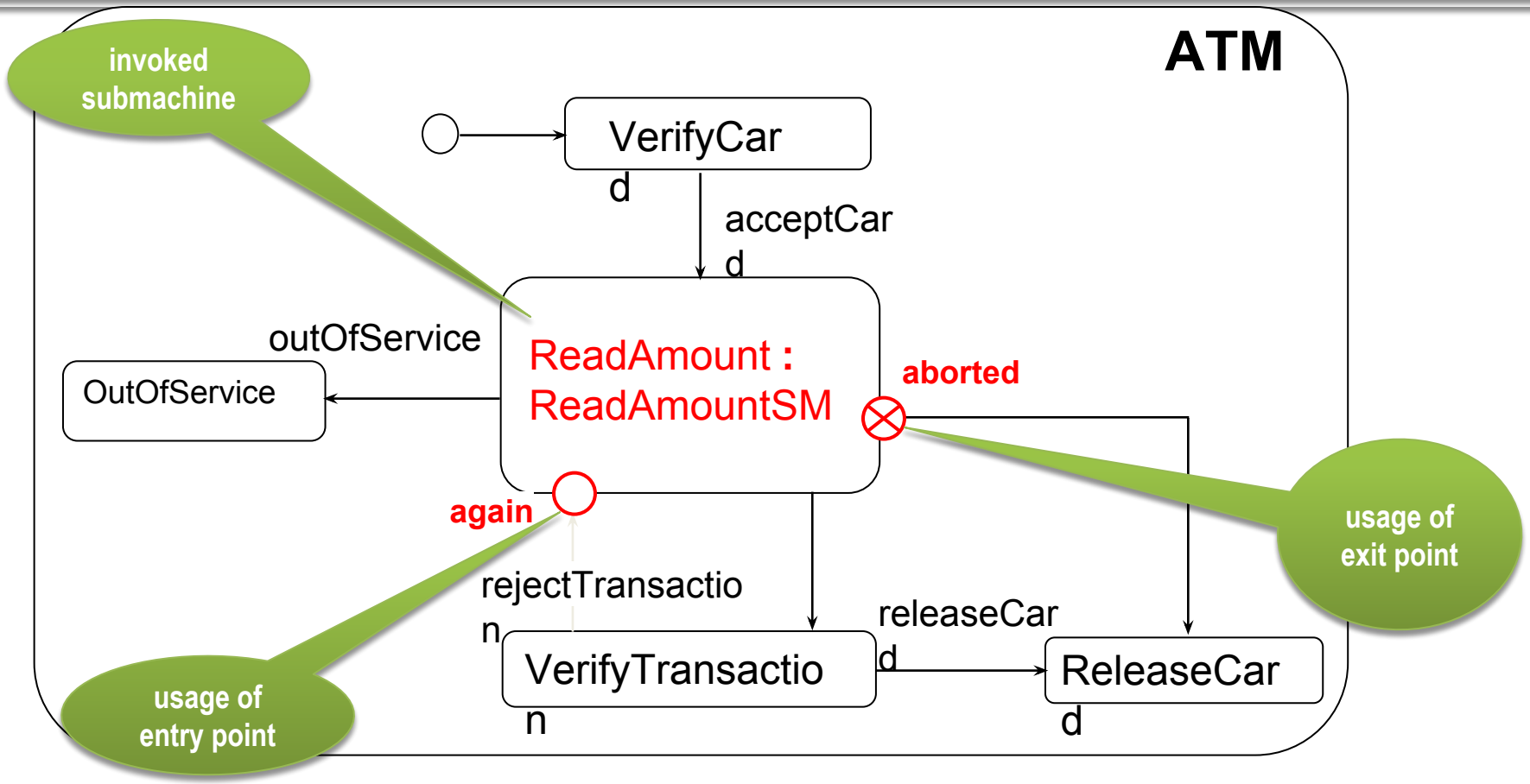
Exit point


sm Exit Point
Processing
Reading Instructions
Initial
Processing
Instructions
Final
Failed to
Read
Writing Error Report
Displaying Results
Final

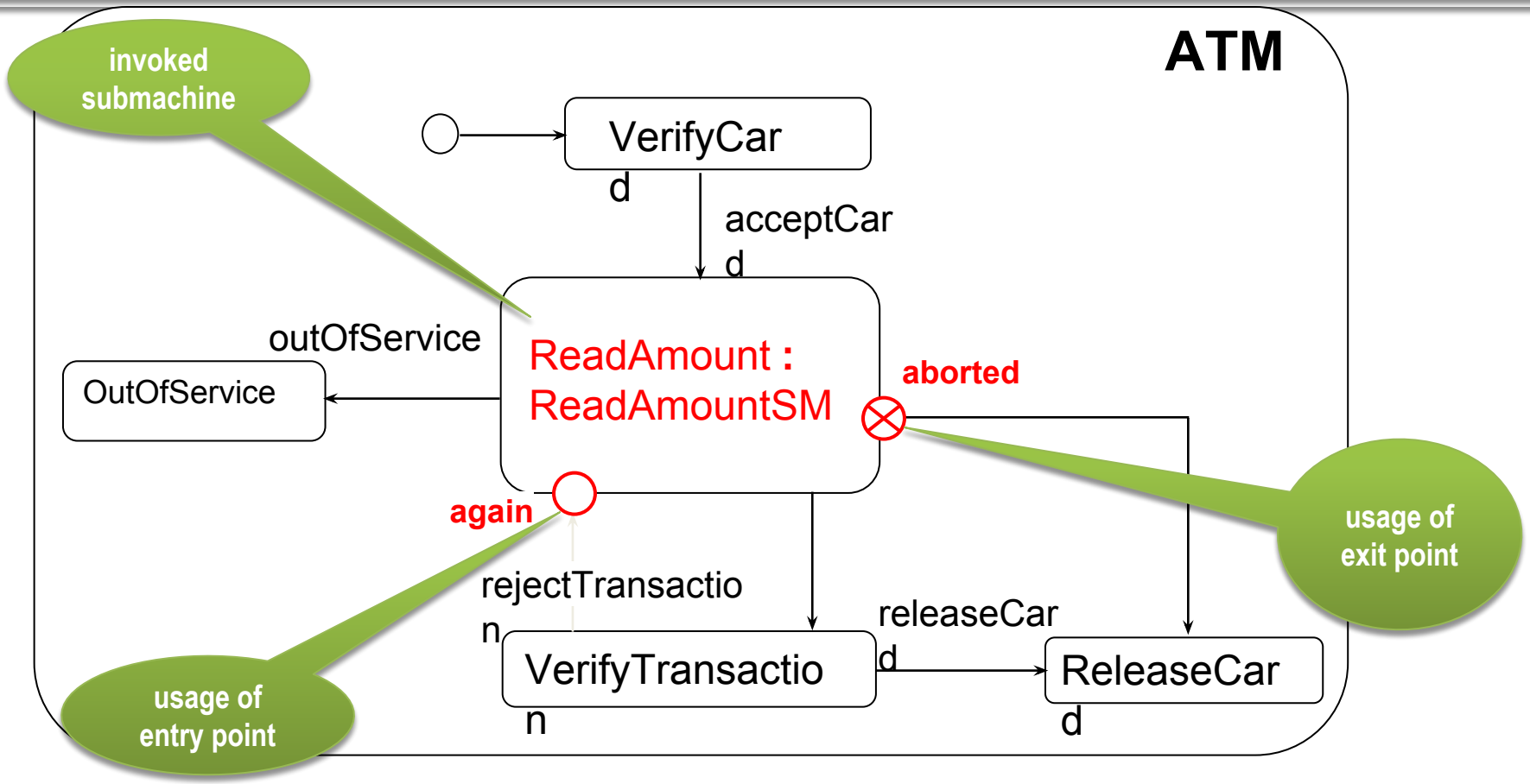
Entry Point


sm Entry Point (Higher)
Not Already Initialized
Already Initialized
Performing Activity
Skip Initializing

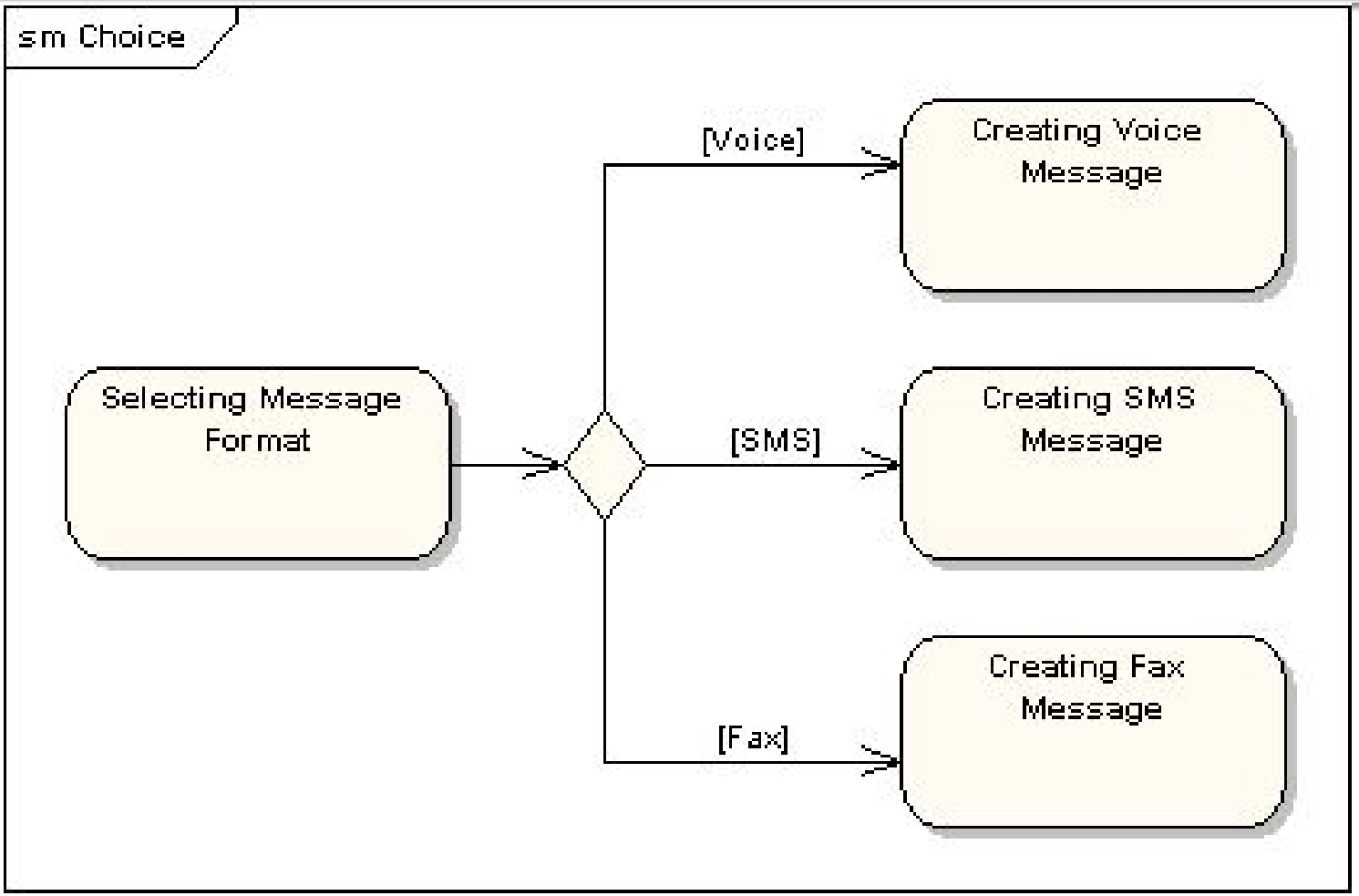
Entry and Exit points



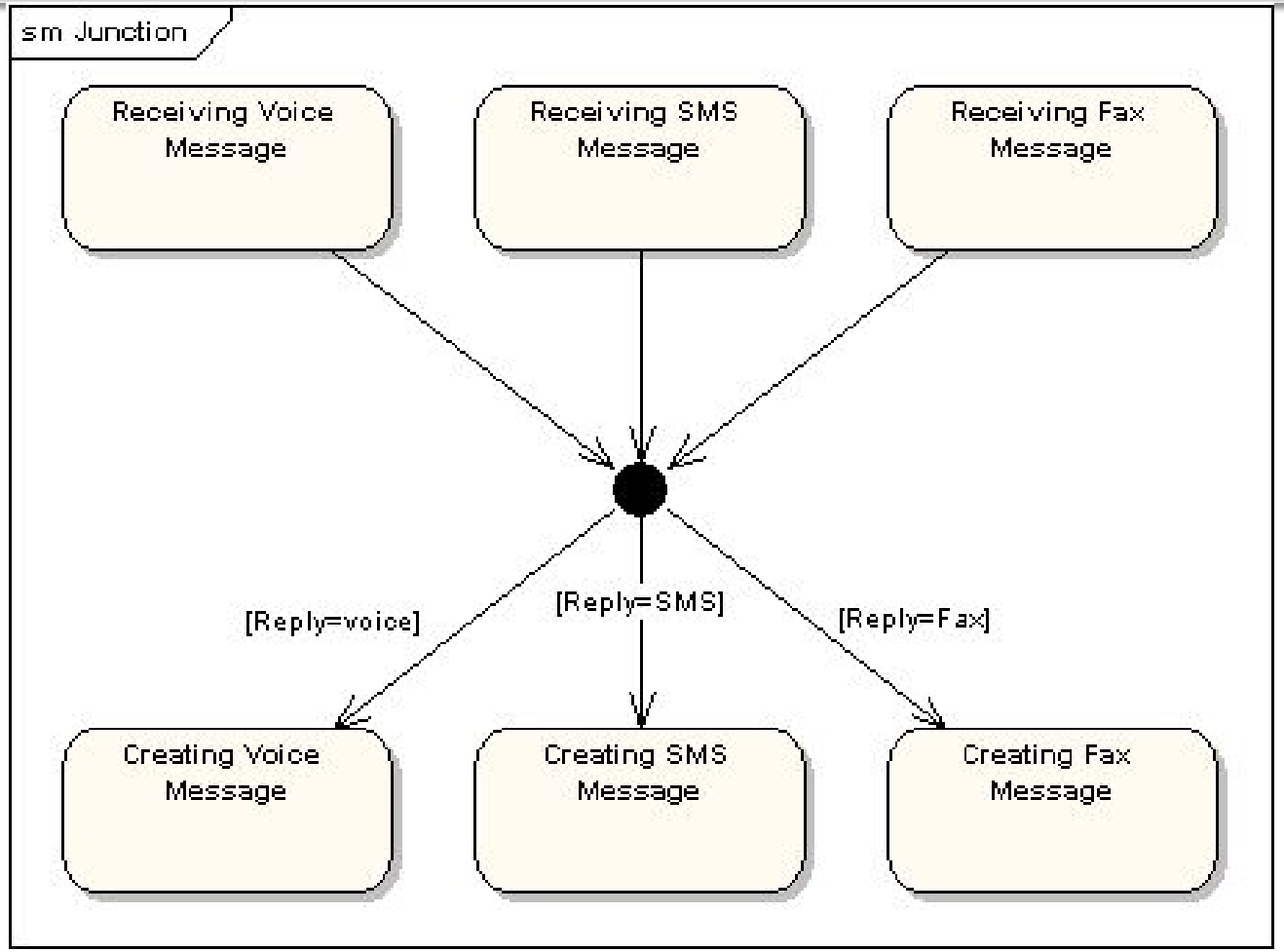
Pontos de entrada e saída



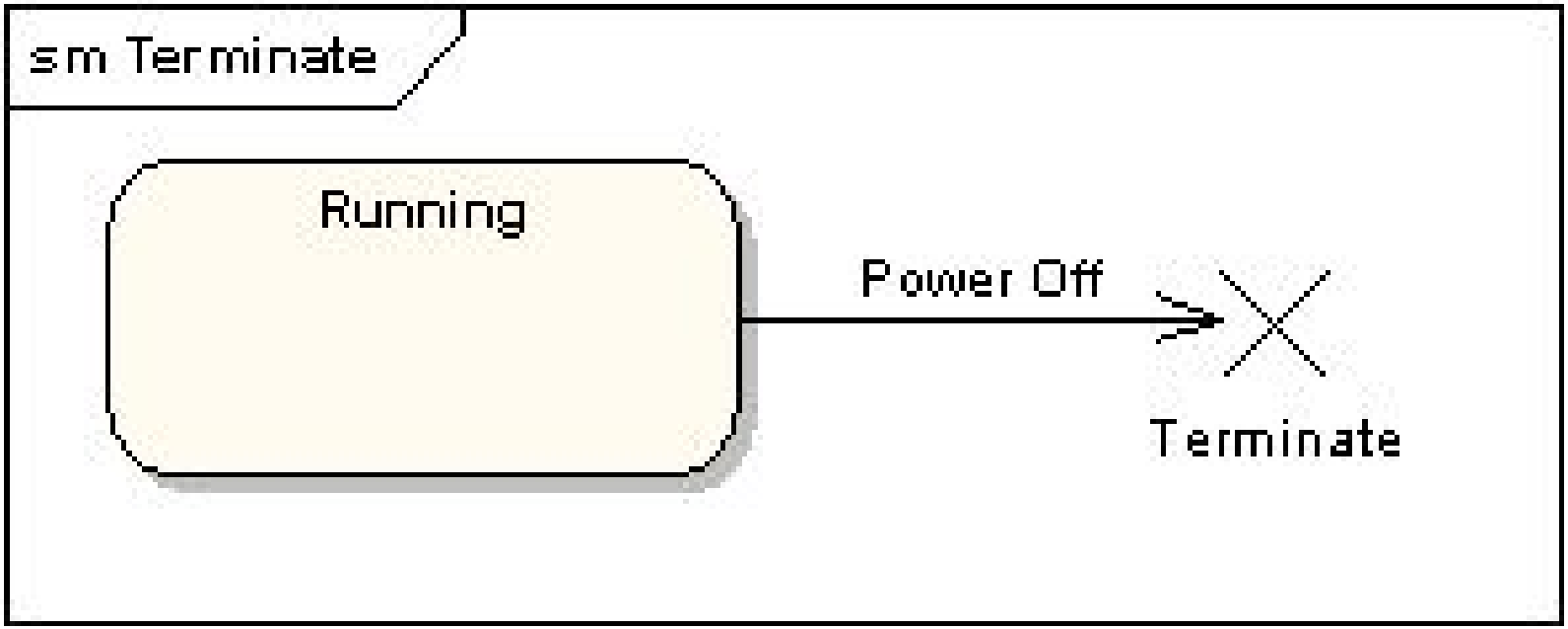
Pseudo-state selection



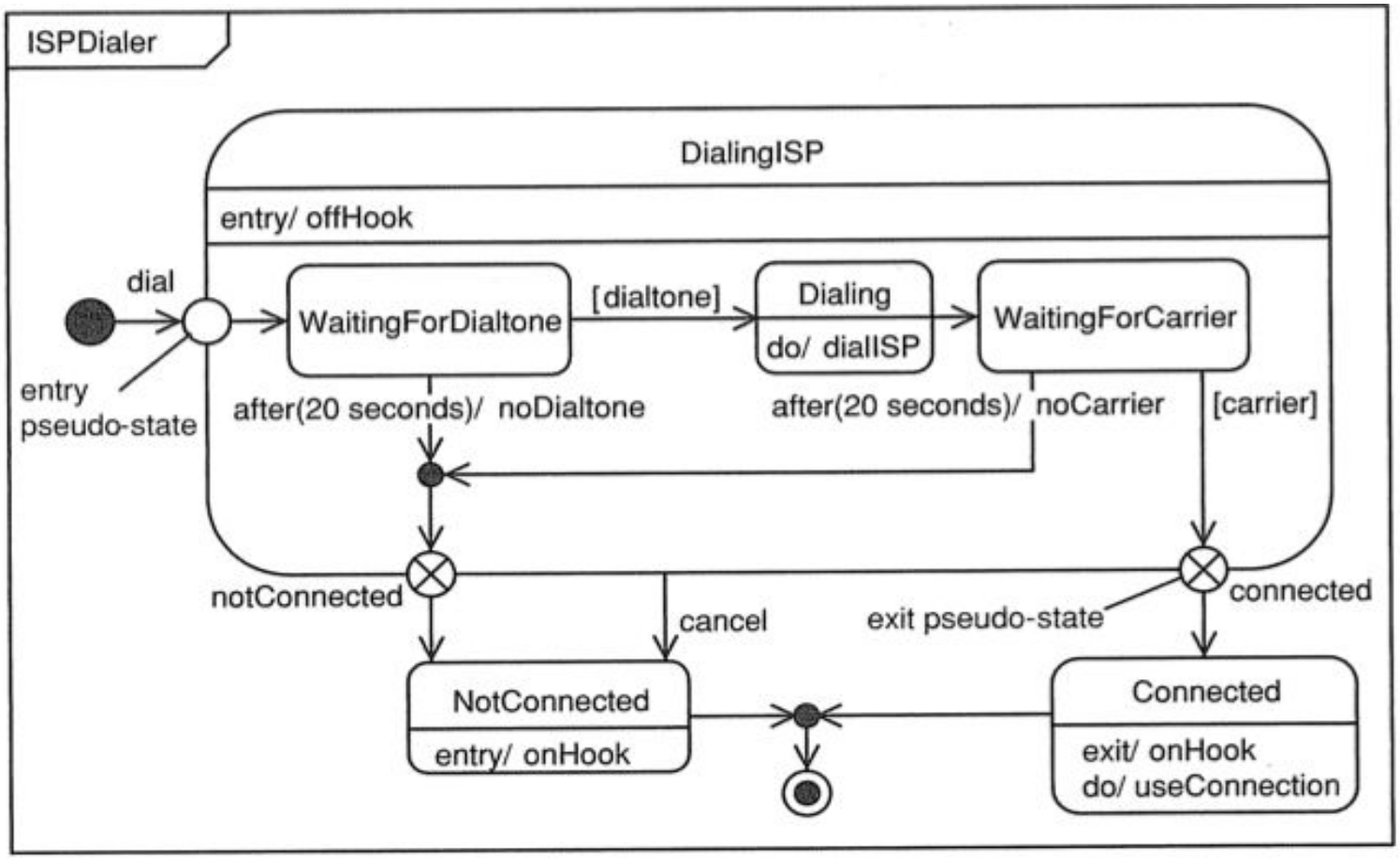
Pseudo-state fusion



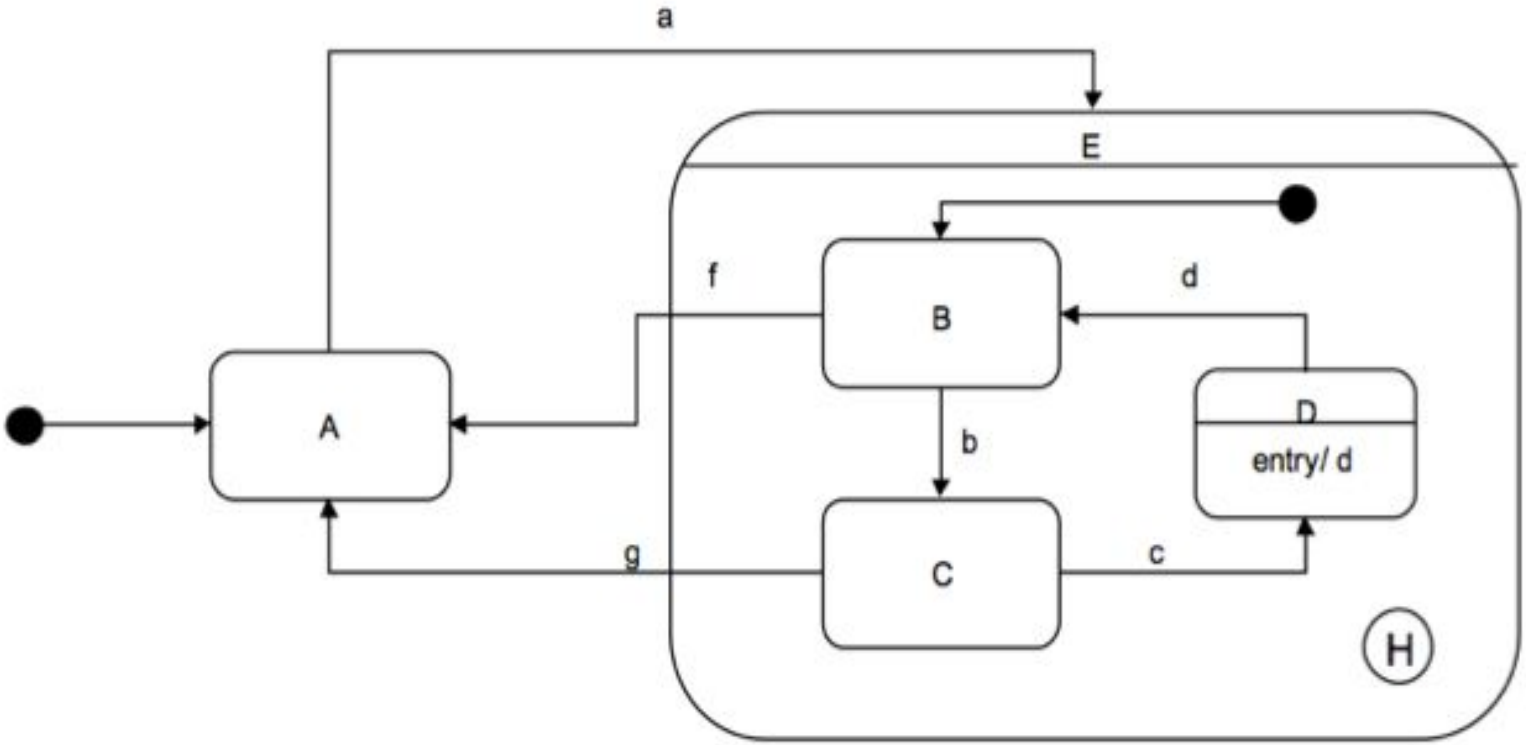
Pseudo termination state



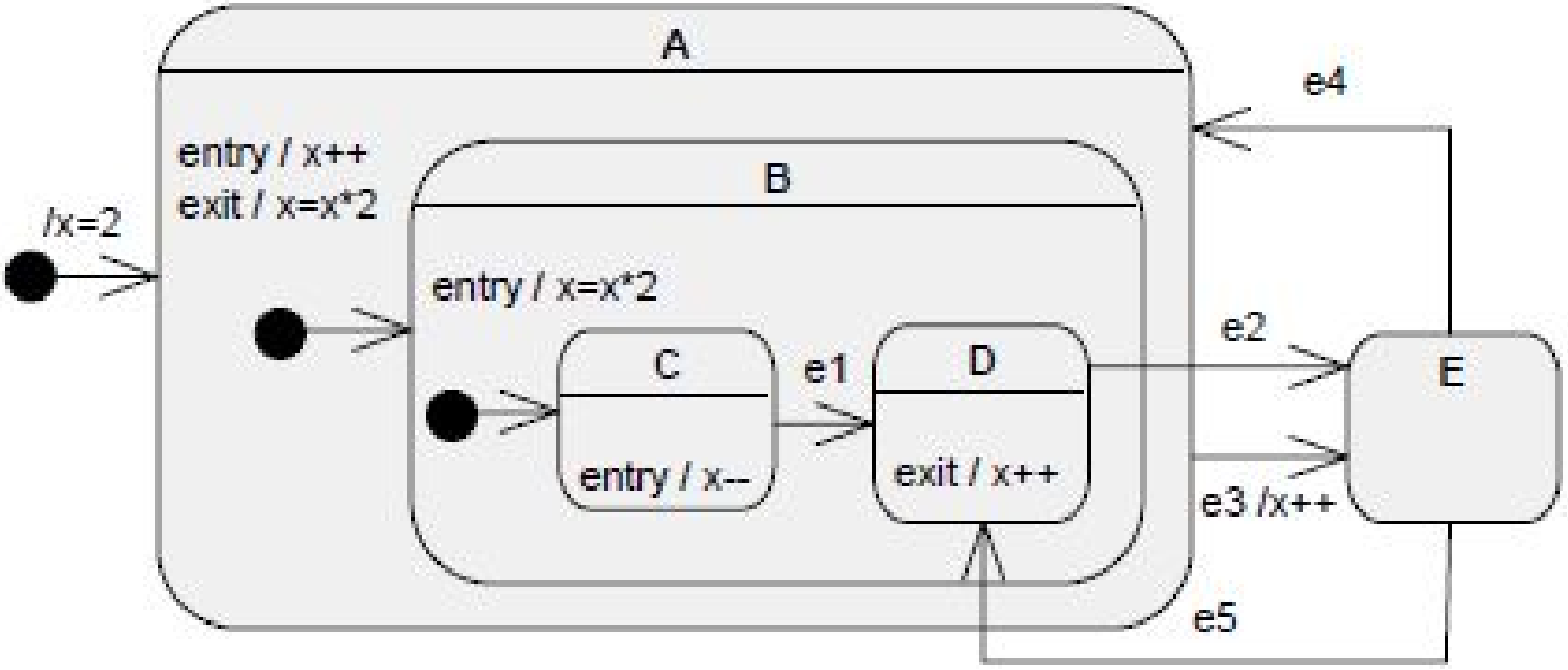
Exercise


ISPDialer
DialingISP
entry/ offHook
dial
entry
pseudo-state
WaitingForDialtone
[dialtone]
Dialing
do/ dialISP
WaitingForCarrier
after(20 seconds)/ noDialtone
after(20 seconds)/ noCarrier
[carrier]
notConnected
cancel
exit pseudo-state
connected
NotConnected
entry/ onHook
Connected
exit/ onHook
do/ useConnection

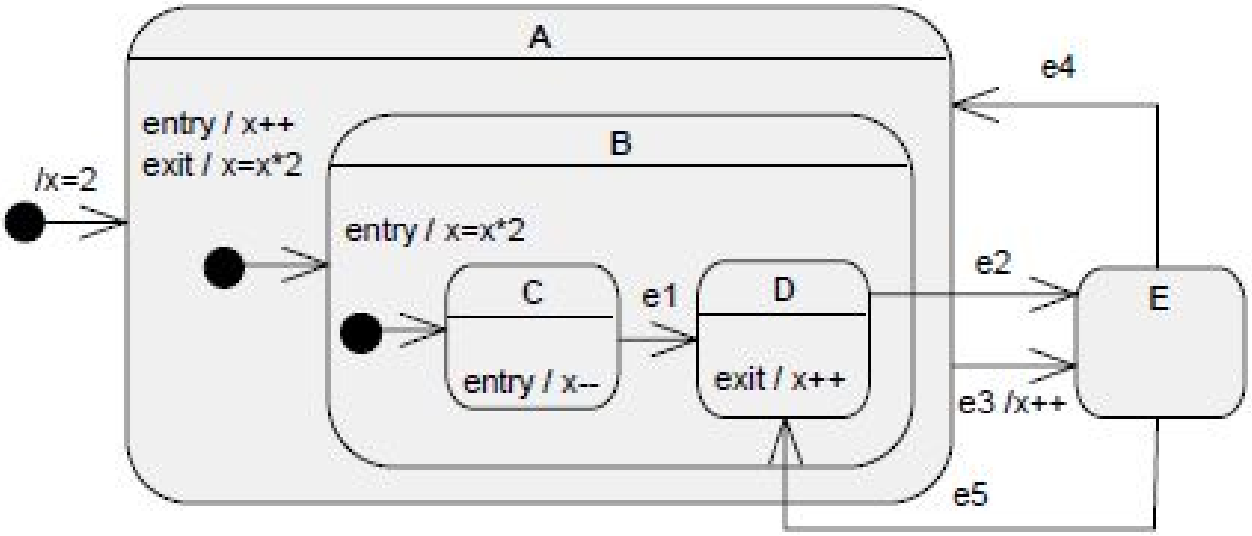
What state(s) is the object in after this sequence of events? -> a -> b -> c -> b -> g -> a



What is the value of x after the occurrence of the event chain: **e1 e2 e4 e2 e1** ?



What is the value of **x** after the occurrence of the event chain: **e1 e2 e4 e2 e1** ?



event	state	comment	x
		start	2
	A	entry of A	3
	A/B	entry of B	6
	A/B/C	entry of C	5
e1	A/B/D		
e2		exit of D	6
		exit of A	12
	E		
e4	A	entry of A	13
	A/B	entry of B	26
	A/B/C	entry of C	**25**
e2			
e1	A/B/D		