Module 8

State Machines Diagrams

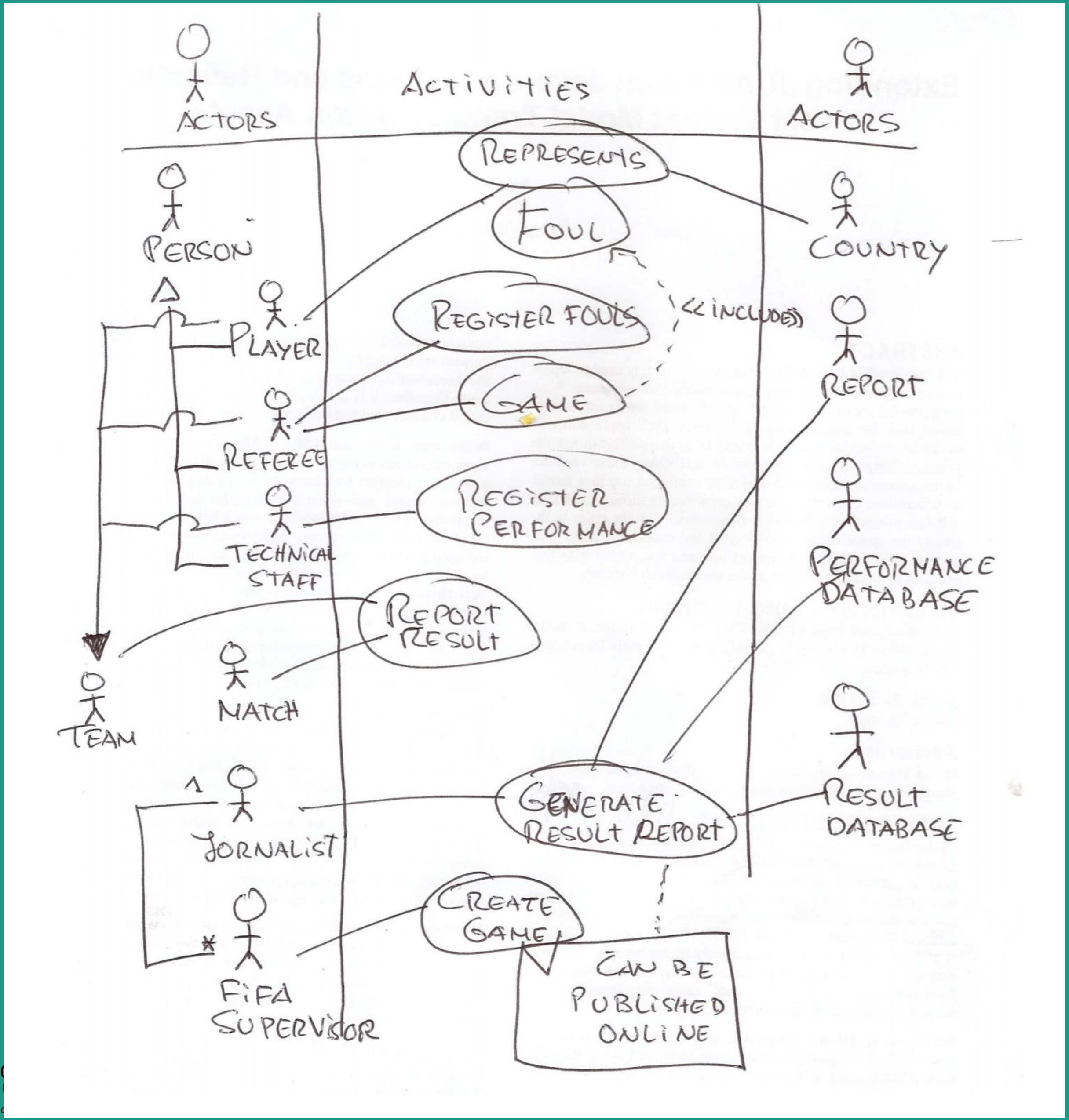
Vasco Amaral
vma@fct.unl.pt

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

Quiz time

Identify the problems of the following Use Case Diagram

FACULDADE DE CIÊNCIAS E TECNOLOGIA **UNIVERSIDADE NOVA** DE LISBOA
Departamento de Informática


ACTORS
ACTIVITIES
ACTORS
REPRESENTS
FOUL
COUNTRY
PERSON
PLAYER
REGISTER FOULS
<<INCLUDES>>
REPORT
GAME
REFEREE
REGISTER PERFORMANCE
TECHNICAL STAFF
PERFORMANCE DATABASE
REPORT RESULT
TEAM
MATCH
1
JORNALIST
GENERATE RESULT REPORT
RESULT DATABASE
CREATE GAME
*
FIFA SUPERVISOR
CAN BE PUBLISHED ONLINE

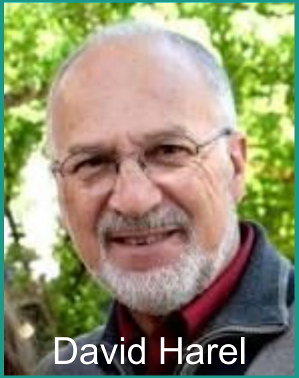
The activity final node of an UML2 activity diagram (which are true and which are false)?

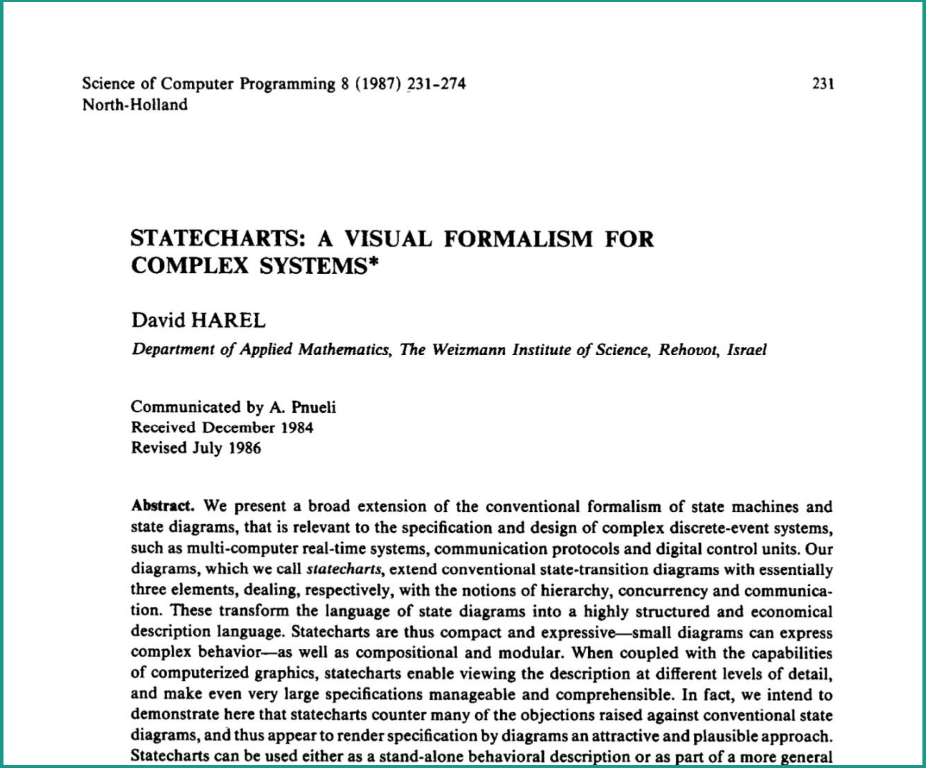
1. ... makes, once reached, the execution of other activities within the graph impossible.
2. ... can only be modeled once in each model.
3. ... terminates all activities within the graph.

Object nodes in a UML2 activity diagram ...

1. ... can be the input to or the output of an action.
2. ... may be constrained to be in a specific state.
3. ... contain data token.
4. ... have no outgoing edges.

Statecharts (aka state machines, state diagrams, ...)




Science of Computer Programming 8 (1987) 231–274 231
North-Holland

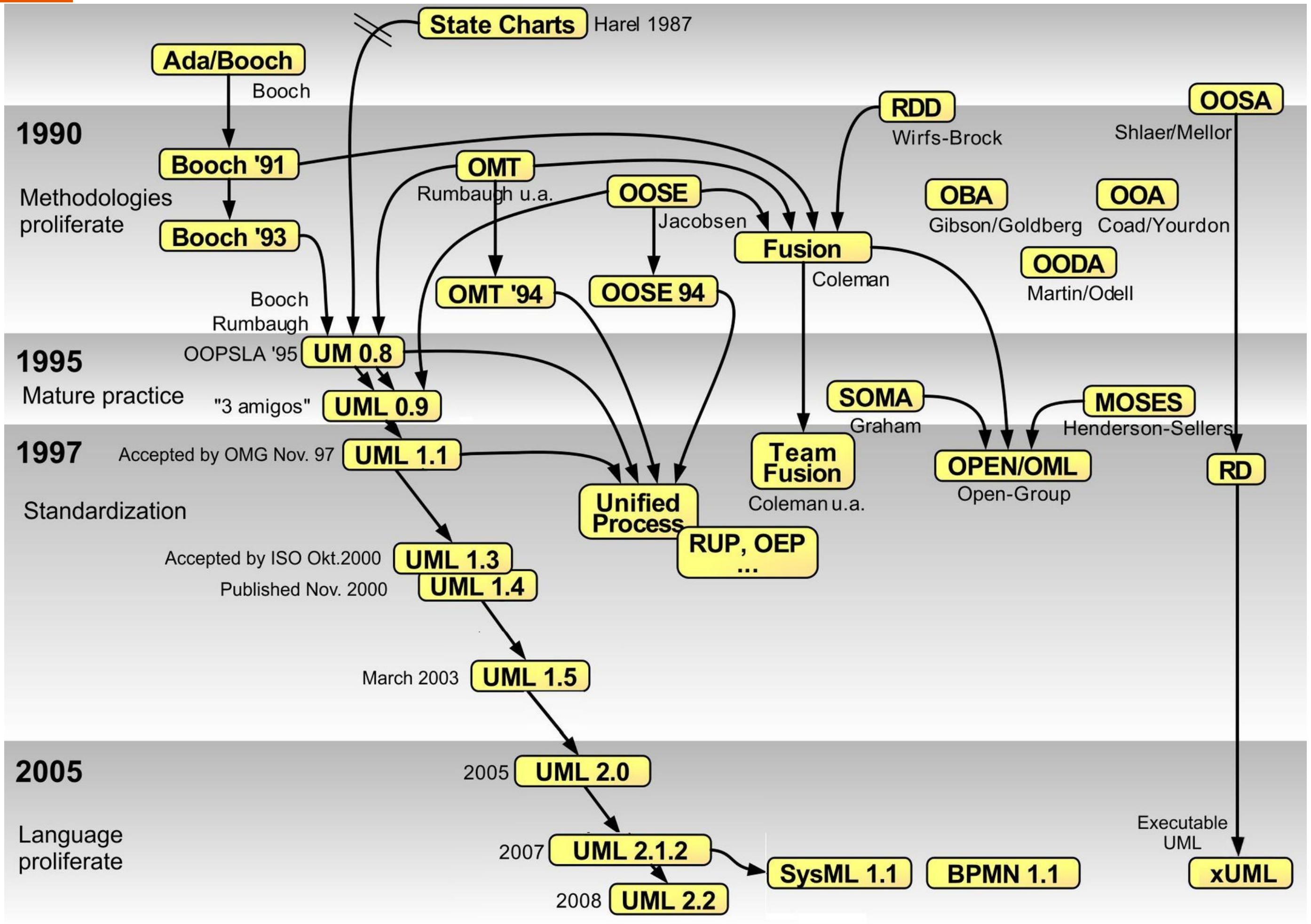
STATECHARTS: A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

David HAREL
Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel

Communicated by A. Pnueli
Received December 1984
Revised July 1986

Abstract. We present a broad extension of the conventional formalism of state machines and state diagrams, that is relevant to the specification and design of complex discrete-event systems, such as multi-computer real-time systems, communication protocols and digital control units. Our diagrams, which we call *statecharts*, extend conventional state-transition diagrams with essentially three elements, dealing, respectively, with the notions of hierarchy, concurrency and communication. These transform the language of state diagrams into a highly structured and economical description language. Statecharts are thus compact and expressive—small diagrams can express complex behavior—as well as compositional and modular. When coupled with the capabilities of computerized graphics, statecharts enable viewing the description at different levels of detail, and make even very large specifications manageable and comprehensible. In fact, we intend to demonstrate here that statecharts counter many of the objections raised against conventional state diagrams, and thus appear to render specification by diagrams an attractive and plausible approach. Statecharts can be used either as a stand-alone behavioral description or as part of a more general

State Charts and their role in OO design



Guido Zockoll, Axel Scheithauer & Marcel Douwe Dekker, “History of Object-Oriented Modeling Languages”, 2009

A state diagram models the dynamic behavior of a reactive object

- Goal: to model behavior (dynamic)
- Use for **modelling objects** (or systems) **with complex behavior** (that would otherwise be hard to understand)
- Use when the **control** is **deeply influenced by external events**
- Don’t use when **several objects are involved**
  - Interaction diagrams are more adequate, in that scenario

Key elements in state machine diagrams

- **State**
  - A condition or situation during the life cycle of an object during which it satisfies some condition, performs some activity, or waits for some event
- **Event**
  - The specification of a noteworthy occurrence that has location in time and space
- **Transition**
  - The movement from a state to another in response to an event

A reactive object provides context for a state machine

- Responds to external events
- May generate and respond to internal events
- Has a life cycle modelled as a progression of **states**, **transitions**, and **events**
- May have current behavior that depends on past behavior

State machines model the behavior of classifiers, such as

- Classes
- Use cases
- Subsystems
- Entire systems

Behavioral state machines specify behavior of the context classifier

- States, transitions and events define the behavior of a context classifier
- Behavioral state machines can only model classifiers with behavior
    - Interfaces and ports have no behavior - they just define a usage protocol

Protocol state machines specify the protocol of the context classifier

- Conditions under which operations may be called on the classifier and its instances
- The results of the operation calls
- The ordering of the operation calls

Behavioral vs protocol state machines

Behavioral state machines

- Specify the implementation of a behavior
- Can be used to specify the behavior of classifiers with implementation
- States can specify one or more actions that execute when the state is entered

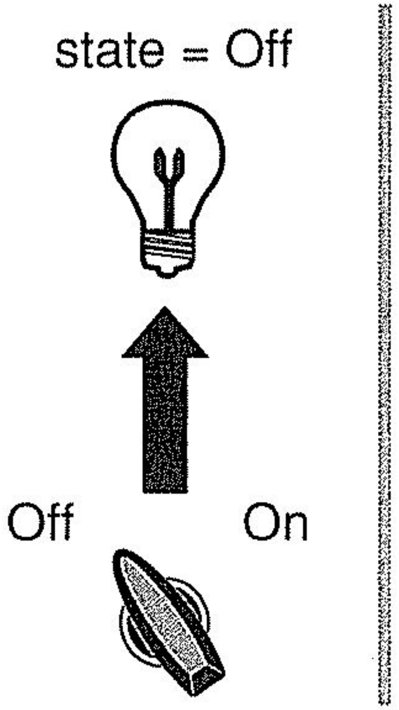
Protocol state machines

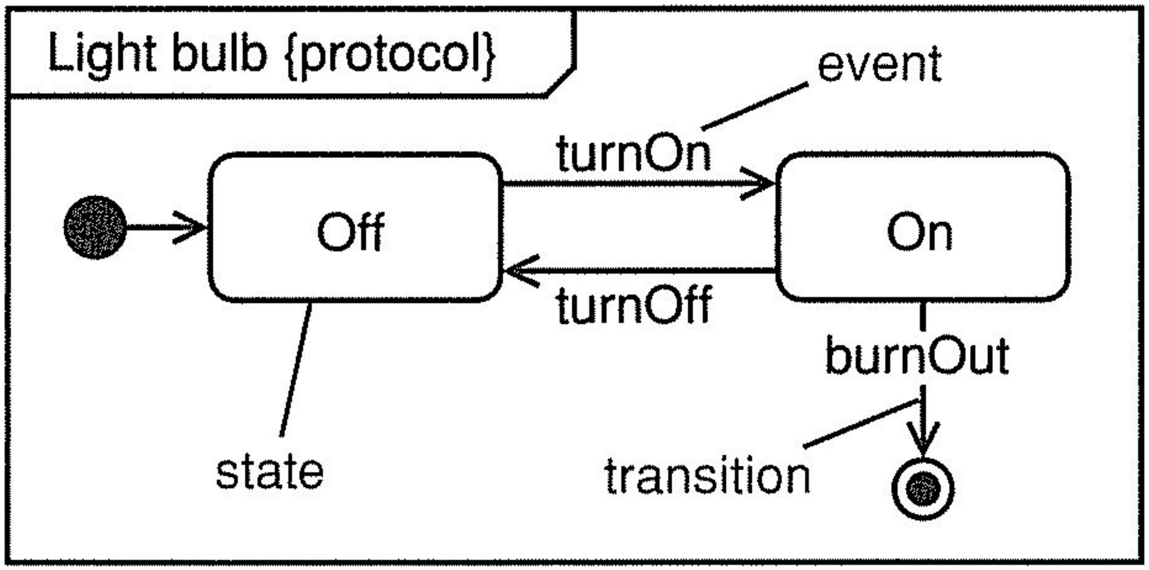
- Do not specify the implementation of this behavior
- Can be used to specify the protocol for all classifier, including those without implementation
- States can't specify actions
- Denoted by the keyword `{protocol}` after the state machine name

State machines are defined during design

- State machines are usually defined during the design workflow
- Create a state machine to understand a complex life cycle or behavior
  - If it is too simple, it may not pay off to do so
- Particularly useful in real-time, embedded systems, where objects may have complex behavior and life cycles

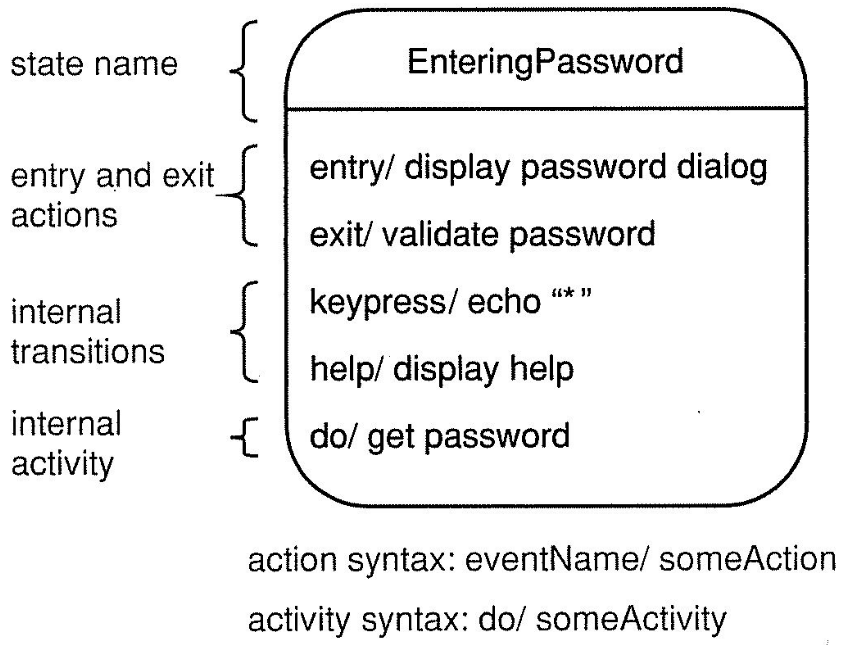
State machine diagrams





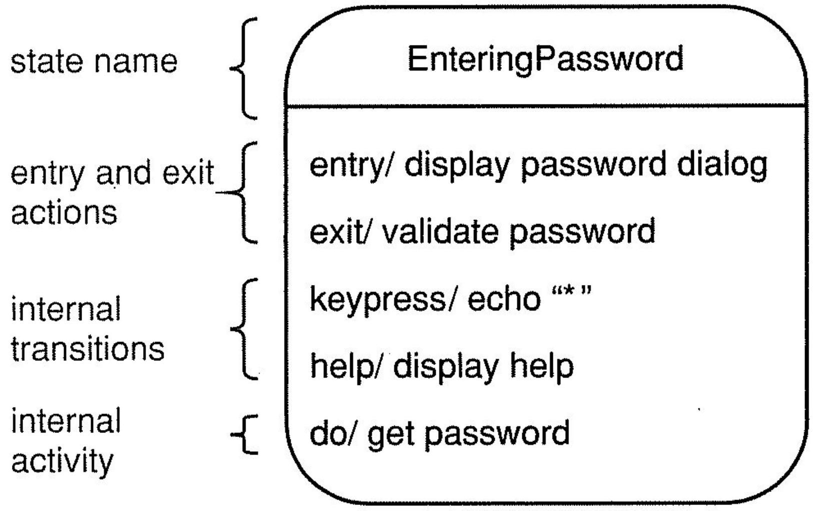
State is a semantically significant condition of an object

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



Actions are instantaneous and uninterruptible

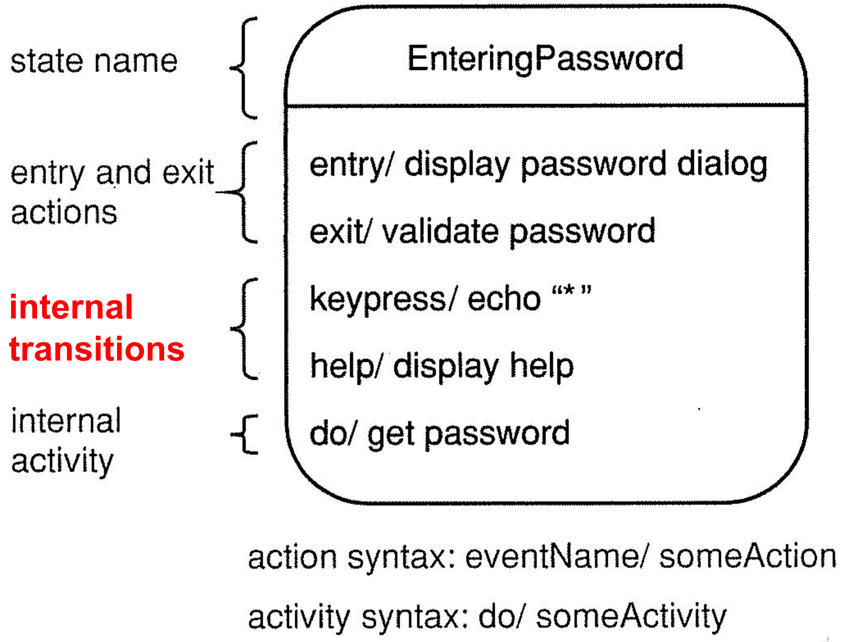
- Each **action** in a state is associated with an internal transition triggered by an **event**
    - In this example, two special actions, entry and exit are modelled
        - The entry event occurs immediately after entering a state and causes the event action to execute
        - The exit event is the last thing that happens on exiting the state and causes the associated event action to execute



action syntax: **eventName**/ **someAction**
activty syntax: do/ someActivity

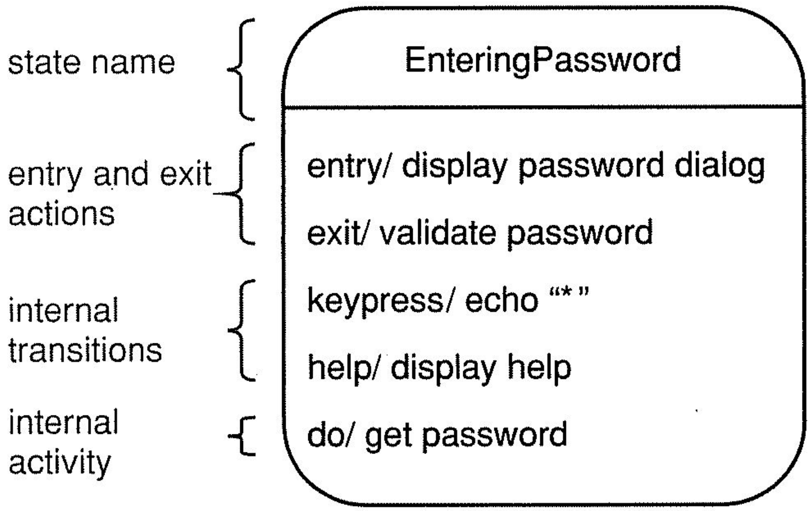
Internal transitions capture that some transition within the same state has happened

- Each **internal transition** captures an event which is noteworthy but does not cause a state transition
  - In this example, it is relevant to know that a key was pressed, or that help is being displayed, but none of those cause leaving the state EnteringPassword



Actions are instantaneous and uninterruptible

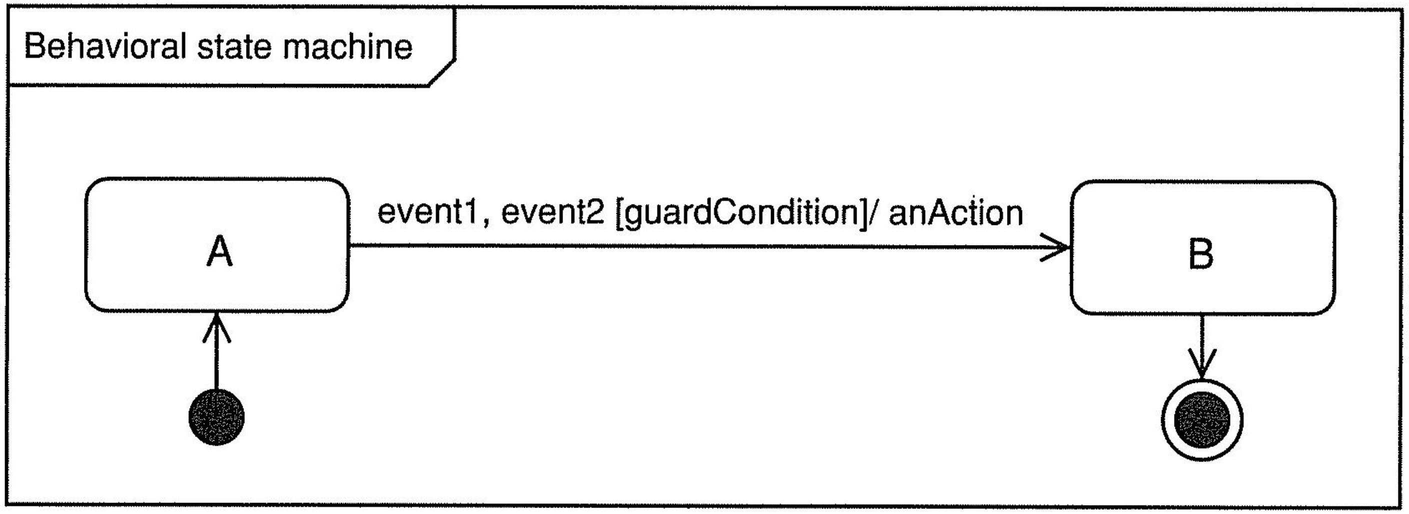
- Each **activity** takes a finite time to execute and is interruptible by an event
- The keyword **do** indicates an activity
- Unlike actions, activities can be interrupted before they finish processing



action syntax: eventName/ someAction
activty syntax: **do**/ **someActivity**

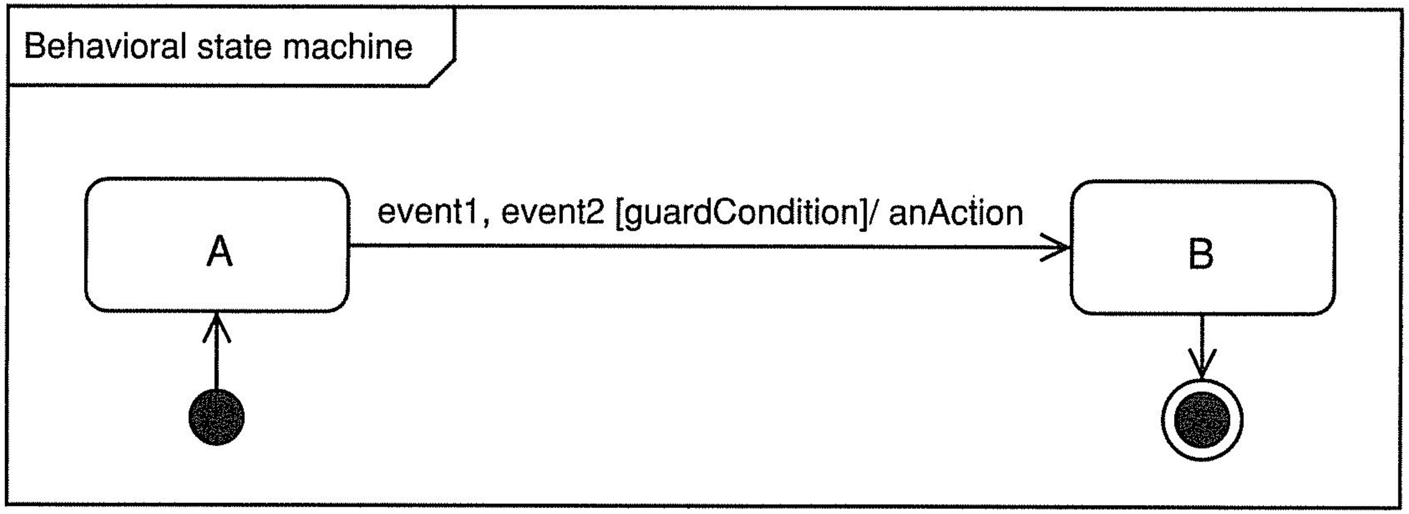
Transitions show movement between states

- Relationship between two states, indicating that an object in the **source state** will execute certain **actions** and transition to a **target state**, when a given set of **events** occur and certain **conditions** are met
- Example:
  *On (**event1** or **event2**), if (**guardCondition**) then perform **anAction** and enter state **B**.*



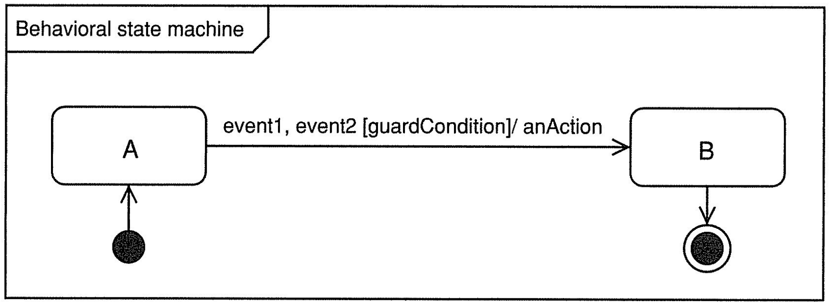
A transition occurs when

- An object is in its **source state**
- An **event** occurs
- If a guard **condition** is true
  - An **action** is executed
  - The object changes to the **target state**



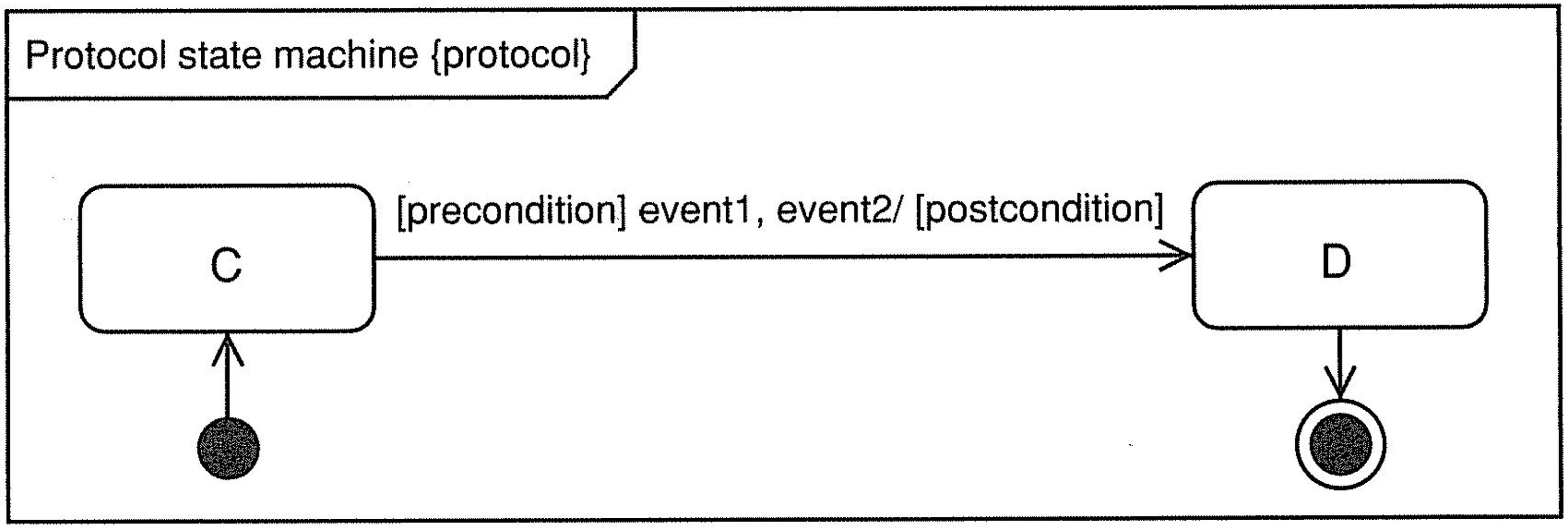
Transitions have three optional elements

- Zero or more events
  - These specify external or internal occurrences that can trigger the transition
- Zero or one guard condition
  - This is a boolean expression that must evaluate to true before the transition can occur
- Zero or more actions
  - This is a piece of work associated with the transition and occurs when the transition fires



Transitions in state protocols have a slightly different syntax

- There is no action, as we are specifying a protocol rather than an implementation
- The guard condition is replaced by pre and post conditions
  - Notice how the **precondition** is placed **before the slash** while the **postcondition** is placed **after the slash**

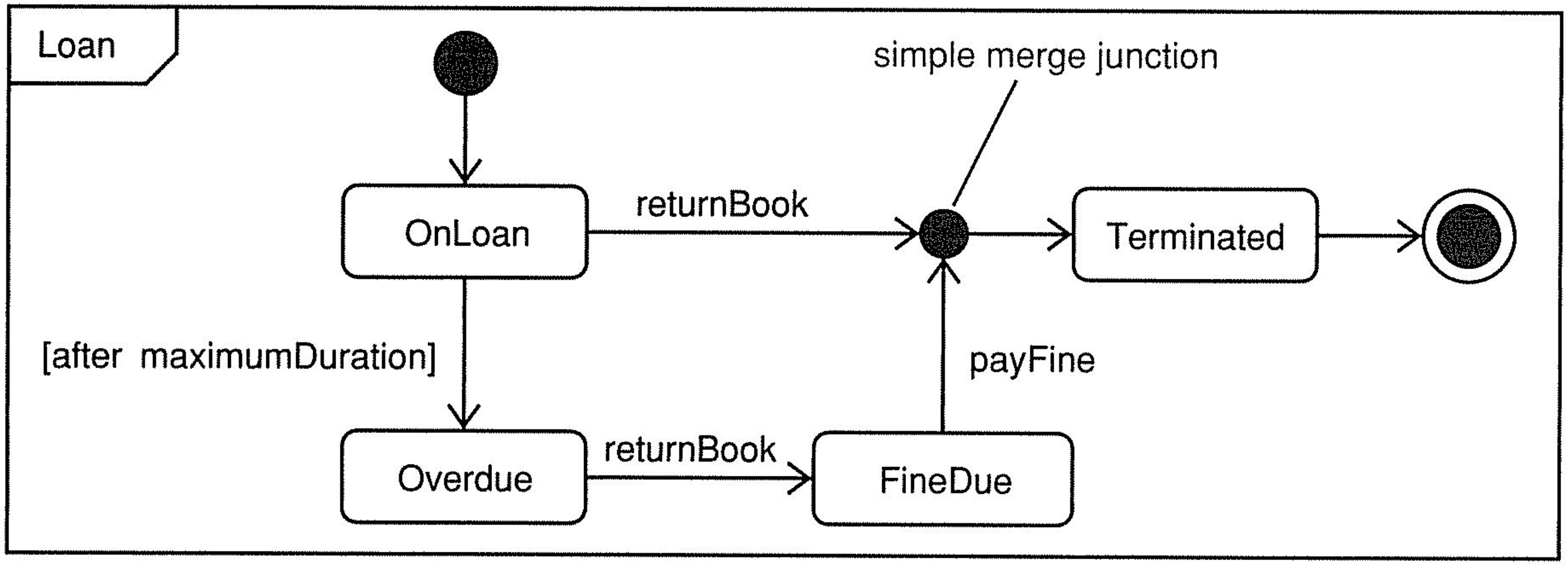

Protocol state machine {protocol}
[precondition] event1, event2/ [postcondition]
C
D

What if the transition has no event?

- Both in behavioral and protocol state machines, if a **transition has no event** it is an **automatic transition**
- Automatic transitions
  - do not wait for an event
  - fire when their guard conditions or preconditions are true

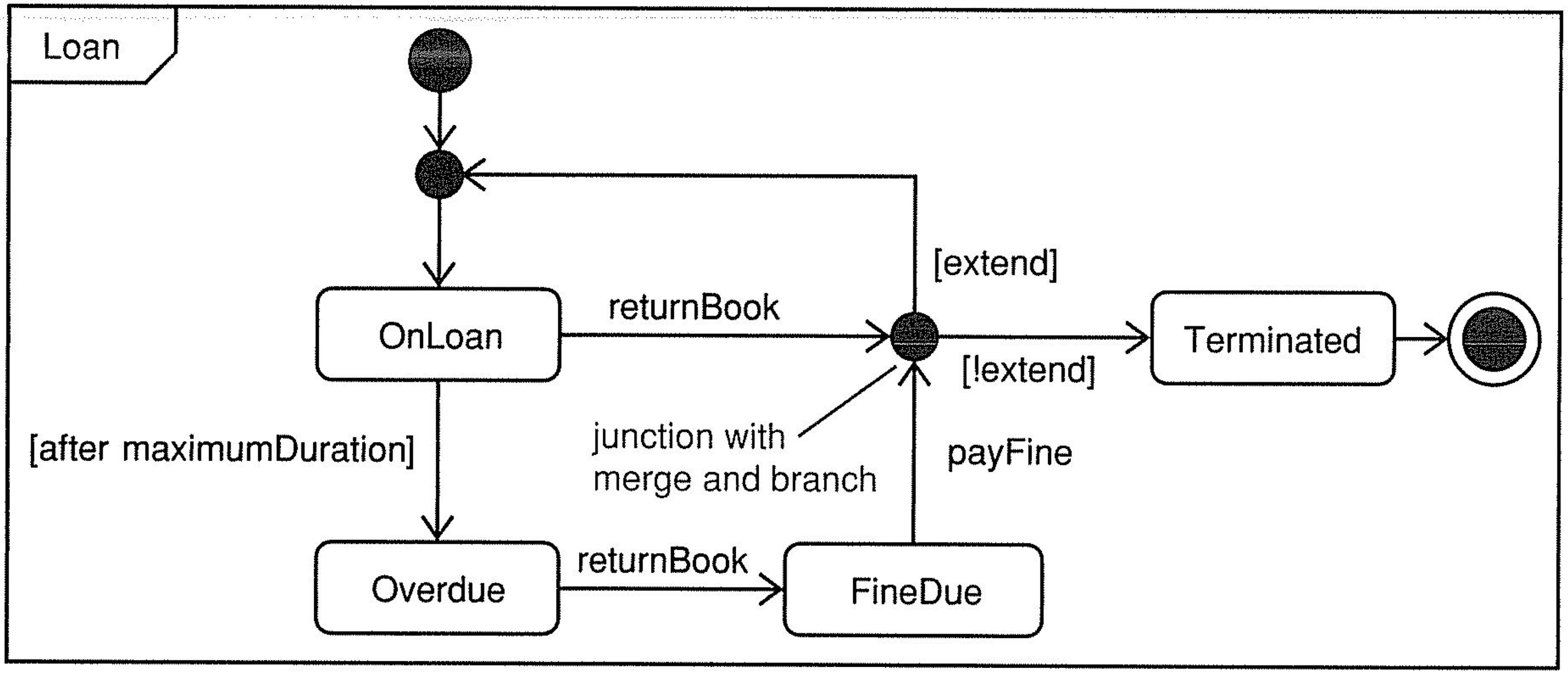
Junction pseudo-states join or branch transitions

- Junction pseudo-states represent points where transitions merge or branch
- They may have one or more input transitions and one or more output transitions

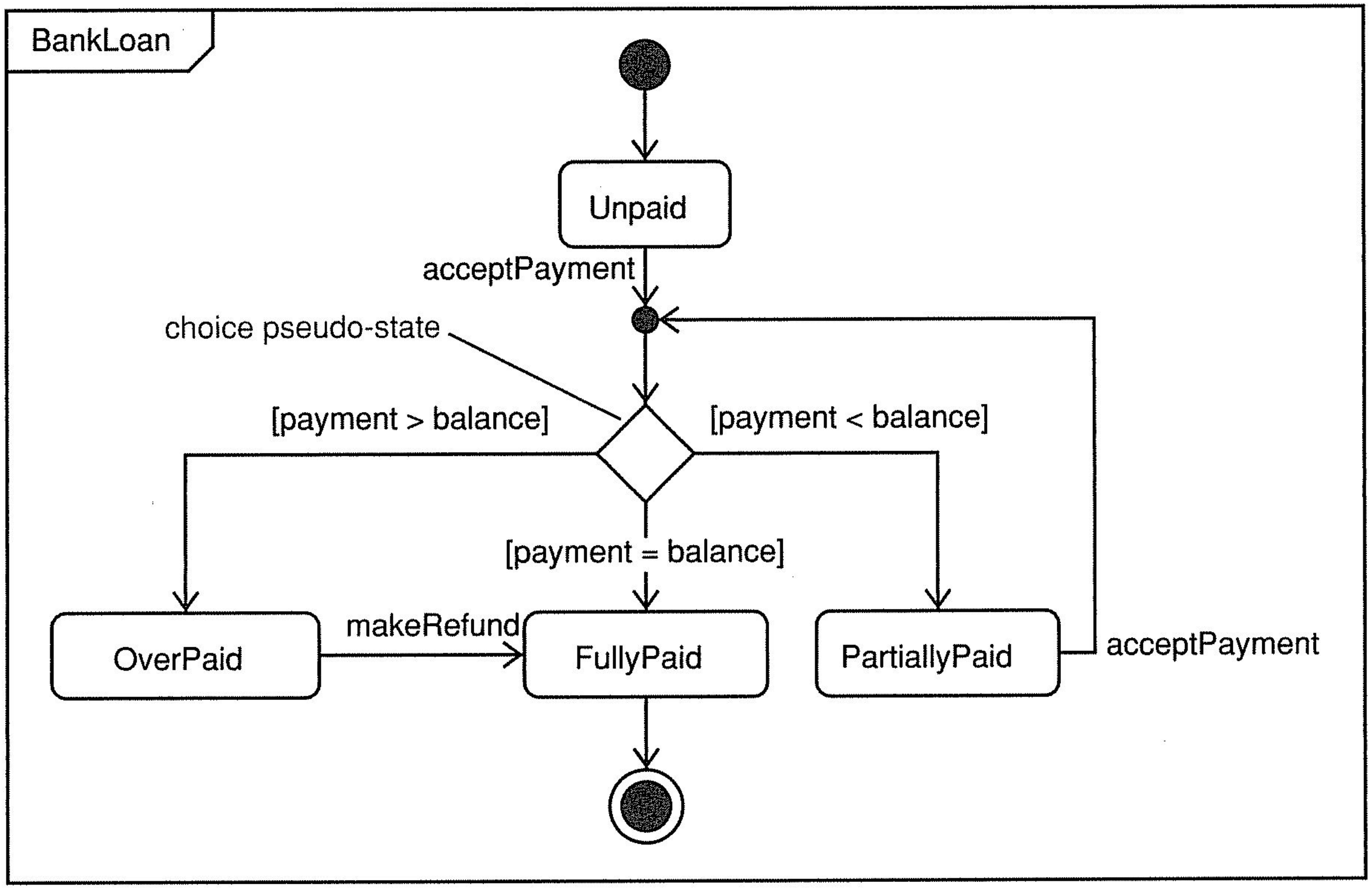


Protect junction pseudo-states with mutually exclusive guard conditions

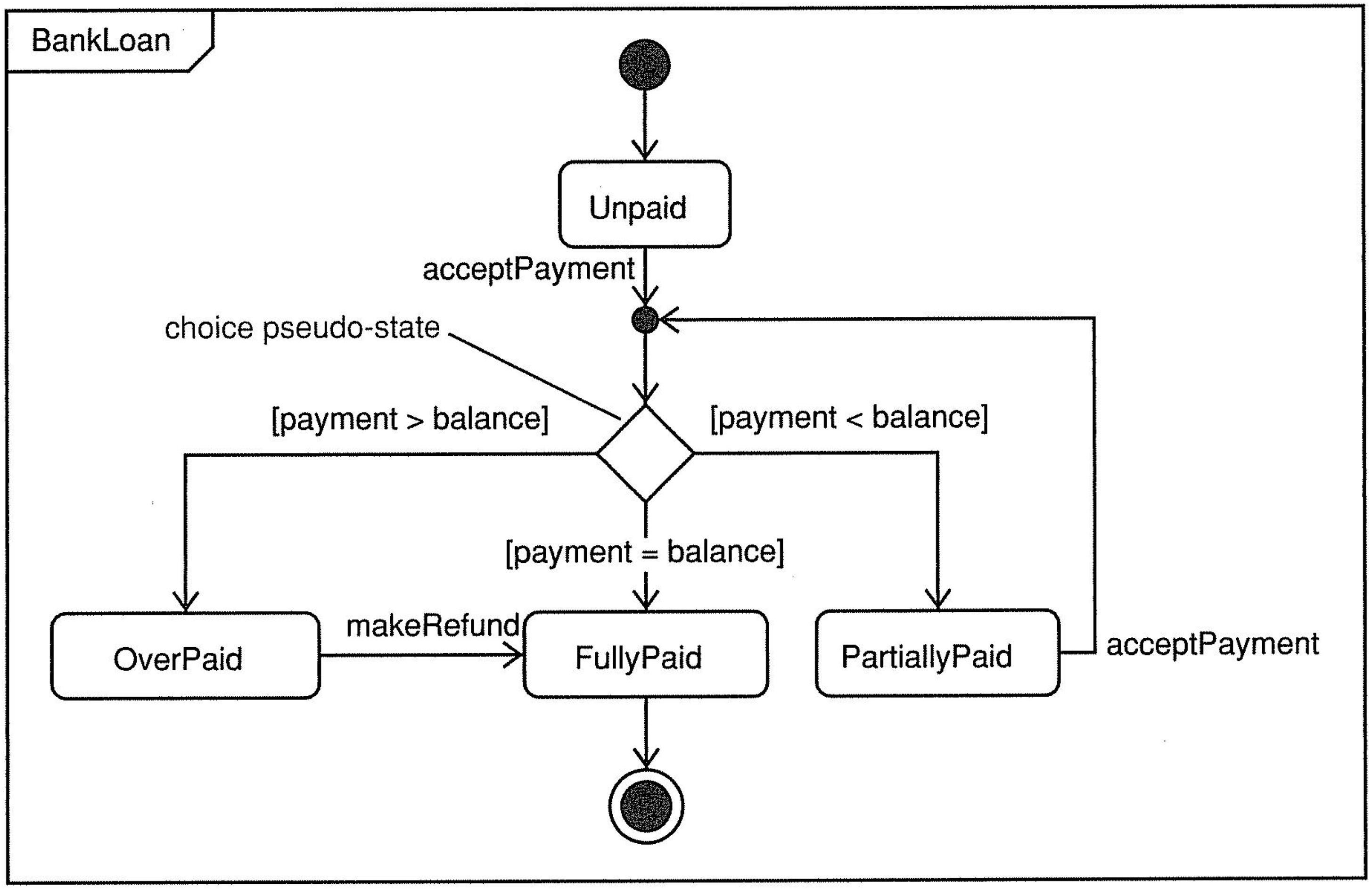
- Note how the junction pseudo-state has two inputs and two mutually exclusive guards, [extend] and [!extend]
- Note also the junction pseudo-state **before** the OnLoan state



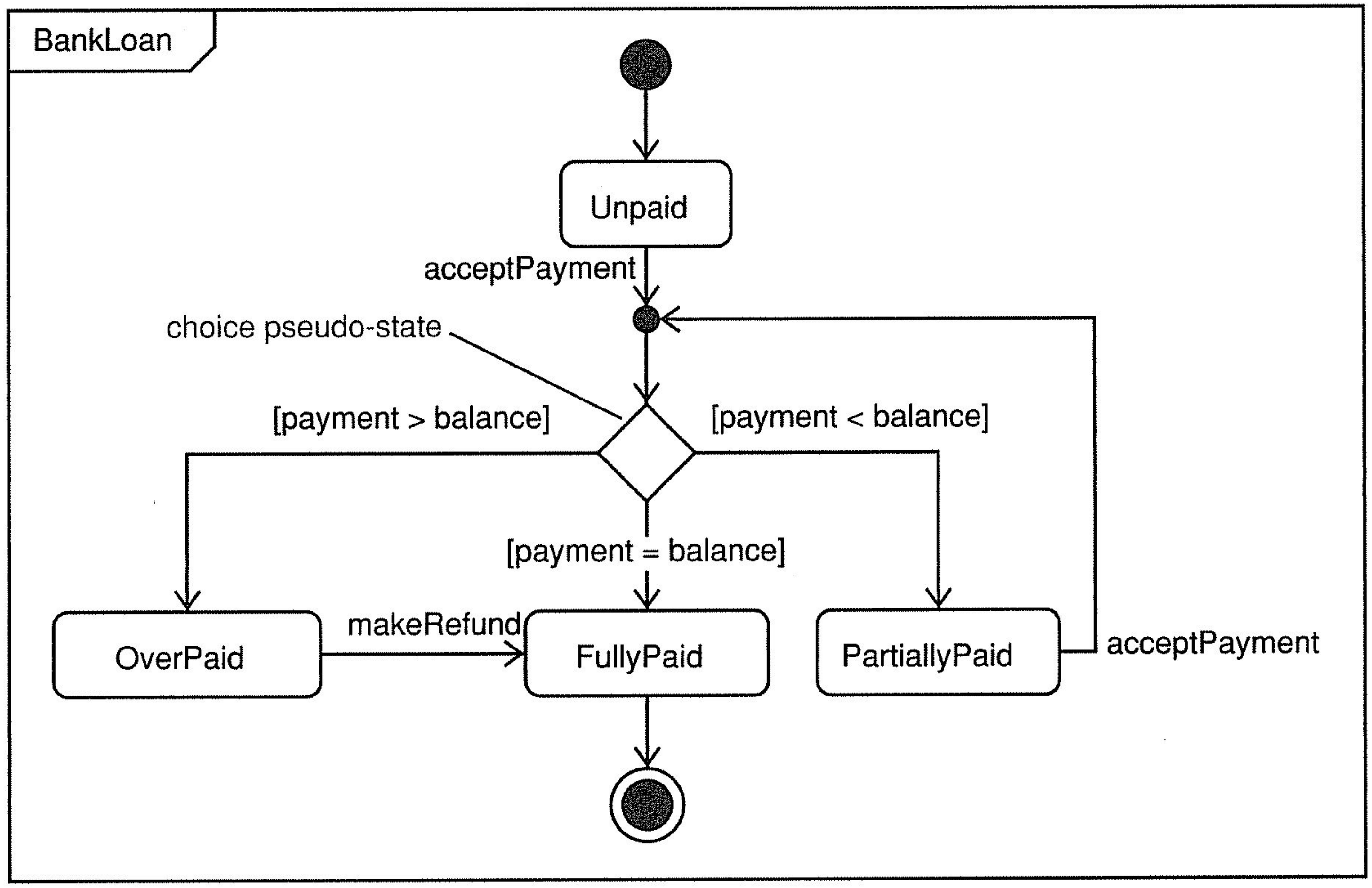
The choice pseudo-state directs the flow through the state machine according to guard conditions



Upon receiving the acceptPayment event, the BankLoan object transitions to one of three states

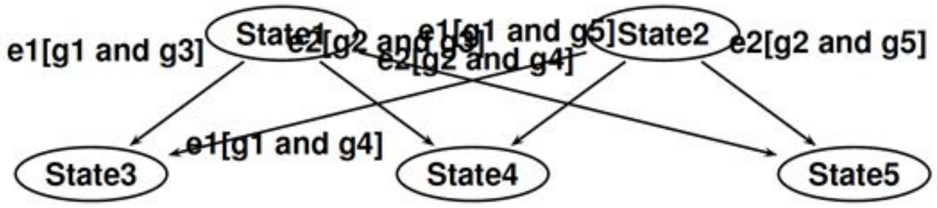


The choice pseudo-state guard conditions are mutually exclusive, so that only one transition is fired

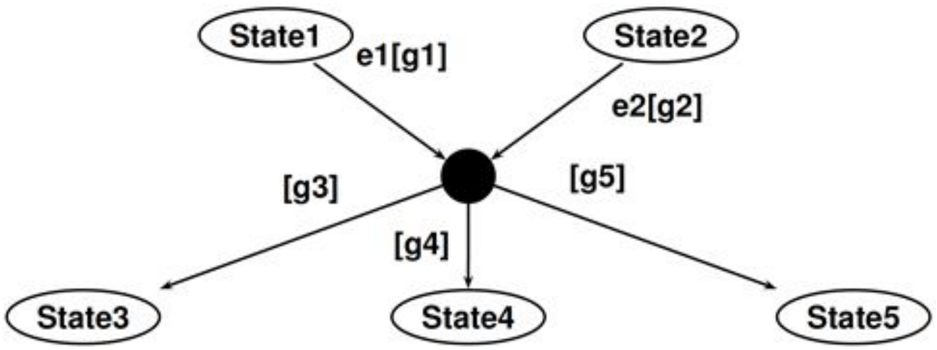


Junction points can also help tidying up you diagram, to increase its understandability

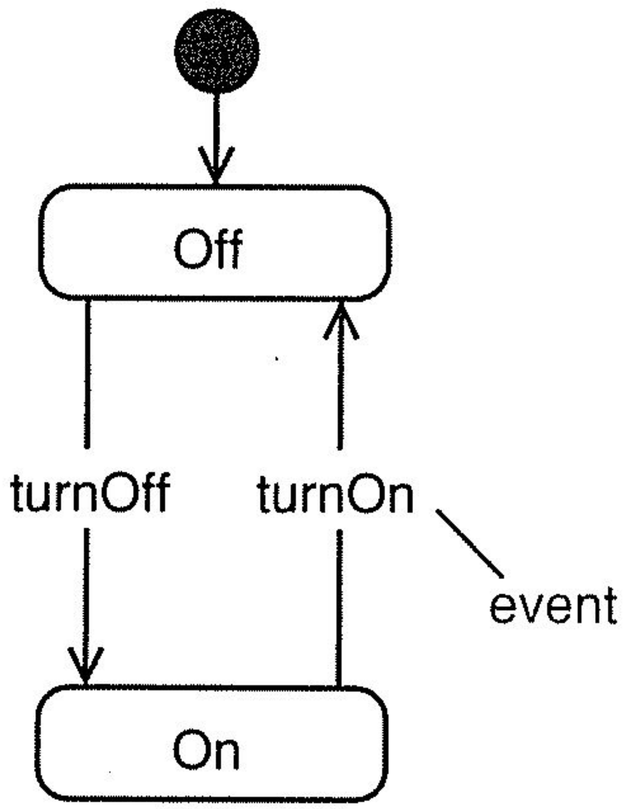
Without the junction



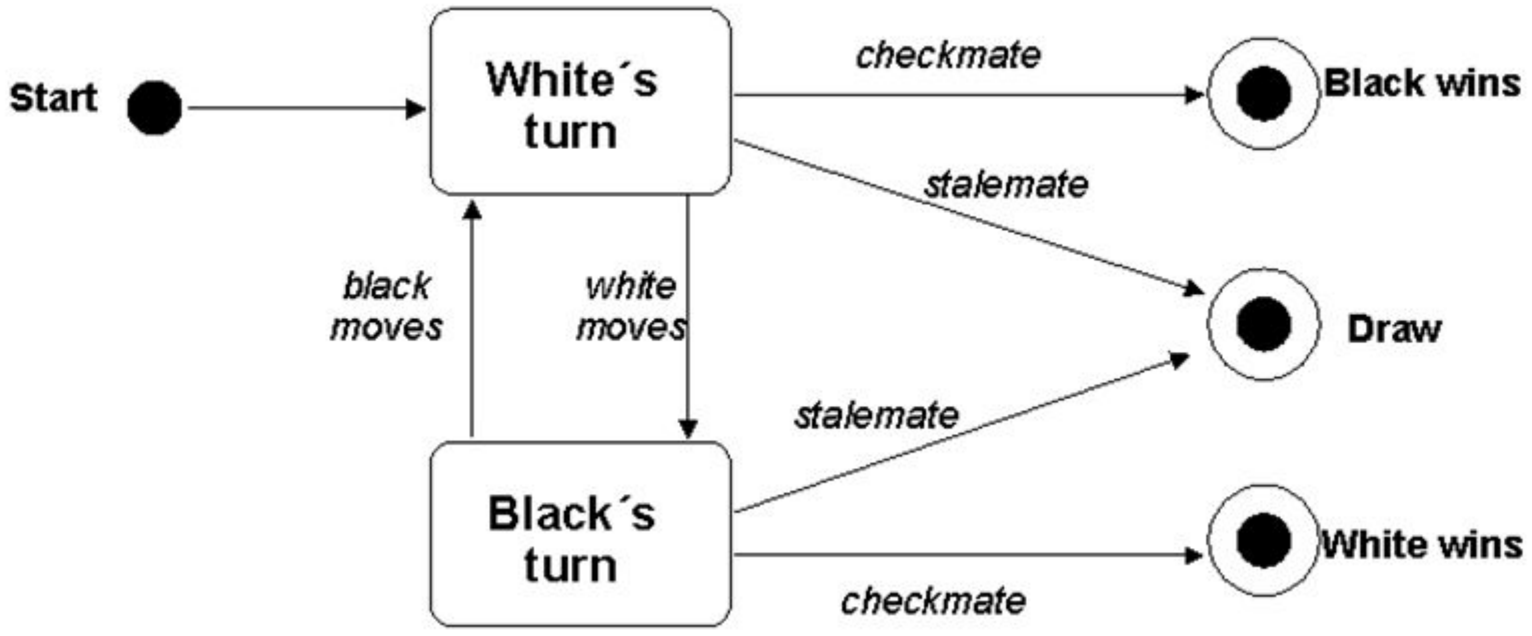
With the junction



Events trigger transitions



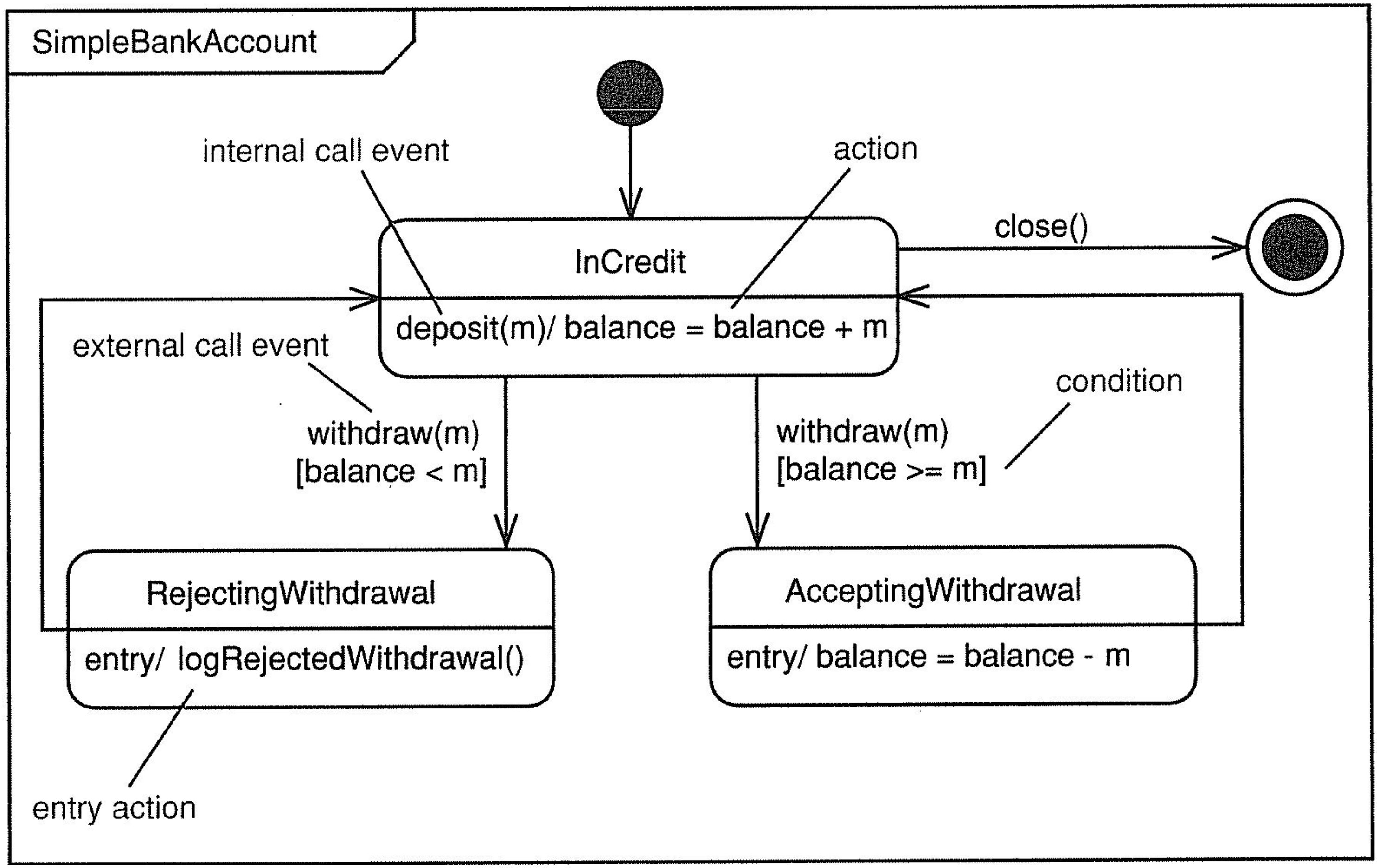
The same event (e.g. *checkmate*), when on different states, may lead to different outcomes



There are four kinds of events

- Call events
- Signal events
- Change events
- Time events

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

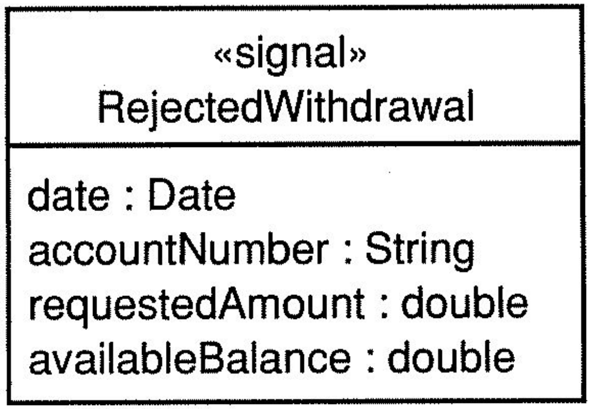


Call events

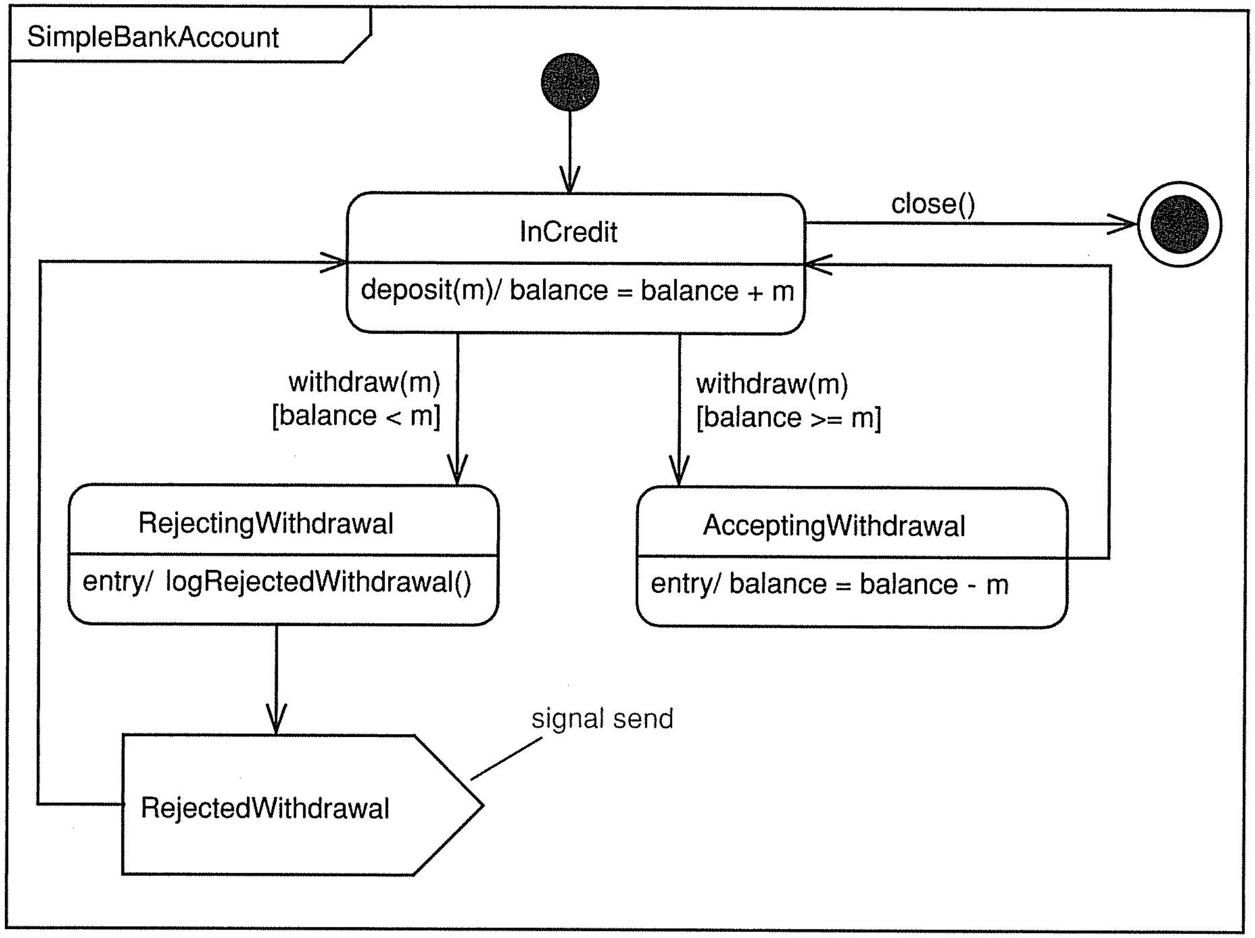
- A call event should have the same signature as an operation of the context class of the state machine
  - This applies to internal and external call operations
- Receiving a call operation triggers the operation to execute
- You can specify a sequence of actions, separated by ";", for a call event
  - these specify the semantics of the operation and can use attributes and operations of the context class and may have a return type - the call event has a value of the same return type

A signal event is a one-way asynchronous communication between objects

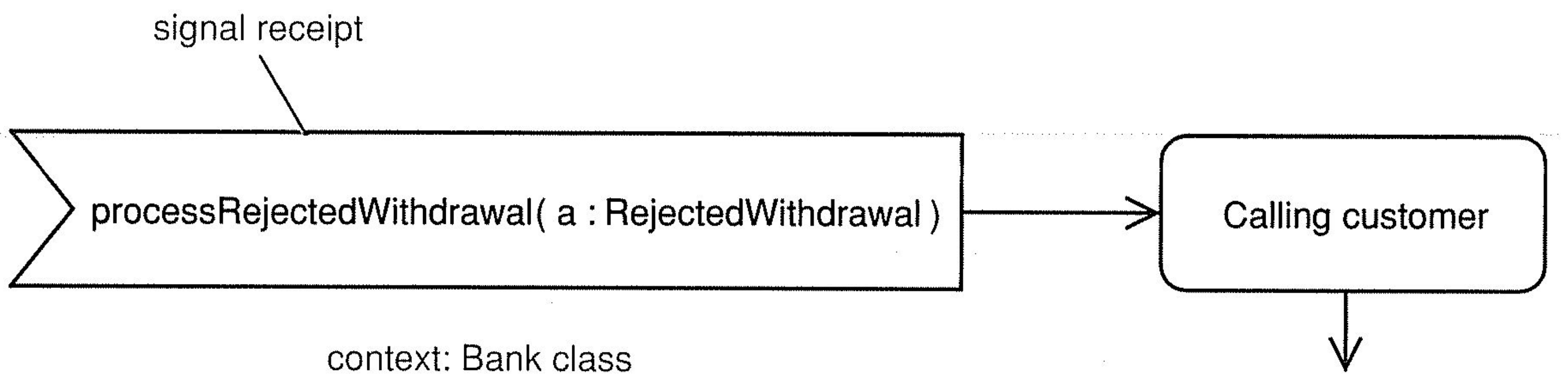
- A signal event is a package of information sent asynchronously between objects
  - It usually does not have operations, as its purpose is to carry information



SimpleBankAccount sends a signal when withdrawal is rejected



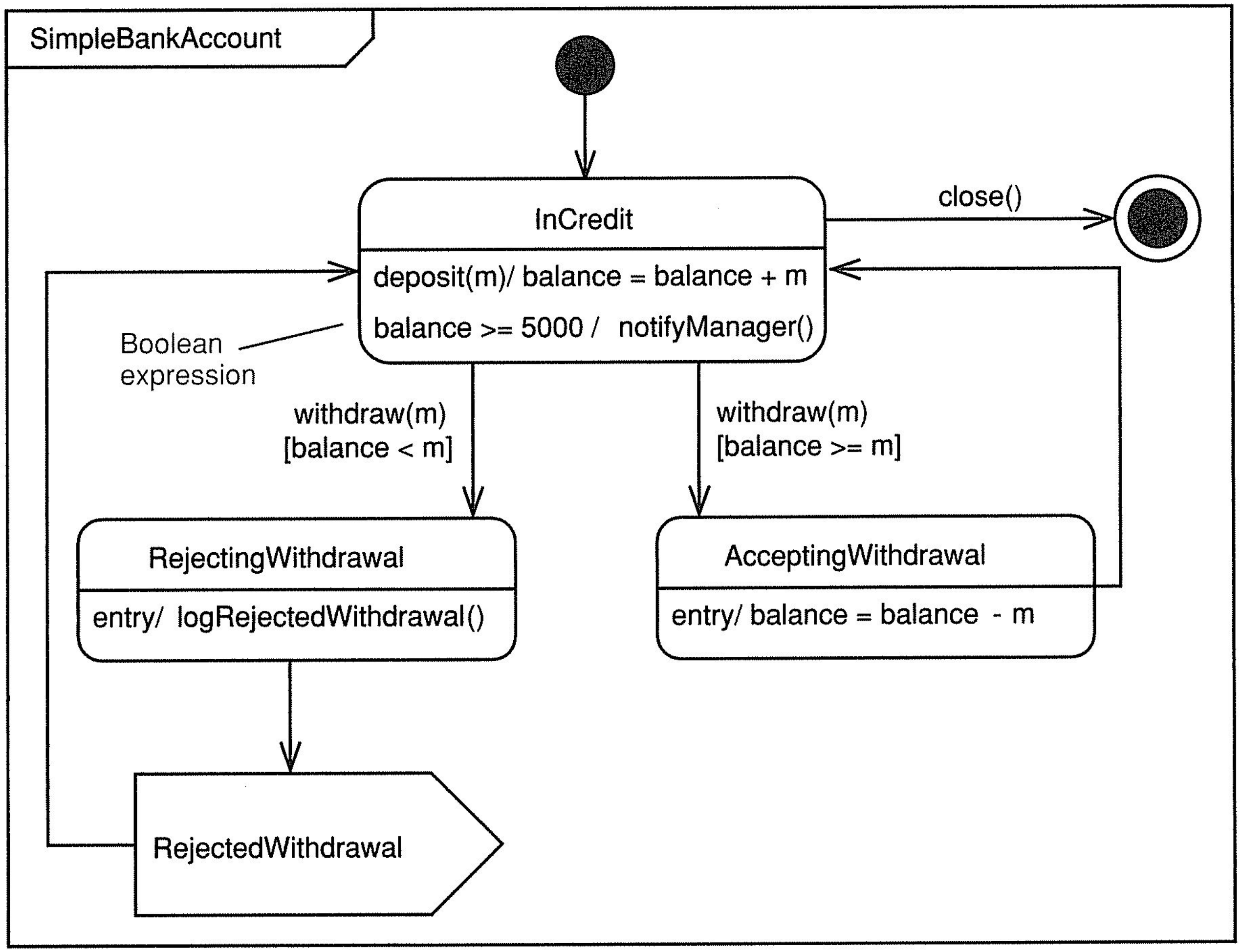
A signal receipt accepts the signal as a parameter and leads to a new state



Change events occur when a Boolean expression changes value from false to true

A change event implies **continuously testing** the Boolean condition **while in state**.

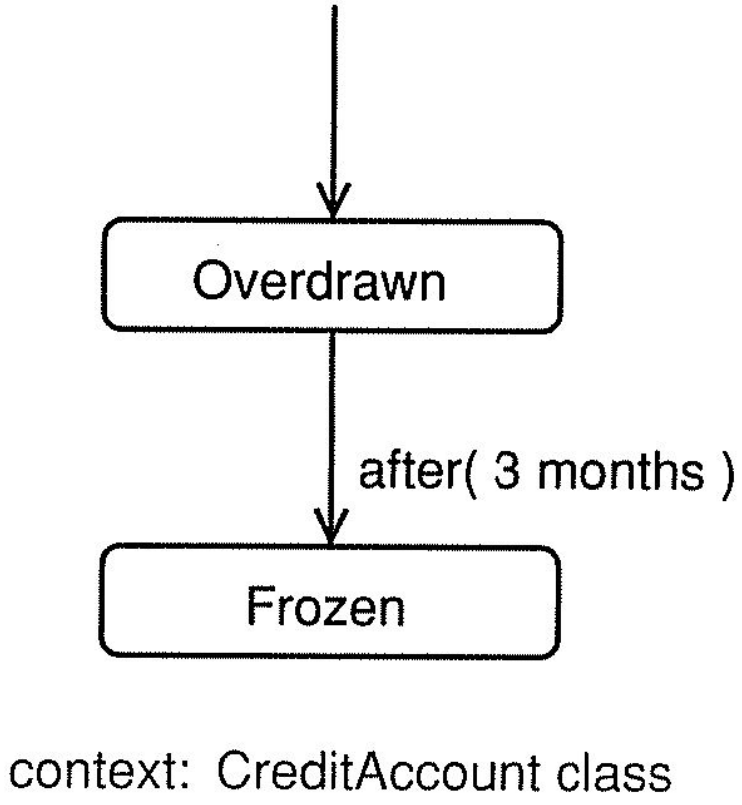
Change events are **positive-edge triggered** - they are **triggered whenever the condition changes from false to true**.


SimpleBankAccount
InCredit
deposit(m)/ balance = balance + m
balance >= 5000 / notifyManager()
close()
Boolean expression
withdraw(m) [balance < m]
withdraw(m) [balance >= m]
RejectingWithdrawal
entry/ logRejectedWithdrawal()
AcceptingWithdrawal
entry/ balance = balance - m
RejectedWithdrawal

Time events occur in response to time

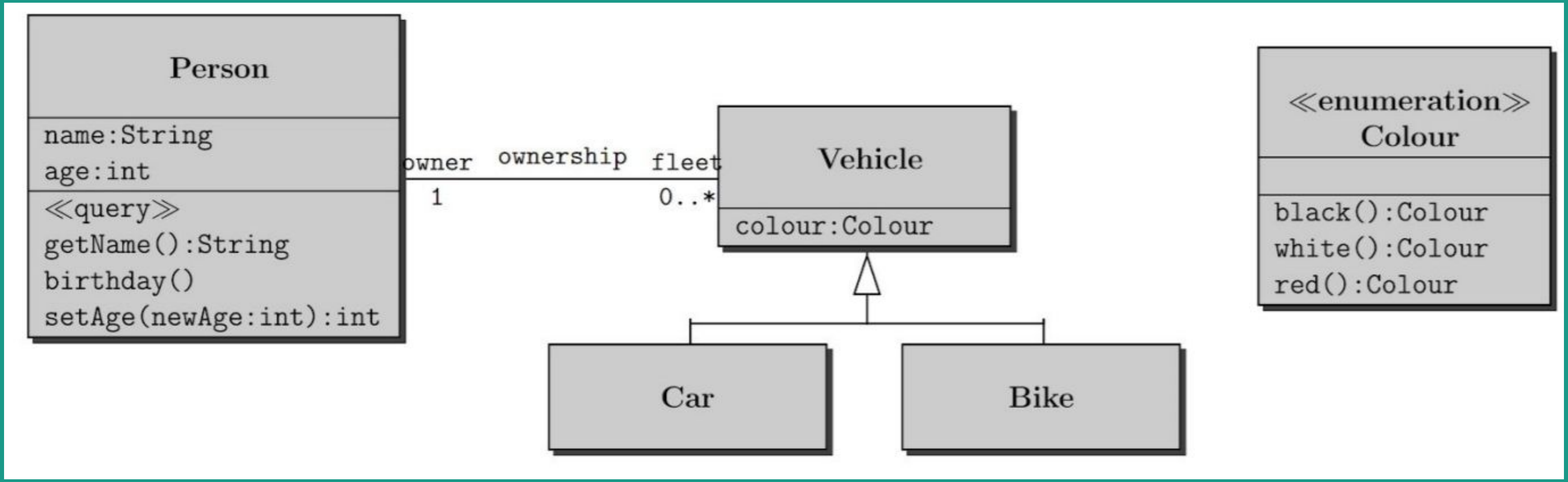
Time events are usually indicated with the keywords **when**, or **after**. **when** specifies a particular time in which the event is triggered.
[e.g. when (date = 2018/12/07)]

after specifies a threshold time **after** which the event is triggered.
[e.g. after (3 months)]

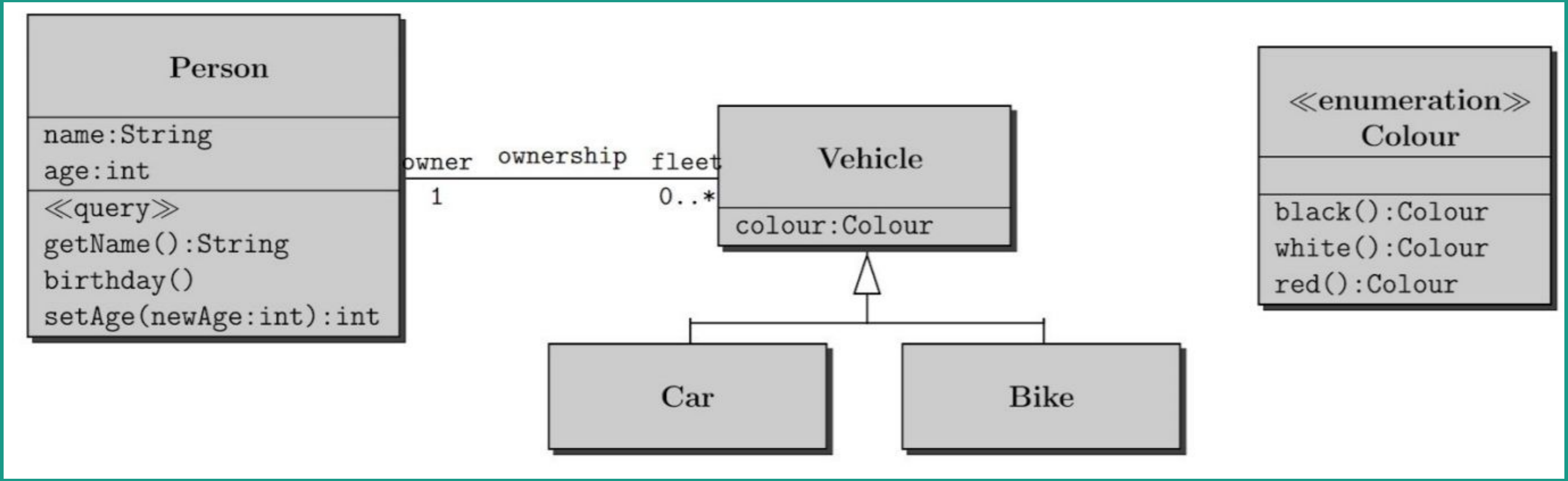


Quiz time

How do we express in OCL that the owner of a car must be older than 18?



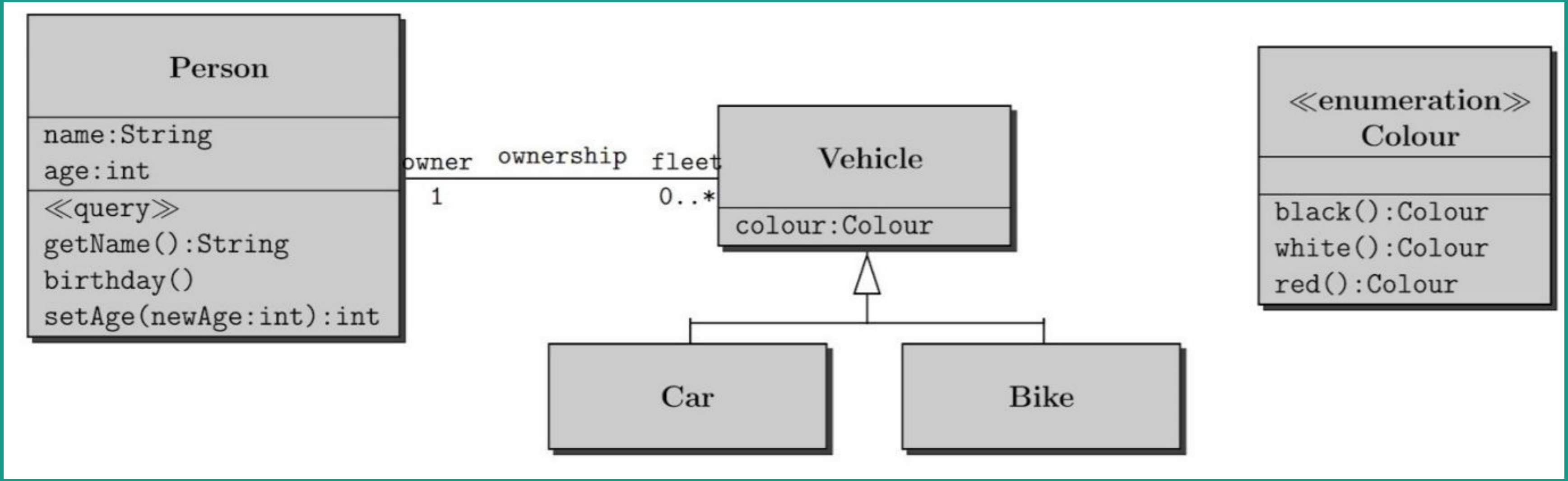
How do we express in OCL that the owner of a car must be older than 18?



```
context Vehicle
    inv: self.owner.age >= 18
```

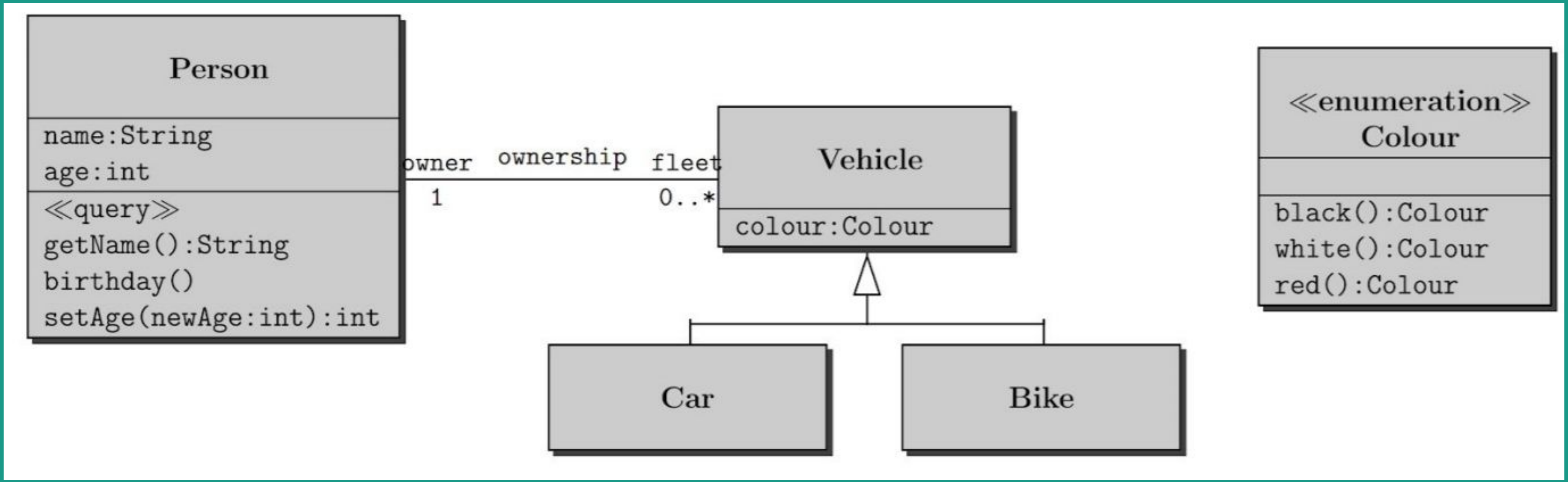
...but this way all persons are over 18, how about a child having a bike?

How do we express in OCL that the owner of a car must be older than 18?

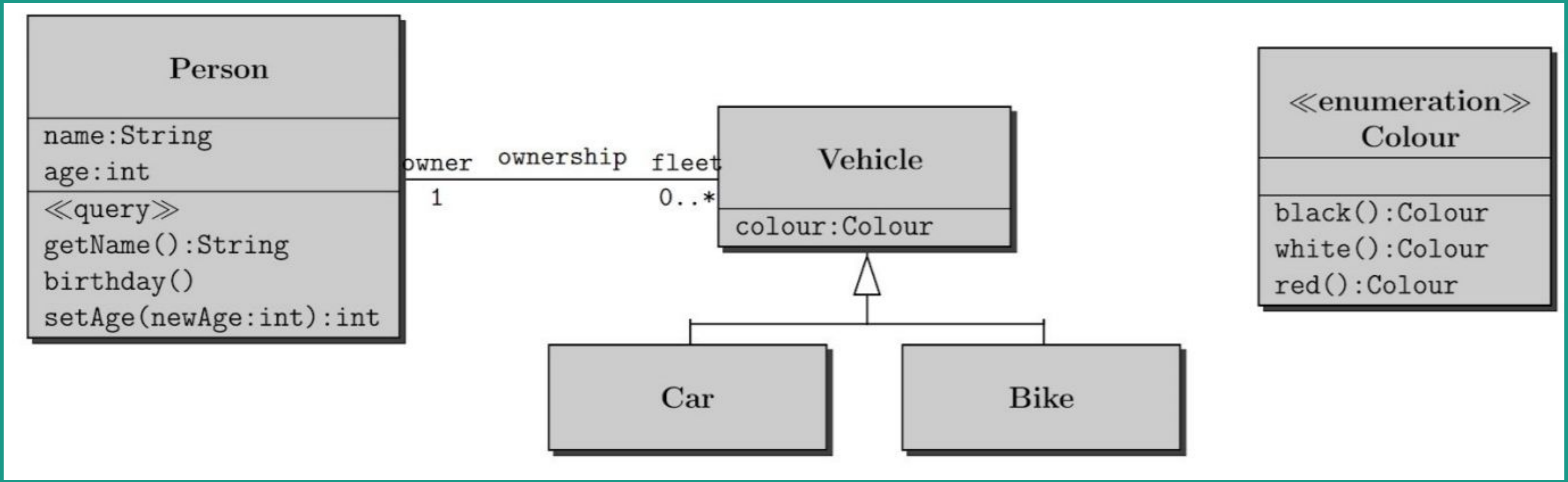


```
context Person
    inv: self.age < 18 implies
            self.fleet->forAll( v | not v.isKindOf(car))
```

How do we express in OCL that there ~~must~~ be a red car?

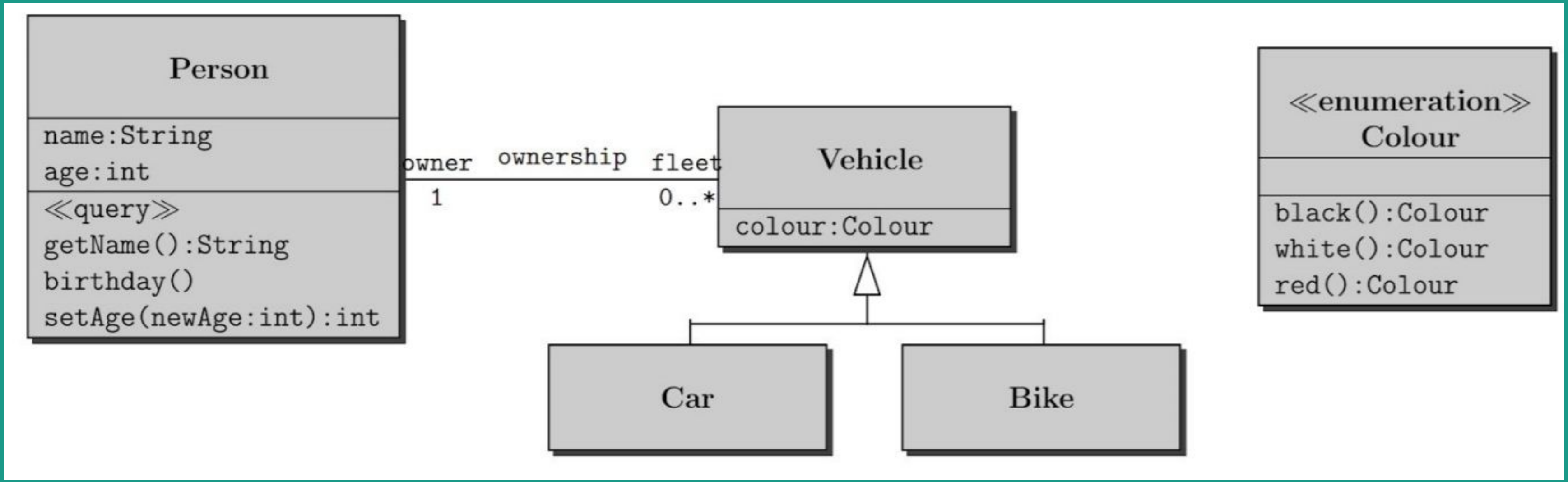


How do we express in OCL that there must be a red car?

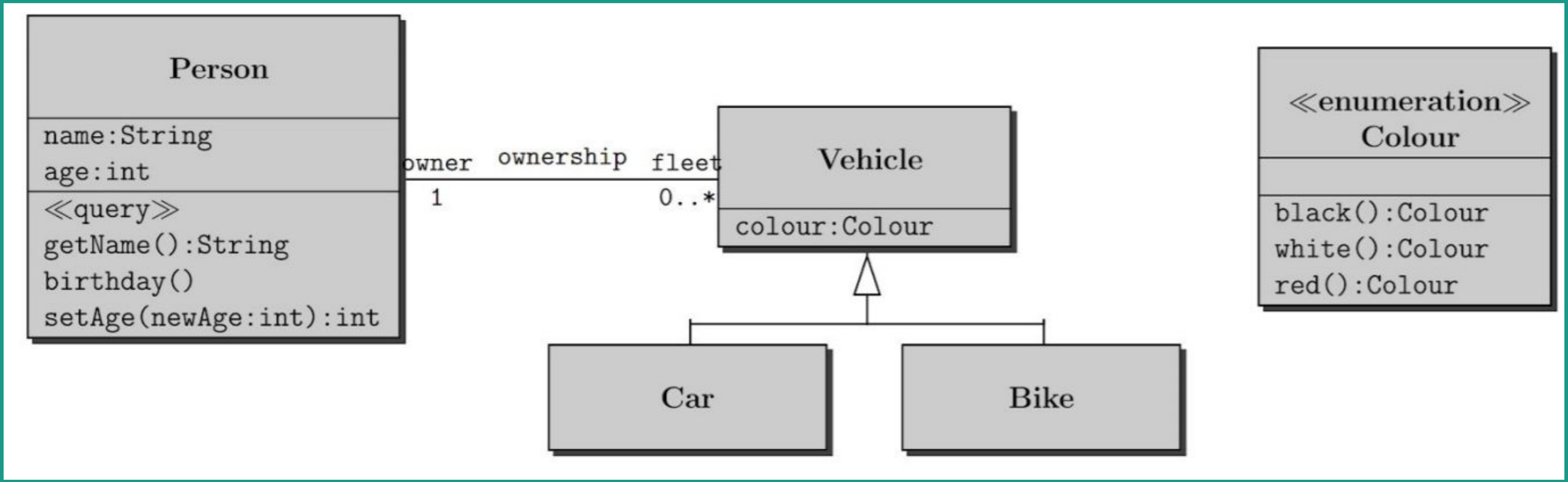


```
context Car
inv: Car.allInstances()->
          exists( c | c.colour = Colour::red )
```

How to express in OCL that the result of calling the method birthday increases the age by 1 ?



How to express in OCL that the result of calling the method birthday increases the age by 1 ?

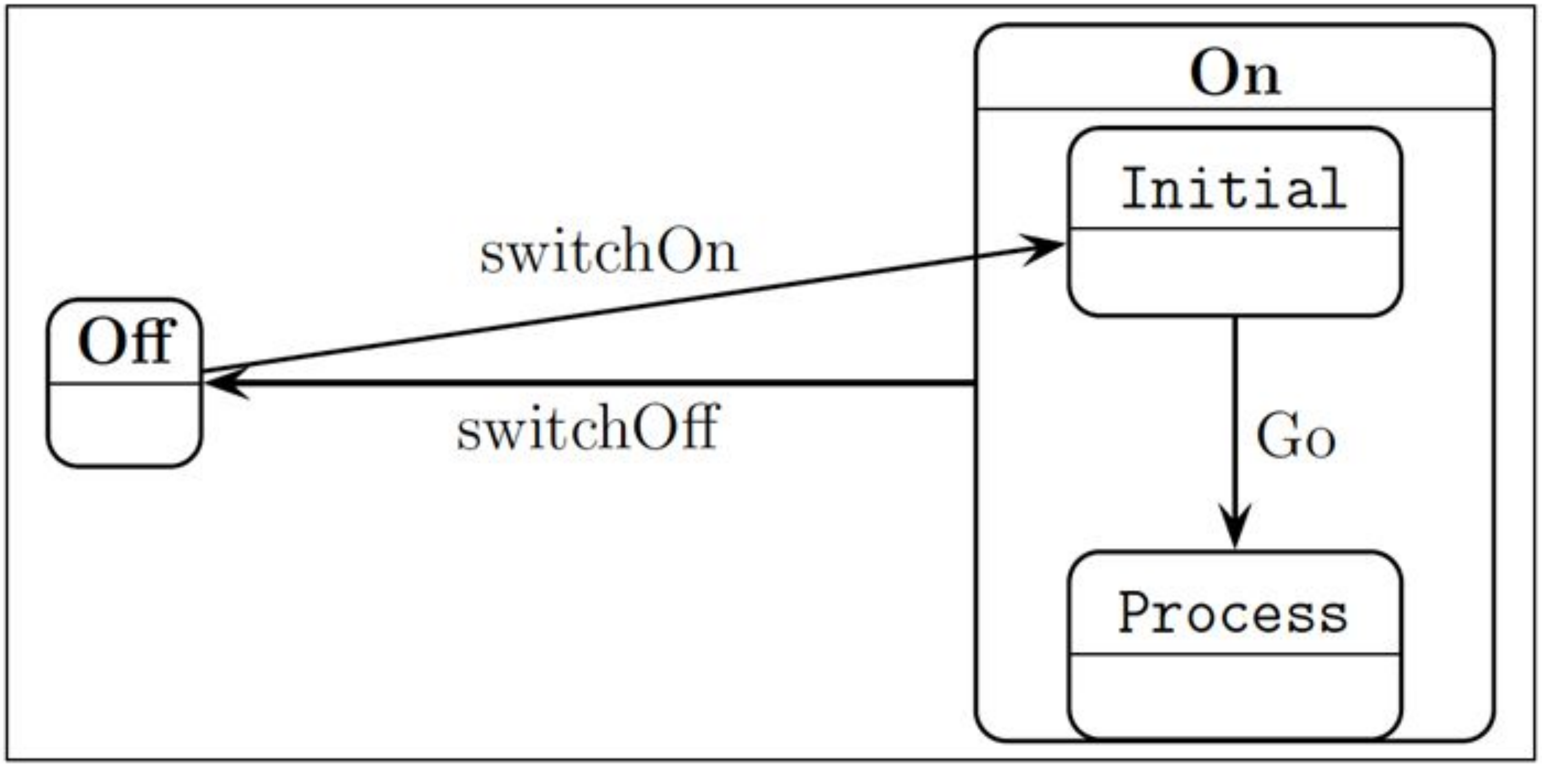


```
context Person::birthday()
    post: self.age = self.age@pre + 1
```

Composite states

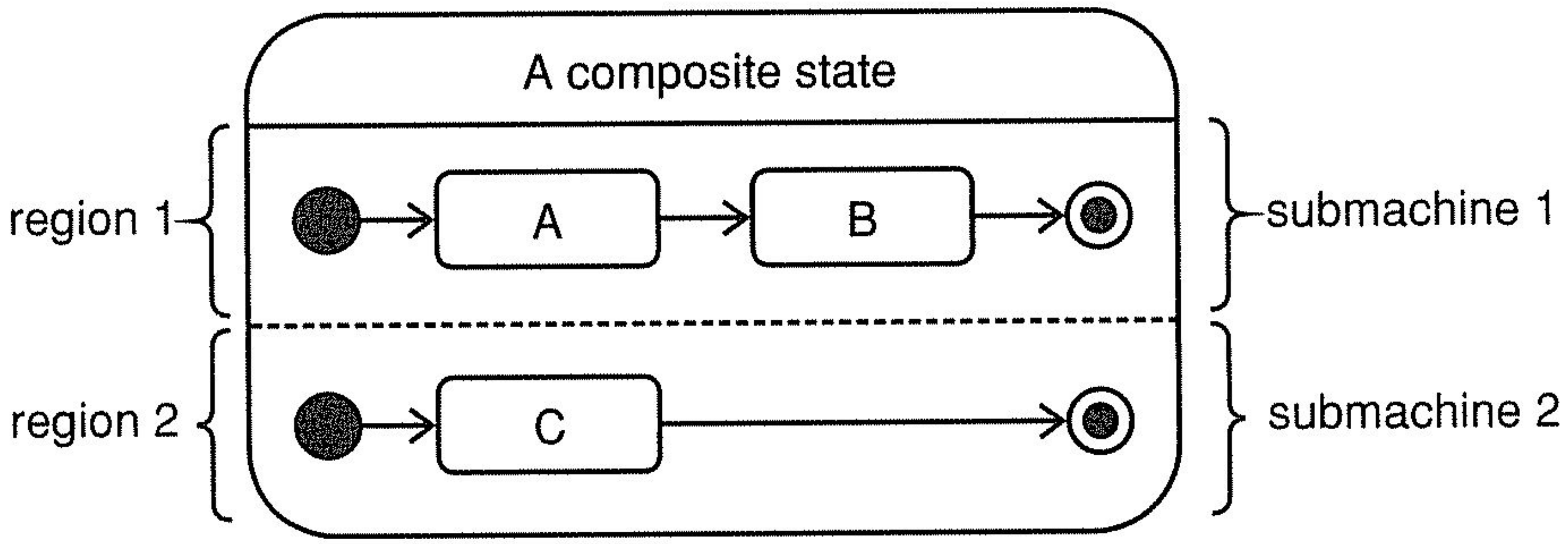
Dealing with model complexity

`Initial` and `process` are substates of `On`
`Initial` and `Process` "inherit" the `switchOff`

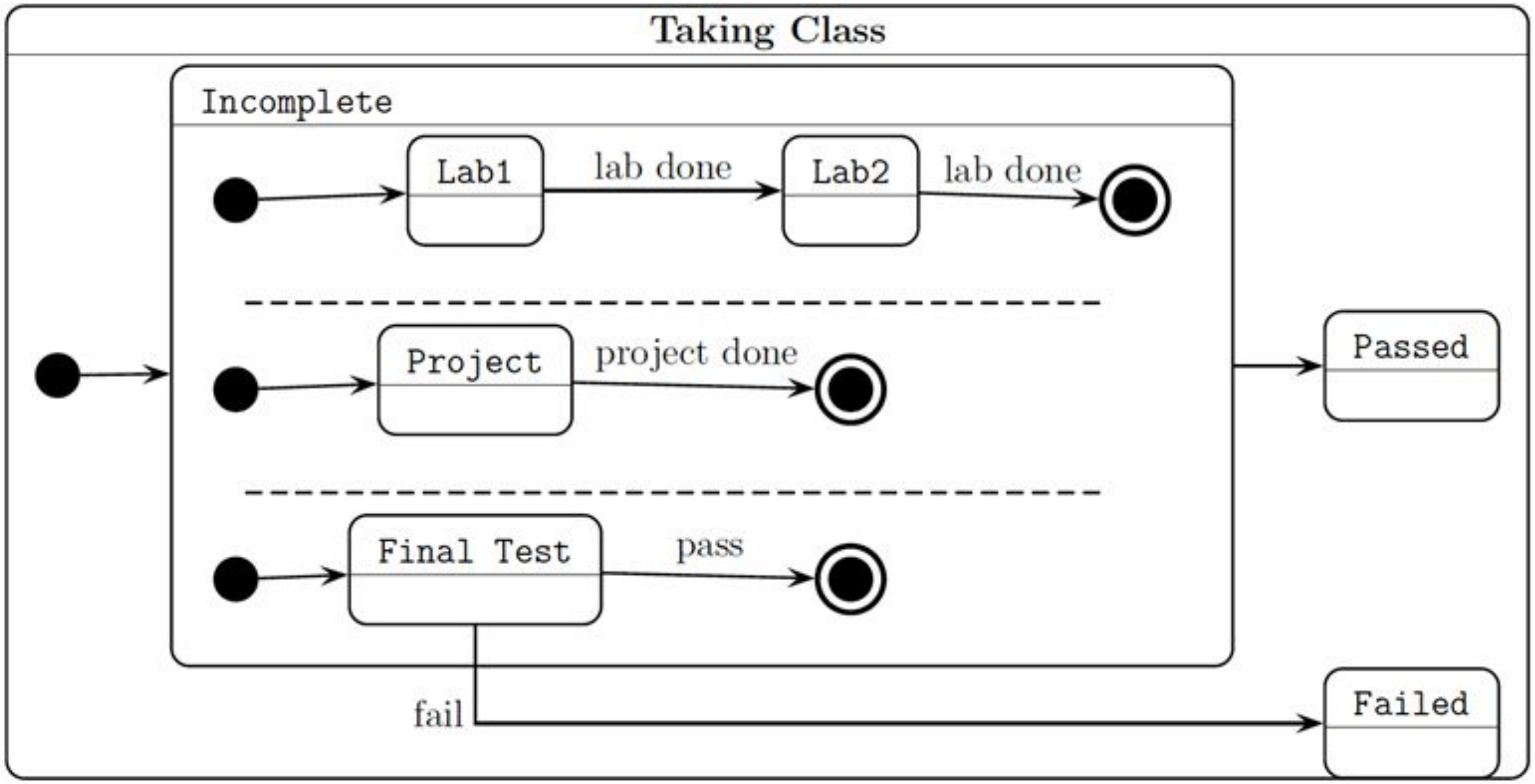


Composite states contain one or more nested submachines

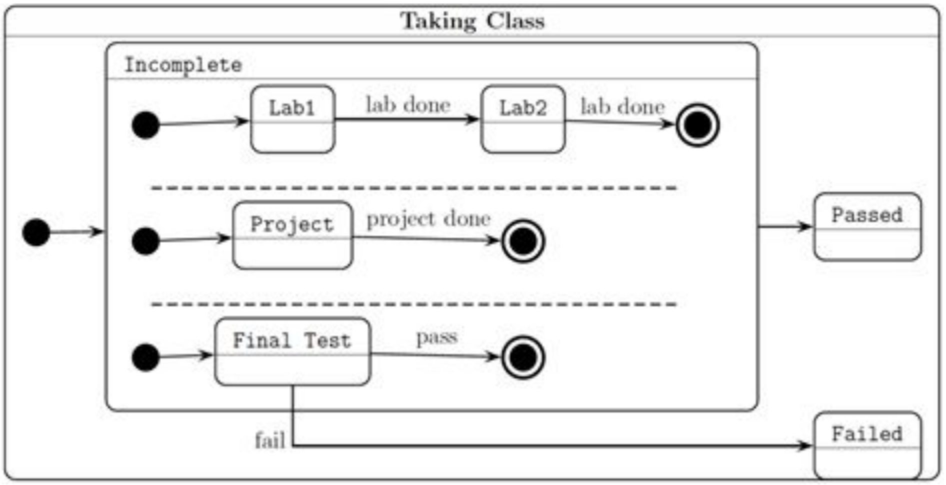
- Each submachine exists in its own region within the composite state

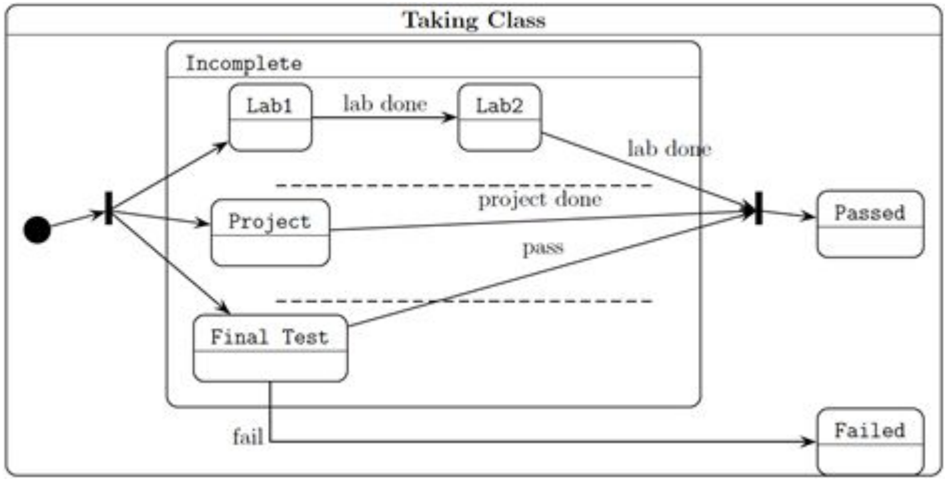


Regions are independent, concurrent parts of a composite state - they are activated synchronously



Entering and exiting composite states



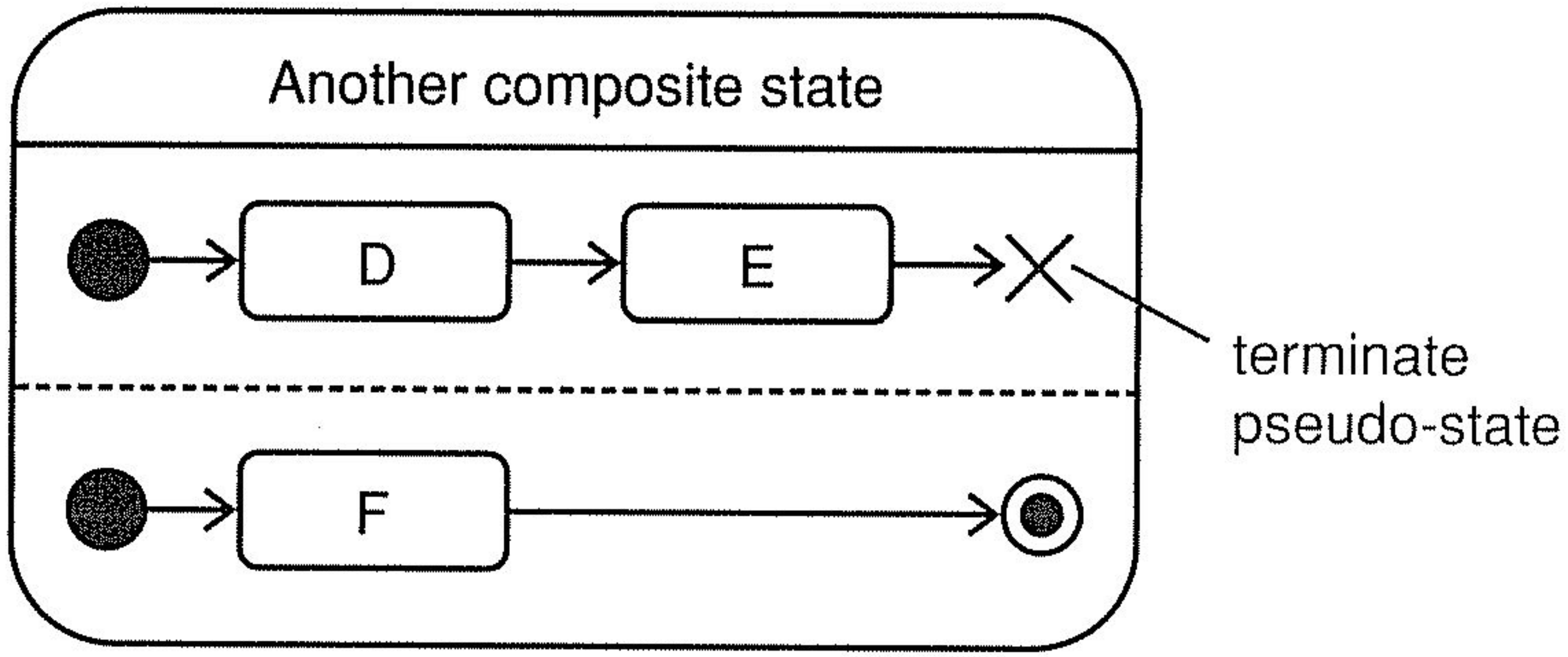


To enter in a composed state, you need to have an initial substate in each region

This is an equivalent alternative to representing this model, using pseudo-states "**fork**" and "**join**".

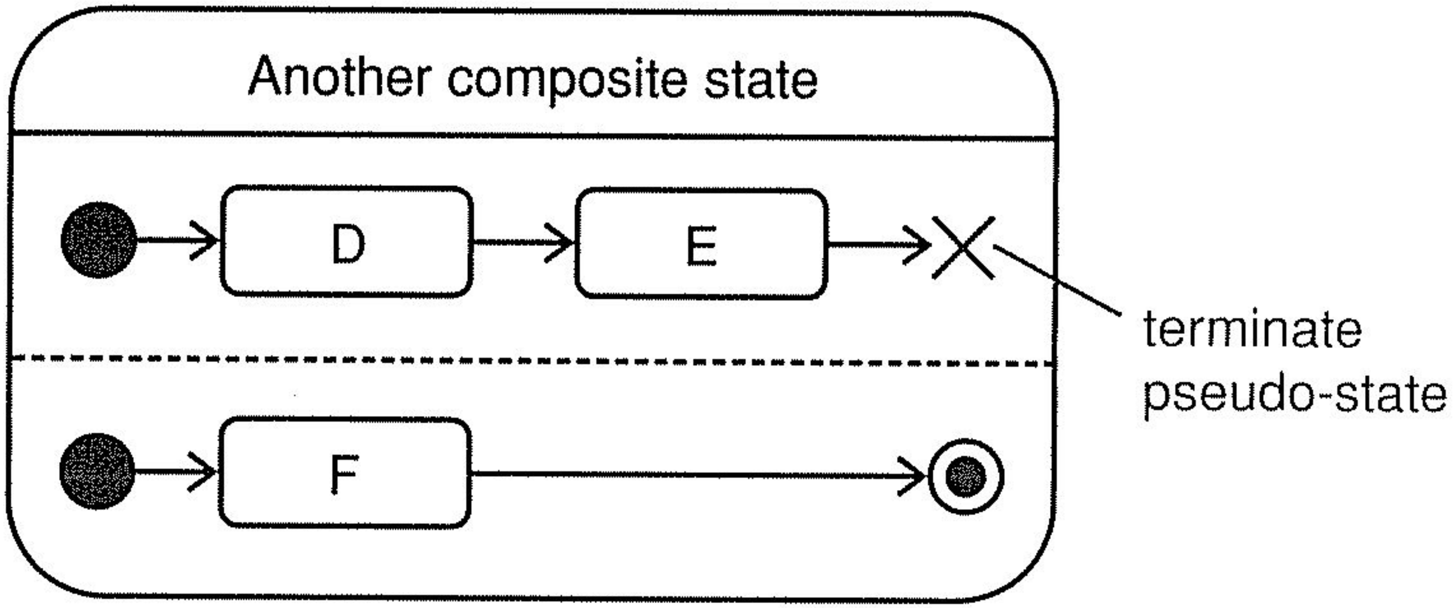
Nested substates inherit all of the transitions of their containing superstates

- If the composite state has a transition, each of its nested states also has this transition



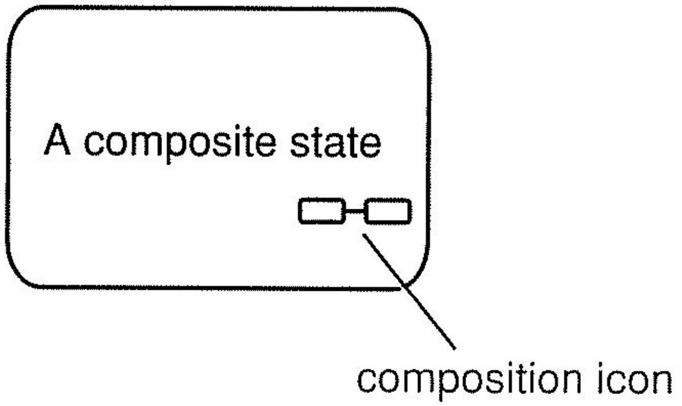
The final pseudo-state of a submachine only applies within that region

- If the first submachine reaches its final state first, that region will terminate, but the second region will continue to execute
- In contrast, when the terminate pseudo-state of the second region is reached, the whole composite state stops



Nested states can also be composite states

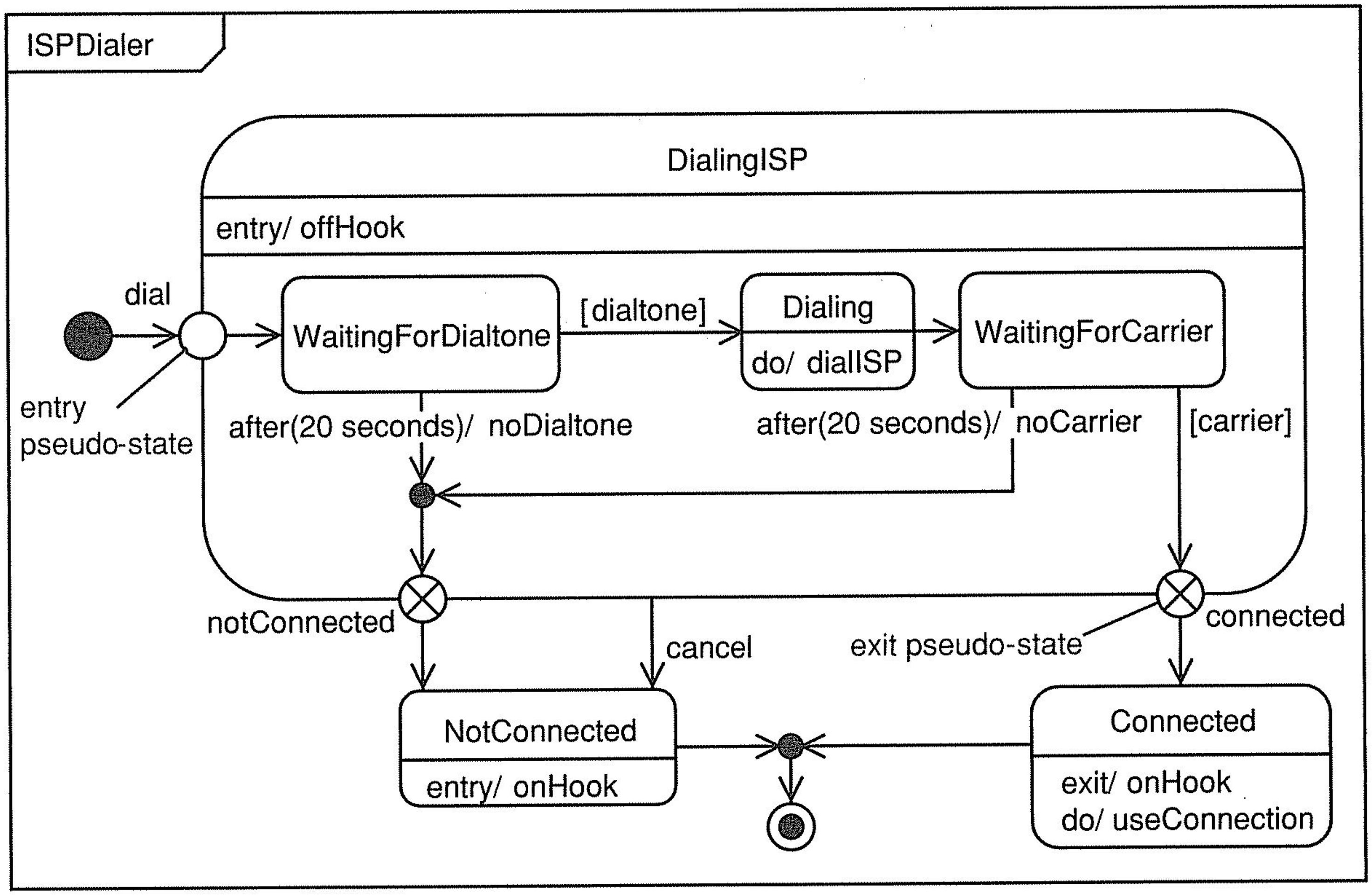
- This mechanism should be used with parsimony
  - When possible keep nesting to two or three levels
    - Beyond that, the state machine tends to become hard to read and understand
- To keep the state machine simple, you may hide the details of a composite state
  - Use the composition icon to denote that a given state is a composite state



Simple vs orthogonal composite spaces

- Simple composite state
  - One region only
- Orthogonal composite state
  - Two or more regions

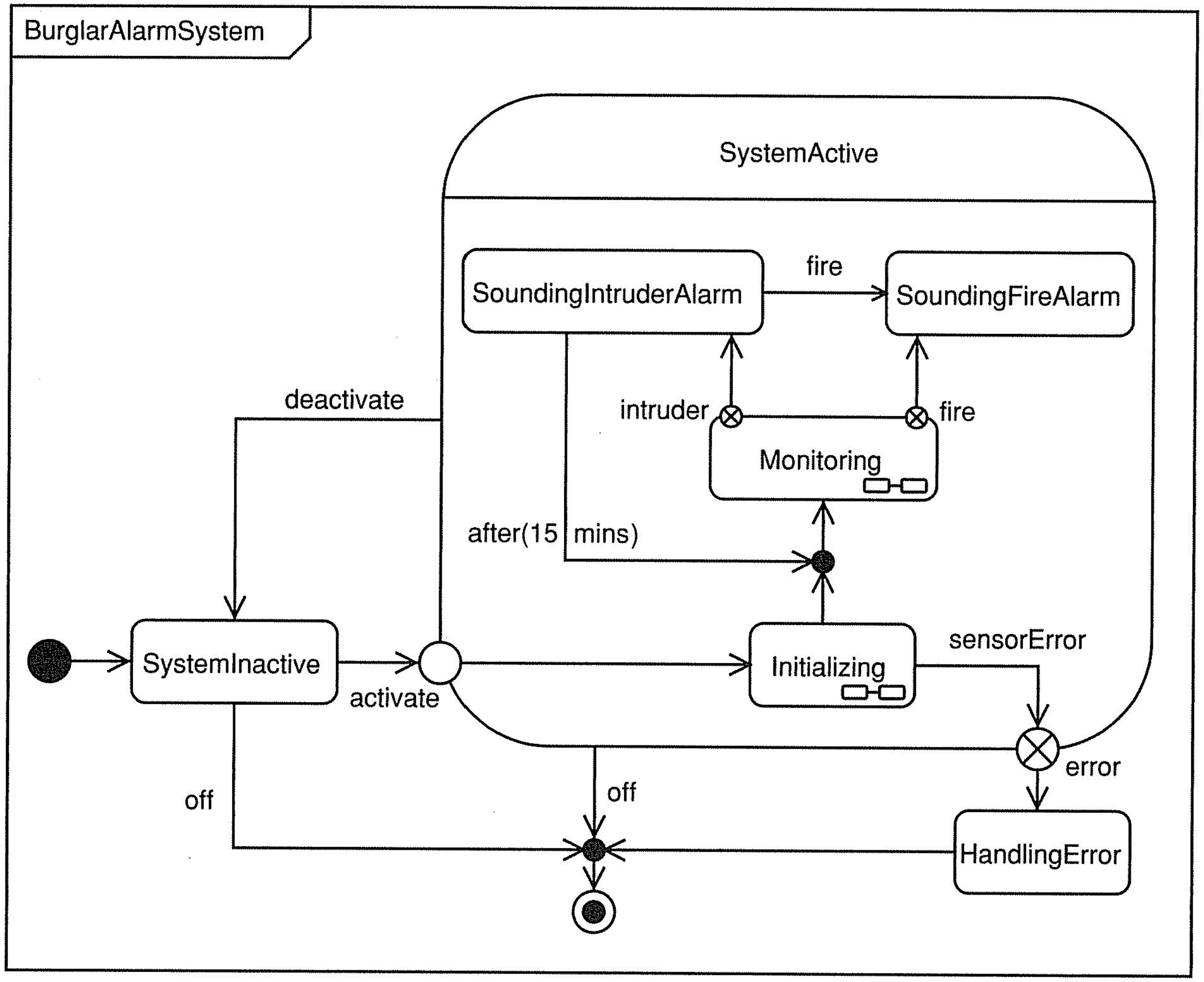
Simple composite states contain a single nested state machine



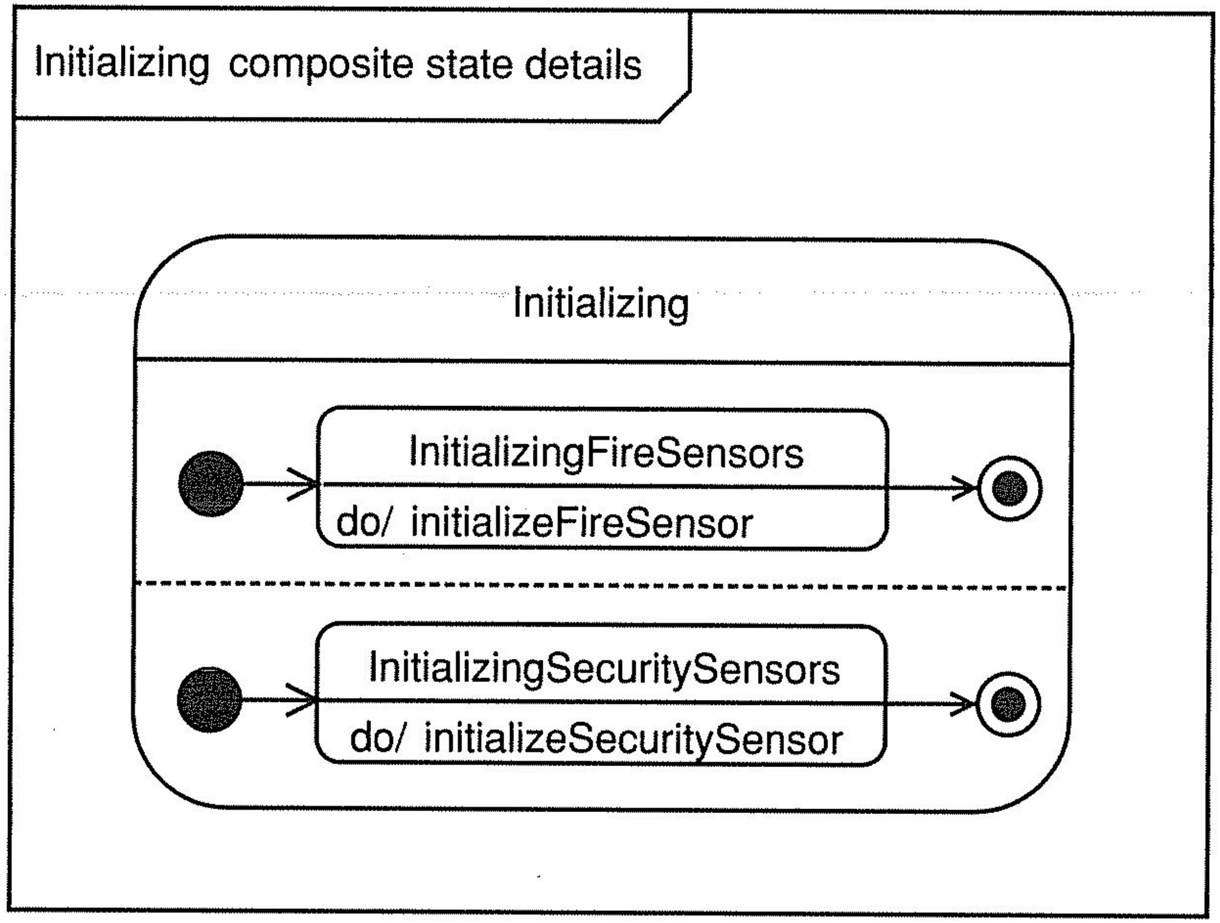
Orthogonal composite states contain two or more nested state machines that execute concurrently

- When you enter an orthogonal composite state, all its submachines start at once (this is an implicit fork)
- You can exit an orthogonal composite state
  - When all its submachines finish (this is an implicit join)
  - When one of its submachines transitions to a state outside the supermachine, usually via an exit pseudo-state
    - This does not cause a join - no synchronization among submachines
    - The remaining submachines simply abort

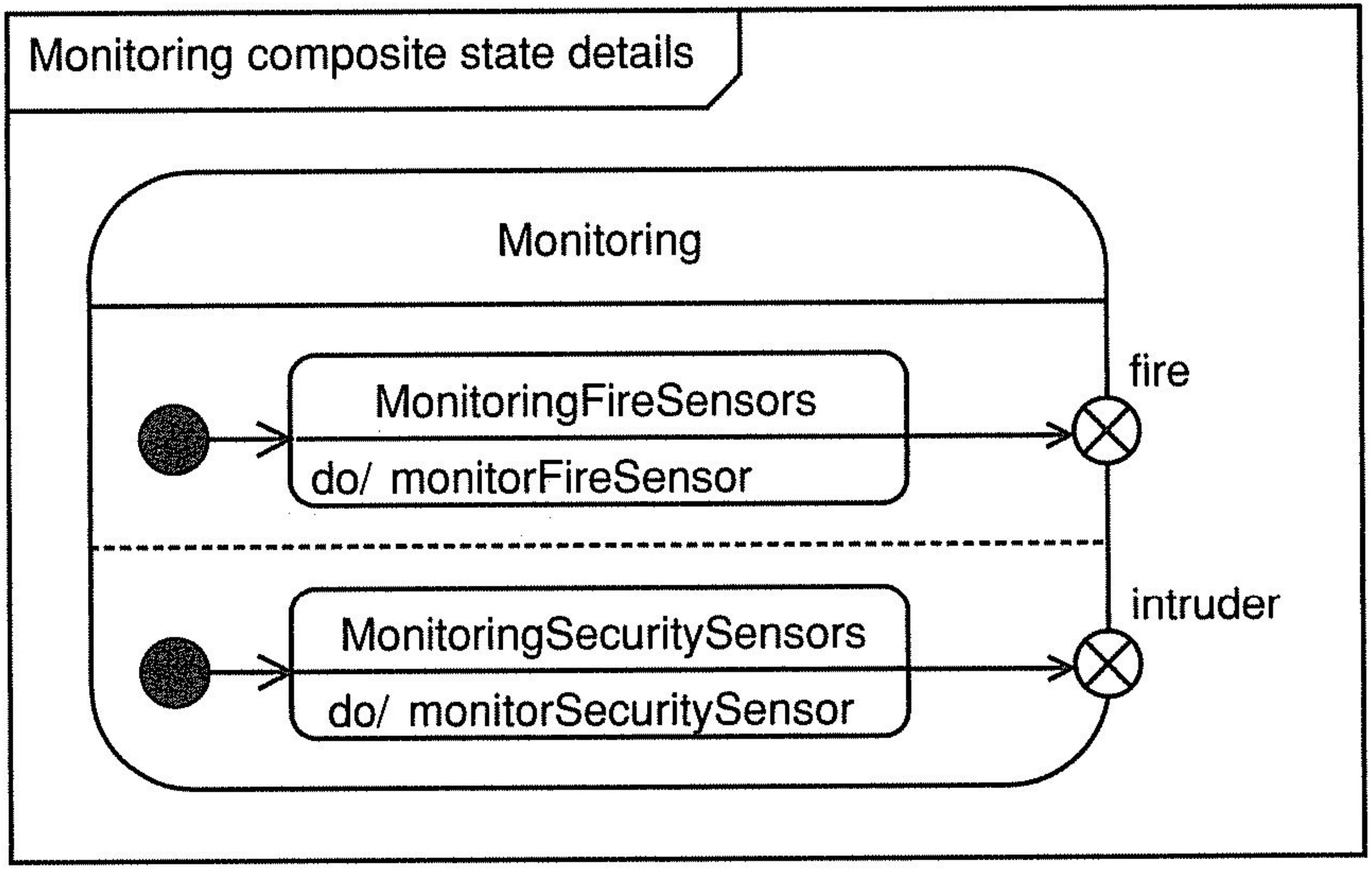
Orthogonal composite state example



Initializing composite state - synchronization between submachines


Initializing composite state details
Initializing
InitializingFireSensors
do/ initializeFireSensor
InitializingSecuritySensors
do/ initializeSecuritySensor

Monitoring composite state - no synchronization between the two submachines

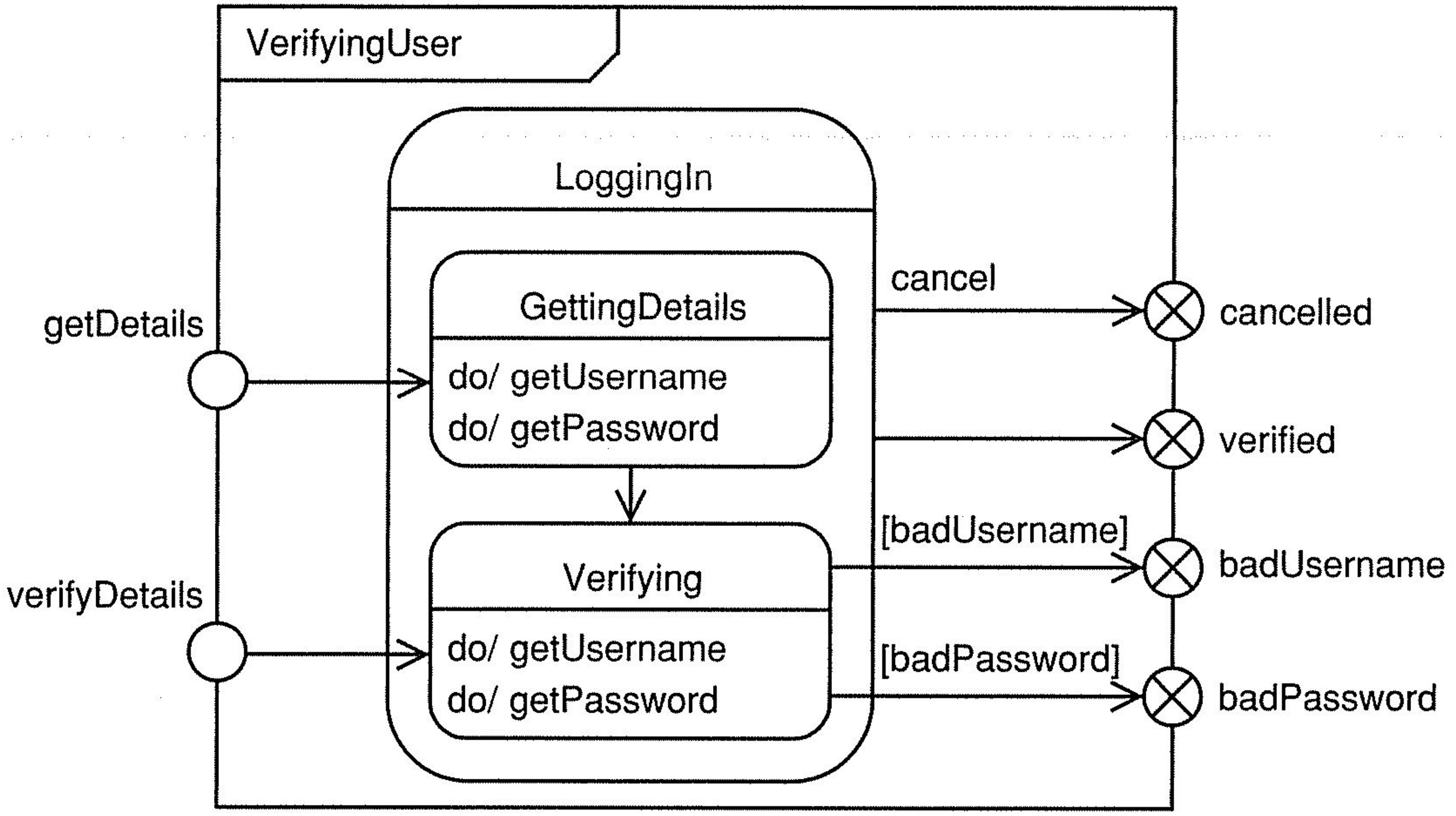

Monitoring composite state details
Monitoring
MonitoringFireSensors
do/ monitorFireSensor
fire
MonitoringSecuritySensors
do/ monitorSecuritySensor
intruder

A submachine state references another state machine (recorded in a separate diagram)

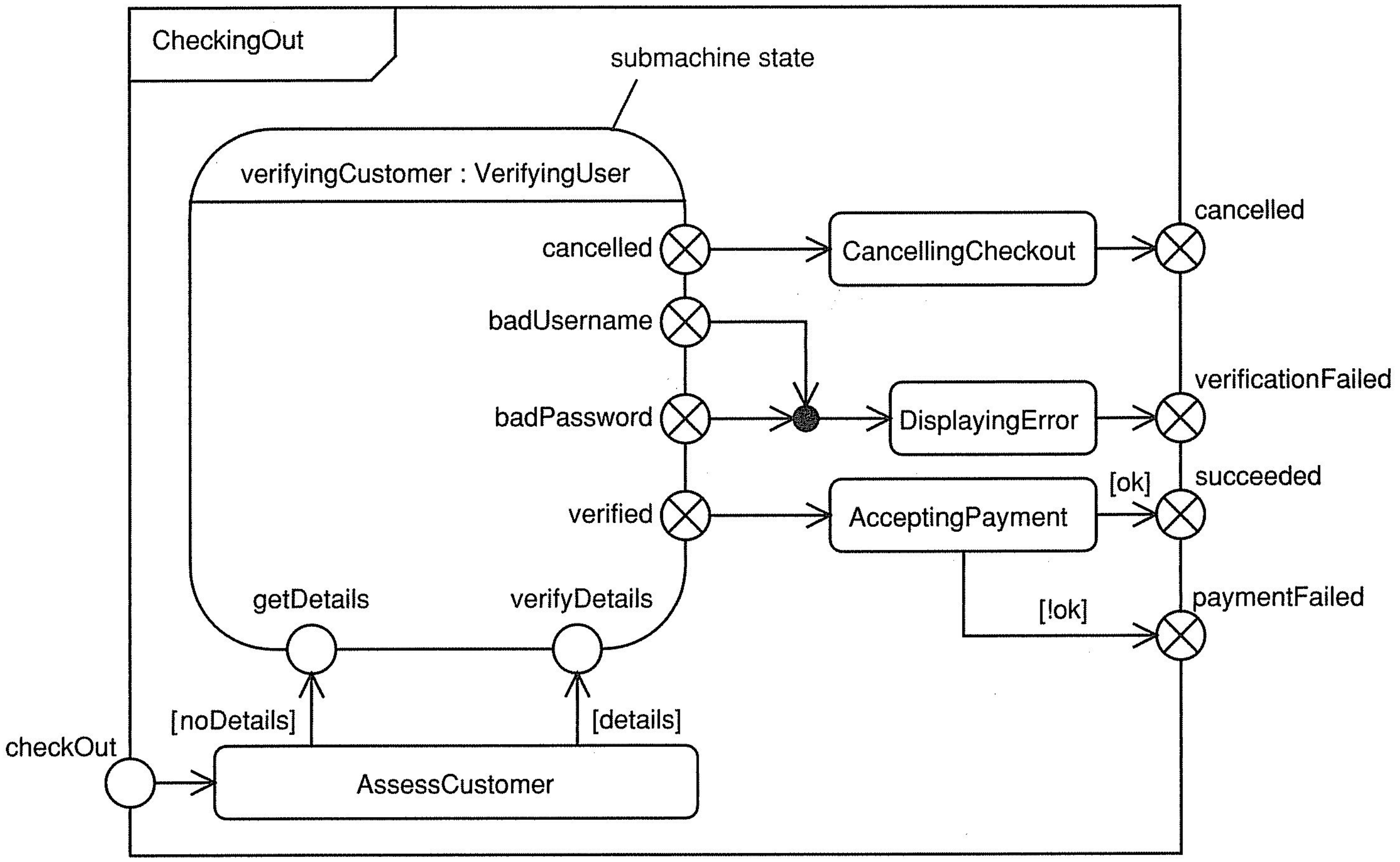
- Semantically equivalent to a composite state
- Submachine states are used to simplify complex state machines
- Submachine states provide a reuse mechanism
- Syntax:

```
state name: name of referenced state machine diagram
```

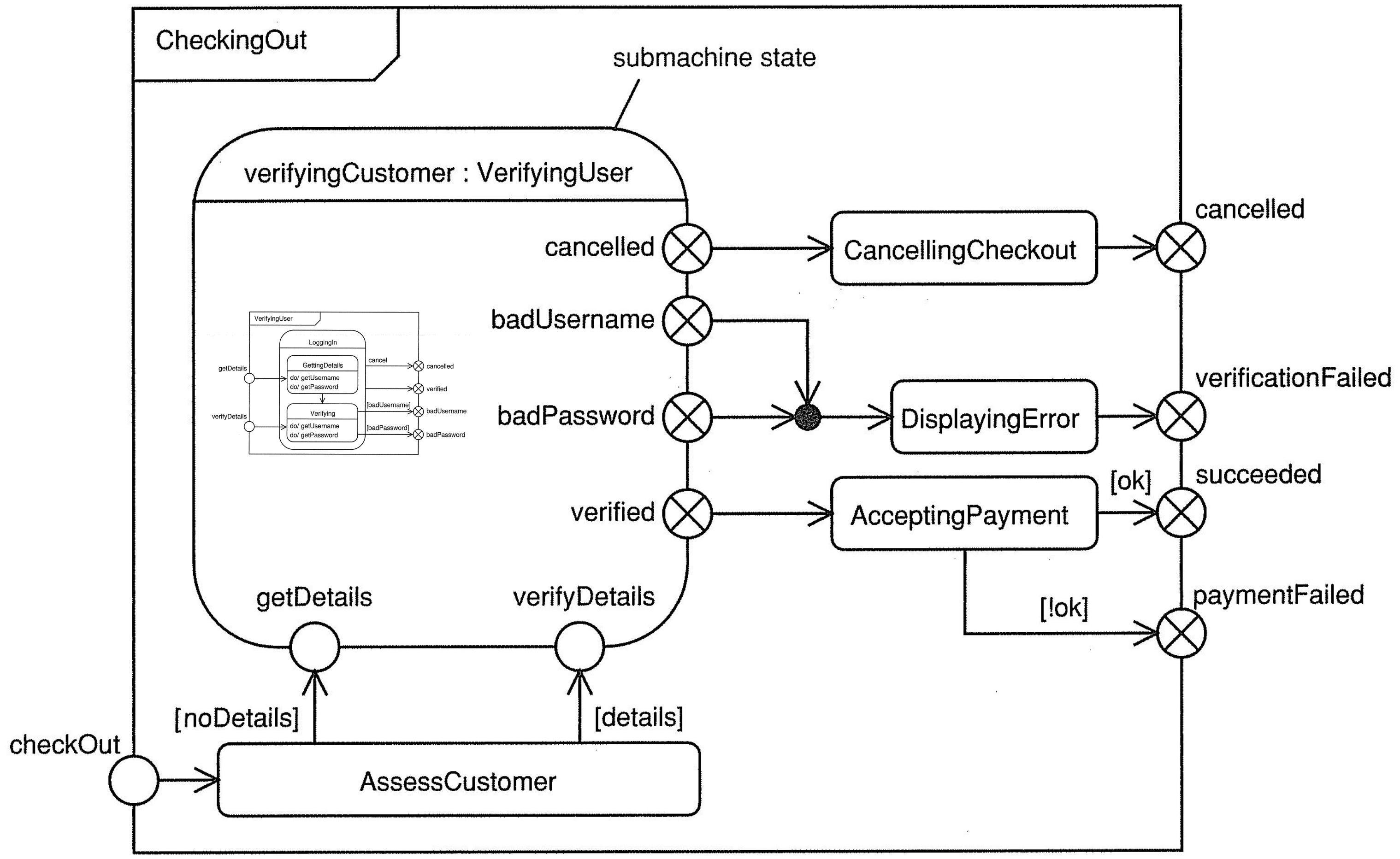
Consider this machine state



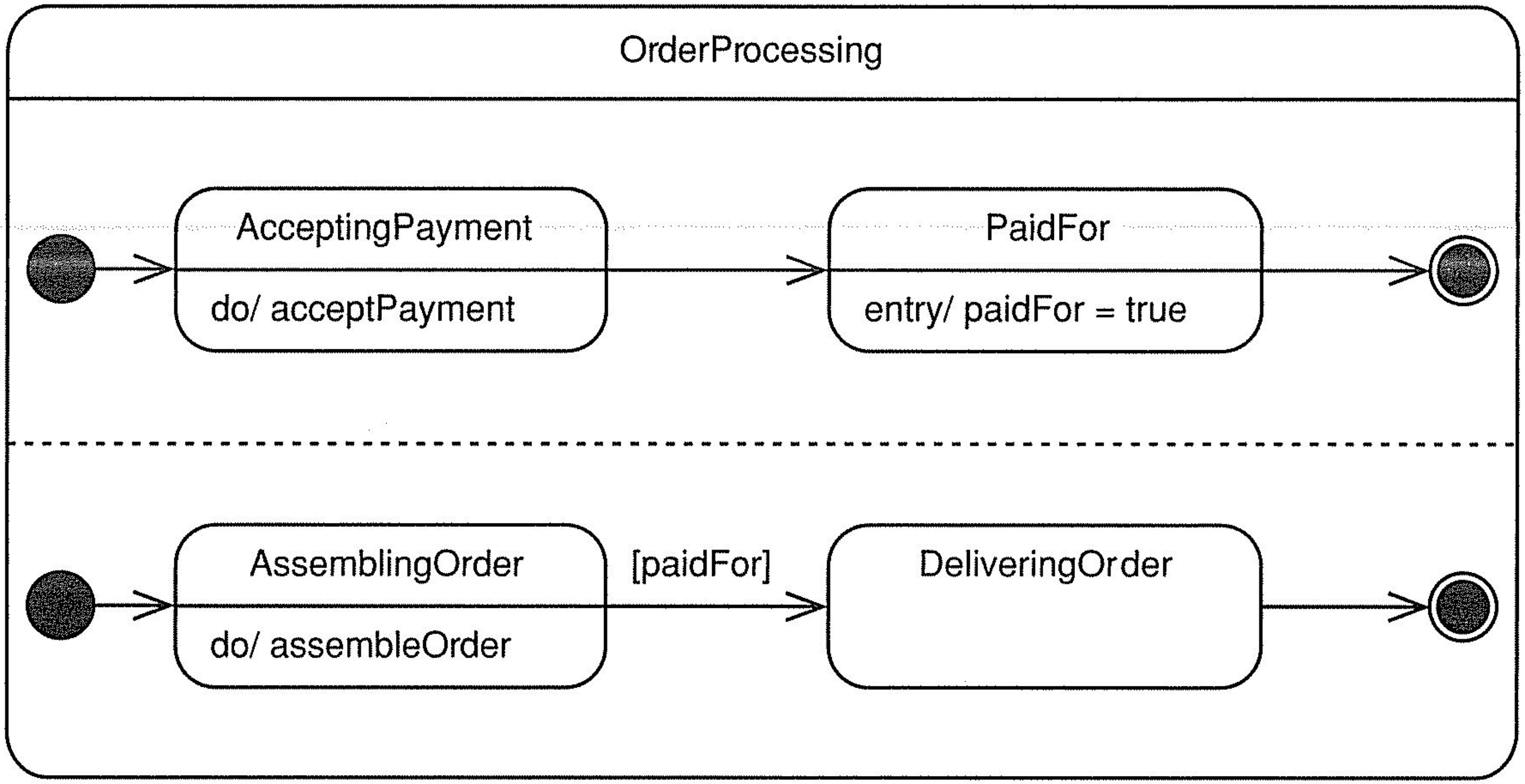
Now, we can reuse the VerifyingUser submachine state



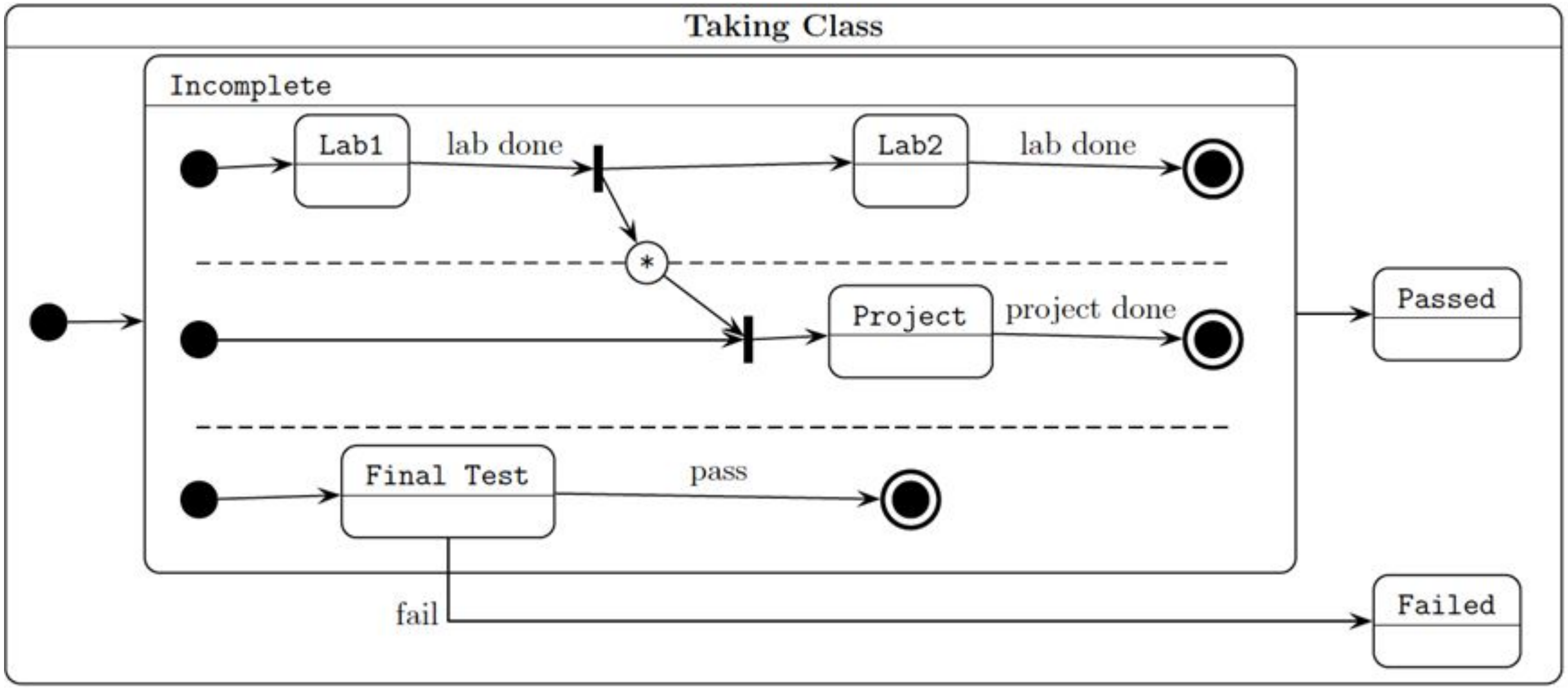
It is as if the contents of the submachine state are replaced by VerifyingUser



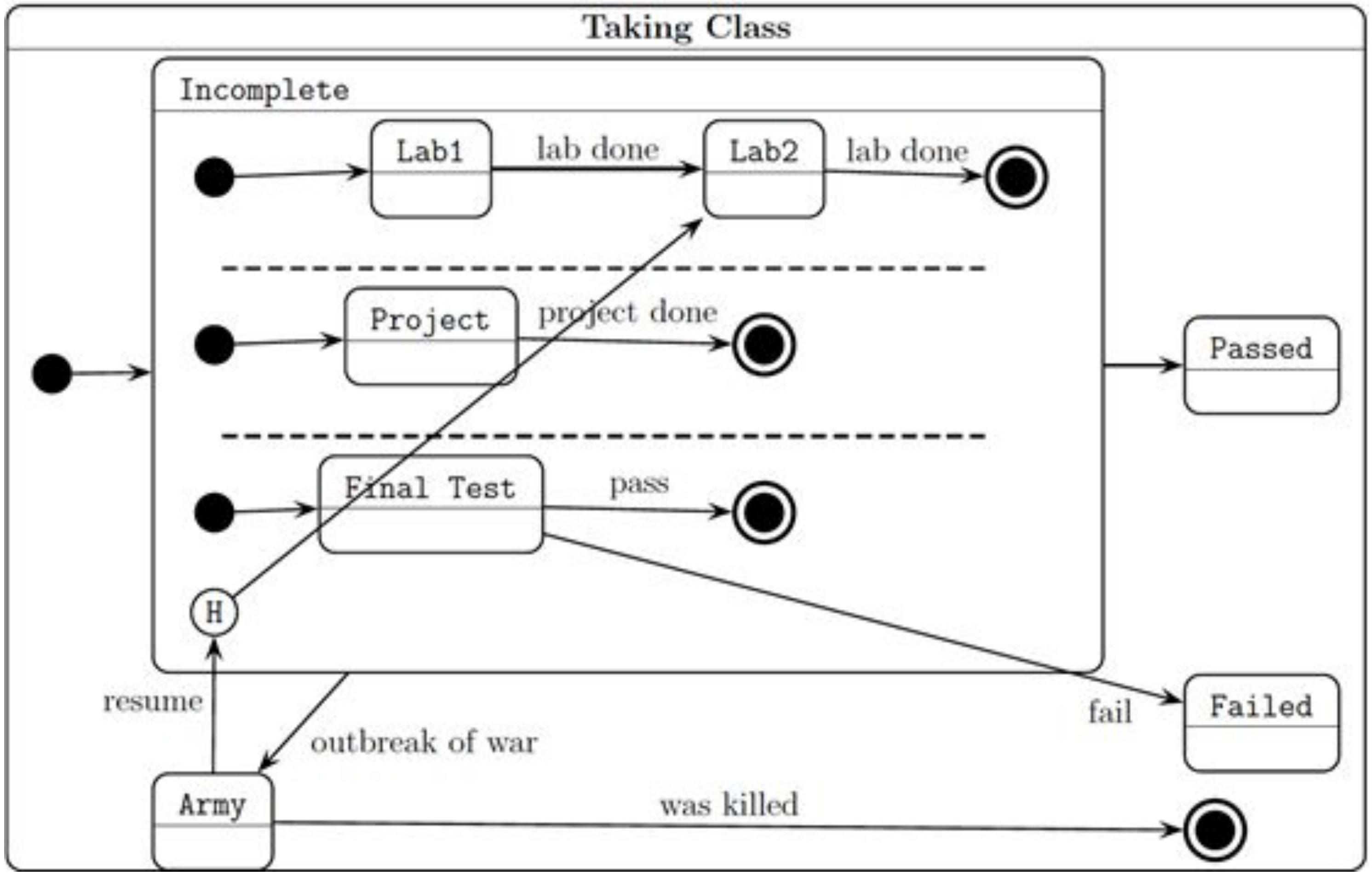
You can use attribute values to communicate asynchronously between concurrent submachines



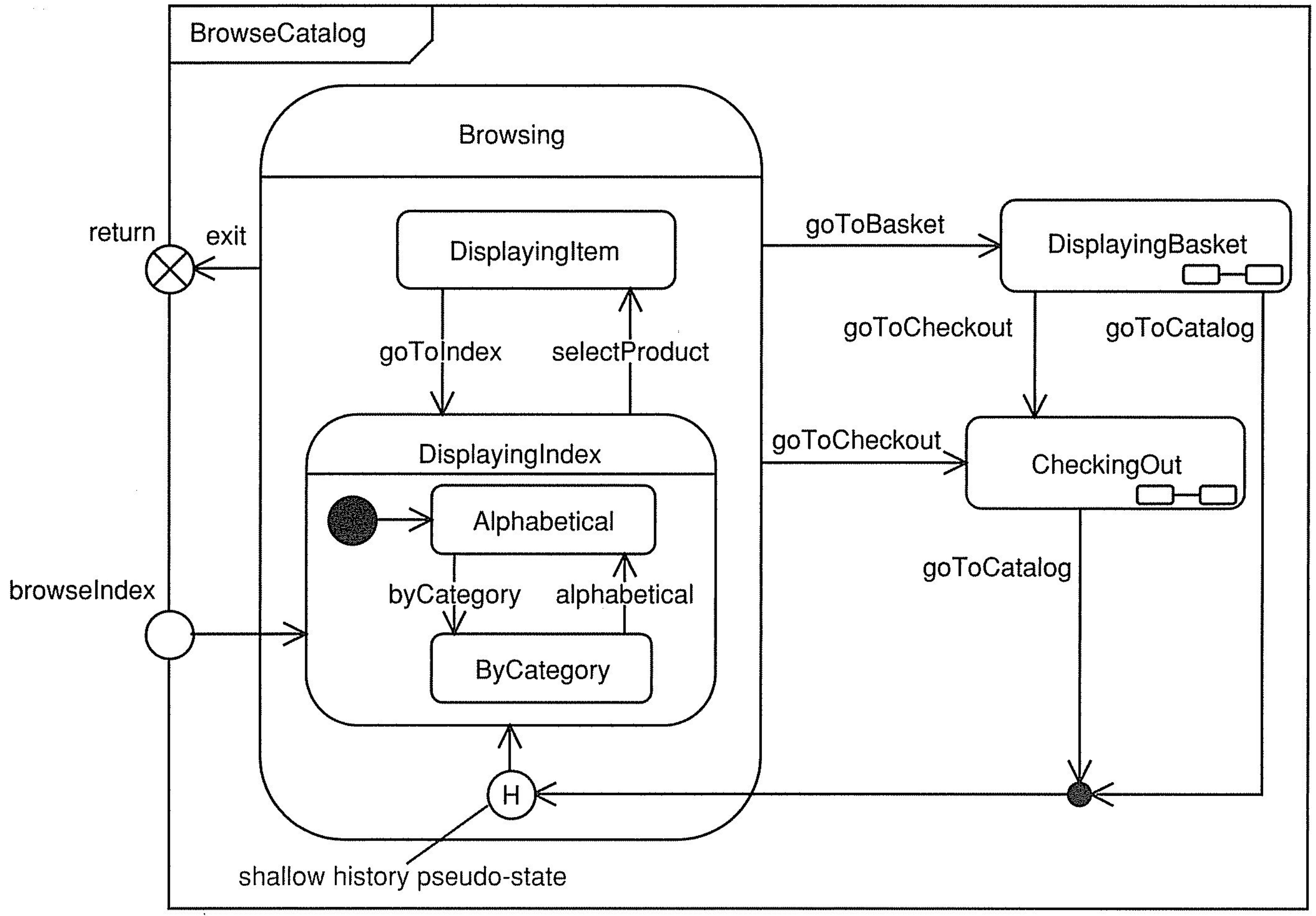
You can also use Synch state synchronization points to synchronize different regions (in combination with fork and join)



Shallow history remembers the last substate you were in at the same level as the shallow history pseudo-state

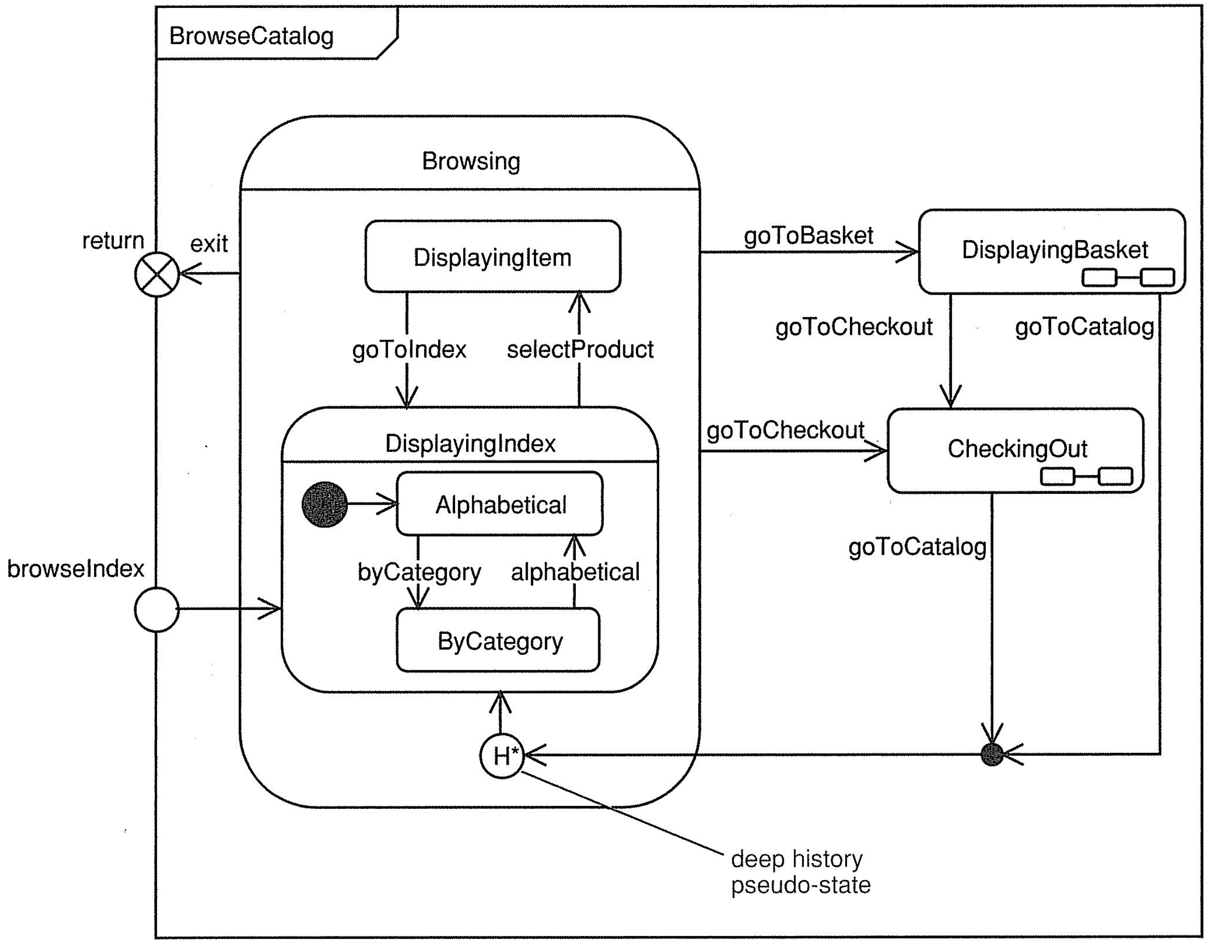


Shallow history remembers the last substate you were in at the same level as the shallow history pseudo-state



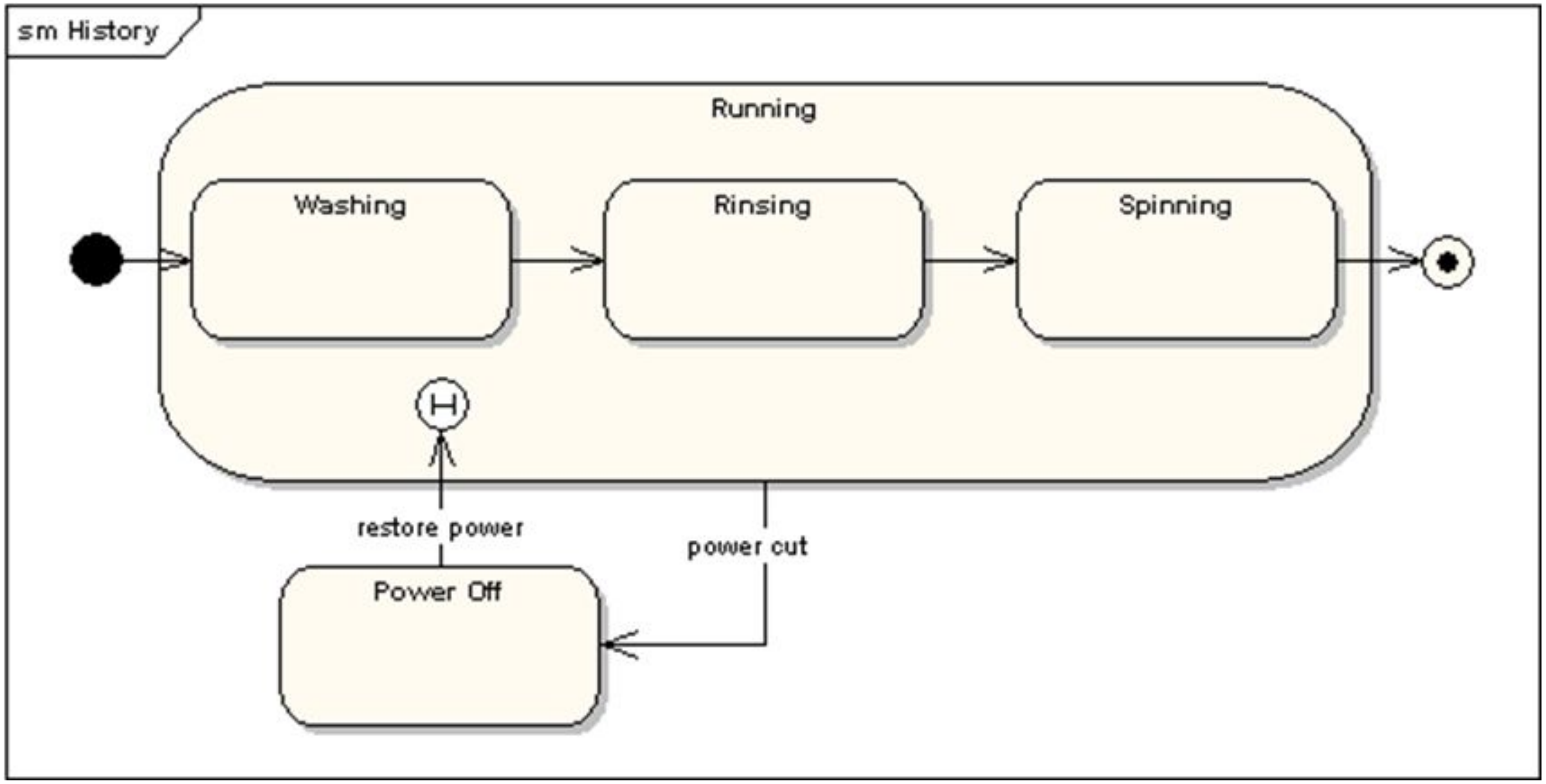
FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

Deep history remembers the last substate you were in at the same level, or lower, than the deep history pseudo-state

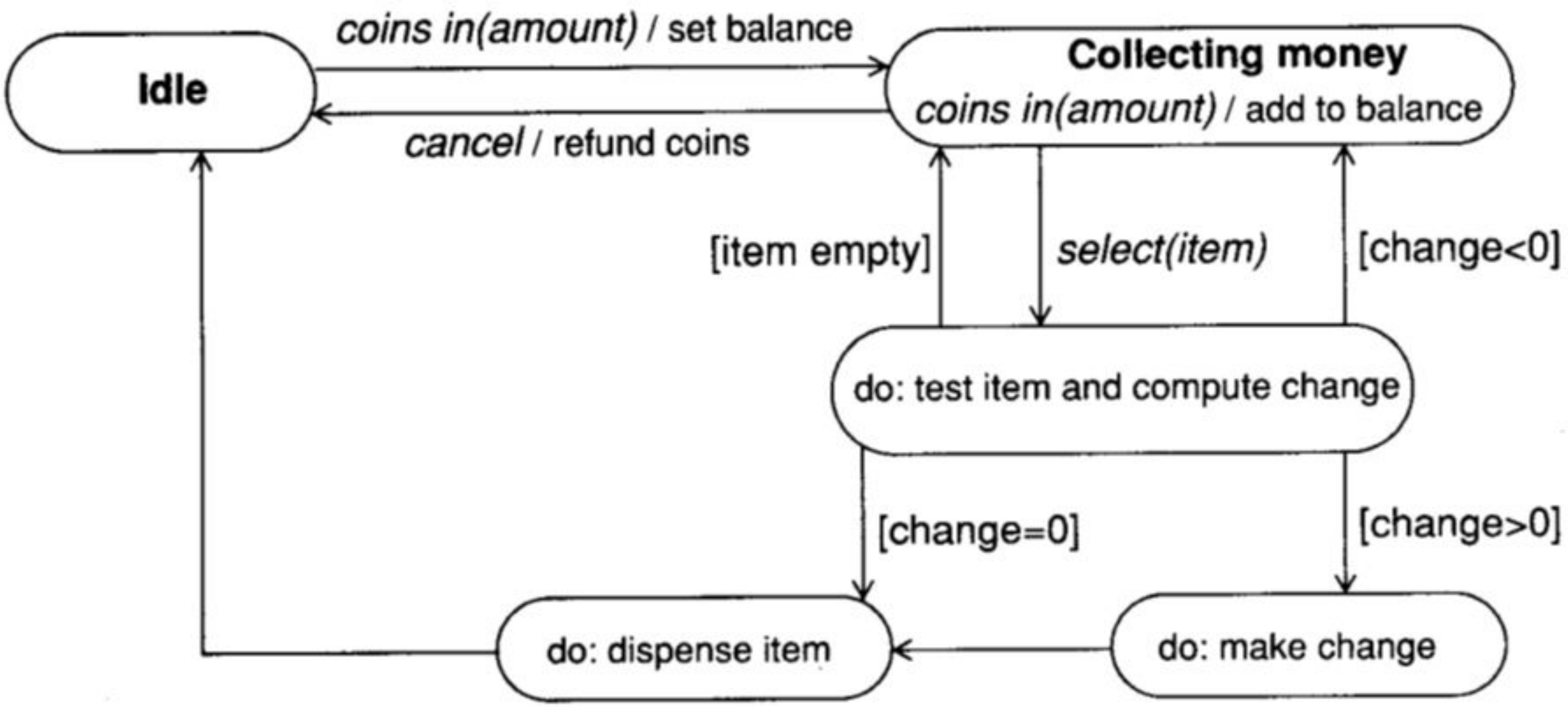


Examples

Examples with memory (history state)

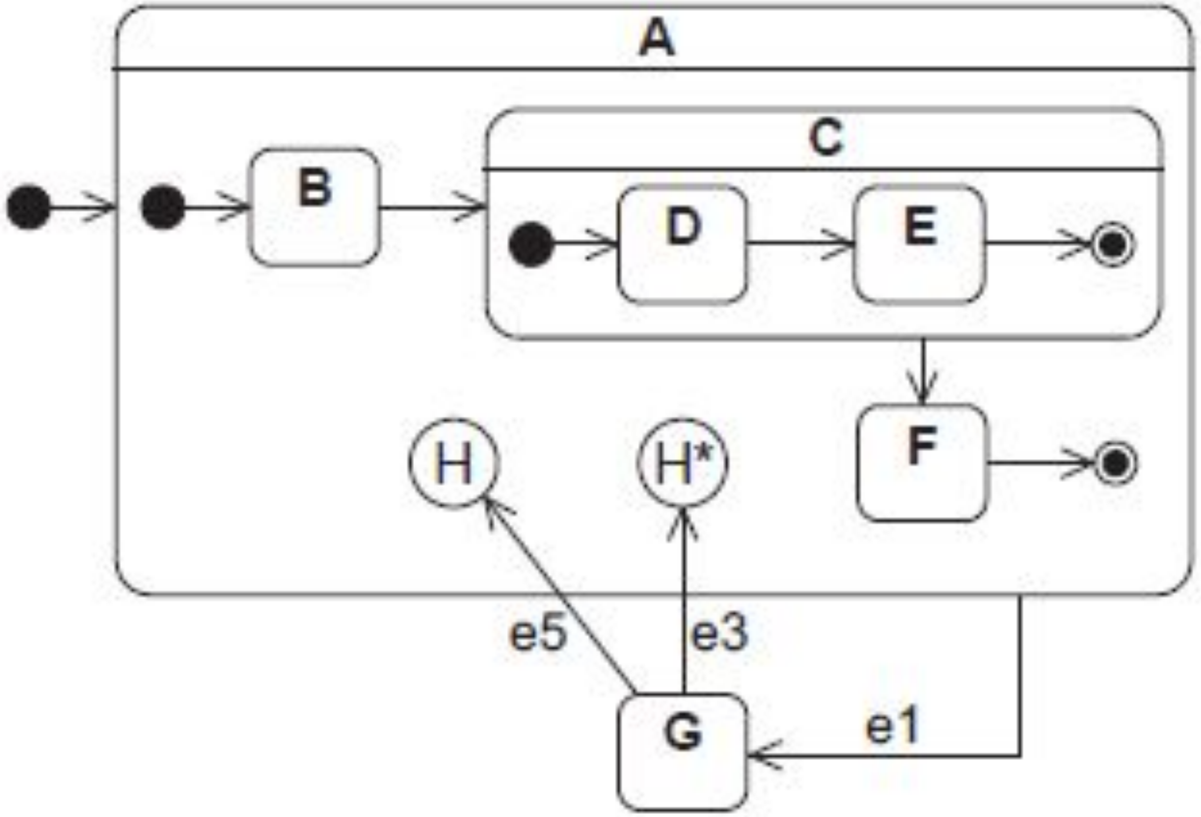


Drink vending machine



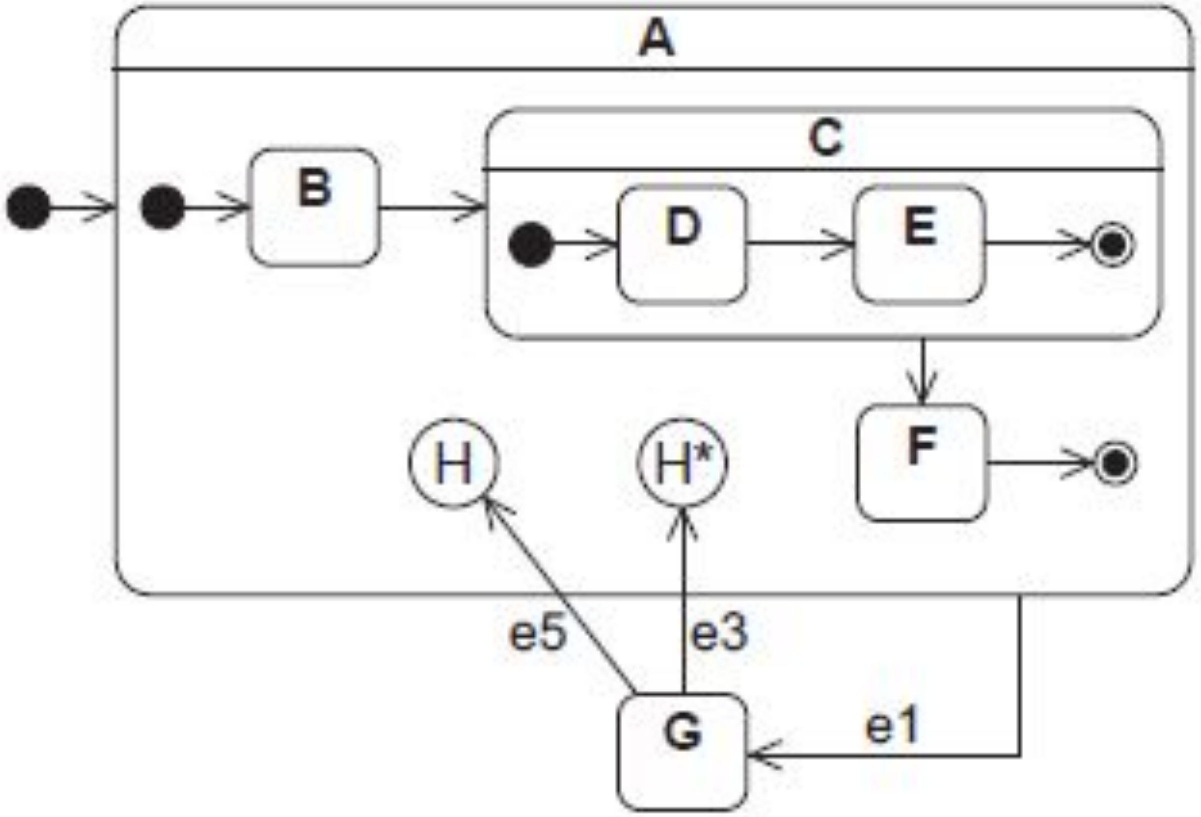
H is called a shallow history state

- It remembers and restores the previously active state at the same nesting depth
- Assuming that G is active and event e5 occurs, the next state might be B, C, or F (depending on which was active before)
- The next state is not D, or E, even if they were previously active, because H does not remember it.



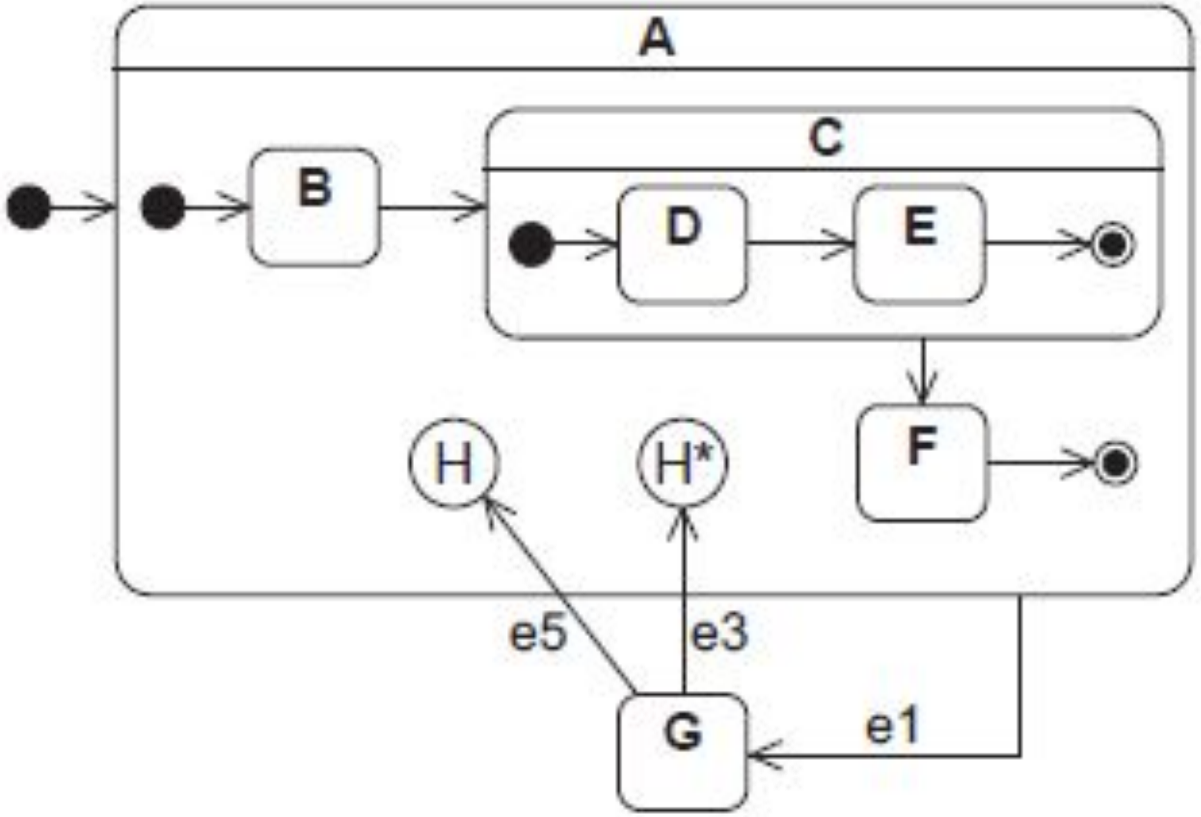
H* is called deep history state

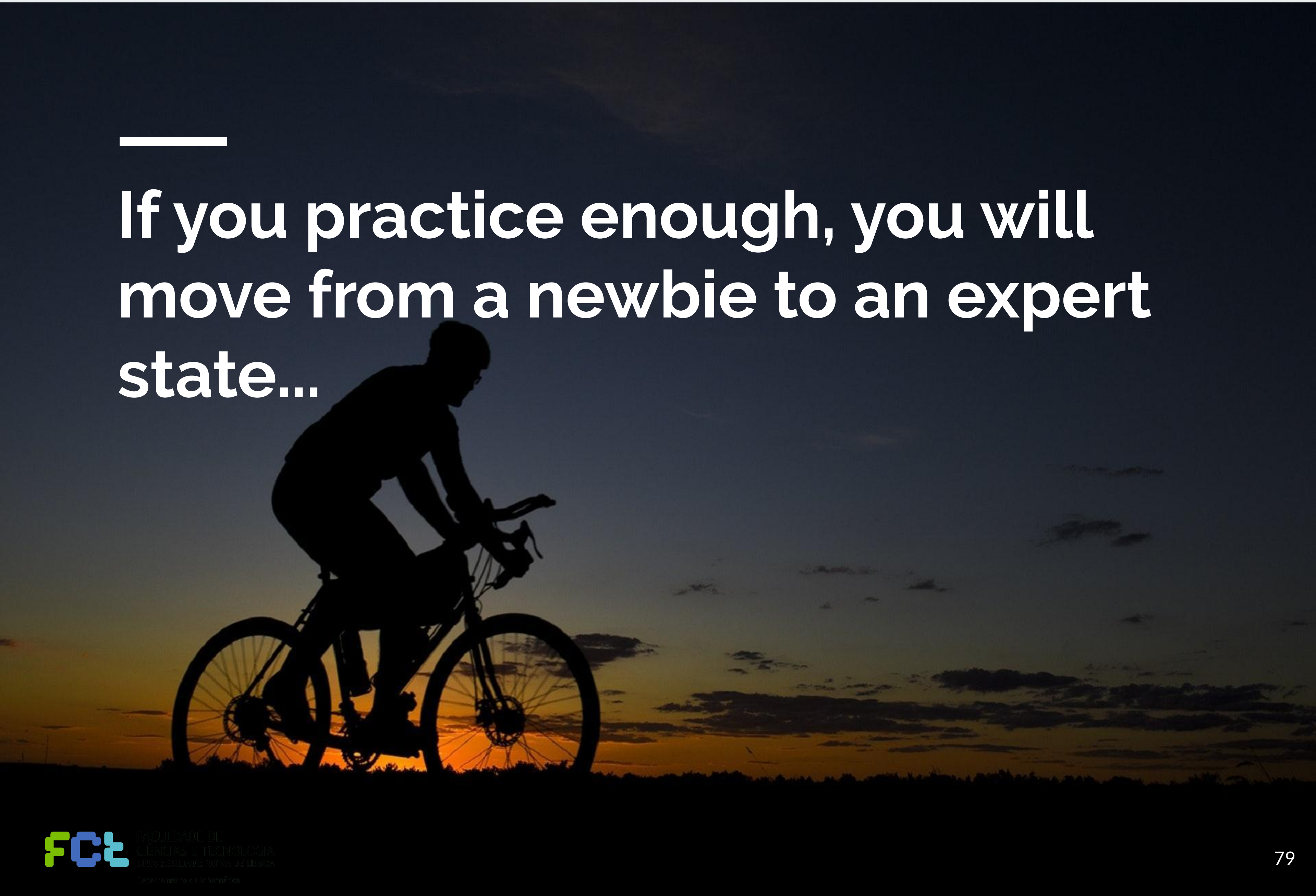
- H* remembers and restores the previously active state at any depth
- Assuming that E is active and event e1 occurs, G is the next active state
- If now e3 occurs, C and more concretely E becomes the next active state



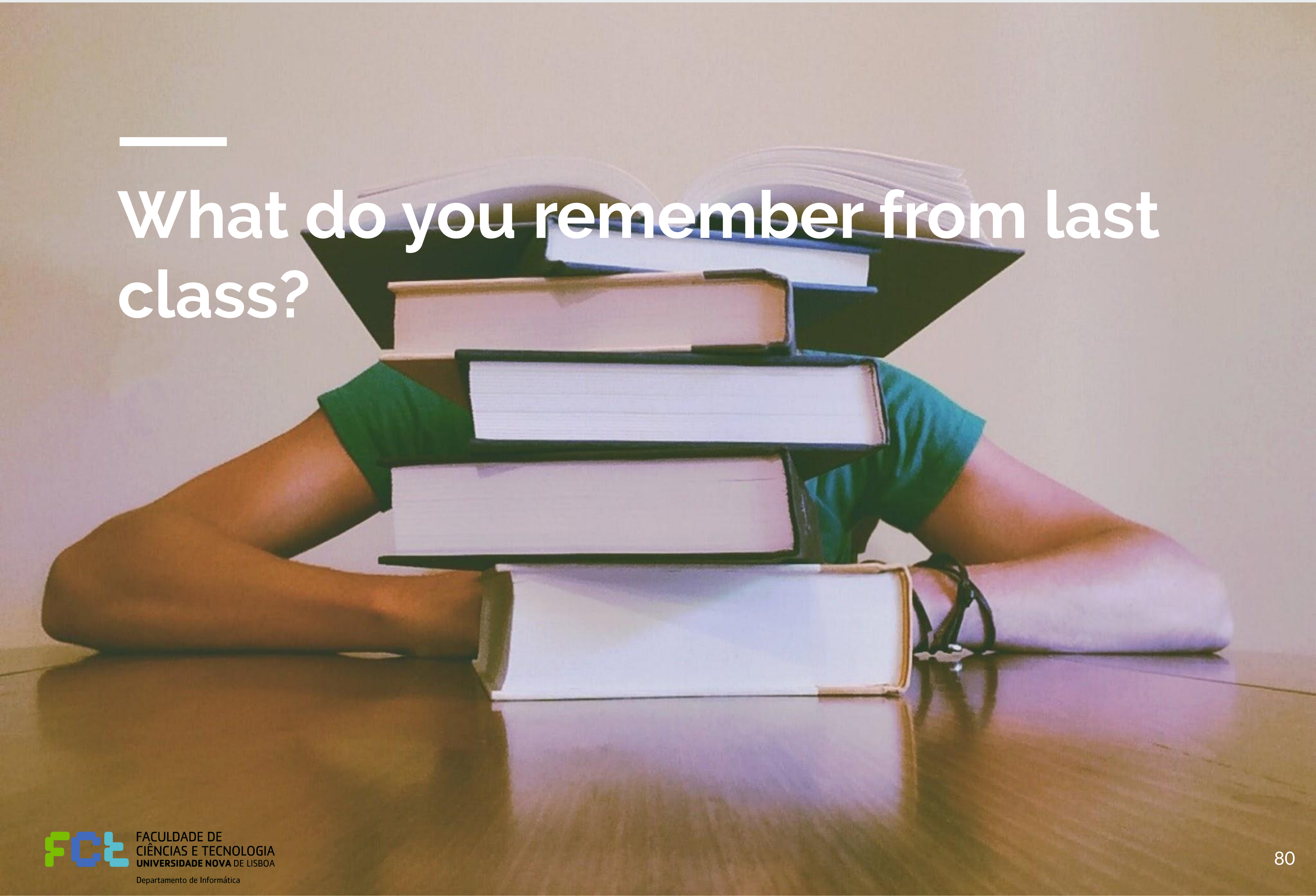
H is called a shallow history state

- It remembers and restores the previously active state at the same nesting depth
- Assuming that G is active and event e5 occurs, the next state might be B, C, or F (depending on which was active before)
- The next state is not D, or E, even if they were previously active, because H does not remember it.





If you practice enough, you will move from a newbie to an expert state...



What do you remember from last class?

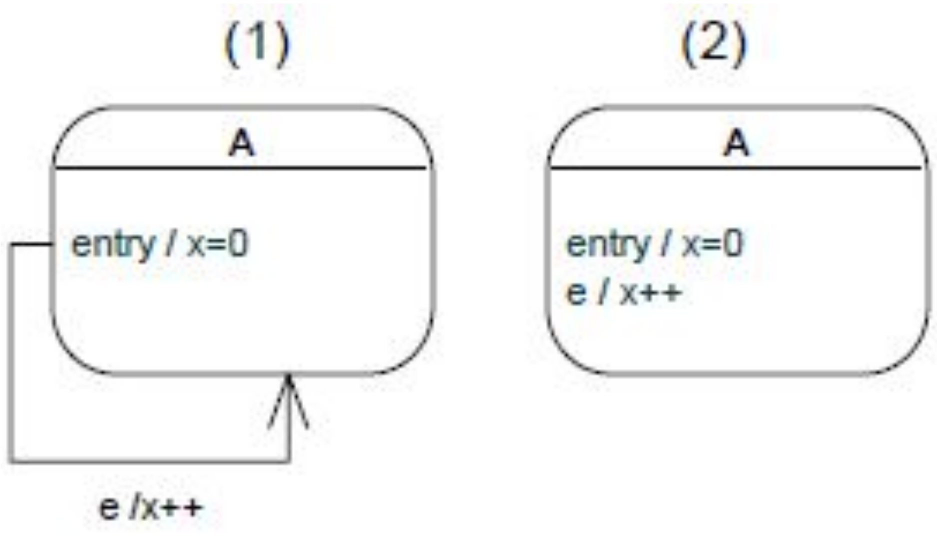
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

I will help you in the first one…

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

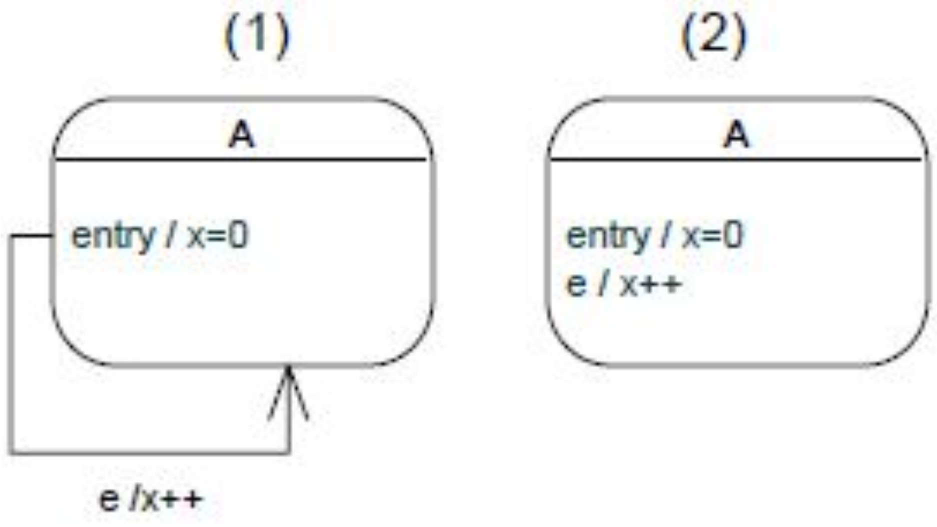
Which of the following statements about figures (1) and (2) are true?

- The two images are not equivalent
- The two images are equivalent
- Internal transitions behave like normal transitions, except that they do not cause a change of state



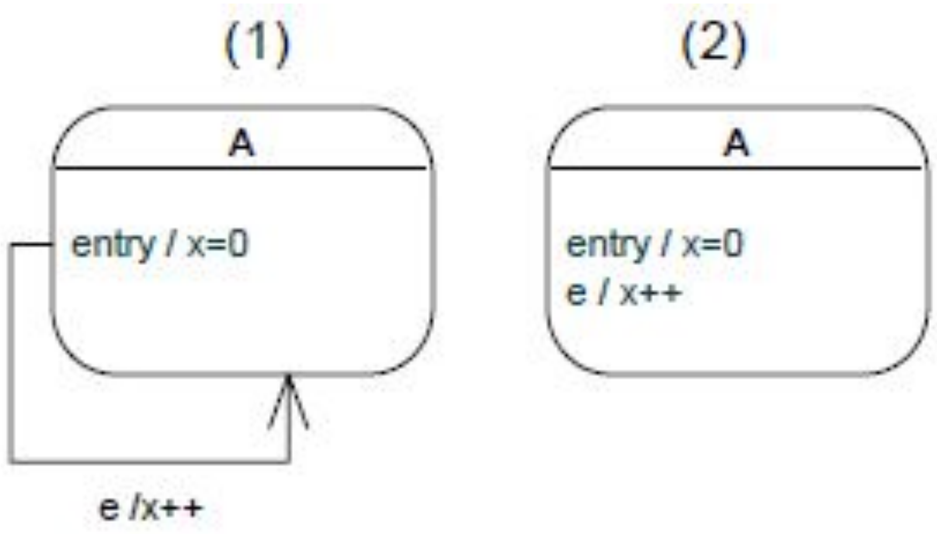
Which of the following statements about figures (1) and (2) are true?

- The two images are not equivalent
- The two images are equivalent
- Internal transitions behave like normal transitions, except that they do not cause a change of state



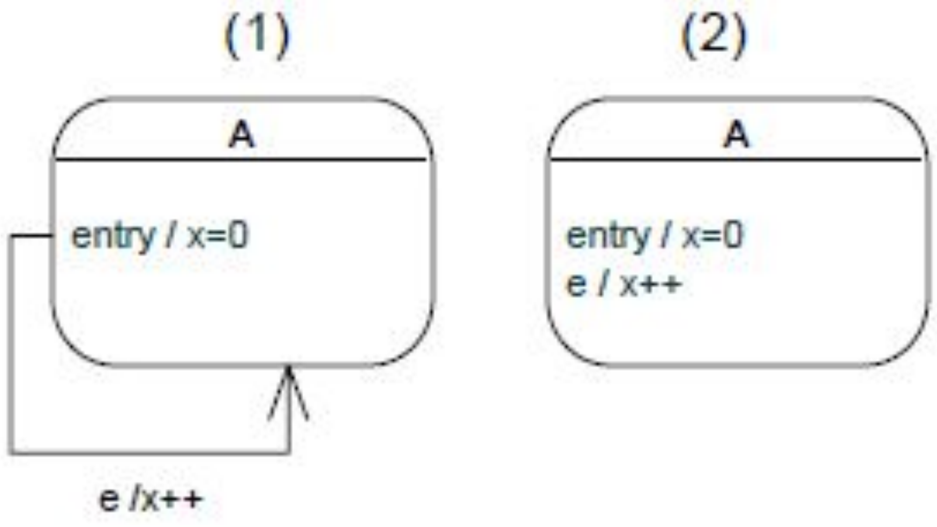
The two images are not equivalent

- In image (1) the entry-activity is executed every time **e** occurs
  - (after x++ is executed and the state is entered again)
- In image (2) the entry-activity is only executed once



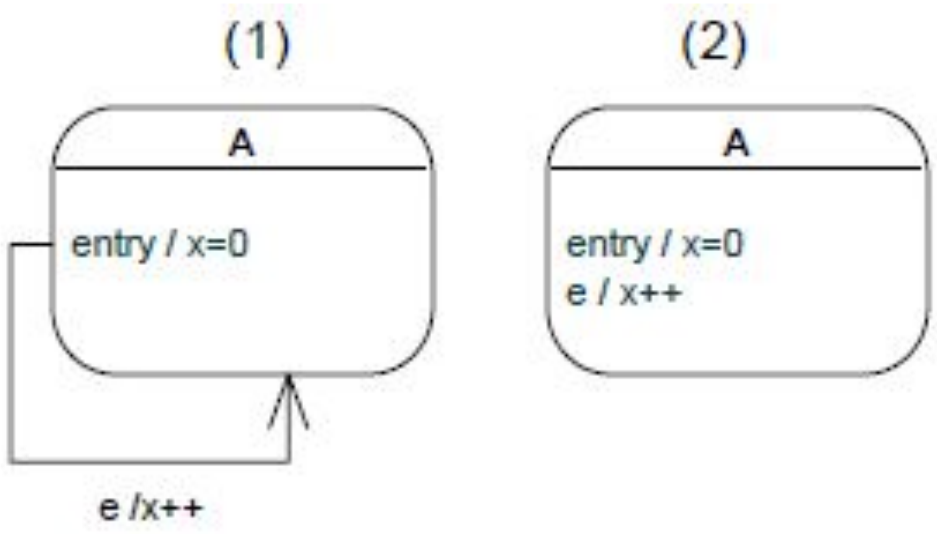
Which of the following statements about figures (1) and (2) are true?

- The two images are not equivalent
- The two images are equivalent
- Internal transitions behave like normal transitions, except that they do not cause a change of state



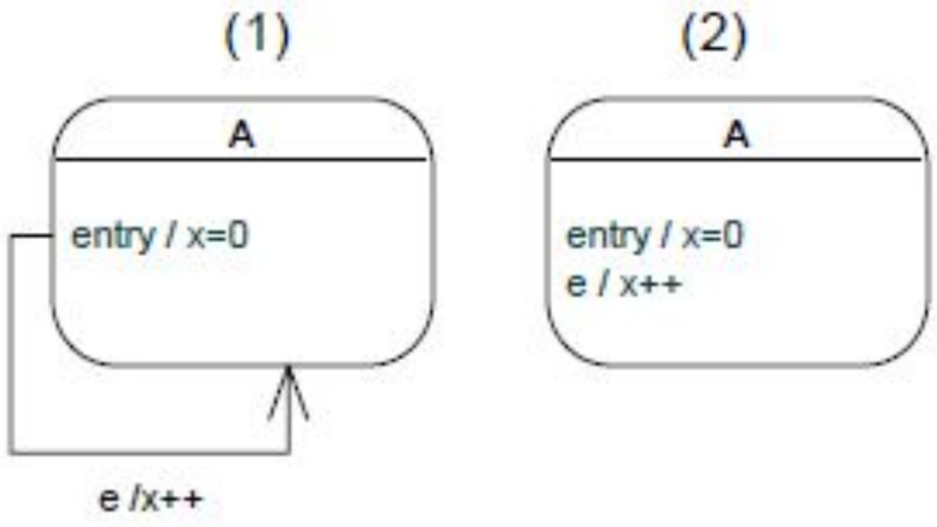
The two images are equivalent

- No, they are not equivalent!
- In image (1) the entry-activity is executed every time e occurs
  - (after x++ is executed and the state is entered again)
- In image (2) the entry-activity is only executed once



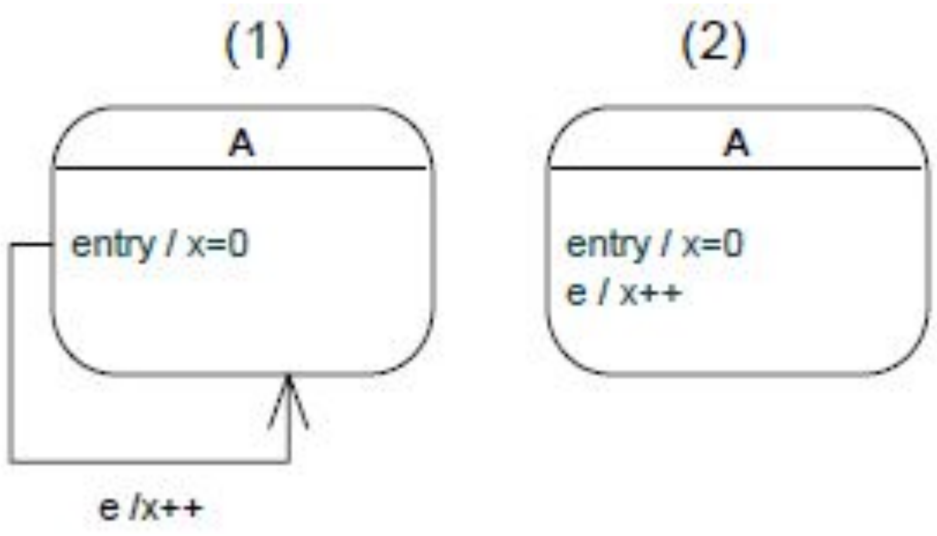
Which of the following statements about figures (1) and (2) are true?

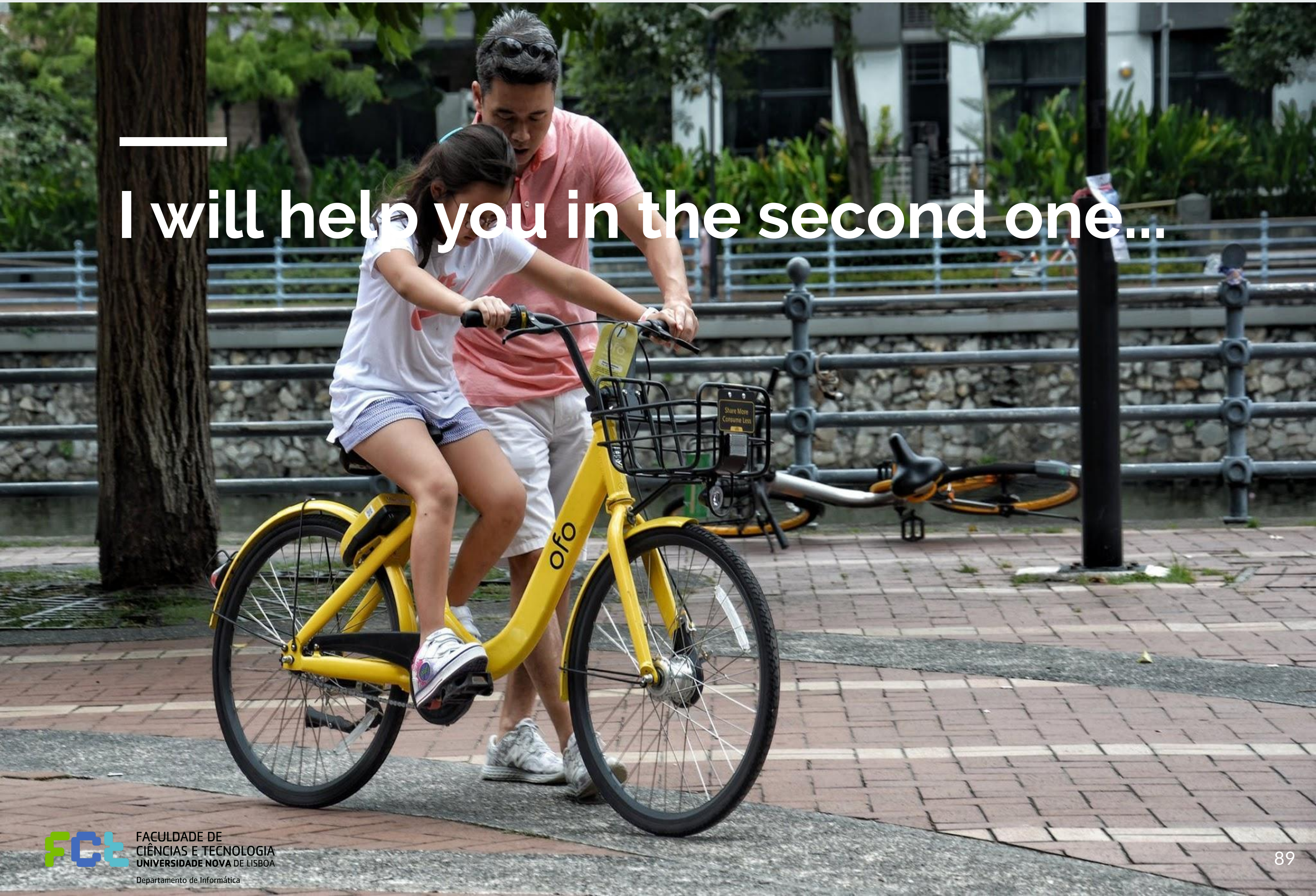
- The two images are not equivalent
- The two images are equivalent
- Internal transitions behave like normal transitions, except that they do not cause a change of state



Internal transitions behave like normal transitions except that they do not cause a change of state.

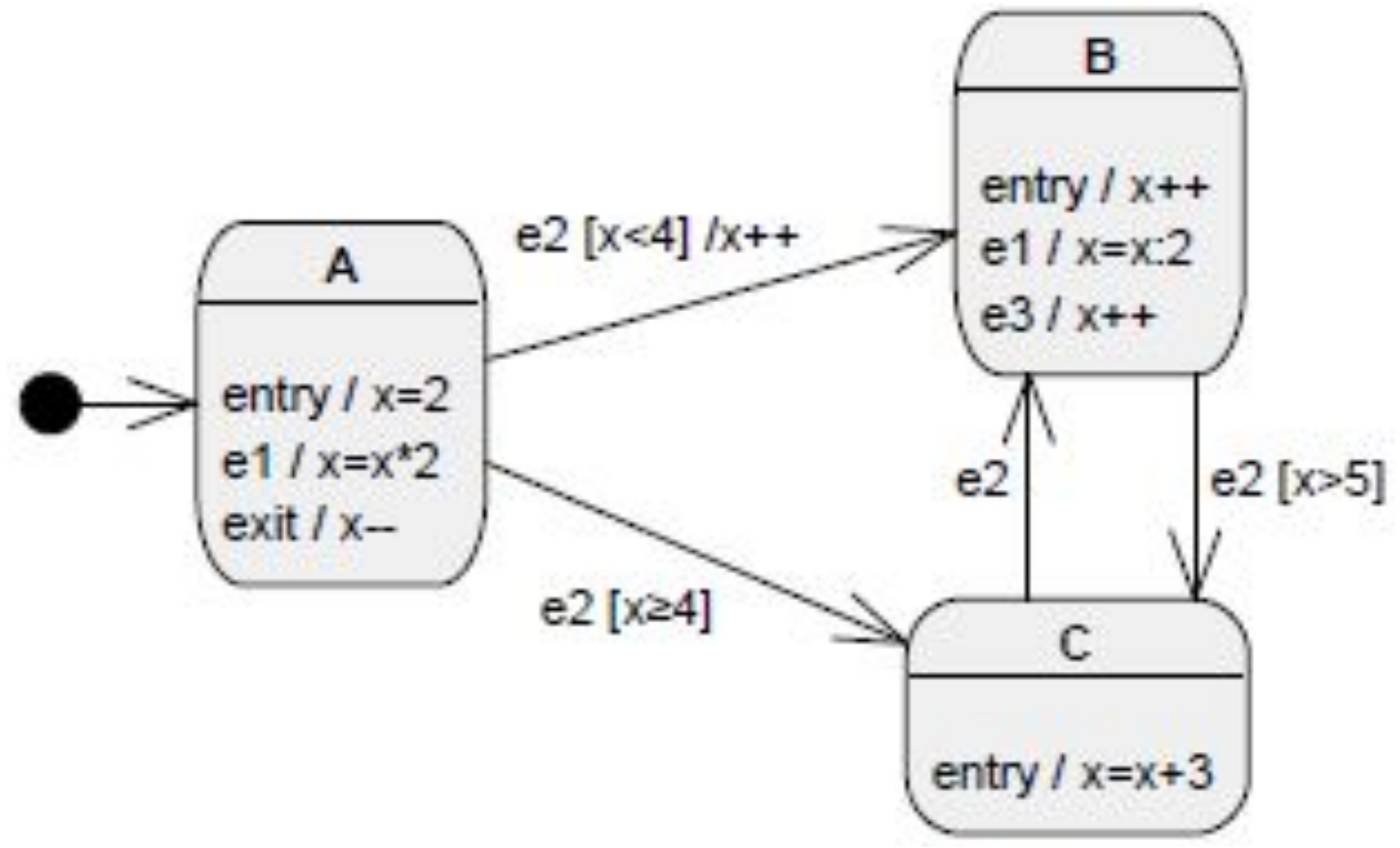
- This is correct
- Note how the transition triggered by event **e** does not change the state from A to some other state
- Note also how the x value update is exactly the same (x++) in both diagrams





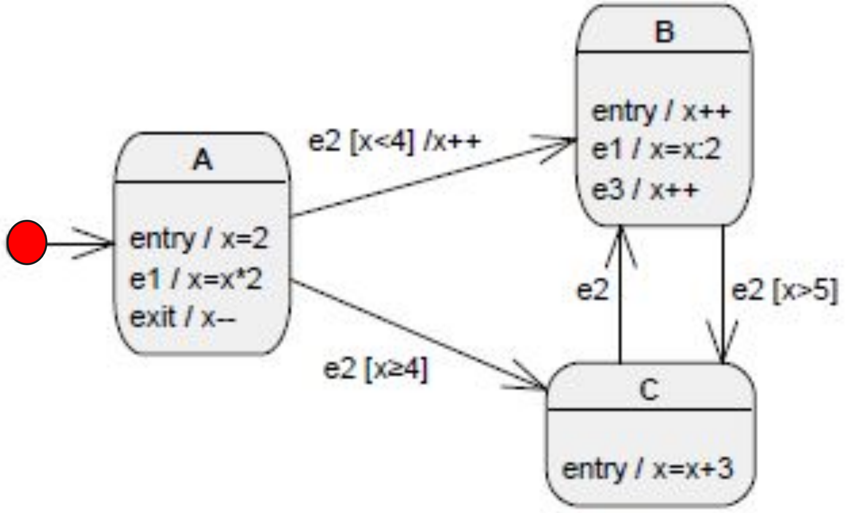
I will help you in the second one…

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



Create a state chart to figure it out! Start on the start node.

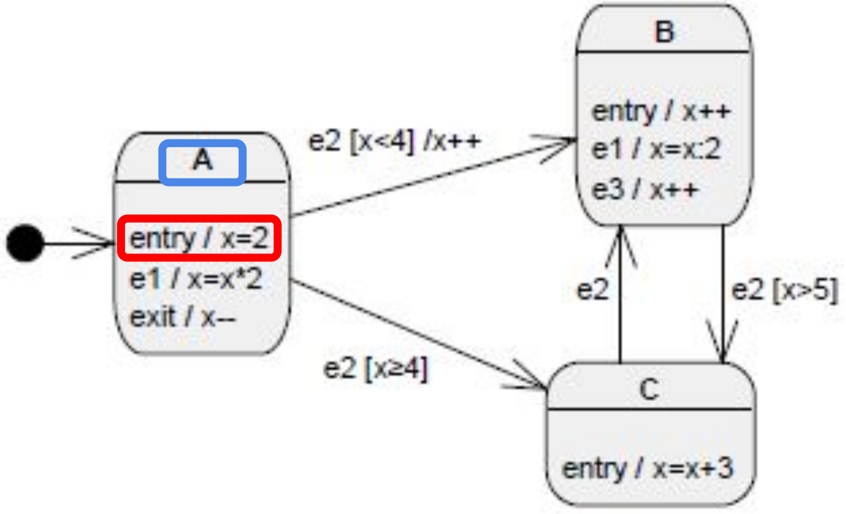
event	state	comment	x
		start	



Event chain: **e1 e2 e2 e2**

You immediately enter state A, where x receives 2 as its new value

event	state	comment	x
		start	
	A	entry of A	2

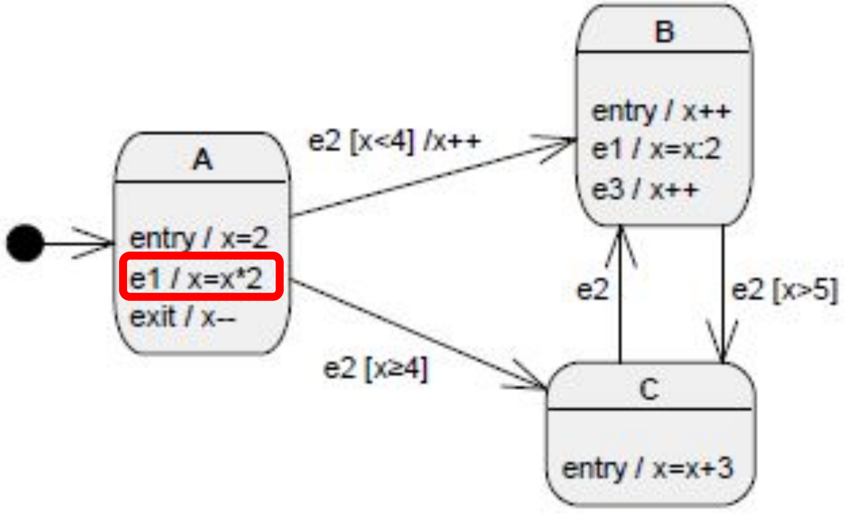


entry / x = 2

Event chain: **e1 e2 e2 e2**

Now, event e1 occurs. As you can see, this is an internal event, so we remain in state A

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4



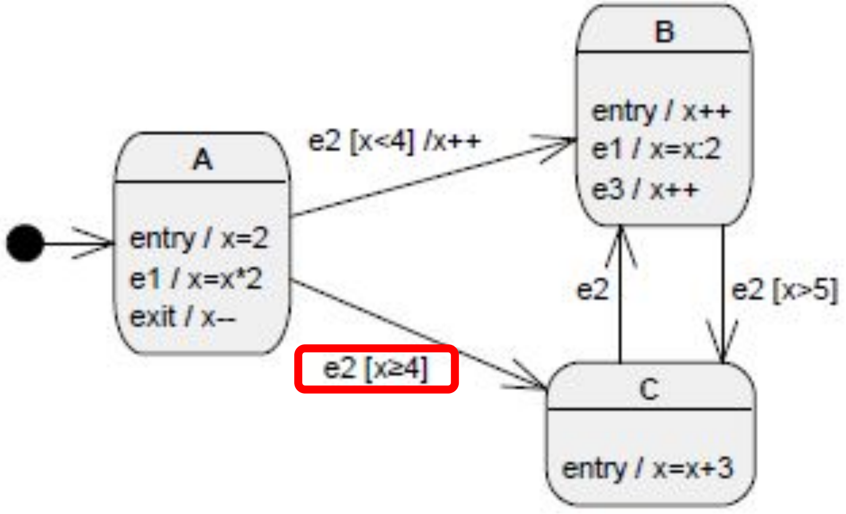
e1 / x = x * 2

Event chain: e1 e2 e2 e2

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

Now, event e2 occurs. Evaluate the guard condition and choose which transition occurs.

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4
e2			

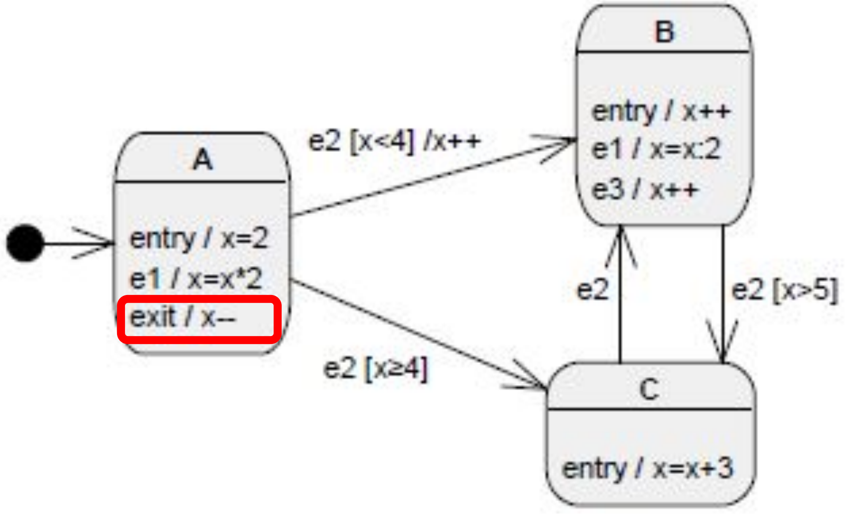


e2 [x >= 4]

Event chain: e1 e2 e2 e2

Compose the exit action with any actions in the transition (in this case, none).

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4
e2		exit of A	3

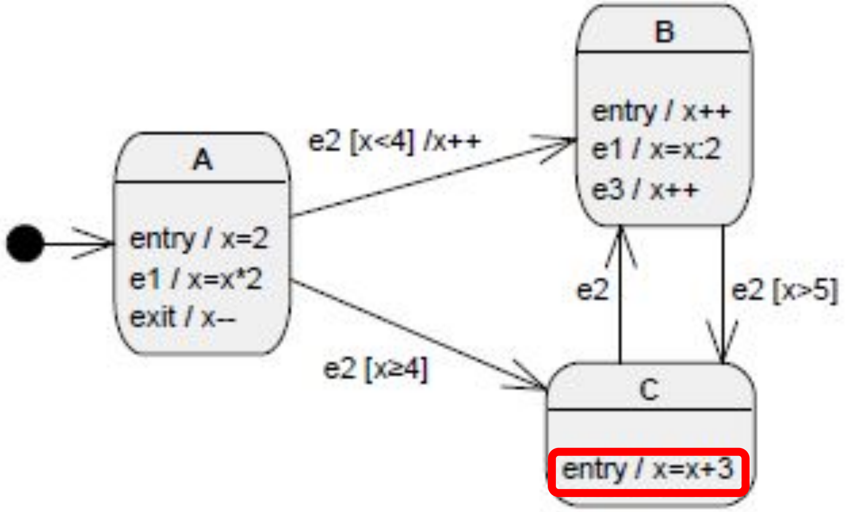


e2 [x >= 4]
exit / x--

Event chain: e1 e2 e2 e2

And now, compose the entry action of state C

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4
e2 	 C	exit of A entry of C	3 6

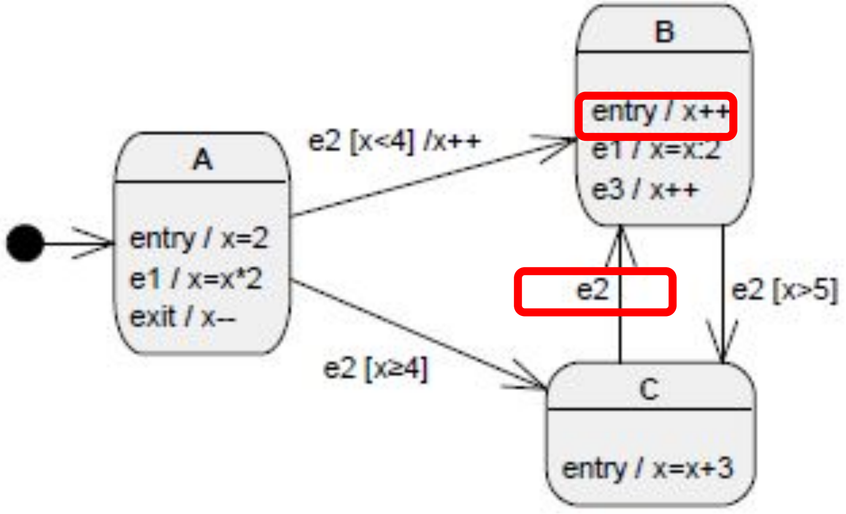


e2 [x >= 4]
exit / x--
entry / x = x + 3

Event chain: e1 e2 e2 e2

When event e2 occurs, transition to state B and execute the entry action

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4
e2 	 C	exit of A entry of C	3 6
e2	B	entry of B	7

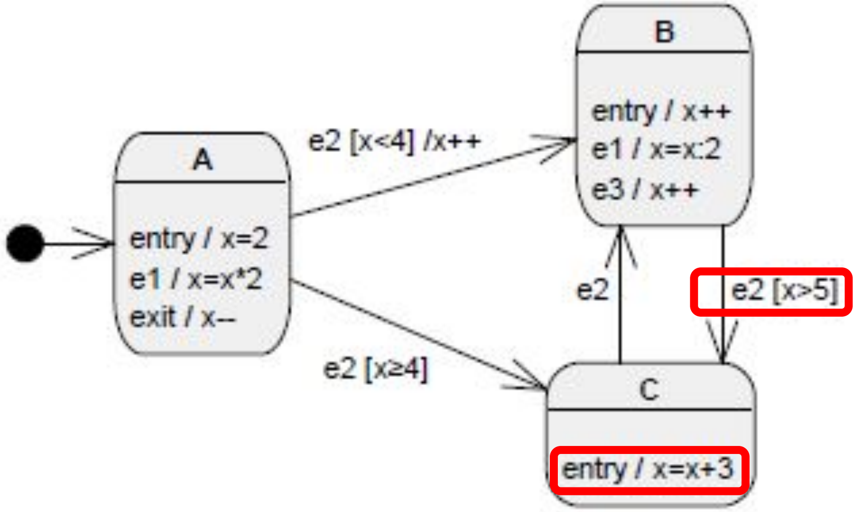


e2
entry / x++

Event chain: e1 e2 e2 e2

When event e2 occurs, transition to state B and execute the entry action

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4
e2	 C	exit of A entry of C	3 6
e2	B	entry of B	7
e2	C	entry of C	10

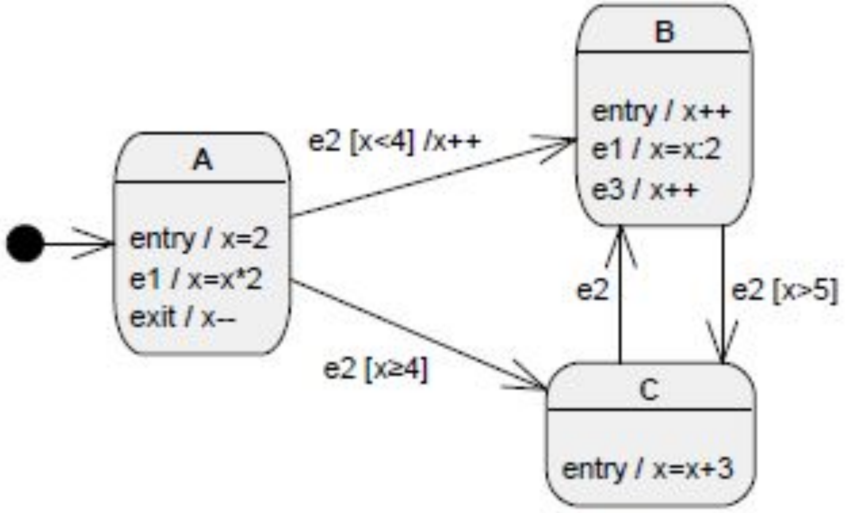


e2 [x > 5]
entry / x = x + 3

Event chain: e1 e2 e2 **e2**

What is the value of x after the occurrence of the event chain e1 e2 e2 e2 ? 10

event	state	comment	x
		start	
	A	entry of A	2
e1	A	x=x*2 in A	4
e2	C	exit of A entry of C	3 6
e2	B	entry of B	7
e2	C	entry of C	10



Event chain: e1 e2 e2 e2

Your turn...

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

In an orthogonal state, ...

- exactly one substate is active.
- it is possible that no substate is active.
- at least one substate in each region is active.

In an orthogonal state, ...

- Exactly one substate is active
    - No. Why just one? They execute concurrently!
- It is possible that no substate is active
    - Nope. If no substate is active, then the orthogonal state is not active either. And so, we are not in it.
- at least one substate in each region is active
    - There can be more, but at least one must be active.

Which of the following is true about SM diagrams?

- Do-activities in states are atomic, they can't be aborted by any event
- The final state in a state machine diagram is a pseudostate
- Activities can be executed within states and during transitions
- A transition can have an event trigger, a guard condition and a state
- Events trigger states

do-activities within states are atomic, they can't be aborted by any event

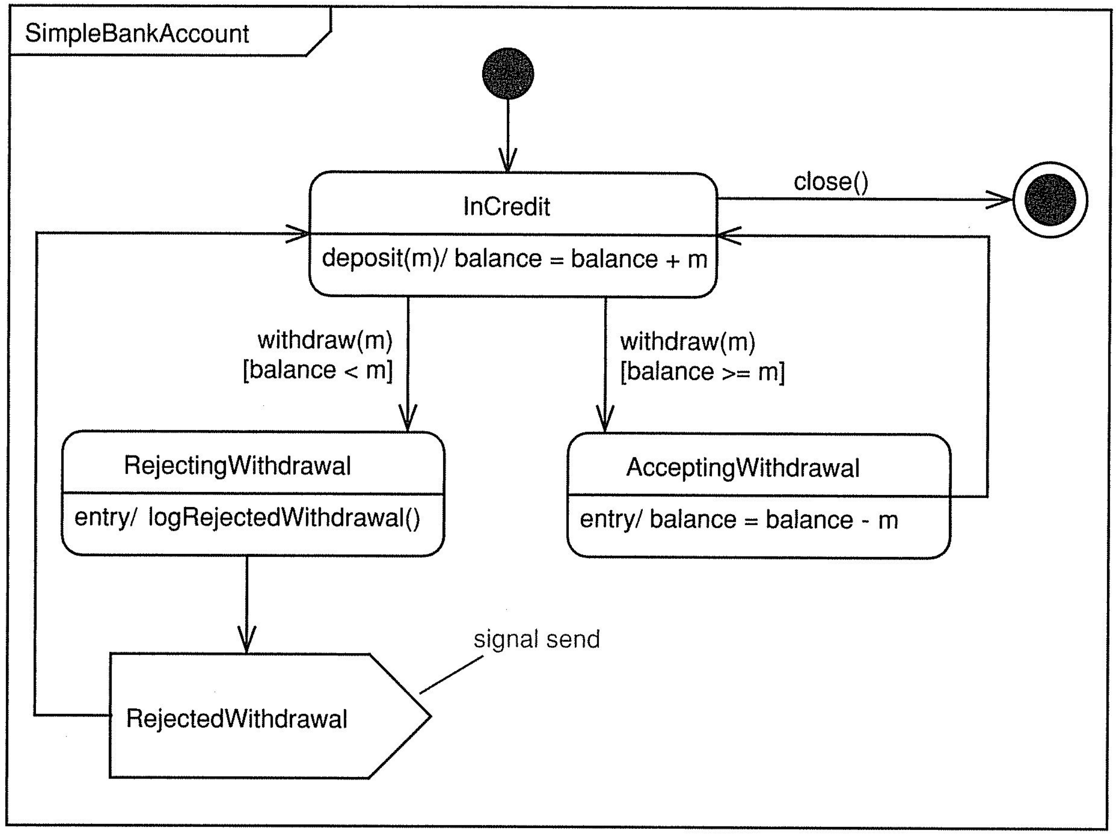
- The do-activity starts as soon as the state is entered
- It continues until either the activity is completed or the state is exited
- An event triggering a transition that leaves the current state aborts the do-activity.

The final state in a state machine diagram is a pseudostate

- The **final state** is a special kind of state signifying that the enclosing region is completed.
- The **final state** is not a pseudostate.
  - It may be active for a period of time until all other orthogonal regions of the composite state are completed.

Activities can be executed within states and during transitions

- Remember this example? There they are...



A transition can have an event trigger, a guard condition and a state

- A transition can have
  - an event trigger,
  - a guard condition and
  - an action [ooopsy :-)]

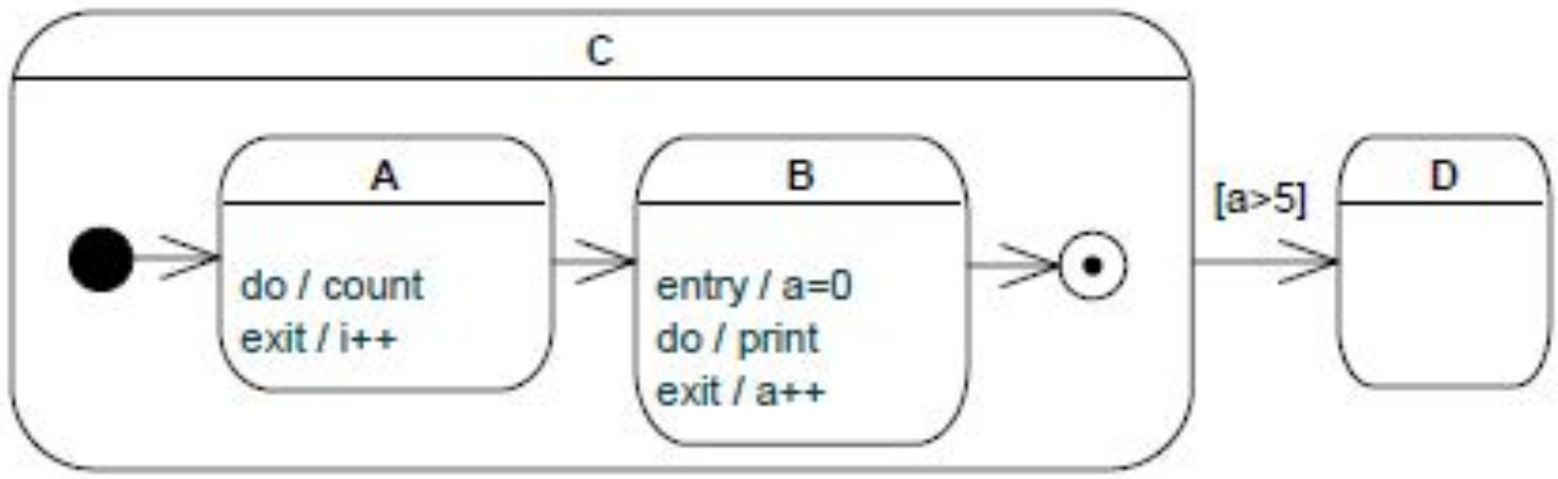
Events trigger states

- No, they don’t
- Events trigger transitions

Which of the following is true about SM diagrams? (wrap-up)

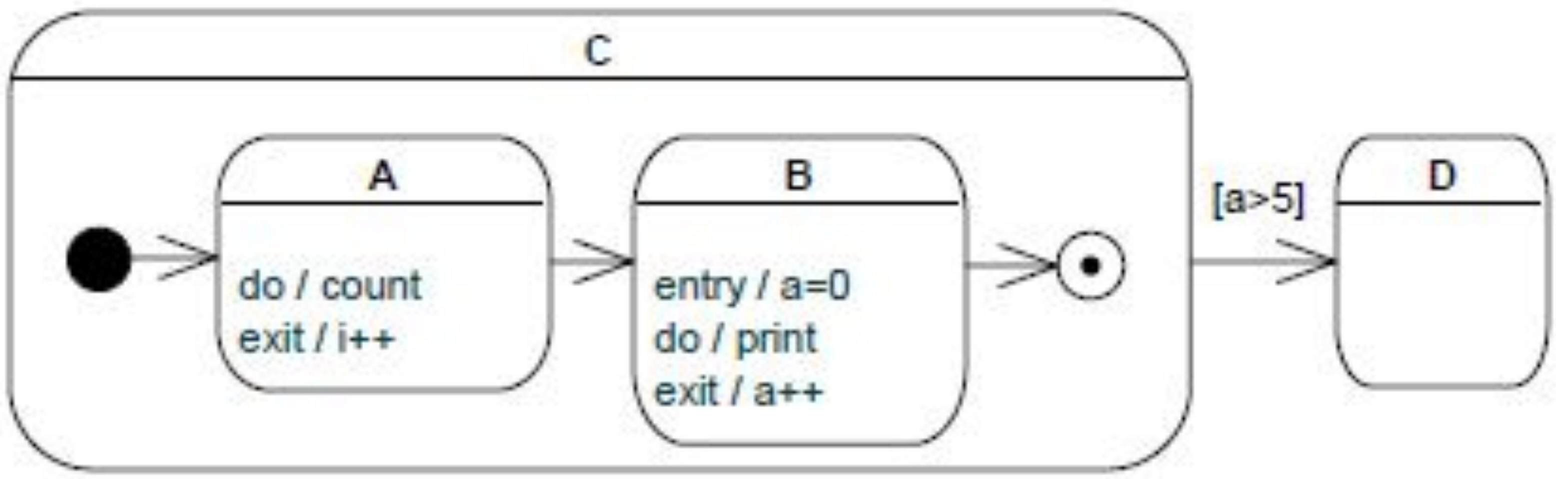
- Do-activities in states are atomic, they can't be aborted by any event
- The final state in a state machine diagram is a pseudostate
- Activities can be executed within states and during transitions
- A transition can have an event trigger, a guard condition and a state
- Events trigger states

When does the transition to state D take place?



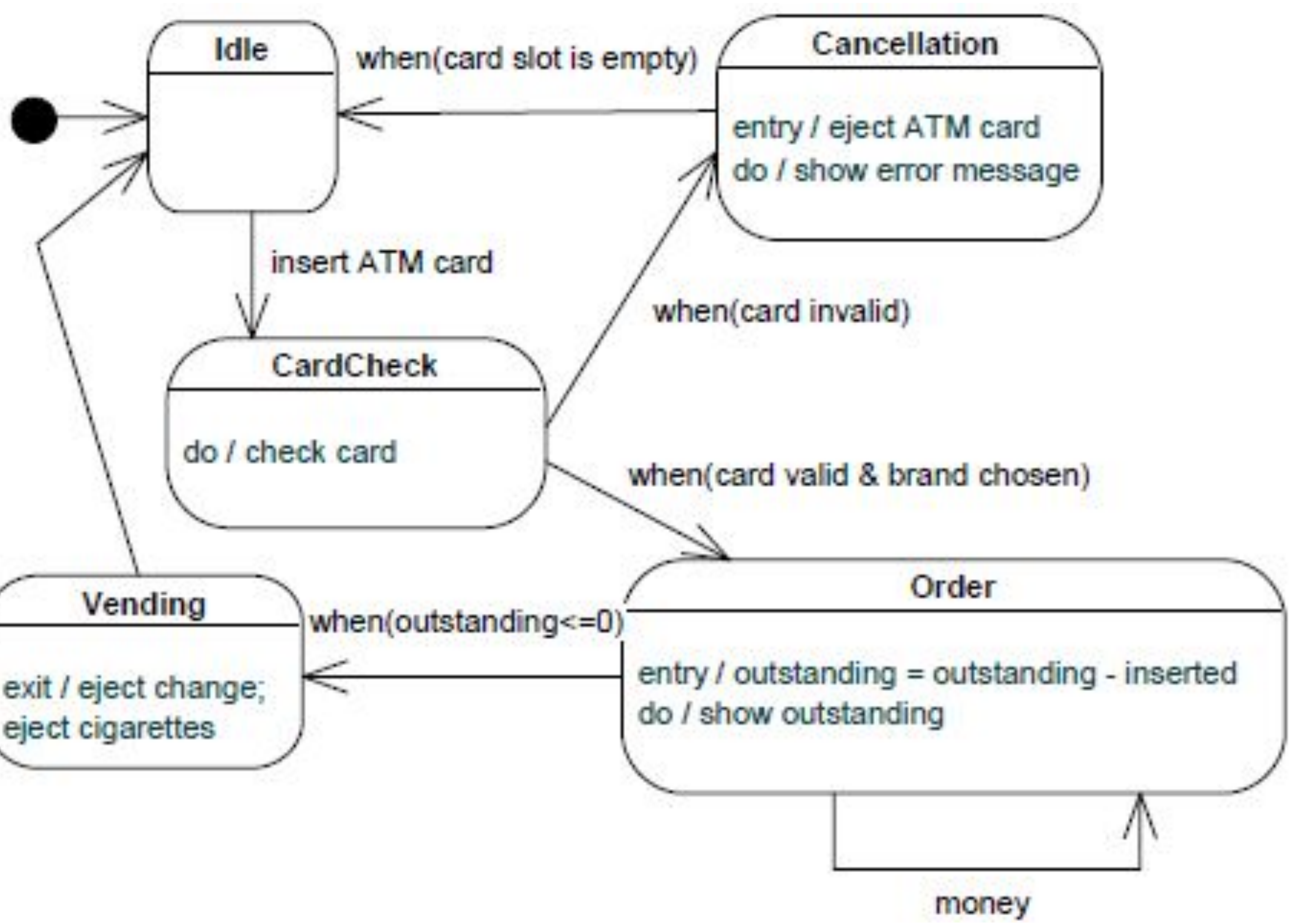
- As soon as the final state in C is reached, the guard condition a>5 is evaluated. The transition to D only happens if a>5 is true at that exact moment.
- As soon as all activities within A are completed and the guard condition a>5 evaluates to true.
- As soon as the guard condition a>5 is evaluated to true.

When does the transition to state D take place?



- As soon as the final state in C is reached, the guard condition a>5 is evaluated. The transition to D only happens if a>5 is true at that exact moment.
- As soon as all activities within A are completed and the guard condition a>5 evaluates to true.
- As soon as the guard condition a>5 is evaluated to true.

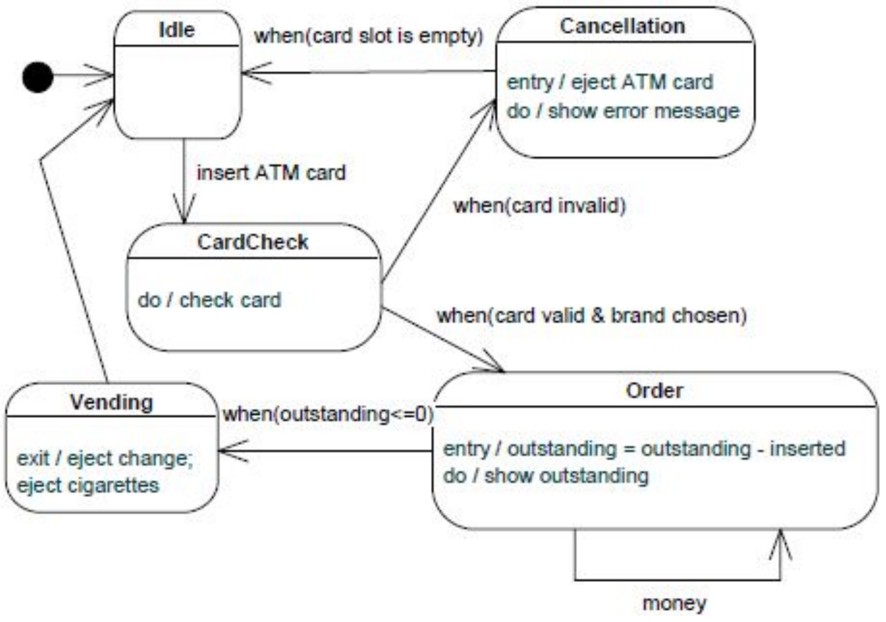
Which of the following are true?



FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA Departamento de Informática

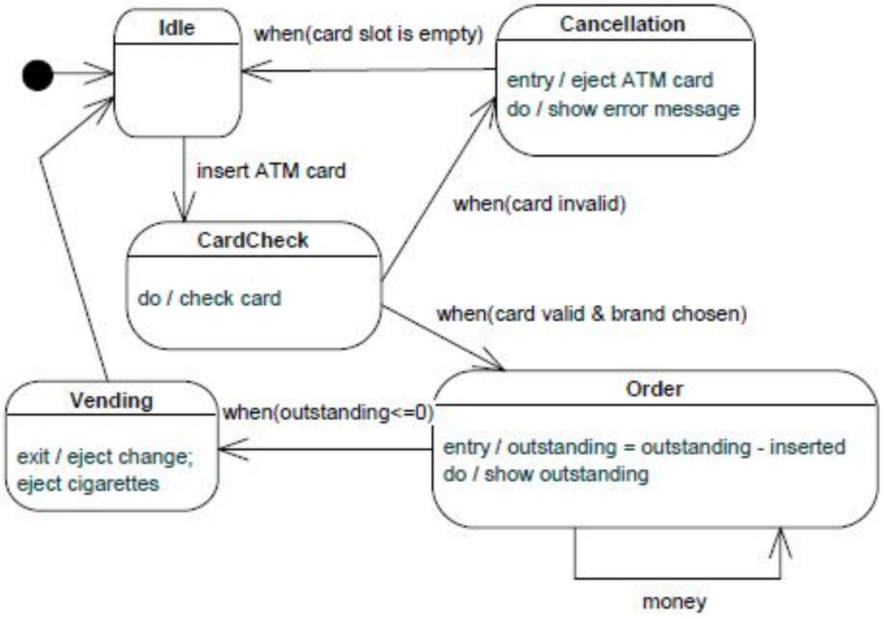
Which of the following are true?

- If CardCheck is the active state and the card is invalid, the machine changes into state Idle.
- If the card is invalid, an error message is shown and the ATM card is ejected. After that a transition to Idle takes place.
- As soon as a customer has inserted an ATM card into the machine, the active state changes from Idle to CardCheck.
- If the card is valid and a cigarette brand has been chosen, but the customer does not insert any money, the card is ejected and an error message is shown.



If CardCheck is the active state and the card is invalid, the machine changes into state Idle

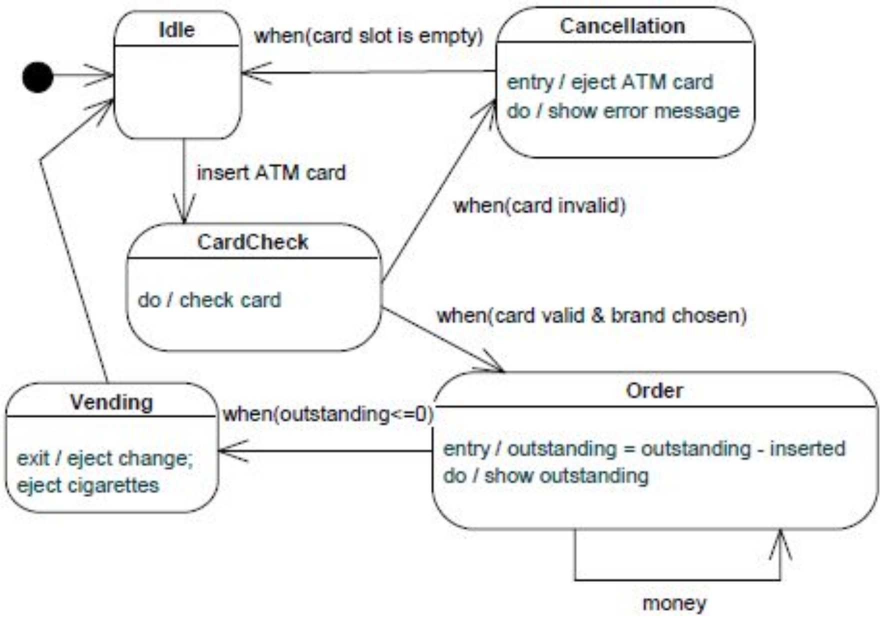
- If CardCheck is the active state and the card is invalid, the machine changes into state Cancellation.



If the card is invalid, an error message is shown and the ATM card is ejected. After that a transition to Idle takes place.

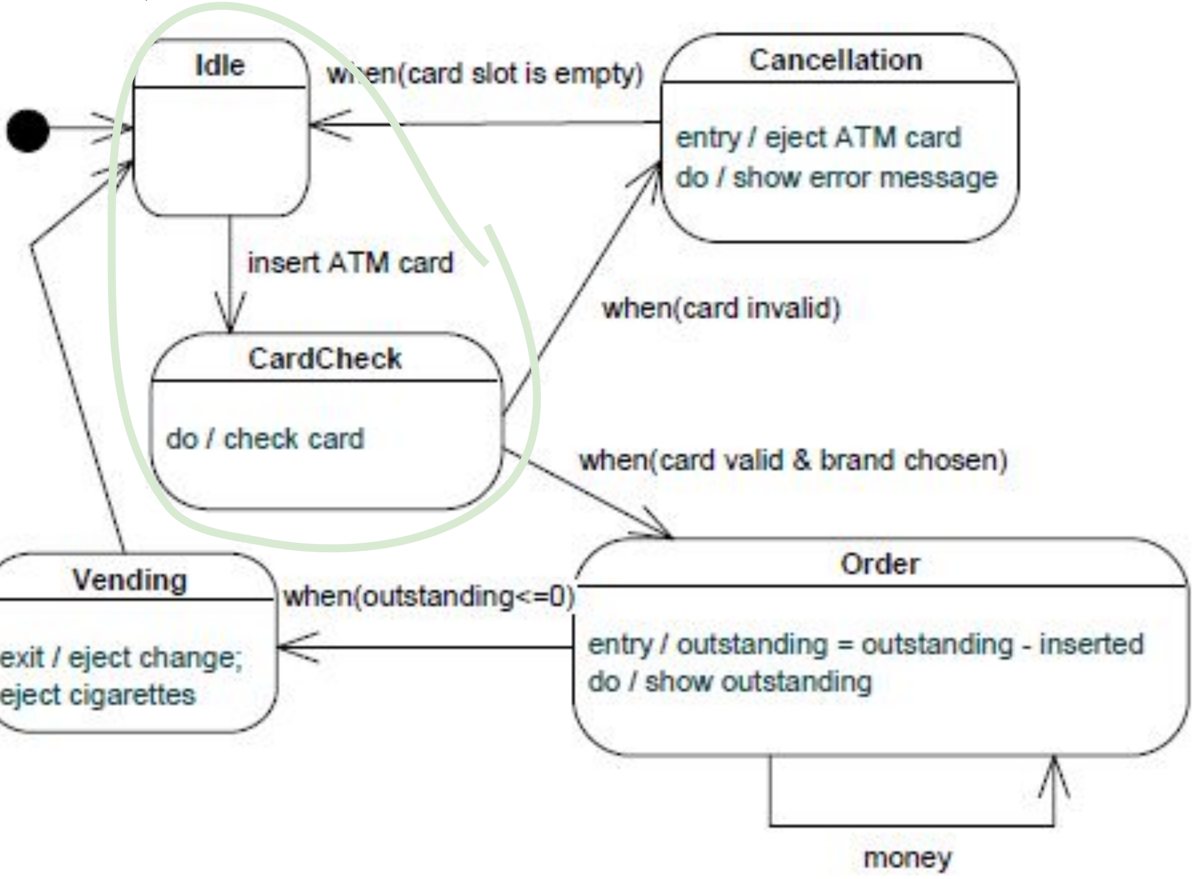
This is not very accurate. Actually:

- If CardCheck is the active state and the card is invalid, the machine changes into state Cancellation.
- On entering Cancellation, the ATM card is ejected, then an error message is displayed.
- The transition to Idle happens as soon as the ATM card was taken from the machine (and therefore the card slot is empty).



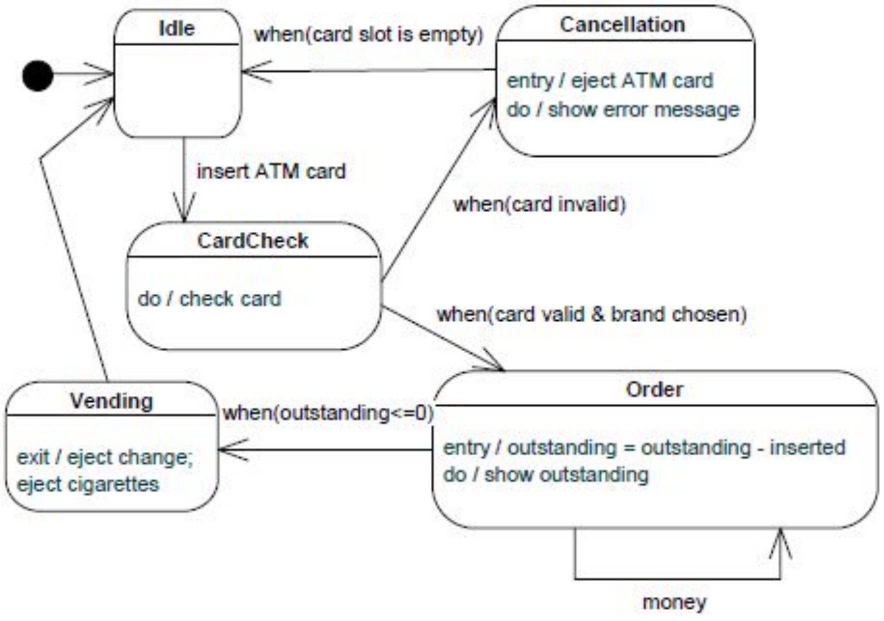
As soon as a customer has inserted an ATM card into the machine, the active state changes from Idle to CardCheck.

- So true… :-)



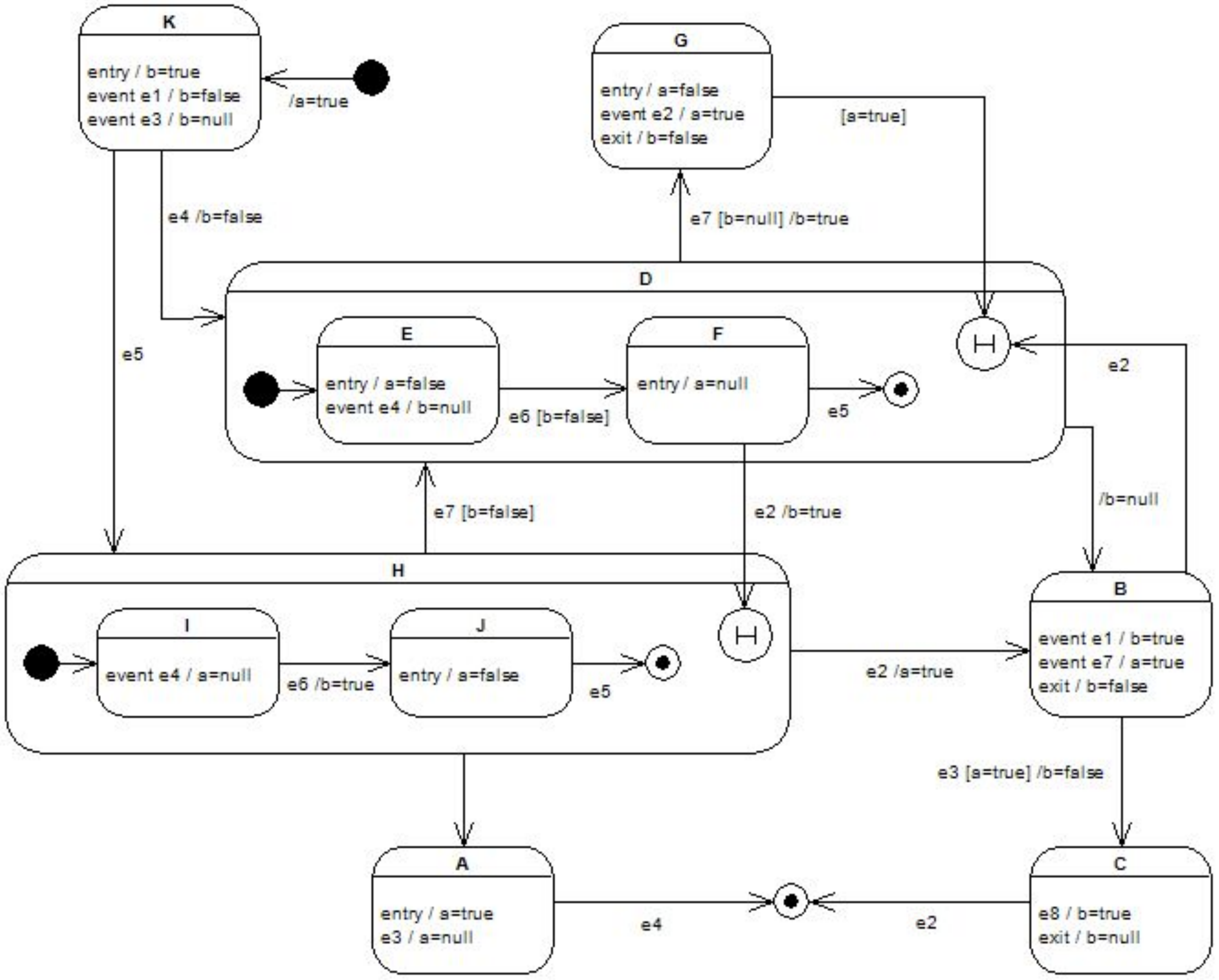
If the card is valid and a cigarette brand has been chosen, but the customer does not insert any money, the card is ejected and an error message is shown.

- The model does not handle this problem.
- If the card is valid and a cigarette brand has been chosen, but the customer does not insert any money, nothing happens.
    - Order stays the active state.



After the occurrence of which event chain(s) is state B the active state?

- e3 e2 e1 e3 e2 e3 e4 e7 e2
- e1 e2 e3 e4 e5 e6 e5 e4 e3
- e1 e3 e4 e4 e6 e2 e5
- e1 e5 e3 e2 e3 e4 e5

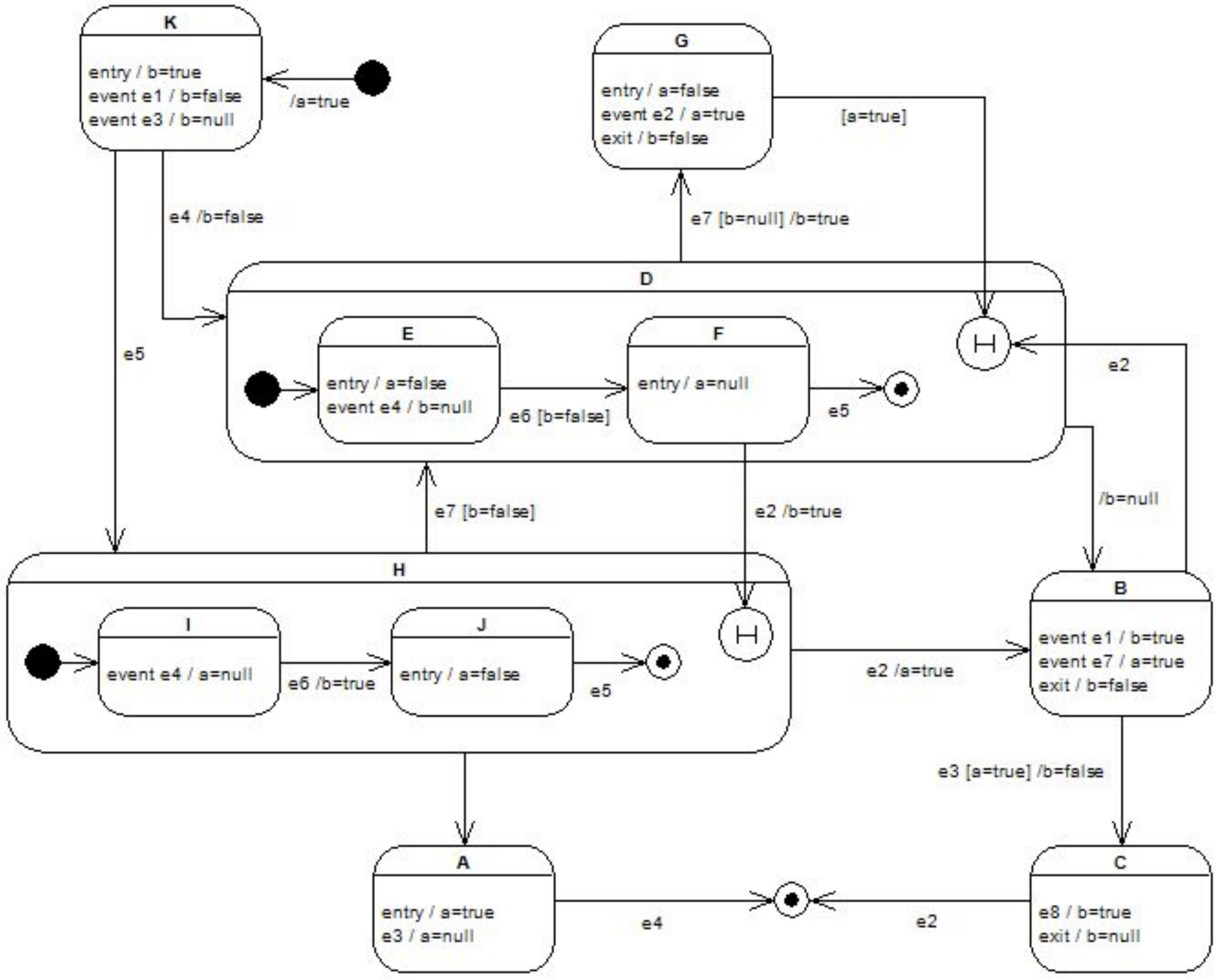


FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

After the occurrence of which event chain(s) is state B the active state?

- e3 e2 e1 e3 e2 e3 e4 e7 e2
- e1 e2 e3 e4 e5 e6 e5 e4 e3
- e1 e3 e4 e4 e6 e2 e5
- e1 e5 e3 e2 e3 e4 e5

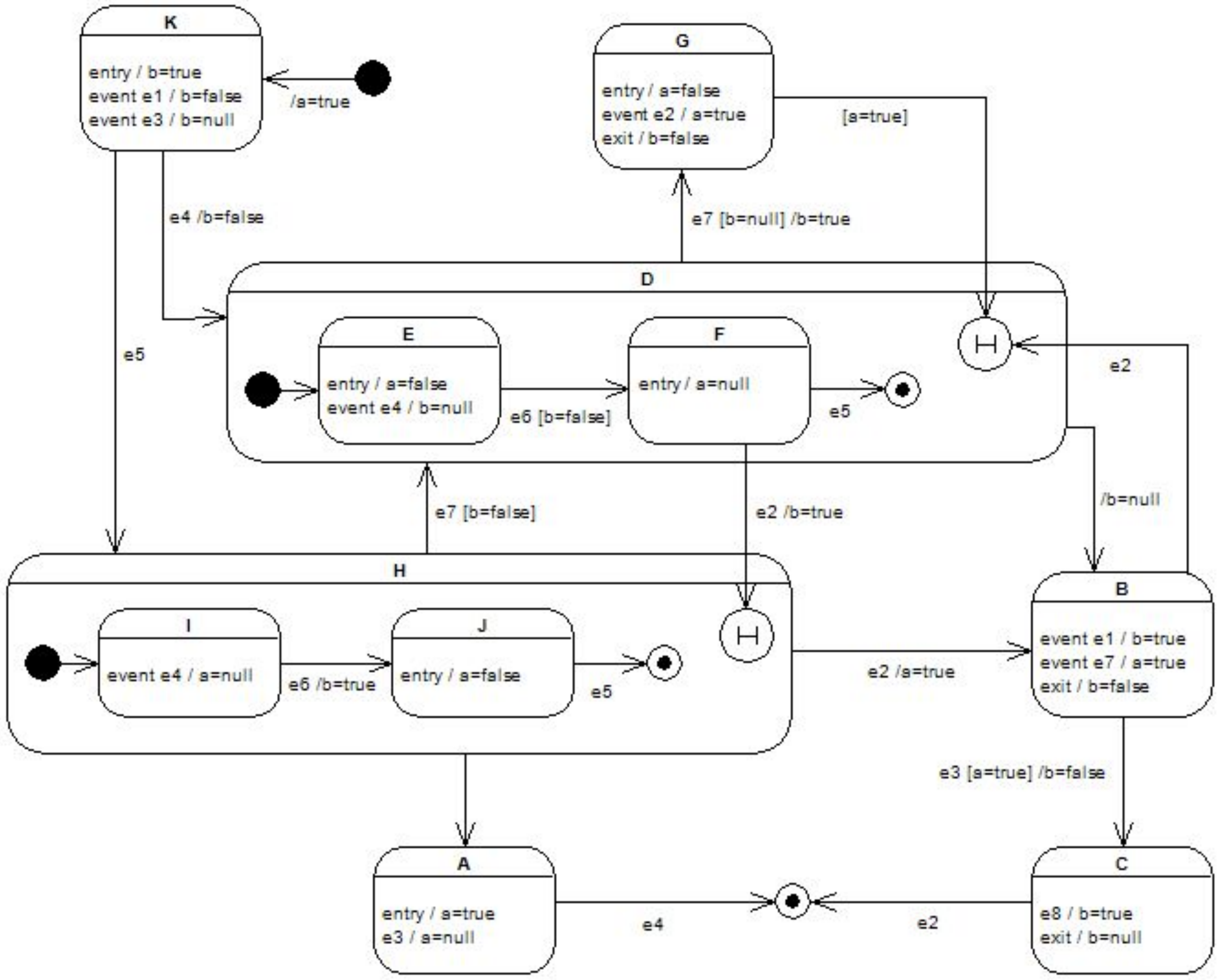
```
State chain: K - E
```



FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

After the occurrence of which event chain(s) is state B the active state?

- e3 e2 e1 e3 e2 e3 e4 e7 e2
- e1 e2 e3 e4 e5 e6 e5 e4 e3
- e1 e3 e4 e4 e6 e2 e5
- e1 e5 e3 e2 e3 e4 e5

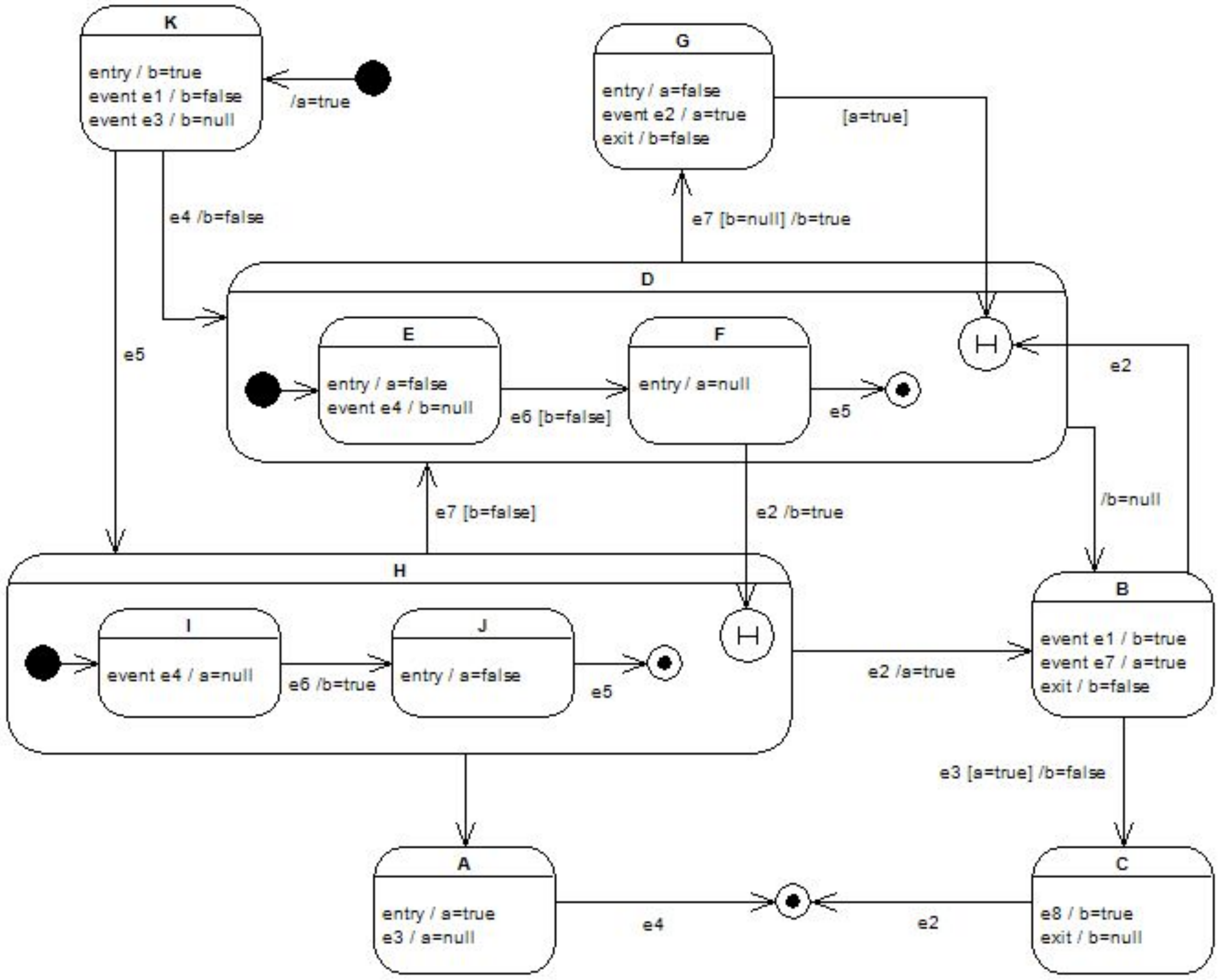


State chain: K - E - F - B

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

After the occurrence of which event chain(s) is state B the active state?

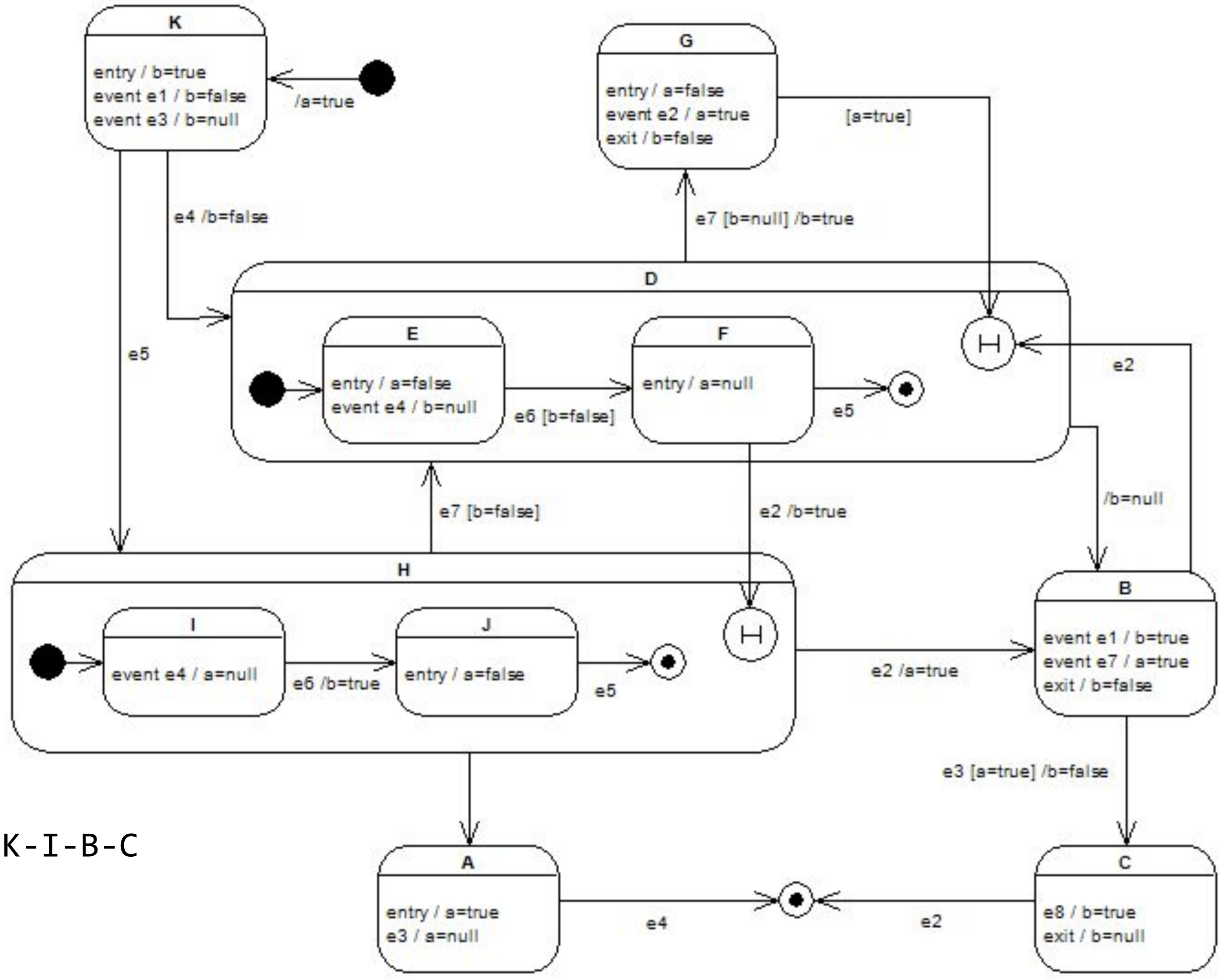
- e3 e2 e1 e3 e2 e3 e4 e7 e2
- e1 e2 e3 e4 e5 e6 e5 e4 e3
- e1 e3 e4 e4 e6 e2 e5
- e1 e5 e3 e2 e3 e4 e5



```
State chain: K - E
```

After the occurrence of which event chain(s) is state B the active state?

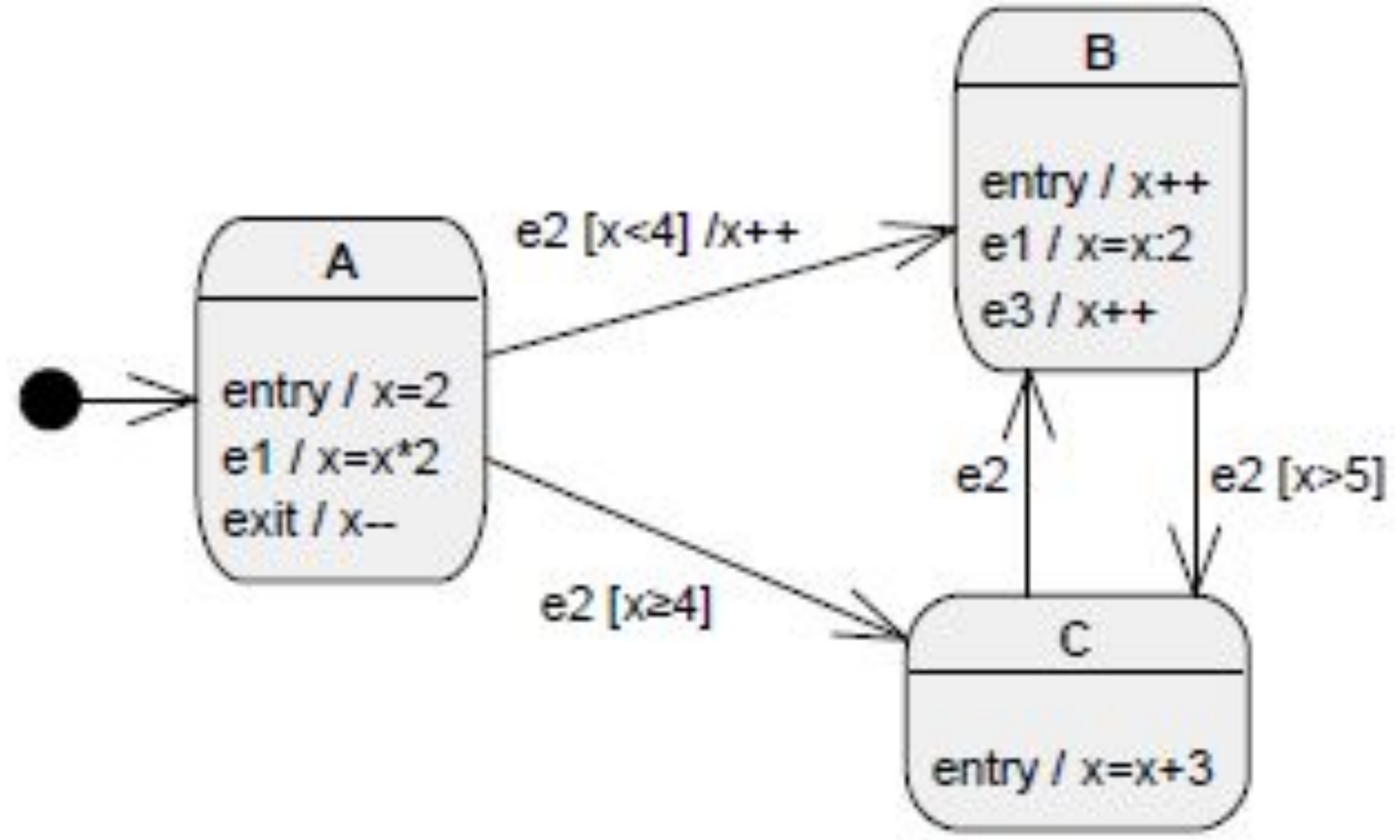
- e3 e2 e1 e3 e2 e3 e4 e7 e2
- e1 e2 e3 e4 e5 e6 e5 e4 e3
- e1 e3 e4 e4 e6 e2 e5
- e1 e5 e3 e2 e3 e4 e5



```
State chain: State chain: K-I-B-C
```

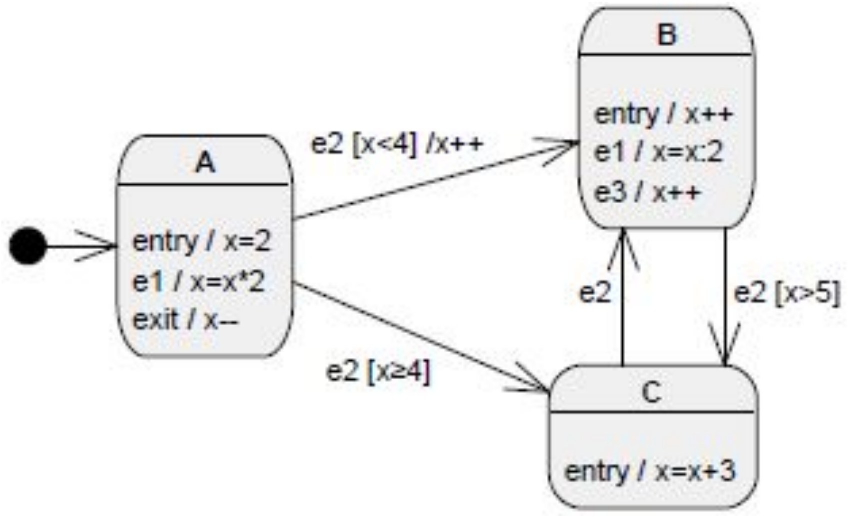
Time to make it on your own!

What is the value of x after the occurrence of the event chain e2 e3 e1 e2 e3 ?

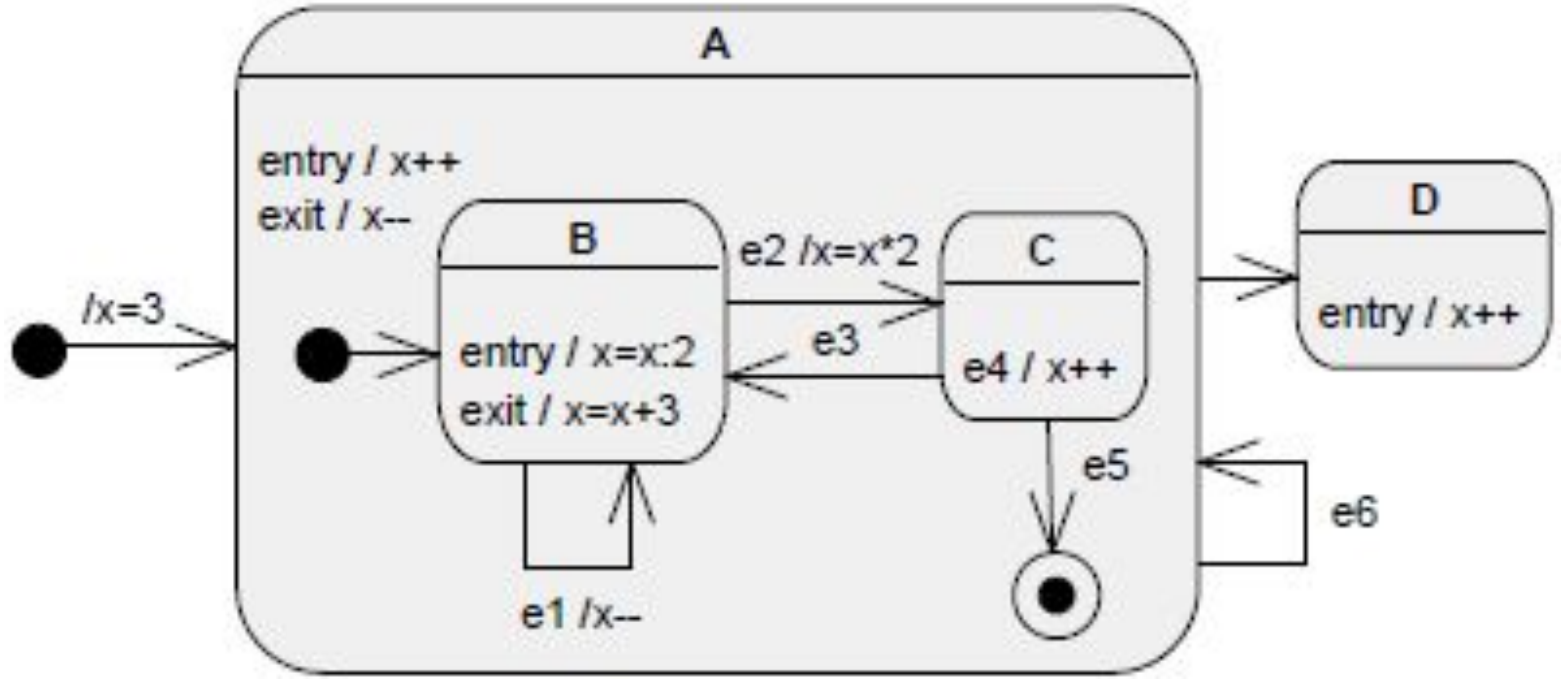


What is the value of x after the occurrence of the event chain e2 e3 e1 e2 e3 ? 3

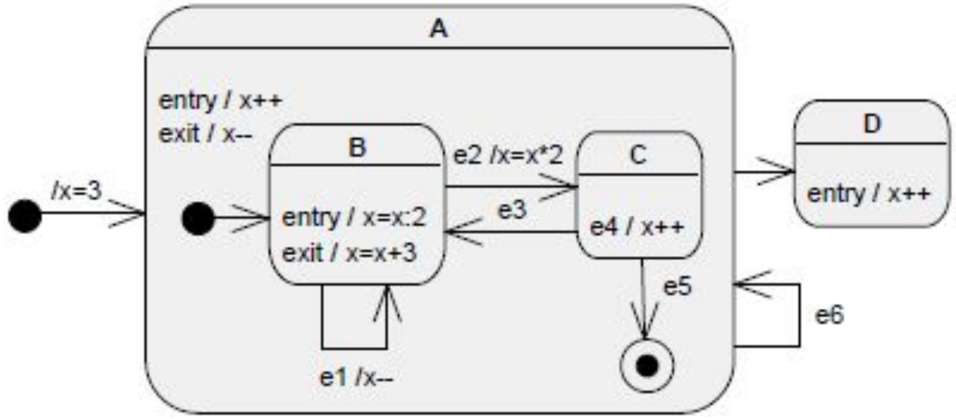
event	state	comment	x
		start	
	A	entry of A	2
e2 	 B	exit of A x++ (Transition) entry of B	1 2 3
e3	B	x++ in B	4
e1	B	x = x:2 in B	2
e2	B	nothing happens	2
e3	B	x++ in B	3



What is the value of x after the occurrence of the event chain e2 e3 e2 e4 e5 ?



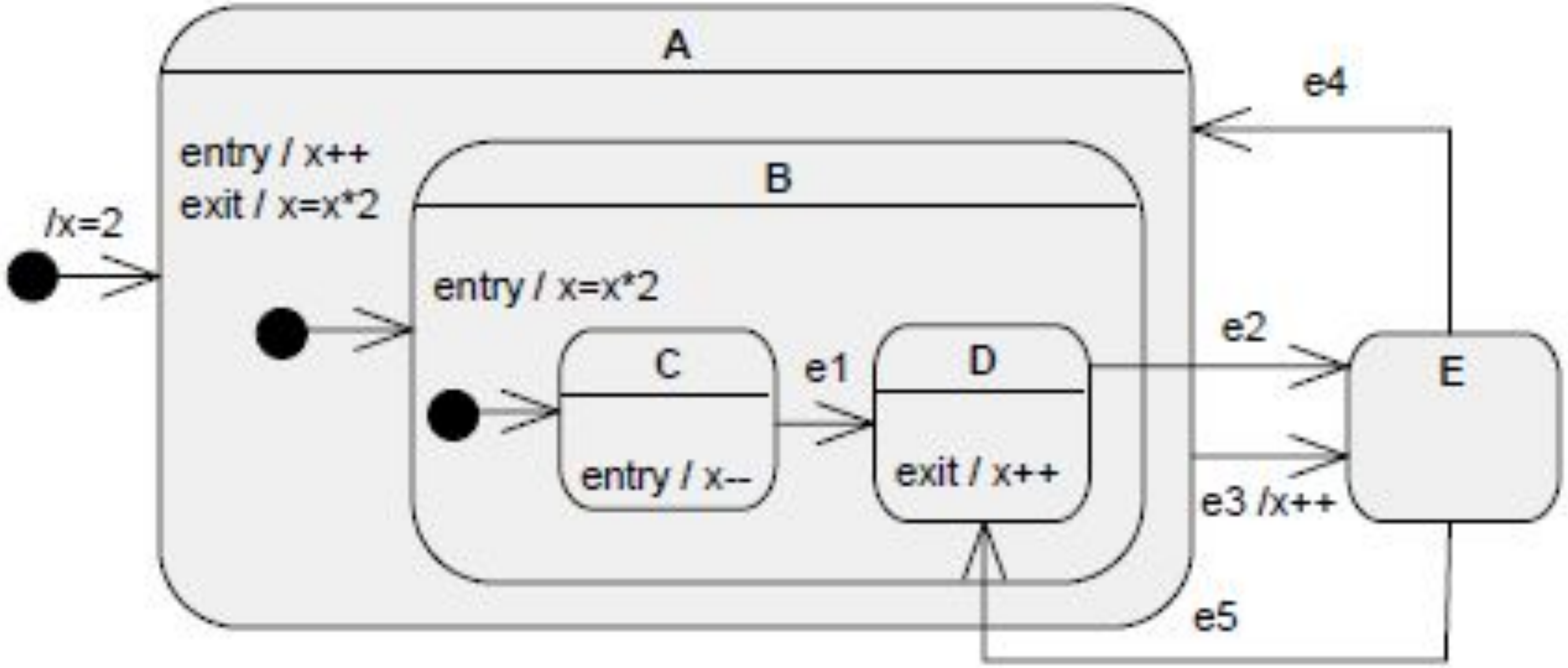
What is the value of x after the occurrence of the event chain e2 e3 e2 e4 e5 ?



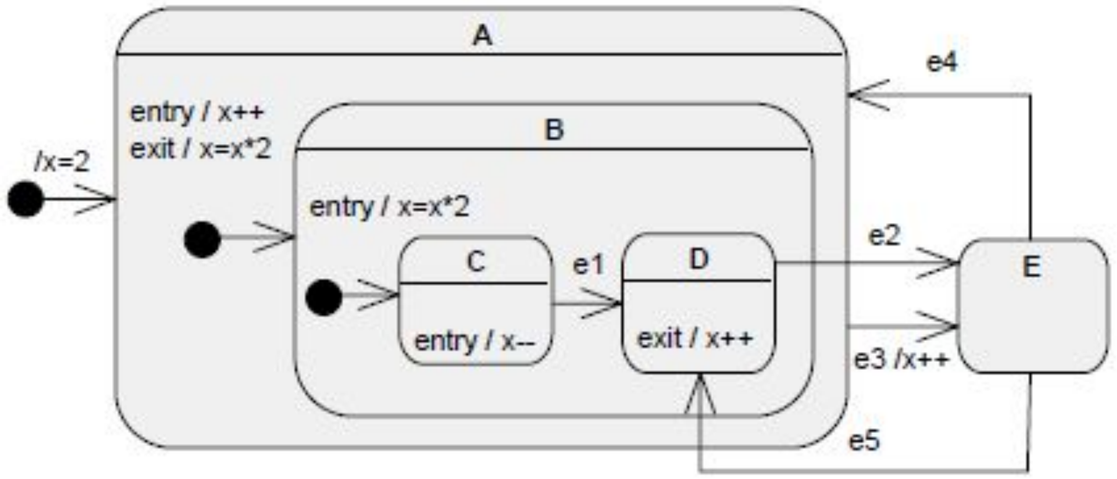
event	state	comment	x
		start	3
	A	entry of A	4
	A/B	entry of B	2
e2	A/C	exit of B x = x * 2	5 10
e3	A/B	entry of B	5
e2	A/C	exit of B x = x * 2	8 16
e4		x++ in C	17
e5		exit of A	16
	D	entry of D	17

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

What is the value of x after the occurrence of the event chain e1 e3 e5 e2 ?

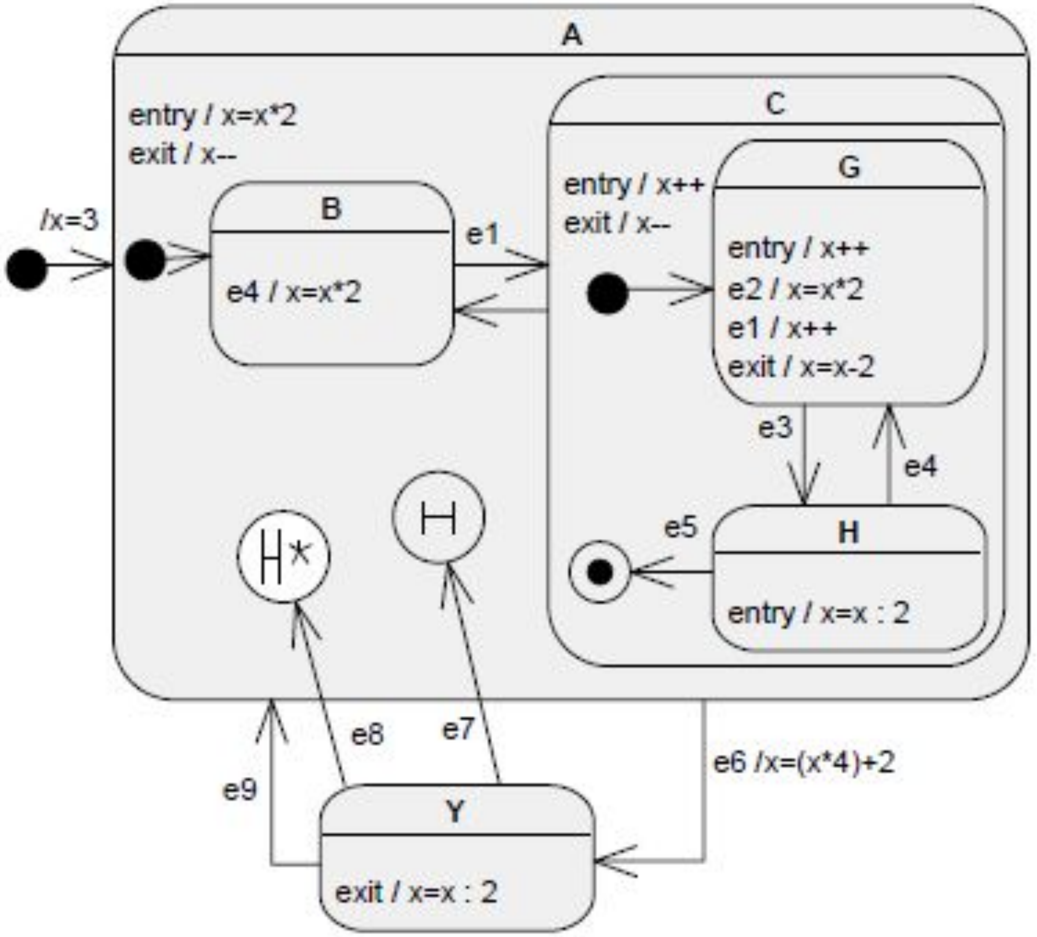


What is the value of x after the occurrence of the event chain e1 e3 e5 e2 ?

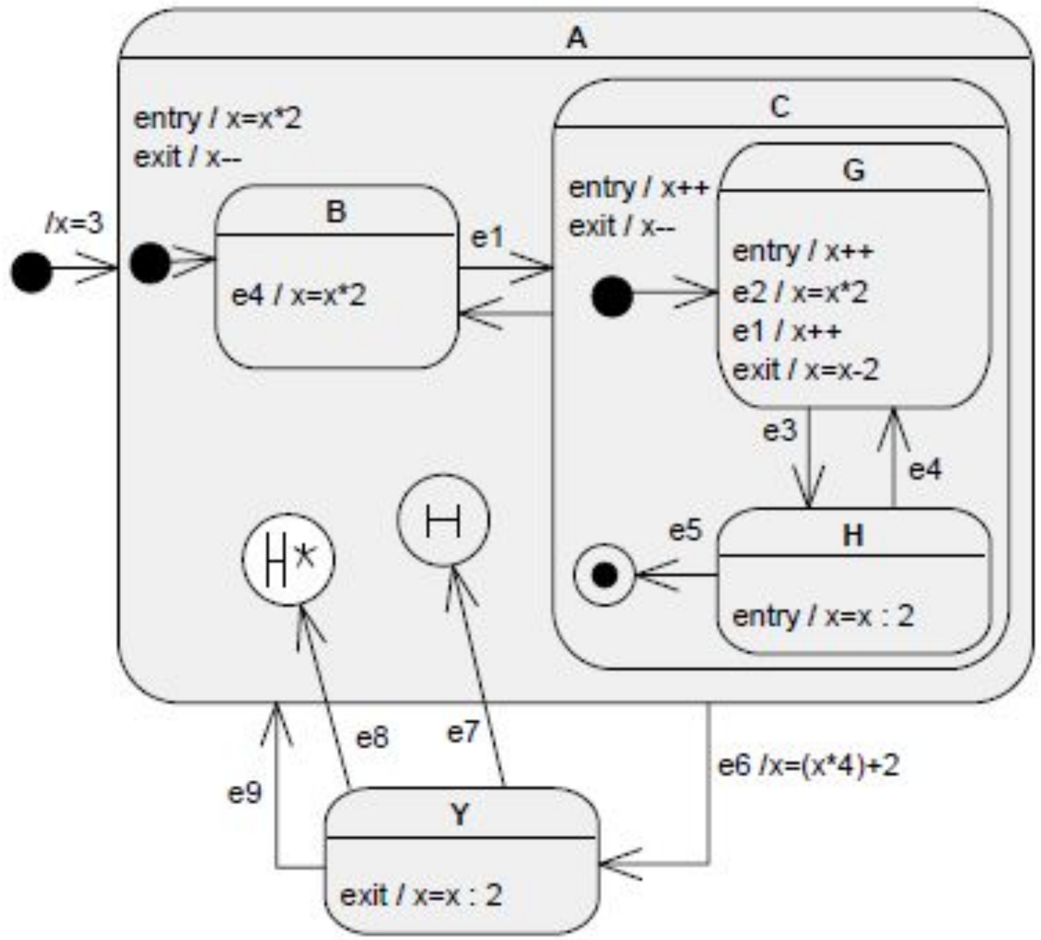


event	state	comment	x
		start	2
	A	entry of A	3
	A/B	entry of B	6
	A/B/C	entry of C	5
e1	A/B/C		
e3		exit of D exit of A x++	6 12 13
e5	A/B/D	entry of A entry of B	14 28
e2		exit of D exit of A	29 **58**
	E		

What is the value of x after the occurrence of the event chain e1 e6 e9 e1 e3 ?



What is the value of x after the occurrence of the event chain e1 e6 e9 e1 e3 ?



event	state	comment	x
		start	3
	A	entry of A	6
	A/B		
e1	A/C	entry of C	7
	A/C/G	entry of G	8
e6		exit of G	6
		exit of C	5
		exit of A	4
		x=(x*e)+2 (Transition)	18
	Y		
e9		exit of Y	9
	A	entry of A	18
	A/B		1
e1	A/C	entry of C	19
	A/C/G	entry of G	20
e3		exit of G	18
	A/C/H	entry of H	**9**

Find these and many other examples in http://elearning.uml.ac.at/

Bibliography

Jim Arlow and Ila Neustadt, “UML 2 and the Unified Process”, Second Edition, Addison-Wesley 2006

- Chapters 21, 22

UML Quizz, Business Informatics Group, Vienna University of Technology (online, last checked in December 2018)

- http://elearning.uml.ac.at/