

UML: Diagramma delle classi; Diagramma degli oggetti

Vincenzo Gervasi, Laura Semini
Ingegneria del Software
Dipartimento di Informatica
Università di Pisa

Riassunto lezione precedente

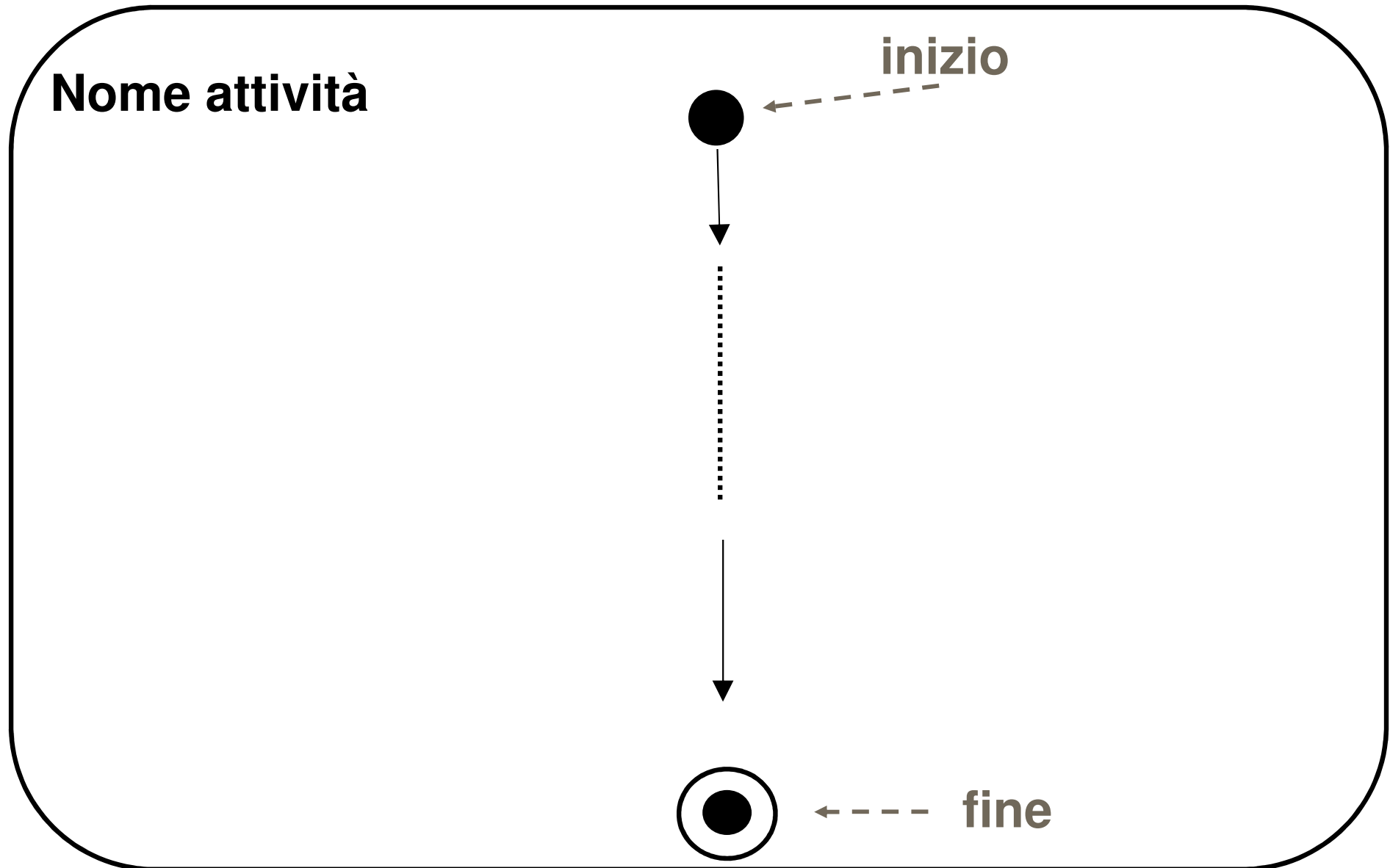
Outline della lezione

- Lezioni precedente:
 - Descrizione del dominio: modello statico
- Questa lezione
 - Descrizione del dominio: modello dinamico
 - - diagrammi di attività (business model)
 - diagrammi di macchina a stati

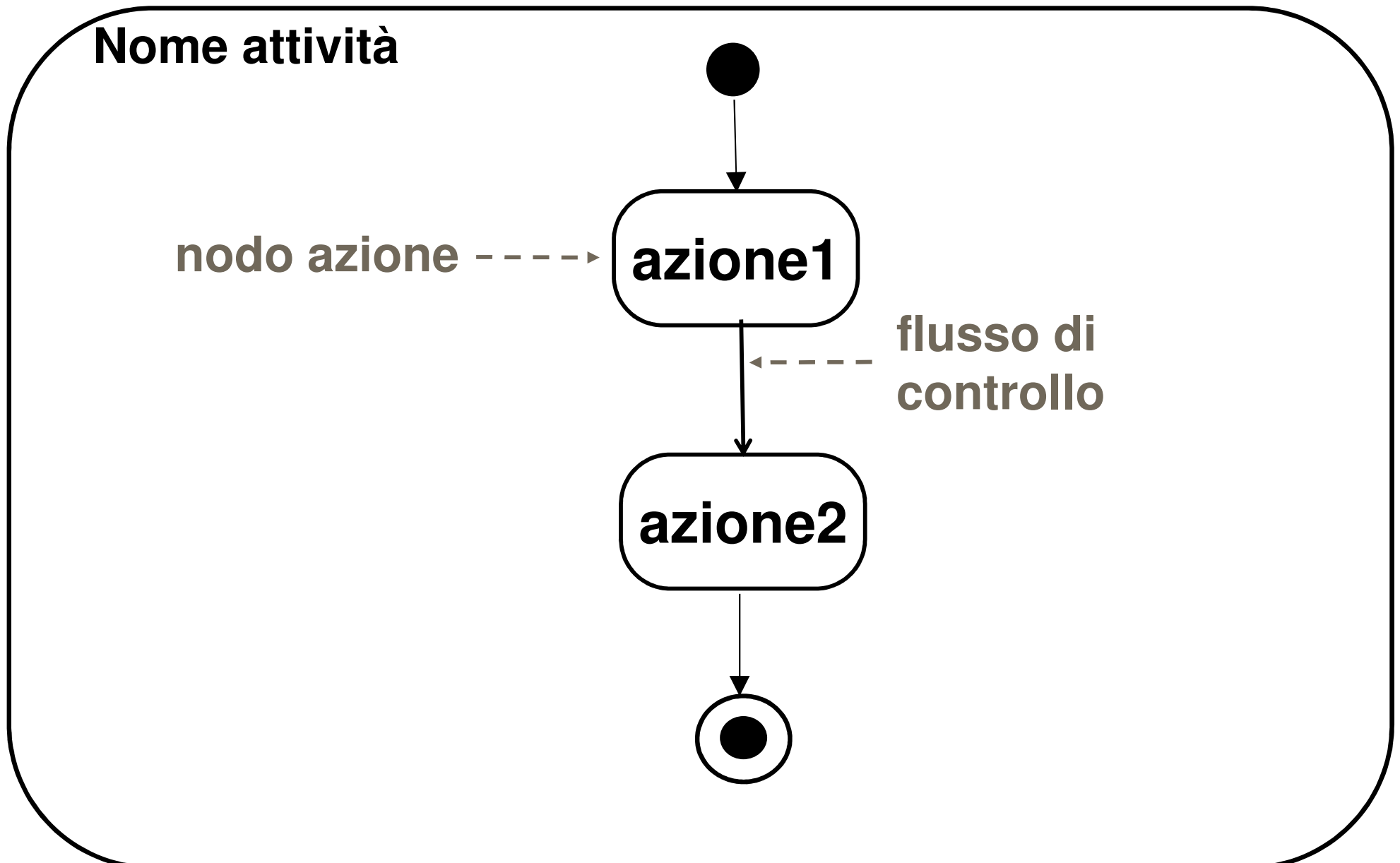
Diagrammi di attività

- Modellano il flusso di lavoro (workflow, business model)
 - di una computazione (prospettiva software) o
 - di un processo/attività (business)
- Un'attività descrive la coordinazione di un insieme di azioni. Centrata su:
 - sequenza e concorrenza delle azioni
 - e sulle condizioni che le abilitano
 - piuttosto che sui classificatori che eseguono queste azioni
- Antenati: flow charts e Reti di Petri

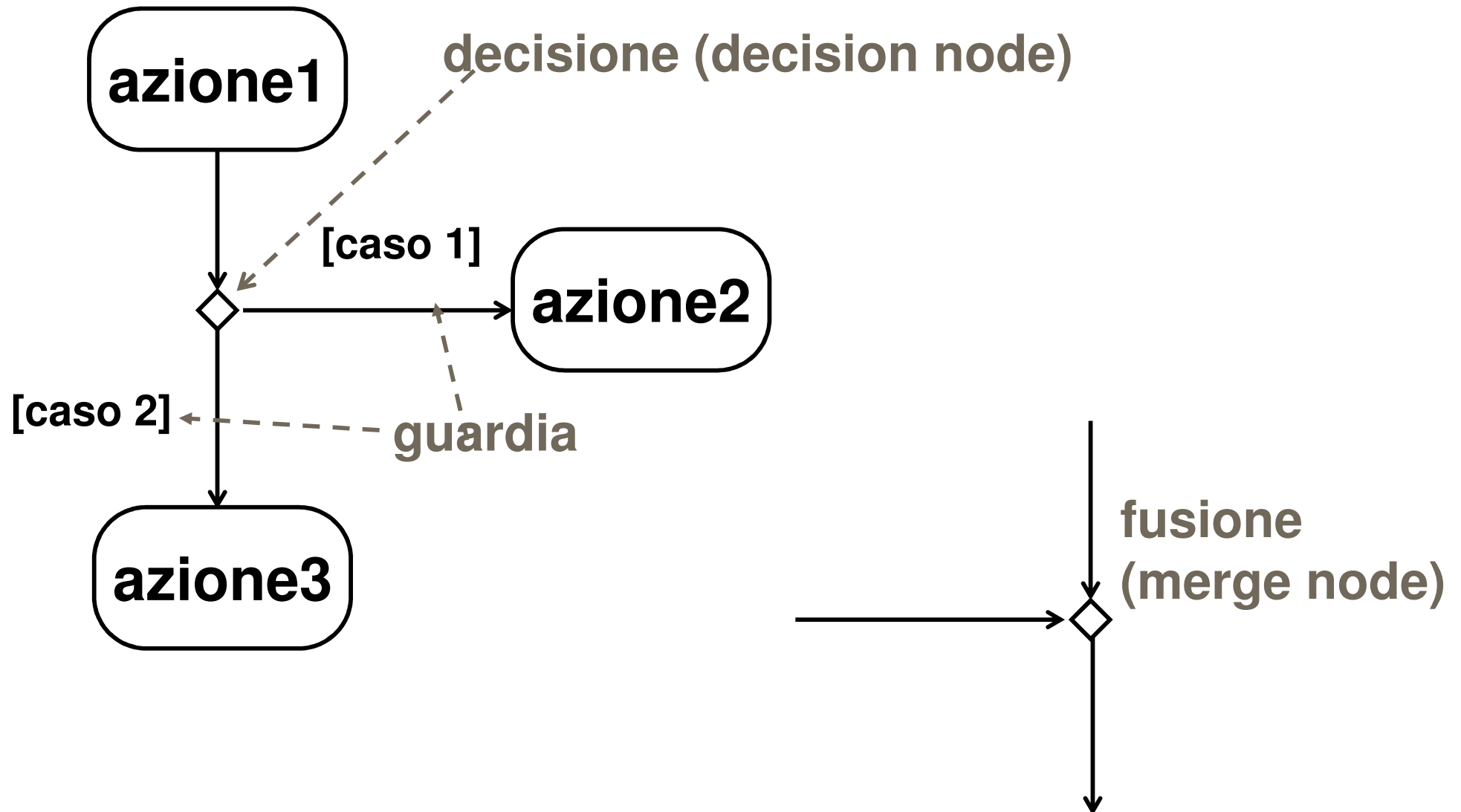
Diagrammi di attività: inizio e fine



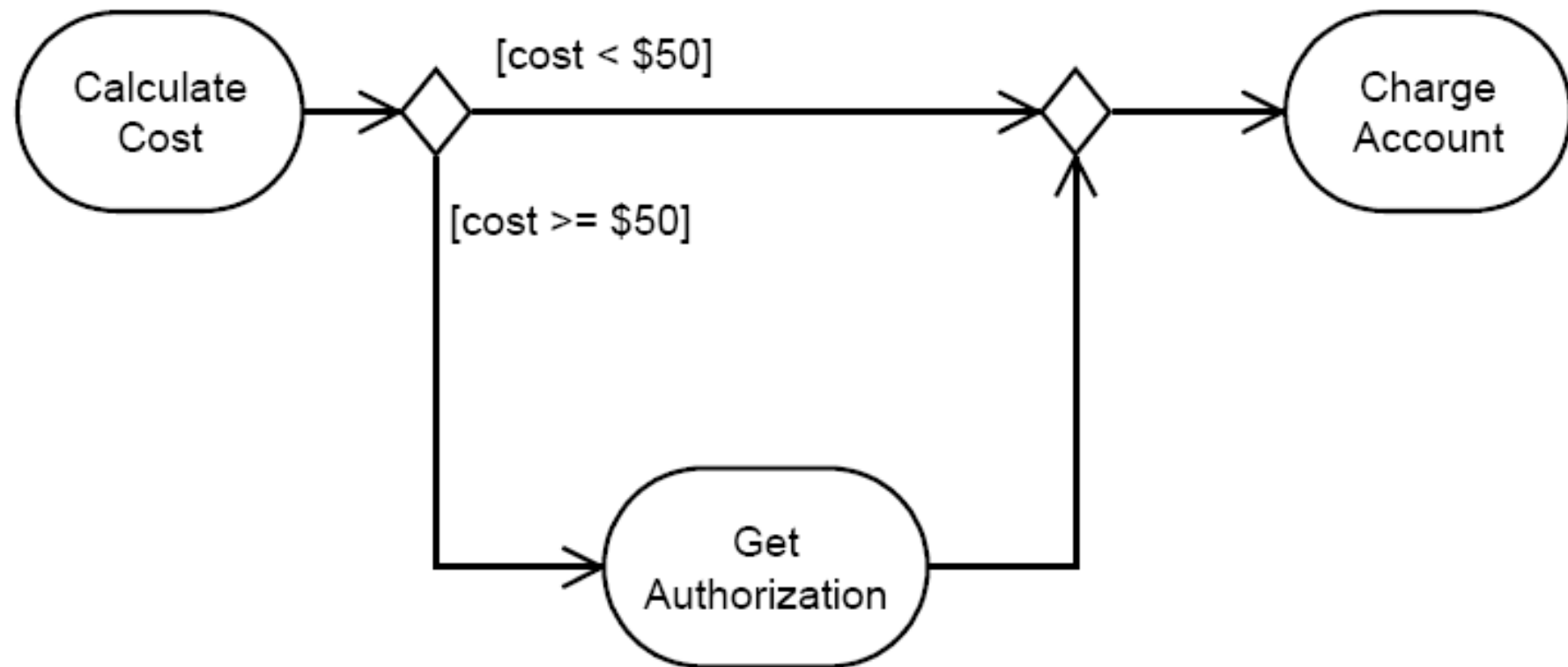
Diagrammi di attività: sequenza



Diagrammi di attività: scelta



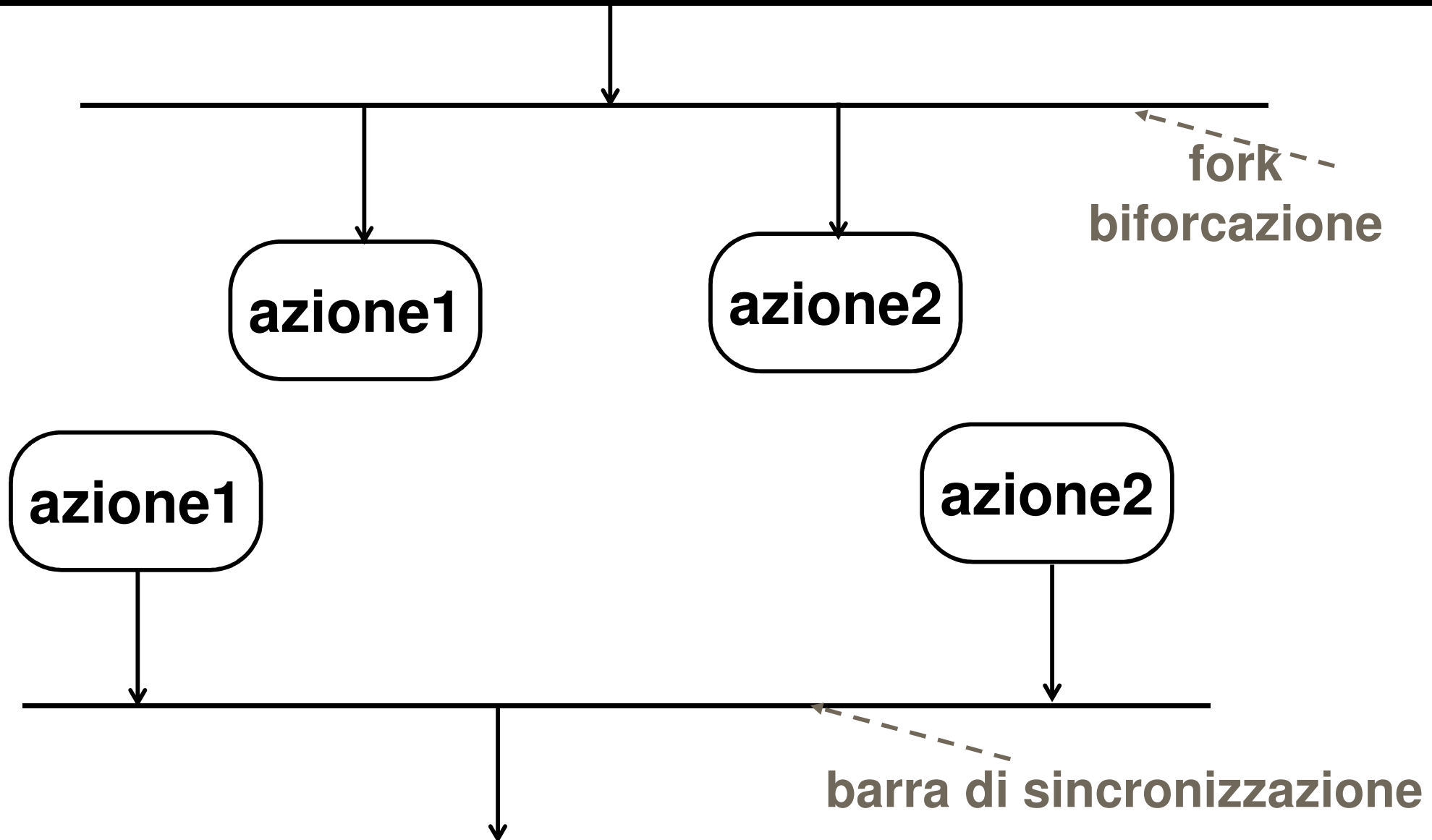
esempio “Branching” e “merging”



Decisione e fusione / decision and merge

- Token
- Le guardie devono coprire tutte le possibilità
 - In caso si usa [else]
- E' bene che siano mutualmente esclusive altrimenti comportamento non definito.
- Le condizioni di guardia sempre tra []
 - (in generale in UML)
- Dato un nodo decisione non è obbligatorio un nodo fusione corrispondente.
 - Potrebbe esserci un nodo di fine flusso

Diagrammi di attività: fork e join



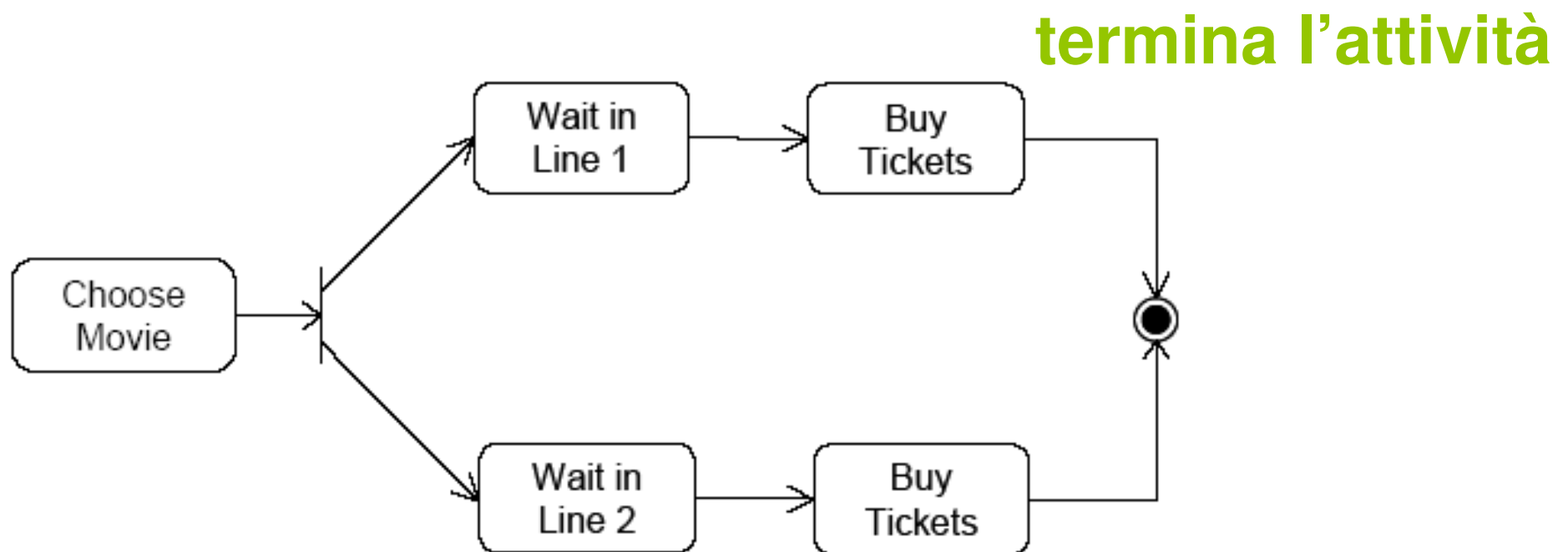
Biforcazione e ricongiunzione / fork and join

- Token
- Non vincolo a join per ogni fork

Diagrammi di attività (2)

- Un'attività non può essere interrotta da segnali esterni
- In UML 2 solo una freccia entrante per azione (OLD: Un'azione inizia solo quando tutti i flussi l'hanno raggiunta)

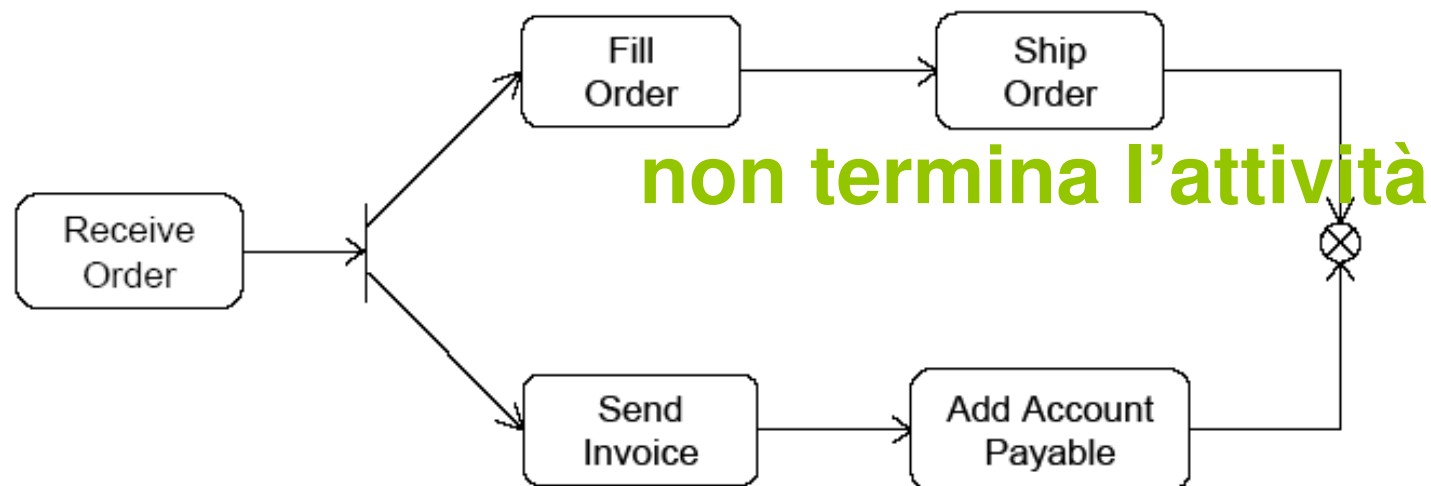
ESEMPIO: in 2 a compare un biglietto, il primo che compra finisce tutto



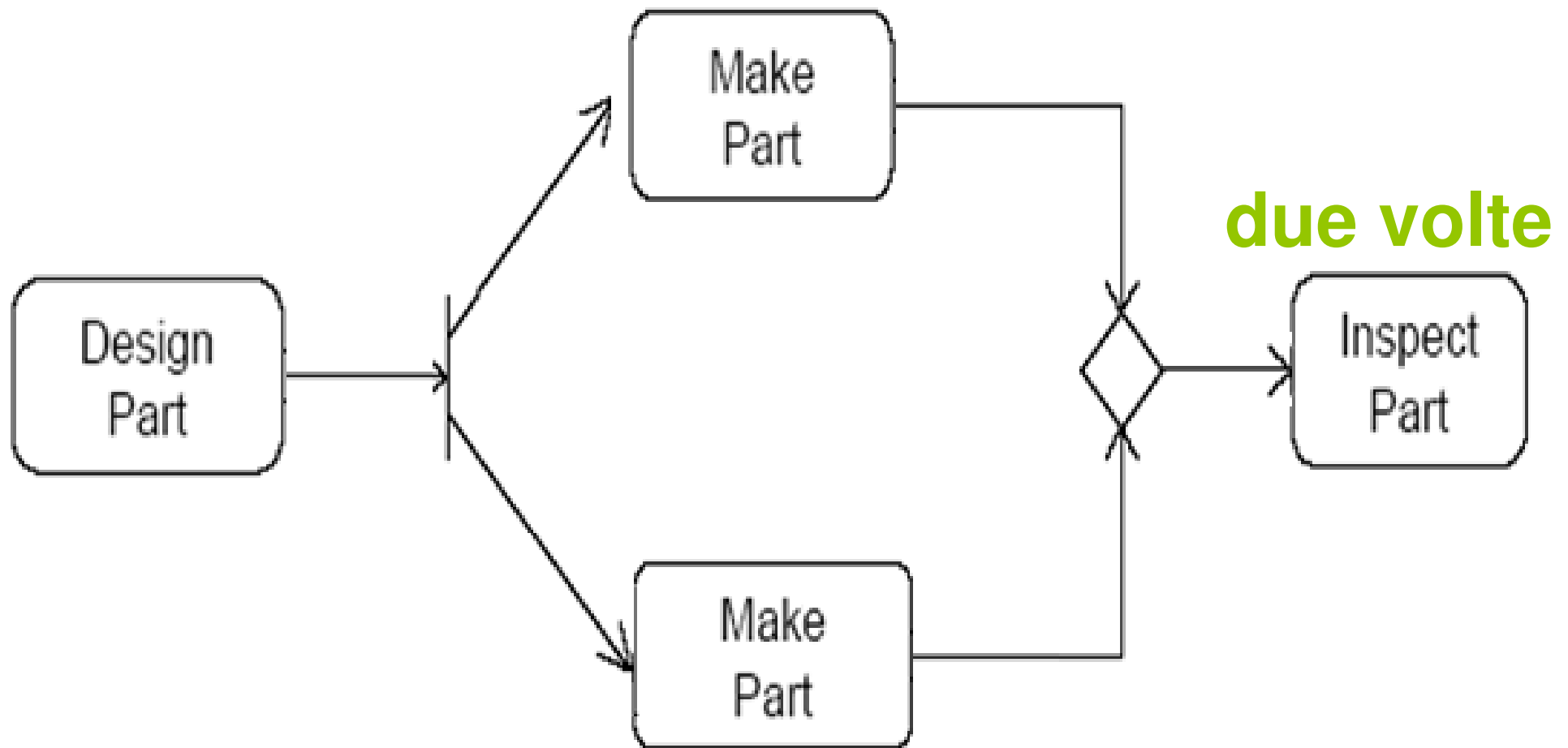
Fine di flusso

flow final node

A node in an **activity** that destroys all **tokens** that reach it. It has no other effect on the execution of the activity.

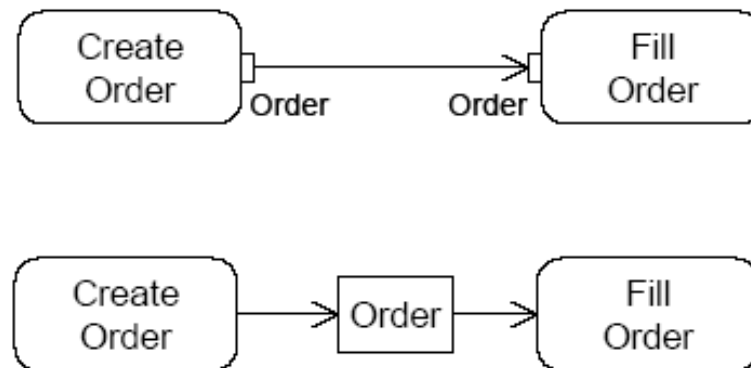


Fork e merge



Dati

- Le azioni possono avere input e output: i dati sono scambiati tramite oggetti (nodi oggetto)

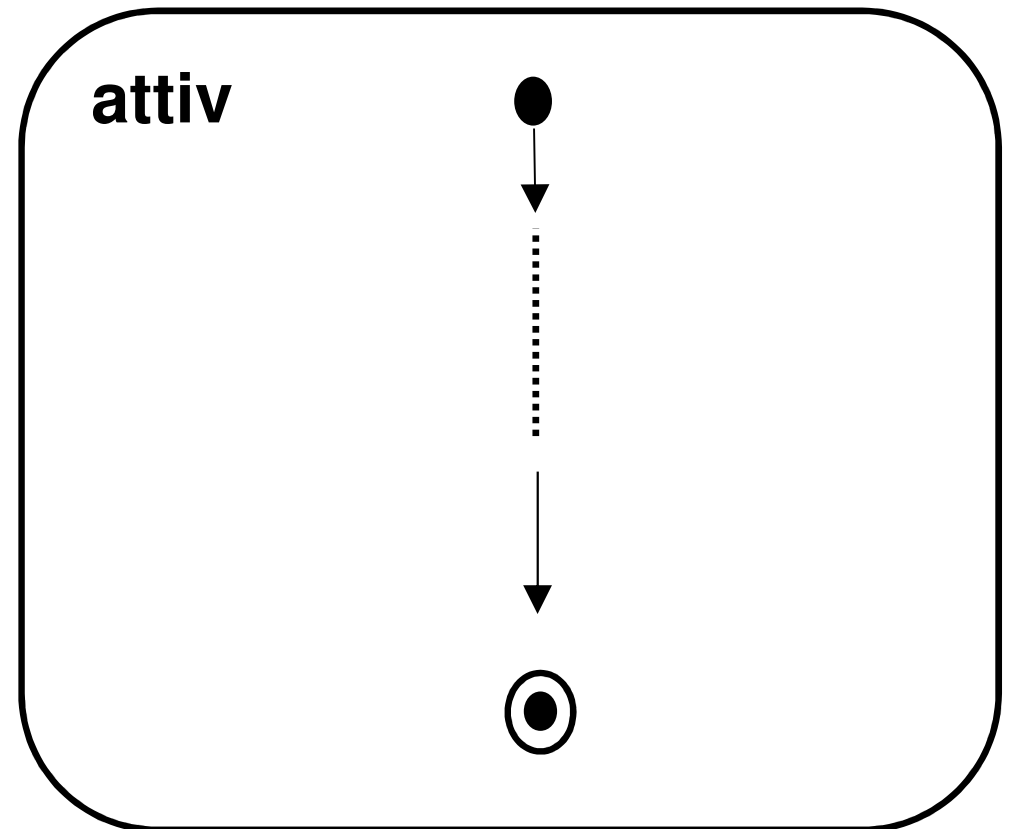
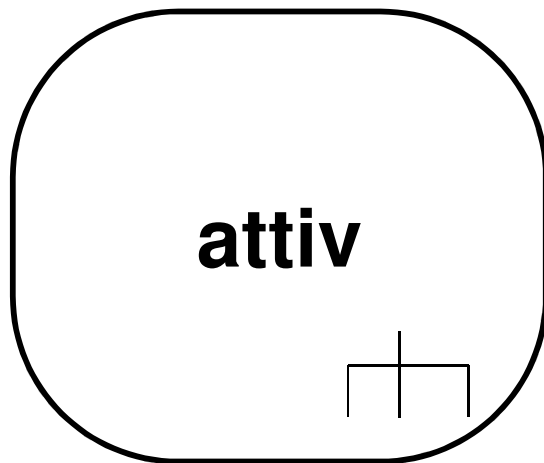


Gli oggetti scambiati

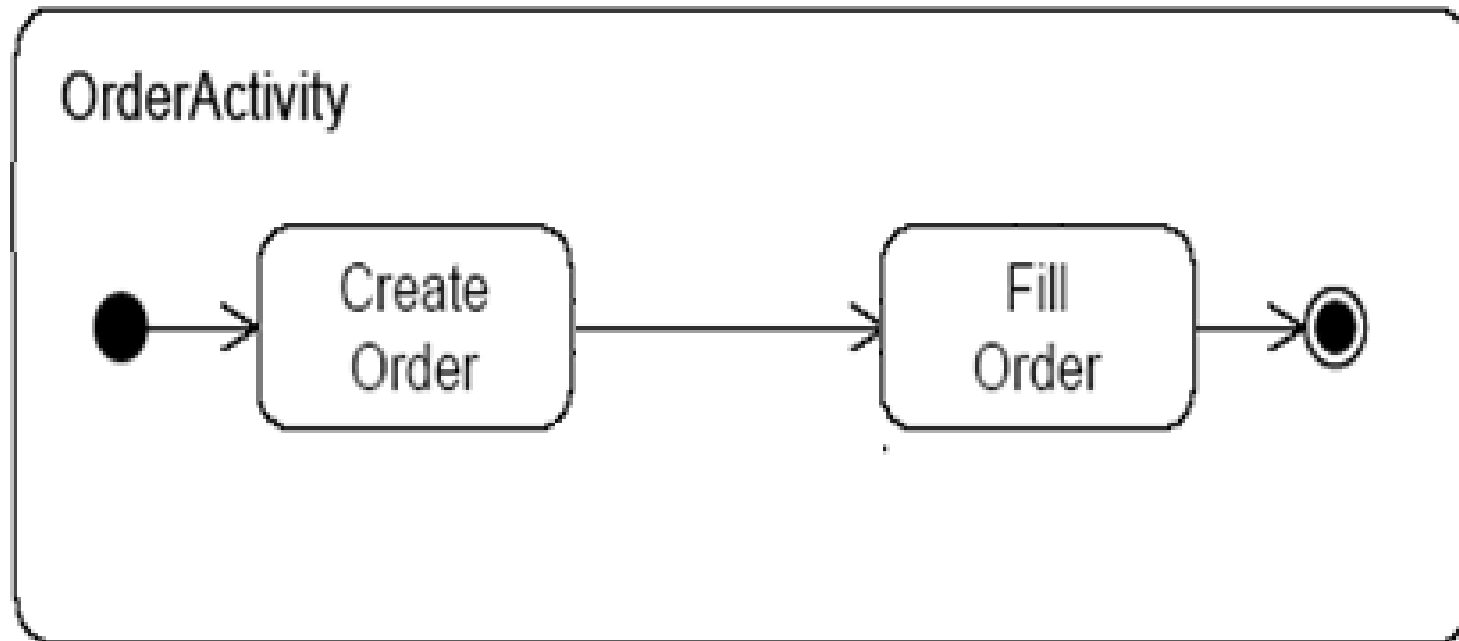
- possono avere uno stato [paid]
- nome del classificatore (maiuscolo e non sottolineato)

SottoAttività

- Un'azione può includere (chiamare) un'altra attività (secondaria) : si usa il "rastrello" (rake) in basso a dx.



(Sotto)attività



Azione OrderActivity con rastrello in altro diagramma

Azioni legate agli eventi

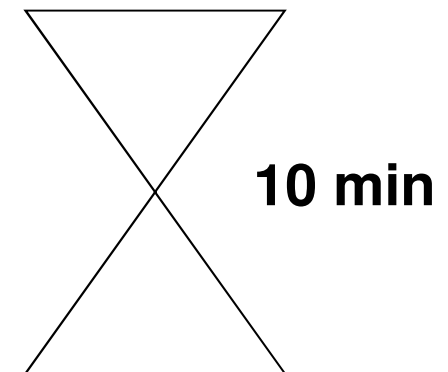
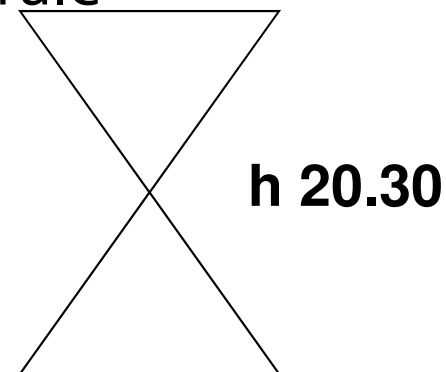
- Accettazione di evento esterno



- Invio di un segnale



- Accettazione di evento temporale



Azione di accettazione evento

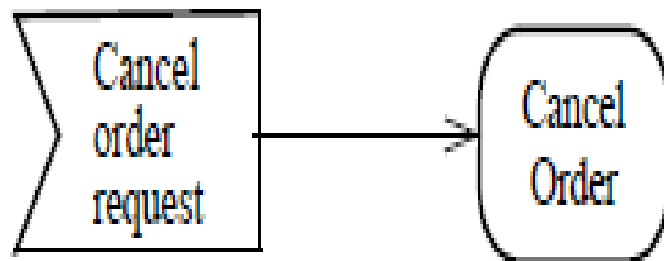
- arco entrante non necessario
- In caso, l'azione rimane abilitata e può ricevere altri eventi
- Stessa cosa vale per la clessidra

Semantics

If an `AcceptEventAction` has no incoming edges, then the action starts when the containing activity or structured node does, whichever most immediately contains the action. In addition, an `AcceptEventAction` with no incoming edges remains enabled after it accepts an event. It does not terminate after accepting an event and outputting a value, but continues to wait for other events. This semantic is an exception to the normal execution rules in Activities. An `AcceptEventAction` with no incoming edges and contained by a structured node is terminated when its container is terminated.

Da superstructure

Figure 12.25 is an example of the acceptance of a signal indicating the cancellation of an order. The acceptance of the signal causes an invocation of a cancellation behavior. This action is enabled on entry to the activity containing it, therefore no input arrow is shown.



Da superstructure

In Figure 12.26, a request payment signal is sent after an order is processed. The activity then waits to receive a payment confirmed signal. Acceptance of the payment confirmed signal is enabled only after the request for payment is sent; no confirmation is accepted until then. When the confirmation is received, the order is shipped.

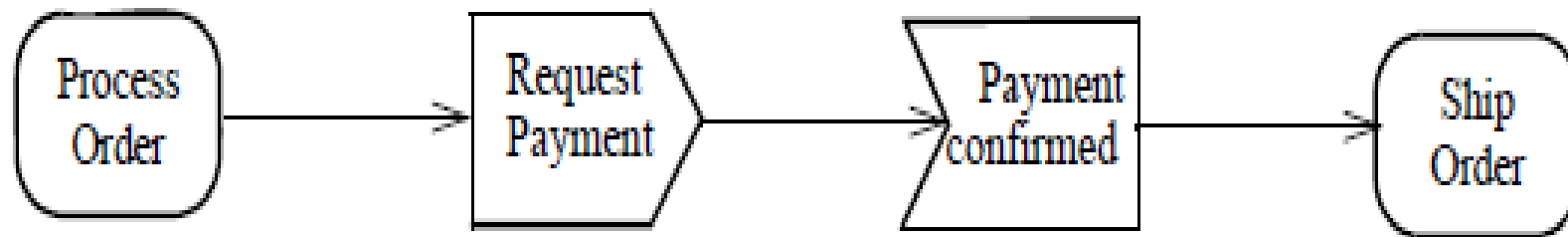


Figure 12.26 - Accept signal, explicit enable

Da superstructure

In Figure 12.27, the end-of-month accept time event action generates an output at the end of the month. Since there are no incoming edges to the time event action, it is enabled as long as its containing activity or structured node is. It will generate an output at the end of every month.

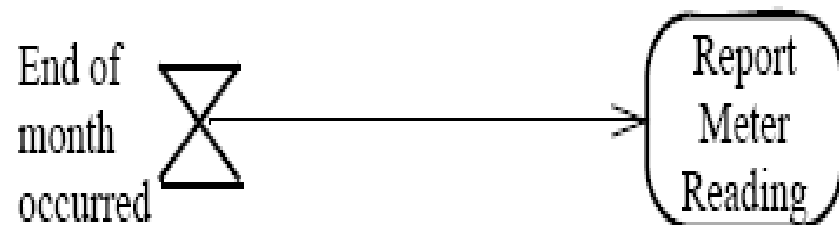
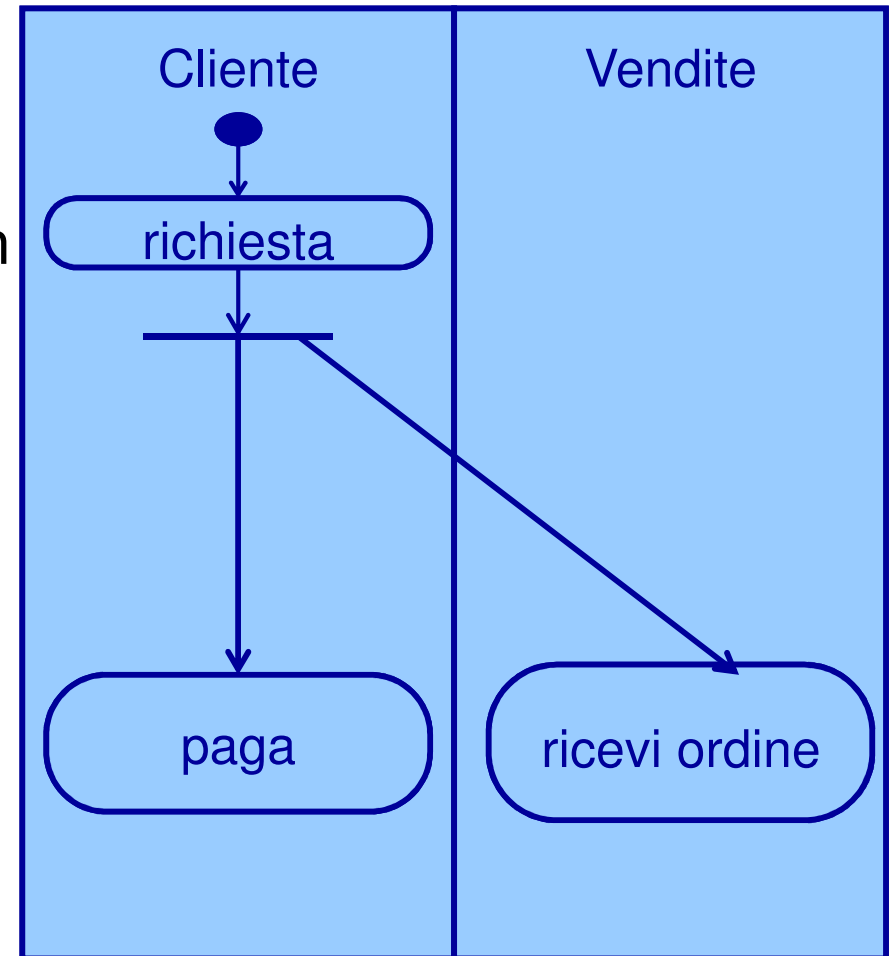


Figure 12.27 - Repetitive time event

Partizioni

- Una *partizione*
 - per dividere le azioni in gruppi
 - Spesso corrisponde alla divisione in modello di business.
- Permettono di
 - descrivere le partizioni
 - assegnare le responsabilità



Macchine a stati

Macchina a stati

- Una macchina a stati è un grafo di stati e transizioni, associata a un classificatore (classe, sistema..)
- Per ogni classe può esistere una macchina a stati che modella tutte le transizioni di stato di tutti gli oggetti di quella classe, in risposta a diversi tipi di evento
- Gli eventi sono tipicamente dei messaggi
 - inviati da altri oggetti
 - generati internamente
- La macchina a stati di una classe modella il comportamento degli oggetti della classe in modo trasversale per tutti i casi d'uso interessati

Stato

- Uno stato descrive un periodo di tempo durante la vita di un oggetto
- Può essere caratterizzato come:
 - Un insieme di valori qualitativamente simili
 - Un periodo di tempo durante il quale un oggetto attende il verificarsi di un evento
 - Un periodo di tempo durante il quale un oggetto svolge un'attività
- Uno stato ha un nome unico
- Uno stato può essere composito (più avanti)

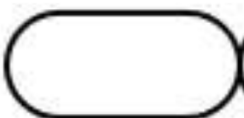
Sintassi di base

Gli stati sono rappresentati con rettangoli arrotondati

Il disco nero marca l'inizio dell'esecuzione. Non è uno stato vero e proprio ma un marcatore che punta allo stato da cui partire.

Il disco nero bordato (nodo finale), indica che l'esecuzione è terminata.

Possono comparire in qualunque numero all'interno di un diagramma

Stato: 

Stato iniziale: ●

Stato finale: 

Transizione

- Una transizione collega tra loro due stati, è rappresentata con una freccia
- L'uscita da uno stato definisce la risposta dell'oggetto all'occorrenza di un evento



eventi ::= evento | evento , eventi (disgiunz)
azioni ::= azione | azione, azioni (sequenza)

Tutti opzionali anche se l'evento è bene che ci sia.
Nelle transizioni di completamento (più avanti) non serve.

Esempio

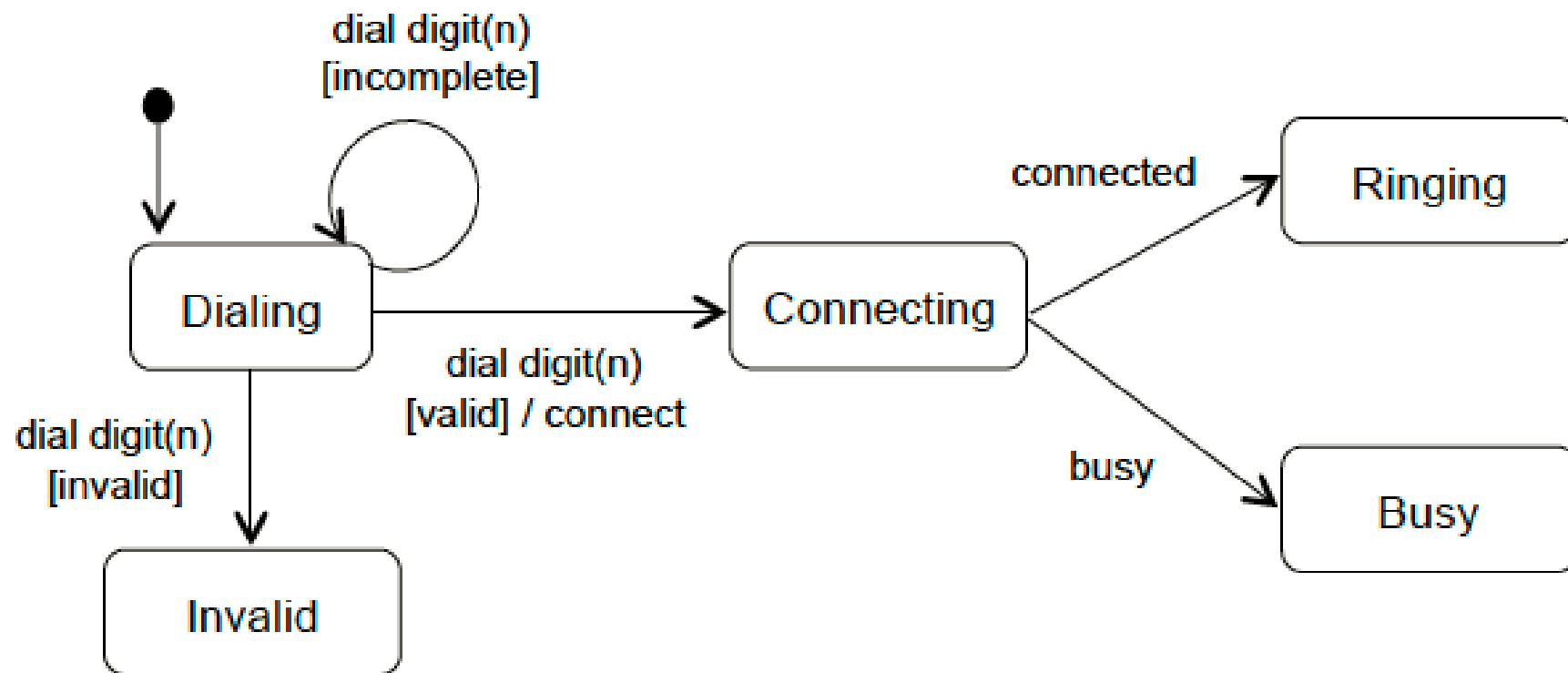


Evento

- Un evento è l'occorrenza di un fenomeno collocato nel tempo e nello spazio
- Un evento occorre istantaneamente
- Gli eventi che arrivano in uno stato per cui non è prevista alcuna transizione vengono ignorati
- È ammesso il non-determinismo

may fire (within one thread of control) in response to one event occurrence. If an event enables more than one transition, only one of them fires. A transition on a nested state takes precedence over a transition on one of its enclosing states. If two conflicting transitions are enabled at the same time, one of them fires nondeterministically. The choice may be random or it may depend on implementation de-

Esempio

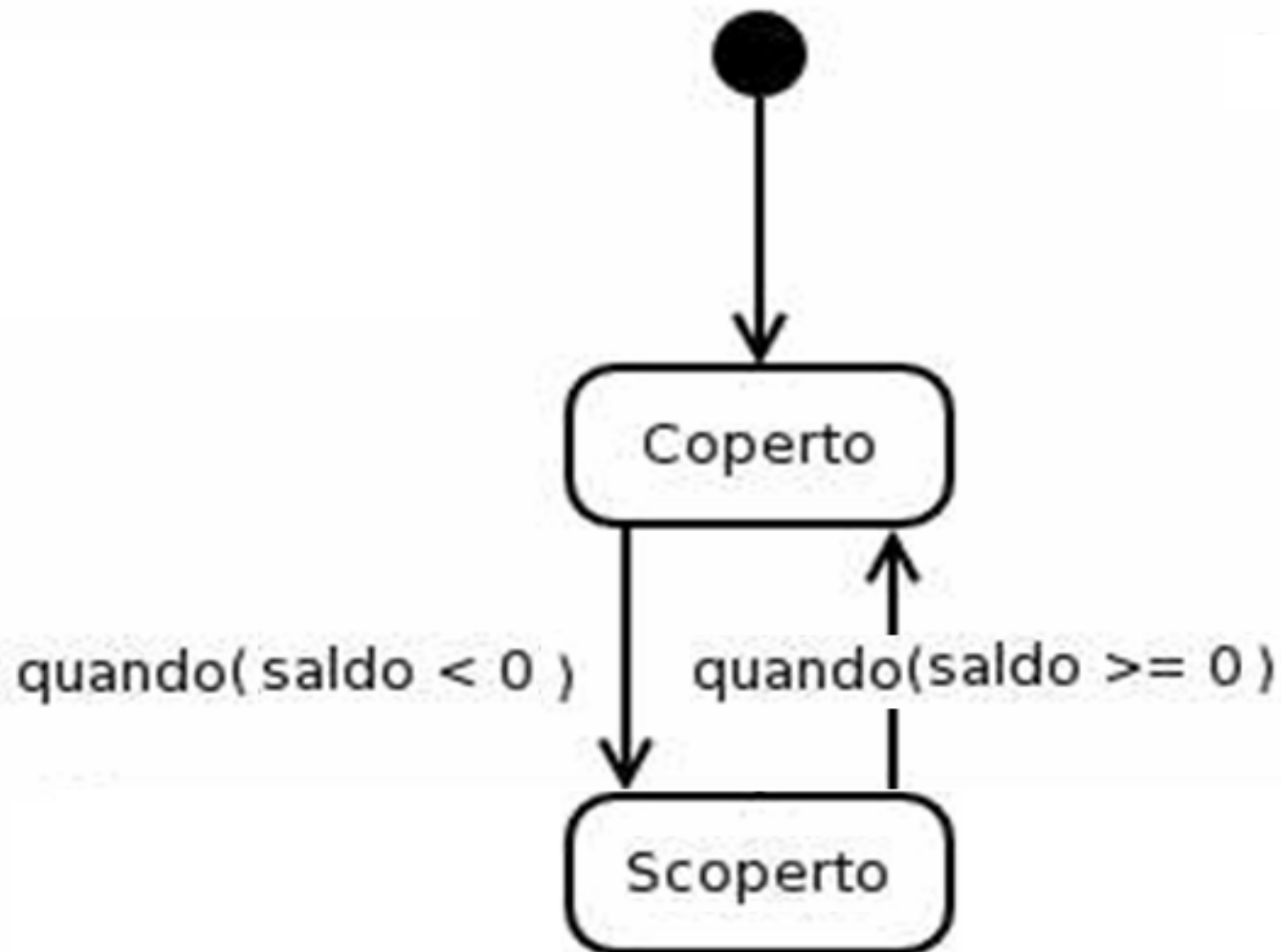


Tipi di eventi

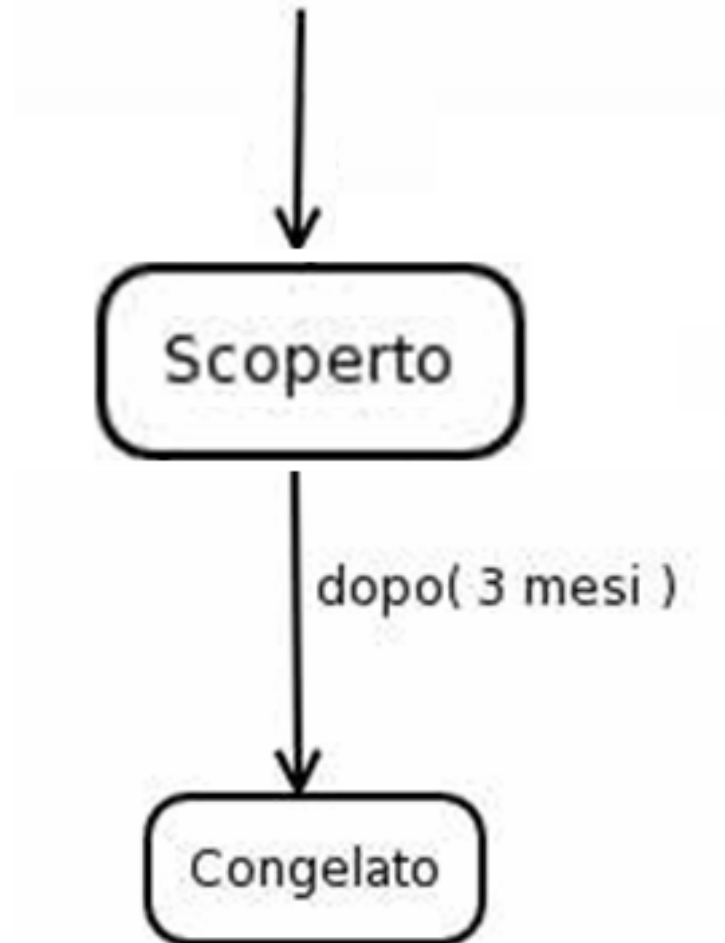
Table 7-1: *Kinds of Events*

| <i>Event Type</i> | <i>Description</i> | <i>Syntax</i> |
|---------------------|---|---------------------|
| call event | Receipt of an explicit synchronous call request by an object | op (a:T) |
| change event | A change in value of a Boolean expression | when (exp) |
| signal event | Receipt of an explicit, named, asynchronous communication among objects | sname (a:T) |
| time event | The arrival of an absolute time or the passage of a relative amount of time | after (time) |

Eventi di variazione (esempio)



Eventi del tempo (esempio)



Tipi di transizioni

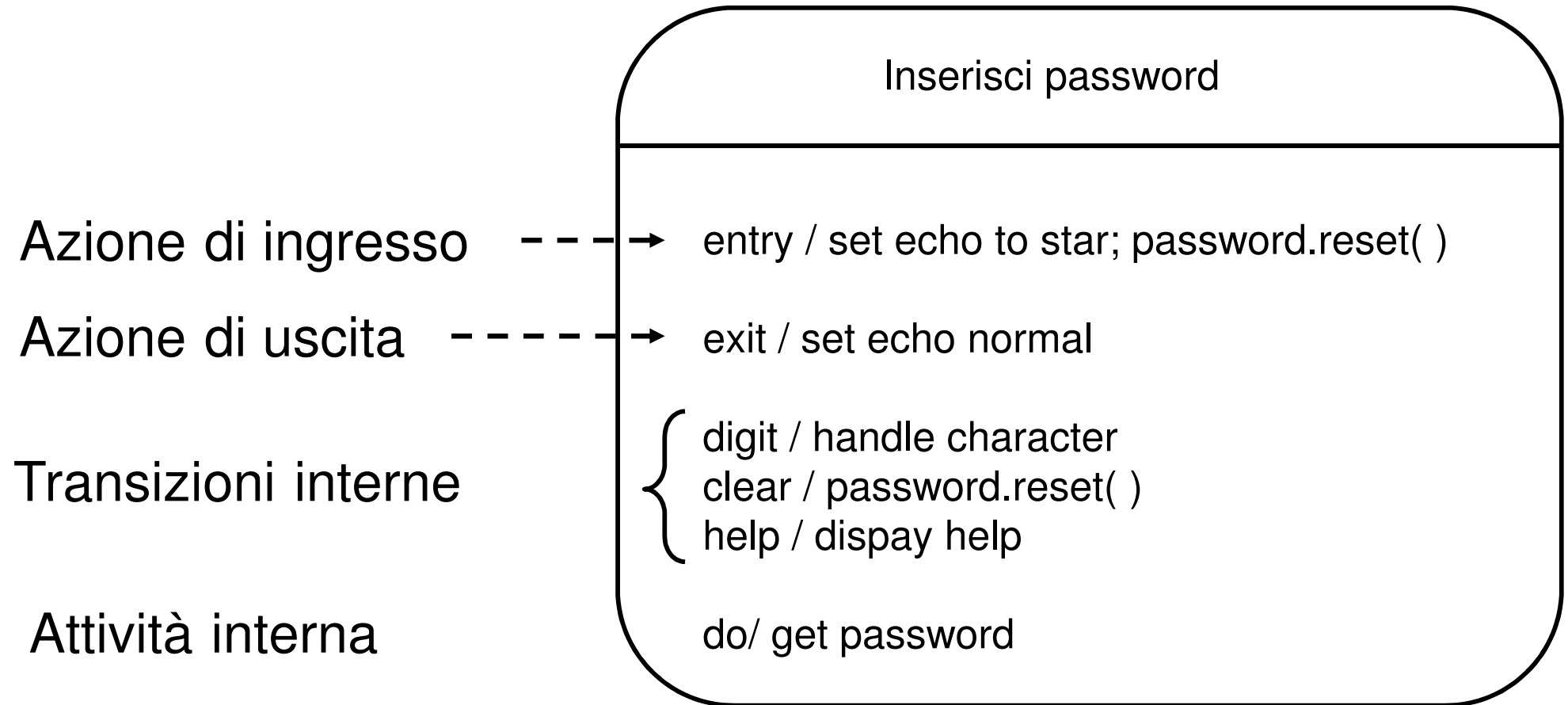
Table 7-2: *Kinds of Transitions and Implicit Effects*

| <i>Transition Kind</i> | <i>Description</i> | <i>Syntax</i> |
|----------------------------|---|---|
| entry transition | The specification of an entry activity that is executed when a state is entered | entry/ activity |
| exit transition | The specification of an exit activity that is executed when a state is exited | exit/ activity |
| external transition | A response to an event that causes a change of state or a self-transition, together with a specified effect . It may also cause the execution of exit and/or entry activities for states that are exited or entered. | e(a:T)[guard]/ activity |
| internal transition | A response to an event that causes the execution of an effect but does not cause a change of state or execution of exit or entry activities | e(a:T)[guard]/ activity |

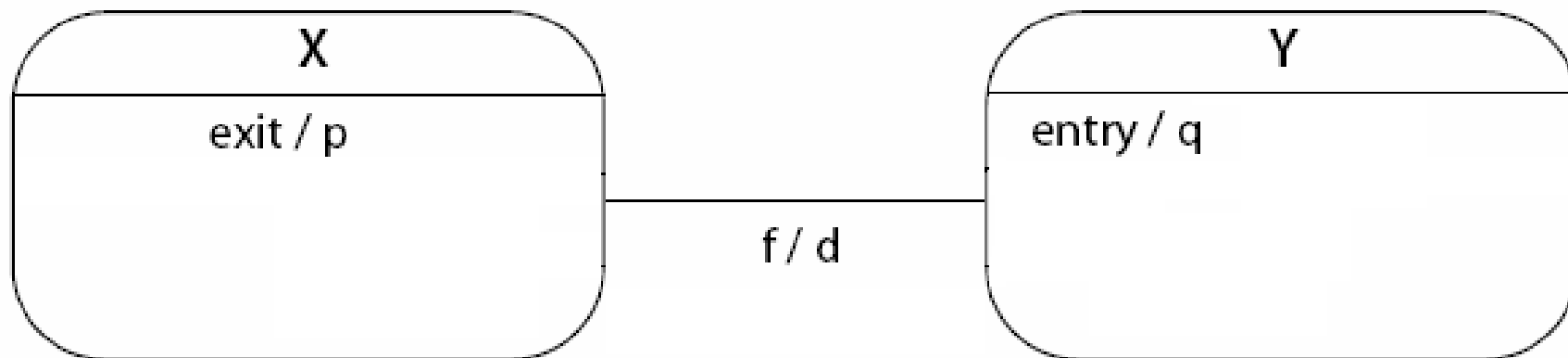
Transizioni e attività interne

- Transizione interna: risposta a un evento che causa solo l'esecuzione di azioni. Esempi:
 - Azione di entrata: eseguita all'ingresso in uno stato
 - Azione di uscita: eseguita all'uscita di uno stato
 - Transizione interna: risposta ad un evento
- Attività interna (Do-activity): eseguita in modo continuato (senza necessità di un evento scatenante) consuma del tempo e può essere interrotta

Esempio



Esempi

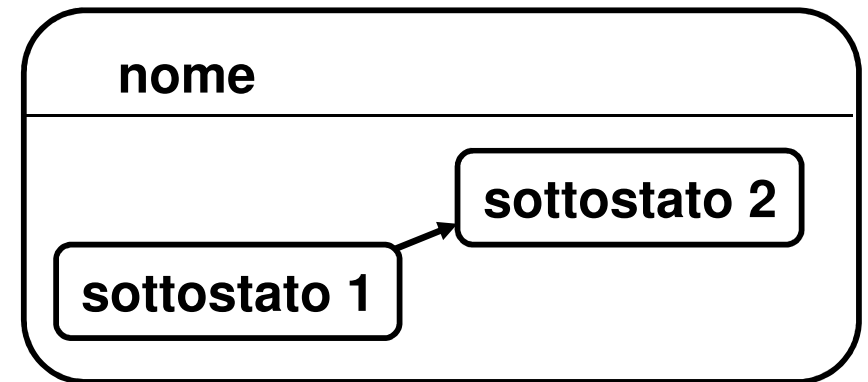


effective result: f / p; d; q



Stati compositi

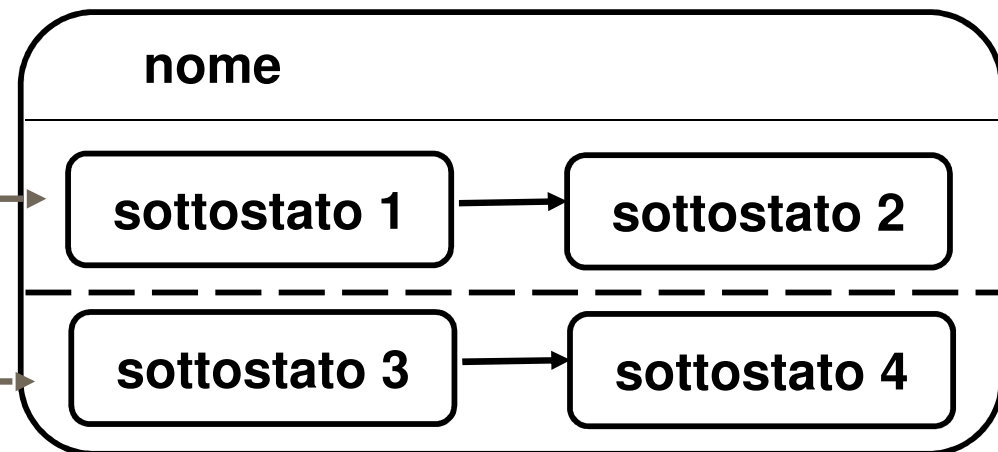
- Composito sequenziale:
 - uno attivo per ogni istante



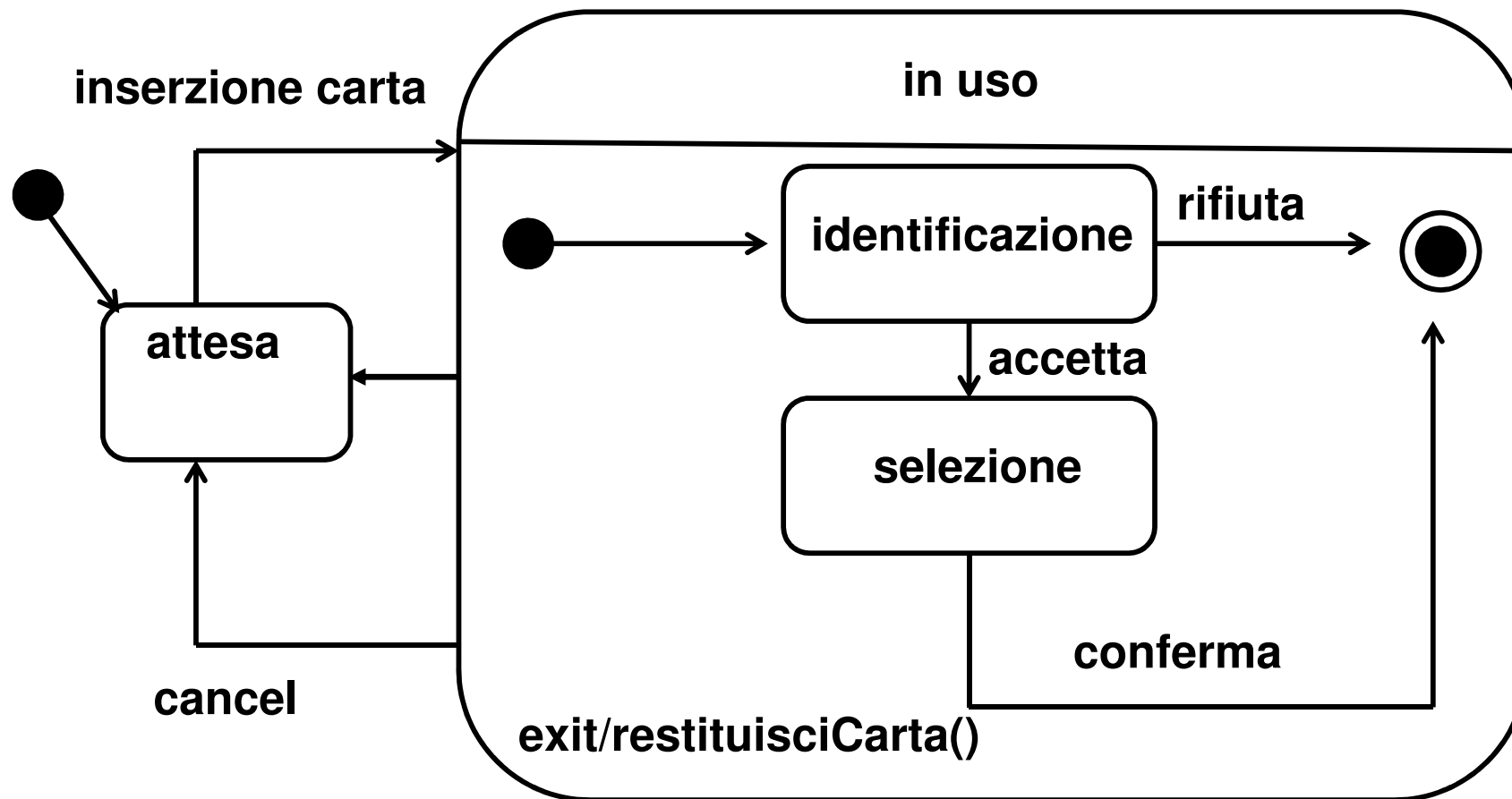
- Composito parallelo:
 - sottostati attivi contemporaneamente

regione 1

regione 2

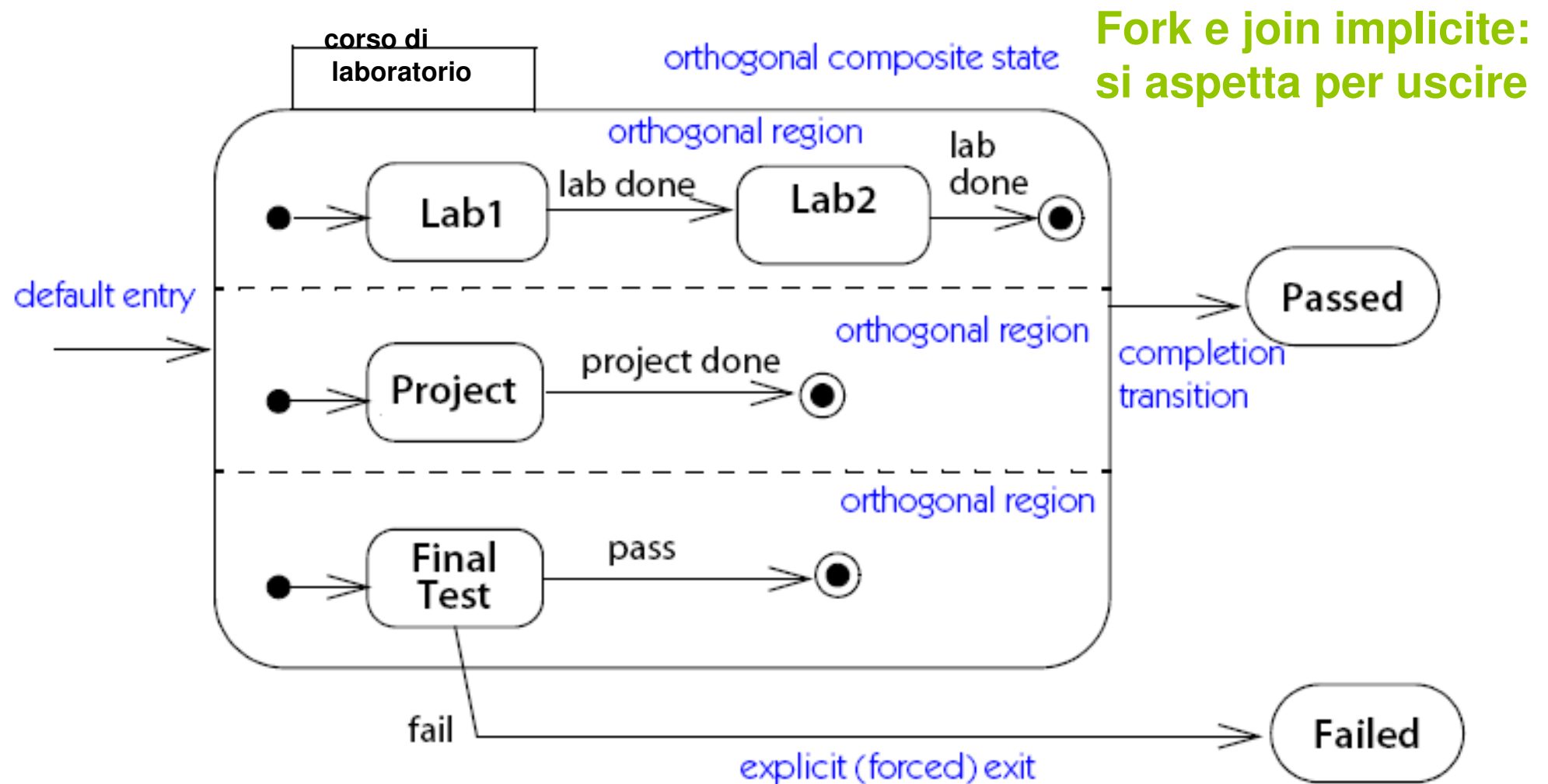


Esempio stato composito sequenziale



Each region of a composite state may have an **initial state**. A transition to the composite state boundary is implicitly a transition to the initial state. A new object

stato composito parallelo un esempio: corso di laboratorio



starts at the initial state of its outermost state. Similarly, a composite state can have a **final state**. A transition to the final state triggers a **completion transition** on the composite state. If an object reaches the final state of its outermost state, it is de-

Altri tipi di stato (pseudostati)

■ Giunzione



■ Decisione



■ Fork, join



➤ **Storia**




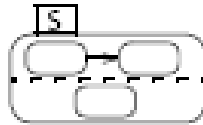







➤ **Iniziale**



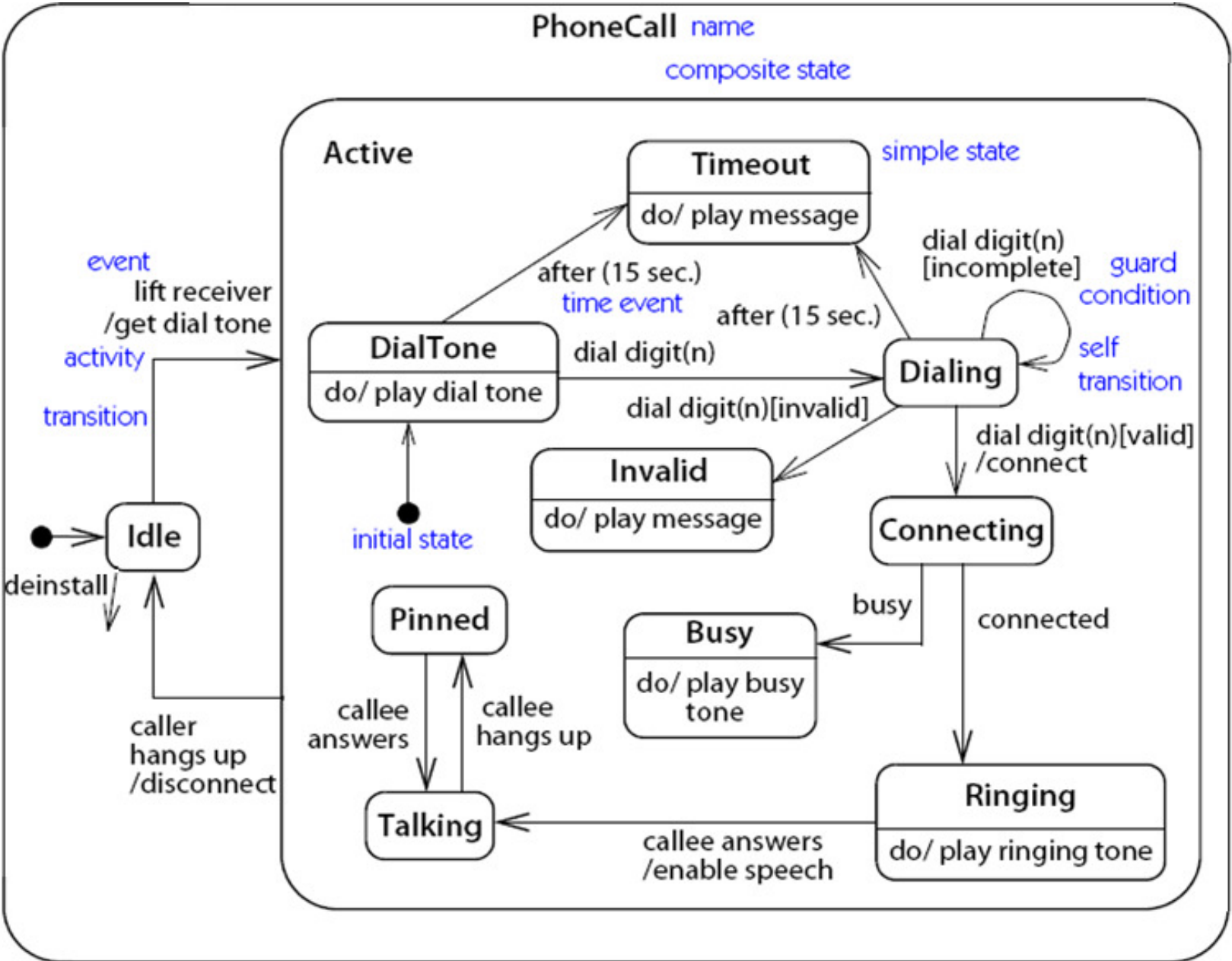
➤ **Finale**



Table 7-3: *Kinds of States*

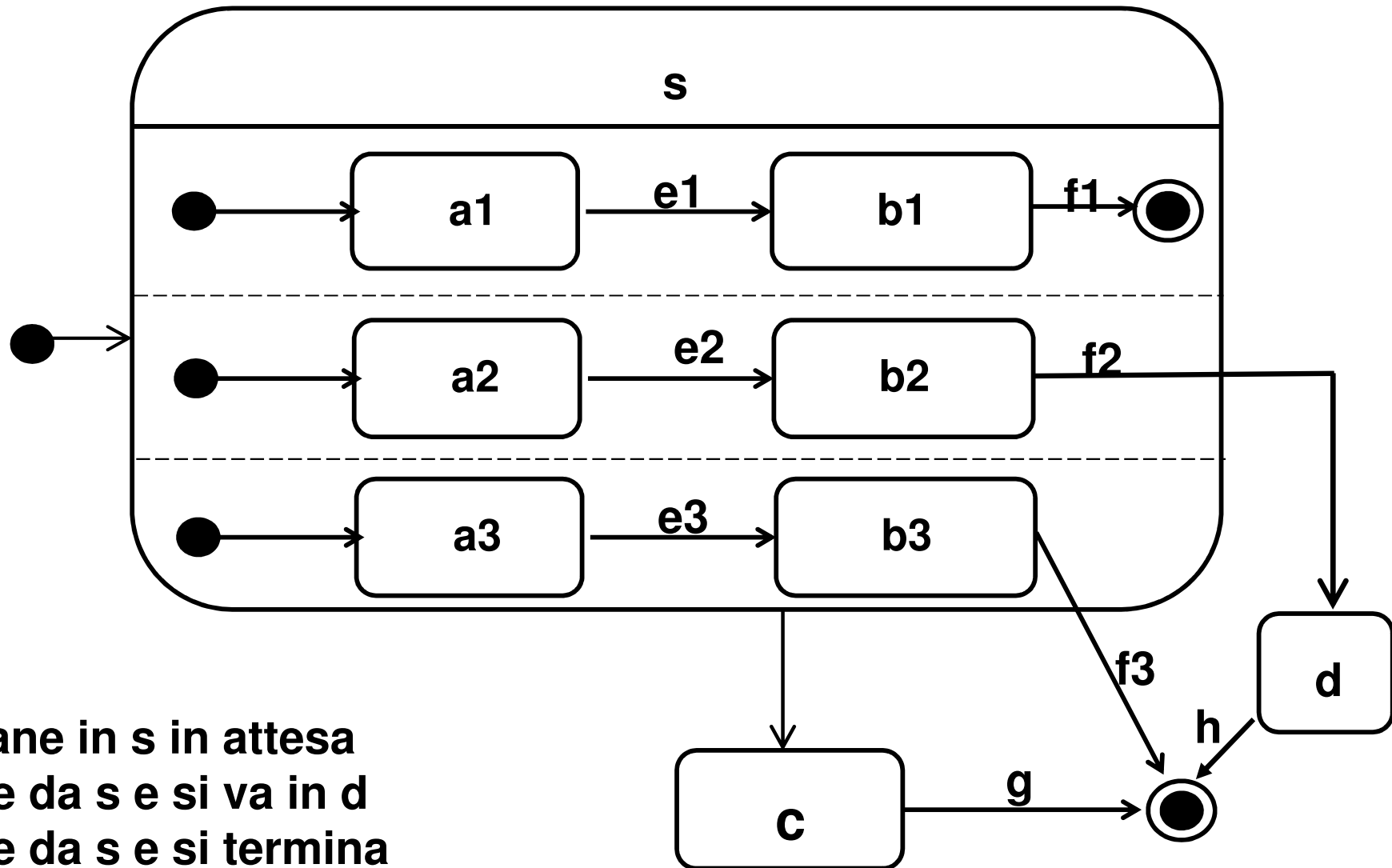
| <i>State Kind</i> | <i>Description</i> | <i>Notation</i> |
|---------------------|---|---|
| simple state | A state with no substructure |  |
| orthogonal state | A state that is divided into two or more regions. One direct substate from each region is concurrently active when the composite state is active. |  |
| nonorthogonal state | A composite state that contains one or more direct substates, exactly one of which is active at one time when the composite state is active |  |
| initial state | A pseudostate that indicates the starting state when the enclosing state is invoked |  |
| final state | A special state whose activation indicates the enclosing state has completed activity |  |
| terminate | A special state whose activation terminates execution of the object owning the state machine |  |
| junction | A pseudostate that chains transition segments into a single run-to-completion transition |  |
| choice | A pseudostate that performs a dynamic branch within a single run-to-completion transition |  |
| history state | A pseudostate whose activation restores the previously active state within a composite state |  |

Esempio



Pinned=trattenuto

Stati composti e terminazione



Dopo

f1: si rimane in s in attesa

f2: si esce da s e si va in d

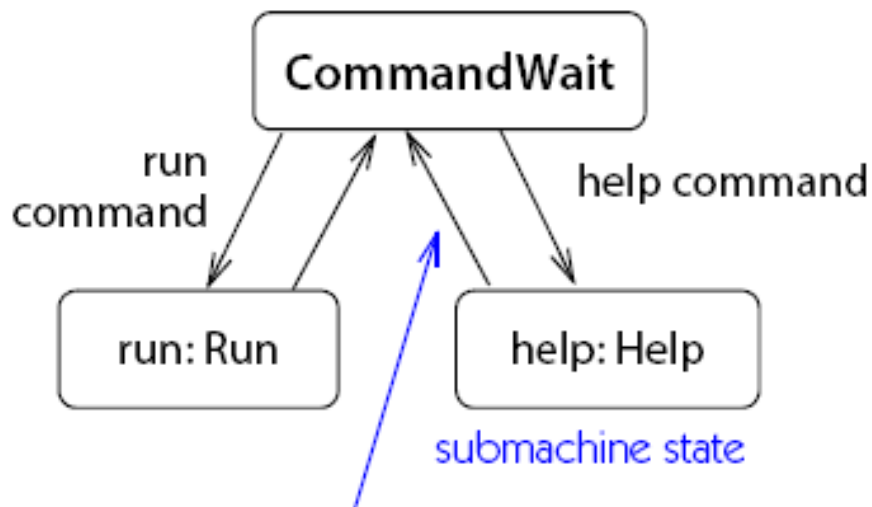
f3: si esce da s e si termina

Uscendo da s si esce anche da tutti i sottostati

Sottomacchine

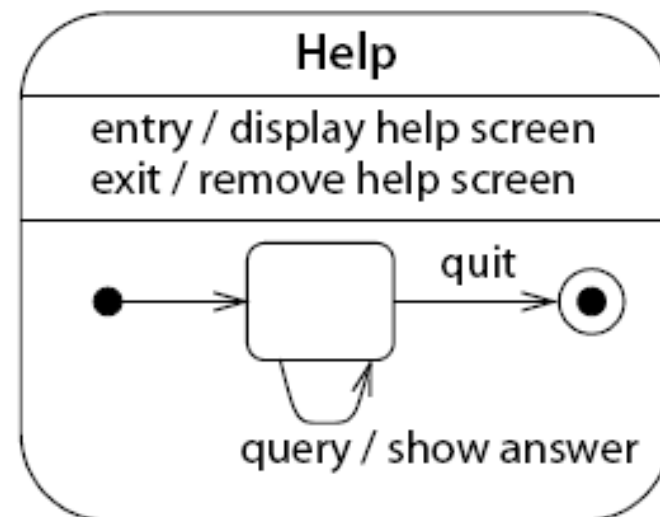
Submachine state. The invocation of a nested **submachine** is shown by a name string of the following form:

`state-name : Machine-name`



Completion transition fires when submachine completes.

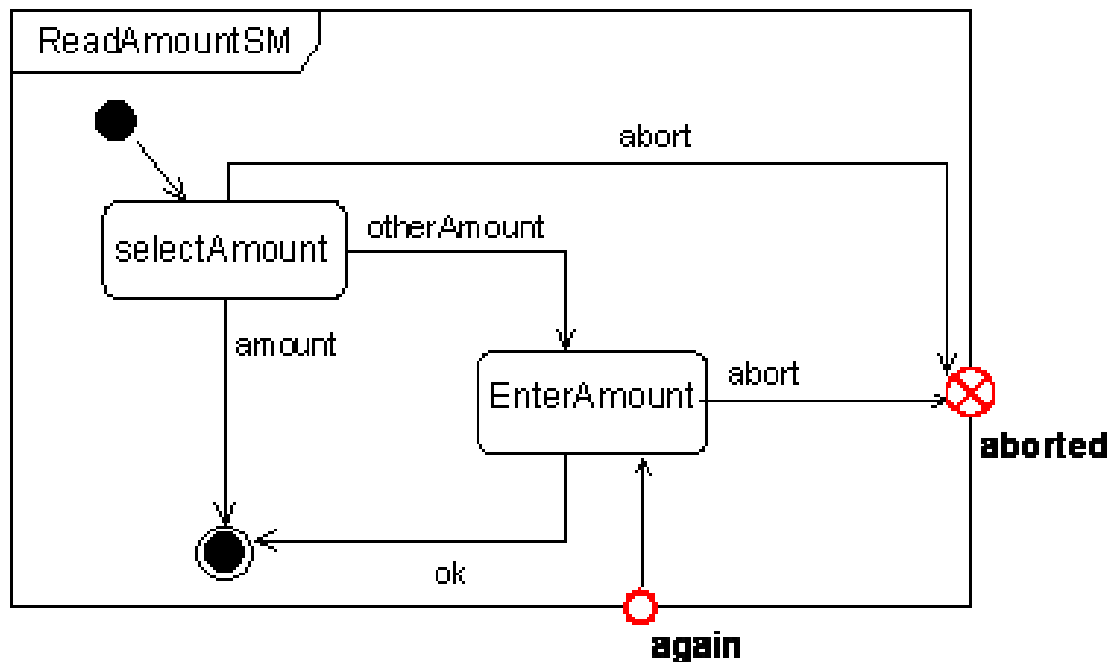
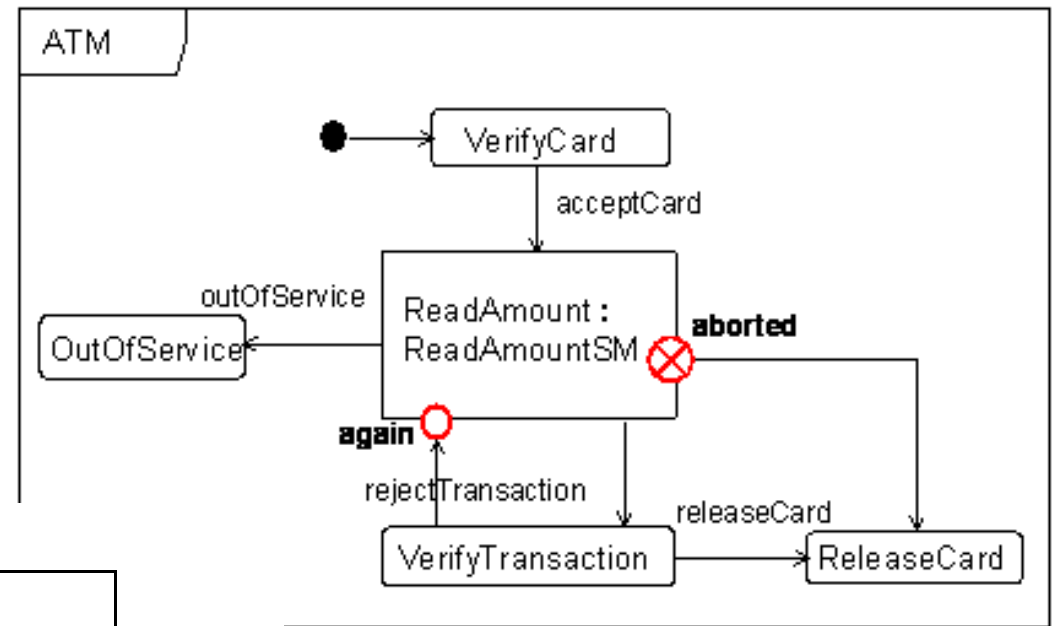
submachine definition



This submachine can be used many times.

A submachine can define named **entry points** and **exit points** that connect to internal states. Transitions to a submachine state can use **connection points** that reference these entry and exit points, hiding the internal structure

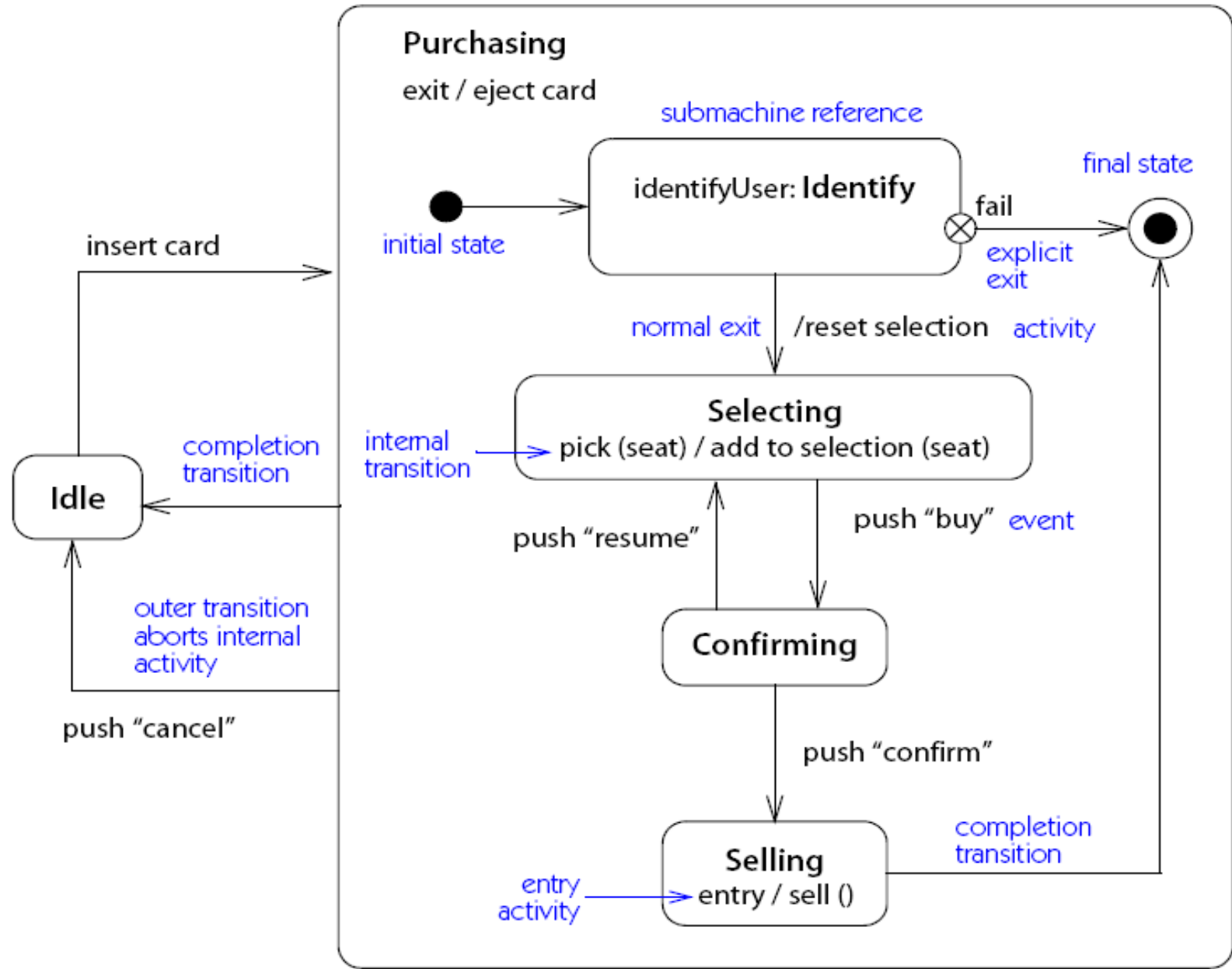
Sottomacchine: entry e exit poins



Transizioni di completamento

- Senza evento, scattano al raggiungimento
 - Della terminazione di un'attività composta, i.e. al raggiungimento
 - Dello stato finale in un stato composto non-ortogonale
 - Degli stati finali di tutte le regioni ortogonali di un stato composto
 - Del pseudo-stato di terminazione di una regione di uno stato composto ortogonale
 - Di un exit point
 - Alla terminazione di entry e/o di do activity (la exit activity viene eseguita quando scatta la transizione di completamento)
 - Di uno pseudo-stato giunzione
- Hanno priorità sugli eventi normali

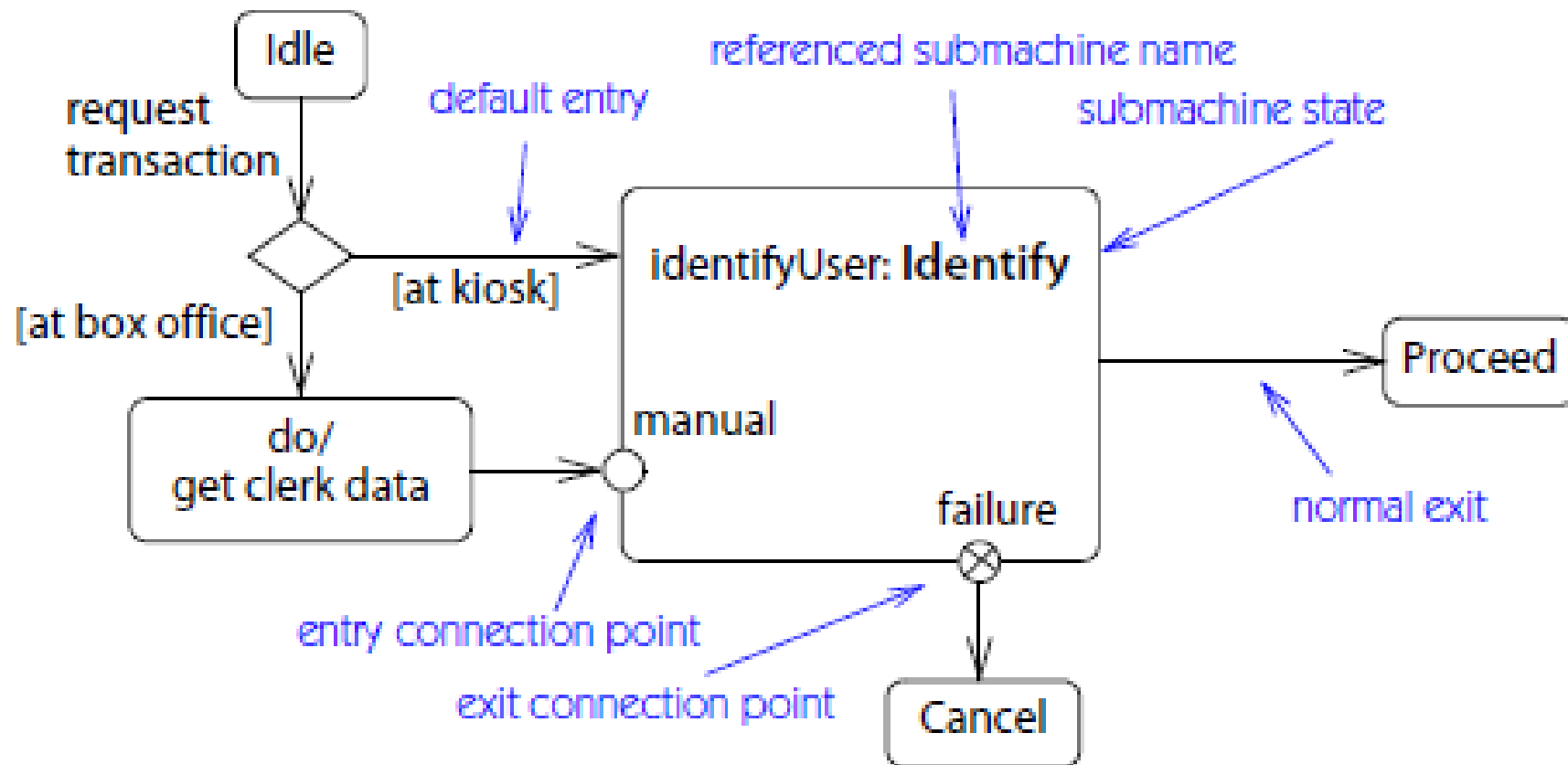
Esempio: acquisto biglietti




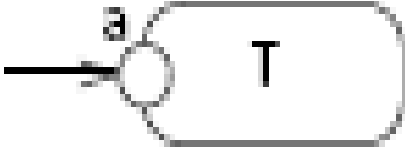
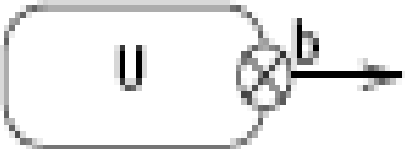
Variazione sul tema

Acquisto biglietti a clienti con account:

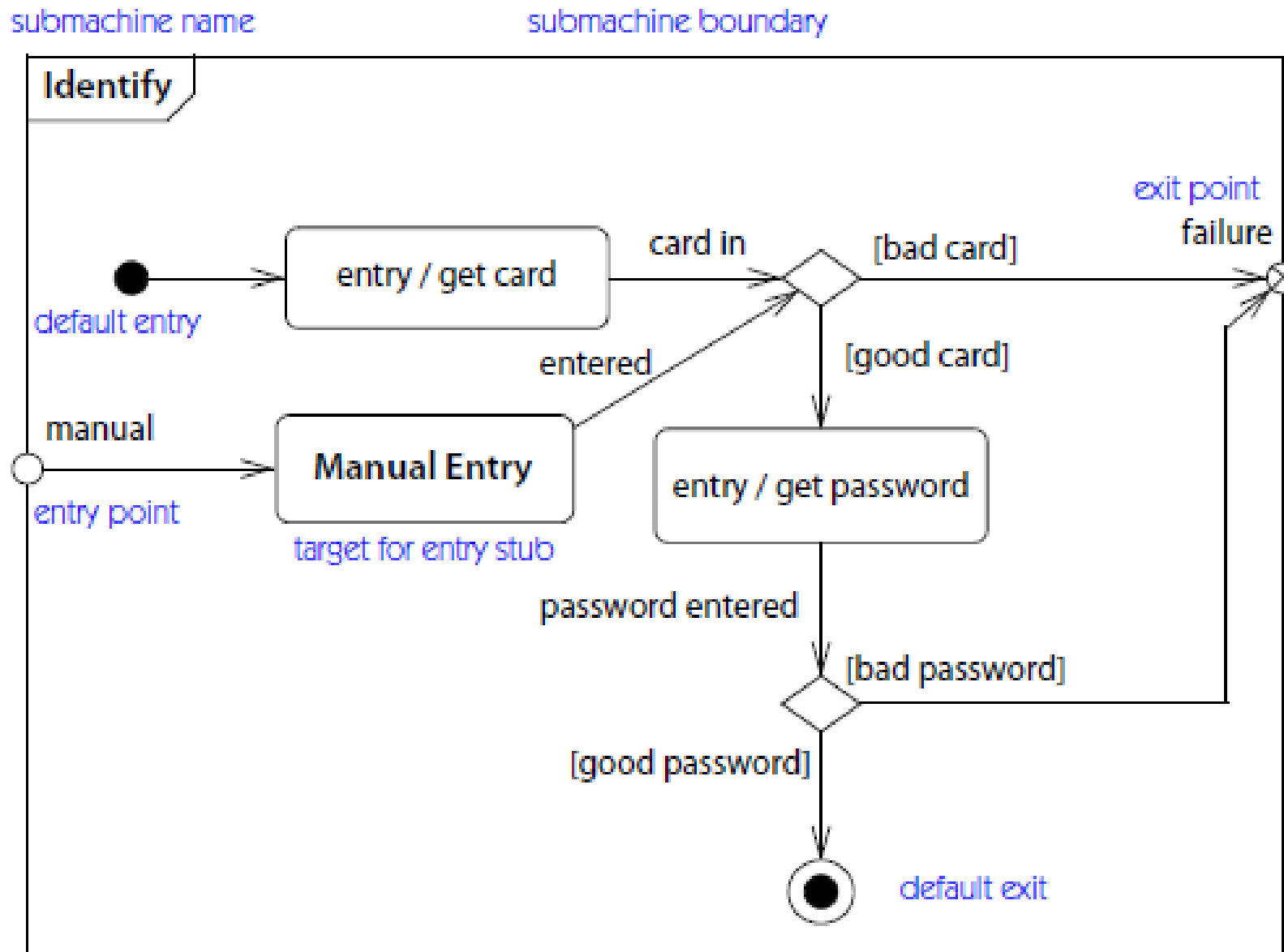
identificazione di default leggendo una carta, manuale, con inserimento nome



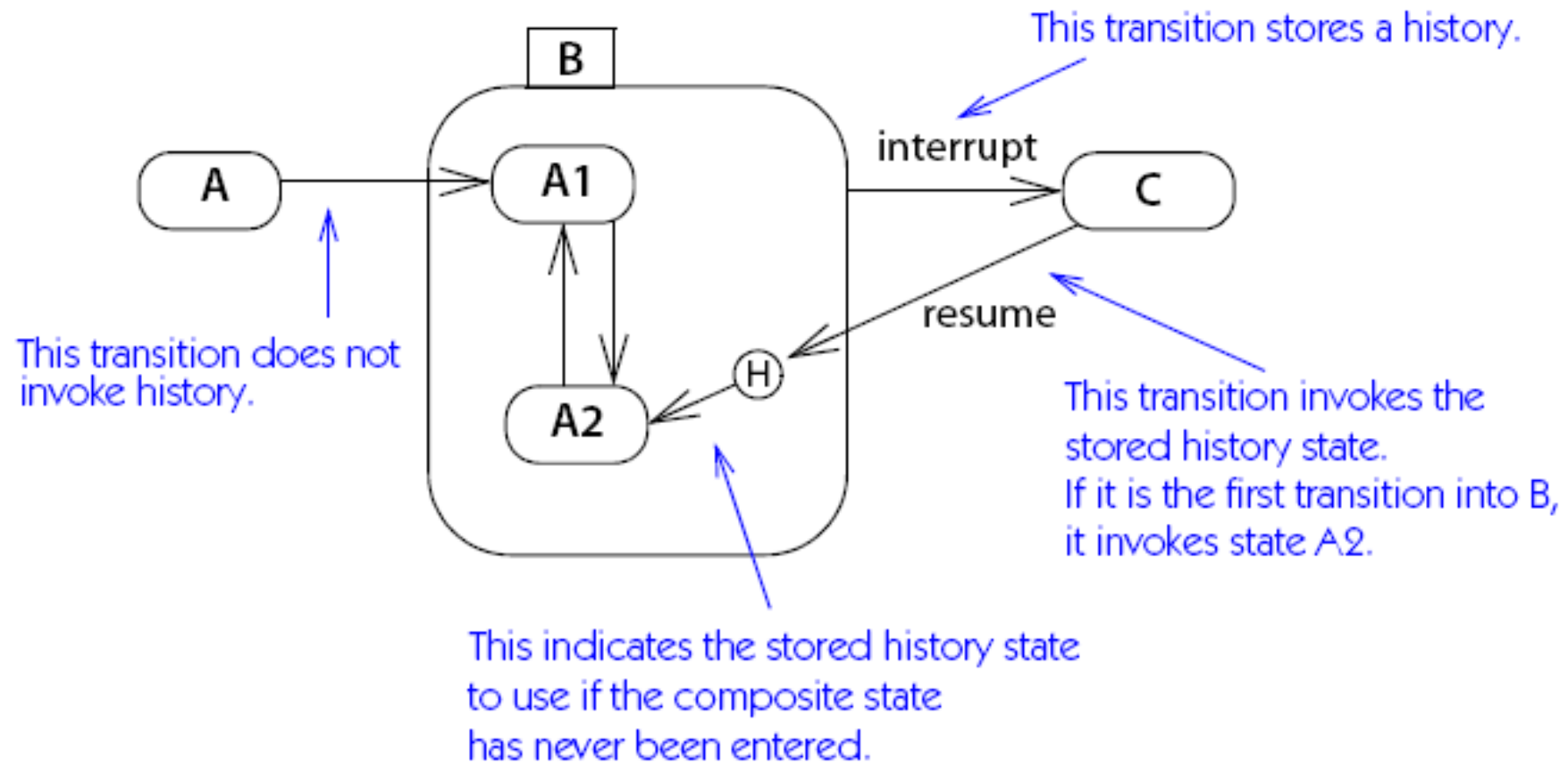
Sottomacchine

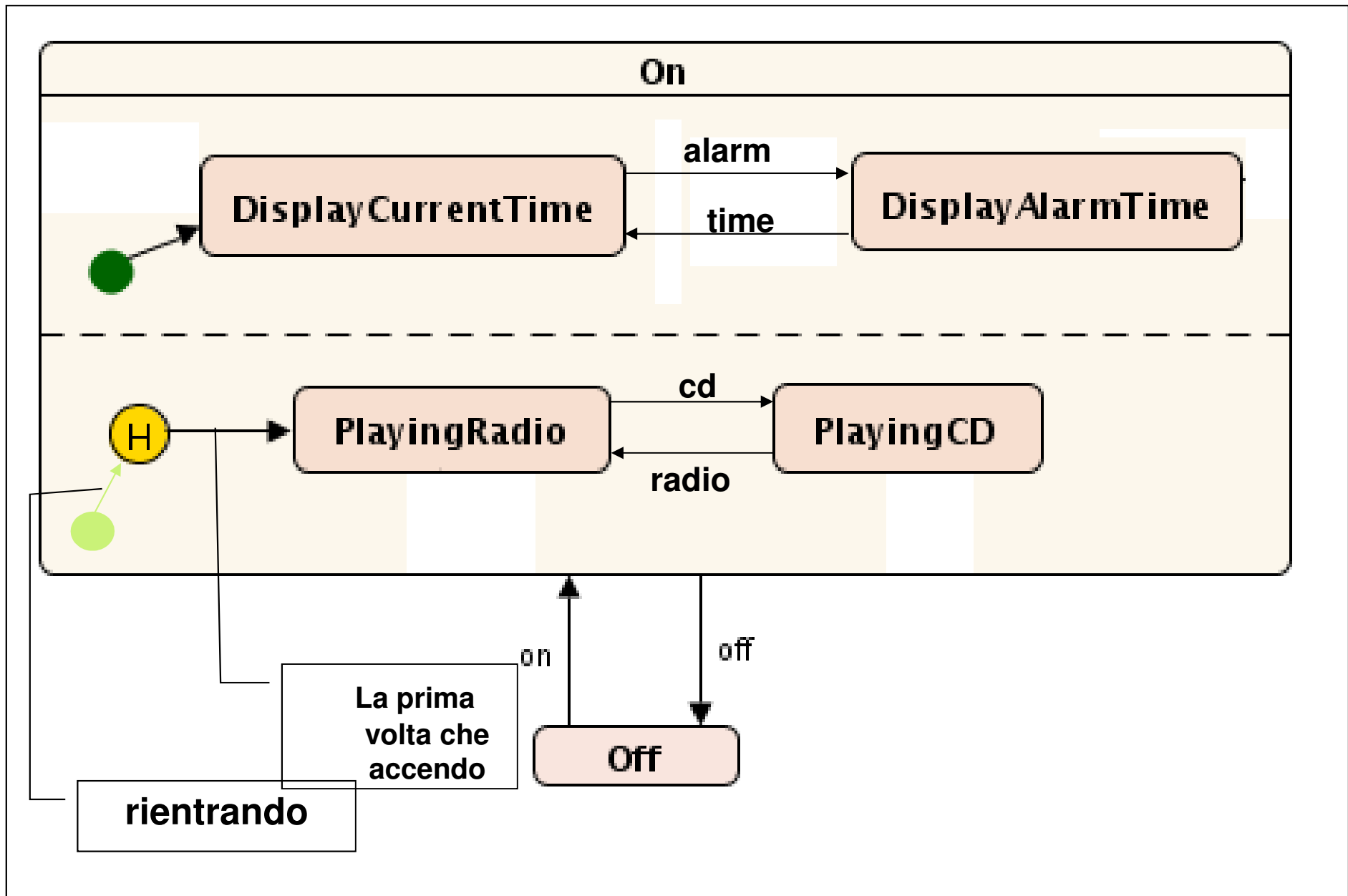
| <i>State Kind</i> | <i>Description</i> | <i>Notation</i> |
|-------------------------|---|---|
| submachine state | A state that references a state machine definition, which conceptually replaces the submachine state |  |
| entry point | A externally visible pseudostate within a state machine that identifies an internal state as a target |  |
| exit point | A externally visible pseudostate within a state machine that identifies an internal state as a source |  |

Definizione di una sottomacchina



History state





Esempio d'uso dello pseudo-stato giunzione

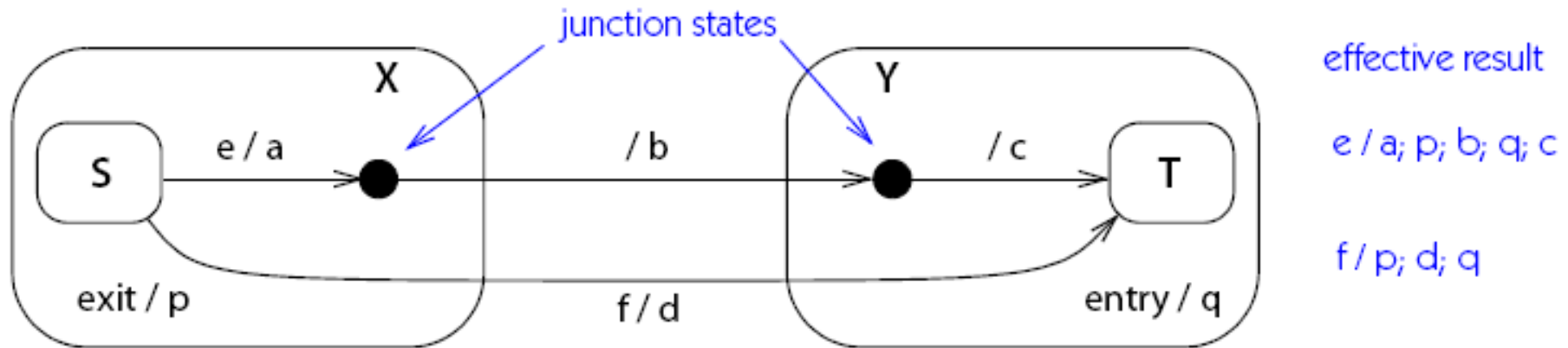


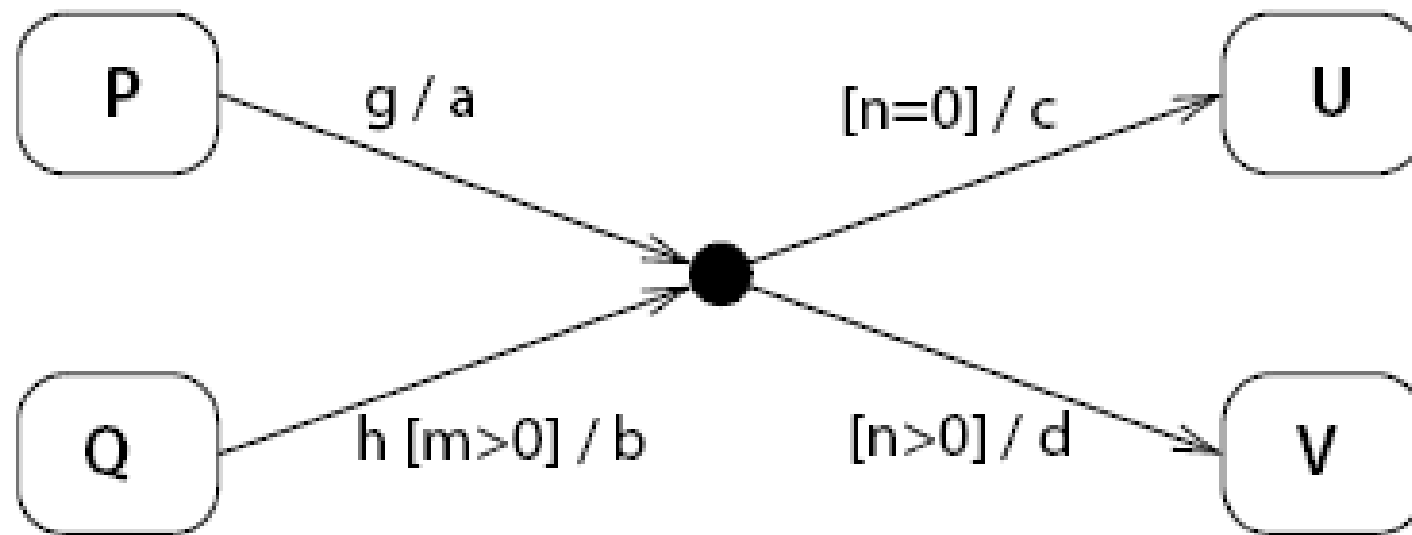
Figure 14-170. Junction pseudostates with multiple actions

Giunzione vs choice

- Giunzione statica
- Choice dinamica

Note that a **choice** pseudostate also connects multiple segments into a **compound transition**, but it has different rules for guard conditions. When a path contains a choice vertex, the guard conditions on the path before the choice vertex are evaluated before the transition fires. Guard conditions on the path after the choice vertex are evaluated dynamically after any actions on the initial segments have been performed. The modeler must guarantee that a valid subsequent path will then exist. Unless one segment from each choice vertex contains an else condition or can otherwise be guaranteed to cover all possible results, there is the danger of an invalid execution scenario. There is no such danger with a junction vertex because all of the conditions are evaluated in advance before the transition fires. Junction vertices and choice vertices each have their place.

Esempio di Giunzione



Equivalent transitions:

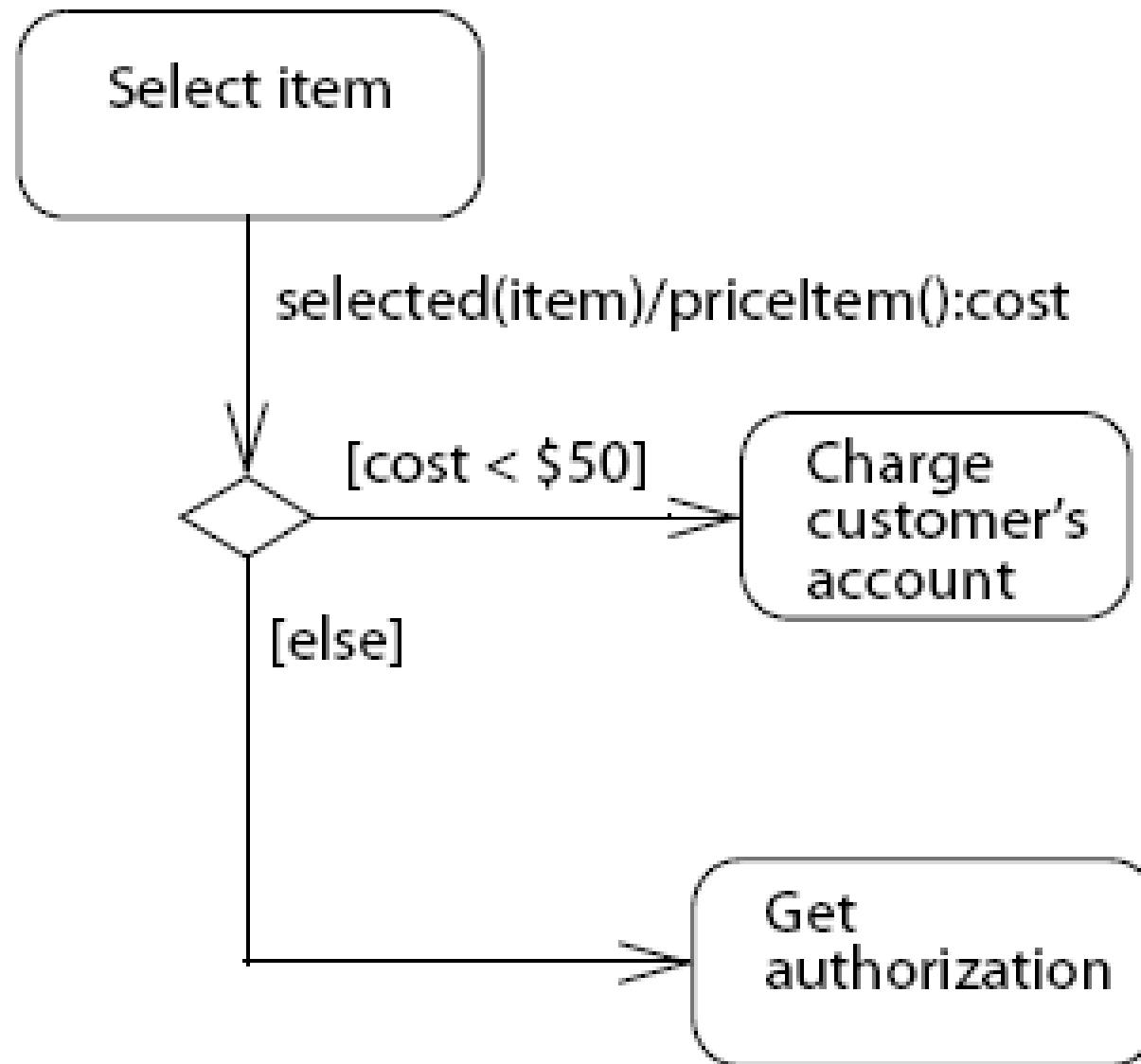
$P g [n = 0] / a; c U$

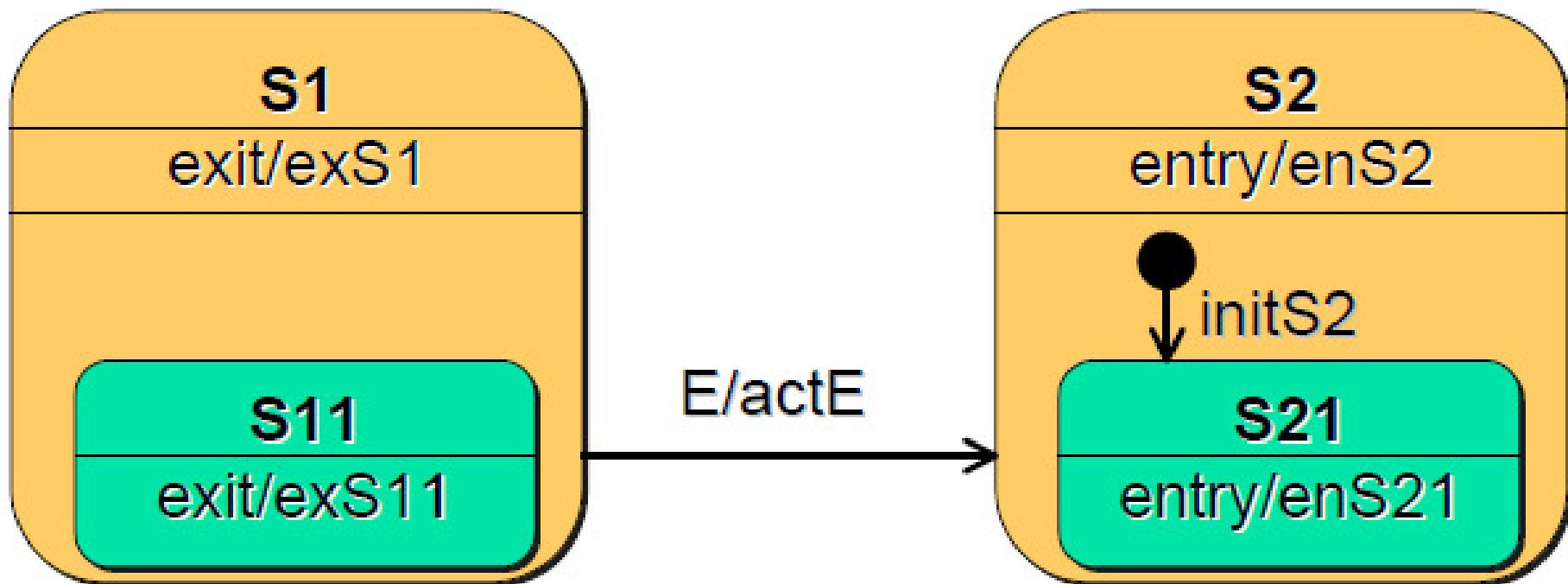
$P g [n > 0] / a; d V$

$Q h [m > 0 \text{ and } n = 0] / b; c U$

$Q h [m > 0 \text{ and } n > 0] / b; d V$

Esempio di choice





Actions execution sequence:

exS11 ⇒ exS1 ⇒ actE ⇒ enS2 ⇒ initS2 ⇒ enS21

Esercizi "per casa"

- Diagramma di macchina a stati
 - della classe Utente (o Situazione Utente) di Myair
 - del semaforo (ex. Semafori)
- Diagramma attività
 - EasyPark
 - Myair

Syllabus

- Arlow:
 - Paragrafi 16.3, 16.4, 16.5
 - Cap 13 tranne 13.10
 - Arlow cap 19