Macchine di stato in UML

H1 Introduzione

Lo ***Statechart UML*** (**macchina di stato**) è solo uno dei diversi tipi di diagrammi UML che consentono **modellazione dinamica**.

H2

Una **macchina di stato** è *associata a una classe* e descrive il *comportamento* di un oggetto della classe.

Definizione

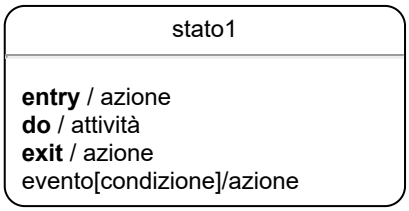
> Una **macchina di stato** è un grafo i cui nodi rappresentano gli **stati** in cui può trovarsi un oggetto ed i cui archi rappresentano le **transizioni** tra gli stati

H2

H3 Stato

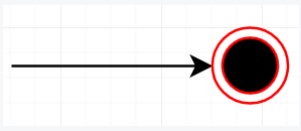
> Uno **stato** rappresenta una condizione di esistenza dell'oggetto:
>
> - attesa di un **evento**
> - esecuzione di certe **azioni** o **attività**, dove
>   - **azione** : operazione atomica e non interrompibile
>   - **entry**: azione eseguita entrando nello stato
>   - **exit**: azione eseguita uscendo dallo stato
>   - **attività** : operazione che richiede un certo tempo

Uno **stato** generico sarà così rappresentato:



Come si nota è possibile indicare una **transizione interna** con la dicitura `evento[condizione]/azione` , cioè una transizione che non fa mutare lo stato dell'oggetto. La stessa dicitura viene usata per le **transizione esterne**, indicate sopra gli archi, che al contrario implicano un cambiamento di stato.

Sono inoltre presenti degli **pseudo-stati**, tra cui:

	
stato iniziale	***stato finale***

H3 Transizione

> Una **transizione** (se **esterna**) indica un *passaggio di stato*. Essa è etichettata con le seguenti informazioni:
>
> - **eventi** che comportano il cambiamento di stato e le **condizioni** (**guardia**) sotto cui si verificano
> - le **azioni** che l'oggetto esegue prima di cambiare stato

Operativamente, diciamo che:

- una **transizione** *scatta* quando occorre un **evento** (interno o esterno) e l'eventuale **guardia** sulla transizione è vera

Una transizione può essere:

- **interna** poiché lo stato non cambia; viene indicata all'interno dello stato
- **esterna** poiché lo stato cambia; viene indicata sopra gli archi

Una generica **transizione esterna** con un arco con la dicitura: `evento(parametri)[condizione]/azione`

H3 Evento

Gli **eventi** che fanno scattare le **transizioni** sono degli eventi ben definiti e suddivisi nelle seguenti categorie:

- **Interazioni con gli oggetti**

    - **Call event** : si verifica nel momento in cui un oggetto invoca un'operazione propria o di un altro oggetto (spesso presente nei **sistemi distribuiti**)
    - **Signal event** : si verifica nel momento in cui un oggetto riceve un *segnale* da un altro oggetto o dall'ambiente
- **Occorrenza di istanti di tempo**
    - **Time event** : intervallo temporale, data o timestamp
        - `after` seguito da una quantità temporale (es. `after(10)`)
- **Cambiamento del valore**
    - **Change event** : indicato con `when` seguito da un'espressione booleana (es. `when(bilancio < 0)`)
        - L'espressione booleana viene testata quando variano i parametri coinvolti

Un ulteriore caso è l'**evento di completamento**, cioè l'evento che viene *implicitamente* generato al completarsi di un'**attività** dell'oggetto. Per indicare la transizione generata da un evento di completamento si usa un *arco vuoto*.

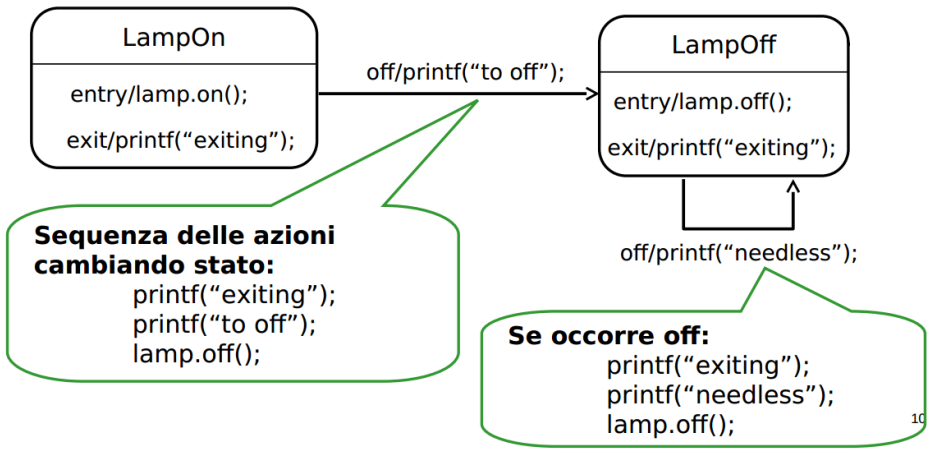
Ordine delle azioni

Le **azioni** verranno svolte in base al loro posto nel modello:

H4

- Le azioni **exit** precedono quelle della transizione
- Le azioni **entry** seguono quelle della transizione

Nel momento in cui avviene una transizione esterna avremo quindi il seguente ordine: `exit -> transizione -> entry`



Nell'esempio vediamo l'ordine delle azioni descritto sopra: `printf("exiting") -> printf("to off") -> lamp.off()`, che sono rispettivamente l'azione di exit dello stato `LampOn`, l'azione della transizione e infine l'azione di entry dello stato `LampOff`.

In questo esempio è inoltre possibile vedere un caso di applicazione di una transizione esterna che rientra nello stesso stato ("*cappio*"). Abbiamo quindi sia questa soluzione che la **transizione interna**. Qual è la differenza?

- "cappio" - in risposta ad un evento che fa scattare la transizione:
  - si esegue l'azione di exit
  - si esegue l'azione della transizione
  - si esegue l'azione di entry
- transizione interna - in risposta ad un evento che fa scattare la transizione:
  - si esegue l'azione della transizione

Sono quindi approcci simili ma che modellano situazioni differenti, e pertanto è utile usare entrambi.

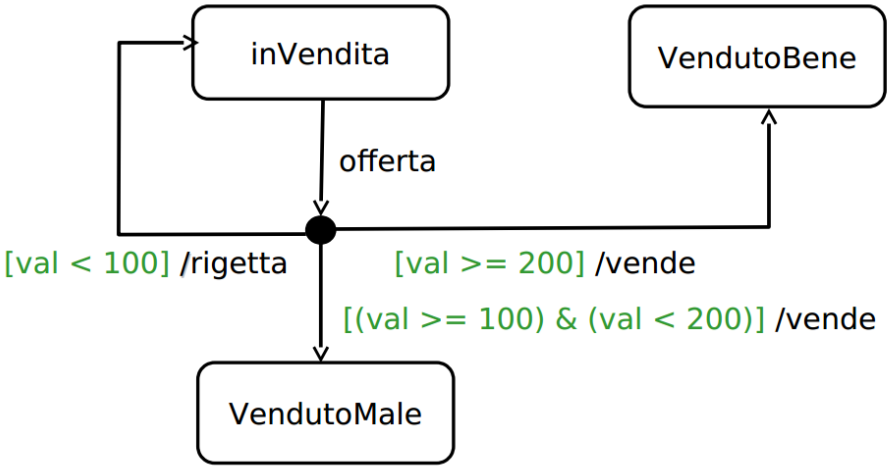
H3 # Conditional Branching

Introduciamo ora due **pseudo-stati** che consentono di prendere decisioni su quale sarà lo stato di arrivo di una transizione. Essi sono:

- ***Junction pseudostate***
- ***Choice pseudostate***

Static Conditional Branching

H4 > Viene rappresentato con il ***junction pseudostate***, e in questo caso il valore valutato dalle guardie è già disponibile.

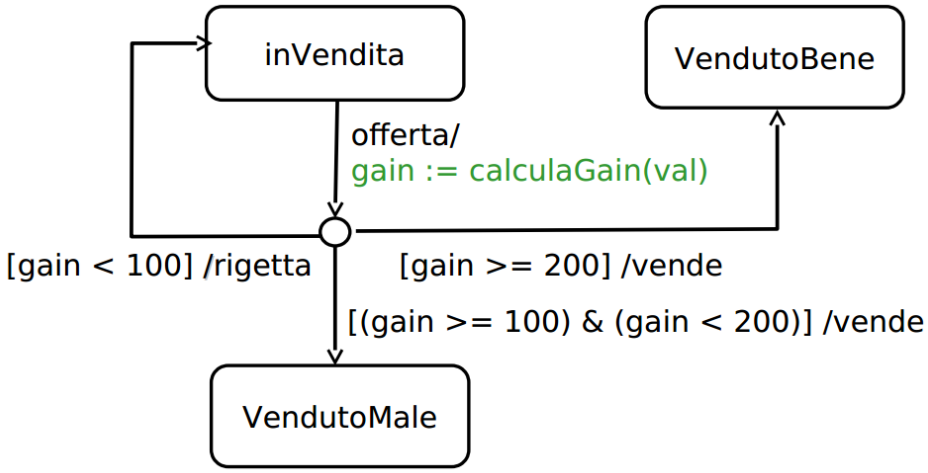


In questo esempio la variabile `val` è già pronta per essere valutata.

Dynamic Conditional Branching

H4

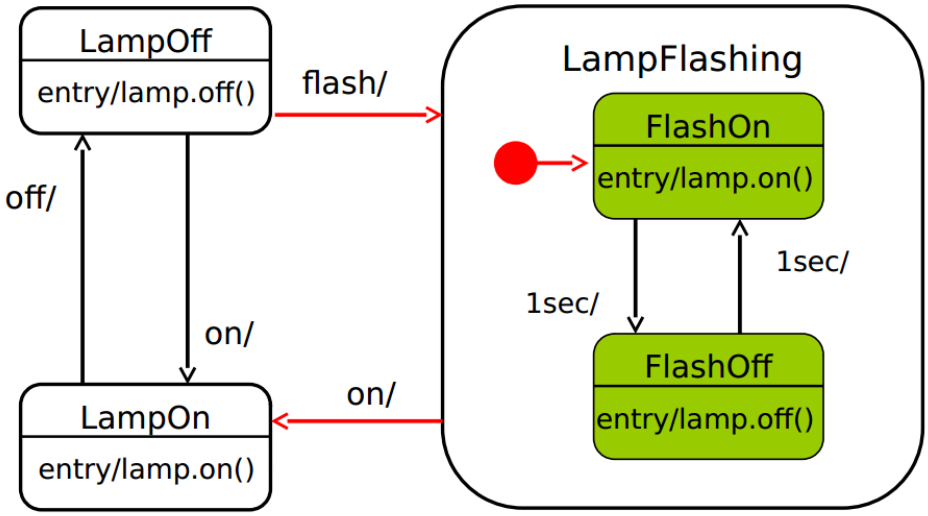
> Viene rappresentato con il ***choice pseudostate***, e in questo caso il valore testato dalle guardie viene calcolato quando viene raggiungo il punto di decisione.



In questo esempio la variabile `val` viene passata all'azione `calcolaGain()`, la quale salva il risultato nella variabile `gain`, e sarà quest'ultima a essere valutata dalle guardie per la decisione. Il risultato è un *"ritardo"* della decisione rispetto al branching statico.

H3 # Sottomacchine

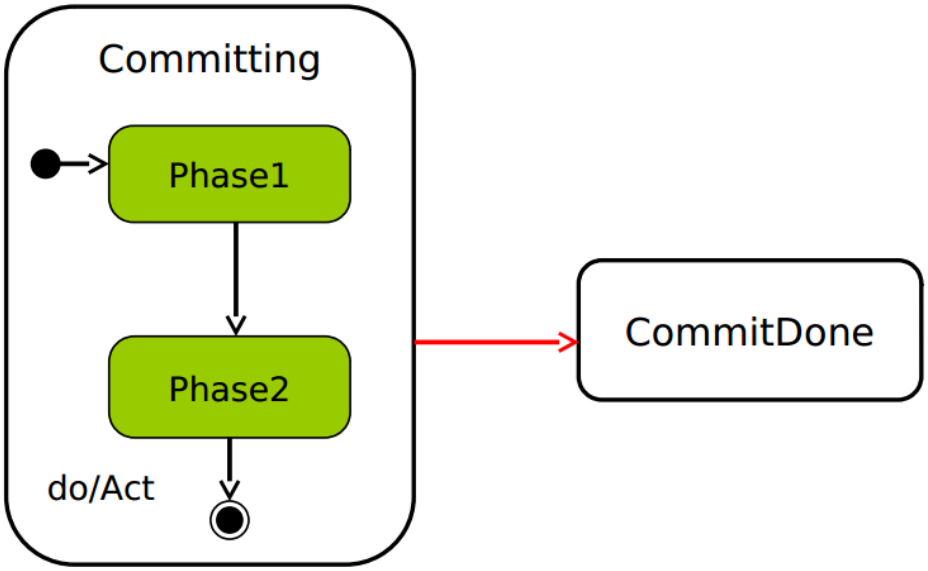
È presente nello standard UML la possibilità di introdurre nel modello degli **stati composti**, cioè degli stati che contengono delle **sottomacchine**.



In questo esempio vediamo come a questa lampada viene introdotto uno stato di luce intermittente. Per modellarlo, usiamo una sottomacchina. In particolare, se avviene il `<<signal>>flash/` (*signal event*), la lampada entra nello **stato composto** `LampFlashing`, il quale ha `FlashOn` come *stato iniziale*, dopodiché ogni secondo gli stati `FlashOn` e `FlashOff` si alternano.

Evento di completamento

Abbiamo già visto l'evento di completamento come evento generato implicitamente al termine di un'attività di uno stato. Abbiamo un evento di
H4 completamento anche quando una sottomacchina termina.

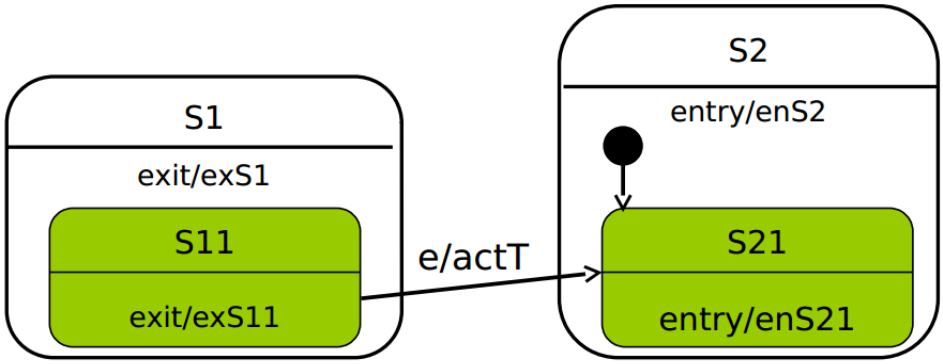


Ordine delle azioni

In che ordine vengono eseguite le azioni quando sono coinvolti **stati composti**?

H4

- Se usciamo da uno sottostato, attraversiamo lo stato esterno prima di cambiare stato
- Se entriamo in un sottostato, attraversiamo lo stato esterno prima di entrare nel sottostato



In questo esempio, la sequenza di azioni per la transizione che va da `S11` a `S21` è la seguente:
`exS11 -> exS1 -> actT -> enS2 -> enS21`

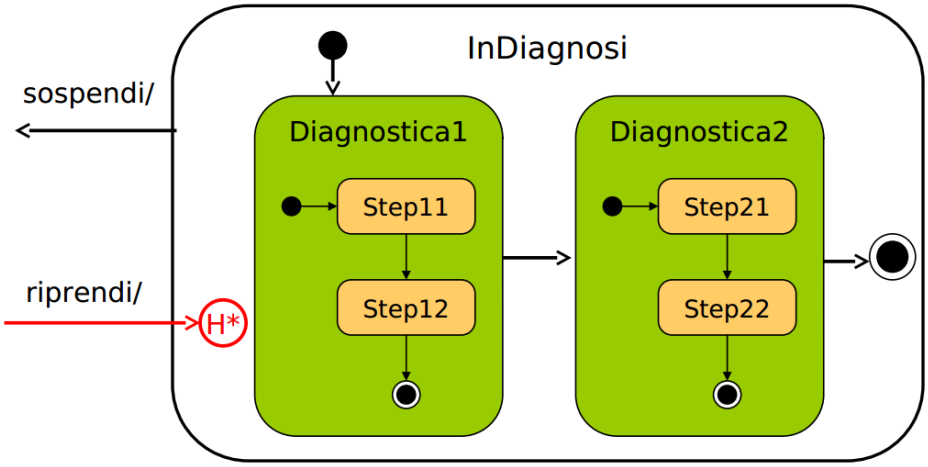
Ordine delle transizioni

Vi è la possibilità che due o più transizioni siano abilitate dallo *stesso evento* nello *stesso istante*.

H4 Si decide quindi che le *transizioni più interne* hanno priorità.

Stati history

Può capitare di modellare quei casi in cui vogliamo indicare che una sottomacchina all'interno di uno stato abbia la possibilità di ripartire da
H4 dove era stata sospesa. Per far ciò si usano gli pseudo-stati **history**, i quali vengono posizionati all'interno degli stati contenenti la sottomacchina.



In questo esempio vediamo che inserendo lo pseudo-stato history (cerchio rosso con `H`), nel momento in cui avviene l'evento `riprendi`, la sottomacchina riprenderà da dove aveva interrotto quando era avvenuto l'evento `sospendi`.

H3 Stati composti paralleli