Azzolini Riccardo 2019-11-21

State diagram

1 Automi a stati finiti

Gli oggetti sono **macchine/automi a stati finiti**, in quanto caratterizzati da un numero finito di possibili stati e dalle transizioni tra di essi.

Ad esempio, una pila può essere vuota, parzialmente riempita, o piena, e passa da uno stato all'altro con l'aggiunta/rimozione di un elemento.

2 Notazione

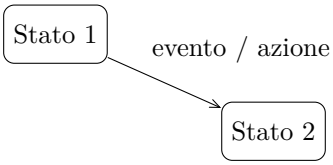
Gli **state diagram** UML sono diagrammi *dinamici* che permettono di descrivere automi a stati finiti, usando una notazione grafica derivata da una proposta precedente (StateCharts). I principali elementi di tale notazione sono:

- stato:

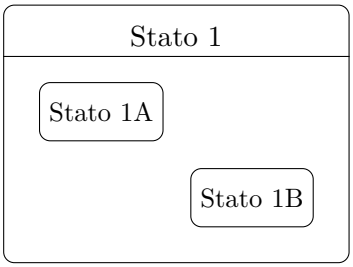
- stato iniziale:

- stato finale:

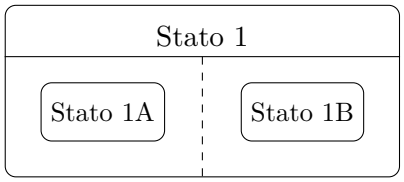
- transizione:



- decomposizione OR:

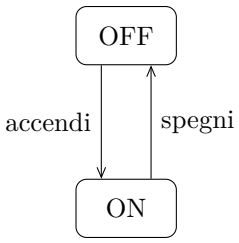


- decomposizione AND:



3 Stati

Uno **stato** rappresenta una situazione durante la quale è verificata una condizione (che è solitamente "implicita", cioè non specificata esplicitamente, ma comunque indicata implicitamente dal nome dello stato stesso). Tale condizione può corrispondere a una situazione statica, come l'attesa di un evento esterno, o, più in generale, a una qualsiasi situazione stabile, come lo svolgimento di un'attività.



Lo stato di un oggetto è l'insieme dei valori dei suoi attributi e link in un certo istante. Uno stato è però **astratto**, in quanto può corrispondere a diverse, anche infinite, combinazioni di valori degli attributi/link. Ad esempio, per una pila, lo stato "parzialmente riempita" corrisponde a diversi numeri di elementi (e valori di tali elementi): esso viene comunque trattato come un unico stato perché interessa solo sapere che si possono aggiungere e rimuovere elementi (a differenza di una pila completamente piena o vuota). Un altro esempio è un motore, il cui stato "acceso" potrebbe corrispondere a diversi numeri di giri al minuto.

Lo stato di un oggetto ne influenza il comportamento: la stessa operazione può avere effetti/risultati diversi, se eseguita in stati diversi. Ad esempio, l'inserimento in una pila vuota

o parzialmente riempita ha l'effetto di aggiungere un elemento, mentre l'inserimento in una pila piena produce un errore.

3.1 Durata degli stati

Uno stato perdura nel tempo (cioè ha una durata, a differenza di una transizione), finché un evento non fa cambiare stato all'oggetto. Ad esempio:

- una lampadina rimane nello stato "OFF" finché non viene accesa, dopo di che passa allo stato "ON" (e si assume, per astrazione, che la transizione sia istantanea, senza durata);
- una pila rimane vuota finché non si verifica un inserimento, e allora diventa parzialmente riempita.

Uno stato, però, può anche rappresentare lo svolgimento di un'attività. In questo caso, l'oggetto lascia tale stato appena l'attività viene completata, senza bisogno di un evento esterno.

4 Eventi

Un **evento** è uno *stimolo esterno*:

- nel caso di un oggetto, è una chiamata di un metodo estensore (*set*), che può alterare i valori delle variabili interne dell'oggetto;
- può provocare una transizione di stato (ma non è detto che ciò avvenga in seguito a tutti gli eventi).

5 Identificazione degli stati

Per costruire uno state diagram relativo a uno o più oggetti, è necessario identificare tutti gli stati in cui ciascuno di essi si può trovare. Alcuni suggerimenti utili sono:

- trascurare gli attributi ininfluenti, che non determinano il comportamento dell'oggetto (ad esempio, i valori degli elementi contenuti in una pila);
- definire un livello di astrazione corretto (ad esempio, per una pila, è più utile considerare solo gli stati "vuota", "parzialmente riempita", e "piena", piuttosto che "0 elementi", "1 elemento", "2 elementi", ecc.).

6 State diagram

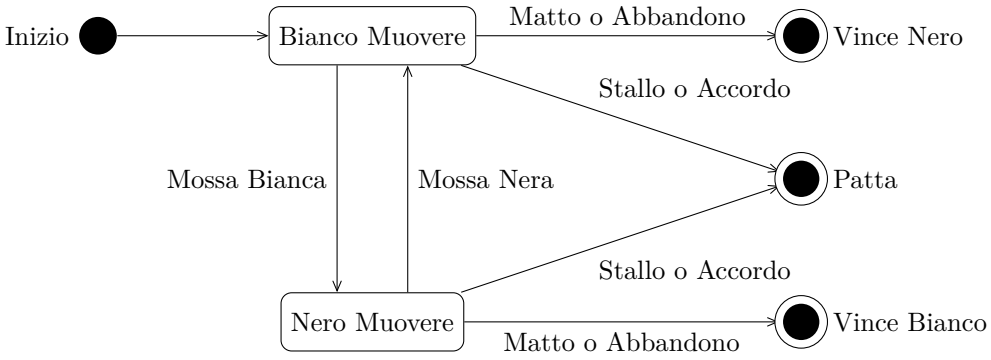
Uno state diagram rappresenta il comportamento di una classe di oggetti, descrivendone:

- i possibili stati;
- la reazione a eventi esterni, in termini di cambiamenti di stato (transizioni) e/o azioni svolte.

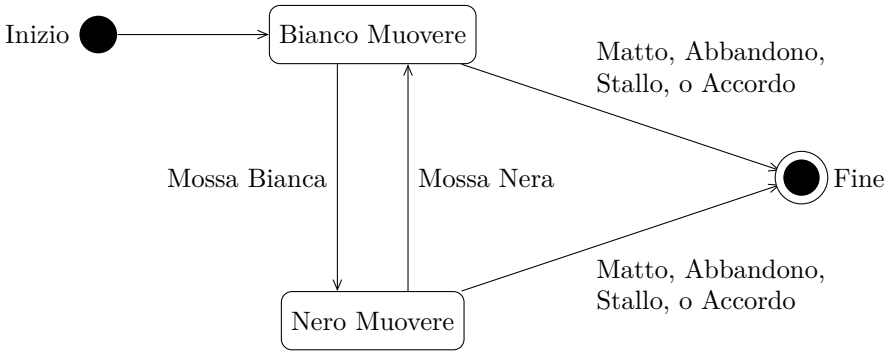
Una macchina a stati, però, può essere usata anche per descrivere il comportamento di una parte del sistema che non è un oggetto.

7 Stati iniziali e finali

Lo stato iniziale è unico, ma possono invece essere presenti più stati finali. Ciò permette di rappresentare informazioni aggiuntive all'interno del diagramma: ad esempio, per una partita di scacchi, usando stati finali multipli si può specificare il vincitore della partita,



mentre con un singolo stato finale si indicherebbe solo che la partita è finita.



8 Condizioni

Oltre a un evento, per una transizione è possibile specificare una **condizione** booleana: quando avviene l'evento, si ha la transizione solo se tale condizione è verificata.

A differenza degli eventi, che sono istantanei, queste condizioni sono *condizioni di stato*, in quanto sono valide per un certo intervallo di tempo.

Nelle condizioni si possono utilizzare anche variabili di altri oggetti. Esse sono quindi uno dei meccanismi che gli state diagram mettono a disposizione per modellare l'interazione tra oggetti diversi (ad esempio, nel diagramma precedente, si ha un'interazione tra l'oggetto automobile e l'oggetto incrocio).

9 Operazioni

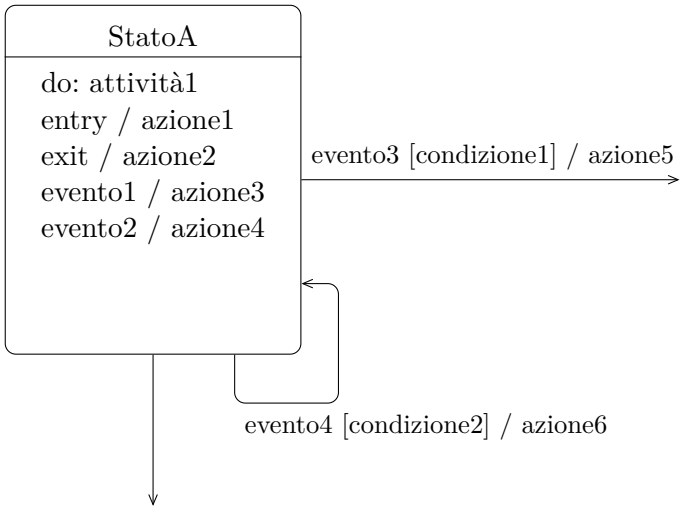
Durante la loro vita, gli oggetti eseguono **operazioni**, che si dividono in:

azioni: sono associate alle transizioni (oppure all'ingresso/uscita da uno stato), e si assume che non abbiano durata, cioè che siano istantanee;

attività: sono associate agli stati, e durano tanto quanto gli stati a cui corrispondono.

Da uno stato al quale è associata un'attività, può uscire una freccia senza eventi: essa indica una **transizione automatica**, che viene svolta automaticamente al completamento dell'attività, senza bisogno di un evento esterno.

9.1 Notazione

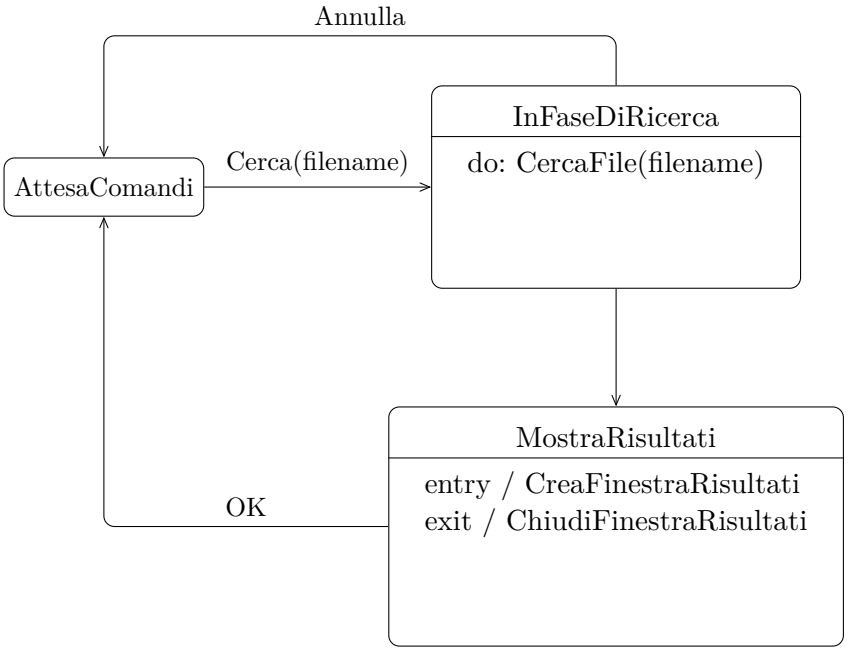


- *do* specifica l'attività associata allo stato (attività1);
- Oltre all'attività, all'interno dello stato sono indicate anche varie azioni:
  - *entry* ed *exit* specificano delle azioni che vengono sempre eseguite quando, rispettivamente, si entra e si esce dallo stato (in questo caso, azione1 e azione2);
  - le restanti azioni (azione3 e azione4) vengono eseguite in risposta a eventi esterni (evento1 ed evento2), i quali però non provocano transizioni di stato.
- Sono specificate varie transizioni:
  - quando l'attività1 termina, si esce da StatoA, seguendo la transizione automatica indicata dalla freccia non etichettata (e viene eseguita l'azione di *exit*, cioè azione2);
  - quando si verifica evento3, ed è soddisfatta condizione1, si esce da StatoA e viene eseguita azione5 (dopo aver eseguito azione2, l'azione di *exit*);
  - se si verifica evento4, ed è soddisfatta condizione2, si ha una **pseudo transizione**: lo stato non cambia "veramente", ma vengono eseguite, in ordine, azione2 (*exit*), azione6, e azione1 (*entry*).

Osservazione: Un'azione associata a un evento che non provoca un cambiamento di stato può essere scritta:

- all'interno dello stato;
- come pseudo transizione, ma in questo caso vengono eseguite anche le (eventuali) azioni di exit ed entry (che, a seconda della situazione, può essere o meno l'effetto desiderato).

9.2 Esempio



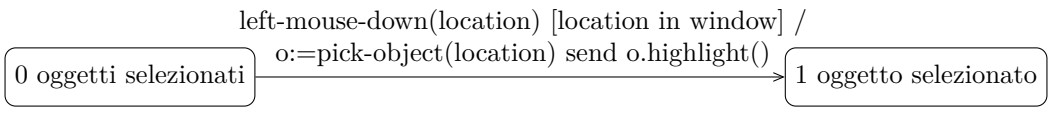
10 Eventi generati da azioni

Spesso, le azioni consistono nell'inviare un evento a un altro oggetto. Ciò si indica con la parola chiave *send*:

evento(arg) [cond] / azione send ogg.evento(arg)

Questo è un altro meccanismo per modellare le interazioni tra oggetti diversi. È quindi possibile decomporre un sistema in più automi, che hanno vita apparentemente indipendente, ma in realtà interagiscono mediante l'invio di eventi.

10.1 Esempio 1

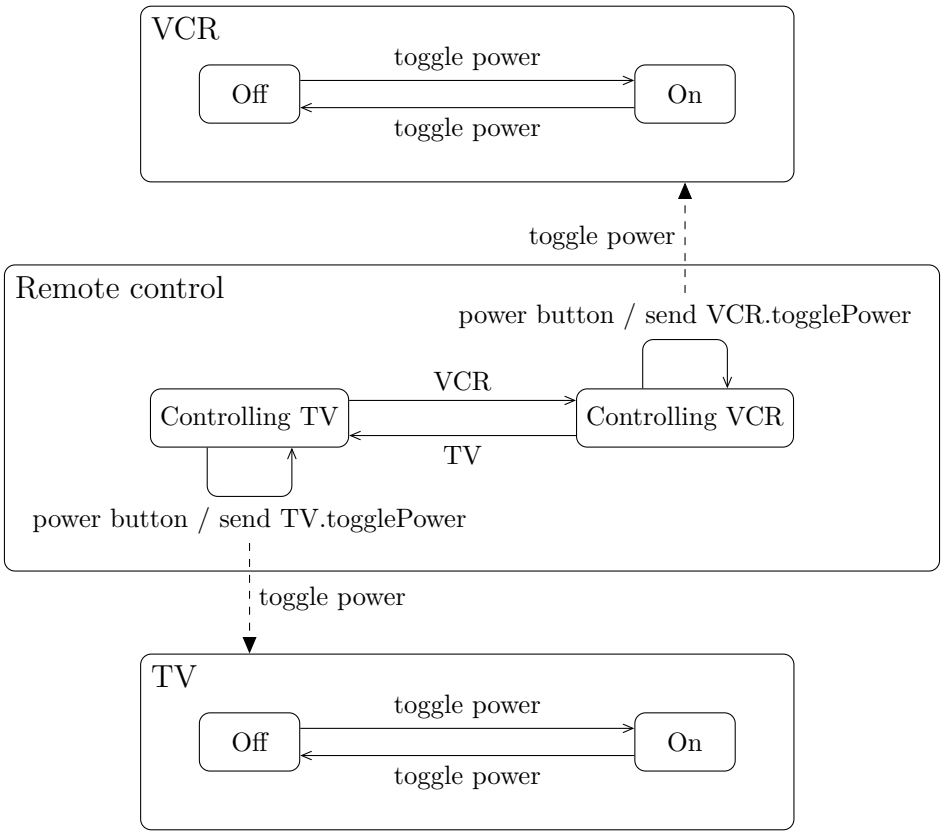


Nello stato "0 oggetti selezionati", se l'utente fa click con il mouse in una determinata locazione (evento left-mouse-down), e tale locazione si trova all'interno della finestra, avviene una transizione allo stato "1 oggetto selezionato", e si esegue un'azione:

1. viene ottenuto l'oggetto cliccato, qui rappresentato dalla variabile o;
2. si invia a tale oggetto l'evento (chiamata di metodo) highlight (in risposta al quale l'oggetto o effettuerà, presumibilmente, delle transizioni di stato e/o delle azioni che lo faranno apparire evidenziato all'interno della finestra).

10.2 Esempio 2

Il diagramma seguente modella un telecomando che è in grado di controllare sia una TV che un VCR, mediante due pulsanti che permettono di selezionare a quale dispositivo indirizzare i comandi:



- Quando l'utente preme il pulsante di accensione/spegnimento (power button), viene inviato un evento, toggle power, al VCR o alla TV, in base allo stato attuale del telecomando.
- Tale evento accende o spegne il dispositivo, a seconda dello stato di quest'ultimo al momento della ricezione.

- Le frecce tratteggiate, che indicano esplicitamente l'invio degli eventi, si possono omettere, perché il destinatario degli eventi inviati è già indicato nelle azioni di invio (send *VCR*.togglePower, send *TV*.togglePower).

11 State diagram strutturati

Gli state diagram **piatti**, cioè senza decomposizione, diventano poco espressivi e troppo "ingarbugliati" al crescere delle dimensioni del problema, cioè del numero di stati e transizioni.

La soluzione sono gli state diagram **strutturati**, che permettono la descrizione di sistemi complessi a diversi livelli di astrazione:

- Con la decomposizione OR:
  - l'attività corrispondente a uno stato può essere espansa in un diagramma a stati di più basso livello, dove ogni stato rappresenta una fase dell'attività;
  - si possono raccogliere stati che hanno le stesse transizioni, mettendo queste ultime a fattor comune.
- La decomposizione AND permette di rappresentare:
  - più processi che avvengono in parallelo;
  - il comportamento di oggetti composti, mostrando gli stati di ciascun componente.

11.1 Decomposizione OR

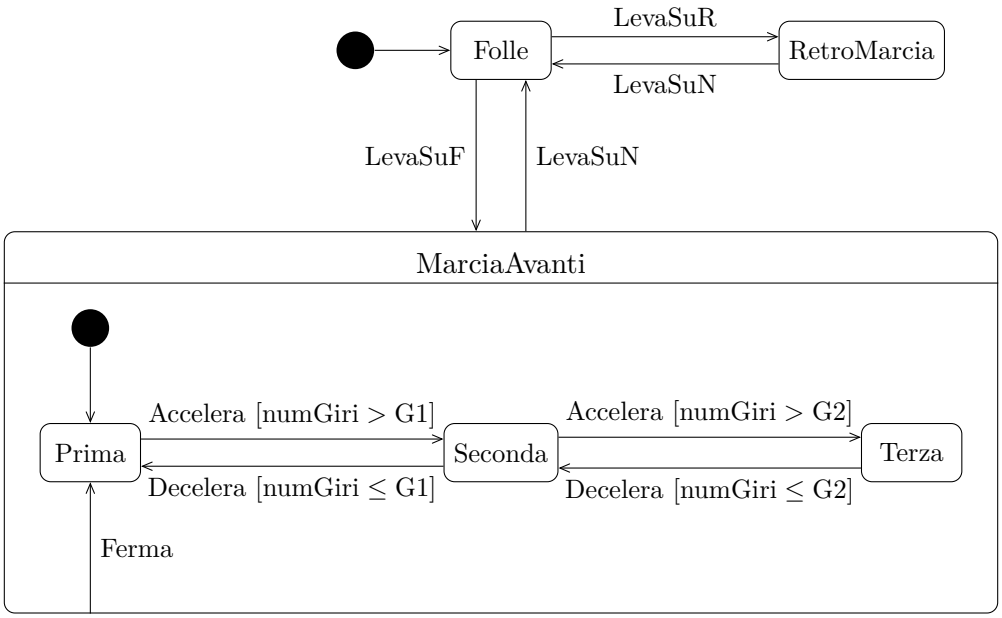
La **decomposizione OR** suddivide un *macro-stato* in più *sotto-stati*, che formano un automa interno al macro-stato. Essa è chiamata decomposizione OR perché, quando il sistema è nel macro-stato, si può trovare, in un certo momento, in *uno solo* dei sotto-stati.

I sottostati ereditano le transizioni dello stato composto che li contiene:

- una transizione che entra in uno stato composto "porta" (solitamente) allo stato iniziale del sotto-automa;
- una transizione che esce da uno stato composto esce effettivamente da tutti i sotto-stati.

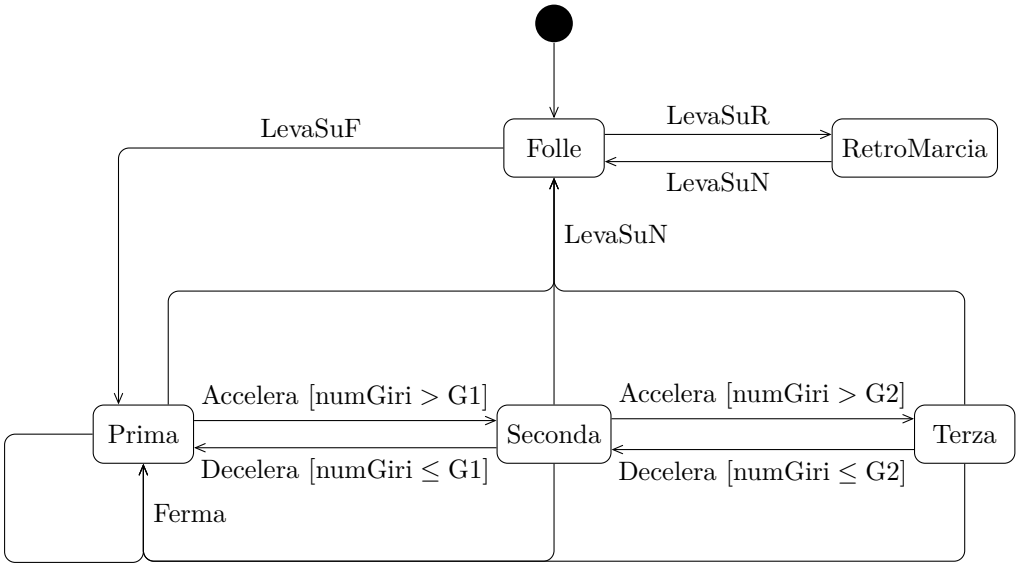
11.1.1 Esempio

Questo diagramma modella il comportamento di un cambio automatico:



- MarciaAvanti è mostrato come stato composto, ma, a più alto livello, lo si potrebbe vedere come uno stato unico.
- La transizione allo stato Folle, in risposta all'evento LevaSuN, è messa a fattor comune tra tutti i sotto-stati di MarciaAvanti: ciò significa che è possibile passare allo stato Folle da qualsiasi marcia (Prima, Seconda, o Terza).
- La transizione in risposta all'evento Ferma collega il macro-stato MarciaAvanti al suo sotto-stato Prima, e viene quindi ereditata da tutti i sotto-stati: da qualsiasi marcia, quando avviene l'evento Ferma si torna alla Prima.

Il diagramma piatto equivalente è:

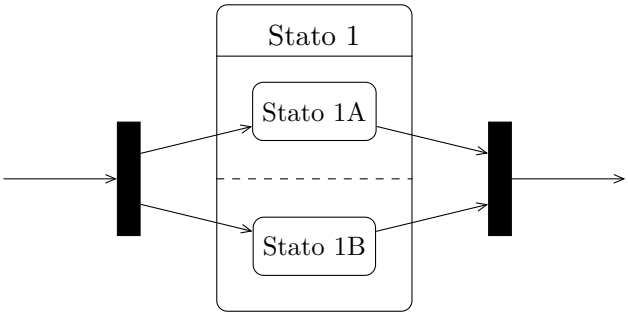


11.2 Decomposizione AND

Con la **decomposizione AND**, uno stato viene suddiviso in due o più **regioni**, contenenti sotto-automi che evolvono *in parallelo*. In un certo momento, ogni sotto-automa si trova in un determinato stato, quindi il numero di stati "attivi" è pari al numero di regioni in cui è suddiviso il macro-stato.

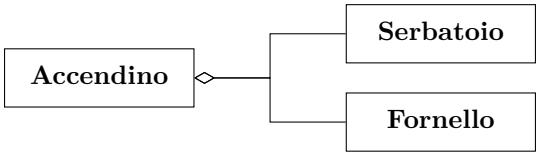
La decomposizione AND può essere usata per:

- Modellare un oggetto composto: lo state diagram di tale oggetto è l'aggregazione dei diagrammi di ciascun componente.
- Rappresentare flussi di esecuzione paralleli: un singolo flusso di esecuzione si "divide" (*fork*), si passa in parallelo da stati diversi, e infine i flussi si "riuniscono" (*join*), sincronizzandosi (l'esecuzione non prosegue finché non terminano tutti i flussi paralleli).

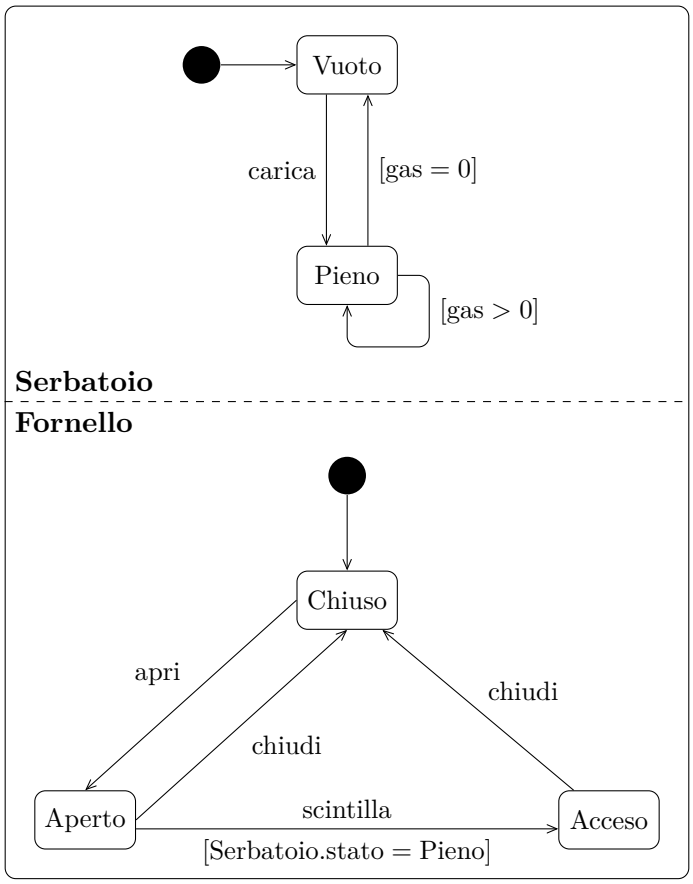


11.2.1 Esempio

Un Accendino è composto da un Serbatoio e da un Fornello:



Lo state diagram dell'Accendino corrisponde quindi a quelli di Serbatoio e Fornello, messi in AND:



In questo caso, c'è un vincolo tra i due componenti: quando il Fornello è aperto, e si ha la scintilla, il Fornello si accende solo se il Serbatoio è nello stato Pieno.

12 History

La **history** può essere associata a stati composti.

Quando l'esecuzione lascia un macro-stato dotato di history, viene salvato l'ultimo sotto-stato visitato. È quindi possibile definire transizioni che tornano direttamente a tale sotto-stato (invece di ripartire dal sotto-stato iniziale del macro-stato), collegando le frecce al simbolo della history.

Esistono due varianti di history:

- con la **shallow history**,

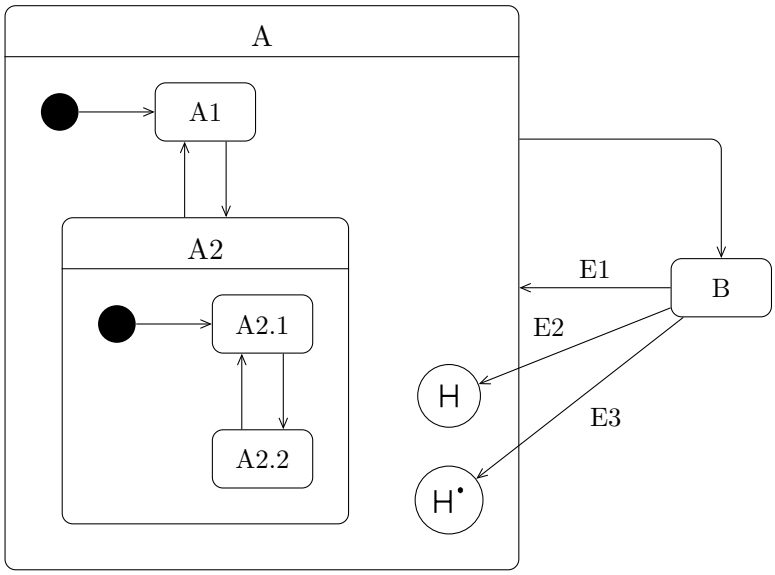
  si ritorna all'ultimo sotto-stato visitato, considerando solo un livello di decomposizione;

- la **deep history**, invece,

  considera tutti i livelli di decomposizione, permettendo quindi di tornare all'ultimo sotto-stato foglia (cioè non ulteriormente decomposto) che è stato visitato.

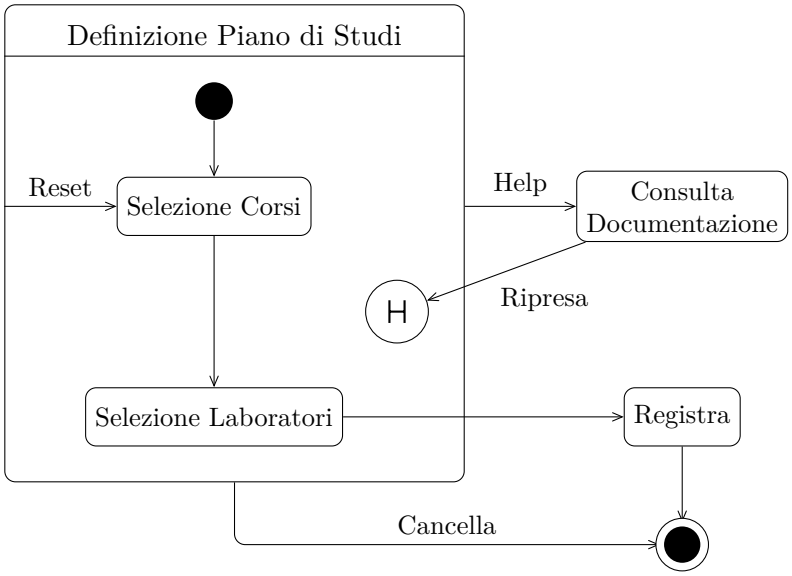
Ad esempio:



Si suppone che lo stato attuale sia B, e che lo stato precedente fosse A2.2. Allora, dallo stato B:

- In risposta all'evento E1, si passa al sotto-stato iniziale di A, quindi l'esecuzione prosegue da A1.
- La transizione in risposta all'evento E2 usa la *shallow history*: si torna all'ultimo sotto-stato visitato tra i sotto-stati al livello di A, ovvero A2. All'interno di quest'ultimo, si riparte dal suo sotto-stato iniziale, e allora l'esecuzione continua da A2.1.
- In risposta all'evento E3, si considera la *deep history*: l'esecuzione ritorna all'ultimo sotto-stato visitato all'interno di A, considerando tutti i livelli di decomposizione. Perciò, il nuovo stato è A2.2.

12.1 Esempio



- Si può passare alla registrazione solo dal sotto-stato Selezione Laboratori, mentre la cancellazione, la consultazione dell'help, e il Reset sono disponibili da entrambi i sotto-stati di Definizione Piano di Studi.
- Dopo aver consultato la documentazione, si riprende la definizione del piano di studi dal punto in cui era stata interrotta (per effetto della history).