Grafi di pianificazione: GRAPHPLAN

http://www.cs.cmu.edu/∼avrim/graphplan.html

Applicabili a problemi **proposizionali**

Grafo di pianificazione: grafo a livelli. Ogni livello ha due “strati”: uno strato di letterali (possibilmente veri in uno stato) e uno di azioni (possibilmente eseguibili nello stato).

Livello 0, stato: stato iniziale

Livello 0, azioni: eseguibili nello stato iniziale

Proposizionalizzazione di un problema: esempio

```
(:fluents (ha ?t) (mangiato ?t))
(:objects torta spaghetti)
(:init (ha torta) (ha spaghetti))
(:goal (mangiato torta) (ha torta) (mangiato spaghetti) (ha spaghetti))
(:action mangia
         :parameters (?x)
         :precondition (ha ?x)
         :effect (not(ha ?x)) (mangiato ?x))
(:action cucina
         :parameters (?x)
         :precondition (not (ha ?x))
         :effect (ha ?x))
```

Problema proposizionale:

```
(:fluents (ha_torta)(mangiato_torta)(ha_spaghetti)(mangiato_spaghetti))
(:init (ha_torta) (ha_spaghetti))
(:goal (mangiato_torta) (ha_torta) (mangiato_spaghetti) (ha_spaghetti))

(:action mangia_torta
         :precondition (ha_torta)
         :effect (not(ha_torta)) (mangiato_torta))
(:action mangia_spaghetti
         :precondition (ha_spaghetti)
         :effect (not(ha_spaghetti)) (mangiato_spaghetti))
(:action cucina_torta
         :precondition (not (ha_torta))
         :effect (ha_torta))
(:action cucina_spaghetti
         :precondition (not (ha_spaghetti))
         :effect (ha_spaghetti))
```

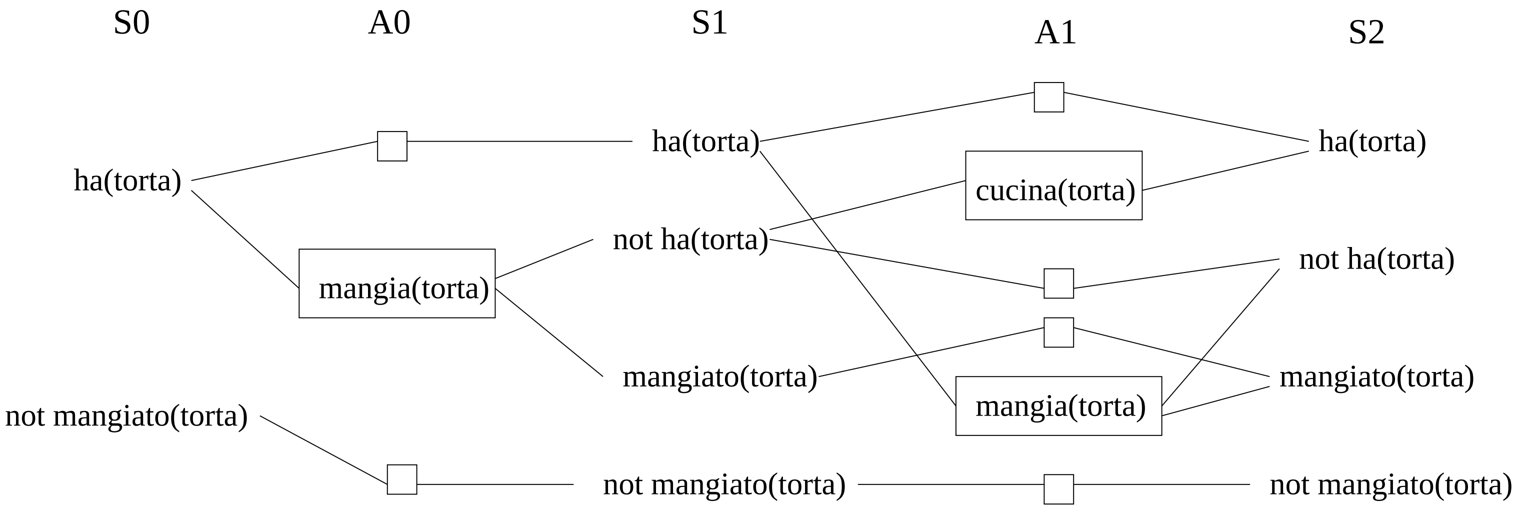
Grafo di pianificazione: esempio

```
(:fluents (ha ?t) (mangiato ?t))
(:objects torta)

(:action mangia
         :parameters (?x)
         :precondition (ha ?x)
         :effect (not(ha ?x))
                 (mangiato ?x))
```

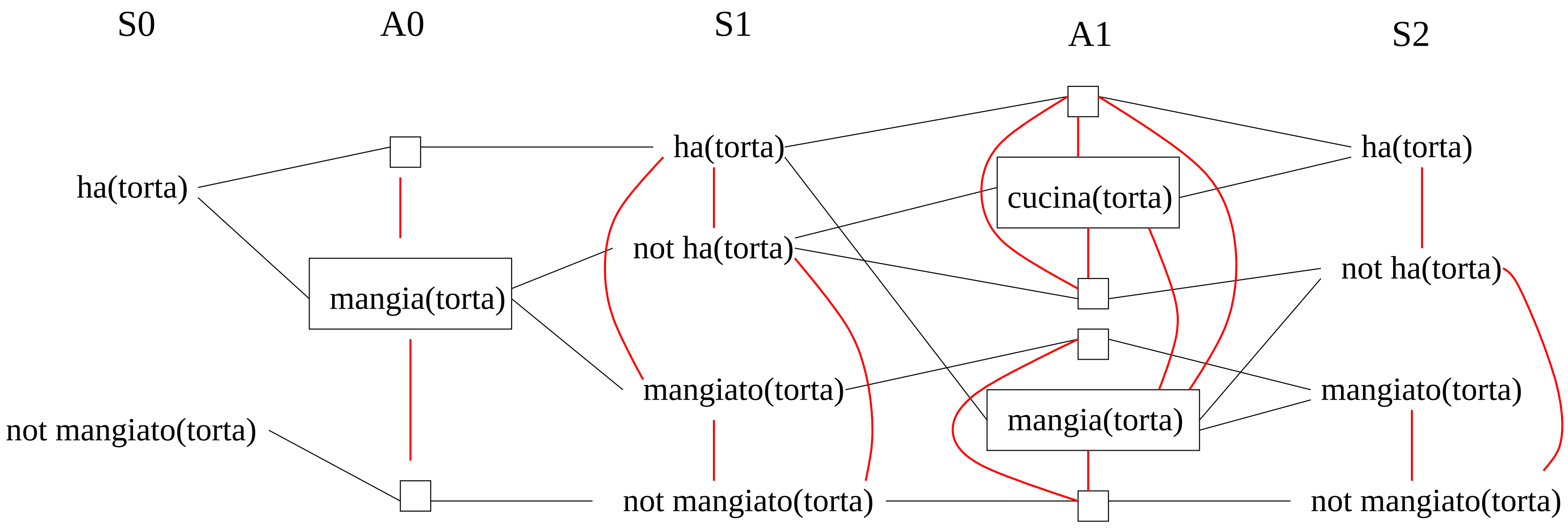
```
(:init (ha torta))
(:goal (mangiato torta)
       (ha torta))

(:action cucina
         :parameters (?x)
         :precondition (not (ha ?x))
         :effect (ha ?x))
```



□: azioni di **persistenza**

Mutua esclusione: MUTEX



Letterali in collegamento mutex: non possono apparire insieme, indipendentemente dalla scelta delle azioni

Azioni in collegamento mutex: non possono essere eseguite insieme.

Ad esempio:

In A0: mangia(torta) e l'azione di persistenza di ha(torta) hanno effetti contraddittori: sono mutex

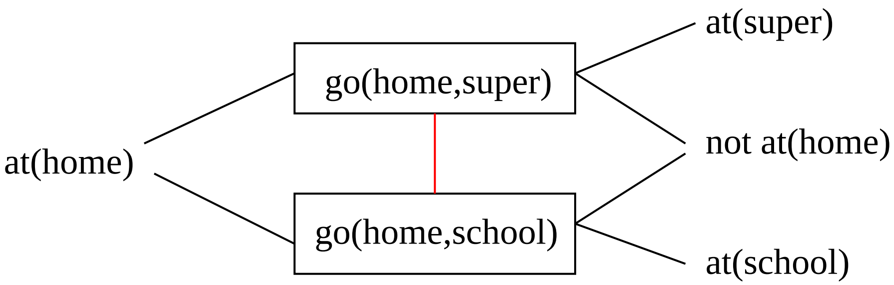
In S1: ha(torta) e mangiato(torta) sono effetti di due azioni mutex in A0: sono mutex

Calcolo dei collegamenti mutex

Tra azioni:

effetti inconsistenti: un effetto di un'azione è la negazione di un effetto dell'altra (nel livello degli stati successivo)

interferenze: un effetto di un'azione è la negazione di una precondizione dell'altra (le due azioni non possono essere linearizzate in un modo qualsiasi)



necessità in competizione: una precondizione di un'azione è in collegamento mutex con una precondizione dell'altra (nel livello degli stati precedente)

Tra letterali:

uno è la negazione dell'altro, oppure

ogni possibile coppia di azioni (del livello di azioni precedente) che hanno come effetto i due letterali sono in collegamento mutex

Esercizio: disegnare il grafo di pianificazione completo per un problema con fluenti $at(home)$, $at(super)$, $at(school)$, in cui le azioni possibili sono istanze di $go(x, y)$ con $x \neq y$ e $x, y \in \{home, super, school\}$. Nello stato iniziale l'unico atomo vero è $at(home)$.

Caratteristiche dei grafi di pianificazione

- Alcuni insiemi crescono/decrescono monotonicamente:
  - i letterali che appaiono in un livello S_n compaiono anche in S_{n+1}
  - le azioni che compaiono al livello A_n compaiono anche in A_{n+1}
  - se due azioni o letterali sono mutex al livello n, lo sono anche al livello $n-1$ (se vi occorrono).
- La costruzione del grafo termina quando il grafo "si livella": quando un livello è identico al precedente, anche tutti i livelli successivi saranno identici.
- Anche in problemi in cui il numero di azioni proposizionali è elevato, la costruzione del grafo di pianificazione è relativamente rapida (richiede tempo polinomiale) e si è dimostrata uno strumento molto efficace.
- Il grafo di pianificazione di un problema si può utilizzare per assegnare una stima euristica al costo del raggiungimento di un letterale, in base al livello in cui occorre per la prima volta nel grafo: **costo di livello** di un letterale (∞ se non vi occorre mai). La valutazione euristica può essere utilizzata dagli algoritmi di ricerca del piano.

L'algoritmo GRAPHPLAN

- Costruire il grafo iniziale, costituito soltanto dal livello S_0.
- Ciclo, per $i = 0, 1, ...$ fino a trovare una soluzione o a riconoscere che nessuna soluzione è possibile:
  - Se gli obiettivi del problema sono tutti non MUTEX al livello S_i, allora cercare una soluzione nel grafo costruito, altrimenti espandere il grafo di un livello.
  - Se si trova una soluzione, riportarla e terminare, altrimenti espandere il grafo di un livello.

Si può dimostrare che l'algoritmo termina sempre, anche se il problema non ha soluzione.

Ricerca della soluzione all'indietro, nel grafo

- Inizializzazione:
  - L'insieme A, inizialmente vuoto, conserva l'insieme delle azioni scelte per essere inserite nel piano, ciascuna con il suo livello.
  - L'insieme G conserva i letterali ancora da soddisfare, ciascuno associato al suo livello. Inizialmente G contiene i letterali dell'obiettivo del problema
- Ciclo:
  - Se G è vuoto, l'algoritmo termina con successo riportando A.
  - Altrimenti:
    * **scegliere** un letterale $\ell \in G$ e cancellarlo da G
    * se ℓ è di livello 0, proseguire nel ciclo.
      Altrimenti, se ℓ è nel livello $n > 0$, **scegliere** un'azione a del livello $n - 1$ che ha ℓ come effetto e che non sia in relazione mutex con nessuna altra azione del livello $n - 1$ già inserita in A.
      Se non esiste una tale azione, fallimento. Altrimenti l'azione a viene inserita in A e le precondizioni di a del livello $n - 1$ vengono inserite in G.

La soluzione è una sequenza di insiemi di azioni (un piano parallelo o parziale): $A_0, A_1, ..., A_n$ dove A_i contiene tutte le azioni del livello i che sono state scelte (in A).

Valutazione euristica per la scelta dei letterali e delle azioni

- La **scelta** del letterale da soddisfare non è un punto di scelta, ma si può basare sul costo di livello dei letterali: scegliamo prima quello a costo più alto (se la strada intrapresa porta al fallimento, meglio accorgersene il più presto possibile).

- La scelta dell'insieme di azioni ad ogni livello è un **punto di scelta**: vi possono essere diverse alternative, di cui solo alcune portano a una soluzione. Se una scelta fallisce, si devono comunque provare le altre.

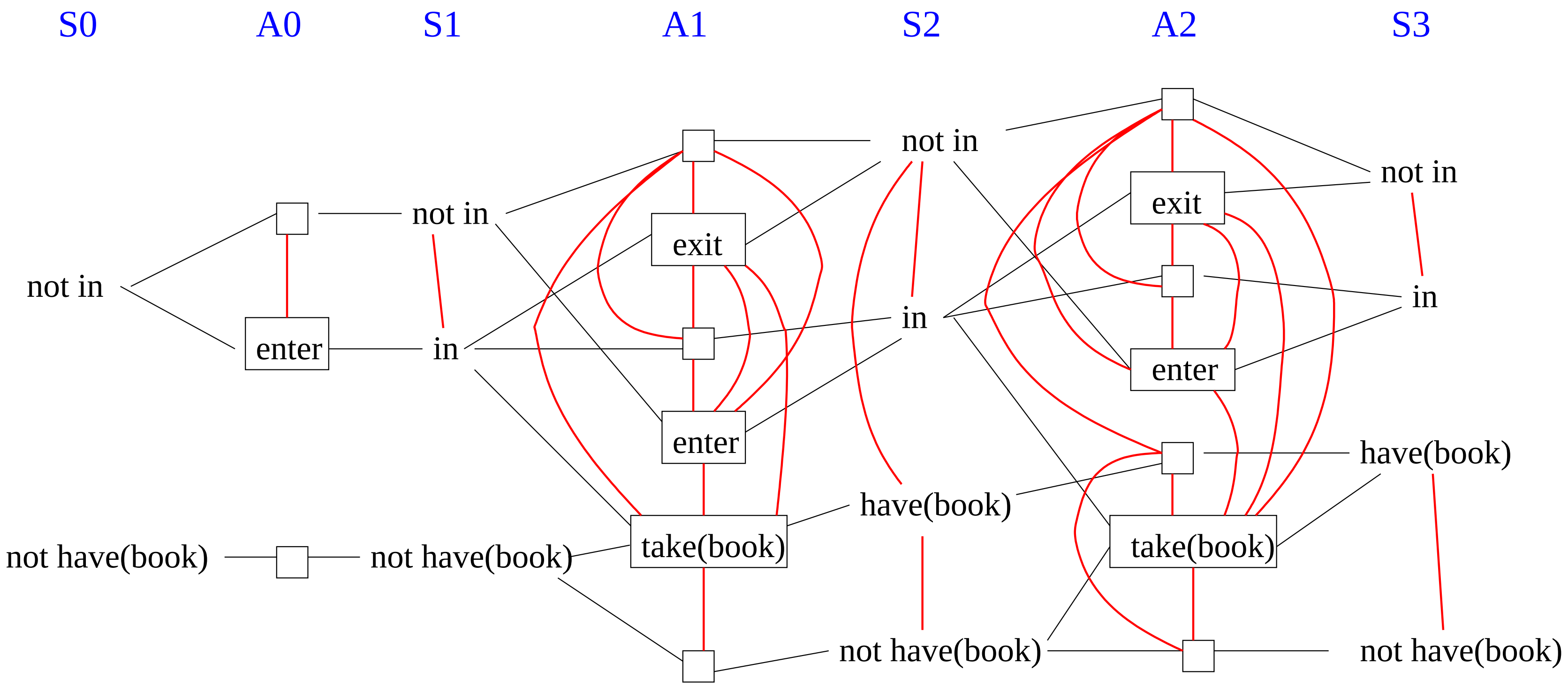
  Per la scelta delle azioni si può ad ogni modo usare una strategia che utilizza la valutazione euristica: le alternative sono ordinate secondo i costi di livello delle relative precondizioni, in modo da scegliere per prima l'azione che minimizza (la somma o il massimo di) tali costi di livello (le più facili da ottenere).

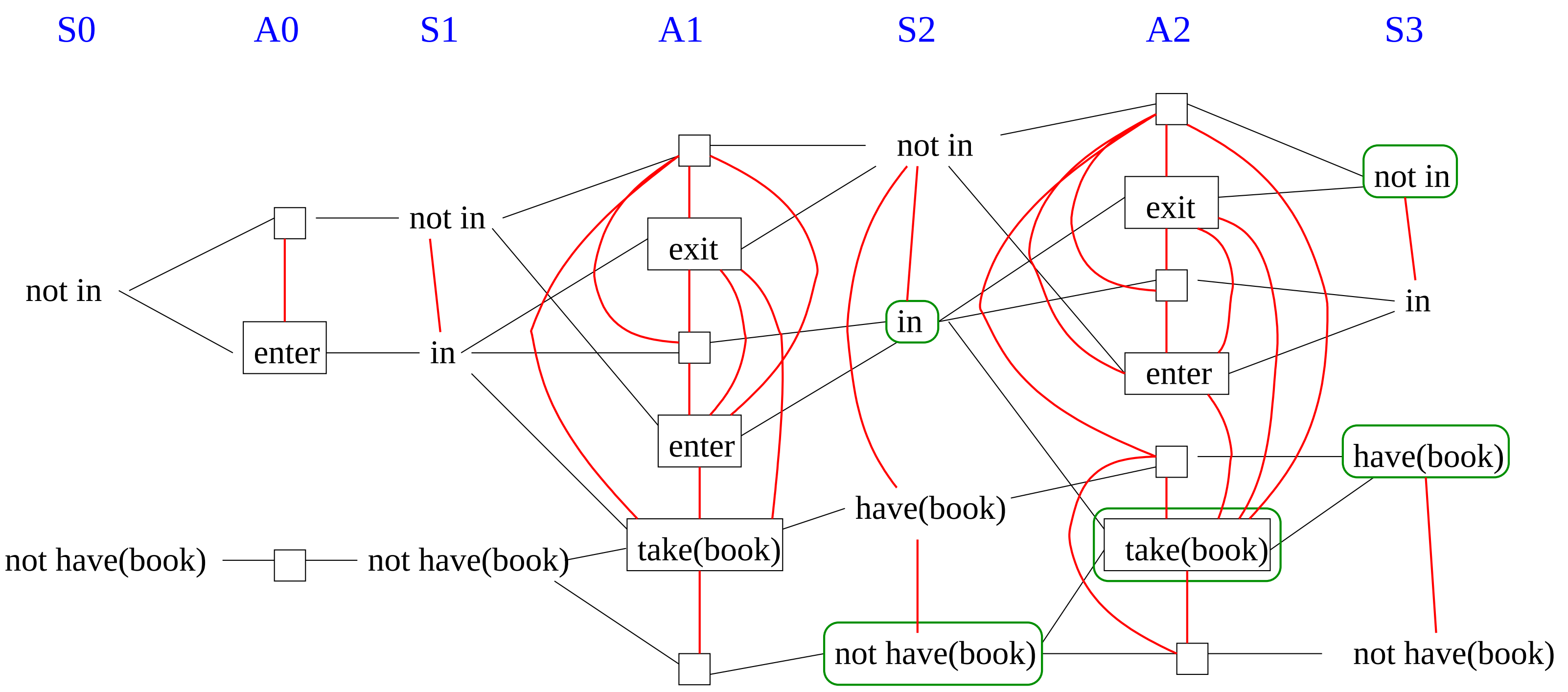
Esempio (linguaggio che ammette precondizioni negative)

```
(:fluents (have ?x) (in))
(:objects book)
  /* init: tutto falso */
(:goal (have book) (not (in)))

(:action enter
         :precondition (not (in))
         :effect (in))
```

```
(:action exit
         :precondition (in)
         :effect (not (in)))
(:action take
         :parameters (?x)
         :precondition (in) (not (have ?x))
         :effect (have ?x))
```





S2 contiene tutti i letterali del goal, ma sono MUTEX, quindi la costruzione del grafo prosegue fino a S3 prima di iniziare la ricerca

1a) $G = \{S3 : have(book), S3 : \neg in\}$.

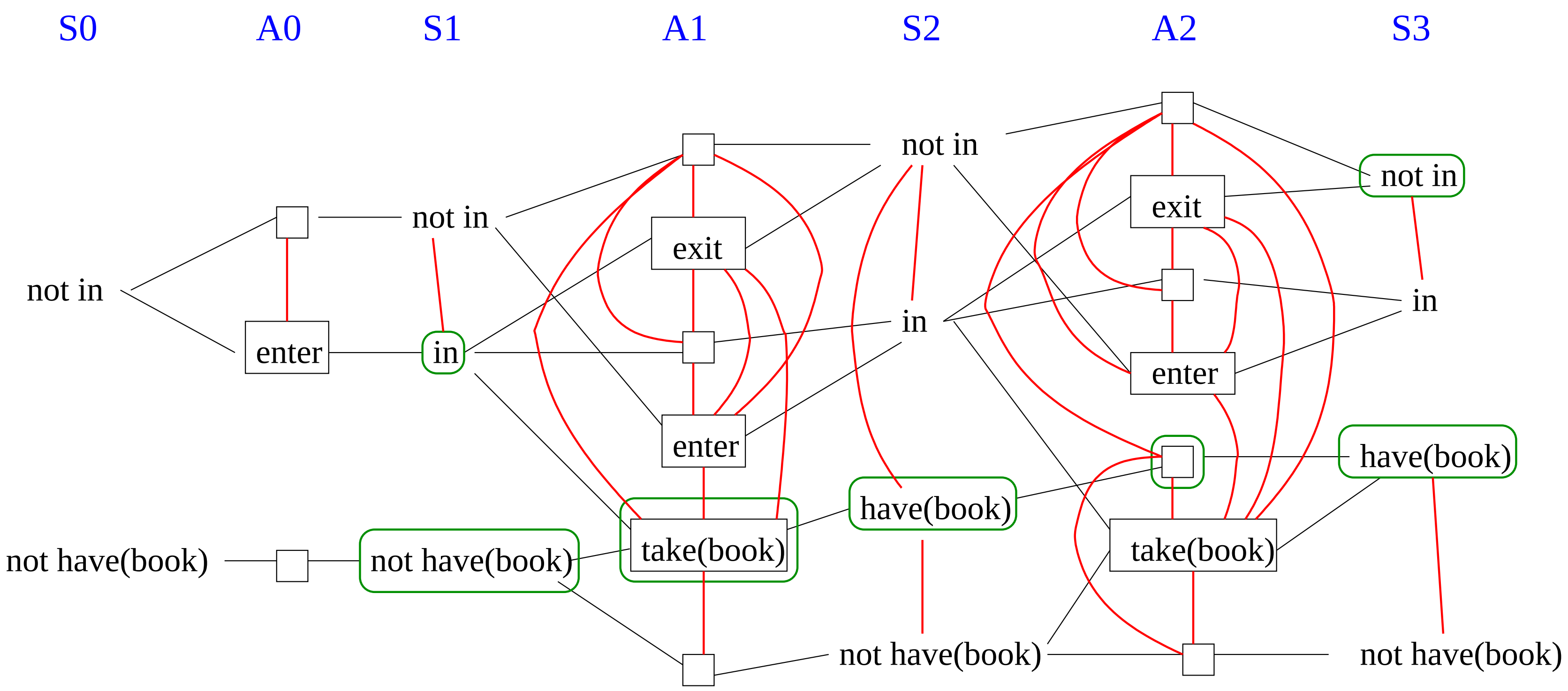
Scegliamo $S3 : have(book)$ (costo di livello maggiore).

Alternative: $\begin{cases} \bullet \text{ azione di persistenza (costo delle precondizioni: 2)} \\ \bullet \; take(book) \text{ (costo delle precondizioni: } 1 + 0) \end{cases}$

Si inserisce in A l'azione $A2 : take(book)$, ma si ricorda questa scelta: se la ricerca fallisce, si ritratta la scelta e si prova l'alternativa.

$$A = \{A2 : take(book)\}, \qquad G = \{S3 : \neg in,\ S2 : in,\ S2 : \neg have(book)\}$$

Effettivamente la ricerca fallirà per questa strada ...



Backtracking:

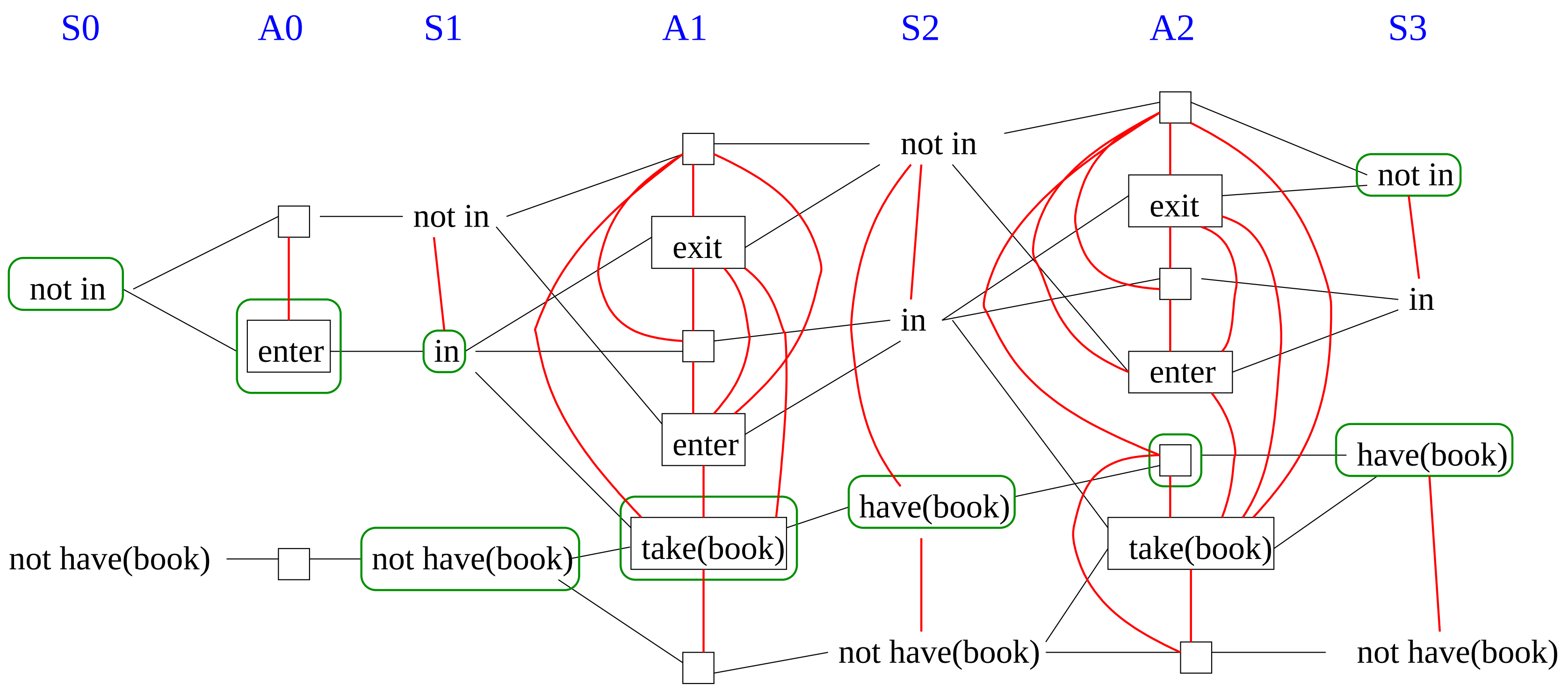
1b) Si sceglie l'azione di persistenza, che viene inserita in A, e viene modificato G:

$$A = \{A2 : AP(have(book))\}, \qquad G = \{S3 : \neg in,\ S2 : have(book)\}$$

Se la ricerca fallisse, non vi sarebbero alternative da provare.

2) Si sceglie il lettarale $S2 : have(book)$. Non vi sono alternative: si inserisce in A l'azione $A1 : take(book)$:

$$A = \{A1 : take(book),\ A2 : AP(have(book))\},$$
$$G = \{S3 : \neg in,\ S1 : in,\ S1 : \neg have(book)\}$$



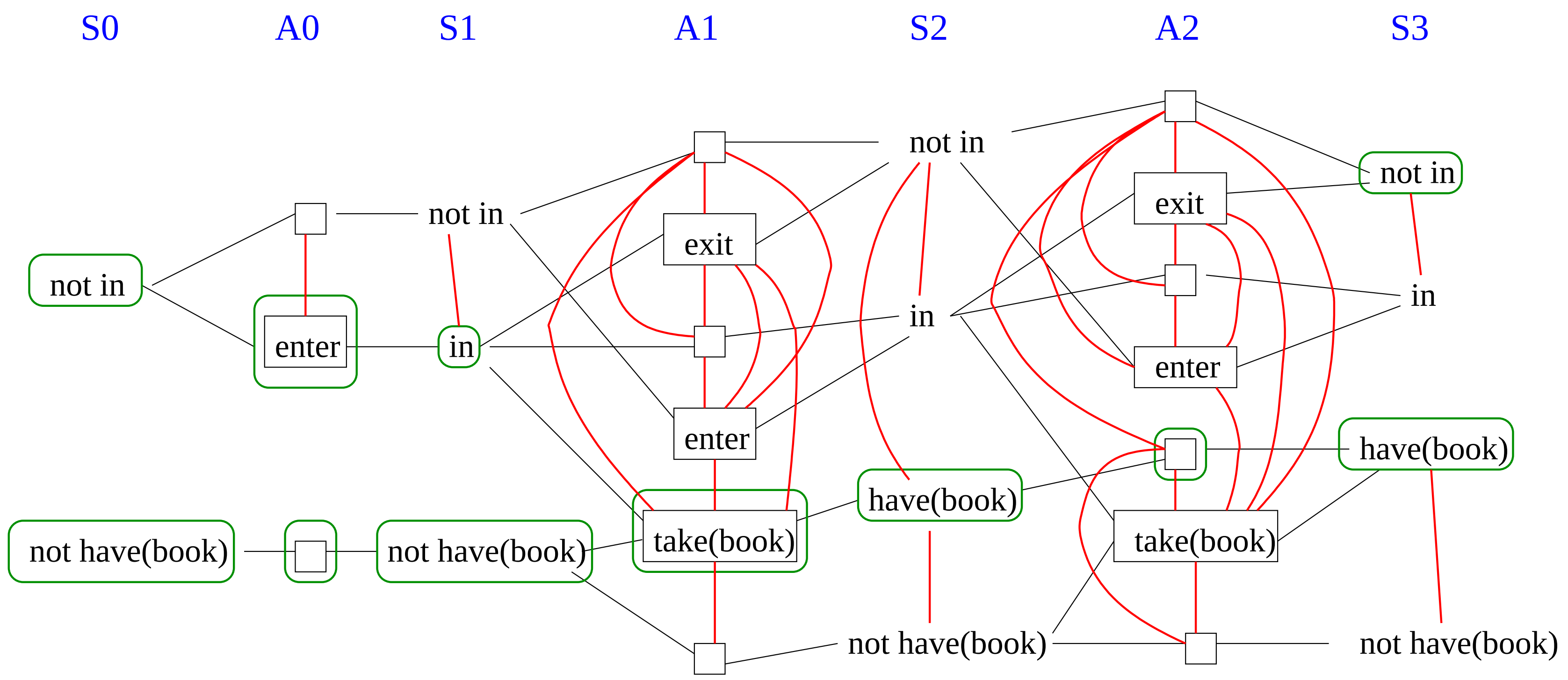
3) Si sceglie il letterale $S1 : in$ (costo maggiore).

L'unica azione che ha come effetto $S1 : in$ è $A0 : enter$:

$$A = \{A0 : enter,\ A1 : take(book),\ A2 : AP(have(book))\},$$
$$G = \{S3 : \neg in,\ S1 : \neg have(book),\ S0 : \neg in\}$$

4) $S0 : \neg in$ viene eliminato da G:

$$A = \{A0 : enter,\ A1 : take(book),\ A2 : AP(have(book))\},$$
$$G = \{S3 : \neg in,\ S1 : \neg have(book)\}$$

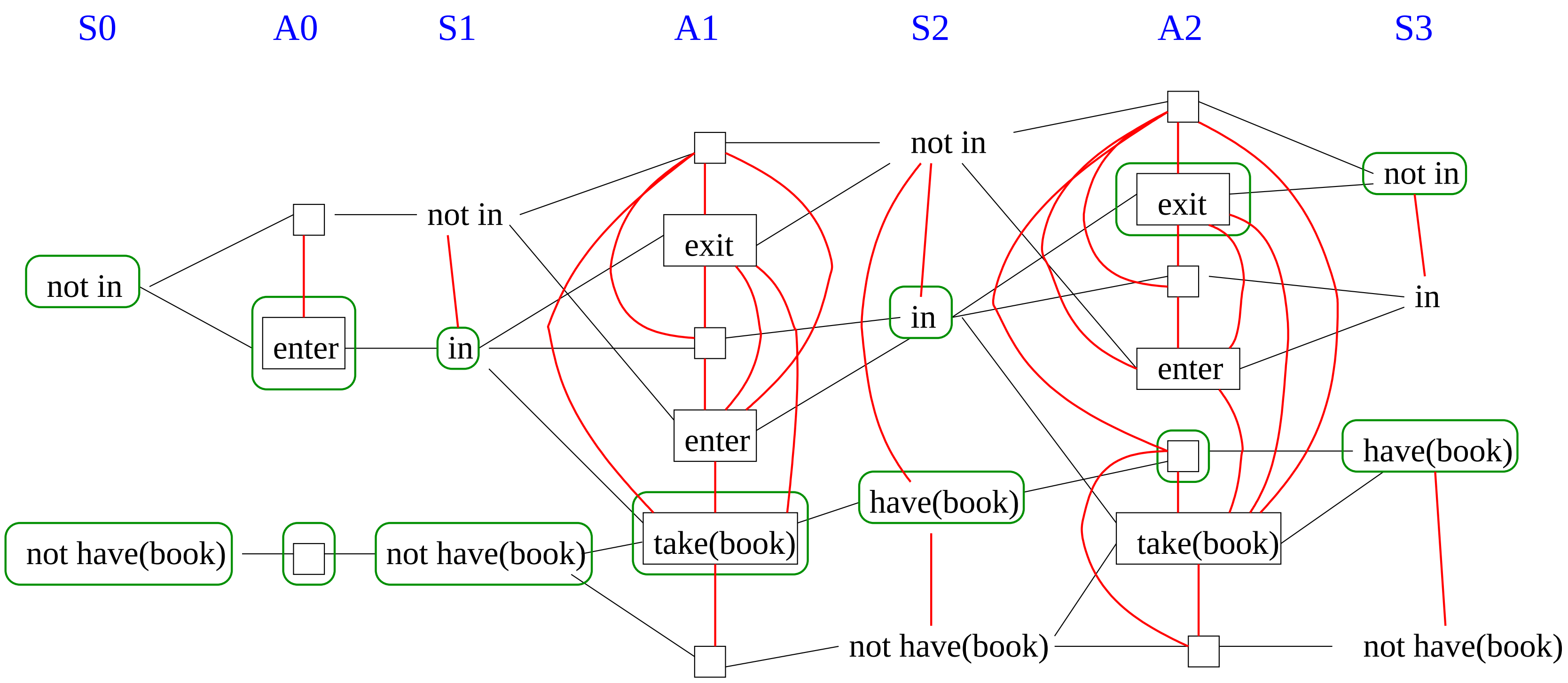


5) Si sceglie il letterale $S1 : \neg have(book)$, che è possibile soddisfare soltanto mediante l'azione di persistenza del livello $A0$:

$$A = \{A0 : AP(\neg have(book)),\ A0 : enter,\ A1 : take(book),\ A2 : AP(have(book))\},$$
$$G = \{S3 : \neg in,\ S0 : \neg have(book)\}$$

6) Il letterale $S0 : \neg have(book)$ viene eliminato da G:

$$A = \{A0 : AP(\neg have(book)),\ A0 : enter,\ A1 : take(book),\ A2 : AP(have(book))\},$$
$$G = \{S3 : \neg in\}$$

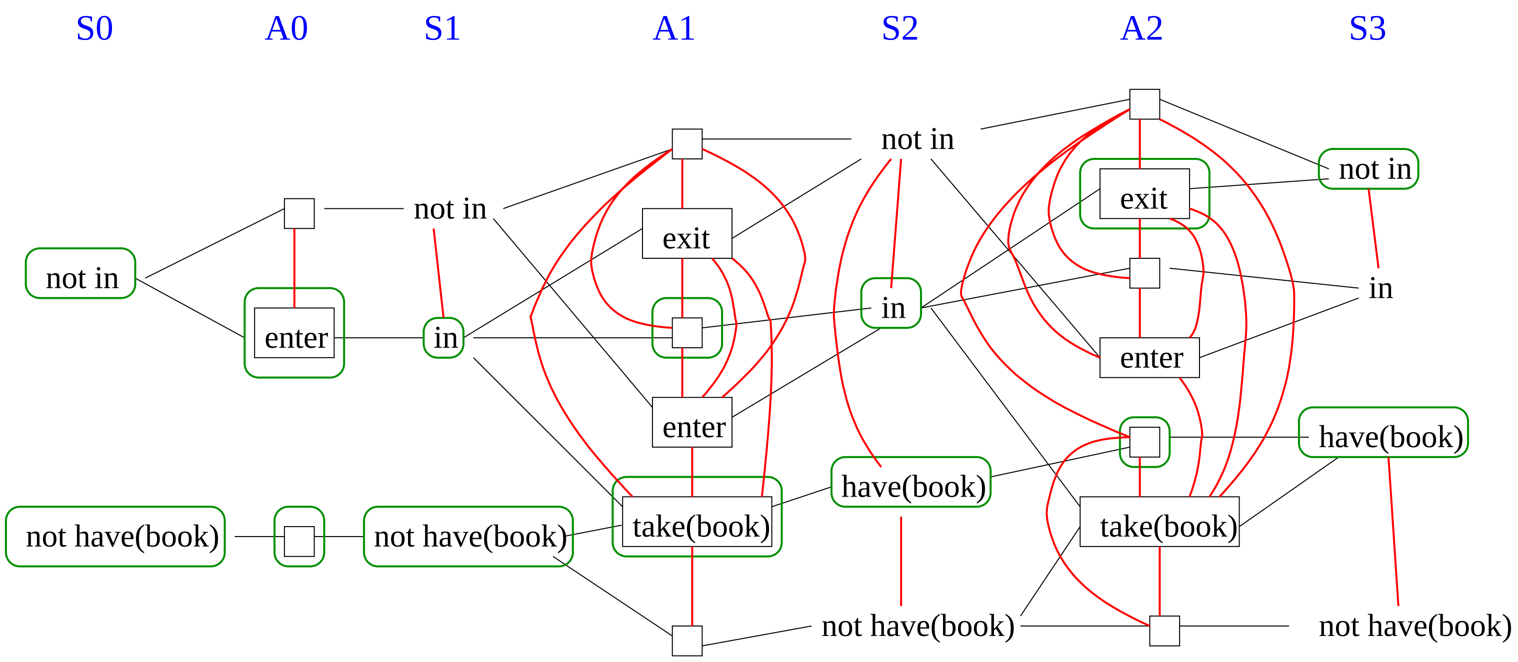


7) Si sceglie il letterale $S3 : \neg in$.

L'azione di persistenza $AP(\neg in)$ del livello $A2$ è in collegamento MUTEX con $A2 : AP(have(book))$, già inserita in A.

Quindi l'unica scelta possibile per soddisfare $S3 : \neg in$ è $A2 : exit$:

$$
\begin{aligned}
A = \{ \;& A0 : AP(\neg have(book)),\; A0 : enter,\; A1 : take(book), \\
& A2 : AP(have(book)),\; A2 : exit\}, \\
G = \{ \;& S2 : in\}
\end{aligned}
$$



Backtracking:

8) Per soddisfare $S2 : in$ si può scegliere solo l'azione di persistenza in $A1$ ($A1 : enter$ è mutex con $A1 : take(book)$):

$$\begin{aligned} A = \{ \quad & A0 : AP(\neg have(book)),\ A0 : enter,\ A1 : take(book),\ A1 : AP(in) \\ & A2 : AP(have(book)),\ A2 : exit\}, \\ G = \{ \quad & S1 : in\} \end{aligned}$$

9) L'unica azione che provoca $S1 : in$ è $A0 : enter$, che è già presente in A.

Ora G è vuoto e l'algoritmo termina.

Soluzione (eliminando le azioni di persistenza):

$$\langle \{enter\},\ \{take(book)\},\ \{exit\} \rangle$$

Esercizi

Considerare una semplificazione di alcuni dei problemi già presentati nei lucidi sul linguaggio STRIPS e rappresentare il relativo grafo di pianficazione, evidenziando tutti i collegamenti mutex.

In particolare:

1. Il problema della scimmia e della banana, in cui le posizioni della stanza sono soltanto A, B e C, e non ci sono né la palla né il libro.
2. Il problema del robot che deve prendere il libro e metterlo nella borsa, senza le azioni di entrare e uscire dalla stanza (l'ambiente è costituito soltanto dalla stanza).
3. Il problema del robot che deve comprare oggetti, in cui gli unici posti dell'ambiente sono *casa* e *super*, e l'obiettivo è quello di avere latte e banana.