Planning as satisfiability

Approcci logici alla pianificazione:

- Pianificazione come dimostrazione di teoremi (nel calcolo delle situazioni)
- Pianificazione come soddisfacibilità (in logica proposizionale):
  riduzione della pianificazione alla **ricerca di modelli** di insiemi di formule della forma

$$\{stato\_iniziale,\ descrizione\_delle\_azioni,\ obiettivo\}$$

SATPLAN: `http://www.cs.washington.edu/homes/kautz/satplan/`

Rappresentazione del tempo in logica proposizionale

Ad ogni fluente ground P corrisponde un insieme di lettere proposizionali: $P^0,\ P^1,\ P^2,\ P^3, \ldots$

$$P^i:\ \ P \text{ è vero al tempo } i$$

$$on(A,B)\ \Rightarrow \begin{cases} on\_A\_B^0 & : \ on(A,B) \text{ è vero al tempo } 0 \\ on\_A\_B^1 & : \ on(A,B) \text{ è vero al tempo } 1 \\ on\_A\_B^2 & : \ on(A,B) \text{ è vero al tempo } 2 \\ \ldots \end{cases}$$

Rappresentazione delle azioni

Ad ogni azione A corrisponde un insieme di lettere proposizionali: $A^0,\ A^1,\ A^2,\ A^3, \ldots$

$$A^i : \ A \text{ è eseguita al tempo } i$$

$$putOn(A,B) \ \Rightarrow \begin{cases} putOn\_A\_B^0 & : \ putOn(A,B) \text{ è eseguita al tempo } 0 \\ putOn\_A\_B^1 & : \ putOn(A,B) \text{ è eseguita al tempo } 1 \\ putOn\_A\_B^2 & : \ putOn(A,B) \text{ è eseguita al tempo } 2 \\ \ldots \end{cases}$$

Quando viene eseguita un'azione A al tempo i, le precondizioni di A devono essere vere al tempo i, gli effetti di A sono veri al tempo $i+1$.

Piani e modelli: esempio

stato iniziale: $\begin{array}{cc}\boxed{\mathrm{A}} & \\ \boxed{\mathrm{B}} & \boxed{\mathrm{C}}\end{array}$ obiettivo: $\begin{array}{c}\boxed{\mathrm{A}}\\ \boxed{\mathrm{B}}\\ \boxed{\mathrm{C}}\end{array}$

L'interpretazione $\mathcal{M}$ in cui sono vere esattamente le seguenti lettere proposizionali (e false tutte le altre):

$$\begin{array}{l}
on_A_B^0,\ onTable_B^0,\ onTable_C^0,\ putOnTable_A^0,\\
onTable_A^1,\ onTable_B^1,\ onTable_C^1,\ putOn_B_C^1,\\
onTable_A^2,\ on_B_C^2,\ onTable_C^2,\ putOn_A_B^2,\\
on_A_B^3,\ on_B_C^3,\ onTable_C^3,
\end{array}$$

rappresenta il piano

$$< putOnTable(A),\ putOn(B, C),\ putOn(A, B) >$$

Rappresentazione di un problema di pianificazione

Dato un problema di pianificazione Π, si deve costruire (in modo automatico, dalla descrizione STRIPS del problema) un insieme di formule S_Π tale che:

$$\begin{array}{c}\text{un'interpretazione } \mathcal{M} \text{ è un modello di } S_\Pi \\ \text{sse} \\ \mathcal{M} \text{ rappresenta un piano che risolve } \Pi\end{array}$$

Cioè: la sequenza di (insiemi di) azioni A^i tali che $\mathcal{M} \models A^i$

$$< \{A_j^0 \mid \mathcal{M} \models A_j^0\}, ..., \{A_j^k \mid \mathcal{M} \models A_j^k\} >$$

è una soluzione del problema Π

Quindi occorre definire un algoritmo per **codificare** in un insieme di formule proposizionali un problema di pianificazione descritto in STRIPS

Esempio

```
(:fluents (on ?x ?y) (onTable ?x) (clear ?x))
(:action putOnTable :parameters (?x ?from)
        :precondition (and (clear ?x) (on ?x ?from))
        :effect (and (onTable ?x) (clear ?from) (not(on ?x ?from))))
(:action stack :parameters (?x ?onto)
        :precondition (and (onTable ?x) (clear ?x) (clear ?onto))
        :effect (and (on ?x ?onto) (not(onTable ?x)) (not (clear ?onto))))
(:action move :parameters (?x ?from ?onto)
        :precondition (and (clear ?x) (clear ?onto) (on ?x ?from))
        :effect (and (on ?x ?onto) (clear ?from)
                     (not(on ?x ?from)) (not (clear ?onto))))
(:objects A B C)
(:init  (on A B) (onTable B) (onTable C) (clear A) (clear C) )
(:goal  (on A B) (on B C) (onTable C) )
```

N.B. Le restrizioni di STRIPS rendono necessaria l'aggiunta di parametri alle azioni, e la duplicazione di alcune azioni:

$$\begin{array}{ll} putOnTable(x) & \Longrightarrow\ putOnTable(x, y) \\ putOn(x, y) & \Longrightarrow\ stack(x, y),\ move(x, z, y) \end{array}$$

Inoltre, è necessario utilizzare anche il fluente $clear(x)$: STRIPS non consente formule universali $\forall y \neg on(y, x)$ (nessun blocco è sopra x)

Codifica dello stato iniziale

$$on\_A\_B^0 \wedge onTable\_B^0 \wedge onTable\_C^0 \wedge clear\_A^0 \wedge clear\_C^0 \wedge$$
$$\neg on\_A\_C^0 \wedge \neg on\_B\_A^0 \wedge \neg on\_B\_C^0 \wedge$$
$$\neg on\_C\_A^0 \wedge \neg on\_C\_B^0$$
$$\wedge \neg onTable\_A^0 \wedge \neg clear\_B^0$$

(assumiamo che il linguaggio non contenga atomi che rappresentano fluenti impossibili, come $on\_A\_A^i$)

La formula che rappresenta lo stato iniziale è la congiunzione di tutti i letterali:

$$\begin{array}{rl} P_i^0 & \text{tale che } P_i \text{ è vero nello stato iniziale, e} \\ \neg P_i^0 & \text{tale che } P_i \text{ è falso nello stato iniziale} \end{array}$$

Non si può assumere l'ipotesi del mondo chiuso: l'insieme delle formule di cui si cerca un modello deve contenere una **descrizione completa** dello stato iniziale, niente può essere "sottinteso".
Nel nostro esempio: se $\neg on\_B\_C^0$ non fosse parte della descrizione dello stato iniziale, la ricerca potrebbe fornire un modello in cui anche $on\_B\_C^0$ è vero (da nessuna parte è detto che B non possa stare contemporaneamente sul tavolo e su C), dunque in cui il goal è già soddisfatto nello stato iniziale.

Dalla descrizione STRIPS del problema: se $Init$ è la formula (congiunzione di atomi) dichiarata nel campo `:init`, allora la formula che rappresenta lo stato iniziale è la congiunzione di tutti i letterali in:

$$\{P_i^0 \mid P_i \text{ occorre in } Init\} \cup \{\neg P_j^0 \mid P_j \text{ non occorre in } Init\}$$

Rappresentazione dell'obiettivo: bounded time planning problems

Non è possibile sapere a priori quanti passi sono necessari per raggiungere l'obiettivo:

la pianificazione come soddisfacibilità in logica proposizionale può risolvere solo **problemi di pianificazione con tempo "limitato"**, in cui cioè l'orizzionte temporale è fissato a priori.

Per risolvere un problema generale, si deve procedere per **approfondimento iterativo** del limite temporale:

- ricerca di un piano di lunghezza 0; se si trova, si riporta il piano, altrimenti:
- ricerca di un piano di lunghezza 1; se si trova, si riporta il piano, altrimenti:
- ricerca di un piano di lunghezza 2; se si trova, si riporta il piano, altrimenti:
- ...

Se si vuole garantire la terminazione, si deve comunque imporre un limite temporale massimo.

La ricerca di un piano di lunghezza k costruisce la rappresentazione del problema utilizzando soltanto fluenti e azioni relativi al tempo compreso tra 0 e k.

Codifica dell'obiettivo per la ricerca limitata al tempo k

Se G è la formula dichiarata nel campo `:goal` della rappresentazione STRIPS, allora la formula che rappresenta l'obiettivo è la congiunzione dei letterali in

$$\{L^k \mid L \text{ occorre in } G\}$$

Esempio: rappresentazione dell'obiettivo per i primi 4 cicli di ricerca:

- $on\_A\_B^0 \wedge on\_B\_C^0 \wedge onTable\_C^0$ (la ricerca fallisce)
- $on\_A\_B^1 \wedge on\_B\_C^1 \wedge onTable\_C^1$ (la ricerca fallisce)
- $on\_A\_B^2 \wedge on\_B\_C^2 \wedge onTable\_C^2$ (la ricerca fallisce)
- $on\_A\_B^3 \wedge on\_B\_C^3 \wedge onTable\_C^3$ (la ricerca ha successo)

Rappresentazione delle azioni

Assiomi delle precondizioni: se l'azione A è eseguita al tempo i, allora le sue precondizioni sono vere al tempo i, per $i = 0, ..., k$ (k limite alla lunghezza del piano)

$$putOnTable\_A\_B^i \rightarrow clear\_A^i \wedge on\_A\_B^i$$

$$stack\_A\_B^i \rightarrow onTable\_A^i \wedge clear\_A^i \wedge clear\_B^i$$

$$move\_A\_B\_C^i \rightarrow clear\_A^i \wedge clear\_C^i \wedge on\_A\_B^i$$

(assumiamo che il linguaggio non contenga atomi che rappresentano fluenti o azioni impossibili, come $on\_A\_A^i$ o $putOn\_A\_A^i$)

Dalla rappresentazione STRIPS: per ogni istanza ground di azione A, se Pre_A è la congiunzione di letterali dichiarata nel campo `:precondition` di A, e per ogni $i = 0, ...k$, la codifica contiene la formula

$$A^i \rightarrow Pre_A^i$$

Assiomi dello stato successore:

$$
\begin{aligned}
on\_A\_B^{i+1} \equiv\ & (move\_A\_C\_B^i \vee stack\_A\_B^i) \vee \\
& (on\_A\_B^i \wedge \neg(putOnTable\_A^i \vee putOn\_A\_C^i))
\end{aligned}
$$

$$
\begin{aligned}
onTable\_A^{i+1} \equiv\ & (putOnTable\_A\_B^i \vee putOnTable\_A\_C^i) \vee \\
& (onTable\_A^i \wedge \neg(stack\_A\_B^i \vee stack\_A\_C^i))
\end{aligned}
$$

$$
\begin{aligned}
clear\_A^{i+1} \equiv\ & (move\_B\_A\_C^i \vee move\_C\_A\_B^i \vee putOnTable\_B\_A^i \vee putOnTable\_C\_A^i) \vee \\
& (clear\_A^i \wedge \neg(move\_B\_C\_A^i \vee move\_C\_B\_A^i \vee stack\_B\_A^i \vee stack\_C\_A^i))
\end{aligned}
$$

Dalla rappresentazione STRIPS: per ogni fluente (ground) P, e per ogni $i = 0, ...k$, si calcolano:

- $(G^+)^i$ la disgiunzione degli atomi A^i tali che A è un'azione (ground) che ha P nel campo `:effect`;
- $(G^-)^i$ la disgiunzione degli atomi A^i tali che A è un'azione (ground) che ha $\neg P$ nel campo `:effect`

La codifica contiene allora (per ogni P e ogni $i = 0, ..., k$) la formula

$$P^{i+1} \equiv (G^+)^i \vee (P^i \wedge \neg(G^-)^i)$$

Mutua esclusione tra azioni

Se due azioni hanno precondizioni o effetti inconsistenti, non potranno essere contemporaneamente vere in nessuna interpretazione che sia un modello degli assiomi delle precondizioni e degli assiomi dello stato successore

Ma se l'azione A "cancella" una precondizione di B, o viceversa, l'incompatibilità di A e B va rappresentata esplicitamente.

Assiomi di esclusione tra azioni: per ogni coppia di azioni (ground) A e B che **interferiscono** una con l'altra (gli effetti di A contengono un letterale complementare a una precondizione di B, o viceversa), la codifica contiene (per ogni $i = 0, ..., k$):

$$\neg(A^i \wedge B^i)$$

$$\neg(move\_A\_C\_B^i \wedge move\_A\_B\_C^i)$$
$$\neg(move\_A\_C\_B^i \wedge putOnTable\_A\_C^i)$$
$$\ldots$$

Pianificazione in logica proposizionale: algoritmo

Input: descrizione STRIPS del problema

Inizializzare $T \leftarrow 0$, $found \leftarrow false$

while not $found$ **do**
- costruire la codifica S_T del problema limitato al tempo T
- cercare un modello $\mathcal{M}$ di S_T
- se la ricerca ha successo
  - allora porre $found \leftarrow true$
  - altrimenti porre $T \leftarrow T + 1$

done

estrarre il piano da $\mathcal{M}$ e riportare la soluzione

Pianificatori basati sulla ricerca di modelli

SATPLAN e **BLACKBOX** sono algoritmi di pianificazione basati sulla logica proposizionale.

- sono abbastanza efficienti, grazie allo sviluppo di algoritmi efficienti per la logica proposizionale;
- quando il problema ha una soluzione, riportano una soluzione "ottimale" in termini di numero di passi necessari per raggiungere l'obiettivo (non necessariamente numero di azioni);
- se il problema non ha soluzione, non terminano a meno di non porre un limite massimo di tempo (cosa che, ad ogni modo, rovina la completezza)

Pianificazione in logica temporale lineare: Pdk

`http://pdk.dia.uniroma3.it`

Approccio alternativo: uso di una **logica proposizionale temporale**, contenente operatori che consentono di riferirsi al futuro:

$$\begin{array}{ll} \Box A: & A \text{ è sempre vero (adesso e nel futuro)} \\ \Diamond A: & A \text{ sarà vero prima o poi (adesso o nel futuro)} \\ \bigcirc A: & A \text{ è vero nel prossimo istante temporale} \end{array}$$

- Rappresentazione dello stato iniziale:

$$on\_A\_B \wedge onTable\_B \wedge onTable\_C \wedge clear\_A \wedge clear\_C \wedge$$
$$\neg on\_A\_C \wedge \neg on\_B\_A \wedge \neg on\_B\_C \wedge \neg on\_C\_A \wedge \neg on\_C\_B$$
$$\wedge \neg onTable\_A \wedge \neg clear\_B$$

- Rappresentazione dell'obiettivo:

$$\Diamond(on\_A\_B \wedge on\_B\_C \wedge onTable\_C)$$

- Assiomi delle precondizioni:

$$\Box(putOnTable\_A\_B \rightarrow clear\_A \wedge on\_A\_B)$$

- Asiomi dello stato successore:

$$\begin{array}{rl} \bigcirc on\_A\_B \equiv & (move\_A\_C\_B \vee stack\_A\_B) \vee \\ & (on\_A\_B \wedge \neg(putOnTable\_A\_B \vee move\_A\_B\_C)) \end{array}$$

- Assiomi di esclusione:

$$\Box\neg(move\_A\_C\_B \wedge move\_A\_B\_C)$$

Il linguaggio di pianificazione usato da Pdk

Pdk utilizza un linguaggio di pianificazione simile a PDDL:

- stessa sintassi
- alcune restrizioni (accetta comunque STRIPS)
- alcune estensioni:
  - è possibile specificare proprietà del mondo sempre vere (che vengono utilizzate per semplificare la codifica del problema, anche per eliminare azioni);
  - è possibile definire predicati, anche ricorsivamente (ad esempio, le "buone torri" nel mondo dei blocchi);
  - è possibile specificare **conoscenza di controllo**, mediante "schemi di controllo": campi addizionali nella definizione degli operatori.
  - è possibile fare riferimento all'obiettivo e allo stato iniziale.

Esempio: nel mondo dei blocchi

```
.........

(:theory (forall (?x) (not (on ?x ?x))) )

(:define goodTower (?x)
         (or (and (goal(ontable ?x)) (ontable ?x))
             (exists (?y) (and (goal(on ?x ?y))
                               (on ?x ?y)
                               (goodTower ?y)))))

(:action unstack :parameters (?x)
        :precondition (clear ?x) (not (ontable ?x))
        :effect
           (and (ontable ?x)
                (forall (?from)
                      (when (on ?x ?from)(and (clear ?from)
                                              (not(on ?x ?from)))))
        :only-if (not(goodTower ?x))
        :s-asap (goal (ontable ?x)))

...........
```