Idea chiave della pianificazione

"Aprire" la rappresentazione di stati, obiettivo e azioni: descrizione in qualche linguaggio formale (logica del primo ordine o sottolinguaggio)

stati e obiettivo : rappresentati da insiemi di enunciati

azioni : descrizione logica di precondizioni e effetti

Pianificazione basata sul Situation Calculus:

- non ci sono restrizioni di linguaggio: si utilizza la logica dei predicati;
- la specifica di un problema è codice Prolog, a volte non facile da scrivere;
- la codifica del problema non è generale: utilizzabile soltanto dai pianificatori GOLOG.

Linguaggi di pianificazione

Restrizione del linguaggio di rappresentazione: uso di un linguaggio *ad hoc*, che si può vedere come la restrizione di un linguaggio logico. La specifica di un problema è generale: può essere utilizzata da diversi pianificatori.

Pianificazione basata su search

Uso di un algoritmo specializzato (*planner*) invece di un theorem prover generale. Il planner può codificare anche principi "difficili" da esprimere in modo dichiarativo.

Il linguaggio di pianificazione STRIPS

La rappresentazione dei problemi di pianificazione deve permettere ai pianificatori di trarre vantaggio dalla struttura logica del problema.

Linguaggio di pianificazione: sufficientemente espressivo, ma abbastanza ristretto da poter essere utilizzato da parte di algoritmi efficienti

STRIPS utilizza un linguaggio della logica dei predicati **senza simboli funzionali**

Stati : congiunzioni di **atomi ground** (senza variabili)

$$at(home)$$

In particolare, lo **stato iniziale** è rappresentato da una congiunzione di atomi ground

Ipotesi del mondo chiuso: quello che non viene specificato è falso.

Nello stato rappresentato da $at(home)$, l'atomo $have(milk)$ è falso.

Obiettivo : congiunzione di **atomi ground**

$$at(home) \wedge have(milk)$$

Uno stato soddisfa un obiettivo G se contiene tutti gli atomi di G (possibilmente anche altri)

Azioni : specificate da **SCHEMI DI OPERATORE**, in cui le variabili (i parametri) sono intese come quantificate universalmente.

Uno schema di operatore è costituito da

- **nome** dell'operatore
- **parametri**
- **precondizioni** : congiunzione (o lista) di atomi (positivi)
- **effetti** : congiunzione (o lista) di letterali;
  a volte descritti come una coppia di insiemi di atomi: una *ADD-LIST* e una *DELETE-LIST*

```
Action (go(from to)
       Precondition:  at(from) ∧ path(from,to)
       Effect:  at(to) ∧ ¬ at(from))
```

Nella sintassi del linguaggio PDDL (Planning Domain Description Language, un'estensione di STRIPS) accettato da molti pianificatori:

```
(:action go
         :parameters (?from ?to)
         :precondition (and (at ?from) (path ?from ?to))
         :effect (and (at ?to) (not (at ?from))))
```

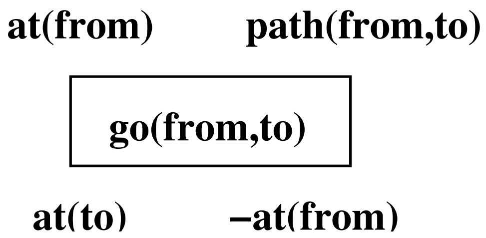
N.B. Le variabili che possono occorrere in precondizioni ed effetti sono soltanto i parametri dell'operatore

Azioni e operatori

Ogni istanza ground di un operatore è un'azione

Quindi un operatore rappresenta un insieme di azioni

Rappresentazione grafica di un operatore



Esercizio: è corretto definire l'operatore `go` come segue?

```
Action (go(there)
       Precondition:  at(here) ∧ path(here,there)
       Effect:  at(there) ∧ ¬ at(here))
```

La semantica del linguaggio STRIPS

Uno stato S soddisfa una congiunzione di atomi G se S contiene tutti gli atomi di G

Un'azione A è **applicabile** in uno stato S se S soddisfa la precondizione di A

Un operatore è applicabile in uno stato S se è applicabile in S un'istanza dell'operatore: è possibile istanziare le variabili dell'operatore con una sostituzione θ in modo tale che S soddisfa $P\theta$, dove P è la precondizione dell'operatore.

Risultato dell'applicazione di un'azione A a uno stato S (a cui l'azione è applicabile): lo stato risultante è quello che contiene

- tutti gli atomi (positivi) dell'effetto di A
- tutti gli atomi di S, eccetto quelli che occorrono negati negli effetti di S

Esempio:

$$S: \ at(home) \land \ path(home, super) \land have(money)$$

L'operatore go è applicabile con $from = home$ e $to = super$:

$$go(home, super)$$

L'applicazione dell'azione $go(home, super)$ in S risulta nello stato

$$S' : at(super) \land path(home, super) \land have(money)$$

Assunto di STRIPS: i letterali non menzionati nell'effetto non sono modificati

Così STRIPS evita il problema di rappresentazione del frame (i *frame axioms* sono impliciti nella semantica del linguaggio)

Gli oggetti del dominio

In pianificazione classica il mondo è finito e discreto: la specifica STRIPS di un problema contiene anche la definizione dell'insieme degli oggetti del dominio, denotati da un insieme di **costanti**

```
Objects: (home super milk banana money)
```

Nella sintassi di PDDL:

```
(:objects home super milk banana money)
```

STRIPS assume che gli oggetti del dominio siano **solanto** quelli dichiarati nella specifica

Un esempio completo in PDDL

Definizione del **dominio** di pianificazione:

```
(define (domain blocksworld)

   (: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))))
)
```

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

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

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)

Definizione di un **problema** su questo dominio di pianificazione:

```
(define (problem blocks3-reverse)
  (:objects A B C)
  (:init (and (on A B) (on B C) (onTable C) (clear A)))
  (:goal (and (on C B) (on B A) (onTable A)))
)
```

Proposizionalizzazione del problema

Assumendo che gli oggetti del dominio siano soltanto quelli dichiarati nella specifica (e data l'assenza di simboli funzionali), ogni schema di operatore può essere proposizionalizzato: trasformato in un insieme finito di azioni (prive di variabili)

PDDL: Planning Domain Definition Language

Alcune delle estensioni di STRIPS previste da PDDL:

- Dichiarazione di **tipi** e tipizzazione degli oggetti

```
(define (problem gripper)
   (:requirements :typing)
   (:fluents (at ?b - ball ?r - room)
             (atRobby ?r - room)
             (free ?g - gripper)
             (carry ?b - ball ?g - gripper))
   (:action pick :parameters (?b - ball ?r - room ?g - gripper)
        :precondition (atRobby ?r)(at ?b ?r)(free ?g)
        :effect (not(at ?b ?r))(not(free ?g))(carry ?b ?g))
...)

(define (problem balls4)
   (:objects A B - room
          b1 b2 b3 b4 - ball
          Left Right - gripper)
...)
```

La tipizzazione non aggiunge potere espressivo, ma limita il numero di possibili istanze proposizionali degli operatori

- Azioni con **effetti condizionali**, con variabili quantificate universalmente

```
(:action putOnTable
        :parameters (?x)
        :precondition (and (clear ?x) (not (onTable ?x)))
        :effect (and (onTable ?x)
                     (forall (?from)
                         (when (on ?x ?from)
                               (and (clear ?from)
                                    (not (on ?x ?from))))))
```

Semantica: se l'effetto dell'azione A con parametri $x_1, ..., x_n$ contiene

$$\forall y\ (\texttt{when}\ P(x_1, ..., x_n, y)\ E(x_1, ..., x_n, y))$$

allora l'esecuzione di $A(c_1, ..., c_n)$ nello stato S (se applicabile) provoca nello stato risultante l'effetto $E(c_1, ..., c_n, y)$ **per ogni oggetto** y **tale che** S **soddisfa** $P(c_1, ..., c_n, y)$

Esercizio: definire l'azione `putOn` con due parametri, `?x` e `?onto`, utilizzando effetti condizionali quantificati universalmente in modo tale che rappresenti l'azione di spostare un blocco su un altro sia a partire da un altro blocco (`move`), sia a partire dal tavolo (`stack`)
(per ogni oggetto `?from`, se `?x` è su `?from`, allora ...; se `?x` è sul tavolo, allora ...)

- Sono ammesse **precondizioni negative e disgiuntive** nelle azioni. Negazione e disgiunzione sono ammesse anche nella specifica del goal

  N.B. Ammettere **effetti disgiuntivi** delle azioni significa includere una forma di non determinismo, che si deve trattare in modo diverso: pianificazione condizionale o monitoraggio dell'esecuzione

Esercizi

Considerare i seguenti problemi e formularne una descrizione in STRIPS o in PDDL

1. **Dominio:** una scimmia è in una stanza, dove delle banane sono appese al soffitto, fuori della sua portata. Nella stanza c'è anche un panchetto, sul quale la scimmia potrebbe salire per raggiungere le banane. Le azioni possibili per la scimmia sono: andare da un posto all'altro nella stanza, spingere un oggetto da un posto all'altro, salire su un oggetto e afferrare un oggetto.

   Problema: le posizioni della stanza sono A, B, C e D; inizialmente, la scimmia è in A, le banane sono appese in B e il panchetto è in C. Gli altri oggetti presenti nella stanza sono una palla, inizialmente in D, e un libro, inizialmente in B. L'obiettivo è quello di avere le banane.

2. **Dominio:** Un robot con un solo braccio agisce in un ambiente costituito da una sola stanza e un corridoio. Le azioni che può compiere sono entrare e uscire dalla stanza, aprire borse, chiudere borse, mettere oggetti piccoli dentro borse (solo se queste sono aperte), prendere in mano borse (solo se queste sono chiuse).

   Problema: nello stato iniziale il robot è fuori della stanza. Nella stanza ci sono un tavolo (grande), una borsa chiusa e un libro (piccolo) appoggiato sul tavolo. L'obiettivo del robot è quello di trovarsi fuori della stanza, con il libro dentro la borsa.

3. **Dominio:** un robot si può muovere tra diversi posti e comprare oggetti. Per comprare un oggetto deve essere in un posto in cui si vende tale oggetto. Dopo aver comprato un oggetto, il robot lo possiede.

   Problema: i posti dell'ambiente sono $casa$, $super$ (negozio di alimentari) e hs (hardware store, che vende attrezzi da bricolage). Inizialmente il robot è a casa e non possiede alcun oggetto. L'obiettivo è quello di essere a casa con un litro di latte, una banana e un trapano.

4. **Dominio (Gripper):** Un robot con due braccia agisce in un ambiente costituito da due stanze, A e B. Ogni braccio del robot può tenere un solo oggetto alla volta. Le azioni che il robot può compiere sono "prendere la palla X con la mano Y nella stanza Z", "lasciare la palla X tenuta con la mano Y nella stanza X", andare dalla stanza X alla stanza Y".

   Problema: Inizialmente la stanza A contiene 4 palle. L'obiettivo è quello di avere le 4 palle nella stanza B.

5. **Dominio (Briefcase):** Un robot agisce in un ambiente costituito da un certo numero di locazioni, e dispone di una borsa che può trasportare oggetti. Si assume che la borsa sia di capienza illimitata. Le azioni che può compiere sono andare da una locazione all'altra, mettere un oggetto nella borsa, togliere un oggetto dalla borsa.

   Problema: nell'ambiente ci sono 4 locazioni ($casa$, A. B, C) e 3 oggetti. Inizialmente il robot è a $casa$, l'oggetto 1 è in A, l'oggetto 2 è in B e l'oggetto 3 è in C. L'obiettivo è quello di avere tutti e 3 gli oggetti a $casa$ (fuori della borsa).

6. **Dominio (Tea):** Un robot agisce in un ambiente costituito da un certo numero di stanze e un corridoio. Da qualche parte ci sono una macchina del tè e una pila di tazze. Il robot può andare da una locazione all'altra (stanze o corridoio), purché origine e destinazione siano connesse, può prendere una tazza vuota se non ha oggetti in mano, può riempire di tè la tazza vuota che ha in mano, può consegnare il tè a un abitante di una stanza che lo abbia ordinato.

   Problema: L'ambiente è costituito da 4 stanze, ciascuna con un abitante che ha ordinato il tè. Le stanze sono tutte connesse al corridoio; inoltre la stanza 1 è connessa alla stanza 2, e la stanza 3 alla 4. La macchina del tè è nella stanza 1 e la pila di tazze nella stanza 2. L'obiettivo è di consegnare il tè a tutti e 4 gli abitanti delle stanze.