Ciclo di lezioni su pianificazione automatica

- Riprendere alcuni concetti visti nel modulo di Fondamenti di Intelligenza Artificiale (ricerca, logica e risoluzione) e vederne l'applicazione per costruire sistemi di pianificazione automatica
  - Pianificazione come deduzione
  - Pianificazione come ricerca nello spazio degli stati
  - Pianificazione come ricerca nello spazio dei piani

Automated planning

La pianificazione automatica **(automated planning)** rappresenta un'importante attività di *problem solving* che consiste nel sintetizzare una sequenza di azioni che eseguite da un agente a partire da uno stato "iniziale" del mondo provocano il raggiungimento di uno stato "desiderato".

Esempio

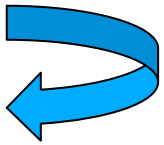
- Pianificare è una attività che tutti gli esseri umani svolgono nel loro quotidiano

- Supponiamo di avere come obiettivo (**goal**) di seguire una lezione di Intelligenza Artificiale e di essere attualmente a casa e di possedere un'automobile
- (**stato iniziale**)

- Al fine di raggiungere lo scopo prefisso dobbiamo fare una serie di **azioni** in una certa sequenza

Esempio: sequenza di azioni:

1. *prendere materiale necessario per gli appunti,*
2. *prendere le chiavi dell'auto,*
3. *uscire di casa,*
4. *prendere l'auto,*
5. *raggiungere la facoltà,*
6. *entrare in aula e così via*.



- Pianificare ci permette di fare le azioni giuste nella sequenza giusta: ad esempio, non possiamo pensare di invertire l'ordine delle azioni 2 e 3.

Esempio

- Per alcune di queste azioni non è necessario pianificare l'ordine (il piano può contenere un ordinamento parziale).
- Ad esempio, invertendo l'ordine delle azioni 1 e 2 si ottiene un piano corretto analogo al precedente:

1. *prendere le chiavi dell'auto,*
2. *prendere materiale necessario per gli appunti,*
3. *uscire di casa,*
4. *prendere l'auto,*
5. *raggiungere la facoltà,*
6. *entrare in aula e così via*.

Esempio

- Ci possono essere piani alternativi per raggiungere lo stesso obiettivo (posso pensare di andare in facoltà a piedi o in autobus), ad esempio:

1. *prendere materiale necessario per gli appunti,*
2. *prendere biglietto autobus,*
3. *uscire di casa,*
4. *raggiungere la fermata,*
5. *salire sull'autobus,*
6. *raggiungere la facoltà,*
7. *entrare in aula e così via.*

Planning

Dati:

- uno stato iniziale
- un insieme di azioni eseguibili
- un obiettivo da raggiungere (**goal**)

un problema di pianificazione consiste nel determinare un **piano**, ossia un insieme (parzialmente o totalmente) ordinato di azioni necessarie per raggiungere il goal.

Pianificare è:

- Un'applicazione di per se'
- un'attività comune a molte applicazioni quali
  - diagnosi: pianificazione di test o azioni per riparare (riconfigurare) un sistema
  - scheduling
  - robotica, etc

La Pianificazione Automatica: alcuni concetti base

- Un **pianificatore automatico** è un *agente intelligente* che opera in un certo dominio e che date:

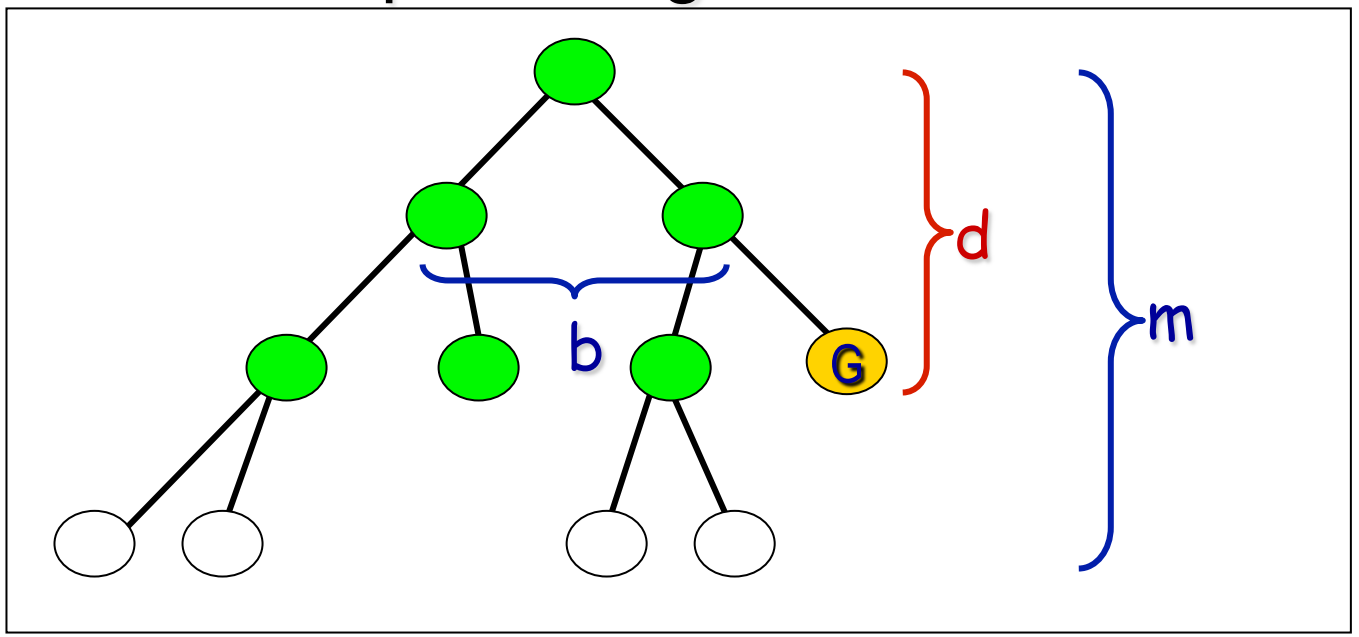
1. una rappresentazione dello stato iniziale
2. una rappresentazione del goal
3. una descrizione formale delle azioni eseguibili

- sintetizza dinamicamente il piano di azioni necessario per raggiungere il goal a partire dallo stato iniziale.

Vi ricorda qualcosa?

Vi ricorda qualcosa?

- Ricerca nello spazio degli stati



- Infatti una delle tecniche di pianificazione automatica è proprio mediante ricerca nello spazio degli stati, ma non è l'unica ….

Ricerca nello spazio degli stati

- Uno **stato iniziale** in cui l'agente sa di trovarsi;
- Un **insieme di azioni** possibili che sono disponibili da parte dell'agente (operatori che trasformano uno stato in un altro o più formalmente una funzione successore S(X) che riceve in ingresso uno stato e restituisce l'insieme degli stati raggiungibili);
- Uno **stato obiettivo** o **goal** in cui l'agente vuole portarsi.

Un **cammino** è una sequenza di azioni che conduce da uno stato a un altro.

Problem-solving agents (da Russel, Norvig)

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

    state ← UPDATE-STATE(state, percept)
    if seq is empty then do
        goal ← FORMULATE-GOAL(state)
        problem ← FORMULATE-PROBLEM(state, goal)
        seq ← SEARCH(problem)
    action ← FIRST(seq)
    seq ← REST(seq)
    return action
```

Planning: gli ingredienti

1. una rappresentazione dello stato iniziale
2. una rappresentazione del goal
3. una descrizione formale delle azioni eseguibili

Planning: gli ingredienti

1. una rappresentazione dello stato iniziale
2. una rappresentazione del goal
3. una descrizione formale delle azioni eseguibili

Rappresentazione dello stato

- Occorre fornire al pianificatore un modello del sistema su cui opera.
- In genere lo stato è **rappresentato in forma dichiarativa** con una congiunzione di formule atomiche che esprime la situazione di partenza. Ad esempio:

$on(book,table) \wedge name(book,xyz) \wedge atHome(table)$

Rappresentazione dello stato

- Lo stato di un sistema spesso può essere osservato solo in modo ***parziale*** per una serie di motivi:
  - perché alcuni aspetti **non sono osservabili**;
  - perché il **dominio è troppo vasto** per essere rappresentato nella sua interezza (limitate capacità computazionali per la rappresentazione);
  - perché le osservazioni sono soggette a **rumore** e quindi si hanno delle osservazioni parziali o imperfette;
  - perché il dominio è troppo **dinamico** per consentire un aggiornamento continuo della rappresentazione.

Planning: gli ingredienti

1. una rappresentazione dello stato iniziale
2. una rappresentazione del goal
3. una descrizione formale delle azioni eseguibili

Rappresentazione del goal

- Per rappresentare il goal si utilizza lo stesso linguaggio formale con cui si esprime lo stato iniziale.
- In questo caso, la congiunzione rappresenta una descrizione parziale dello stato finale che si vuole raggiungere: descrive solo le condizioni che devono essere verificate affinché il goal sia soddisfatto.

- Ad esempio:

on(book,shelf)

Planning: gli ingredienti

1. una rappresentazione dello stato iniziale
2. una rappresentazione del goal
3. una descrizione formale delle azioni eseguibili

Rappresentazione delle azioni

È necessario fornire al pianificatore una descrizione formale delle azioni eseguibili detta ***Teoria del Dominio***.

Ciascuna azione è identificata da un nome e modellata in forma dichiarativa per mezzo di ***precondizioni*** e ***postcondizioni***.

Le precondizioni rappresentano le condizioni che devono essere verificate affinché l'azione possa essere eseguita; le postcondizioni rappresentano gli effetti dell'azione stessa sul mondo.

Spesso la Teoria del Dominio è costituita da operatori con variabili che definiscono **classi di azioni**. A diverse istanziazioni delle variabili corrispondono diverse azioni.

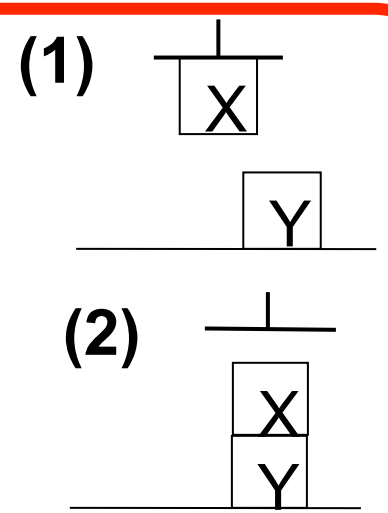
Esempio: mondo a blocchi

Spostare blocchi su un tavolo con un braccio; azioni:

- **STACK(X,Y)**

  SE: holding(X) and clear(Y)

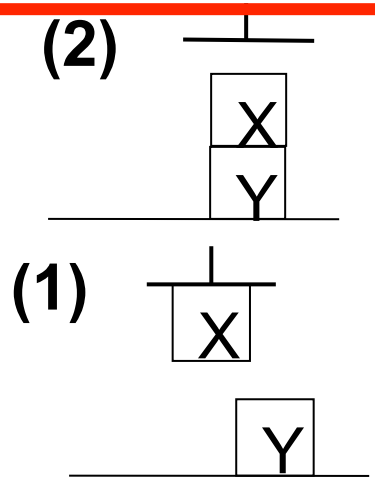
  ALLORA: handempty and clear(X) and on(X,Y);



- **UNSTACK(X,Y)**

  SE: handempty and clear(X) and on(X,Y)

  ALLORA: holding(X) and clear(Y);



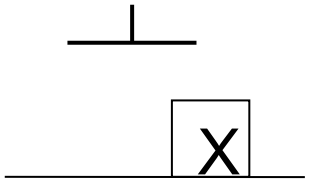
Esempio: mondo dei blocchi

- **PICKUP(X)**

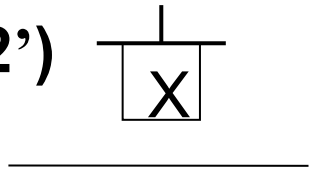
  SE: ontable(X) and clear(X) and handempty

  ALLORA: holding(X);

(1')



(2')

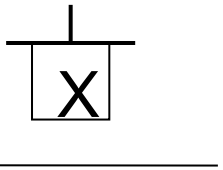


- **PUTDOWN(X)**

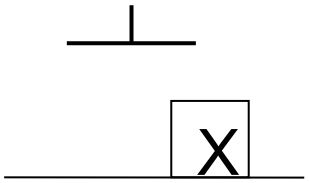
  SE: holding(X)

  ALLORA: ontable(X) and clear(X) and handempty.

(2')



(1')



Un altro linguaggio:

- Azioni: espresse mediante operatori (o schemi di azione)

Stack(x, y)

Precondizioni:

Clear(x), Table(x), Clear(y)	precondizioni

Effetti:

Add-list: On(x, y)	effetto positivo
Delete-list: Table(x), Clear(y)	effetto negativo

Unstack(x, y)

Precondizioni:

Clear(x), On(x, y)	precondizioni

Effetti:

Add-list: Table(x), Clear(y)	effetto positivo
Delete-list: On(x, y)	effetto negativo

Planning Domain Definition Language (PDDL)

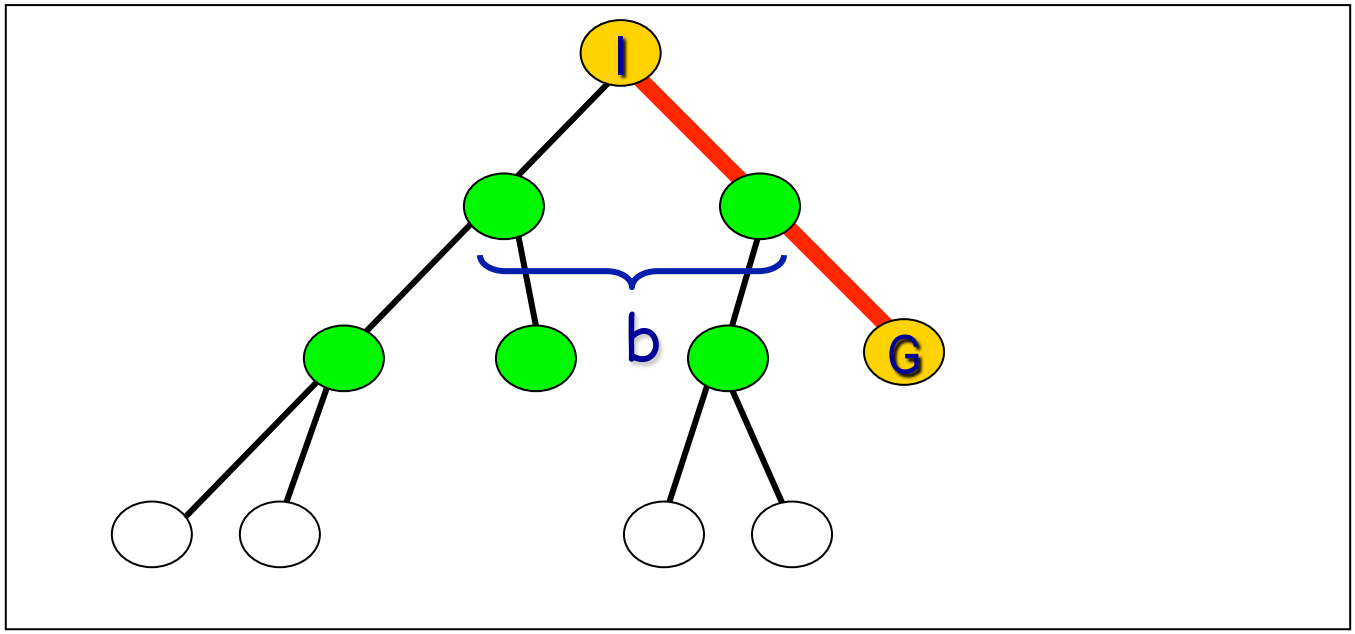
- E’ un tentativo di standardizzare I linguaggi di descrizione del dominio e delle azioni per la pianificazione automatica

- (include STRIPS, ADL, etc)

Planning: come te lo cucino …

Planning: come te lo cucino …

- Ricerca



Planning: come te lo cucino …

- La pianificazione automatica cerca una sequenza di azioni che raggiungano l'obiettivo dallo stato iniziale
- Nella nostra metafora, il pianificatore produce una "ricetta" (appunto la sequenza di azioni)
- Tecniche diverse di pianificazione automatica, in base a quali sono gli operatori …

Planning come ricerca

- Lo spazio di ricerca è definito da che cosa sono gli stati e gli operatori:
  - **Pianificazione come theorem proving**: stati come insiemi di formule e operatori come regole di inferenza
  - **Pianificazione nello spazio degli stati**: stati come descrizioni di situazioni e operatori come modifiche dello stato (sono le azioni stesse)
  - **Pianificazione nello spazio dei piani**: stati come piani parziali e operatori di raffinamento e completamento di piani

Pianificazione

Processo per il calcolo dei diversi passi di una procedura di soluzione di un problema di pianificazione:

- *non decomponibile*

  ci può essere interazione fra i sottoproblemi

- *reversibile*

  le scelte fatte durante la **generazione** del piano sono revocabili (backtracking).

Un pianificatore è **completo** quando riesce sempre a trovare la soluzione se esiste.

Un pianificatore è **corretto** quando la soluzione trovata porta dallo stato iniziale al goal finale in modo consistente con eventuali vincoli.

Esecuzione

Processo di applicazione della procedura di soluzione

- *irreversibile*

  l'esecuzione delle azioni determina spesso un cambiamento di stato non reversibile,

- *non deterministica*

  il piano può avere un effetto diverso quando applicato al mondo reale che è spesso impredicibile. In questo caso è possibile rifare il piano solo parzialmente, oppure invalidarlo tutto a seconda del problema.

Pianificazione classica

- Tipo di pianificazione *off-line* che produce l'intero piano prima di eseguirlo lavorando su una rappresentazione istantanea (*snapshot*) dello stato corrente.

- È basata su alcune assunzioni forti:
  - tempo atomico di esecuzione delle azioni
  - determinismo degli effetti
  - stato iniziale completamente noto a priori
  - esecuzione del piano unica causa di cambiamento del mondo

Pianificazione reattiva

- Metodo di pianificazione *on-line*

  - considera l'ambiente non deterministico e dinamico
  - è capace di osservare il mondo sia in fase di pianificazione sia in fase di esecuzione (*sensing actions*)
  - spesso alterna il processo di pianificazione a quello di esecuzione reagendo ai cambiamenti di stato

Cosa vedremo:

- Tecniche di Pianificazione Classica
  - Planning Deduttivo
    - Situation Calculus
      - Formalizzazione di Green
      - Formalizzazione di Kowalsky
  - Planning mediante ricerca
    - Ricerca nello spazio degli stati ⇨ Planning Lineare
      - STRIPS
      - Esercitazioni Prolog/STRIPS forward
    - Ricerca nello spazio dei piani ⇨ Planning Non Lineare
      - Partial Order Planning (POP)
- Cenni ad altre tecniche

Planning Deduttivo

- La tecnica di **pianificazione deduttiva** utilizza la logica per rappresentare stati, goal e azioni e genera il piano come dimostrazione di un teorema

- La tecnica nasce dal Calcolo di Situazioni, una formalizzazione in First Order Logic (FOL) (McCarthy 1969) che tratta stati, cambiamenti e azioni

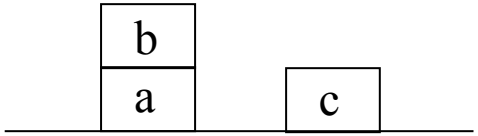
Situation Calculus *(fine anni '60)*

Formalizzazione del linguaggio (basato sulla logica dei predicati del primo ordine) in grado di rappresentare stati e azioni in funzione del tempo

- **Situation**: "fotografia" del mondo e delle proprietà (*fluent*) che valgono in un determinato istante/stato s
  - Esempio: mondo dei blocchi

    on(b,a,s0)

    ontable(c,s0)



- **Azioni**: definiscono quali *fluent* saranno veri come risultato di un'azione. Esempio

on(X,Y,S) and clear(X,S) → **Pre-condizioni**

(ontable(X,do(putOnTable(X),S))) and

(clear(Y,do(putOnTable(X),S))) **Post-condizioni**

Situation Calculus

- **Costruzione di un piano**: deduzione, dimostrazione di un goal
  - Esempio

    :- ontable(b,S). Significa: esiste uno stato S in cui è vero ontable(b)?

    YES per S=do(putOntable(b),s0)

- **Vantaggi**: elevata espressività, permette di descrivere problemi complessi

- **Problema**: *frame problem*

Frame problem

- Il problema del contorno (*frame problem*) è uno dei problemi più classici dell'AI [McCarthy-Hayes, 1969]

- Nello stato risultante non si sa nulla delle cose che erano vere (o false) prima dell'azione e che rimangono vere (o false) perché non influenzate. E queste sono la maggioranza!

- Analogia col mondo dell'animazione: il *frame problem* è il problema di distinguere il *background* (che rimane fisso) dal *foreground* (che cambia).

Frame problem

- Occorre specificare esplicitamente tutti i *fluent* che cambiano dopo una transizione di stato e anche quelli che NON cambiano (*assiomi di sfondo*: "Frame axioms").

- Al crescere della complessità del dominio il numero di tali assiomi cresce enormemente.

- Il problema della rappresentazione della conoscenza diventa intrattabile

Pianificazione come deduzione (Green)

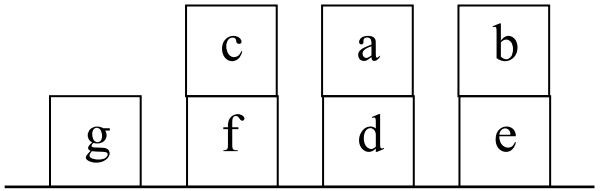
- Green usa il *situation calculus* per costruire un pianificatore basato sul metodo di risoluzione.

- Si cerca la dimostrazione di una formula contenente una variabile di stato che alla fine della dimostrazione sarà istanziata al piano di azioni che permette di raggiungere l'obiettivo (*backward*)

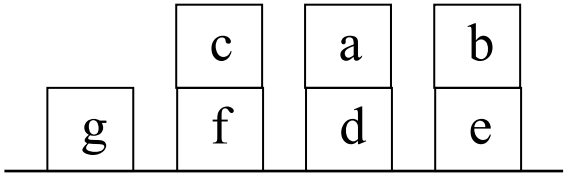
Esempio

- I seguenti assiomi descrivono tutte le relazioni vere nello stato iniziale s0:

A.1 *on(a,d,s0).*
A.2 *on(b,e,s0).*
A.3 *on(c,f,s0).*
A.4 *clear(a,s0).*
A.5 *clear(b,s0).*
A.6 *clear(c,s0).*
A.7 *clear(g,s0).*
A.8 *diff(a,b)*
A.9 *diff(a,c)*
A.10 *diff(a,d)…*



Esempio



- Le azioni si esprimono con assiomi nella forme a clausole

L'azione *move(X,Y,Z)*

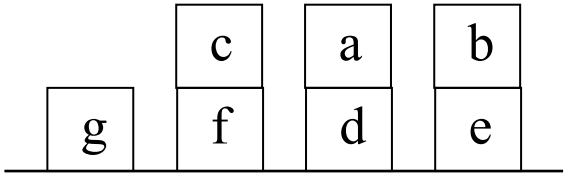
clear(X,S) and clear(Z,S) and on(X,Y,S) and diff(X,Z)
→ *clear(Y,do(move(X,Y,Z),S)), on(X,Z,do(move(X,Y,Z),S))*.

che sposta un blocco X da Y a Z, partendo dallo stato S e arriva allo stato *do(move(X,Y,Z),S)*) si esprime con i seguenti assiomi (*effect axioms,* clausole)

A.11 *~clear(X,S)* or *~clear(Z,S)* or *~on(X,Y,S)* or *~diff(X,Z) or clear(Y,do(move(X,Y,Z),S))*.

A.12 *~clear(X,S)* or *~clear(Z,S)* or *~on(X,Y,S)* or *~diff(X,Z) or on(X,Z,do(move(X,Y,Z),S))*.

Esempio



Dato un goal vediamo un esempio di come si riesce a trovare una soluzione usando il metodo di risoluzione:

GOAL :- on(a,b,S1)

~on(a,b,S1)

(A.12) {X/a,Z/b,S1/do(move(a,Y,b),S)}

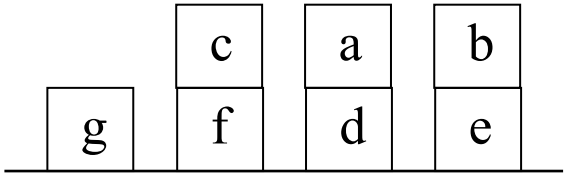
~clear(a,S) or ~clear(b,S) or ~on(a,Y,S) or ~diff(a,b)

(A.4)	*(A.5)*	*(A.1)*	*(A.8)*
{S/s0},	*{S/s0},*	*{S/s0, Y/d}*	*true*

~on(a,b,S1) porta a una contraddizione quindi on(a,b,S1) risulta dimostrato con la sostituzione *S1/do(move(a,d,b),s0)*

Questo è il piano: *S1/do(move(a,d,b),s0)*

Esempio



Supponiamo di voler risolvere un problema un po' più complesso

Goal: :-on(a,b,S), on(b,g,S).

Soluzione*: S/do(move(a,d,b),do(move(b,e,g),s0)).*

Ovvero, spostiamo b da e su g, e poi a da d su b

Per poterlo risolvere occorre una descrizione completa dello stato risultante dall'esecuzione di ciascuna azione.

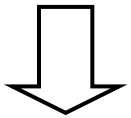
Frame problem

Per descrivere un'azione oltre agli *effect axioms* occorre specificare tutti i *fluent* che NON sono invalidati dall'azione stessa (*frame axioms)*. Nel nostro esempio occorrono i seguenti assiomi:

on(U,V,S) and diff(U,X) → *on(U,V,do(move(X,Y,Z),S))*

clear(U,S) and diff(U,Z) → *clear(U,do(move(X,Y,Z),S))*

Occorre esplicitare un *frame axiom* per ogni relazione NON modificata dall'azione.



Se il problema è complicato la complessità diventa inaccettabile

Esercizi proposti

- *Per il problema del mondo a blocchi, nella formulazione di Green trovare, tramite risoluzione, la soluzione al goal:*

:-on(b,g,S), on(a,b,S).

- *Per il problema del mondo a blocchi, nella formulazione di Kowalski, trovare tramite risoluzione, la soluzione al goal:*

:-poss(S),holds(on(b,g),S),holds(on(a,b),S).

Formulazione di Kowalsky

Rappresenta una formulazione più semplice:

- Viene utilizzato il predicato *holds(rel, s/a)* per indicare tutte le relazioni *rel* vere in un certo stato *s* o rese vere da una certa azione *a*
- Il predicato *poss(s)* indica che uno stato *s* è possibile ovvero raggiungibile.
- Si utilizza il predicato *pact(a,s)* per affermare che è lecito compiere una determinata azione *a* in un particolare stato *s*, ovvero le precondizioni per svolgere l'azione sono soddisfatte in s.

Formulazione di Kowalsky

Se uno stato S è possibile e se le precondizioni *pact* di un'azione A sono soddisfatte in quello stato, allora anche lo stato prodotto do(A,S) è possibile:

poss(S) and pact(A,S) → poss(do(A,S))

Quelli che nella formulazione di Green sono predicati qui diventano termini.

In pratica, guadagniamo i vantaggi di una formulazione del II ordine mantenendoci in un sistema del I ordine.

Formulazione di Kowalsky

In questo modo abbiamo bisogno di una singola *frame assertion* per ogni azione (vantaggio rispetto a Green)

Nell'esempio precedente l'unica *frame assertion* che deve essere esplicitata è:

$holds(V,S) \wedge diff(V,clear(Z)) \wedge diff(V,on(X,Y)) \rightarrow holds(V,do(move(X,Y,Z),S)))$

che afferma che tutti i termini differenti da *clear(Z)* e *on(X,Y)* valgono ancora in tutti gli stati prodotti dall'esecuzione di *move*.

Esempio

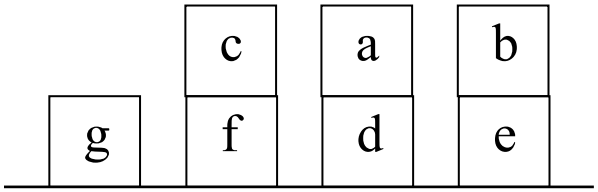
Goal

:- *poss(S), holds(on(a,b),S), holds(on(b,g),S)*.

Usiamo PROLOG per risolverlo

Stato iniziale

poss(s0).
holds(on(a,d),s0).
holds(on(b,e),s0).
holds(on(c,f),s0).
holds(clear(a),s0).
holds(clear(b),s0).
holds(clear(c),s0).
holds(clear(g),s0).



Esempio

Effetti dell'azione move(X,Y,Z):

holds(clear(Y),do(move(X,Y,Z),S)).

holds(on(X,Z),do(move(X,Y,Z),S)).

Clausola che esprime le precondizioni dell'azione move(X,Y,Z):

pact(move(X,Y,Z),S):-

holds(clear(X),S), holds(clear(Z),S), holds(on(X,Y),S), X\=Z.

Clausola per esprimere le condizioni di frame:

holds(V,do(move(X,Y,Z),S)):- holds(V,S), V\=clear(Z), V\=on(X,Y).

Clausola per esprimere la raggiungibilità di uno stato:

poss(do(U,S)):- poss(S), pact(U,S).

Soluzione

:- *poss(S), holds(on(b,g),S), holds(on(a,b),S)*.

Yes per *S* = *do(move(a, d, b), do(move(b, e, g), s0))*

Esercizi proposti

- *Per il problema del mondo a blocchi, nella formulazione di Green trovare, tramite risoluzione, la soluzione al goal:*

:-on(b,g,S), on(a,b,S).

- *Per il problema del mondo a blocchi, nella formulazione di Kowalski, trovare tramite risoluzione, la soluzione al goal:*

:-poss(S),holds(on(b,g),S),holds(on(a,b),S).

Esame - date

- *15 Giugno 2015*
- *14 Luglio 2015*
- *Mattino, h. 10.30 prova CLP in Lab. Informatica grande*
- *Pomeriggio h. 14 parte scritta in aula (apprendimento, pianificazione, reti bayesiane)*