

# PLANNING

---

La pianificazione automatica (**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".

# 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:

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

# 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 una macchina (**stato iniziale**).

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

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

Pianificare ci permette di fare le azioni giuste nella sequenza giusta: 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). Invertendo l'ordine delle azioni 1 e 2 si ottiene un piano corretto analogo al precedente.

Ci possono essere piani alternativi per raggiungere lo stesso obiettivo (posso pensare di andare in facoltà a piedi o in autobus).

Es di piano alternativo

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.

# 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.

# 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.

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

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.

# 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.

# 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. 8



# Esempio: mondo dei blocchi

---

**Problema:** spostare blocchi su un tavolo con un braccio

**Azioni:**

**STACK(X,Y)**

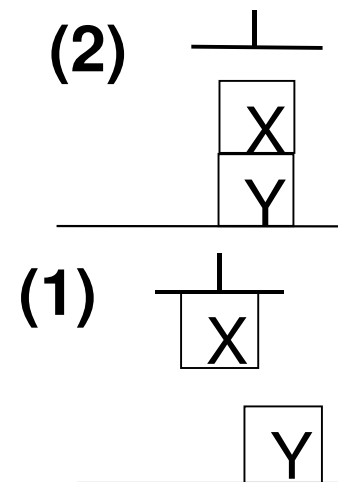
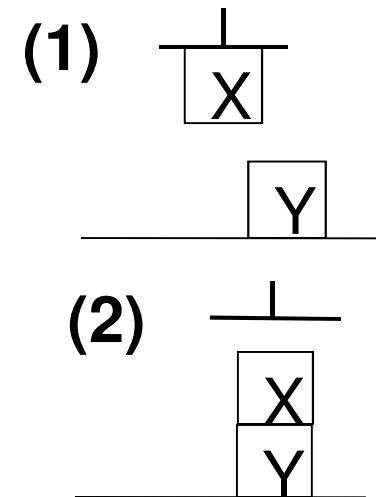
SE: holding(X) and clear(Y)

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);

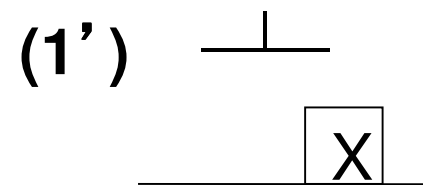


# Esempio: mondo dei blocchi

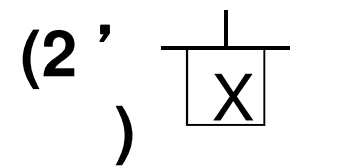
---

## PICKUP(X)

SE:  $\text{ontable}(X)$  and  $\text{clear}(X)$  and  $\text{handempty}$

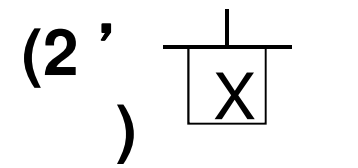


ALLORA:  $\text{holding}(X)$ ;

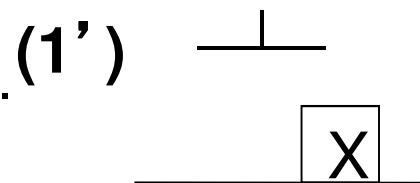


## PUTDOWN(X)

SE:  $\text{holding}(X)$



ALLORA:  $\text{ontable}(X)$  and  $\text{clear}(X)$  and  $\text{handempty}$ .



# 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 imprevedibile. 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

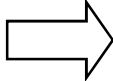
---

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
- spesso alterna il processo di pianificazione a quello di esecuzione reagendo ai cambiamenti di stato

# Cosa vedremo (Fondamenti di AI):

---

- Tecniche di Pianificazione Classica
    - Planning Deduttivo
      - Situation Calculus
    - Planning mediante ricerca
      - Ricerca nello spazio degli stati
        - STRIPS
- Planning Lineare
- 
- Il seguito (planning non lineare/reattivo/condizionale/basato su grafi) nel Corso di Sistemi Intelligenti.

# 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

Formulazioni di Green e Kowalsky



# Situation Calculus (*fine anni '60*)

---

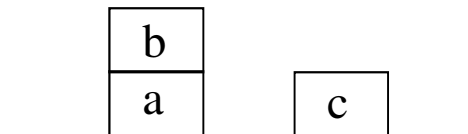
Formalizzazione del linguaggio (basato sulla logica dei predicati del primo ordine) in grado di rappresentare stati e azioni in funzione del tempo

➤ **Situation:** “fotografia” del mondo e delle proprietà (*fluent*) che valgono in un determinato istante/stato  $s$

– Esempio: mondo dei blocchi

$on(b,a,s)$

$ontable(c,s)$



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

$on(X,Y,S)$  and  $clear(X,S) \rightarrow$

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

$(clear(Y,do(putOnTable(X),S)))$

# Situation Calculus

---

- **Costruzione di un piano:** deduzione, dimostrazione di un goal
  - Esempio
    - $\text{:- ontable}(b,S)$ . Significa: esiste uno stato  $S$  in cui e' vero  $\text{ontable}(b)$
    - YES per  $S=\text{putOntable}(b,s)$
- **Vantaggi:** elevata espressività, permette di descrivere problemi complessi
- **Problema:** frame problem

# 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 prova 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

# Esempio

---

- I seguenti assiomi descrivono tutte le relazioni vere nello stato iniziale  $s_0$

A.1  $on(a,d,s_0)$ .

A.2  $on(b,e,s_0)$ .

A.3  $on(c,f,s_0)$ .

A.4  $clear(a,s_0)$ .

A.5  $clear(b,s_0)$ .

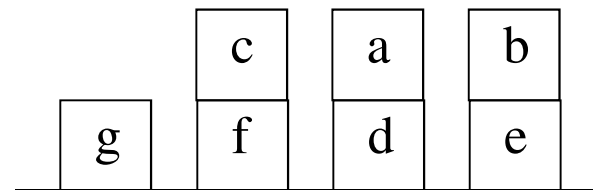
A.6  $clear(c,s_0)$ .

A.7  $clear(g,s_0)$ .

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*)

A.11  $\sim clear(X, S)$  or  $\sim clear(Z, S)$  or  $\sim on(X, Y, S)$  or  $\sim diff(X, Z)$  or  $clear(Y, do(move(X, Y, Z), S))$ .

A.12  $\sim clear(X, S)$  or  $\sim clear(Z, S)$  or  $\sim on(X, Y, S)$  or  $\sim diff(X, Z)$  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)*

*{S/s0},*

*(A.5)*

*{S/s0},*

*(A.1)*

*{S/s0, Y/d}*

*(A.8)*

*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)*.

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)).*

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

# Frame problem

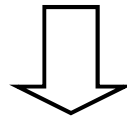
---

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:

$on(U, V, S) \text{ and } diff(U, X) \rightarrow on(U, V, do(move(X, Y, Z), S))$

$clear(U, S) \text{ and } diff(U, Z) \rightarrow clear(U, do(move(X, Y, Z), S))$

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



Se il problema è complicato la complessità diventa inaccettabile



# Pianificazione come ricerca

---

- *Quello che cambia è lo spazio di ricerca, definito da che cosa sono gli stati e gli operatori:*
  - *Pianificazione deduttiva 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*
  - *Pianificazione nello spazio dei piani: stati come piani parziali e operatori di raffinamento e completamento di piani (non la vedremo in questo corso)*

# Planning classico non deduttivo

---

La pianificazione classica non deduttiva

- utilizza linguaggi specializzati per rappresentare stati, goal e azioni
- gestisce la generazione del piano come un problema di ricerca (*search*).

La ricerca può essere effettuata:

- nello **spazio degli stati** o situazioni  
*Nell'albero di ricerca ogni nodo rappresenta uno stato e ogni arco un'azione.*
- nello **spazio dei piani**  
*Nell'albero di ricerca ogni nodo rappresenta un piano parziale e ogni arco un'operazione di raffinamento del piano*

# Ricerca nello spazio degli stati: Planning Lineare

---

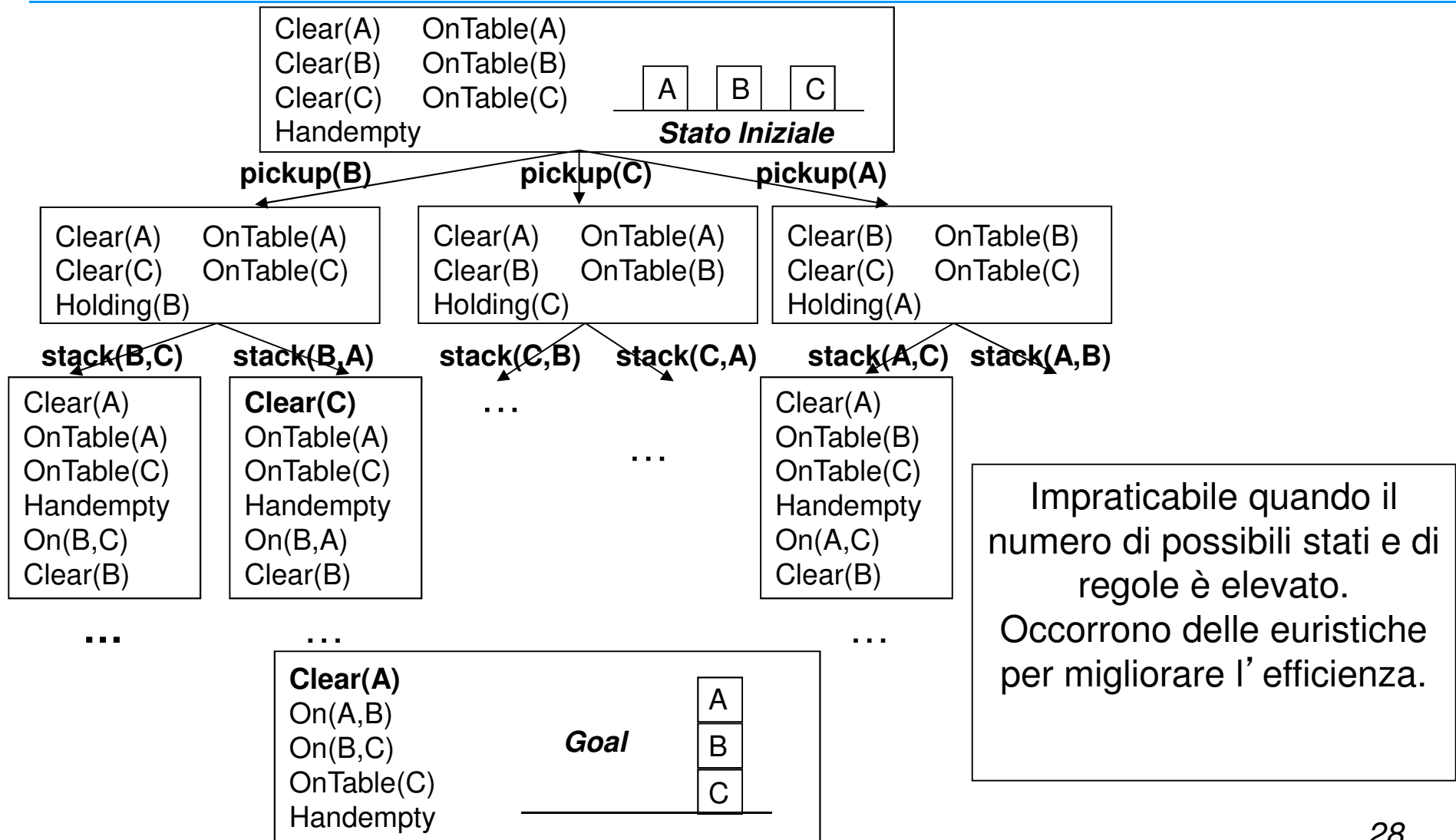
Un **pianificatore lineare** riformula il problema di pianificazione come problema di ricerca nello spazio degli stati e utilizza le strategie di ricerca classiche:

L' algoritmo di ricerca può essere:

- **Forward**: se la ricerca avviene in modo progressivo partendo dallo stato iniziale fino al raggiungimento di uno stato che soddisfa il goal.
- **Backward**: quando la ricerca è attuata in modo regressivo a partire dal goal fino a *ridurre* il goal in sottogoal soddisfatti dallo stato iniziale.



# Ricerca Forward: un esempio



# STRIPS

---

**STRIPS** - Stanford Research Institute Problem Solver

Antenato degli attuali sistemi di pianificazione. (primi anni '70)

- Linguaggio per la rappresentazione di azioni. Sintassi molto più semplice del Situation Calculus (meno espressività, più efficienza).
- Algoritmo per la costruzione di piani.

# Linguaggio Strips

---

- Rappresentazione dello stato
  - Insieme di fluent che valgono nello stato  
Esempio: *on(b,a)*, *clear(b)*, *clear(c)*, *ontable(c)*
- Rappresentazione del goal
  - Insieme di fluent (simile allo stato)
  - Si possono avere variabili  
Esempio: *on(X,a)*

# Linguaggio Strips

---

- Rappresentazione delle azioni (3 liste)
  - PRECONDIZIONI: fluent che devono essere veri per applicare l'azione
  - DELETE List: fluent che diventano falsi come risultato dell'azione
  - ADD List: fluent che diventano veri come risultato dell'azione

Esempio  $Move(X, Y, Z)$

Precondizioni:  $on(X, Y), clear(X), clear(Z)$

Delete List:  $clear(Z), on(X, Y)$

Add list:  $clear(Y), on(X, Z)$

A volte ADD e DELETE list sono rappresentate come **EFFECT** list con atomi positivi e negativi

Esempio  $Move(X, Y, Z)$

Precondizioni:  $on(X, Y), clear(X), clear(Z)$

Effect List:  $\neg clear(Z), \neg on(X, Y), clear(Y), on(X, Z)$

Frame problem risolto con la **Strips Assumption**:

*tutto ciò che non è specificato nella ADD e DELETE list resta immutato*

# *AZIONI in STRIPS (1)*

---

pickup(X)

PRECOND: ontable(X), clear(X), handempty

DELETE: ontable(X), clear(X), handempty

ADD: holding(X)

putdown(X)

PRECOND: holding(X)

DELETE: holding(X)

ADD: ontable(X), clear(X), handempty



## *AZIONI IN STRIPS (2)*

---

stack(X,Y)

PRECOND: holding(X), clear(Y)

DELETE: holding(X), clear(Y)

ADD: handempty, on(X,Y), clear(X)

unstack(X,Y)

PRECOND: handempty, on(X,Y), clear(X)

DELETE: handempty, on(X,Y), clear(X)

ADD: holding(X), clear(Y)

# Algoritmo STRIPS

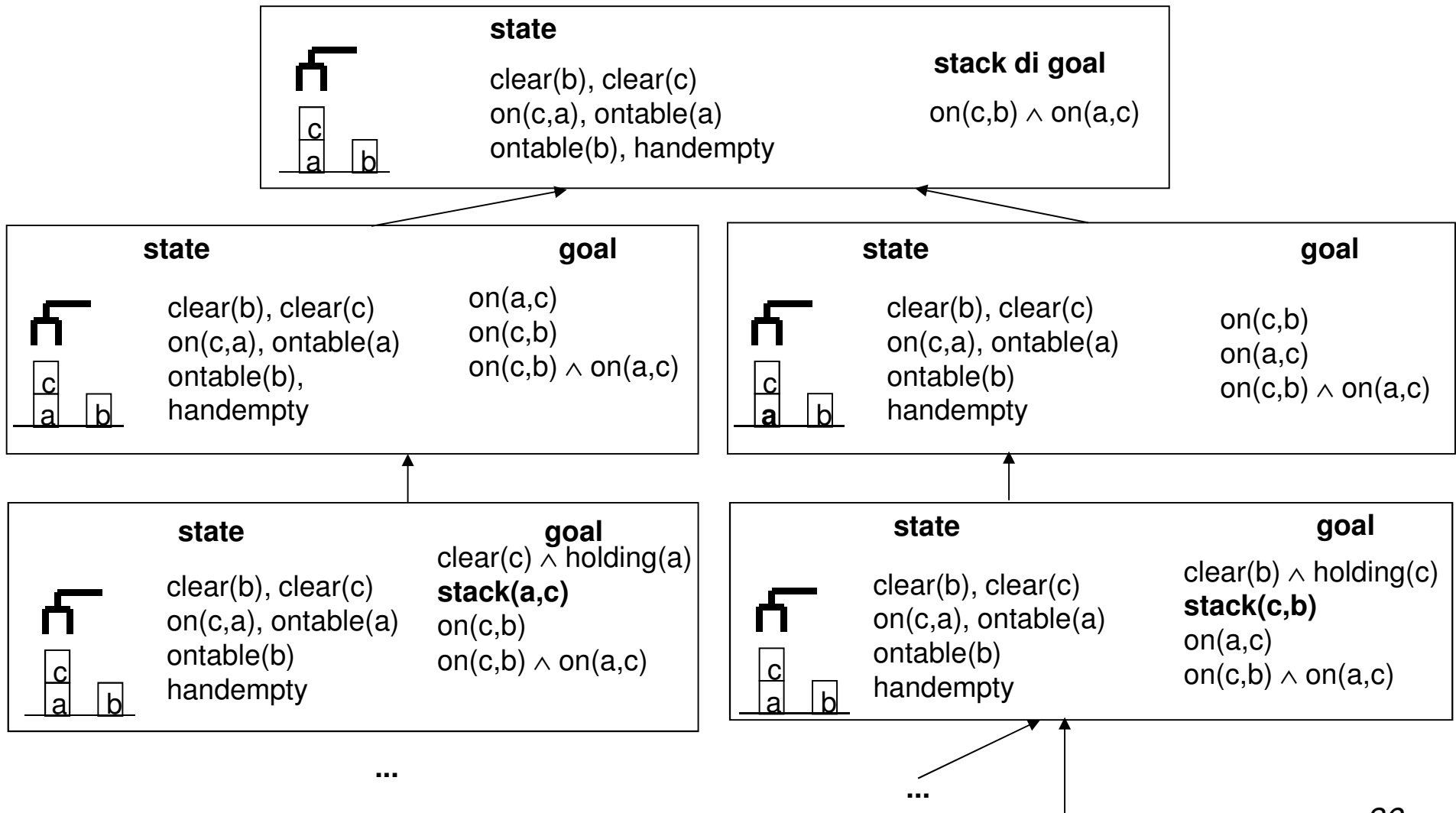
- Planner lineare basato su ricerca backward
- Assume che lo stato iniziale sia completamente noto (**Closed World Assumption**)
- Utilizza due strutture dati
  - stack di goal
  - descrizione S dello stato corrente
- Algoritmo
  - Inizializza stack con la congiunzione di goal finali
  - while stack non è vuoto do
    - if top(stack) = A and  $A \theta \subseteq S$  (si noti che A puo essere un **and** di goals o un atomo)
    - then pop(a) ed esegui sost  $\theta$  sullo stack
    - else if top(stack) = a
      - then
        - seleziona regola R con  $a \in \text{Addlist}(R)$ ,
        - pop(a), push(R), push(Precond(R));
      - else if top(stack) =  $a_1 \wedge a_2 \wedge \dots \wedge a_n$ 
        - (\*) then push( $a_1$ ), ..., push( $a_n$ )
        - else if top(stack) = R
    - then pop(R) e applica R trasformando S
- (\*) si noti che l'ordine con cui i sottogoal vengono inseriti nello stack rappresenta un punto di scelta non deterministica. La congiunzione rimane sullo stack e verrà riverificata dopo - interacting goals)

# Considerazioni sull' algoritmo

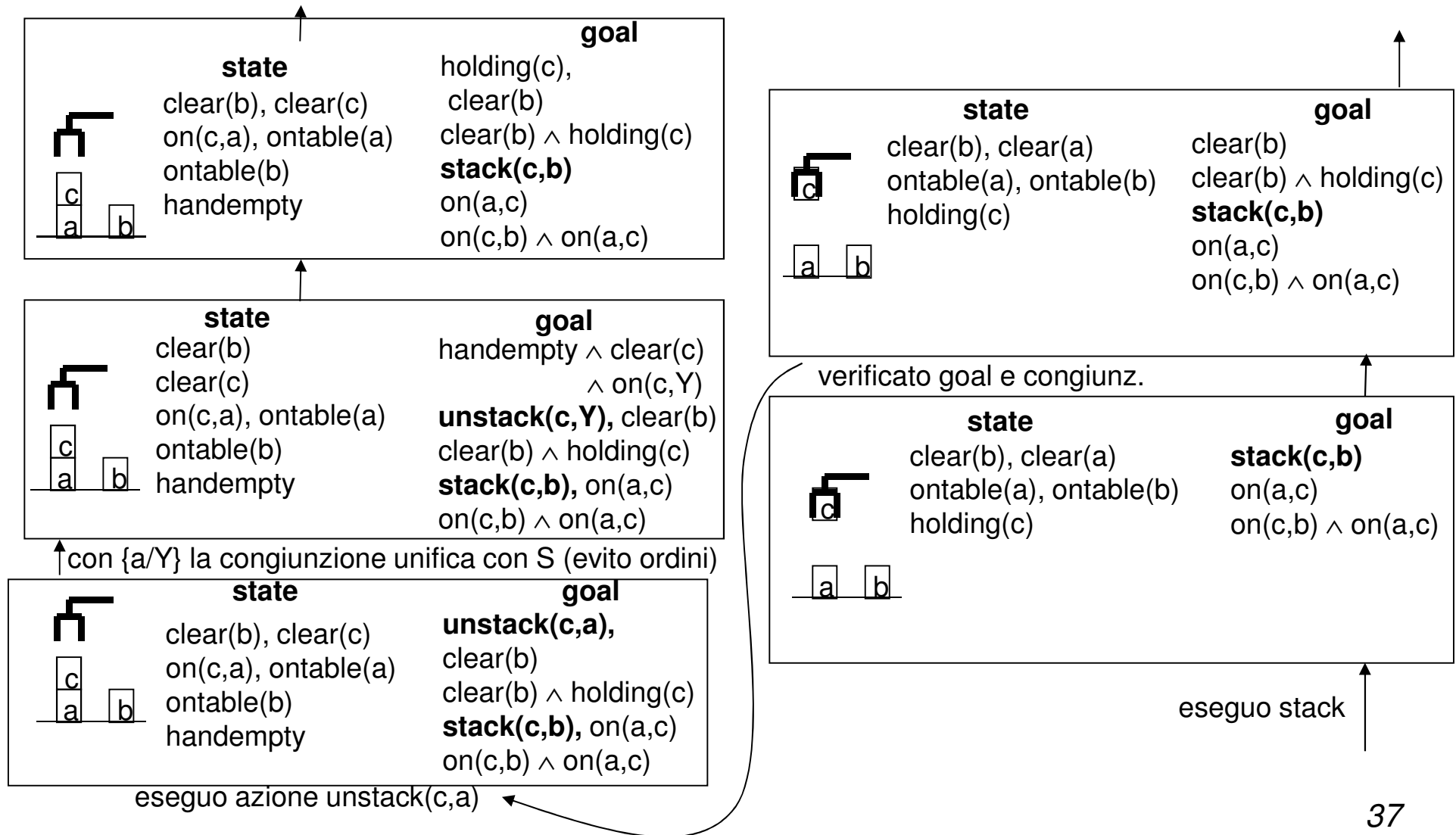
---

1. Il goal è la prima pila di obiettivi.
2. Suddividere il problema in sottoproblemi: ciascuno per un componente dell'obiettivo originale. Tali sottoproblemi possono interagire
3. Abbiamo tanti possibili ordini di soluzione.
4. Ad ogni passo del processo di risoluzione si cerca di risolvere il goal in cima alla pila.
5. Quando si ottiene una sequenza di operatori che lo soddisfa la si applica alla descrizione corrente dello stato ottenendo una nuova descrizione.
6. Si cerca poi di soddisfare l'obiettivo che è in cima alla pila partendo dalla situazione prodotta dal soddisfacimento del primo obiettivo.
7. Il procedimento continua fino allo svuotamento della pila.
8. Quando in cima alla pila si incontra una congiunzione si verifica che tutte le sue componenti siano effettivamente soddisfatte nello stato attuale. Se una componente non è soddisfatta (problema dell' interazione tra goal è spiegato più avanti) si reinserisce nella pila e si continua.

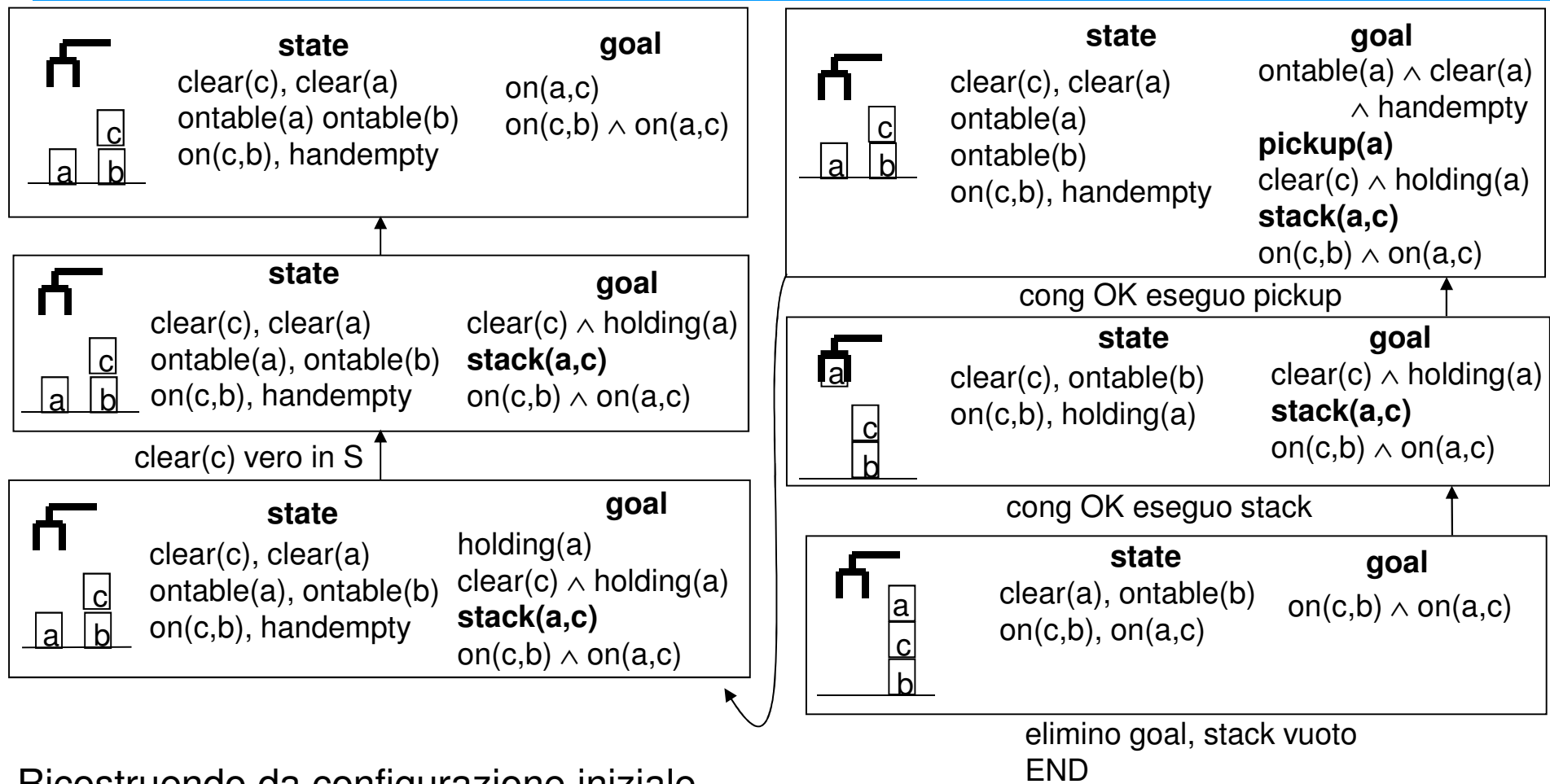
# Esempio



# Esempio



# Esempio



Ricostruendo da configurazione iniziale a finale ho una soluzione:

1. unstack(c,a)

2. stack(c,b)

3. pickup(a)

4. stack(a,c)

38

# Problemi (1)

---

**1. Grafo di ricerca molto vasto.** Nell' esempio abbiamo visto un cammino ma in realtà ci sono varie alternative

- scelte non deterministiche ordinamento dei goal
- più operatori applicabili per ridurre un goal

**Soluzione:** Strategie euristiche

- strategie di ricerca
- strategie per scegliere quale goal ridurre e quale operatore
  - **MEANS-ENDS ANALYSIS**
    - cercare la differenza più significativa tra stato e goal
    - ridurre quella differenza per prima

# Problemi (2)

---

**2. Problema dell'interazione tra goal.** Quando due (o più) goal inter-agiscono ci possono essere problemi di interazione tra le due soluzioni.

Goal G1, G2

- pianifico azioni per G2
  - poi per risolvere G1 potrei dover smontare tutto, compreso lo stato che avevo prodotto con G2 risolto
- Soluzione completa:
- provare tutti gli ordinamenti dei goal e dei loro sottogoal.
- Soluzione pratica (Strips):
- provare a risolverli indipendentemente
  - verificare che la soluzione funzioni
  - se non funziona, provare gli ordinamenti possibili uno alla volta

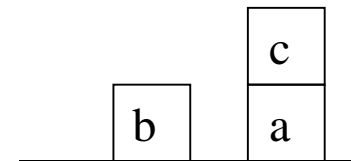


# Esempio: Anomalia di Sussmann

---

Stato iniziale (come esempio precedente):

*clear(b),*  
*clear(c),*  
*on(c,a),*  
*ontable(a),*  
*ontable(b),*  
*handempty*

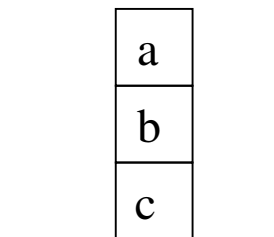


Goal:  $on(a,b), on(b,c)$

Possibili stack iniziali:

(1)  
 $on(a,b)$   
 $on(b,c)$   
 $on(a,b) \wedge on(b,c)$

(2)  
 $on(b,c)$   
 $on(a,b)$   
 $on(a,b) \wedge on(b,c)$



Decidiamo di scegliere (1)

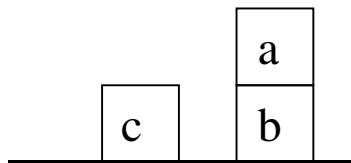
# Esempio: Anomalia di Sussmann

---

Applicando il procedimento di soluzione di STRIPS otteniamo:

1. unstack(c,a)
2. putdown(c)
3. pickup(a)
4. stack(a,b)

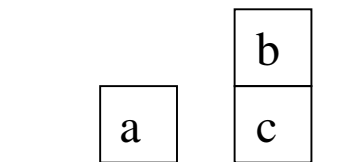
Stato attuale:



Adesso lavoriamo al soddisfacimento del secondo goal  $on(b,c)$ .

5. unstack(a,b)
6. putdown(a)
7. pickup(b)
8. stack(b,c)

Stato attuale:



# Esempio: Anomalia di Sussmann

---

La congiunzione  $\text{on}(a,b) \wedge \text{on}(b,c)$  non è valida.

Allora reinseriamo  $\text{on}(a,b)$  nello stack di goals (è la differenza rispetto allo stato attuale).

Otteniamo:

9. pickup(a)

10. stack(a,b)

Abbiamo ottenuto quello che volevamo, ma in modo scarsamente efficiente.

# Problemi (3)

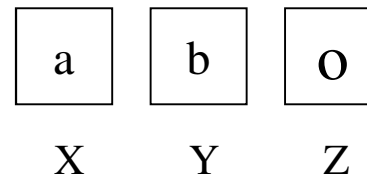
---

## 3. Perdita di informazioni:

Supponiamo di voler utilizzare STRIPS per generare un piano che attui lo scambio di valore fra due celle di memoria o registri X e Y.

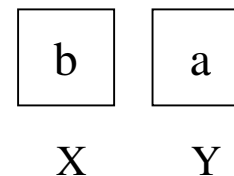
Stato iniziale:

cont(x,a) and cont(y,b) and cont(z,o)



Stato finale:

cont(x,b) and cont(y,a)



Operazione di assegnamento:

ASSIGN(X,Y,T,S)

Precondition: cont(Y,S) and cont(X,T)

Delete: cont(X,T)

Add: cont(X,S)

# Problemi (3)

---

Goal stack:

cont(x,b),  
cont(y,a),  
cont(x,b)  $\wedge$  cont(y,a)

Per ridurre il goal cont(x,b) applica ASSIGN con Y/y e T/a

cont(y,b)  $\wedge$  cont(x,a), **assign(x,y,a,b)**,

Dopo l'esecuzione lo stato diventa:

	<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;">b</div>	<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;">b</div>	<div style="border: 1px solid black; display: inline-block; width: 40px; height: 30px;"></div>
	X	Y	Z

cont(x,b), cont(y,b), cont(z,o)

Il goal cont(y,a) rimasto nello stack non potrà mai essere soddisfatto.

Proprietà distruttiva dello statement di assegnamento!

STRIPS è troppo "ansioso" di risolvere l'obiettivo e quindi distrugge inesorabilmente il contenuto di una cella di memoria!

# Soluzione: Pianificazione Non Lineare

---

- I pianificatori non lineari sono algoritmi di ricerca che gestiscono la generazione di un piano come un problema di ricerca nello spazio dei piani e non più degli stati.
- L' algoritmo non genera più il piano come una successione lineare (completamente ordinata) di azioni per raggiungere i vari obiettivi.
  - Si vedrà nel Corso di Sistemi Intelligenti.
  - POP (partial Order Planning)

# Planning in pratica

---

- Molte applicazioni in domini complessi:
  - Pianificatori per Internet (ricerca di informazioni, sintesi di comandi Unix)  
<http://www.cs.washington.edu/research/projects/softbots/www/softbots.html>
  - Gestione di strumentazione spaziale  
<http://rax.arc.nasa.gov/>
  - Robotica  
<http://www.robocup.org/>
  - Graphplan sviluppato dalla Carnegie Mellon University  
<http://www.cs.cmu.edu/~avrim/graphplan.html>
  - Piani di produzione industriale
  - Logistica
- Algoritmi visti presentano problemi di efficienza in caso di domini con molti operatori
- Tecniche per rendere più efficiente il processo di pianificazione

↓  
**PLANNING GERARCHICO**

# Esecuzione

---

I pianificatori visti finora permettono di costruire piani che vengono poi eseguiti da un agente “**esecutore**”.

Possibili problemi in esecuzione:

- Esecuzione di un'azione in condizioni diverse da quelle previste dalle sue precondizioni
  - conoscenza incompleta o non corretta
  - condizioni inaspettate
  - trasformazioni del mondo per cause esterne al piano
- Effetti delle azioni diversi da quelli previsti
  - errori dell'esecutore
  - effetti non deterministici, imprevedibili

Occorre che l'esecutore sia in grado di *percepire* i cambiamenti e *agire* di conseguenza



# Esecuzione

---

Alcuni pianificatori fanno l'ipotesi di *mondo aperto* (**Open World Assumption**) ossia considerano l'informazione non presente nella rappresentazione dello stato come *non nota* e *non falsa* diversamente dai pianificatori che lavorano nell'ipotesi di mondo chiuso (*Closed World Assumption*)

Alcune informazioni non note possono essere cercate tramite azioni di “raccolta di informazioni” (**azioni di sensing**) aggiunte al piano.

Le **azioni di sensing** sono modellate come le azioni causali.

Le precondizioni rappresentano le condizioni che devono essere vere affinché una certa osservazione possa essere effettuata, le postcondizioni rappresentano il risultato dell'osservazione.

Due possibili approcci:

- Planning Condizionale
- Integrazione fra Pianificazione ed Esecuzione

# Planning Reattivo (Brooks - 1986)

---

Abbiamo descritto fino qui un processo di pianificazione deliberativo nel quale prima di eseguire una qualunque azione viene costruito l'intero piano.

I pianificatori **reattivi** sono algoritmi di pianificazione on-line, capaci di interagire con il sistema in modo da affrontare il problema della dinamicità e del non determinismo dell'ambiente:

- osservano il mondo in fase di pianificazione per l'acquisizione di informazione non nota
- monitorano l'esecuzione delle azioni e ne verificano gli effetti
- spesso alternano il processo di pianificazione a quello di esecuzione reagendo ai cambiamenti di stato

Discendono dai *sistemi reattivi "puri"* che evitano del tutto la pianificazione ed utilizzano semplicemente la situazione osservabile come uno stimolo per reagire.

# Sistemi reattivi puri

---

Hanno accesso ad una base di conoscenza che descrive quali azioni devono essere eseguite ed in quali circostanze. Scelgono le azioni una alla volta, senza anticipare e selezionare un'intera sequenza di azioni prima di cominciare.

Esempio -Termostato utilizza le semplici regole:

- 1) Se la temperatura  $T$  della stanza è  $K$  gradi sopra la soglia  $T_0$ , accendi il condizionatore;
- 2) Se la temperatura della stanza è  $K$  gradi sotto  $T_0$ , spegni il condizionatore.

## **Vantaggi:**

- Sono capaci di interagire con il sistema reale. Essi operano in modo robusto in domini per i quali è difficile fornire modelli completi ed accurati.
- Non usano modelli, ma solo l'immediata percezione del mondo e per questo sono anche estremamente veloci nella risposta.

## **Svantaggio:**

Il loro comportamento in domini che richiedono di ragionare e deliberare in modo significativo è deludente (es. scacchi) in quanto non sono in grado di generare autonomamente piani.

# Pianificatori ibridi

---

I moderni pianificatori reattivi detti **ibridi** integrano **approccio generativo** e **approccio reattivo** al fine di sfruttare le capacità computazionali del primo e la capacità di interagire con il sistema del secondo affrontando così il problema dell' esecuzione.

Un pianificatore **ibrido**:

- genera un piano per raggiungere il goal
- verifica le precondizioni dell' azione che sta per eseguire e gli effetti dell' azione appena eseguita
- smonta gli effetti di un' azione (importanza della reversibilità delle azioni) e ripianifica in caso di fallimenti
- corregge i piani se avvengono azioni esterne impreviste.