



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Planning

Federico Chesani

DISI

Department of Informatics – Science and Engineering

Disclaimer & Further Reading

- These slides are largely based on previous work by Prof. Paola Mello



Planning

La pianificazione automatica (**planning**) è una tipologia specifica di problem solving

- 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”.

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



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 sè
- Un'attività comune a molte applicazioni quali
 - diagnosi: pianificazione di test o azioni per riparare (riconfigurare) un sistema
 - Scheduling
 - Robotica



Esempio di pianificazione

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 di pianificazione (cont.)

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 dell'esempio precedente si ottiene comunque un piano corretto.

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.



(Alcuni) Concetti base

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

- a) una rappresentazione dello stato iniziale
- b) una rappresentazione del goal
- c) una descrizione formale delle azioni eseguibili (cioè della loro applicabilità e dei loro effetti)

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



Rappresentazione dello stato

Nota: è necessario 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.

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

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 tal 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.

A volte 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

Problema: spostare blocchi su un tavolo con un braccio

Azioni:

STACK(X,Y)

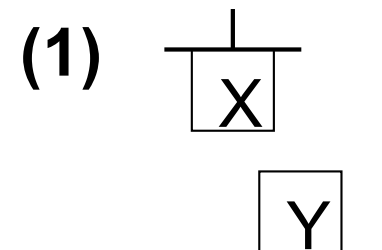
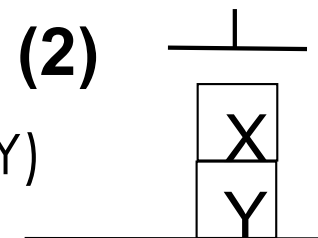
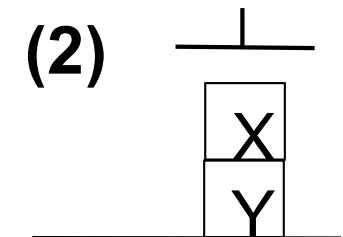
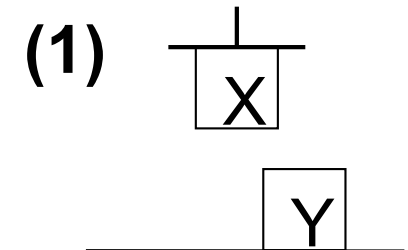
SE: holding(X) and clear(Y)

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);

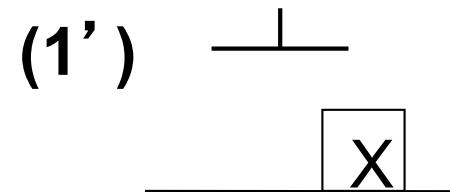


Esempio: mondo a blocchi

PICKUP(X)

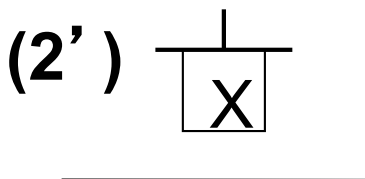
SE: ontable(X) and clear(X) and handempty

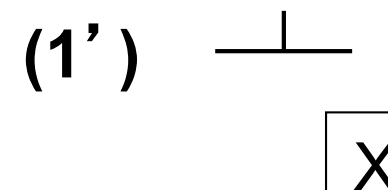
ALLORA: holding(X);



PUTDOWN(X)

SE: holding(X)

ALLORA: ontable(X) and clear(X) and handempty. (2') 



Alcune caratteristiche del processo di pianificazione

Il processo per il calcolo della soluzione può essere:

- **non de-componibile** (spesso!!!): ci può essere interazione fra i sottoproblemi
- **Reversibile**: le scelte fatte durante la generazione del piano sono revocabili solo in fase di pianificazione (backtracking) **ma non in esecuzione**.
- 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

Il processo di applicazione della procedura di soluzione può essere:

- **Irreversibile**: l'esecuzione delle azioni determina spesso un cambiamento di stato non reversibile;
- **non deterministico**: 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

E' un 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



Qualche tecnica...

Tecniche di Pianificazione Classica:

- Planning Deduttivo
 - Situation Calculus
- Planning mediante ricerca
 - Ricerca nello spazio degli stati, ad es. Planning Lineare
 - STRIPS
- Planning non lineare/reattivo/condizionale/basato su grafi: argomento del Corso di Intelligent Systems.



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

Due formulazioni famose:

- Green
- Kowalsky

Accenneremo a quella di Green, basata su Logica del primo ordine (FOL).

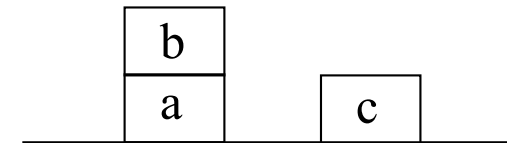


Situation Calculus

Formalizzazione del linguaggio (basato sulla logica dei predicati del primo ordine) in grado di rappresentare stati e azioni in funzione del tempo. Introdotto da John McCarthy nel 1963.

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

$on(b,a,s_0).$ $clear(b,s_0)$
 $ontable(c,s_0).$ $clear(c,s_0)$
 $ontable(a,s_0).$



- **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 è vero $\text{ontable}(b)$?

YES per $S = \text{do}(\text{putOnTable}(b), s_0)$

- 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 dello 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 1969)

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.

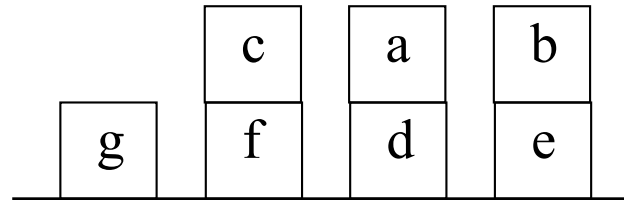


Esempio

➤ **Assiomi che descrivono tutte le relazioni vere nello stato iniziale s0 (fluents in rosso):**

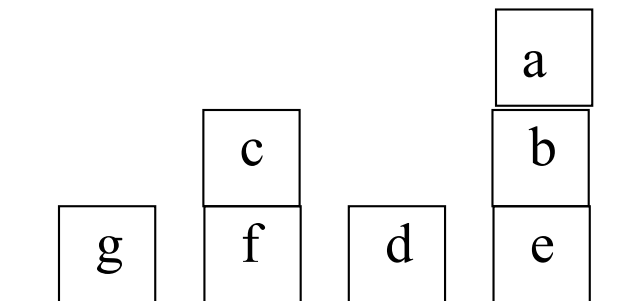
- A.1 *on(a,d,s0).*
- A.2 *on(b,e,s0).*
- A.3 *on(c,f,s0).*
- A.4 *clear(a,s0).*
- A.5 *clear(b,s0).*
- A.6 *clear(c,s0).*
- A.7 *clear(g,s0).*
- A.8 *diff(a,b)*
- A.9 *diff(a,c)*
- A.10 *diff(a,d)...*

Stato s0



Azione di move(a,d,b) nello stato s0

nuovo stato S1 = do(move(a,d,b),s0).



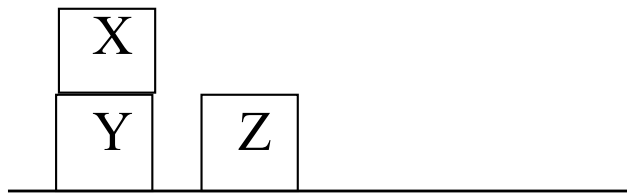
on(a,b, do(move(a,d,b),s0)).
on(b,e, do(move(a,d,b),s0)).
on(c,f, do(move(a,d,b),s0)).
clear(a, do(move(a,d,b),s0)).
clear(c, do(move(a,d,b),s0)).
clear(g, do(move(a,d,b),s0)).
clear(d, do(move(a,d,b),s0)).

.....



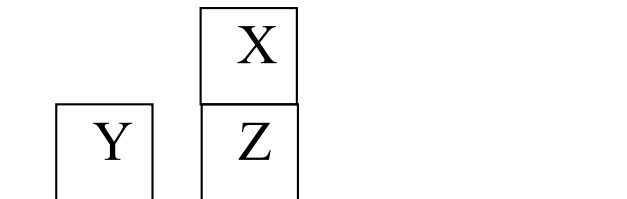
Esempio azione di move(X,Y,Z)

Stato s0



Azione di move(X,Y,Z) nello stato s0

nuovo stato S1



Esempio

Le azioni si esprimono con assiomi nella forma a clausole

Ad esempio, l'azione $\text{move}(X,Y,Z)$

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

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

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

A.12 $\sim \text{clear}(X,S) \text{ or } \sim \text{clear}(Z,S) \text{ or } \sim \text{on}(X,Y,S) \text{ or } \sim \text{diff}(X,Z) \text{ or}$
 $\text{on}(X,Z,\text{do}(\text{move}(X,Y,Z),S)).$



Esempio

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

GOAL:- $\text{on}(a,b,S1)$

Goal Negato: $\sim\text{on}(a,b,S1)$

(A.12) $\{X/a, Z/b, S1/\text{do}(\text{move}(a,Y,b),S)\}$ unifica con
 $\text{on}(X,Z,\text{do}(\text{move}(X,Y,Z),S))$.

$\sim\text{clear}(a,S)$	$\text{or } \sim\text{clear}(b,S)$	$\text{or } \sim\text{on}(a,Y,S)$	$\text{or } \sim\text{diff}(a,b)$
(A.4)	(A.5)	(A.1)	(A.8)
$\{S/s0\},$	$\{S/s0\},$	$\{S/s0, Y/d\}$	Contraddizione

$\sim\text{on}(a,b,S1)$ porta a una contraddizione quindi $\text{on}(a,b,S1)$ risulta dimostrato con la sostituzione $S1/\text{do}(\text{move}(a,d,b),s0)$.

Supponiamo di voler risolvere un problema un po' più complesso

Goal: $\text{on}(a,b,S), \text{on}(b,g,S)$.

Soluzione: $S/\text{do}(\text{move}(a,d,b),\text{do}(\text{move}(b,e,g),s0))$.



Frame Problem

Per descrivere un'azione occorre una descrizione completa dello stato risultante dall'esecuzione di ciascuna azione (effect axioms).

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 axiom per ogni relazione NON modificata dall'azione. **Se il problema è complicato la complessità diventa inaccettabile**



Pianificazione classica come ricerca

- Utilizza linguaggi per rappresentare stati, goal e azioni.
- Gestisce la generazione del piano come un problema di ricerca.
- Quello che cambia è lo spazio di ricerca, definito da che cosa sono gli stati e gli operatori:
 - **Pianificazione deduttiva** come dimostrazione di teoremi: 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 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.



STRIPS

STRIPS - Stanford Research Institute Problem Solving
definito nei primi anni '70 e antenato degli attuali
sistemi di pianificazione.

- Linguaggio per la rappresentazione di azioni con sintassi molto più semplice del Situation Calculus (meno espressività, più efficienza).
- Algoritmo specifico 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)`
- Rappresentazione delle azioni/regole (3 liste)
 - **PRECONDIZIONI**: fluent che devono essere veri per applicare l'azione
 - **DELETE**: fluent che diventano falsi come risultato dell'azione
 - **ADD**: fluent che diventano veri come risultato dell'azione



Linguaggio STRIPS: azioni

Esempio di regola/azione

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

Move(X, Y, Z)

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

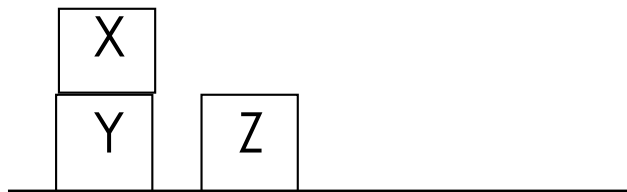
Effect List: \neg clear(Z), \neg on(X,Y), clear(Y), on(X,Z)

Frame problem risolto con la **Strips Assumption**: Tutto ciò che non è specificato nella ADD e DELETE list resta immutato



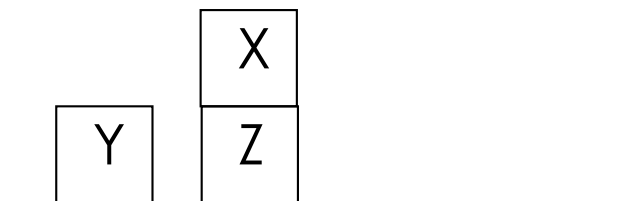
Esempio azione di move(X,Y,Z)

Stato s0



Azione di move(X,Y,Z) nello stato s0

nuovo stato S1



Esempio di AZIONI in STRIPS per il mondo di blocchi

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

stack(X,Y)

PRECOND: holding(X), clear(Y)

DELETE: holding(X), clear(Y)

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

unstack(X,Y)

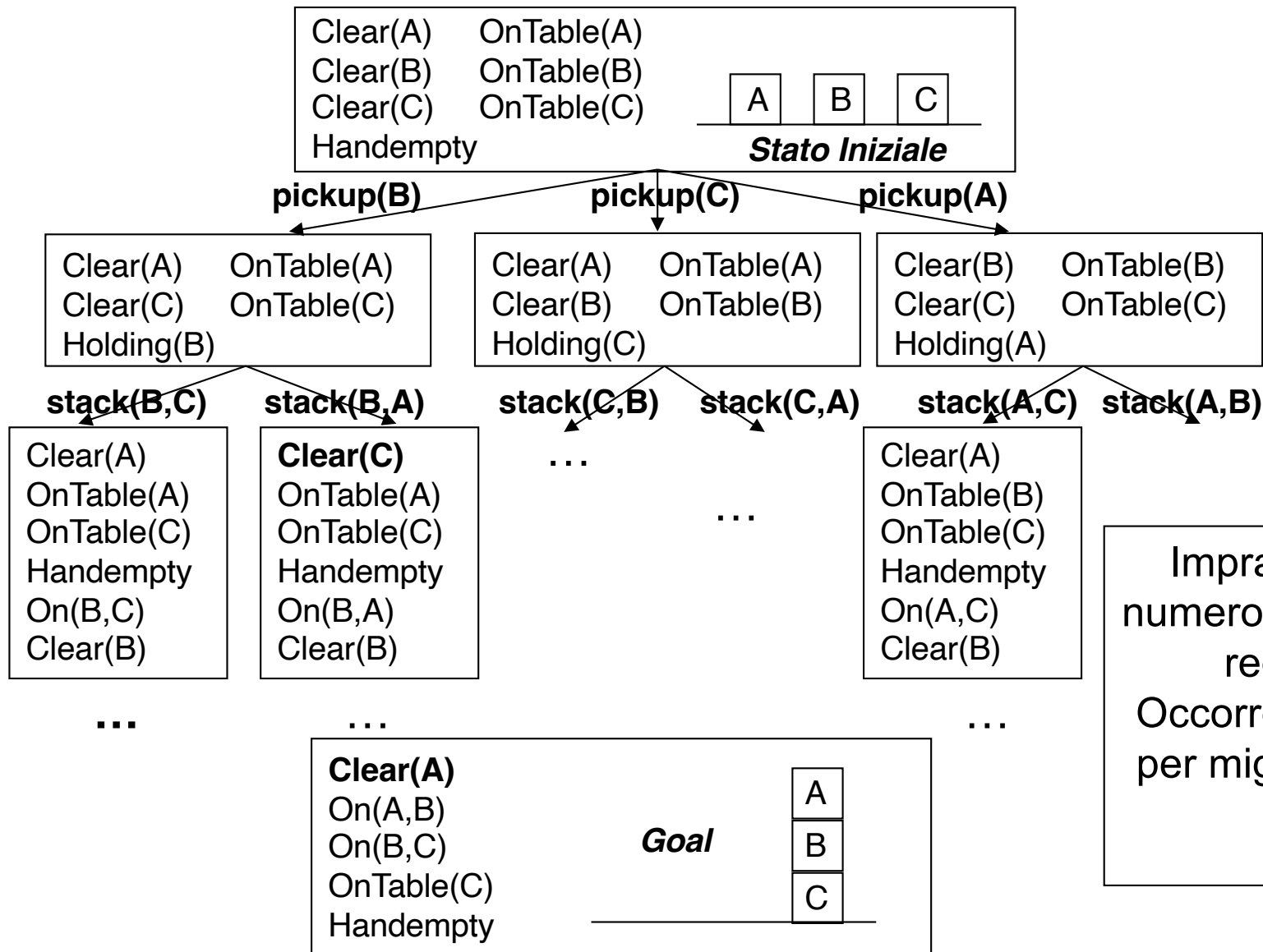
PRECOND: handempty, on(X,Y), clear(X)

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

ADD: holding(X), clear(Y)



Ricerca Forward: un esempio



Impraticabile quando il numero di possibili stati e di regole è elevato. Occorrono delle euristiche per migliorare l'efficienza.



Algoritmo STRIPS

Algoritmo di ricerca nello spazio degli stati.

- Utilizza il linguaggio STRIPS precedentemente presentato con regole precondizione -> azione.
- 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 STRIPS

Algoritmo:

Inizializza stack con la congiunzione di goal finali

while (stack non è vuoto) do

 if $\text{top}(\text{stack}) = A$ and $A \theta \subseteq S$ (si noti che A può essere un and di goals o un atomo)

then pop(A) ed esegui sost θ sullo stack

else if ($\text{top}(\text{stack}) = a$)

then

 – seleziona regola R con $a \in \text{Addlist}(R)$,

 – pop(a), push(R), push($\text{Precond}(R)$);

else if $\text{top}(\text{stack}) = a_1 \wedge a_2 \wedge \dots \wedge a_n$

 (*) then push(a_1), ..., push(a_n)

else if $\text{top}(\text{stack}) = R$

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 verificata dopo - interacting goals

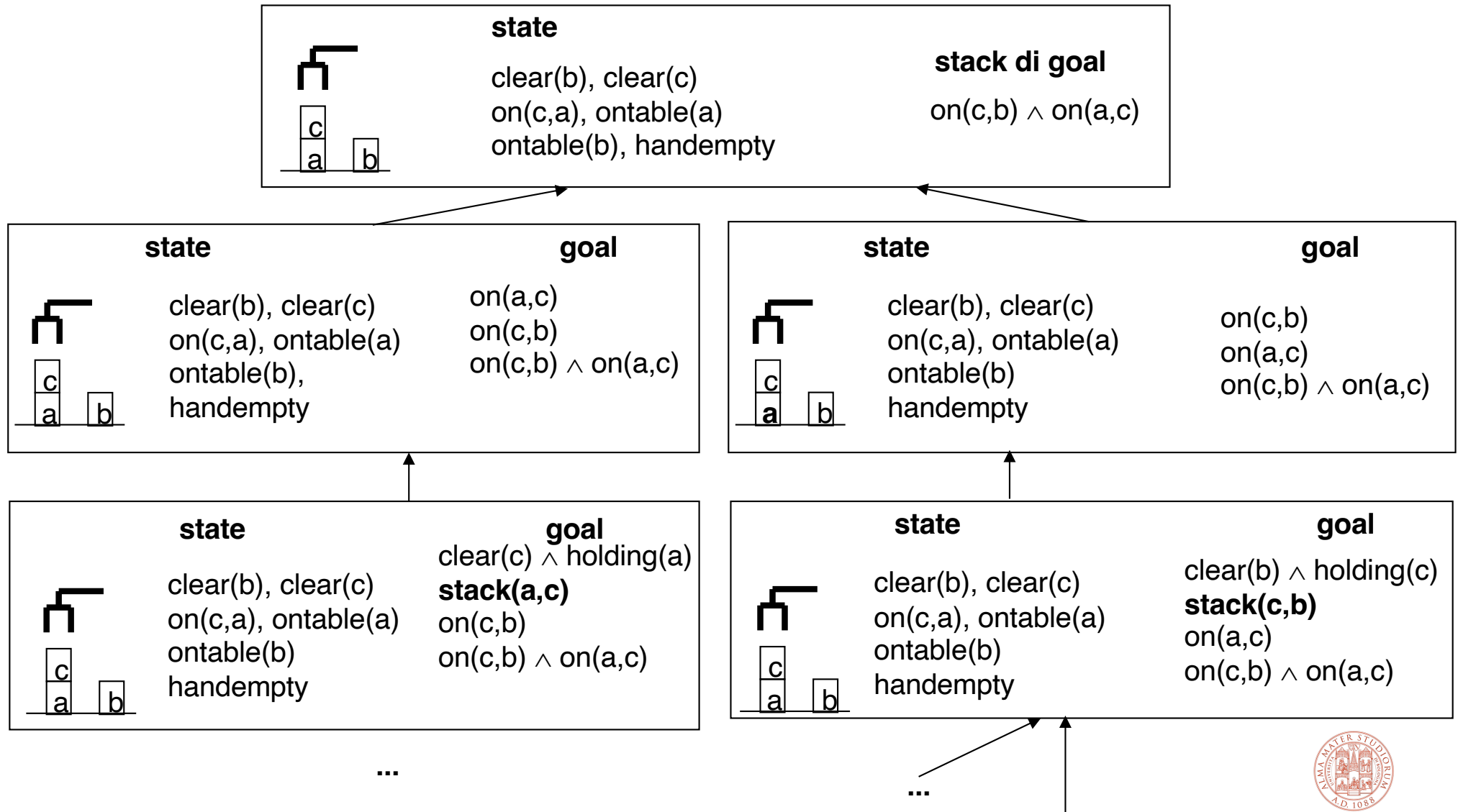


Considerazioni sull'algoritmo

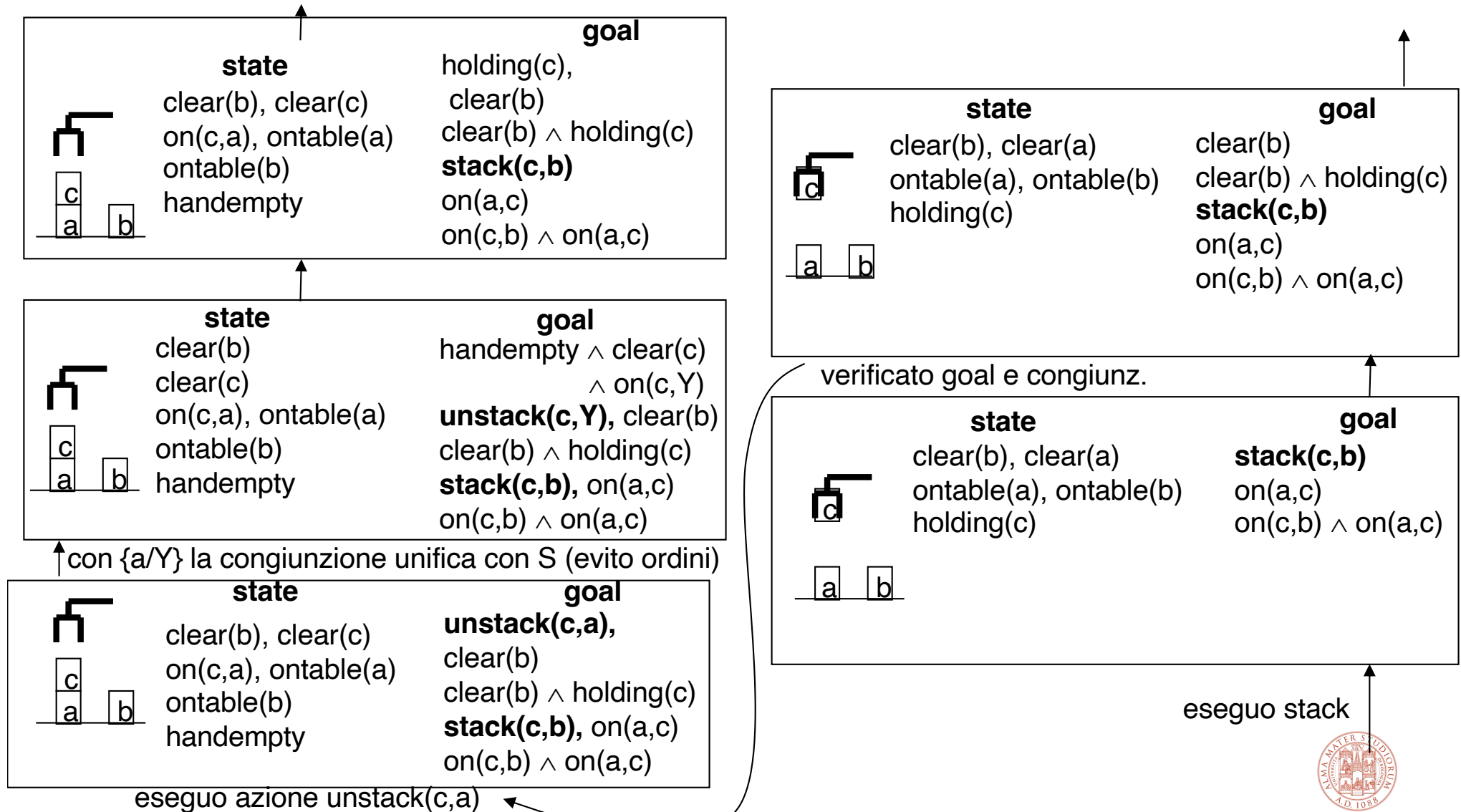
1. Il goal è la prima pila di obiettivi.
2. Suddividere il problema in sottoproblemi: ciascuno per un componente dell'obiettivo originale. Tali sottoproblemi possono interagire!
3. Abbiamo tanti possibili ordini di soluzione.
4. Ad ogni passo del processo di risoluzione si cerca di risolvere il goal 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.



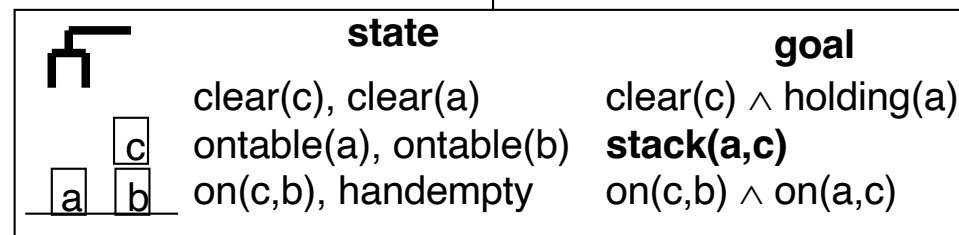
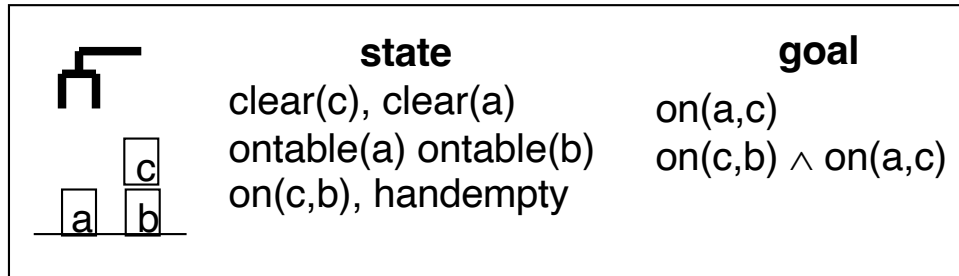
Esempio



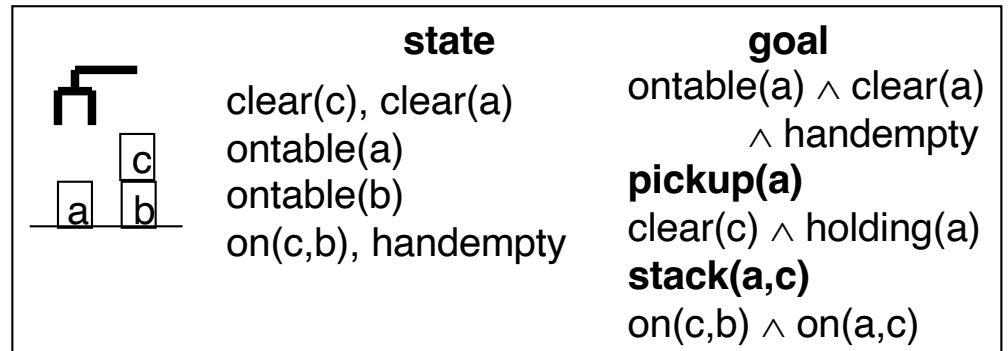
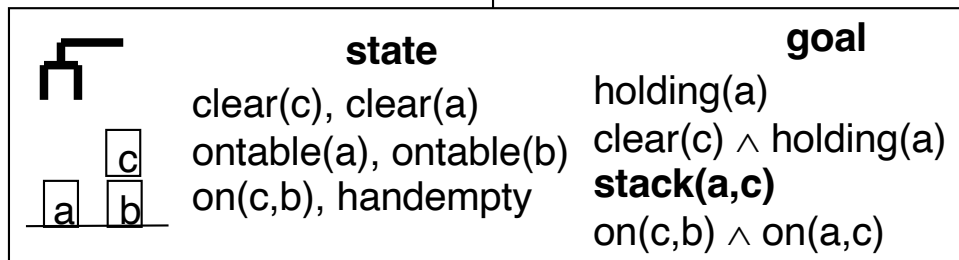
Esempio



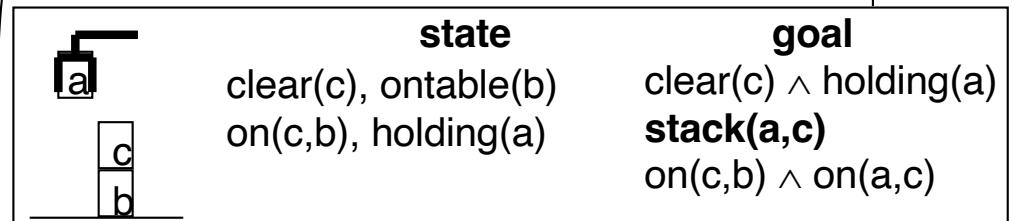
Esempio



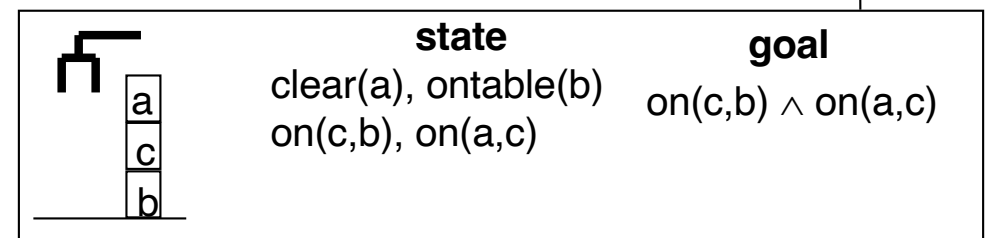
clear(c) vero in S



cong OK eseguo pickup



cong OK eseguo stack



elimino goal, stack vuoto
END

Ricostruendo da configurazione iniziale
a finale ho una soluzione:

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



Alcuni problemi...

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



Alcuni problemi...

- 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).

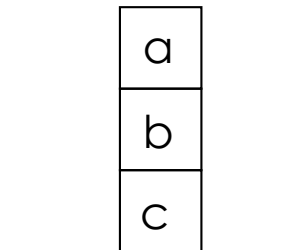
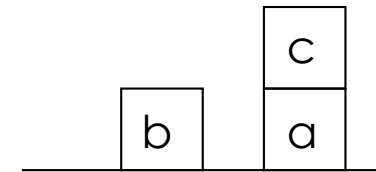
STRIPS: Possibili stack iniziali:

(1)
on(a,b)
on(b,c)
 $\text{on(a,b)} \wedge \text{on(b,c)}$

(2)
on(b,c)
on(a,b)
 $\text{on(a,b)} \wedge \text{on(b,c)}$

La soluzione migliore:

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

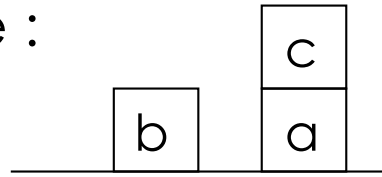


Decidiamo di scegliere (1)

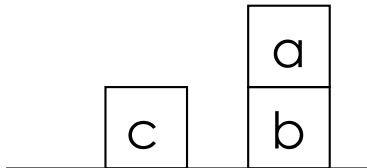


Anomalia di Sussmann

Applicando il procedimento di soluzione di STRIPS otteniamo a partire dallo stato iniziale :



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)

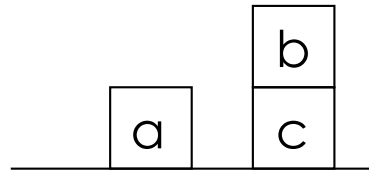
Stato attuale: 



Anomalia di Sussmann

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

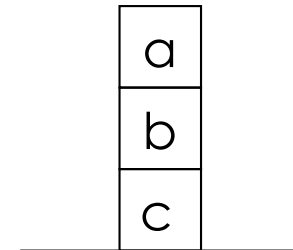
Allora reinseriamo $\text{on}(a,b)$ nello stack di goals (è la differenza rispetto allo stato attuale).



Otteniamo:

9. pickup(a)

10. stack(a,b)



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



Pianificazione Non Lineare

Gli Algoritmi di Planning presentati hanno problemi di efficienza in caso di domini con molti operatori.

Tecniche per rendere più efficiente il processo di pianificazione:

- Pianificatori Gerarchici;
- Pianificatori non Lineari e basati su vincoli
- I pianificatori non lineari, o di ordinamento parziale **POP (Partial Order Planning)** 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 studierà nel Corso di Intelligent Systems.



Planning in pratica

Molte applicazioni in domini complessi:

- Pianificatori per Internet (ricerca di informazioni, softbot)
 - Robotica
 - Piani di produzione industriale
 - Logistica
-
- Possibili progetti (robot ad es. Robot NAO) ed approfondimenti



Esecuzione del piano

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 del piano

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 (pianificatori reattivi).



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 l'ambiente esterno 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.

