Planning

Il Problema

Trovare una **sequenza di azioni** che, dato un particolare **stato iniziale del mondo**, consente di soddisfare un dato **goal**.

Cioè, dati

- Un set di operatori (che definiscono le *azioni primitive* che un agente può eseguire sul mondo),
- Una descrizione dello stato iniziale del mondo, e
- Una descrizione dello stato goal,

individuare un piano che corrisponde a

- Una sequenza di istanze di operatori la cui esecuzione a partire dallo stato iniziale cambi lo stato del mondo in modo da soddisfare la descrizione dello stato goal.

Lo stato goal è descritto come congiunzione di goal elementari

Pianificazione Classica

Assunzioni

- **There is a single causal agent and this agent is the planner;**
  Rappresentare altre entità che possono cambiare lo stato del mondo e ragionare su come possono cambiarlo complica la pianificazione.
  Agli inizi della ricerca su questo argomento, si stabilì che solo il pianificatore potesse avere effetti sul mondo e quindi un controllo completo sui cambiamenti nel mondo.
- **The planner is given a well-defined goal which remains fixed over the course of planning;**
  definire il goal è cruciale; se I goal vengono definiti male è impossibile pianificare. Visto che la scelta delle azioni da eseguire è dettata dai goal da raggiungere, se il goal cambia tutte le decisioni prese devono essere riviste.

Pianificazione Classica

Assunzioni

The planner is assumed to have:

- **a complete and accurate knowledge of the starting situation;**
- **the knowledge required to accurately model the world;**
- **the resources (time and memory) required to use this model to reason about the possible worlds associated with the different courses of action that might be pursued.**

La pianificazione è, in generale, un problema intrattabile. Se il mondo è complesso, anche la rappresentazione del mondo diventa un problema intrattabile.

Altre assunzioni classiche

- Ogni azione è indivisibile
- Non sono ammesse azioni concorrenti
- Deterministic actions: Il risultato delle azioni è completamente determinato – nessuna incertezza sui loro effetti
- Closed World Assumption: tutto quello che si sa essere vero è incluso nella descrizione dello stato. Ciò che non è descritto è falso.

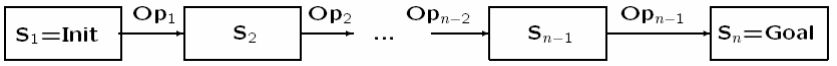
Approcci Principali alla Pianificazione

- **Situation calculus**
- **Planning nello spazio degli stati**
- **Partial order planning**
- **Grafi di Planning**
- **Decomposizione Gerarchica (HTN planning - HDPOP)**
- **Planning Reattivo (Reactive planning)**

Pianificazione Classica

- 2 Fasi: Generazione ed esecuzione del piano
- Dati: Stato Iniziale, Stato Goal, Set di Operatori
- Task: Trovare gli operatori di piano che trasformano lo stato iniziale nello stato goal

- Descrivere gli stati in un linguaggio formale (ad esempio, calcolo dei predicati o un suo sottoinsieme)
- Un piano è una sequenza di azioni che trasformano lo stato iniziale nello stato goal



Blocks world

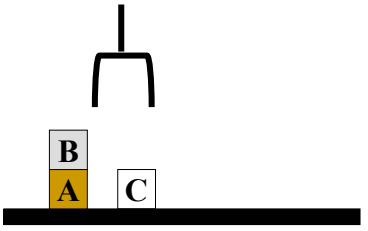
E' un "micro-world" composto da un tavolo "table", un insieme di blocchi (a,b,c, …) ed una mano robotica "robot hand".

Alcuni vincoli:

- Un solo blocco può essere su un altro blocco
- Sul tavolo ci possono essere più blocchi
- La mano può prendere un blocco alla volta

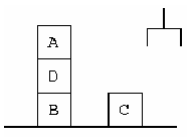
Esempi di predicati:

- Table(x) "x è sul tavolo"
- On(x,y) "x è su y"
- Clear(x) "non c'e' niente su x"
- Handempty "la mano del robot è vuota"
- Holds(x) "x è nella mano del robot"



Cosa è un "Planning Problem"

Un esempio di goal:



Table(B)
Table(C)
On(D,B)
On(A,D)
Clear(A)
Clear(C)
Handempty

Operatori disponibili:

PICKUP(x) " prendere x dal tavolo"
PUTDOWN(x) "mettere x sul tavolo"
STACK(x,y) "mettere x su y"
UNSTACK(x,y) "togliere x da y"

Formalizzare gli operatori!
Trovare un piano

Situation calculus

- Intuizione: Rappresentare problemi di planning con FOL
  - consente di ragionare sui cambiamenti nel mondo
  - usa metodi di "theorem proving" per provare che una particolare sequenza di azioni, se applicata alla situazione che caratterizza lo stato iniziale, condurrà al risultato desiderato.

Situation calculus

- McCarthy e Hayes [1969] aggiungono altri argomenti alla descrizione di uno stato e descrivono gli operatori usando formule della logica dei predicati
- Ad esempio:

Table(A,Init)	Ontable(B,Goal)
Table(B,Init)	Ontable(C,Goal)
On(D,A,Init)	On(D,B,Goal)
On(C,D,Init)	On(A,D,Goal)
Clear(C,Init)	Clear(A,Goal)
Clear(B,Init)	Clear(C,Goal)
Handempty(Init)	Handempty(Goal)

Il secondo termine delle formule atomiche descrive lo stato in cui la proprietà è vera

Situation calculus - operatori

PICKUP(x, ξ) ∀x, ξ Clear(x, ξ) ∧ Ontable(x, ξ) ∧ Handempty(ξ) → Holds(x, PICKUP(x, ξ)) ∧ ¬Handempty(PICKUP(x, ξ)) ∧ ¬Clear(x, PICKUP(x, ξ)) ∧ ¬Ontable(x, PICKUP(x, ξ))	PUTDOWN(x, ξ) ∀x, ξ Holds(x, ξ) → Clear(x, PUTDOWN(x, ξ)) ∧ Ontable(x, PUTDOWN(x, ξ)) ∧ Handempty(PUTDOWN(x, ξ)) ∧ ¬Holds(x, PUTDOWN(x, ξ))
STACK(x, y, ξ) ∀x, y, ξ Holds(x, ξ) ∧ Clear(y, ξ) ∧ → Clear(x, STACK(x, y, ξ)) ∧ On(x, y, STACK(x, y, ξ)) ∧ Handempty(STACK(x, y, ξ)) ∧ ¬Holds(x, STACK(x, y, ξ)) ∧ ¬Clear(y, STACK(x, y, ξ))	UNSTACK(x, y, ξ) ∀x, y, ξ Clear(x, ξ) ∧ On(x, y, ξ) ∧ Handempty(ξ) → Holds(x, UNSTACK(x, y, ξ)) ∧ Clear(y, UNSTACK(x, y, ξ)) ∧ ¬Clear(x, UNSTACK(x, y, ξ)) ∧ ¬On(x, y, UNSTACK(x, y, ξ)) ∧ ¬Handempty(UNSTACK(x, y, ξ))

- Cosa significa OPERATORE(x, ξ)?

ξ rappresenta lo stato raggiunto dopo l'applicazione dell'OPERATORE ad x

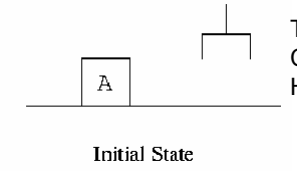
Quindi, ad esempio:

Holds (x, PICKUP(x, ξ))

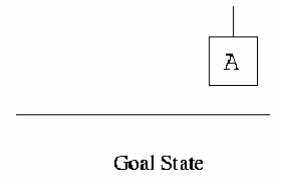
formalizza il fatto seguente:

"x è nella mano del robot, nello stato che si raggiunge applicando l'operatore di PICKUP ad x nello stato ξ"

Situation calculus – un esempio



Table(A,Init)
Clear(A,Init)
Handempty(Init)



Holds(A,Goal)

Applicando PICKUP allo stato iniziale e al blocco A:

Holds(A,PICKUP(A,INIT))
¬ Handempty(PICKUP(A,INIT))
¬ Clear(A,PICKUP(A,INIT))
¬ Ontable(A,PICKUP(A,INIT))

⇓

Goal:=PICKUP(A,INIT)

STRIPS – Un esempio di pianificatore lineare

Fikes and Nilsson 1971: STanford Research Institute Problem Solver

Pianificatore lineare: un pianificatore in cui *l'ordine* in cui i *sottogoal* vengono risolti è in relazione lineare con l'ordine in cui le *azioni* del piano vengono eseguite.

Rappresentazione degli Stati e dei Goal:

- STRIPS:
  - Descrive stati e operatori in un linguaggio "ristretto"
  - Stati: una congiunzione di "fatti" (letterali che non contengono variabili "ground")
  - Goal: congiunzione di letterali positivi

STRIPS – Operatori

Ogni operatore viene definito in base al seguente **schema**:

Azione: nome della azione descritta dall'operatore

Preconditions: espressione logica (solo ^) che descrive le condizioni che devono essere vere per applicare quell'operatore;

Delete List: quello che cessa di essere vero dopo l'esecuzione dell'azione (set di espressioni che devono essere cancellate dal modello dello stato del mondo quando viene applicato l'operatore)

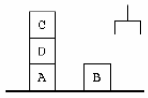
Add List: quello che diventa vero dopo l'applicazione dell'operatore.

Add e Delete List -> Effects

Operatori Block World - STRIPS

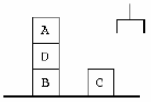
PICKUP(x)		PUTDOWN(x)	
preconditions	Clear(x)	preconditions	Holds(x)
	Ontable(x)	delete list	Holds(x)
	Handempty	add list	Clear(x)
delete list	Clear(x)		Ontable(x)
	Ontable(x)		Handempty
	Handempty		
add list	Holds(x)		
STACK(x,y)		UNSTACK(x,y)	
preconditions	Holds(x)	preconditions	Clear(x)
	Clear(y)		On(x,y)
delete list	Holds(x)		Handempty
	Clear(y)	delete list	Clear(x)
add list	Clear(x)		On(x,y)
	On(x,y)		Handempty
	Handempty	add list	Holds(x)
			Clear(y)

Esempio



Ontable(A)
Ontable(B)
On(D,A)
On(C,D)
Clear(C)
Clear(B)
Handempty

Init

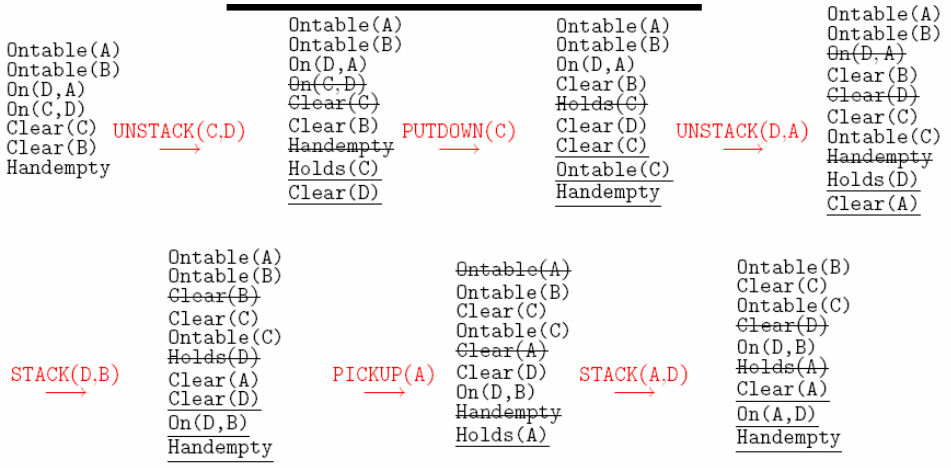


Ontable(B)
Ontable(C)
On(D,B)
On(A,D)
Clear(A)
Clear(C)
Handempty

Goal

Plan:
(UNSTACK(C, D), PUTDOWN(C), UNSTACK(D, A), STACK(D, B), PICKUP(A), STACK(A, D))

Proviamo insieme che è vero

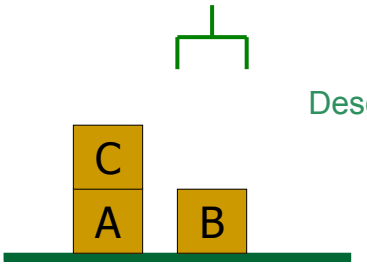


"Planning Problem" in STRIPS

Quindi un "planning problem" in STRIPS si formalizza descrivendo:

- Stato Iniziale *Init* (dato da una formula del calcolo dei predicati (ristretto))
- Stato goal *Goal* (dato da una formula del calcolo dei predicati (ristretto))
- set di operatori (*nome, precondition, add & delete list*)

Esercizio

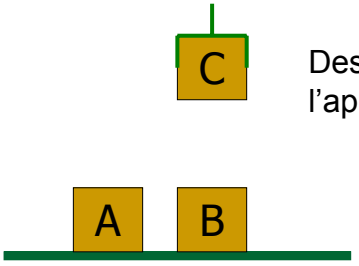


Descrivere lo Stato Iniziale

Descrivere Istanza Operatore Unstack(C,A)

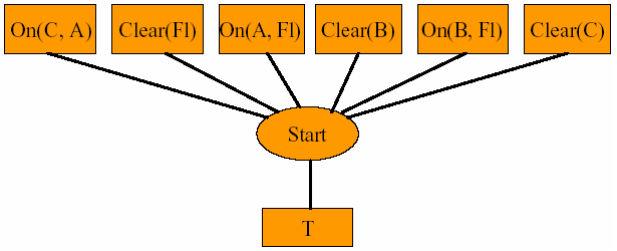
- Precondition
- Effects
  - Add
  - Delete

Esercizio

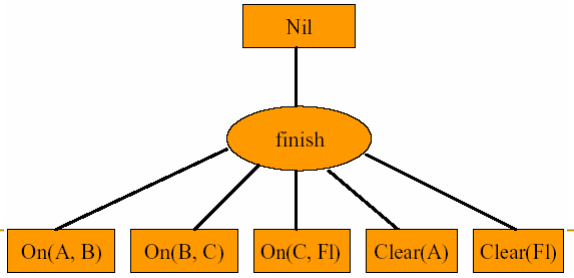


Descrivere il nuovo stato dopo l'applicazione di Unstack(C,A)

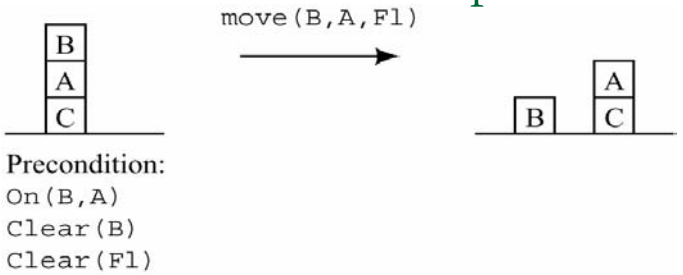
Rappresentazione Grafica di *Init*



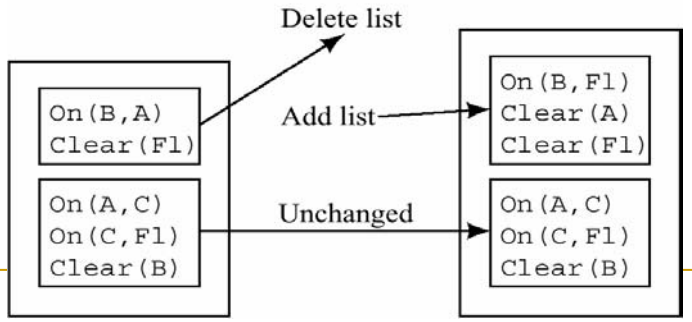
Rappresentazione Grafica di *Goal*



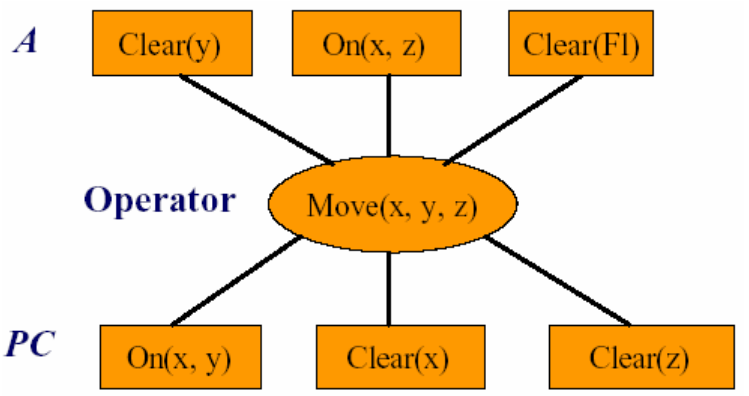
Esercizio: formalizzare l'operatore *move*



Precondition:
On(B,A)
Clear(B)
Clear(Fl)



Rappresentazione grafica di *un operatore*



Planning come ricerca

La pianificazione può essere vista come un problema di ricerca.

- Ingredienti della ricerca:
  - Spazio degli stati contenenti tutte le possibili configurazioni
    - Lo stato iniziale e quello goal devono essere dati
  - Operatori per manipolare gli stati
  - Goal Test per testare se lo stato goal è stato raggiunto
  - Funzioni per calcolare il costo di un particolare percorso

La pianificazione diventa quindi un problema di ricerca nello spazio degli stati possibili rispetto al situation calculus in cui la ricerca era effettuata nello spazio delle formule

COME USARE STRIPS

Ricerca nello *spazio delle sitiuazioni* connesso alle istanze degli operatori.

- A partire da una situazione iniziale a cui appartiene il goal si applicano man mano gli operatori ad essa connessi.
- La sequenza di tali operatori è il piano.
- L'applicazione di una istanza di operatore si simula cambiando la base di conoscenza

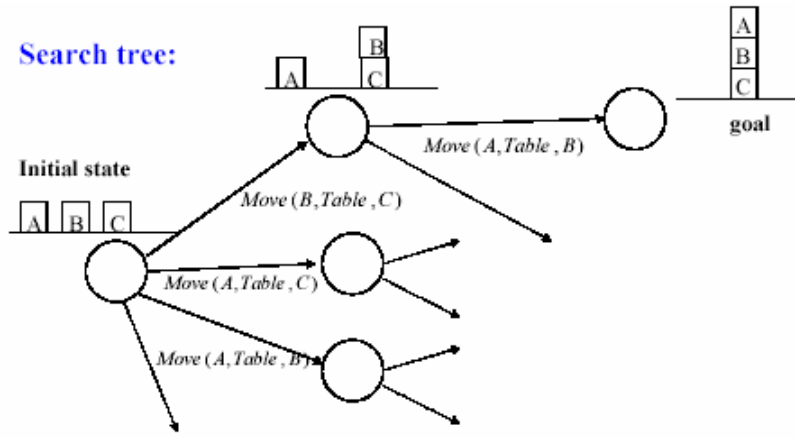
Strategie:
Forward dallo stato iniziale
Backward dallo stato goal

Forward: progressione nello spazio degli stati

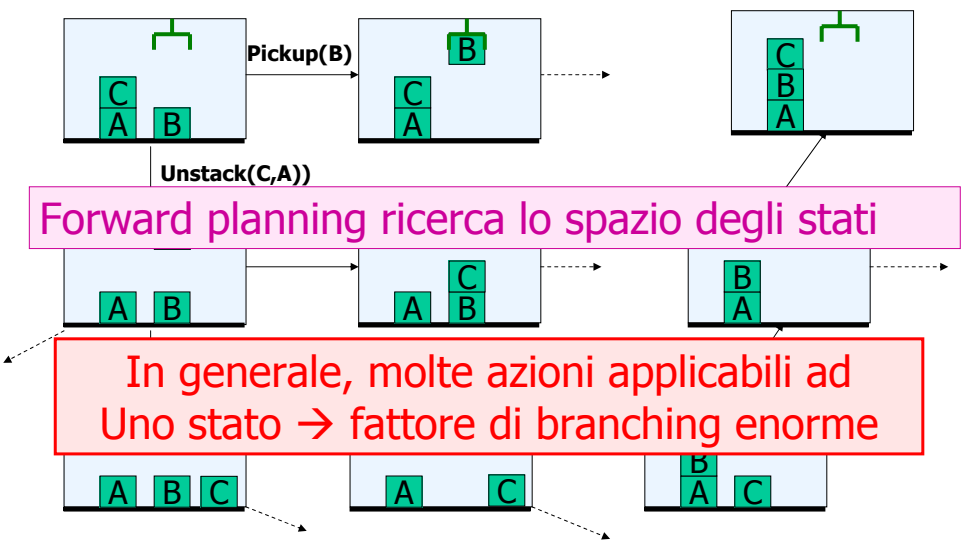
- **Algoritmo:**
  1. Si parte da *Init*
  2. Si trovano tutti gli operatori le cui prencondizioni sono vere nello stato corrente (S_c)
  3. Computare gli effetti degli operatori applicabili per generare gli stati successivi
  4. Ripetere gli step 2-3, finche' il nuovo stato soddisfa le condizioni espresse in *Goal*

Forward Search (progressione verso il goal)

Si usano gli operatori per generare nuovi stati da esplorare
Per ogni nuovo stato si controlla se si è raggiunto il goal (goal test function)



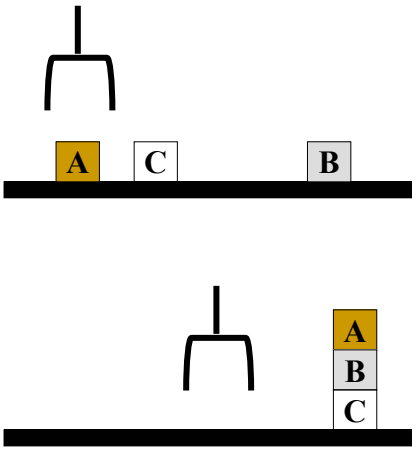
Forward Planning



Esempio

Stato Iniziale:
- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:
- on(b,c)
- on(a,b)
- ontable(c)

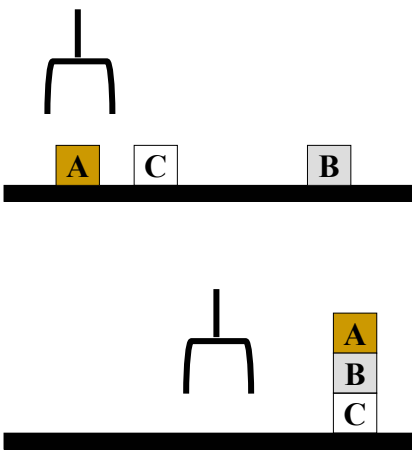


Un piano:
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

Un altro esempio

Stato Iniziale:
- clear(a)
- clear(b)
- clear(c)
- ontable(a)
- ontable(b)
- ontable(c)
- handempty

Goal:
- on(a,b)
- on(b,c)
- ontable(c)



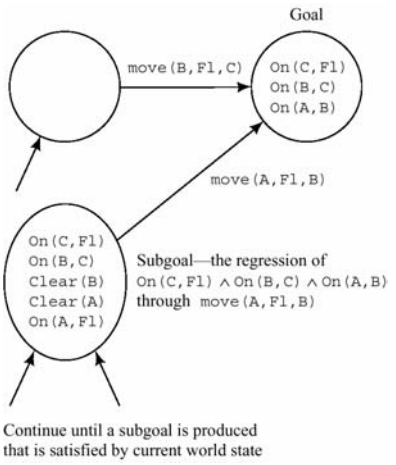
Un piano:
- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)

BackWard Search (goal regression)

Dato un goal g in una lista di goal G:
-Cercare un operatore che soddisfa il goal g (deve essere nella sua add list)
-Aggiungere le sue precondizioni a G.

Algoritmo:

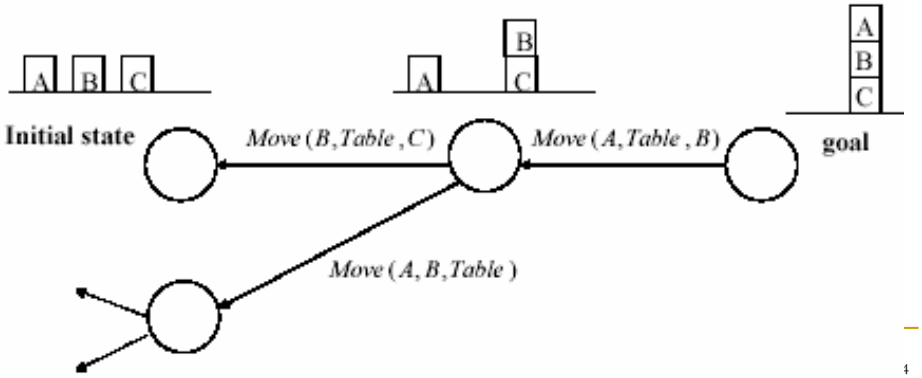
1. Si parte dallo stato *Goal*
2. Si seleziona un operatore che ha nella sua *add list* uno dei letterali che costituiscono *Goal*
3. Si sostituisce quelo goal con le precondizioni di quell'operatore
4. Si ripetono gli steps 2-3 finchè non si raggiunge lo stato iniziale *Init*



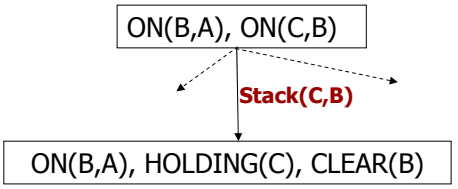
BackWard Search (goal regression)

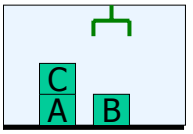
- Use operators to generate new goal conditions
- Check whether the initial state satisfies the current goal

Search tree:



Backward Chaining





In generale, ci sono molte meno azioni rilevanti per un goal che azioni applicabili → fattore di branching più piccolo che con il forward planning

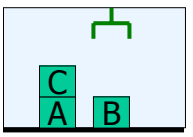
Backward Chaining

ON(B,A), ON(C,B)

Stack(C,B)

ON(B,A), HOLDING(C), CLEAR(B)

Pickup(C)



ON(B,A), CLEAR(B), HANDEMPTY, CLEAR(C), ON(C,TABLE)

Backward planning ricerca lo spazio dei goal

CLEAR(C), ON(C,TABLE), HOLDING(B), CLEAR(A)

Pickup(B)

CLEAR(C), ON(C,TABLE), CLEAR(A), HANDEMPTY, CLEAR(B), ON(B,TABLE)

Putdown(C)

CLEAR(A), CLEAR(B), ON(B,TABLE), HOLDING(C)

Unstack(C,A)

CLEAR(B), ON(B,TABLE), CLEAR(C), HANDEMPTY, ON(C,A)

STRIPS: Goal-Stack Planning

- STRIPS mantiene due strutture dati:
  - **Lista di Stati** – tutti i predicati correntemente veri.
  - **Goal Stack** – una pila di goal da risolvere, con il goal corrente in testa alla pila.

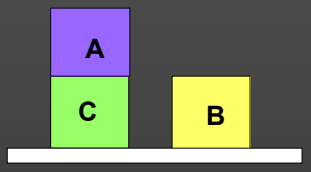
Algoritmo:

- Dato un Goal:
  1. Inizializza: PUSH del goal nello Stack.
  2. Se Top(Stack) è soddisfatto dallo stato corrente, POP.
  3. Altrimenti, se Top(Stack) è una congiunzione, PUSH dei singoli letterali nello stack.
  4. Altrimenti, controlla se esiste un qualche operatore la cui add-list può essere unificata con Top(Stack), PUSH di quell'operatore e delle sue precodizioni nello Stack.
  5. Se Top(Stack) è una azione POP e EXECUTE.
  6. Loop 2-5 finche' Empty(Stack).

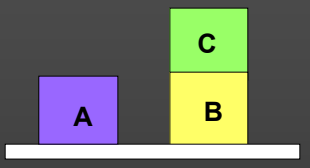
Planning in STRIPS

- STRIPS mantiene due strutture dati:
  - **Lista di Stati** – tutti i predicati correntemente veri.
  - **Pila di Goal** – una pila di goal da risolvere, con il goal corrente in testa alla pila.
- Se il goal corrente non è soddisfatto dallo stato presente, esamina gli effetti positivi degli operatori, e inserisce l'operatore e la lista delle precondizioni sulla pila. (Subgoal)
- Quando il goal corrente è soddisfatto, lo rimuove dalla pila.
- Quando un operatore è in testa alla pila, registra l'applicazione dell'operatore sulla sequenza del piano e usa gli effetti per aggiornare lo stato corrente.

Esempio di funzionamento di STRIPS

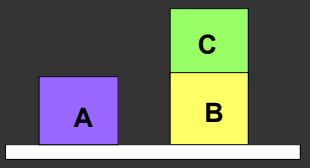


Stato iniziale



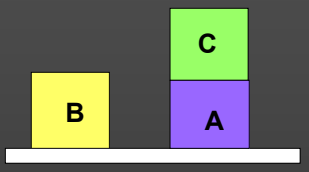
goal

PILA

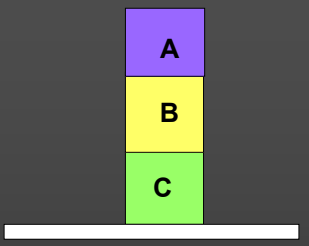


Stato corrente

Esercizio

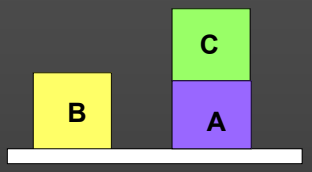


Stato iniziale

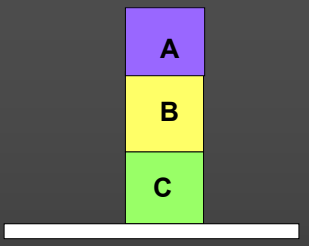


goal

Soluzione

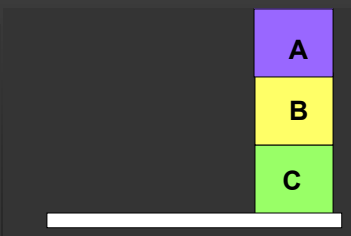


Stato iniziale



goal

PILA



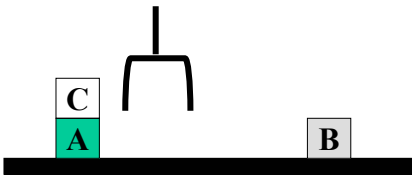
On(b, c) ∧ On(a, b)

Soluzione

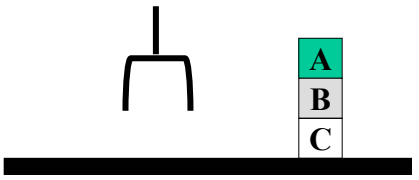
- Il piano generato è:
  [Unstack(c), Stack(a, b), Unstack(a),
  Stack(b, c), Stack(a ,b)]
- Un altro piano generabile è:
  [Stack(b, c), Unstack(b), Unstack(c),
  Stack(a, b), Unstack(a), Stack(b, c), Stack(A,B)]
- Il piano ideale non ottenibile con pianificazione “lineare”:
  [Unstack(c), Stack(b, c), Stack(a, b)]

Interazione fra i goal

- Gli algoritmi di pianificazione visti finora assumono l'indipendenza fra I goal: ogni goal può essere risolto separatamente, alla fine si concatenano le soluzioni
- "Sussman Anomaly," è l'esempio classico per mostrare il problema dell'interazione fra goal. Risolvere prima:
  - On(A,B) (effettuando Unstack(C,A), Stack(A,B) verrà annullato quando si cerca di risolvere il goal On(B,C) (Unstack(A,B),Stack(B,C)).
  - On(B,C) verrà annullato quando si risolve On(A,B)
- STRIPS classico non può risolvere questo problema



Initial state



Goal state

Riassumendo
Planning nello spazio degli stati

- Spazio delle situazioni (localizzazione, possedimenti, etc.)
- Il piano è una soluzione trovata "cercando" tra le situazioni il goal
- Un **planner progressivo** cerca il goal in avanti (forward) a partire dallo stato iniziale
- Un **planner regressivo** cerca all'indietro (backward) a partire dal goal
- Attenzione: problema della Anomalia di Sussman