

Abbiamo bisogno di uno strumento che sia in grado di ragionare sulle azioni, sulla loro applicabilità e i loro effetti.

→ non previsto dalla rappresentazione ontologica.

## Pianificazione (Planning)

- è una branca della IA molto ampia
- il mondo di interesse si trova in una certa situazione in cui valgono determinate proprietà → noi vogliamo modificare questo mondo
- ci limitiamo alla pianificazione delle azioni di un singolo agente

Vogliamo capire quali azioni sono applicabili sulla situazione  $S_1$  per raggiungere la situazione  $S_2$  in cui vale una proprietà obiettivo.



determinare una sequenza di azioni che permette di passare da una situazione corrente ad una situazione in cui vale una certa proprietà obiettivo

- 
- GOAL ACHIEVEMENT (problem solving)
  - mantenere vera una condizione

Tipicamente si interagisce col mondo reale ⇒ NO BACKTRACKING

↳ le azioni consumano risorse

Linguaggio di pianificazione :

PDDL (Planning Domain Definition Language)



Si ispira al Situation Calculus

## Situation Calculus

- è stata la prima proposta di formalismo di rappresentazione di contesti di planning e il primo strumento di inferenza
- si basa sulla logica del prim'ordine

Prevede 4 elementi cardine:

- **AZIONE** : qualcosa che può essere compiuto e influenza il mondo
- **SITUAZIONE** : stato di cose
- **FLUENTE** : predicato il cui valore di verità può cambiare
- **PREDICATO ATEMPORALE** : condizione immutabile

**AZIONE**  $\text{MOVE}(R, L_1, L_2)$  sposta l'oggetto R dalla pos.  $L_1$  alla pos.  $L_2$

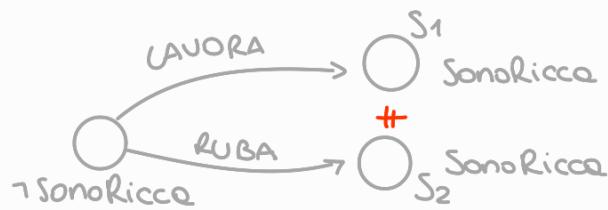
↓  
in FOL è una FUNZIONE

$R$  

funzione che fa riferimento ad un oggetto intangibile

**SITUAZIONE** : effetto dell'applicazione di una sequenza di azioni

$\text{Do}(\[], s) = s$  → se non faccio nulla il mondo resta così com'è  
 $\text{Do}([a/\text{resto}], s) = \text{Do}([\text{resto}], \text{Risultato}(a, s))$   
il percorso (il come) è significativo



> stesso risultato ma due situazioni diverse

Axioma:

$$[\text{Do}(\text{Azioni}_1, s_1) = \text{Do}(\text{Azioni}_2, s_2)] \Leftrightarrow (\text{Azioni}_1 = \text{Azioni}_2) \wedge (s_1 = s_2)$$

## AZIONE:

### • Assioma di applicabilità

$$\forall \text{PARAMS}, s \quad \text{APPLICABLE}(\text{ACTION}(\text{PARAMS}), s) \Leftrightarrow \text{PRECOND}(\text{PARAMS}, s)$$

$$\forall x, y, s \quad \text{APPLICABLE}(\text{GO}(x, y), s) \Leftrightarrow \underbrace{\text{AT}(x, s)}_{\text{Fluente}} \wedge \underbrace{\text{ADJACENT}(x, y)}_{\text{pred. atemporale}}$$

### • Assioma di effetto

$$\forall \text{PARAMS}, s \quad \text{APPLICABLE}(\text{ACTION}(\text{PARAMS}, s)) \Rightarrow \underbrace{\text{EFFECTS}(\text{PARAMS}, \text{RESULT}(\text{ACTION}(\text{PARAMS}, s)))}_{\substack{\text{formula} \\ \text{congiunzione di} \\ \text{f. atomiche}}} \quad \substack{\text{RESULT}(\text{ACTION}(\text{PARAMS}, s)) \\ \hookrightarrow \text{situazione risultato} \\ \text{dell'applicazione} \\ \text{dell'azione}}$$

$$\forall x, y, s \quad \text{APPLICABLE}(\text{GO}(x, y, s)) \Rightarrow \text{AT}(y, \text{RESULT}(\text{GO}(x, y), s))$$

Esempio sul mondo dei blocchi

$\text{MOVE}(x, y, z)$  : sposta  $x$  da  $y$  a  $z$

Ass. applicabilità

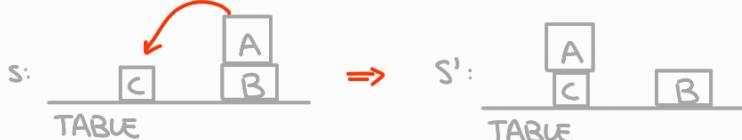
$$\forall x, y, z \quad \text{APPLICABLE}(\text{MOVE}(x, y, z), s) \Leftrightarrow \text{CLEAR}(x, s) \wedge \text{CLEAR}(z, s) \wedge \text{ON}(x, y, z) \wedge x \neq z \wedge y \neq z \wedge x \neq \text{TABLE}$$

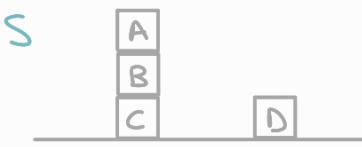


$\text{MOVE}(A, B, C)$  APPLICABLE? ✓

Ass. di effetto

$$\forall x, y, z, s \quad \text{APPLICABLE}(\text{MOVE}(x, y, z), s) \Rightarrow \text{ON}(x, z, \text{RESULT}(\text{MOVE}(x, y, z), s)) \wedge \text{CLEAR}(y, \text{RESULT}(\text{MOVE}(x, y, z), s))$$





MOVE(A,B,D)



Axioma effetto:

CLEAR(B,S')  
ON(A,D,S')

→ Ho perso informaz.  
su ciò che non  
è cambiato

? NON  
PERSISTE

↓  
ON(A,B,S)  
ON(B,C,S)  
CLEAR(D,S)

Ci serve qualche elemento di conoscenza in più per gestire ciò che  
non cambia → FRAME PROBLEM

- Approccio per enumerazione

azione : impatto limitato → voglio ragionare su ciò che non viene  
modificato dall'azione

### ASSIOMA di frame

forall PARAMS, VARS, S

FLUENT(VARS, S) ∧ PARAMS ≠ VARS ⇒ FLUENT(VARS, RESULT(ACTION(PARAMS), S))

Se in una situazione vale un certo fluente e in quella situazione viene applicata un'azione su elementi diversi da quelli del fluente, allora il fluente su quelle variabili rimane vero nella situazione risultante.

→ Vale per ogni fluente che non riguarda l'oggetto dell'azione

Poco pratico : 1) Tantissime formule  
2) se arricchiamo la KB con fluente nuovo, allora  
devo aggiungere tanti assiomi di frame

### ASSIOMA di Stato Successore

Se azione applicabile ⇒

(FLUENTE vero nella situazione risultante ⇔ (l'azione lo rende vero) ∨  
(era vero e l'azione non l'ha reso falso))

Quando si va a descrivere gli effetti di un'azione, questi si dividono in:

- Aggiunte = cosa viene reso vero
- Cancellazioni = cosa viene reso falso

## CONSEGUIRE GOAL COMPLESSI

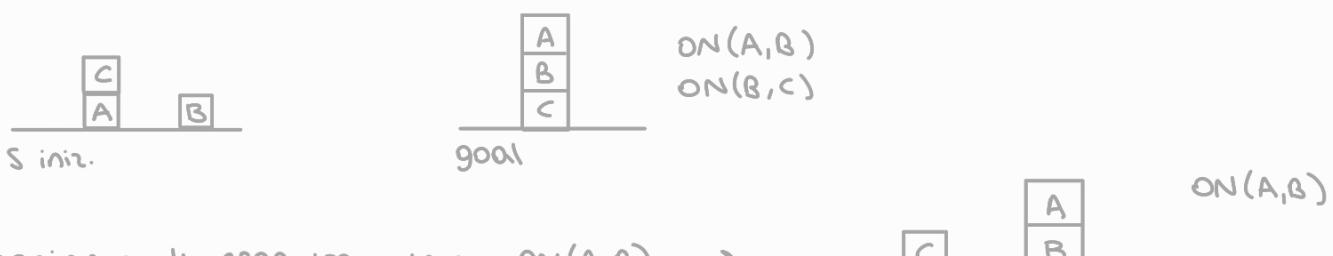
Goal complesso = congiunzione di proprietà

GOAL:  → posso vederlo come sotto-goal più semplici → ON(A,B) ON(B,C)

Strategia lineare :

- divido il goal complesso in sottogoal
- trovo dei piani per conseguire ciascuno dei sottogoal
- li eseguo in sequenza

⚠ NON SEMPRE SUFFICIENTE



i) Supponiamo di eseguire prima  $ON(A,B)$  →



Ora, per eseguire  $ON(B,C)$  devo disfare ciò che ho fatto prima !



2) Supponiamo di eseguire prima  $ON(B,C)$  →



Ancora non va bene, per eseguire  $ON(A,B)$  devo disfare ciò che ho fatto prima



ANOMALIA DI SUSSMAN