



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

# Soluzione di problemi tramite ricerca nello spazio degli stati

**Federico Chesani**

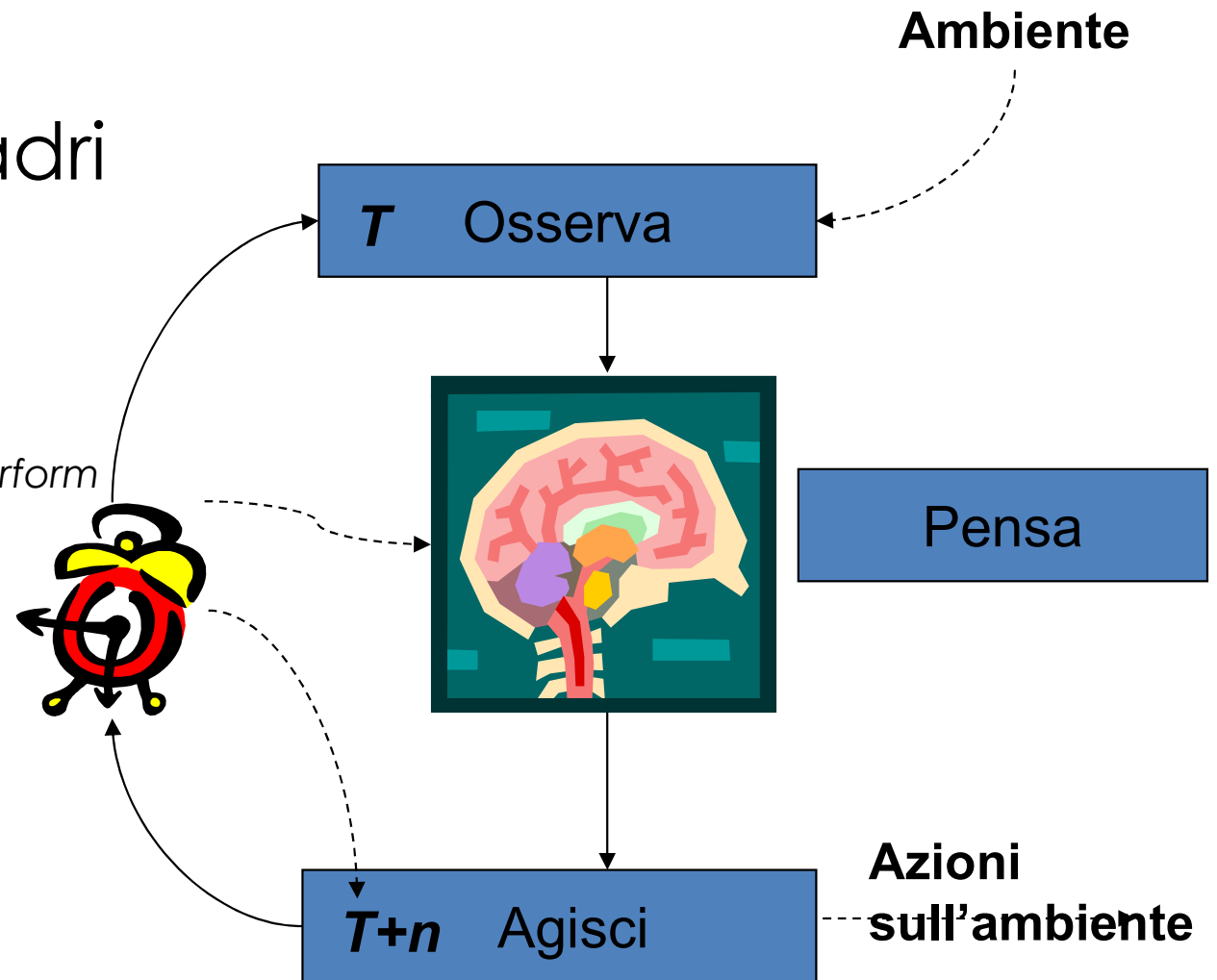
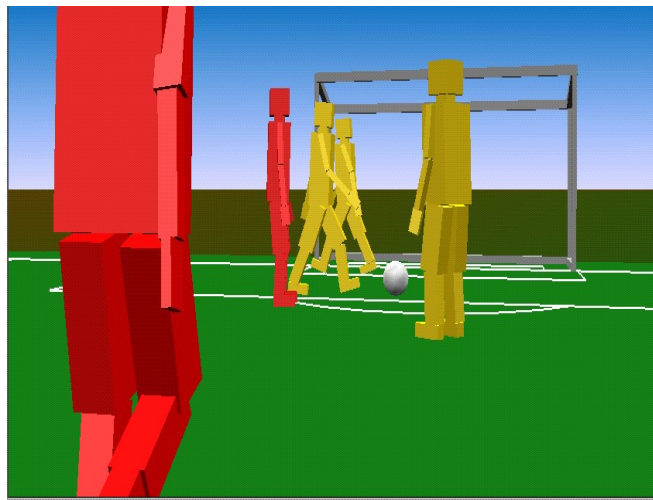
DISI

Department of Informatics – Science and Engineering

# Agenti Razionali

## Observe-think-act agent cycle di Kowalski and Sadri

- To **cycle** at time  $T$ 
  - **observe** any inputs at time  $T$
  - **think**
  - **select** one or more actions to perform
  - **act**
- **cycle** at time  $T+n$



# Problem-solving agents

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

S. Russell & P. Norvig: "Intelligenza Artificiale: un approccio moderno", Pearson ed.



# Soluzione del problema come ricerca in uno spazio degli stati

- Gran parte dei problemi di Intelligenza Artificiale hanno la ricerca (o controllo) come componente fondamentale.
- I problemi si possono modellare come **Problemi di Ricerca** in uno spazio degli stati (**Strategie di Ricerca**).
- **Spazio degli stati**
  - Lo spazio degli stati è l'insieme di tutti gli stati raggiungibili dallo stato iniziale con una qualunque sequenza di operatori.
- **Lo Spazio degli Stati è Caratterizzato da:**
  - Uno **stato iniziale** in cui l'agente sa di trovarsi (non noto a priori);
  - 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).
  - Un **cammino** è una sequenza di azioni che conduce da uno stato a un altro.



# TEST di RAGGIUNGIMENTO DEL GOAL

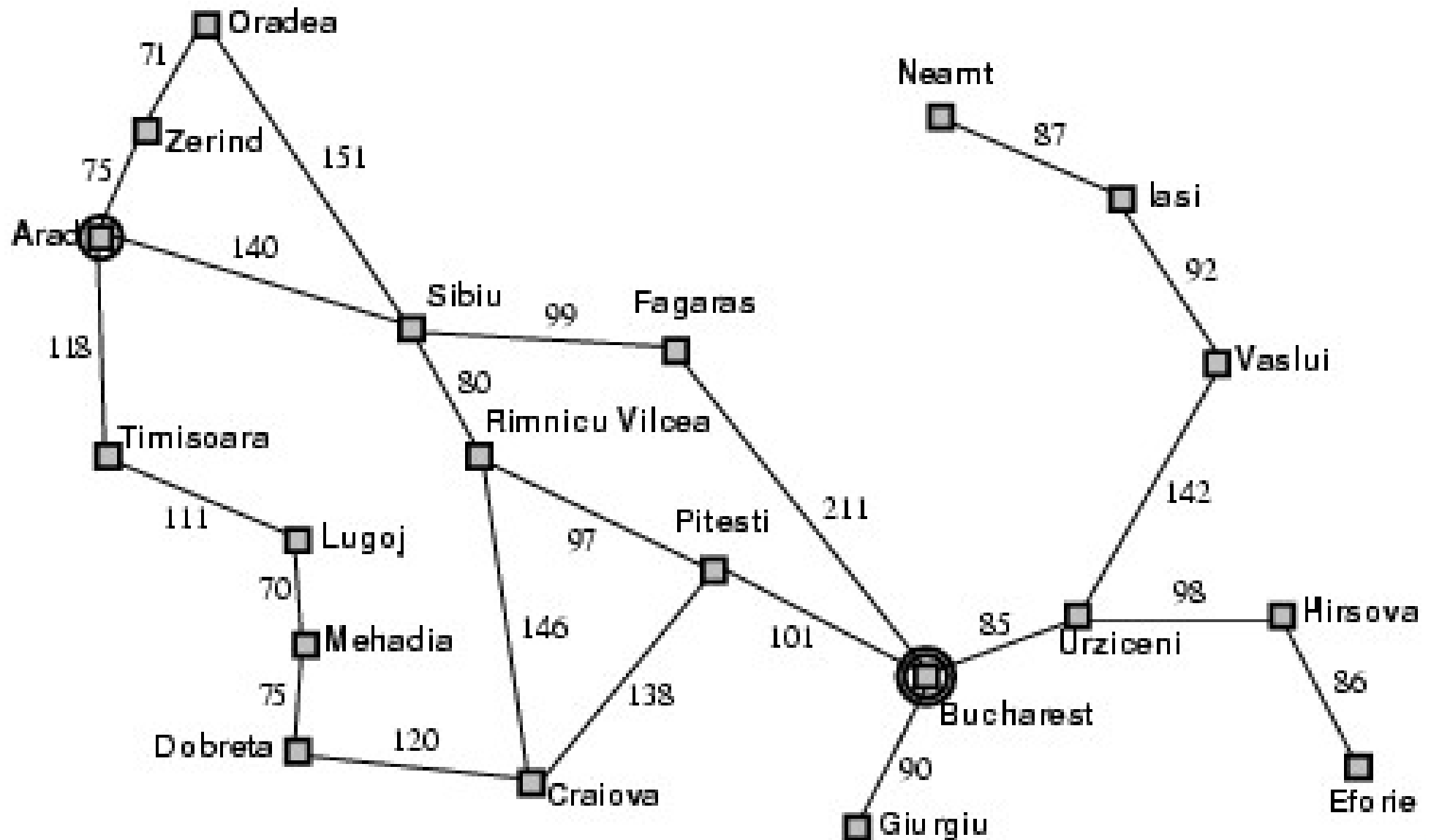
- La verifica può essere solo l'appartenenza dello stato raggiunto all'insieme dello stato (o degli stati) goal.
- A volte lo stato obiettivo può essere descritto in modo “astratto” attraverso proprietà (si pensi allo stato di scacco matto).
- **Altri obiettivi (non solo raggiungere il goal, ma...):**
  - trovare la **sequenza di operatori** che arrivano al goal;
  - trovare **tutte le soluzioni**;
  - trovare una **soluzione ottima**.
  - In quest'ultimo caso vuol dire che una soluzione può essere preferibile a un'altra.
  - Una funzione **costo di cammino** assegna un costo a un cammino (in gran parte dei casi quale somma del costo delle azioni individuali lungo il cammino).



## Esempio: Romania

- Una vacanza in Romania; attualmente in Arad.
- I voli partono da Bucharest domani
- goal:
  - Essere in Bucharest
- problema:
  - stati: varie città
  - azioni: guida fra le città
- soluzione:
  - Sequenza di città, ad es., Arad, Sibiu, Fagaras, Bucharest.

## Esempio: Romania



Da: S. Russell & P. Norvig: "Intelligenza Artificiale: un approccio moderno", Pearson ed.



# Formulazione del problema

Il **problema** è definito da quattro aspetti:

1. **Stato iniziale** es., "at Arad"
2. **Azioni o funzioni successore**  $S(x)$  = insieme di coppie azione-stato
  - es.,  $S(\text{Arad}) = \{ \langle \text{Arad} \rightarrow \text{Zerind}, \text{Zerind} \rangle, \dots \}$
3. **goal test**, può essere
  - **esplicito**, es.,  $x = \text{"at Bucharest"}$
  - **implicito**, es.,  $\text{controllamappa}(x)$
4. **Costo della strada** es., somma delle distanze, oppure numero di azioni eseguite ecc.  $c(x, a, y) \geq 0$

Una **soluzione** è una sequenza di azioni che portano dallo stato iniziale al goal.





## Esempio: 8-puzzle

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

stati? Locazioni delle tessere

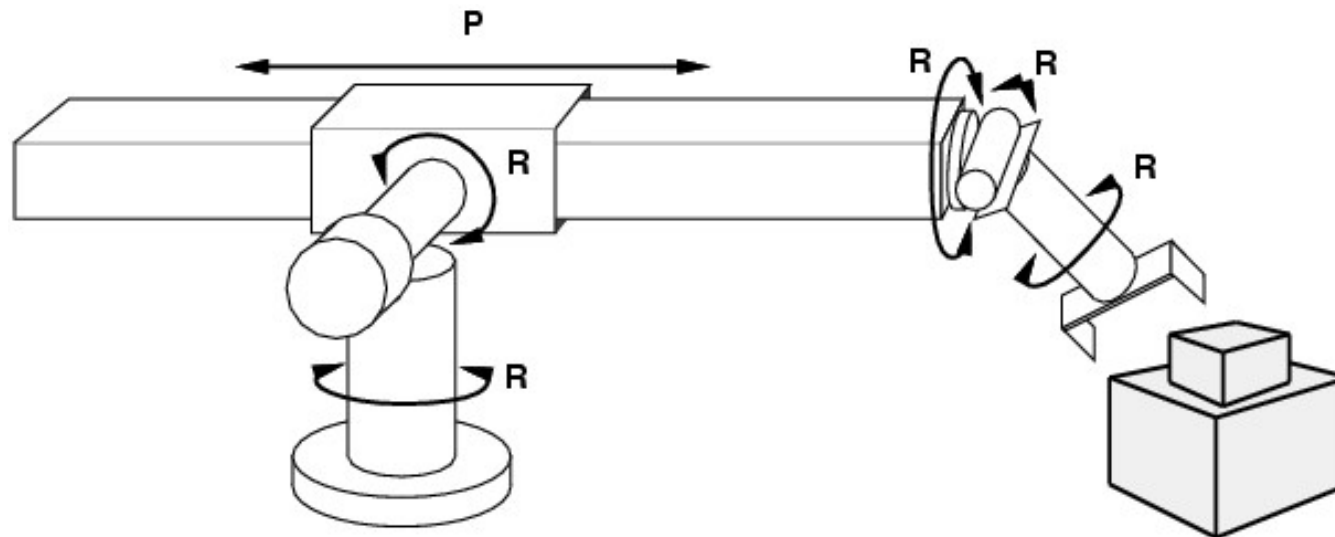
azioni? Muovi la lacuna a sinistra, destra, alto, basso.

goal test? = stato goal (dato)

costo? 1 per mossa.



## Esempio: assemblaggio con un robot



**stati:** coordinate dei giunti del robot, parti dell'oggetto da assemblare e posizione.

**actions?:** movimenti dei giunti del robot.

**goal test?:** completamente assemblato

**costo della strada:** tempo di esecuzione.



## ESEMPIO: MISSIONARI E CANNIBALI

- 3 missionari e 3 cannibali devono attraversare un fiume. C'è una sola barca che può contenere al massimo due persone. Per evitare di essere mangiati i missionari non devono mai essere meno dei cannibali sulla stessa sponda (stati di fallimento).
- Stato: sequenza ordinata di tre numeri che rappresentano il numero di missionari, cannibali e barche sulla sponda del fiume da cui sono partiti.
- Perciò lo stato iniziale è:  $(3,3,1)$  (nota l'importanza dell'astrazione).



# ESEMPIO: MISSIONARI E CANNIBALI

- Operatori: si deve portare in barca
  - 1 missionario, 1 cannibale,
  - 2 missionari,
  - 2 cannibali,
  - 1 missionario
  - 1 cannibale.
- Al più 5 operatori (grazie all'astrazione sullo stato scelta).
- Test Obiettivo: Stato finale (0,0,0)
- Costo di cammino: numero di traversate.



## ESEMPIO: CRIPTOARITMETICA

$$\begin{array}{rcccccc} & S & E & N & D & + \\ & M & O & R & E & = \\ \hline M & O & N & E & Y & \end{array}$$

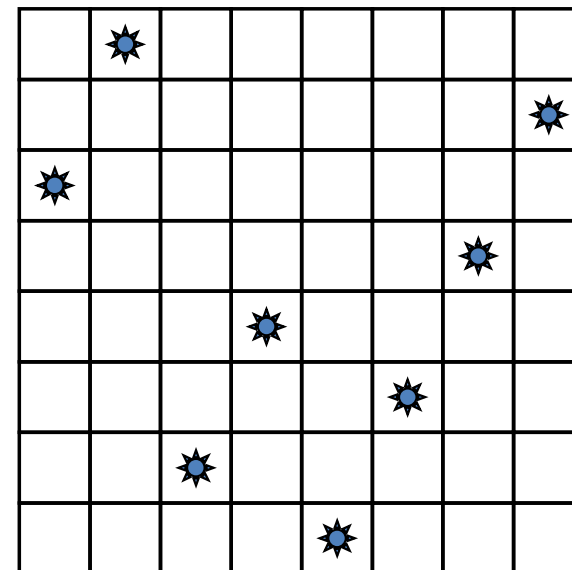
- Operatori: sostituisci tutte le stesse lettere con una cifra che non compare nel rompicapo;
- Test obiettivo: il rompicapo contiene solo cifre e rappresenta una somma corretta;
- Costo di cammino: 0



# IL PROBLEMA DELLE N REGINE

Inserire 8 regine su una scacchiera in modo che non si mangino.

- **Stati**: qualsiasi configurazione da 0 a N regine sulla scacchiera;
- **Operatori**: aggiungi una regina in un qualsiasi quadrato;
- **Test obiettivo**: N regine sulla scacchiera, nessuna minacciata;
- **Costo** di cammino: zero.



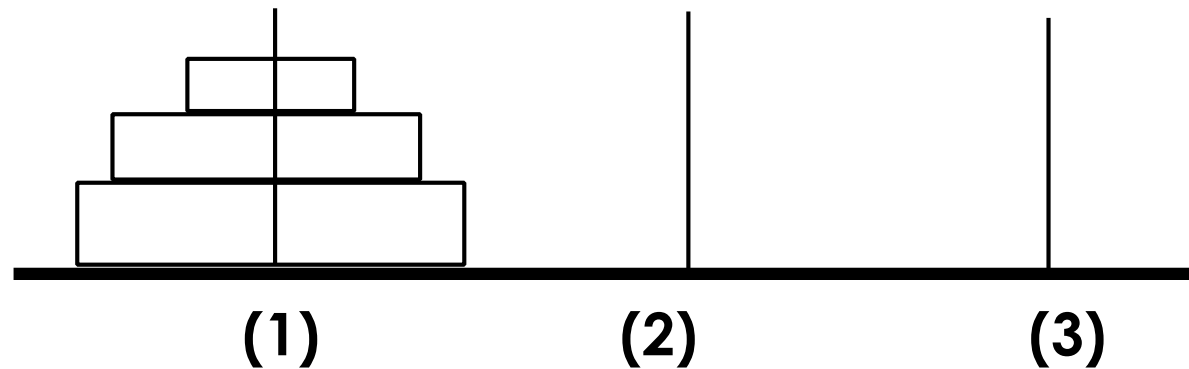
# SUDOKU

- Alcune caselle sono già fissate, le altre vanno riempite con numeri dall'1 al 9
- la tavola è suddivisa in 9 quadranti di 3x3 caselle
- su ogni quadrante devono essere messi tutti e 9 i numeri, senza ripetizioni
- inoltre, ogni riga orizzontale e ogni riga verticale dell'intera tavola non deve contenere ripetizioni di numeri

		9				7		
	4		5		9		1	
3				1				2
	1			6			7	
		2	7		1	8		
	5			4			3	
7				3				4
	8		2		4		6	
		6				5		



## ESEMPIO: LA TORRE DI HANOI



- Spostare i cilindri concentrici da (1) a (3) nella stessa configurazione di (1) utilizzando eventualmente anche (2). Un cilindro più grande non può essere inserito su un cilindro più piccolo.





## ALTRI ESEMPI

- Il problema del commesso viaggiatore:
  - Un commesso viaggiatore ha una lista di città che deve visitare tutte una sola volta. Vi sono strade dirette fra ogni coppia di città. Si trovi la strada più breve che il commesso deve seguire per compiere un viaggio completo che inizi e termini in una qualsiasi delle città.
  - (esplosione combinatoria, per 10 città è 10!)
- Il problema della scimmia e la banana:
  - Afferrare una banana appesa al soffitto (avendo a disposizione una sedia e bastone).



## ESEMPIO: CAPRA, LUPO E CAVOLO

- Portare capra, lupo e cavolo dall'altra parte senza che si mangino (si mangiano se rimangono assieme senza il conducente della barca). Nella barca se ne può trasportare uno solo alla volta.
- Soluzione:
  - Porta la capra sull'altra sponda;
  - Torna indietro
  - Porta il cavolo sull'altra sponda
  - Porta indietro la capra
  - Porta il lupo sull'altra sponda
  - Torna indietro
  - Porta la capra sull'altra sponda



# SPAZIO DI RICERCA MOLTO AMPIO

- Giochi: scacchi (li vedremo)
- Le parole crociate

T	U	O
R		R
E	R	A



# ALTRE CARATTERISTICHE DEL PROBLEMA

## IL SISTEMA È SCOMPONIBILE?

- Esempio:

- $\int X + 3X + \sin X * \cos X \, dX$

corrisponde alla somma degli integrali:

- $\int X \, dX + \int 3X \, dX + \int \sin X * \cos X \, dX$

## NON TUTTI I SISTEMI SONO SCOMPONIBILI:

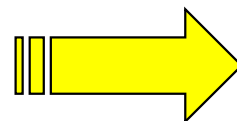
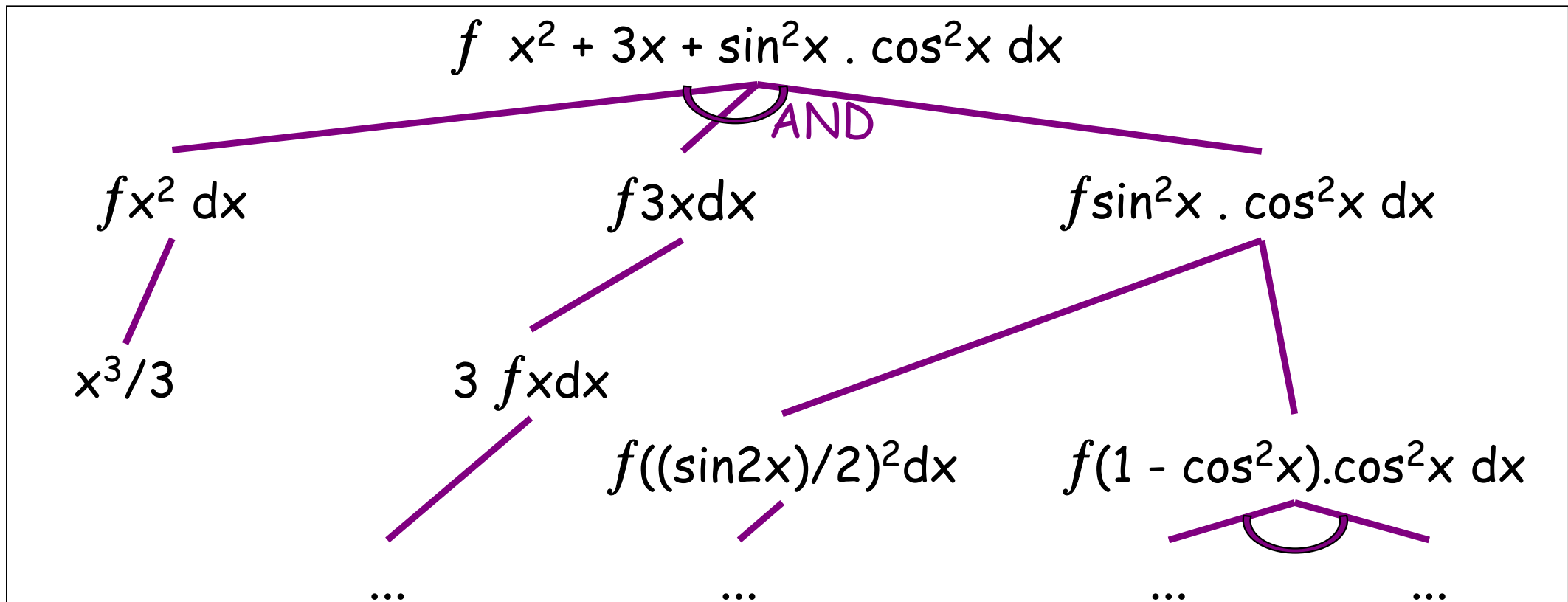
- Si possono avere sotto-problemi interagenti,



# Scomposizione di problemi

Es: Integrali simbolici:

- Stato: l'integrale da calcolare
- Regole: riduzioni di integrali
- Obiettivo: tutti gli integrali devono essere eliminati



Albero AND-OR



## ESEMPIO: II MONDO A BLOCCHI (planning)

- Stato Iniziale:

```
handempty . clear(b) . clear(c) .  
on(c,a) . ontable(a) . ontable(b) .
```

- Operatori:

```
clear(X) → ontable(X)  
clear(X) and clear(Y) → on(X,Y)
```

- Goal:

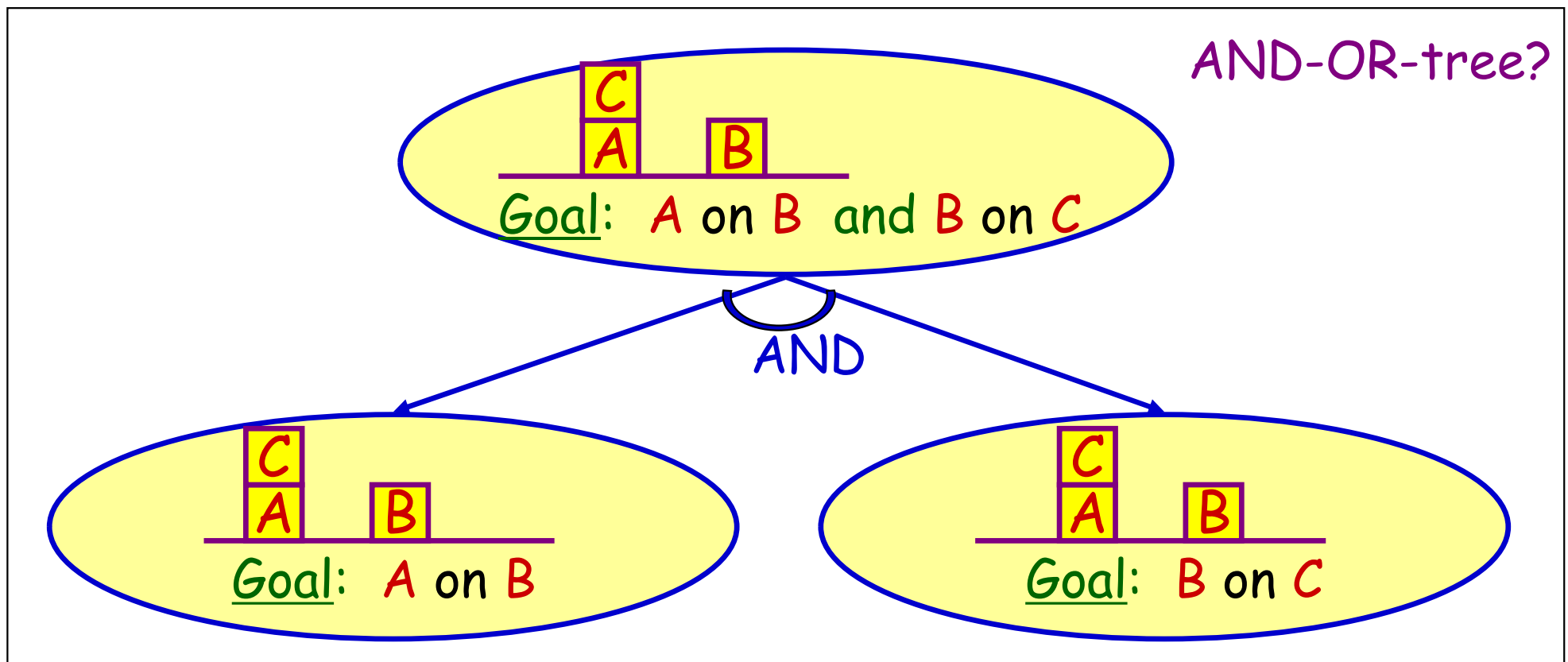
```
on(b,c) and on(a,b)
```

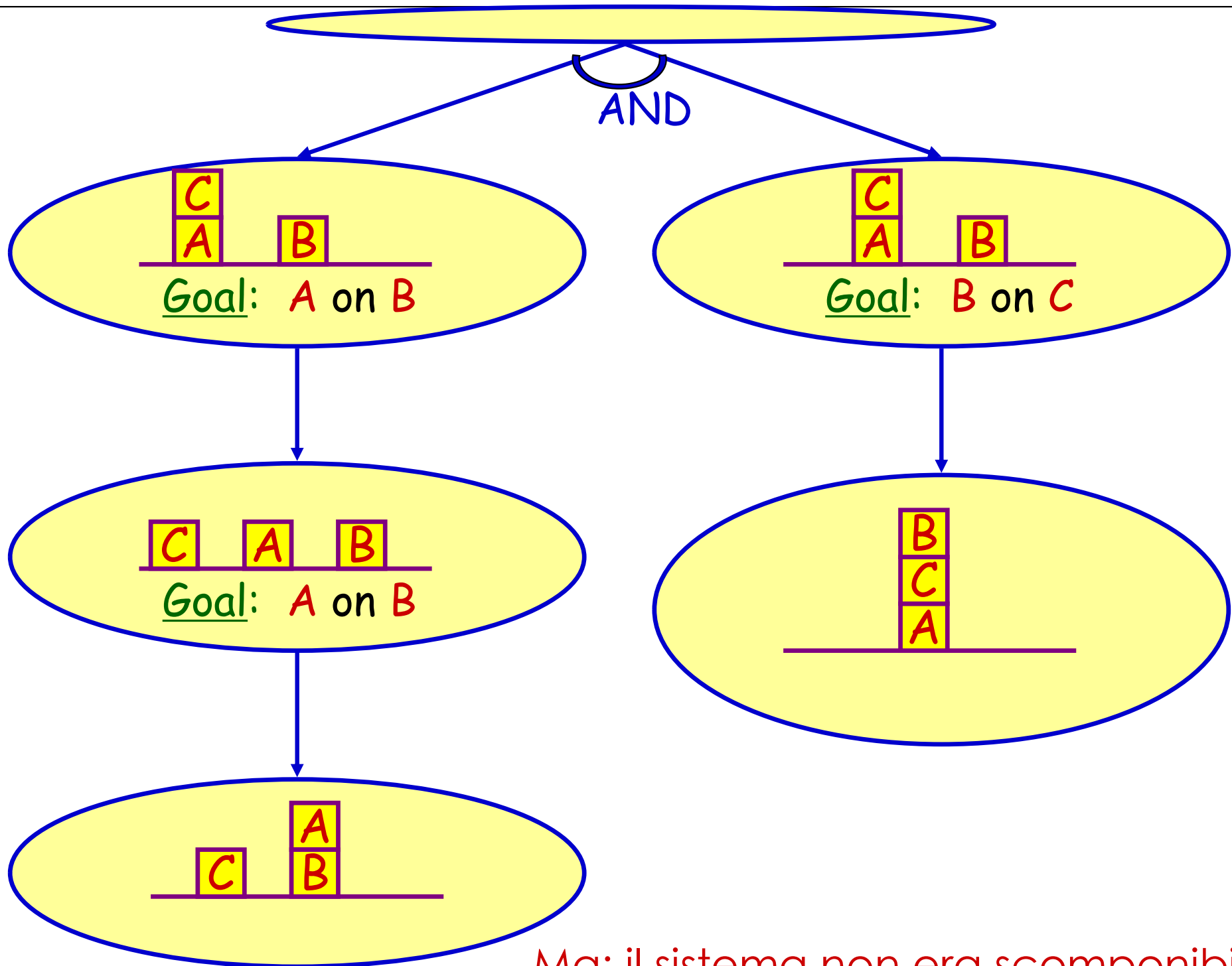
- I due sottogoal NON POSSONO ESSERE RISOLTI SEPARATAMENTE perché interagiscono.



## Mondo a blocchi

- Inizialmente: C is on A and B is on the table.
- regole: to move any free block to another or to the table
- Goal: A is on B and B is on C.





Ma: il sistema non era scomponibile!



# LOGICA e DIMOSTRATORE DI TEOREMI

- Asserzioni Logiche:

**a.**

**a → e.**

**a → d.**

**d → f.**

- Goal (teorema da dimostrare): **f.**

Il problema è scomponibile perché monotono. Non si può invalidare un teorema già dimostrato.

- Sistema di produzioni monotono: L'applicazione di una regola R non invalida la possibilità di applicare regole applicabili all'atto della selezione di R.
- Non è il caso invece del mondo a blocchi.
- È possibile trasformare un sistema di produzione non-monotono in uno monotono. La trattazione, però, si complica notevolmente.

# Problemi a stati singoli

- Problema a stati singoli:
  - Lo stato è sempre accessibile.
  - L' agente conosce esattamente che cosa produce ciascuna delle sue azioni e può calcolare esattamente in quale stato sarà dopo qualunque sequenza di azioni.
- Problema a stati multipli:
  - Lo stato non è completamente accessibile. L' agente deve ragionare su possibili stati che potrebbe raggiungere.
  - In più: anche l'effetto delle azioni può essere sconosciuto o imprevisto.
  - Spesso risolvere questi problemi richiede capacità di rilevamento durante la fase di esecuzione → Agire nel mondo reale piuttosto che in un suo modello.

■ Tratteremo estesamente solo problemi a stati singoli

