GESTIONE DELLA CONOSCENZA
E
INTELLIGENZA ARTIFICIALE

Programma

- Strategie per la risoluzione di problemi:
  - Soluzioni nello spazio degli stati
  - Soluzione per decomposizione in sotto-problemi
  - Ricerca in ampiezza, profondità e mediante euristica
- Logica:
  - La logica proposizionale
  - La logica del primo ordine
  - La logica non monotona (cenni)
  - Procedure di decisione

- Rappresentazione della conoscenza:
  - Le reti semantiche
  - Le regole di produzione
  - I frame
  - Gli approcci ibridi
- Modelli di ragionamento e di apprendimento: incertezza, inferenza bayesiana, belief
- Sistemi basati sulla conoscenza:
  - I sistemi esperti: problematiche e classificazioni, con particolare riguardo alle applicazioni;
  - L'apprendimento automatico; interfaccia utente nell'ambito dei sistemi basata sulla conoscenza (cenni);

- Riconoscimento di configurazioni (*pattern recognition*):
  - preelaborazione ed estrazione delle caratteristiche distintive (*features*)
  - funzioni di decisione
  - metodi di classificazione
  - confronto mediante programmazione dinamica
- Architetture che imitano i sistemi biologici: reti neurali, connessionismo, memoria distribuita sparsa.

- *Laboratori e/o esercitazioni*

  Tesine su tutti i titoli proposti. Possibilità di utilizzare il PROLOG e un tool per sistemi esperti, KappaPC.

- *Modalità d'esame*

  È prevista una prova scritta e un eventuale colloquio ad integrazione della prova scritta. L'allievo potrà approfondire uno degli argomenti del corso, a sua scelta, svolgendo una tesina che verrà valutata in sede di esame.

Testi

Testi di riferimento:

- Stuart J. Russell, Peter Norvig, *"Intelligenza Artificiale. Un approccio Moderno"*, Pearson Education Italia, Milano.
- E. Rich, *"Intelligenza artificiale"*, McGraw Hill, Milano.
- N.J. Nilsson, *"Metodi per la risoluzione dei problemi nell'intelligenza artificiale"*, Angeli, Milano.

Testi ausiliari:

- Nils J. Nilsson, *"Intelligenza Artificiale"*, Apogeo, Milano.
- I. Bratko, *"Programmare in prolog per l'intelligenza artificiale"*, Masson Addison Wesley, Milano.

Introduzione

INTELLIGENZA ARTIFICIALE

- Settore dell'informatica che ha come obiettivo la realizzazione di sistemi che svolgono attività che richiederebbero intelligenza se compiute dall'uomo.
- Difficile definire l'intelligenza.
- Molti temi interessante dell'IA sorgono dal tentativo di realizzare le facoltà mentali di persone normali, come comprensione del linguaggio naturale o di immagini.

APPROCCIO FORTE

- L'IA come aggregato interdisciplinare che ha come obiettivo la comprensione della natura dell'intelligenza e la sua riproduzione con una macchina (contributi di informatici neurofisiologi, linguisti, filosofi, sociologi, …)

APPROCCIO DEBOLE

- L'IA è interessata al comportamento generale che caratterizza l'intelligenza o ma non ad un particolare modo di ottenere i risultati (potrebbe essere diverso da quello usato dall'uomo)
- È comunque molto utile sapere come fa l'uomo a svolgere certi compiti.

STORIA DELL'IA

- **1956 DARTMOUTH CONFERENCE**
  Mc Carthy - Minsky - Newell – Simon

- ANNI 60 RISOLUZIONE DI PROBLEMI
  Strategie di ricerca di soluzioni
  Dimostrazione automatica di teoremi
  Giochi
  Utilizzazione della logica matematica

- ANNI 70 RAPPRESENTAZIONE DELLA CONOSCENZA
  Ruolo della conoscenza
  Tecniche di rappresentazione:
  Linguaggio naturale
  Visione

- ANNI 80

  Utilizzazione delle metodologie di risoluzione di problemi e rappresentazione della conoscenza per risolvere problemi del mondo reale.

PRINCIPALI APPLICAZIONI DELL'IA

- **SISTEMI ESPERTI**

  Sistemi che dimostrano una competenza confrontabile con quella di un esperto umano in un campo specialistico particolare (medicina, chimica, geologia, ingegneria).
- **LINGUAGGIO NATURALE**

  Interfacce per accesso a base dati.
- **COMPRENSIONE DI SEGNALI E IMMAGINI**

  Linguaggio parlato

  Visione per robot

  Immagini in biomedicina

PROSPETTIVE DELL'INFORMATICA

- Sviluppo di sistemi per l'elaborazione della conoscenza
- KNOWLEDGE INFORMATION PROCESSING SYSTEMS
- Penetrazione dei calcolatori in tutte le aree della società e della vita quotidiana.
- Necessità dell'uso anche da parte di non specialisti.
- Elaboratori come strumento di supporto e per l'esecuzione di processi creativi in ambienti industriali e sociali.

Funzioni principali dei futuri elaboratori

- PROBLEM-SOLVING e INFERENZA
- GESTIONI DI BASI DI CONOSCENZA
- INTERFACCE INTELLIGENTI
- linguaggio naturale scritto e parlato
- grafica
- immagini

POSSIBILI APPLICAZIONI

- PROGETTO (VLSI, CAD)
- PROCESSI DI PRODUZIONE (CAM, Robot)
- **SISTEMI ESPERTI**
- SISTEMI DI SUPPORTO ALLE DECISIONI
- AUTOMAZIONE DEGLI UFFICI
- BASI DI DATI
- TRADUZIONE AUTOMATICA
- ISTRUZIONE (CAI)

ESEMPIO DI RISOLUZIONE DI PROBLEMI IN INFORMATICA TRADIZIONALE

- CALCOLO DEL FATTORIALE

  fatt(n) = 1 x 2 x … x n
- Dalla definizione si ricava immediatamente un metodo di soluzione sotto forma di successione di passi elementari (ALGORITMO)

  i := 1;

  fatt := 1;

  while i < n *do begin*

  i := i + 1;

  fatt := fatt * i;

  end

- Il processo di soluzione è caratterizzato da tre componenti:

  DATI OPERAZIONI CONTROLLO

  DATI: sono contenuti in uno STATO

i	1
fatt	1

stato iniziale

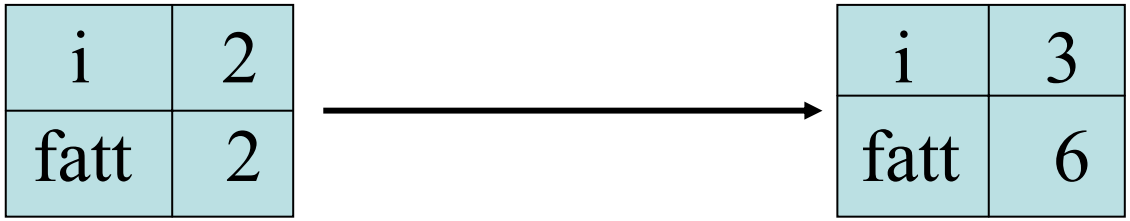
i	4
fatt	24

stato finale per n = 4

OPERAZIONI: trasformazione di stati

Se i <n → {i := i+1;

fatt := fatt*i}

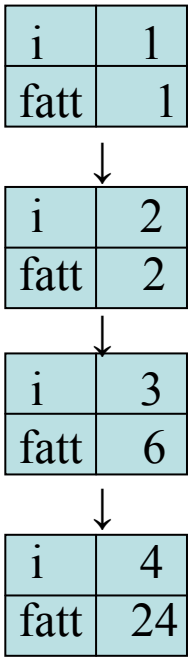


CONTROLLO: determina l'ordine in cui si eseguono le operazioni

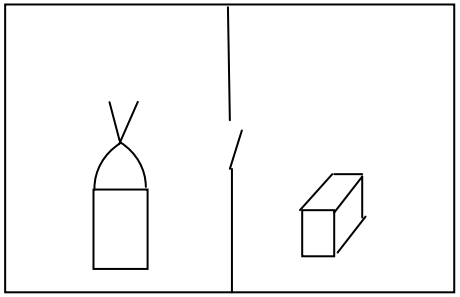
CONTROLLO SEQUENZIALE E DETERMINISTICO

Soluzione come sequenza di operazioni che determina una successione di stati

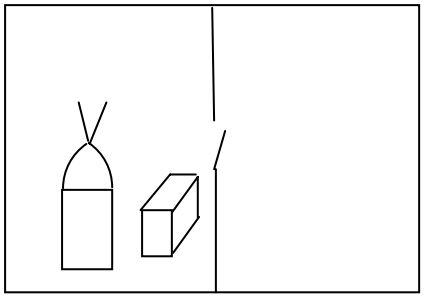
Esempio per n=4



Generazione di piani per robot



Stato iniziale



Stato finale

Operazioni elementari:
Vai a X
Apri la porta
Prendi il pacco
Attraversa la porta

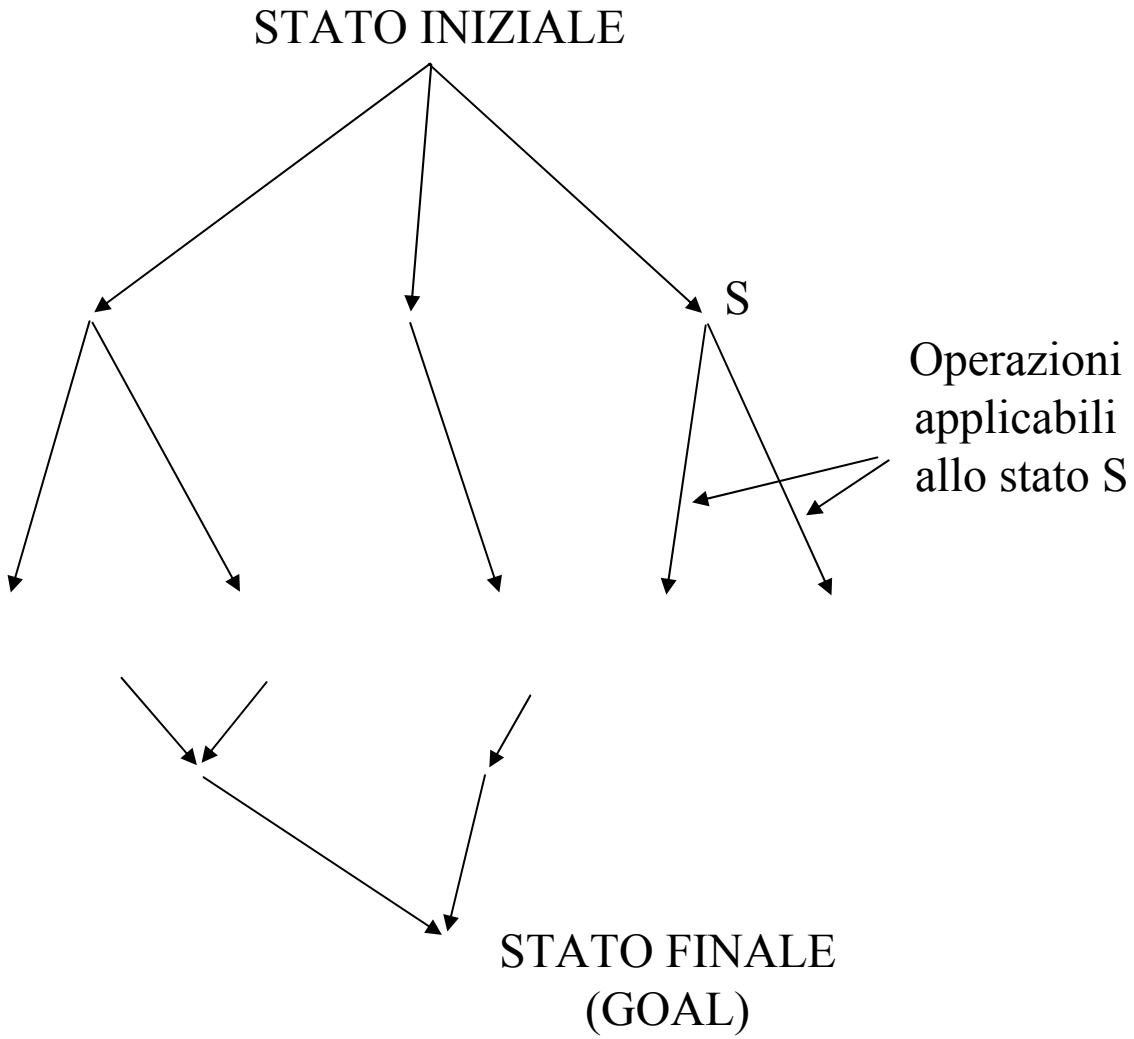
- Problema: Determinare la sequenza di operazioni che porta dallo stato iniziale allo stato finale.

- Soluzione:

  Vai a B - Apri la porta - Attraversa la porta - Vai a C - Prendi il pacco - Vai a B - Attraversa la porta - Vai a A

- Nel caso del fattoriale la sequenza di passi che porta dallo stato iniziale a quello finale è determinato dalla persona che ha inventato l'algoritmo.
- Nel caso del robot, la sequenza di passi che costituisce la soluzione deve essere determinata dalla macchine.
- Possiamo quindi dire che la macchina svolge un compito analogo a quello della persona che ha "inventato" l'algoritmo per il fattoriale.

RAPPRESENTAZIONE DEL PROBLEMA NELLO SPAZIO DEGLI STATI



- Ogni sistema per risolvere problemi che si basi sulle idee di stato e operatore si dice che fa uso del metodo dello spazio degli stati.
- ***Problema:*** come descrivere gli stati?

  (si preferisce lavorare con una *descrizione* delle disposizioni piuttosto che con le *disposizioni stesse*).
- Ci sono varie possibilità, a seconda del problema.

a) Alberi

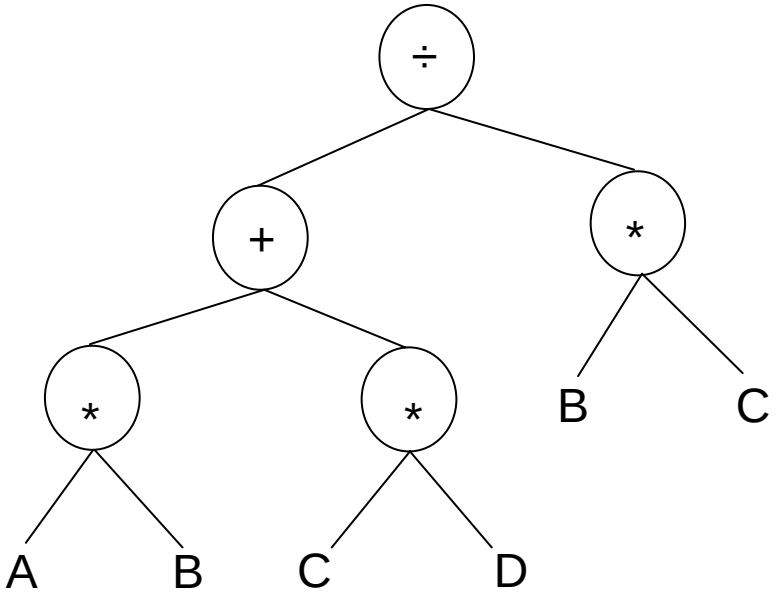
Esempio:

Data l'espressione : $\dfrac{(AB+CD)}{BC}$

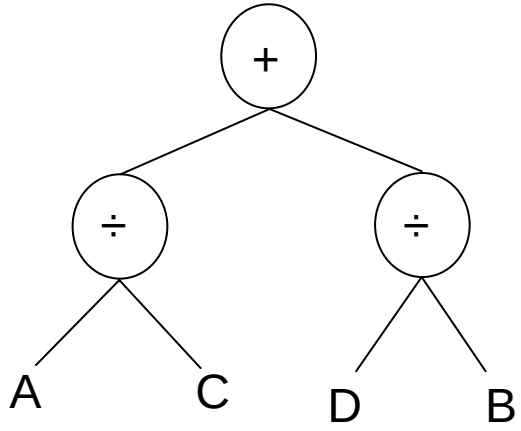
Si vuole ricavare l'espressione più semplice:

$$\frac{A}{C}+\frac{D}{B}$$

Una descrizione di $\dfrac{(AB+CD)}{BC}$ è :



Applicando le leggi dell'algebra (gli operatori dello spazio degli stati):



b) Descrizione mediante stringhe:

$\dfrac{(AB+CD)}{BC}$ si scrive $/+\times AB\times CD\times BC$

dove /,+,× sono *operatori prefissi*.

Utilizzando delle *regole di riscrittura* delle stringhe si arriva a: $+/AC/DB$

che corrisponde a: $\dfrac{A}{C}+\dfrac{D}{B}$

- ***Operatori***

➢ Dato uno stato generano uno (o più) stati successivi (come delle funzioni).

➢ In generale sono costituite da una computazione

➢ Nei casi semplici TABELLE

➢ Se le descrizioni sono stringhe *regole di riscrittura* della forma:

$$S_i \xrightarrow[produce]{} S_J$$

- *Esempi:*

  $A\$ \rightarrow B\$$

  Dove $\$ \Rightarrow$ qualsiasi stringa (anche vuota)

  Significato:

  Il simbolo A ad inizio stringa rimpiazzato da B (il resto della stringa è immodificato).

1. $A\$A \rightarrow A$ (una stringa che inizia e termina con A può essere rimpiazzata da una singola occorrenza di A);

2. $\$_1 BAB\$_2 \rightarrow \$_1 BB\$_2$ (una singola occorrenza di A tra due B può essere eliminata);

3. $\$_1\$_2\$_3 \rightarrow \$_1\$_2\$_2\$_3$ (ogni sottostringa può essere replicata);

4. $\$_1\$_2\$_2\$_3 \rightarrow \$_1\$_2\$_3$ (ogni ripetizione adiacente di una sottostringa può essere eliminata).

Per esempio, possiamo trasformare la stringa $ABCBABC$ nella stringa ABC usando le ultime due regole come segue:

$$\underbrace{AB}CBABC \xrightarrow{3} \underbrace{AB}\underbrace{AB}CBABC$$

Per esempio, possiamo trasformare la stringa *ABCBABC* nella stringa *ABC* usando le ultime due regole come segue:

$$ABCBABC \xrightarrow{3} A\underbrace{BABC}\underbrace{BABC} \xrightarrow{4}$$

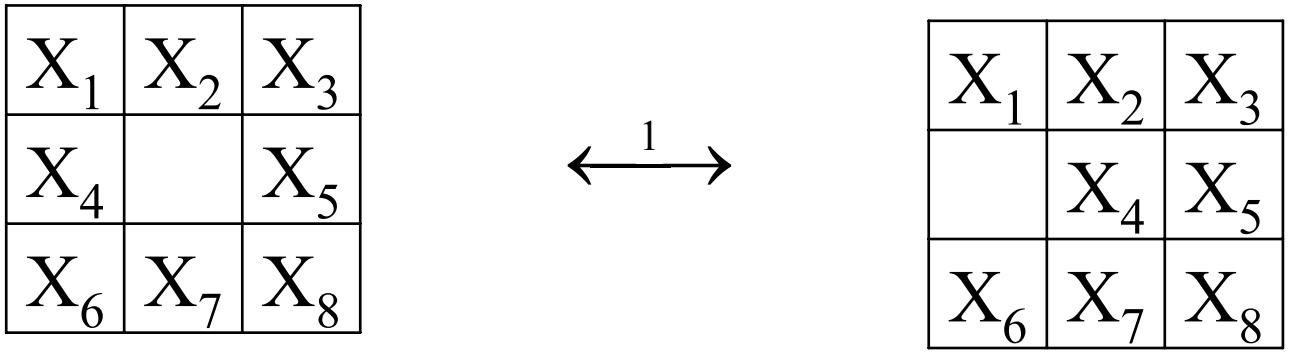
$$ABABC$$

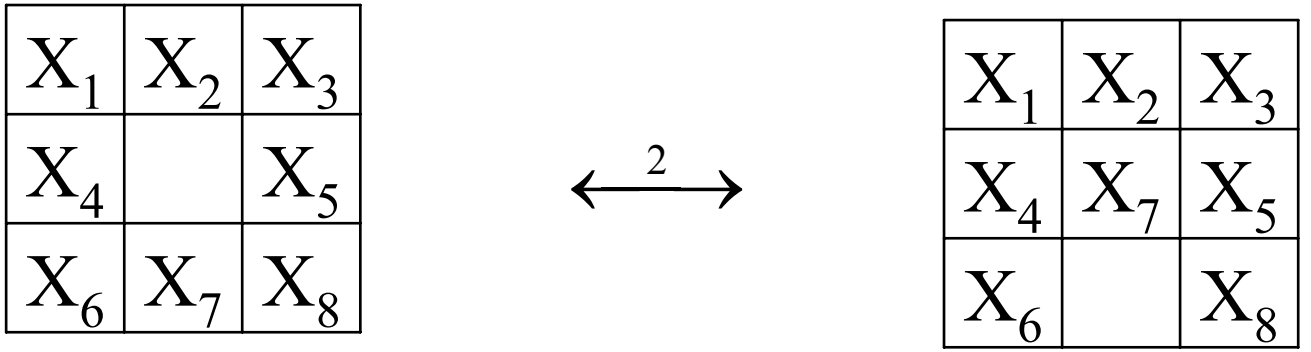
Per esempio, possiamo trasformare la stringa $ABCBABC$ nella stringa ABC usando le ultime due regole come segue:

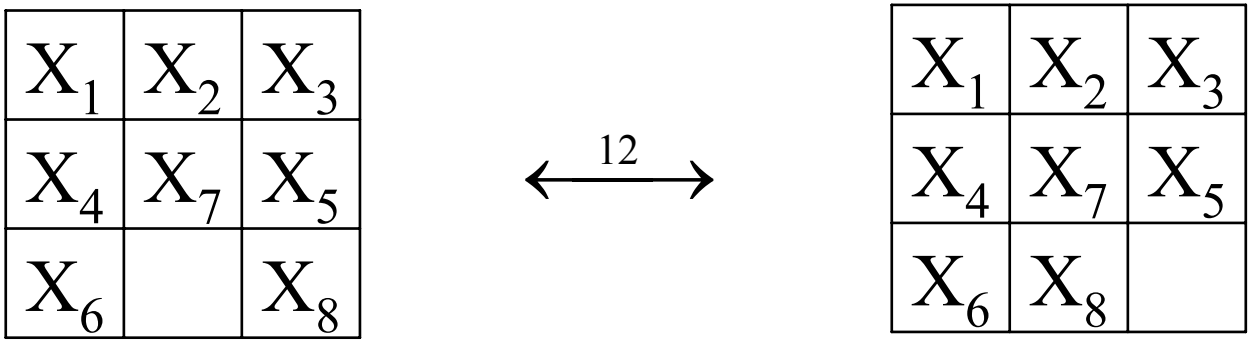
$$ABCBABC \xrightarrow{3} ABABCBABC \xrightarrow{4}$$

$$\underbrace{AB}\underbrace{AB}C \xrightarrow{4} ABC$$

Altro esempio: Regole di riscrittura per il gioco dell'otto:







NB: a sinistra la stessa stringa può comparire più volte (problema dell'*unificazione*).

- ***Stato finale***

  Bisogna descrivere con precisione le proprietà cui una descrizione di stato deve soddisfare per rappresentare uno stato finale.

Conclusione. Descrizione formale di un problema:

- Definire uno spazio di stati con tutte le configurazioni rilevanti
  - Si enumerano tutti gli stati (si consuma spazio)
  - Si cerca un algoritmo generativo di tutti gli stati. (N.B. a volte è utile generare configurazioni impossibili)
- Specificare uno o più stati iniziali
- Specificare uno o più stati finali (stati meta)
- Specificare un insieme di regole che descrivono le azioni (operatori)

Aspetti implicati:

- quali sono le assunzioni implicite presenti nella descrizione dei problemi
- quanta generalità vi è nelle regole
- quanto lavoro per la risoluzione del problema è già rappresentato dalle regole

Quali caratteristiche possiedono le tecniche specifiche dell'intelligenza artificiale?

Risposta: sono basate sulla conoscenza (ma lo sono anche le tecniche classiche!)

Proprietà specifiche di un approccio di intelligenza artificiale:

- Dar conto delle generalizzazioni (per esempio, raggruppare situazioni con proprietà comuni)

- La conoscenza dovrebbe essere compresa dalle persone che devono fornirla (programmi che apprendano o almeno si adattano)
- La conoscenza dovrebbe poter essere modificata facilmente (correzione di errori o adattamento a nuove situazioni)
- Tener conto di conoscenze incomplete

Esempi di tecniche ad hoc (approccio classico):
Filetto n. 1

1	2	3
4	5	6
7	8	9

0 = quadrato vuoto
1 = segno X
2 = segno O

Rappresentazione:
vettore di 3^9 =19683 elementi, ogni elemento è un vettore di nove elementi (numero in base 3)

Algoritmo:

- Si considera una (dis-)posizione, e si converte il numero in decimale
- Si usa il numero come entry nella tabella della posizione attuale, in cui si trova il codice del nuovo stato

Commenti:

- algoritmo semplice
- molto lavoro nel realizzare la tabella (alta possibilità di errore)
- struttura rigida (per esempio, non estensibile a tre dimensioni)

Filetto n. 2

Si assegnano le posizioni secondo lo schema del quadrato magico (somma=15)

8	3	4
1	5	9
6	7	2

Algoritmo:

- si considera ogni coppia di quadrati tenuti da un giocatore
- si calcola 15- <somma dei due quadrati>
- Se differenza <0 oppure >9, si ignora la coppia (quadrati non in linea). Altrimenti se il quadrato con il valore differenza è vuoto lo si occupa (vittoria).

Commento:

- maggiore efficienza
- molta “conoscenza specifica” inserita nella base dati (è precalcolata)

Approccio di Intelligenza Artificiale

- Individuare una rappresentazione del quadrato (OK matrice)
- Individuare una regola per generare, da una posizione, tutte le mosse possibili
- Trovare un modo per “valutare” ciascuna delle mosse possibili
- Individuare una “strategia di gioco” (per esempio, tener conto della prossima mossa o di due successive, ecc.), considerando anche le reazioni dell’avversario.

Riassumendo:

FORMULAZIONE DI UN PROBLEMA NELLO SPAZIO DEGLI STATI

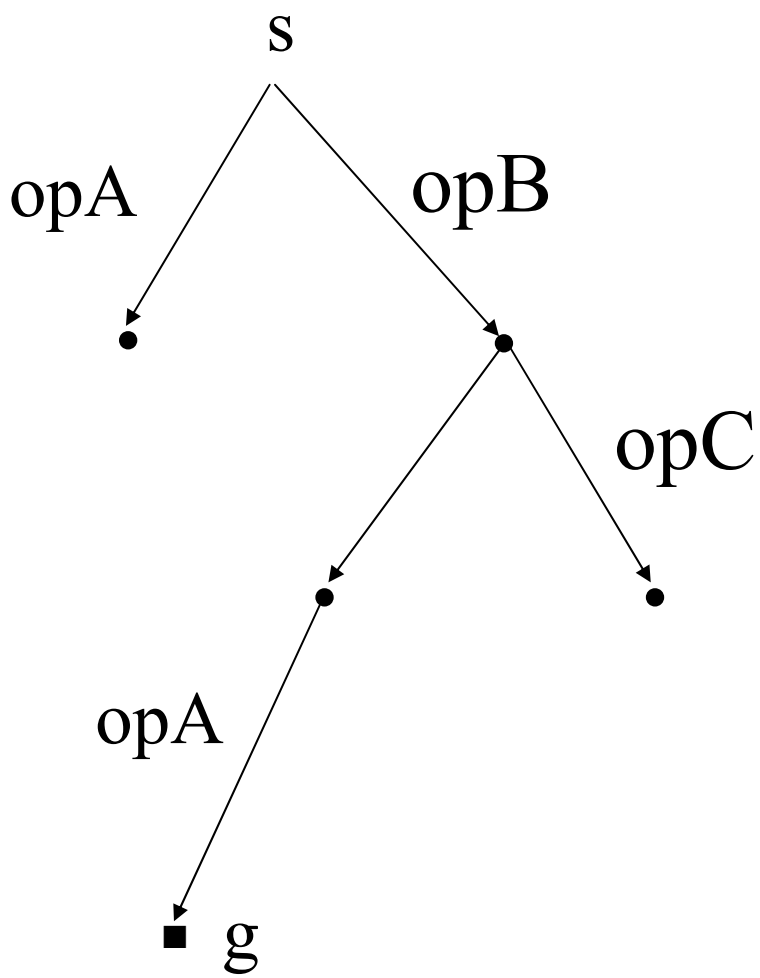
- S: insieme degli stati
- O: S → S insieme degli operatori

  (funzioni parziali nello spazio degli stati) che specificano le trasformazioni da uno stato ad un altro. Sono applicabili solo a certi stati.
- $i \in S$ stato iniziale
- $g \in S$ stato finale (GOAL)

  (ci può essere più di uno stato finale)
- SOLUZIONE: sequenza di operatori o1, o2, …, on tale che

  g = on(…o2(o1(i))…)

Rappresentazione grafica



La soluzione è opB, opB, opA

IL PUZZLE DELL'8

STATI: configurazioni delle tessere

2	8	3
1	6	4
7		5

SPAZIO DEGLI STATI: insieme di tutte le possibili configurazioni

Il numero di possibili stati è 9! = 362.880

1	2	3
8		4
7	6	5

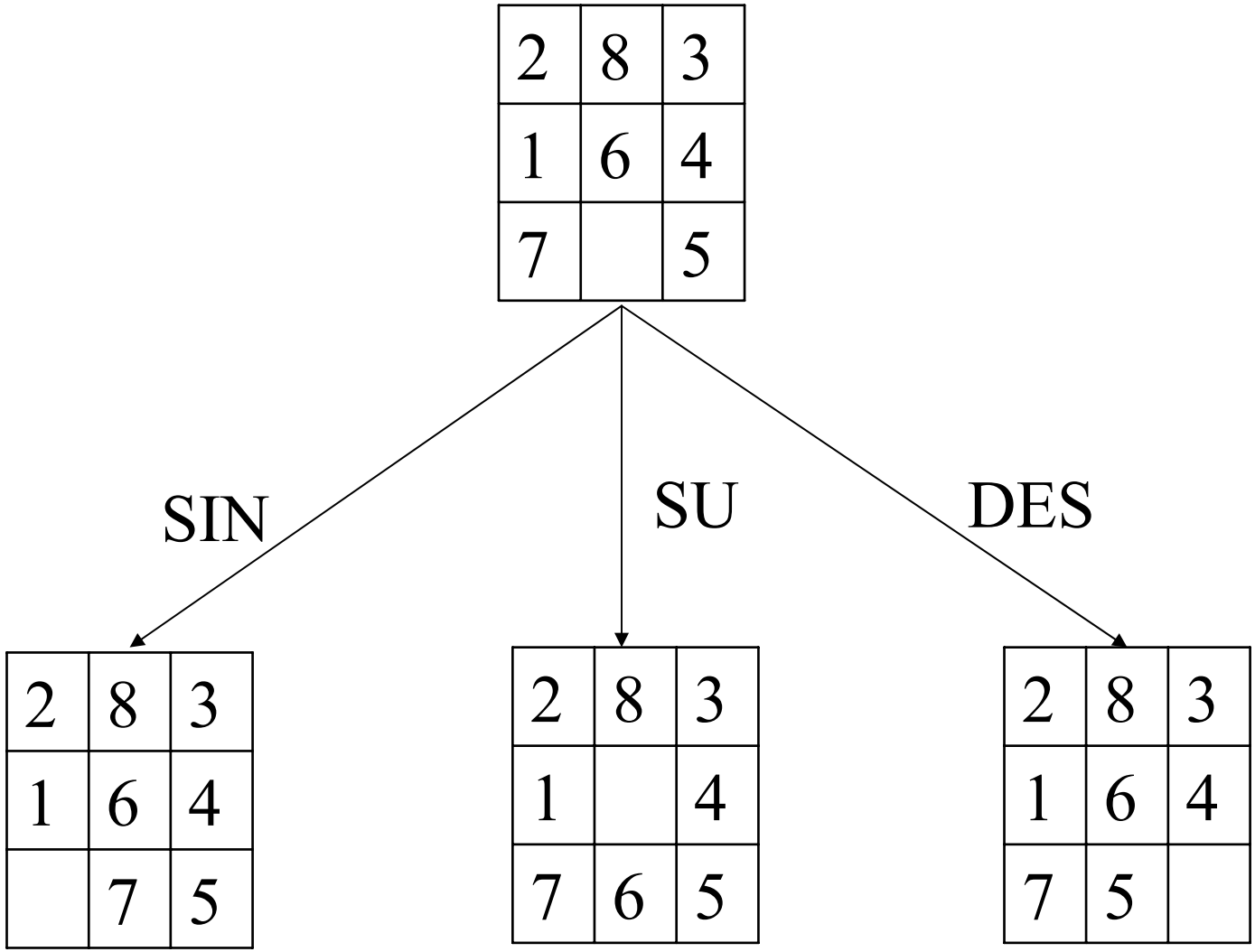
STATO FINALE (GOAL)

OPERAZIONI: mosse

4 mosse per i 4 possibili spostamenti del posto vuoto:

SU muovere il vuoto un posto in su
GIU " " " giù
SIN " " " a sinistra
DES " " " a destra

Non sempre tutti gli operatori sono applicabili (un operatore è una funzione parziale nello spazio degli stati)


2 8 3
1 6 4
7 5
SIN
SU
DES
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
2 8 3
1 6 4
7 5

Grafi

- Un grafo è un insieme di nodi.
- Alcune coppie di nodi sono connesse da archi diretti.
- Se un arco è diretto dal nodo *n* al nodo *m*, *m* è detto un successore del nodo *n* e *n* è un genitore del nodo *m*.
- Un albero è un caso particolare di grafo in cui ogni nodo ha al massimo un genitore.
- Un solo nodo non ha genitori: la radice.
- Un nodo senza successori è un nodo foglia.

- Un cammino è una sequenza di nodi n_1, n_2, …, n_k tale che ogni n_i è successore di n_{i-1}.
- Se fra i nodi *n* e *m* c'è un cammino, *m* è un discendente di *n* e *n* è un antenato di *m*.

- Problema: trovare un cammino tra il nodo iniziale *s* e un nodo appartenente all'insieme dei nodi goal.

- Un grafo si può assegnare:
  - in modo esplicito, elencando i nodi, gli archi ed eventuali costi associati. Scomodo per grafi di grafi di grandi dimensioni;
  - in modo implicito: si assegna un insieme $\{s_i\}$ di nodi di *partenza* e si dà un *operatore di successione* Γ che, applicato ad un nodo, fornisce *tutti* i successori. Quando serve, si rende esplicita una porzione di grafo definito implicitamente da Γ e $\{s_i\}$.

Alberi e grafi

ALBERI

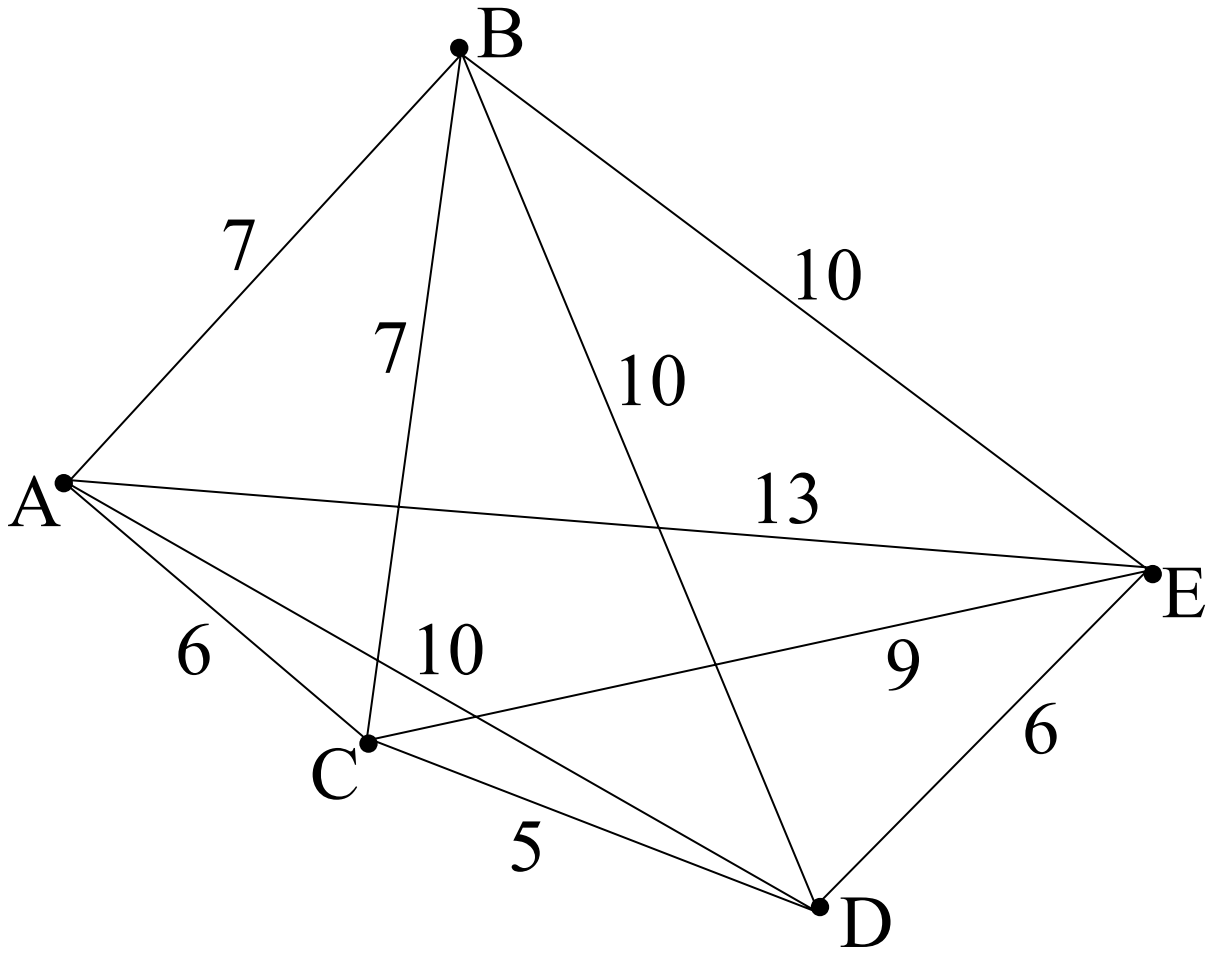
vantaggi	procedure più semplici;
svantaggi	lo stesso nodo può essere generato più volte portando alla duplicazione di passi di ricerca.

GRAFI

vantaggi	nodi non duplicati;
svantaggi	costoso verificare se un nodo era stato già generato; procedure più complesse; il grafo può contenere cicli e quindi bisogna dimostrare che l'algoritmo di ricerca termina.

PROBLEMA DEL COMMESSO VIAGGIATORE

Visitare le 5 città A, B, C, D, E partendo da A e tornando ad A, passando per ogni altra città una sola volta, con un percorso di lunghezza minima.



STATI: percorsi parziali (sequenze di nomi di città).

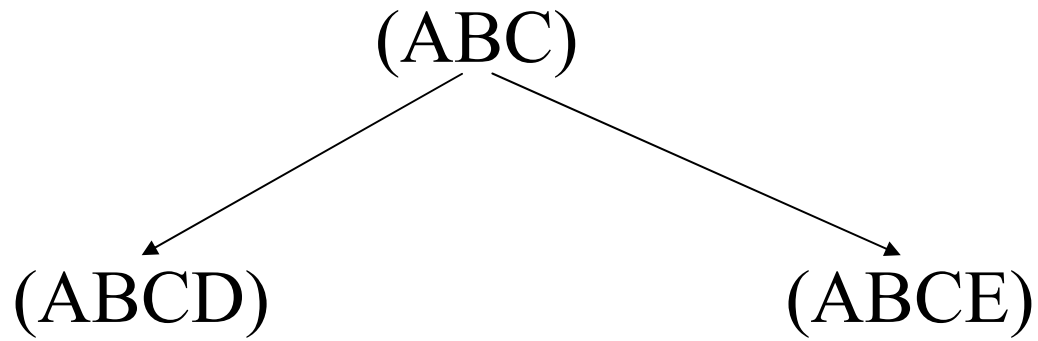
Es. (A) (ACB) (ACDE) (ACDEBA)

↑ stato iniziale

↑ goal

Ci sono molti stati finali

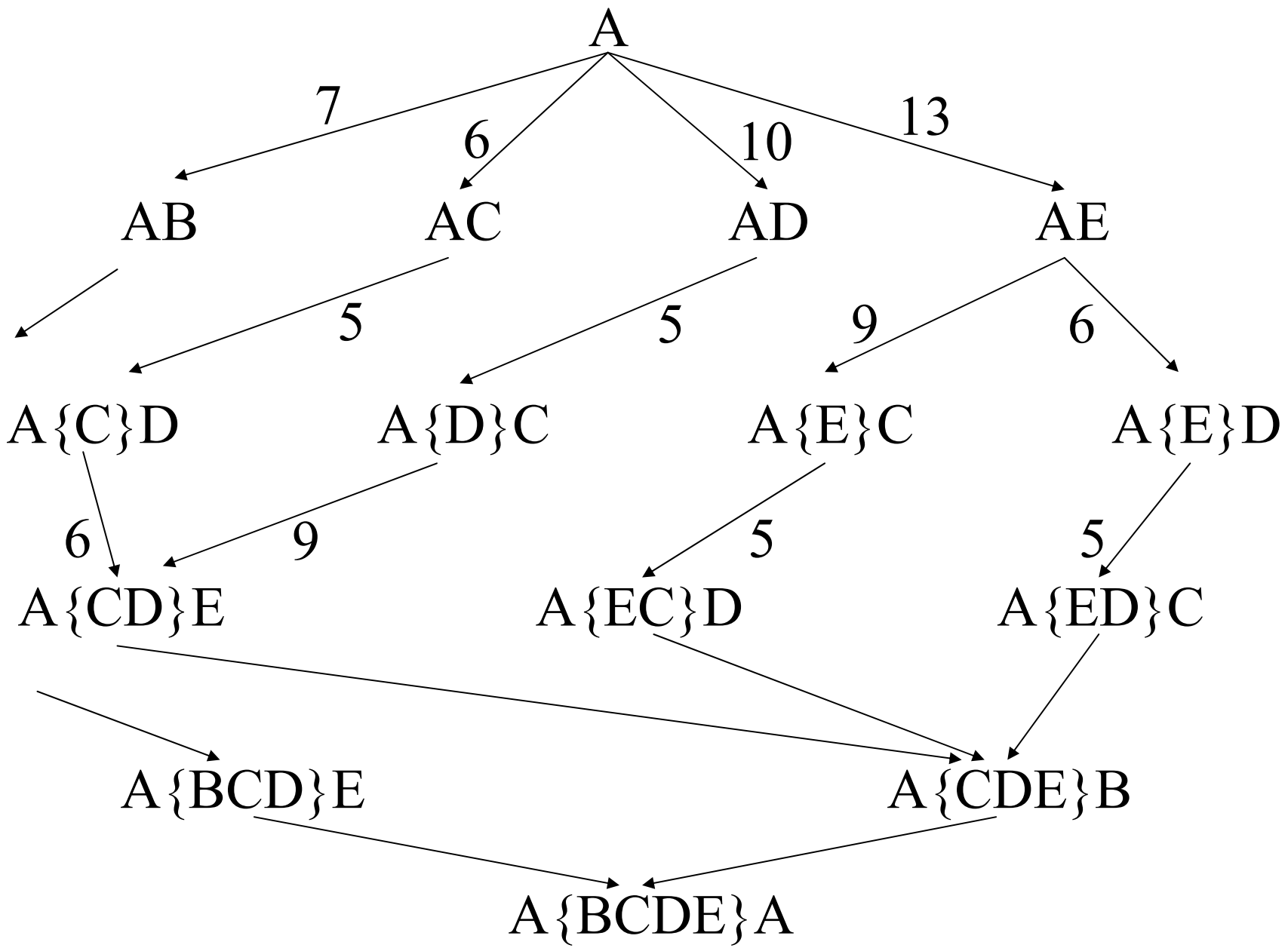
OPERAZIONI: andare alla prossima città



Più in generale, lo spazio di ricerca è un GRAFO, ossia uno stato può essere raggiunto da più stati.

Altra rappresentazione del commesso viaggiatore:

lo stato A{BC}D rappresenta un viaggio da A a D passando attraverso B e C in un ordine qualsiasi.



C’è un solo stato goal

Altro esempio: analisi sintattica.

È data una *grammatica* che definisce una certa classe di stringhe di simboli. Ci si chiede se una stringa appartiene o no alla classe.

Grammatica: una *frase* è definita come:

1. il simbolo *a* seguito dal simbolo *b*
2. il simbolo *a* seguito da una *frase*
3. una *frase* seguita dal simbolo *b*
4. una *frase* seguita da un'altra *frase*

Esempi di *frasi*:

aab, *abaabab*, *aaaaab*

non sono *frasi*:

aaa, *aba*, *abaa*

Formulazione nello spazio degli stati:

- descrizione degli stati: le stringhe stesse. Si assume la stringa da verificare (ad es. *abaabab*) come stato iniziale;
- operatori: regole di riscrittura:

  $\$_1 ab \$_2 \rightarrow \$_1 S \$_2$ (la sottostringa *ab* può essere rimpiazzata dal nome *S* che indica una frase)

  $\$_1 aS \$_2 \rightarrow \$_1 S \$_2$

  $\$_1 Sb \$_2 \rightarrow \$_1 S \$_2$

  $\$_1 SS \$_2 \rightarrow \$_1 S \$_2$

Definizione dell'obiettivo: lo stato finale è descritto da una stringa costituita dal solo simbolo *S*.

Una successione di stati che rappresenta una soluzione del problema è allora la seguente:

abaabab

Saabab

SaSab

SSab

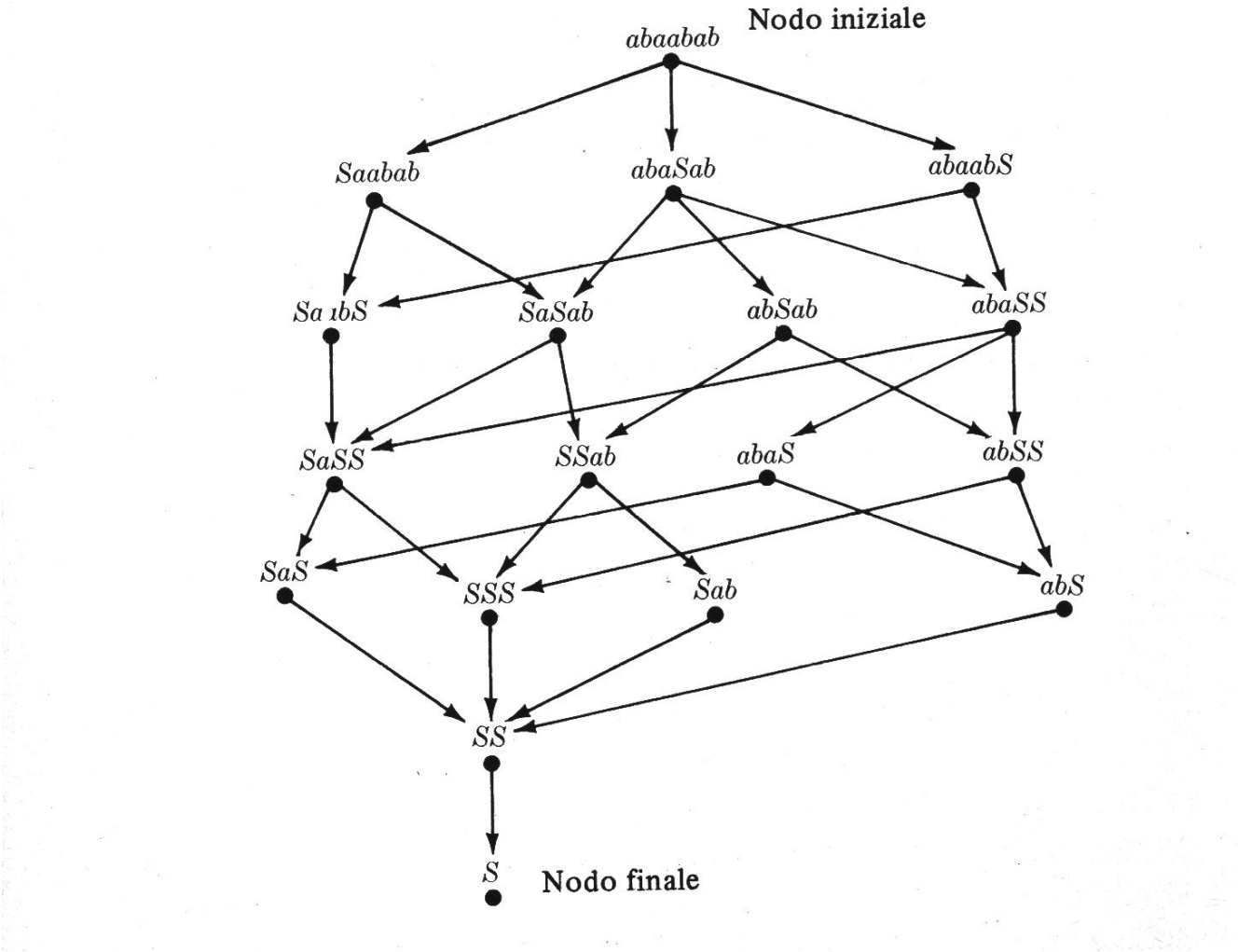
SSS

SS

S

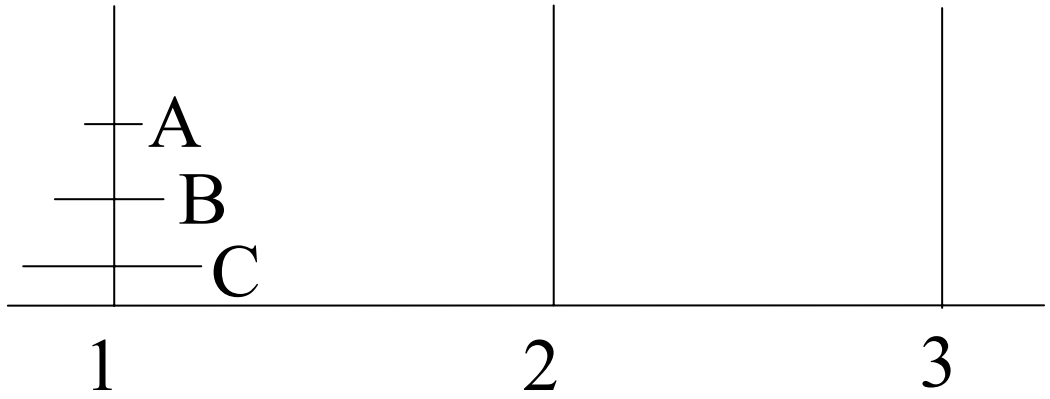
Il problema è descritto dal seguente grafo:

Fig. 2.6 – Grafo del problema di analisi sintattica



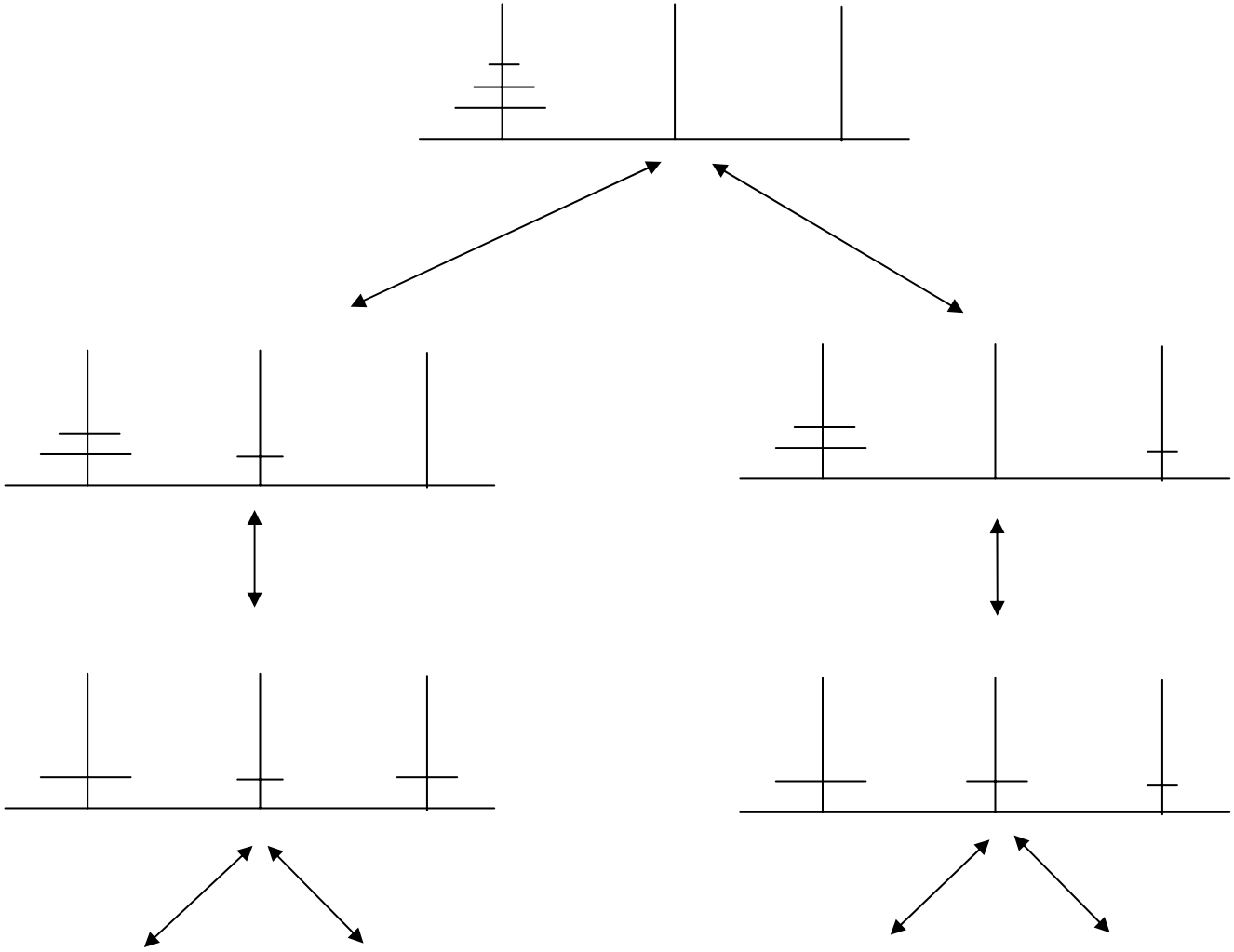
N.B. : osservando che dalla grammatica si deduce che sono frasi le stringhe che iniziano con *a* e terminano con *b*, si poteva ridurre lo sforzo di ricerca (*hindsight*).

Ancora un esempio: la torre di Hanoi



Trasferire la torre dal piolo 1 al piolo 3 usando il piolo 2 come supporto, muovendo un disco alla volta e con il vincolo che un disco di diametro superiore non può mai essere messo sopra un disco di diametro inferiore.

Graficamente, applicando ad ogni stato tutte le mosse possibili a partire dallo stato iniziale, si ottiene il grafo seguente:



Una rappresentazione più sintetica (ed efficiente) è la seguente:

Stato = (i j k)

dove:

i : posizione del disco C (il maggiore)

j : posizione del disco B (il medio)

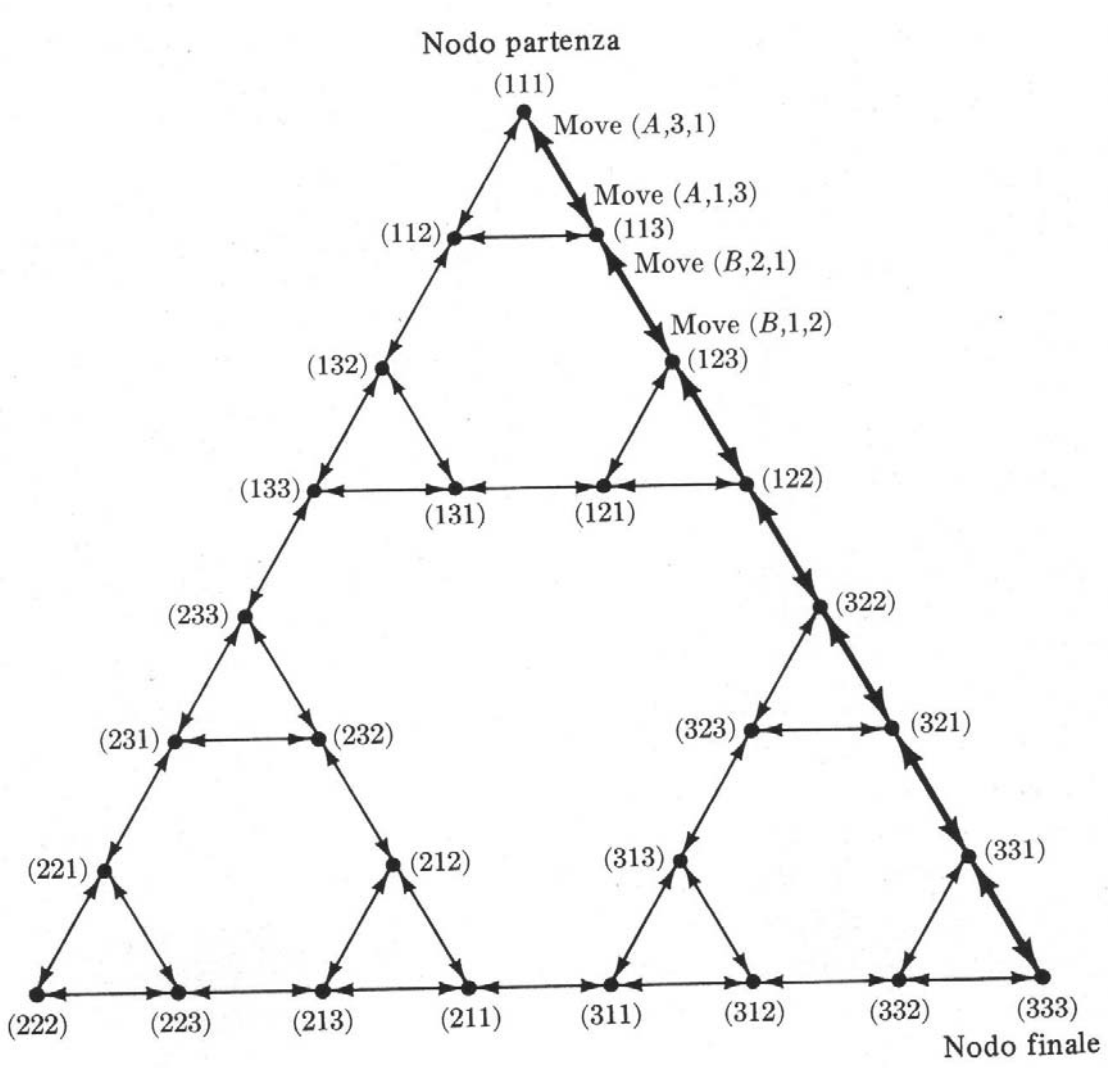
k : posizione del disco A (il minore)

Operatori: MOVE (X, m, n)

Esempio: MOVE (A, 3, 1)

Muove il disco A dal piolo 3 al piolo 1 (e viceversa: l'operatore è bidirezionale).

Il problema è allora così rappresentabile:



Scelta di una “buona” rappresentazione

La scelta della rappresentazione influenza lo sforzo di ricerca.

Meglio rappresentazioni con piccoli spazi degli stati.

A volte è opportuno utilizzare conoscenze in più, per esempio,

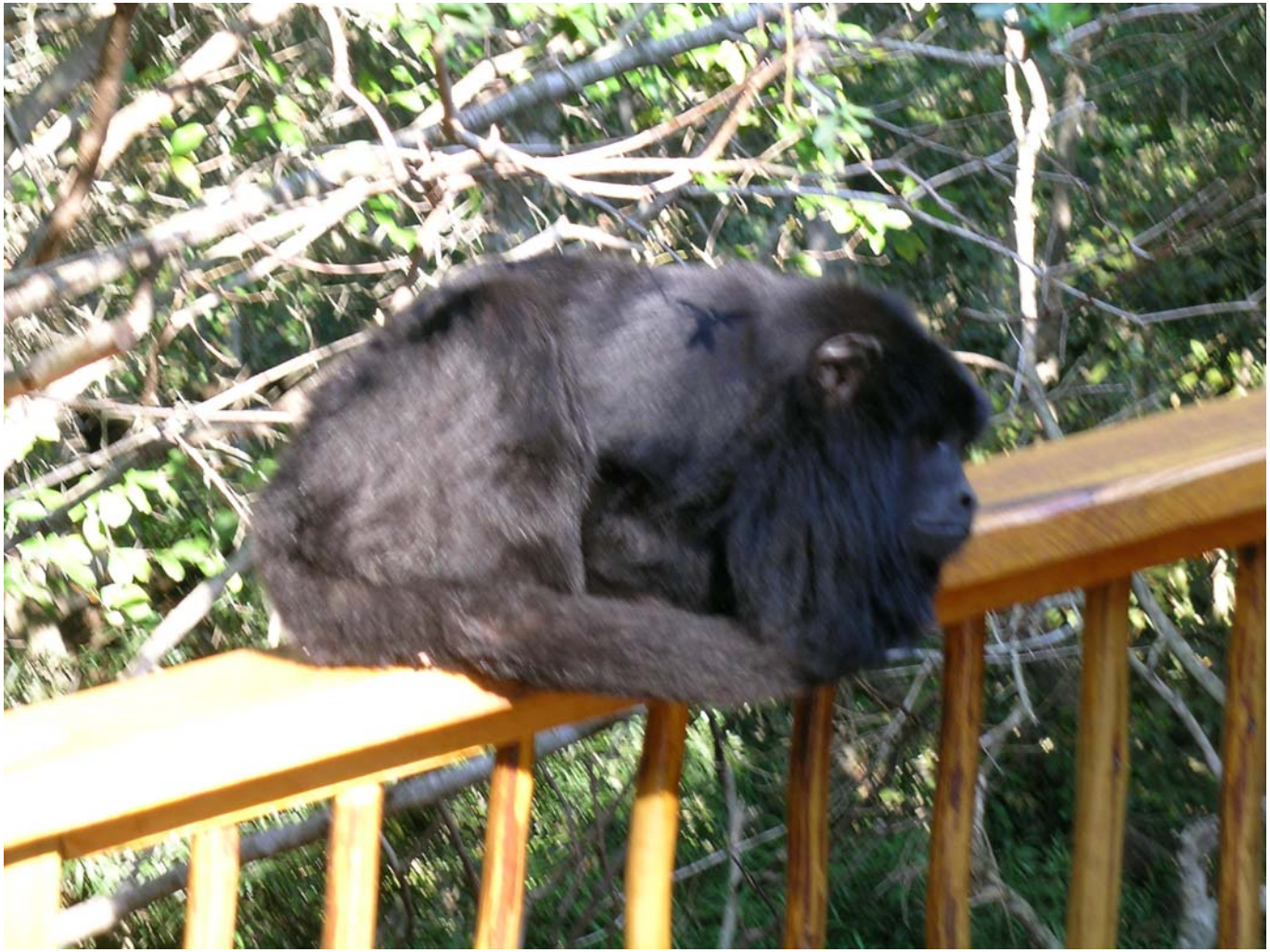
- per riconoscere concetti semplificanti (simmetrie, analogie, ecc.)
- per formare macrooperatori

A volte è utile utilizzare variabili nella descrizione degli stati.

Esempio: *problema della scimmia e delle banane*.

Formulazione: in una stanza c'è una scimmia, una cassa e un casco di banane appeso in alto.

Obiettivo: la scimmia mangia le banane



Non è un mio problema!

Descrizione:

- posizione della scimmia nella stanza (in uno spazio bidimensionale)
- flag scimmia sopra / sotto la cassa
- posizione della cassa (nello spazio bidimensionale)
- flag ha presso / non ha preso banane

Lo stato, quindi, è una lista di 4 elementi (**w**, x, **y**, z) dove:

1. **w** = posizione orizzontale della scimmia (vettore bidimensionale;
2. x = 1 o 0, a seconda che la scimmia si trovi rispettivamente sulla cassa o a terra;
3. **y** = posizione orizzontale della cassa (vettore bidimensionale);
4. z = 1 o 0, a seconda che la scimmia rispettivamente abbia o non abbia preso le banane;

Invece di semplici operatori, si definiscono schemi di operatori, cioè degli operatori con parametro:

1. goto (**u**) la scimmia si porta alla posizione orizzontale **u** (variabile);
2. pushbox (**v**) la scimmia spinge la cassa alla posizione orizzontale **v** (variabile);
3. climbox la scimmia sale sulla cassa;
4. grasp la scimmia afferra le banane.

A causa della presenza delle variabili in goto e pushbox, questi operatori sono in effetti schemi di operatori.

Le condizioni di applicazione e gli effetti degli operatori sono dati dalle seguenti regole di riscrittura:

(**w**, 0, **y**, z) $\xrightarrow{goto(u)}$ (**u**,0, **y**, z)

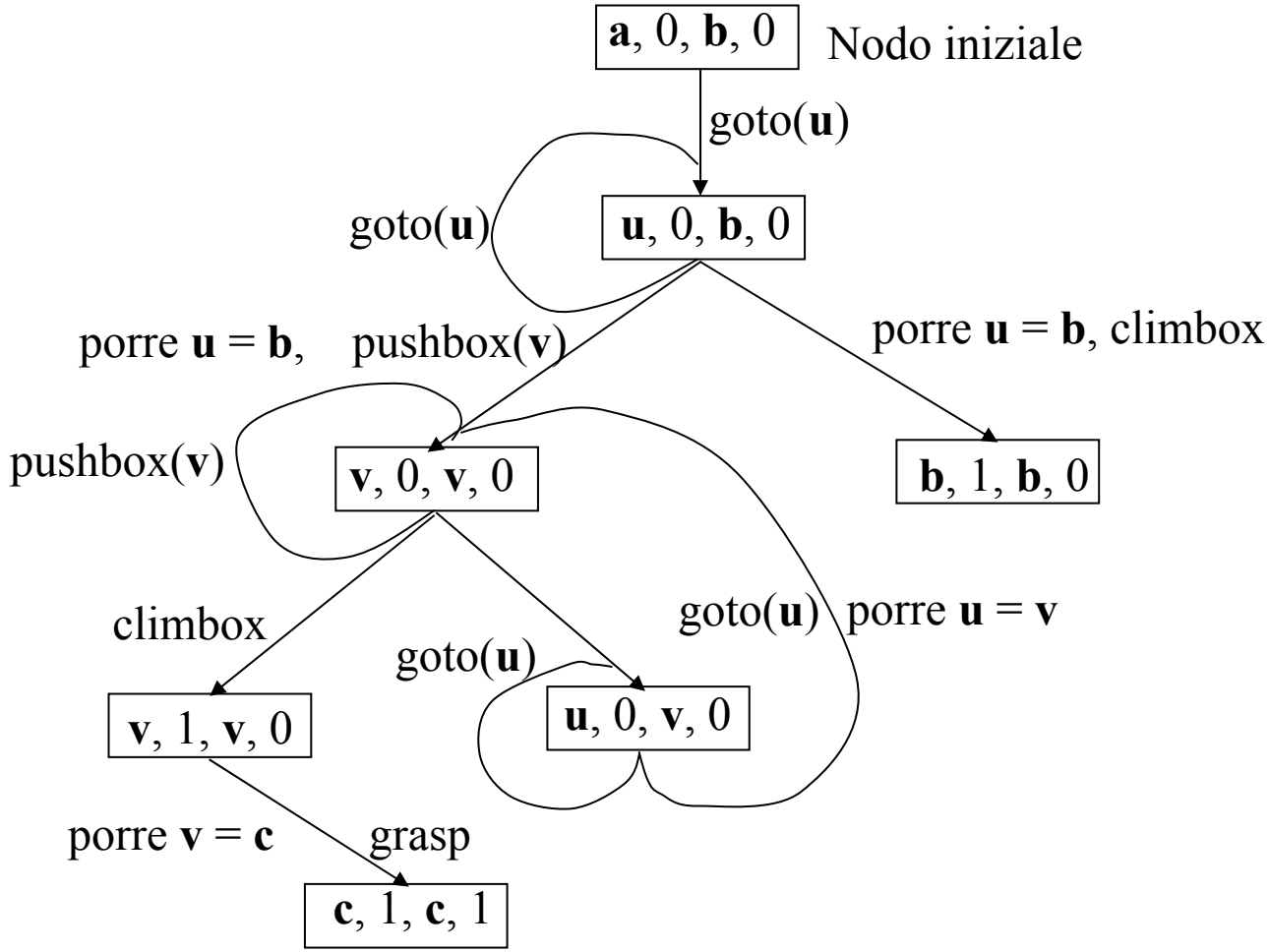
(**w**, 0, **w**, z) $\xrightarrow{pushbox(v)}$ (**v**,0, **v**, z)

(**w**, 0, **w**, z) $\xrightarrow{climbox}$ (**w**,1, **w**, z)

(**c**, 1, **c**, z) $\xrightarrow{grasp}$ (**c**,1, **c**, 1)

Dove **c** = posizione orizzontale banane

Il grafo che descrive il problema della scimmia e delle banane è il seguente:



Strategia di controllo

Caratteristiche di una buona strategia di controllo:

- deve causare movimento (in pratica, deve portare verso la soluzione e non fare eseguire sempre gli stessi passi in modo inconcludente). Problema: identificare situazioni critiche
- deve essere sistematica (ciò corrisponde all'esigenza sia di movimento globale, nel corso di vari passi, sia di movimento locale, nel corso di un singolo passo). Esempi
  - ricerca in ampiezza
  - ricerca in profondità
  - ecc.

Ragionamento in avanti e all'indietro

- In avanti (forward): dagli stati iniziali verso le mete
- All'indietro (backward): dalle mete verso gli stati iniziali

Sono date delle regole di generazione (parte sinistra $\xrightarrow{produce}$ parte destra), per cui, dato uno stato, si applica la regola per produrre un altro stato.

Ragionamento in avanti:

- si costruisce l'albero la cui radice rappresenta lo stato iniziale

- si cerca la regola (o le regole) la cui parte sinistra corrisponde al nodo e si generano i nodi corrispondenti alle parti destre (fino allo/agli stato/i meta)

Ragionamento all'indietro:

- Si costruisce l'albero la cui radice corrisponde allo stato meta

- Si cercano le regole la cui parte destra corrispondono al nodo e si generano i nodi corrispondenti alle parti sinistre (fino allo/agli stato/i iniziale/i)

Criteri per scegliere forward o backward

- Confronto tra numero di stati iniziali e finali (ovvero qual è più facile da verificare)
- Direzione in cui si verifica un maggior ***fattore di ramificazione***
- Se occorre giustificare il ragionamento, procedere nella direzione che corrisponde al modo in cui pensa l'utente

Rappresentazione della conoscenza

Verranno illustrati molti metodi. Schematicamente si suddividono in:

- rappresentazione implicita
- rappresentazione esplicita

Esempio di descrizione del mondo in cui opera un robot:

SU (pianta, tavolo)

SOTTO (tavolo, finestra)

IN (tavolo, stanza)

Ecc.

In generale occorre rappresentare:

- collezione di oggetti
- collezione di attributi (proprietà degli oggetti)
- insiemi di relazioni (tra gli oggetti)

Problema connesso:

rappresentare un mondo complesso in cui ci sono cose che cambiano e cose che non cambiano (*problema del contorno*, *frame problem*)

- Possibili soluzioni:

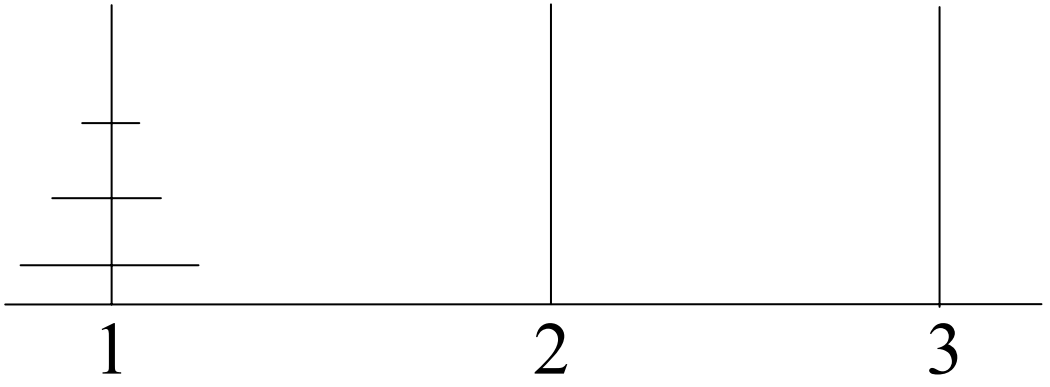
➢ si tiene traccia solo dei cambiamenti (lo stato di partenza è descritto in modo completo)

➢ si modifica lo stato iniziale con operatori “invertibili” in modo che si possa tornare indietro “annullando” i passi effettuati.

Altra tecnica di rappresentazione: RIDUZIONE DI PROBLEMI

La soluzione di un problema viene ridotta alla soluzione di uno o più sottoproblemi più semplici.

Esempio della torre di Hanoi



Per spostare la torre alta *n* dal piolo *i* a *k*:

a) spostare una torre alta *n-1* da *i* a *j*
b) spostare una torre alta 1 da *i* a *k*
c) spostare una torre alta *n-1* da *j* a *k*

(*j* rappresenta il secondo piolo)

Formalizzazione della RIDUZIONE DI PROBLEMI

P: insieme dei problemi

O:$p^n \rightarrow$ P insieme degli operatori che specificano come risolvere un problema dati n problemi risolti

G $\in$P goal - problema da risolvere

SP $\subset$ P insieme dei problemi risolti

SOLUZIONE: un insieme di operatori che, applicati ai problemi risolti, portino ad ottenere g.

GRAFI AND/OR

Un grafo AND/OR è un insieme di *nodi*.

Alcune n-uple di nodi sono connesse da *connettori*.

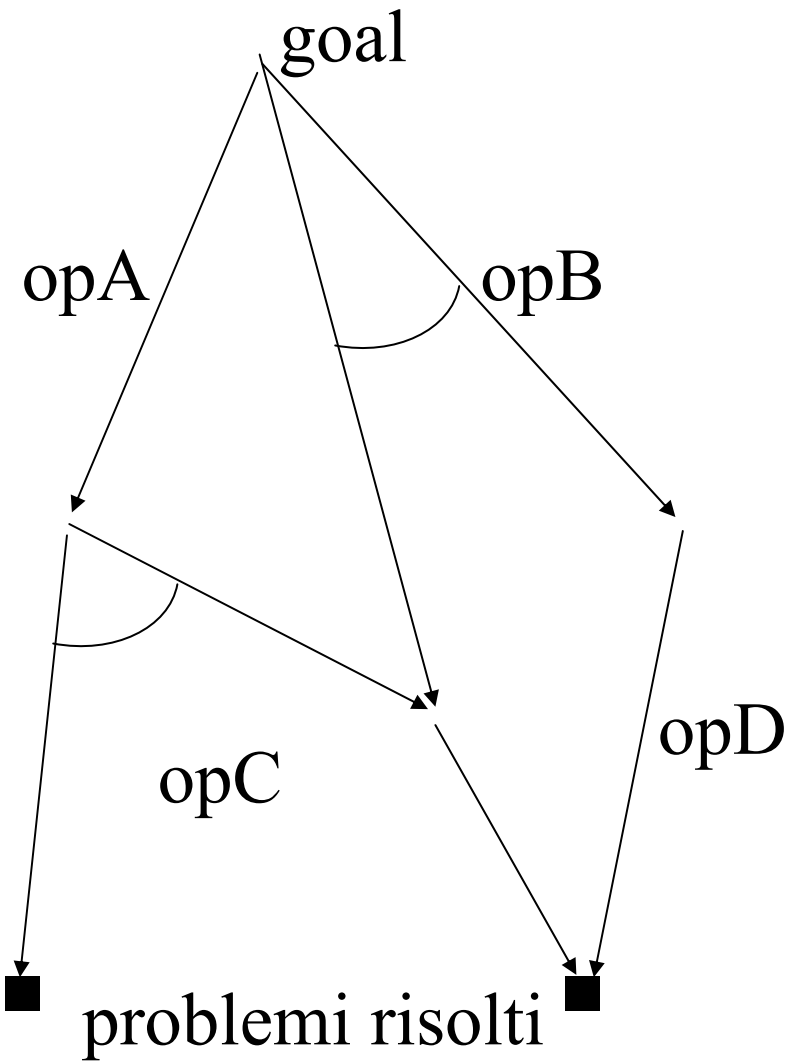
Dato un k-connettore $(n_0, n_1, \ldots, n_k)$ n_0 è il *genitore* di $n_1 \ldots n_k$ che sono i *successori*.

Ci sono un *nodo iniziale* e un insieme di *nodi terminali*.

Si possono definire anche *alberi AND/OR*.

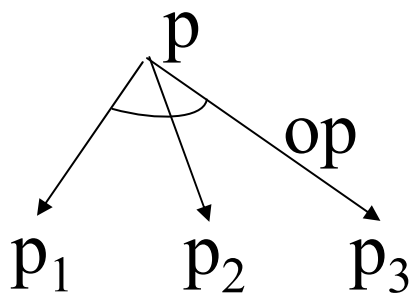
Un albero *AND* è la generalizzazione di un cammino in un grafo ordinario. Ogni nodo, tranne il nodo radice, compare esattamente due volte, una volta come input e una volta come output di qualche connettore.

È possibile dare una rappresentazione grafica della riduzione di problemi mediante un GRAFO AND/OR:

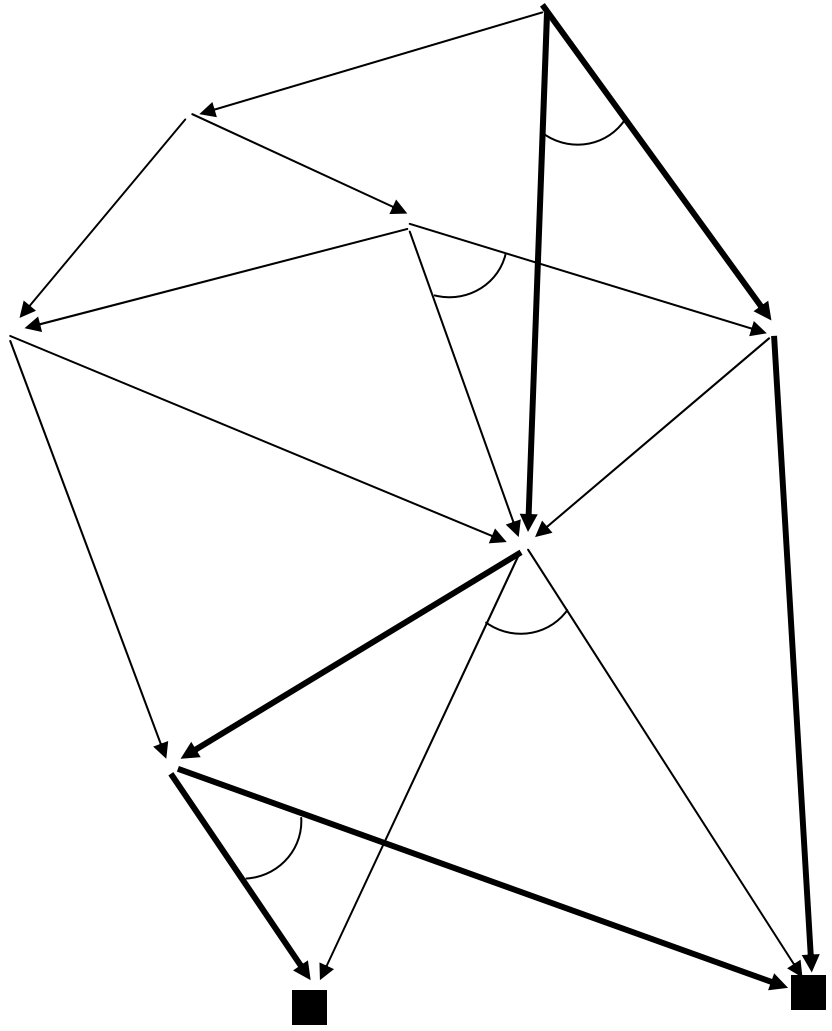


Gli operatori con più di un argomento si rappresentano con archi generalizzati: connettori.

Ad esempio, l'operatore op ci dà la soluzione di p, date le soluzioni di p_1, p_2 e p_3.



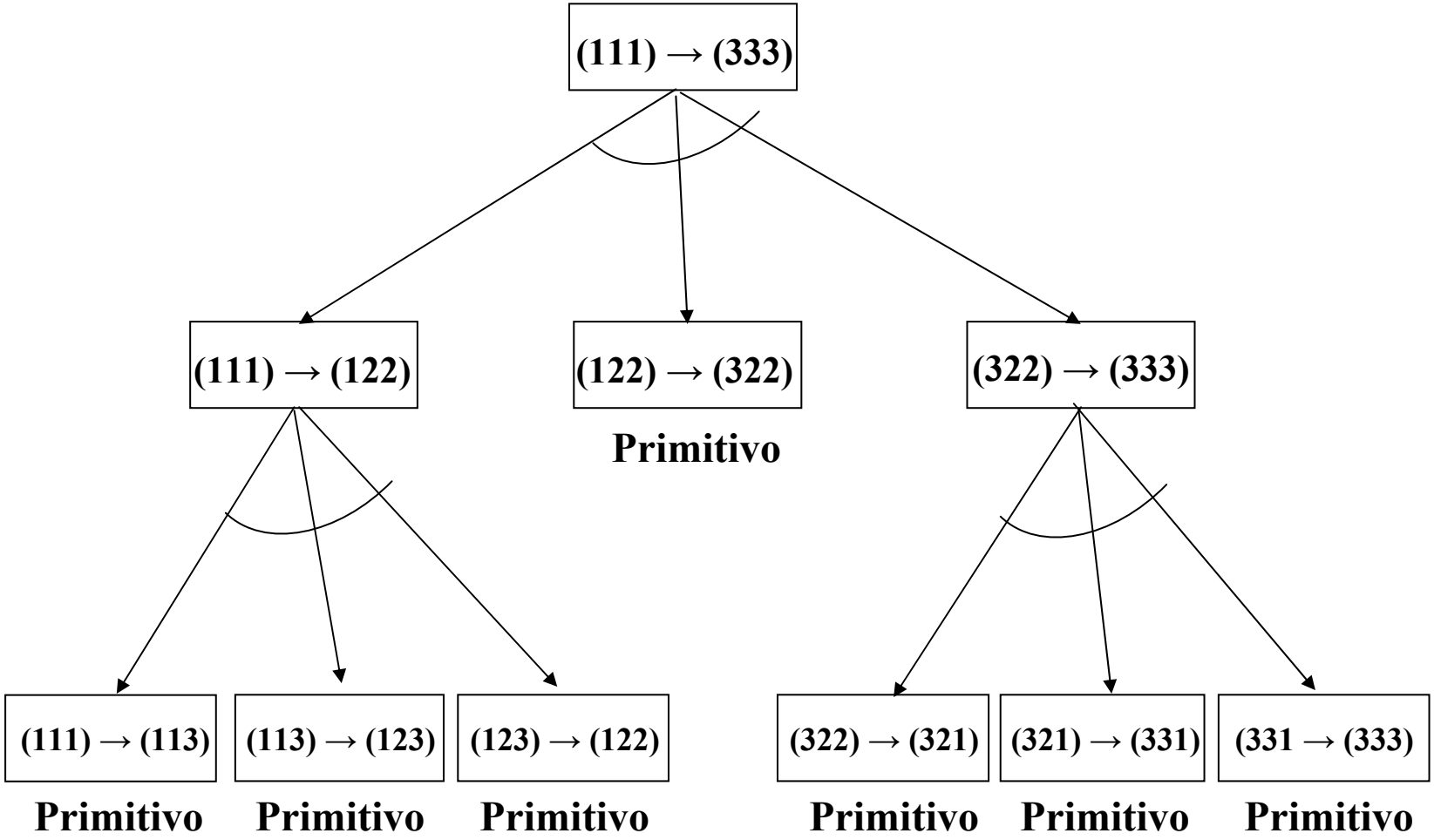
La soluzione in un grafo AND/OR generalizza il concetto di cammino in un grafo normale.



Le linee spesse danno una possibile soluzione.

Esempio: la torre di Hanoi costituita da 3 dischi.

Lo stato è rappresentato dalla terna (*i j k*), dove *i*, *j* e *k* indicano il numero del piolo in cui si trovano rispettivamente il disco di diametro maggiore, intermedio e minore.

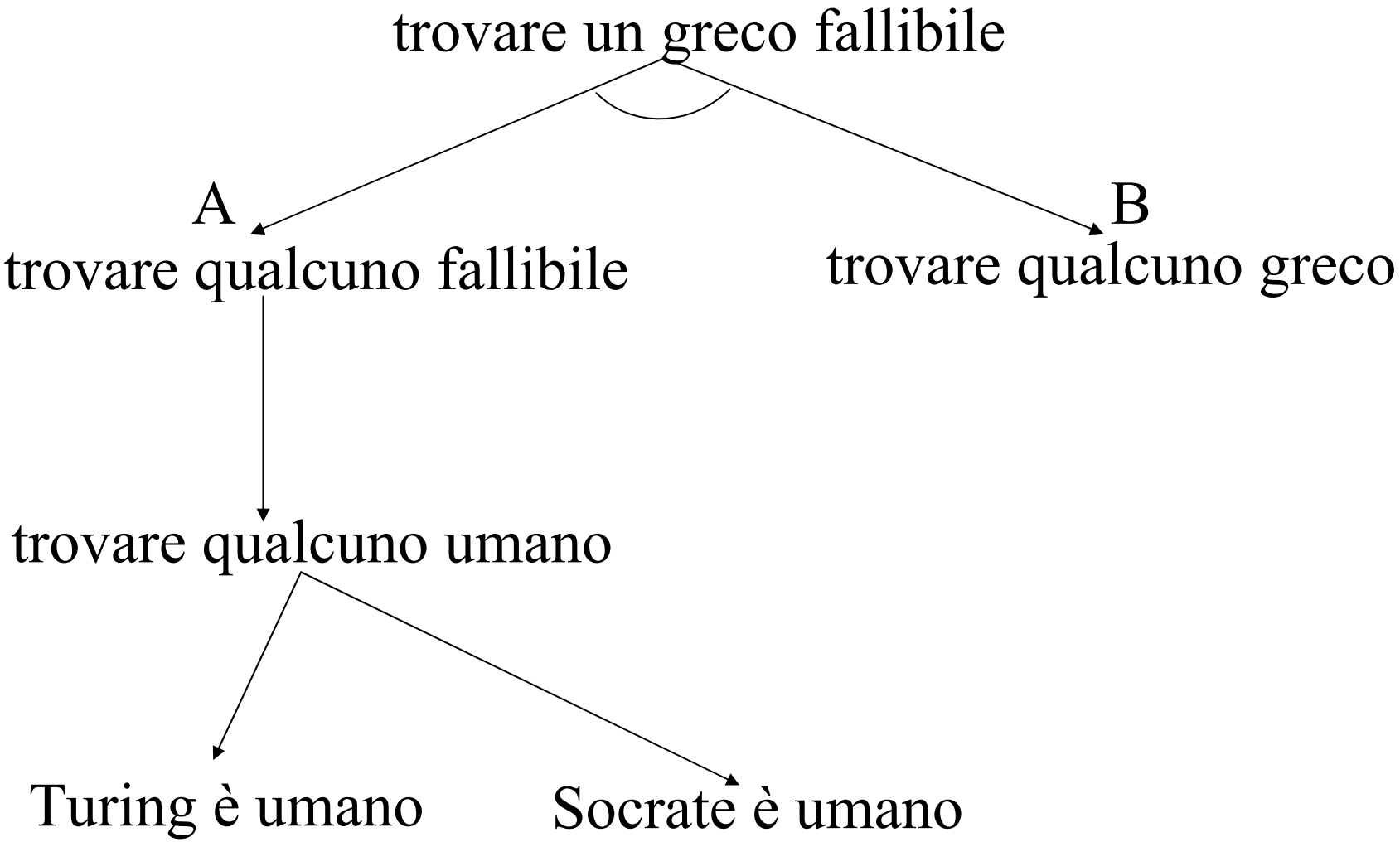


La soluzione è deterministica (non ci sono nodi OR).

PROBLEMI NON INDIPENDENTI

La definizione di soluzione per un grafo AND/OR richiede che i sottoproblemi in cui un problema può essere scomposto siano indipendenti fra loro.

Non sempre questo è vero.



Il "qualcuno" nei due sottoproblemi A e B deve essere la stessa persona.

REGIMI DI CONTROLLO

CONTROLLO: stabilire l'ordine di applicazione delle operazioni per raggiungere la soluzione.

STRATEGIE DI RICERCA

PARALLELE

SEQUENZIALI

REGIMI DI CONTROLLO

IRREVOCABILE: si sceglie una regola senza poter revocare la decisione presa.

BACKTRACKING: se ci si accorge di aver sbagliato si può tornare indietro e fare una scelta diversa.

RICERCA SU GRAFI: la ricerca della soluzione viene fatta sull'intero grafo (o albero) degli stati.

Esempi

- Si sta provando un teorema. Si decide di provare un lemma, che risulta inutile.

  Si elimina semplicemente il lemma (si revoca un passo) : la base di conoscenza preesistente resta valida (struttura di controllo semplice, senza ritorno all'indietro).
- Gioco dell'otto: si effettua lo spostamento di una tessera e ci si accorge che non è un passo buono: si può tornare indietro effettuando un passo in più (struttura di controllo basata su stack).

- Gioco degli scacchi: si studia (e si effettua) una mossa. Ci si accorge solo successivamente che la mossa era dannosa, ma il passo non è annullabile (regime irrevocabile).

  Questi problemi richiedono maggiore studio prima di effettuare una mossa.