FONDAMENTI DI INTELLIGENZA ARTIFICIALE – M (9 CFU)

2 Febbraio 2011

Tempo a disposizione 3 h – Risultato 32/32 punti

Esercizio 1 (punti 5)

Sapendo che:

Tutti gli studenti sono in grado di risolvere alcuni problemi e non sono in grado di risolvere alcuni problemi.

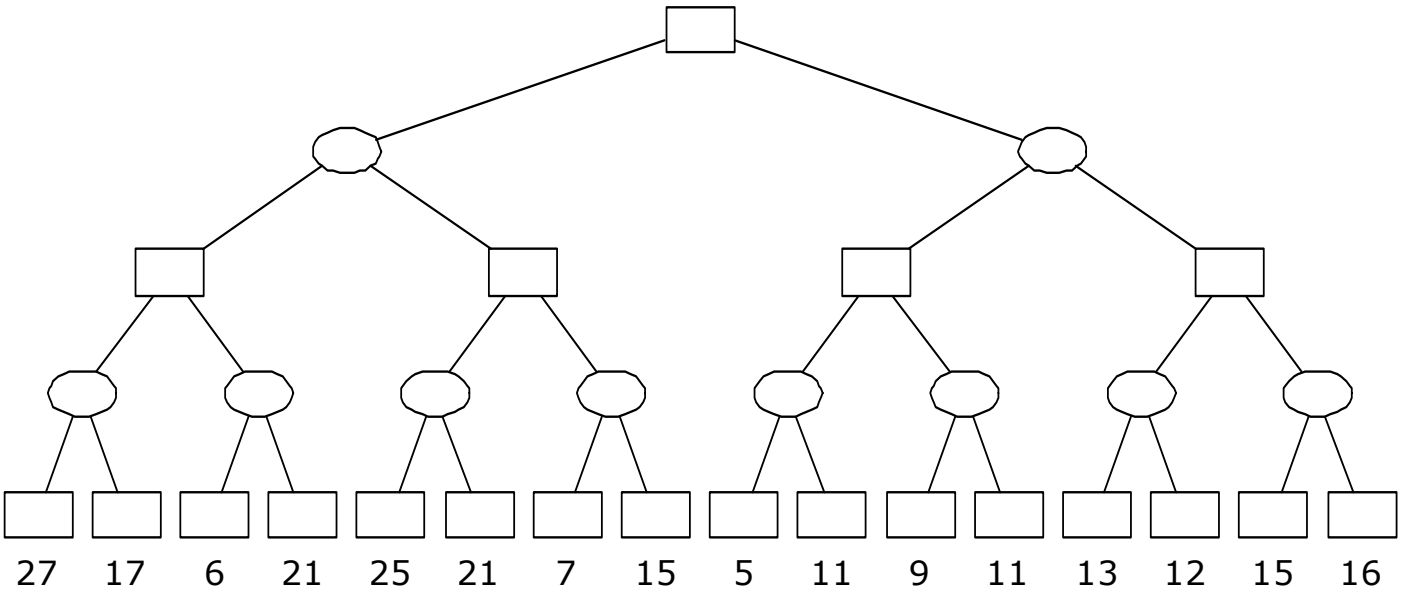
Alcuni insegnanti sono in grado di risolvere tutti i problemi.

a. Dimostrare con il metodo di risoluzione che:
   Alcuni insegnanti non sono studenti.
b. Scrivere il programma logico corrispondente ai primi due fatti o spiegare perché non si può.

Nota: si usino i predicati ins/1, stud/1 e solve/2 tralasciando il predicato relativo a problemi.

Esercizio 2 (punti 5)

Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore. Si assuma che il primo giocatore sia Max. Si mostri come gli algoritmi min-max e alfa-beta risolvono il problema



Esercizio 3 (punti 4)

Si consideri il seguente programma Prolog:

```
qsort([],[]).
qsort([X],[X]):-!.
qsort([H|T],S):-
    dividi(T,H,Min,Mag),
    qsort(Min,MinS),
    qsort(Mag,MagS),
    append(MinS,[H|MagS],S).
dividi([],X,[],[]).
dividi([H|T],X,[H|Min],Mag):- H<X,!, dividi(T,X,Min,Mag).
dividi([H|T],X,Min,[H|Mag]):- dividi(T,X,Min,Mag).
append([],X,X).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

Si mostri l'albero di derivazione SLD relativo alla query `qsort([3, 1], L).`

Esercizio 4 (punti 2)

Date la seguente KB di formule del FOL:

$\forall x$ uomo(x) $\Rightarrow$ uomo(figlio(x))
uomo(figlio(Adamo))

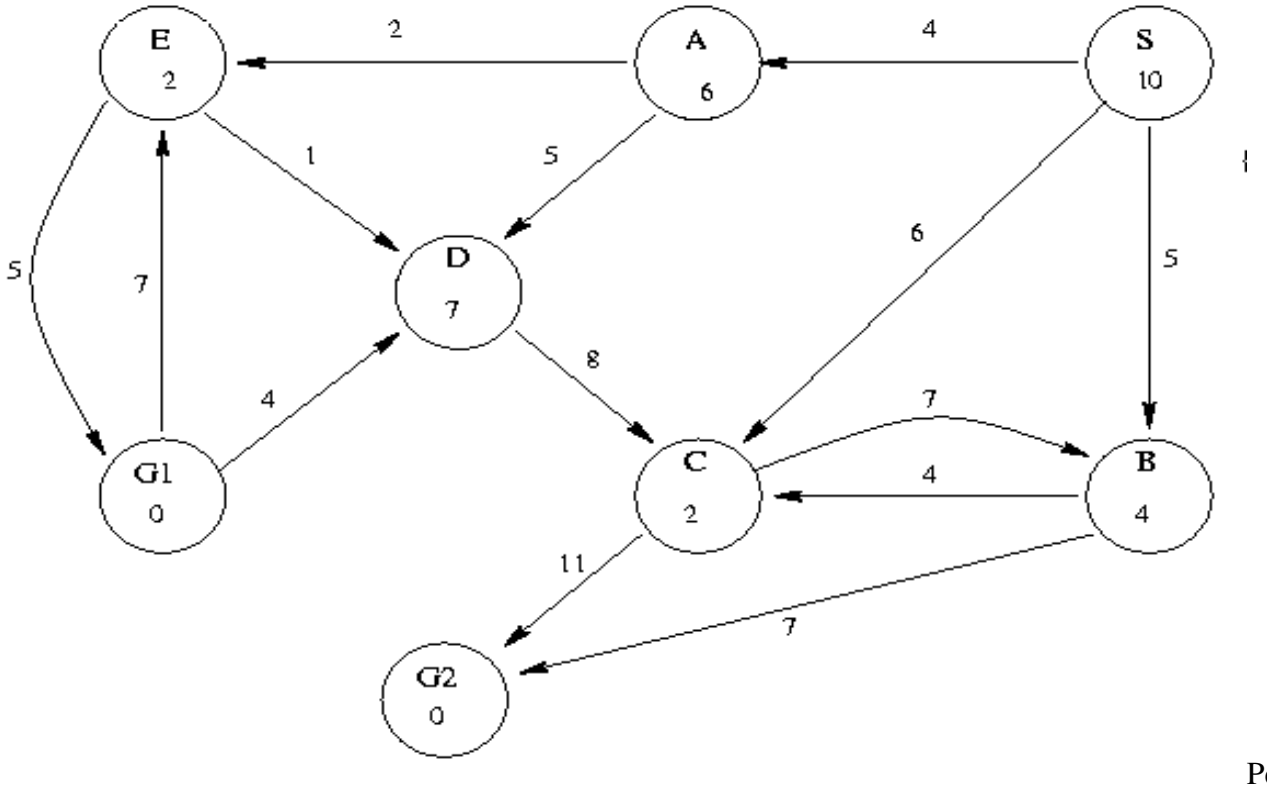
Scrivere il programma logico corrispondente alla KB e verificare se le seguenti formule sono deducibili dal programma, tramite un interprete Prolog:

uomo(adamo)
uomo(figlio(adamo)
$\exists$ Z uomo(Z)

Nota: si richiede che le formule siano convertite in query Prolog opportune, di tracciare il comportamento dell'interprete Prolog e di dire in quali casi l'interprete restituisce la risposta corretta e in quali no.

Esercizio 5 (punti 4)

Si assuma il seguente spazio di ricerca in cui S è lo stato iniziale e G1 e G2 sono stati finali. I numeri sugli archi rappresentano i costi e i numeri dentro i nodi rappresentano una stima della distanza dalla soluzione (h(n)).



Per ognuno degli algoritmi di ricerca menzionati sotto si dica quale stato goal viene raggiunto e si elenchino in ordine i nodi espansi (a parità di preferenza tra i nodi si usi l'ordine alfabetico delle rispettive etichette).

a. Ricerca in ampiezza;
b. Approfondimento iterativo;
c. Hill-climbing; (I nodi sono valutati da h(n))
d. A*

Esercizio 6 (punti 1)

Nel gioco degli scacchi si consideri il problema di pianificare una sequenza di azioni per spostare una torre da una casella A ad una casella B. Come sapete, la torre può essere spostata in orizzontale o verticale per un numero qualsiasi di caselle, ma non può scavalcare altri pezzi. La distanza Manhattan è un'euristica ammissibile per questo problema?

Esercizio 7 (punti 5)

Si costruisca un meta-interprete per Prolog (in Prolog) che chieda all'utente i goal che non è in grado di dimostrare con le clausole del programma, ma solo se essi sono *ground* (si supponga dato un predicato `ground/1` che è vero se il suo argomento è *ground*).

Se l'utente risponde `true`, il goal deve essere poi asserito come fatto *ground*.

Esercizio 8 (punti 3)

Si consideri il seguente problema di pianificazione. Ci sia un robot in grado di muoversi, di afferrare e di depositare oggetti all'interno di un ambiente composto da due stanze comunicanti *s1* e *s2*. Lo stato iniziale, l'obiettivo, le precondizioni e gli effetti di ogni azione sono formulati in
STRIPS:

INIT (At(robot,s1) ∧ At(object2,s1) ∧ At(object1,s2) ∧ Room(s1) ∧ Room(s2) ∧ HandEmpty)

GOAL (At(object1,s1) ∧ At(object2,s1))

ACTION (Go(X,Y),
PRECOND: Room(X) ∧ Room(Y) ∧ At(robot,X) ∧ X≠Y
EFFECT: ¬ At(robot,X) ∧ At(robot,Y))

ACTION (Pick(O),
PRECOND: Room(X) ∧ At(robot,X) ∧ At(O,X) ∧ HandEmpty ∧ O≠robot
EFFECT: ¬HandEmpty ∧ ¬At(O,X) ∧ Holding(O))

ACTION (Drop(O),
PRECOND: Room(X) ∧ At(robot,X) ∧ Holding(O)
EFFECT: HandEmpty ∧ At(O,X) ∧ ¬Holding(O))

Si risolva il problema utilizzando l'algoritmo STRIPS mostrando una sola strada verso la costruzione del piano (i goals si risolvano in ordine left most). Si indichino eventuali punti di scelta aperti individuati durante la ricerca. Per semplicita` si mostra gia' una parte della soluzione che andra' completatata fino al raggiungimento del successo. Si indichino poi le azioni risultanti.

Stato:	**Stack di goal e azioni:**
At(robot,s1) At(object2,s1) At(object1,s2) Room(s1) Room(s2) HandEmpty	At(object1,s1) ∧ At(object2,s1)
idem	At(object1,s1) At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	At(robot,s1) Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(s1) ∧ Room(s2) ∧ At(robot,s1) ∧ s1≠s2 **Go(s2,s1)** Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1)

	At(object1,s1) ∧ At(object2,s1)
idem	**Go(s2,s1)** **% la eseguo** Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s2) At(object2,s1) At(object1,s2) Room(s1) Room(s2) HandEmpty	Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)

Si prosegua……..

Esercizio 9 (punti 1)
Si spieghi cosa e' il meccanismo di regressione nel planning e a cosa serve

Esercizio 10 (punti 2)
Illustrare brevemente cosa sono e in cosa differiscono i modelli semantici chiamati "tassonomie" e "thesaurus", fornendo un esempio di conoscenza espressa in tali modelli.

SOLUZIONE Esercizio 1

A1. Formalizzazione:

1. $\forall x\ Stud(x) \Rightarrow \exists\ y\ Solve(x, y) \wedge \exists\ z \neg Solve(x, z)$
2. $\exists\ x\ Ins(x) \wedge \forall y\ Solve(x, y)$
3. Da dimostrare: $\exists\ x\ Ins(x) \wedge \neg\ Stud(x)$

A2 Trasformazione in forma a clausole:

1. $\forall x\ Stud(x) \Rightarrow \exists\ y\ Solve(x, y) \wedge \exists\ z \neg Solve(x, z)$

$\forall x \neg Stud(x) \vee (\exists\ y\ Solve(x, y) \wedge \exists\ z \neg Solve(x, z))$ [*eliminazione* $\Rightarrow$]

$\forall x \neg Stud(x) \vee (Solve(x, p1(x)) \wedge \neg Solve(x, p2(x)))$ [*skolemizzazione*]

$\neg Stud(x) \vee (Solve(x, p1(x)) \wedge \neg Solve(x, p2(x)))$ [*eliminazione* $\forall$]

$(\neg Stud(x) \vee Solve(x, p1(x))) \wedge (\neg Stud(x) \vee \neg Solve(x, p2(x)))$

1.1 $\{\neg Stud(x1), Solve(x1, p1(x1))\}$

1.2 $\{\neg Stud(x2), \neg Solve(x2, p2(x2))\}$

2. $\exists\ x\ Ins(x) \wedge \forall y\ Solve(x, y)$

$Ins(i) \wedge \forall y\ Solve(i, y)$ [*skolemizzazione*]

$\{Ins(i)\}$

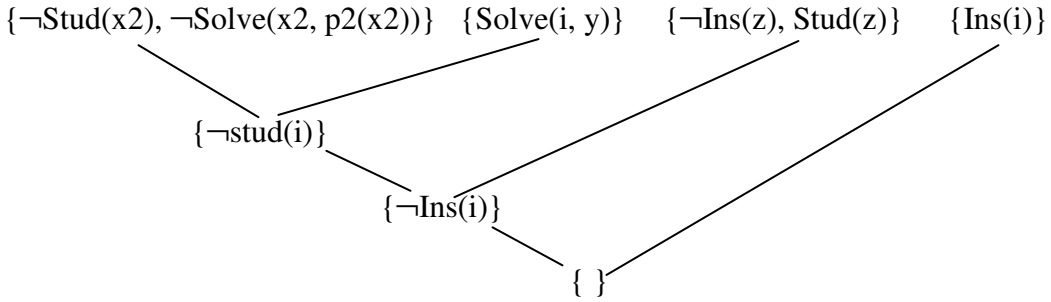
$\{\ Solve(i, y)\}$

Goal negato: $\neg\exists\ x\ Ins(x) \wedge \neg\ Stud(x)$

$\forall\ x \neg Ins(x) \vee Stud(x)$

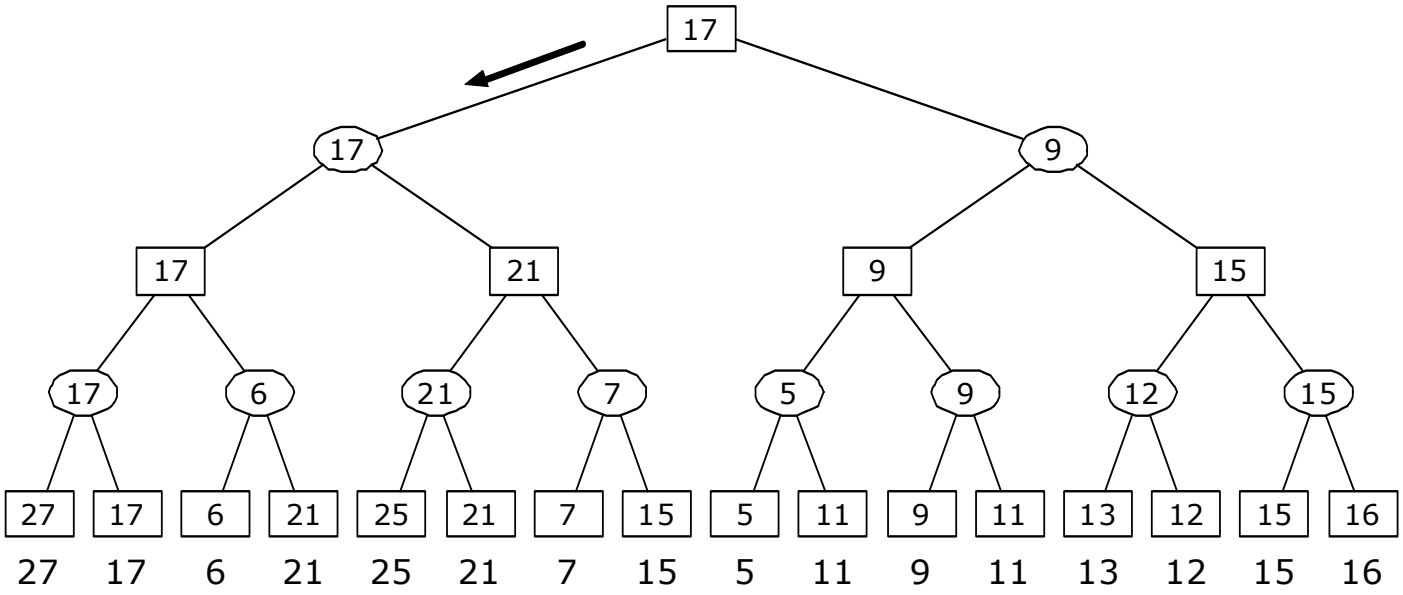
$\{\neg Ins(z), Stud(z)\}$

A3 Dimostrazione per refutazione:

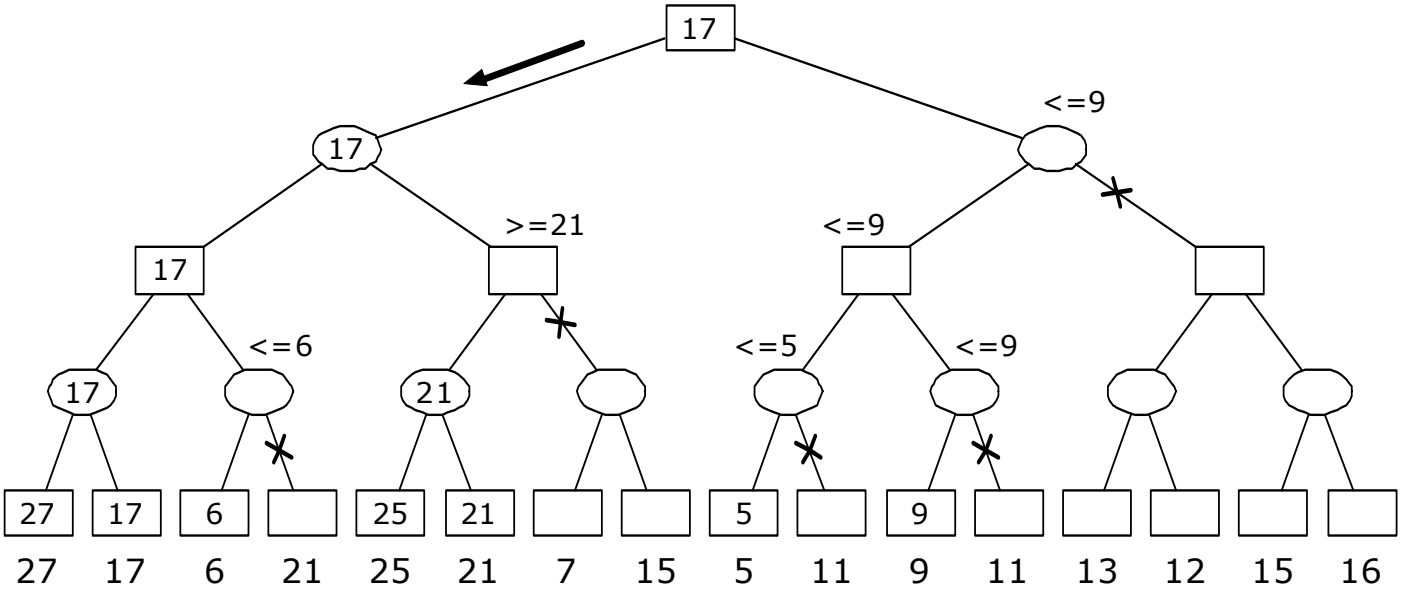


B. Non è possibile rendere come programma logico la KB iniziale in quanto la prima formula, trasformata in forma a clausole, non è una clausola Horn **definita** (non ci sono letterali positivi).

Esercizio 2 min-max:

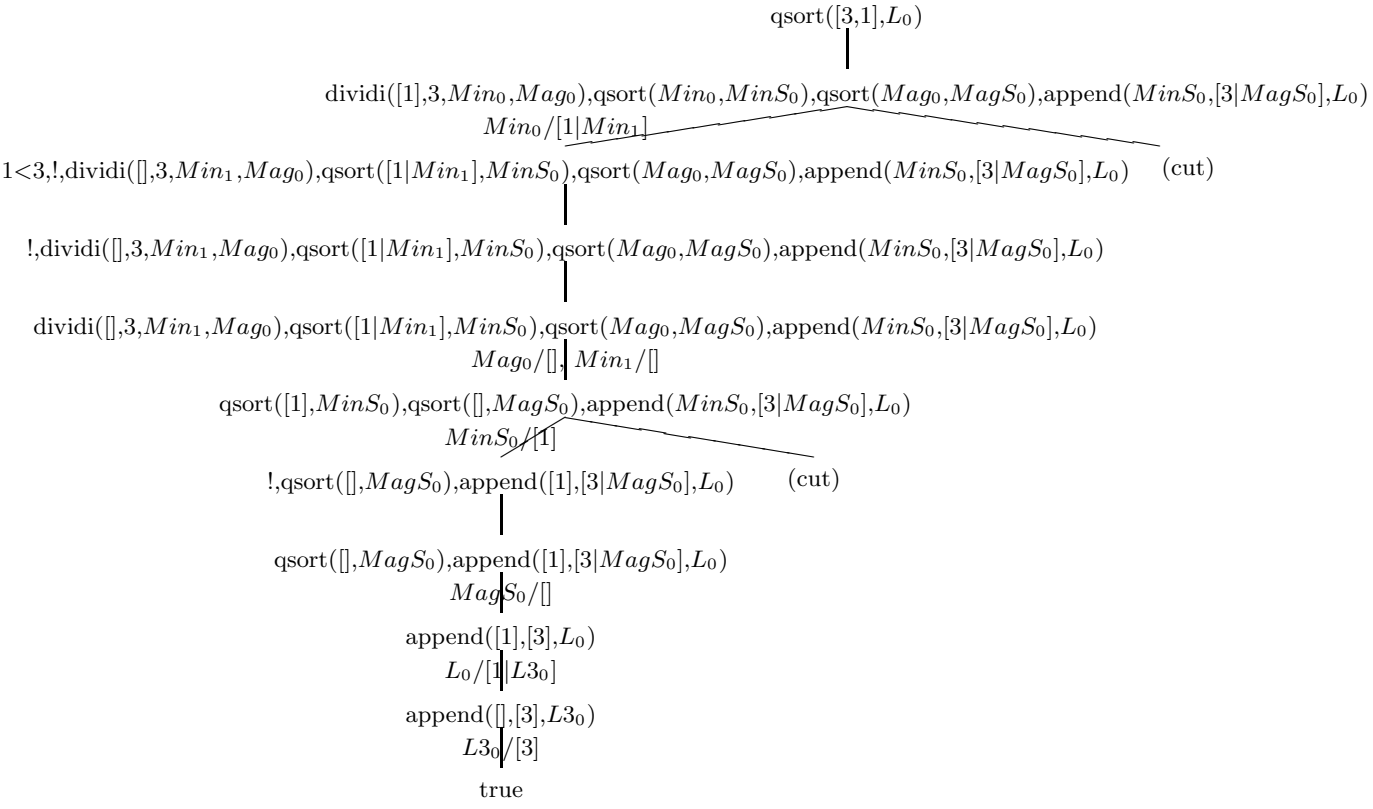


Alfa-beta:



Esercizio 3

Albero SLD:



Esercizio 4

B1. Programma logico:

```
uomo(figlio(X)) :- uomo(X).
uomo(figlio(adamo)).
```

B2. :- uomo(adamo).

NO

La risposta è corretta e infatti uomo(adamo) non è conseguenza logica della KB.

B3. :- uomo(figlio(adamo))
:- uomo(adamo)
fail
:-uomo(figlio(adamo))
YES

La risposta è corretta e infatti uomo(figlio(adamo)) è conseguenza logica della KB.

B4. :- uomo(Z) [figlio(X1)/Z]
:- uomo(Z1) [figlio(X2)/Z1]
:- uomo(Z2) .... [figlio(X3)/Z2]

Il programma non termina e non risponde YES come avrebbe dovuto essendo che ∃ Z uomo(Z) è conseguenza logica della KB.

Esercizio 5

a. Ricerca in ampiezza

Nodi espansi nell'ordine: S, A, B, C, D, E, C, G2
Goal trovato: G2

Per chiarezza includo il dettaglio di come viene aggiornata la frontiera, anche se non era richiesto. I nodi vengono espansi quando sono in testa alla frontiera e come conseguenza dell'espansione i successori vengono aggiunti in coda.

S
A, B, C
B, C, D, E
C, D, E, C, G2
D, E, C, G2, B, G2
E, C, G2, B, G2, D, G1
C, G2, B, G2, D, G1, C
G2, B, G2, D, G1, C, G2 Viene espanso G2 e riconosciuto come Goal.

b. Approfondimento iterativo

Nodi espansi nell'ordine: S, S, A, D, E, B, C, G2
Goal trovato: G2

Dettaglio di come viene aggiornata la frontiera. I nodi sono espansi quando sono in testa alla frontiera e come risultato dell'espansione i successori vengono aggiunti in testa.

S
S
A, B, C azzeramento della frontiera per raggiunto limite di profondità
S si riparte
A, B, C
D, E, B, C
E, B, C
B, C
C, G2, C
G2, C
Viene espanso G2 e riconosciuto come Goal.

c. Hill-climbing (assumo che i nodi siano valutati da h(n), il numero dentro i nodi).

Nodi espansi nell'ordine: S, C, G2
Goal trovato: G2

Dettaglio di come viene aggiornata la frontiera. I nodi sono espansi quando sono in testa alla frontiera ordinata secondo la *h*; come risultato dell'espansione i successori costituiscono la nuova frontiera; i numeri sono la valutazione dei nodi secondo *h*.
S(10)
C(2), B(4), A(6)
G2(0), B(4)
Viene espanso G2 e riconosciuto come Goal.
Nota: usare come funzione di valutazione la $f=g+h$ non ha molto senso: l'algoritmo si blocca dopo aver espanso S e C perché nessuno dei successori di C migliora la situazione (d'altra parte, per come è fatta, la *f* tende a crescere e questo è piuttosto inievitabile). A maggior ragione usare i costi sugli archi per valutare gli stati non ha senso.

d. A* (i nodi sono valutati da $f(n)= g(n) + h(n)$).
Nodi espansi nell'ordine: S, C, B, A, E, C, G1
Goal trovato: G1

Dettaglio di come viene aggiornata la frontiera (i nodi sono espansi quando sono in testa alla frontiera ordinata secondo la *f*; come risultato dell'espansione i successori sia aggiungono alla frontiera; i numeri sono la valutazione dei nodi secondo *f*).
S(10)
C(8), B(9), A(10)
B(9), A(10), B(17), G2(17)
A(10), C(11), G2(12), B(17), G2(17)
E(8), C(11), G2(12), D(16), B(17), G2(17)
C(11), G1(11), G2(12), D(14), D(16), B(17), G2(17)
G1(11), G2(12), D(14), D(16), B(17), B(17), G2(17), G2(20)
Viene espanso G1 e riconosciuto come Goal.

Esercizio 6
L'euristica della Manhattan distance dalla casella occupata dalla torre alla destinazione finale non è un'euristica ammissibile per il problema. Infatti basta considerare un caso in cui la torre dista dalla casella obiettivo un certo numero di celle e niente si frappone tra la torre e la sua destinazione. La stima ci dice che la distanza è pari al numero di caselle che la torre dovrà percorrere (supponiamo 4), ma in realtà la casella obiettivo sarà raggiungibile in una sola mossa.

Esercizio 7
Metainterpreti

```
solve(true):-!.
solve((X,Y)):- !,solve(X),solve(Y).
solve(X):- clause(X,Body),solve(Body),!.
solve(X):-
     ground(X),
     ask(X,Risp),Risp==true, asserta(X).

ask(X,Risp):-
     write(X),
     write("  true or false ? "),
     read(Risp).


% oppure:
solve(true):-!.
```

```
solve((A,B)):-!,solve(A),solve(B).
solve(A):-clause(A,B),solve(B),!.
solve(A):- ground(A),write(A), write("  true or false ? "),
           read(R),R==true,asserta(A),!.
```

Esercizio 8

La soluzione trovata è Go(s1,s2), Pick(object1), Go(s2,s1), Drop(object1).
INIT (At(robot,s1) ∧ At(object2,s1) ∧ At(object1,s2) ∧ Room(s1) ∧ Room(s2) ∧ HandEmpty)
GOAL (At(object1,s1) ∧ At(object2,s1))

ACTION (Go(X,Y),
PRECOND: Room(X) ∧ Room(Y) ∧ At(robot,X) ∧ X≠Y
EFFECT: ¬ At(robot,X) ∧ At(robot,Y))

ACTION (Pick(O),
PRECOND: Room(X) ∧ At(robot,X) ∧ At(O,X) ∧ HandEmpty ∧ O≠robot
EFFECT: ¬HandEmpty ∧ ¬At(O,X) ∧ Holding(O))

ACTION (Drop(O),
PRECOND: Room(X) ∧ At(robot,X) ∧ Holding(O)
EFFECT: HandEmpty ∧ At(O,X) ∧ ¬Holding(O))

Stato:	**Stack di goal e azioni:**
At(robot,s1) At(object2,s1) At(object1,s2) Room(s1) Room(s2) HandEmpty	At(object1,s1) ∧ At(object2,s1)
idem	At(object1,s1) At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	At(robot,s1) Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(s1) ∧ Room(s2) ∧ At(robot,s1) ∧ s1≠s2 **Go(s2,s1)** Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Go(s2,s1)** **% la eseguo** Holding(object1)

	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s2) At(object2,s1) At(object1,s2) Room(s1) Room(s2) HandEmpty	Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(s2) ∧ At(robot,s2) ∧ At(object1,s2) ∧ HandEmpty ∧ object1≠robot **Pick(object1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Pick(object1)** **% la eseguo** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s2) At(object2,s1) Room(s1) Room(s2) Holding(object1)	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	At(robot,s1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object1,s1) ∧ At(object2,s1)
idem	Room(s1) ∧ Room(s2) ∧ At(robot,s2) ∧ s1≠s2 **Go(s2,s1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Go(s2,s1)** **% la eseguo** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s1) At(object2,s1) Room(s1) Room(s2) Holding(object1)	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Drop(object1)** **% la eseguo** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s1)	At(object2,s1)

At(object2,s1) At(object1,s1) Room(s1) Room(s2) Handempty	At(object1,s1) ∧ At(object2,s1)
idem	VUOTO

L'and dei goal e' soddisfatto: stack dei goal vuoto.

NOTA: testo e soluzione avevano delle parti scorrette: versione corretta;

La soluzione trovata è Go(s1,s2), Pick(object1), Go(s2,s1), Drop(object1).
INIT (At(robot,s1) ∧ At(object2,s1) ∧ At(object1,s2) ∧ Room(s1) ∧ Room(s2) ∧ HandEmpty)
GOAL (At(object1,s1) ∧ At(object2,s1))

ACTION (Go(X,Y),
PRECOND: Room(X) ∧ Room(Y) ∧ At(robot,X) ∧ X≠Y
EFFECT: ¬ At(robot,X) ∧ At(robot,Y))

ACTION (Pick(O),
PRECOND: Room(X) ∧ At(robot,X) ∧ At(O,X) ∧ HandEmpty ∧ O≠robot
EFFECT: ¬HandEmpty ∧ ¬At(O,X) ∧ Holding(O))

ACTION (Drop(O),
PRECOND: Room(X) ∧ At(robot,X) ∧ Holding(O)
EFFECT: HandEmpty ∧ At(O,X) ∧ ¬Holding(O))

Stato:	**Stack di goal e azioni:**
At(robot,s1) At(object2,s1) At(object1,s2) Room(s1) Room(s2) HandEmpty	At(object1,s1) ∧ At(object2,s1)
idem	At(object1,s1) At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	At(robot,s1) Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)

idem	Room(X) ∧ At(robot,X) ∧ At(object1,X) ∧ HandEmpty ∧ object1≠robot **Pick(object1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(X) At(robot,X) At(object1,X) HandEmpty object1≠robot Room(X) ∧ At(robot,X) ∧ At(object1,X) ∧ HandEmpty ∧ object1≠robot **Pick(object1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	At(robot,s2) **% X/s2** At(object1,s2) HandEmpty object1≠robot Room(s2) ∧ At(robot,s2) ∧ At(object1,s2) ∧ HandEmpty ∧ object1≠robot **Pick(object1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(X) ∧ Room(s2) ∧ At(robot,X) ∧ X≠s2 **Go(X,s2)** At(object1,s2) HandEmpty object1≠robot Room(s2) ∧ At(robot,s2) ∧ At(object1,s2) ∧ HandEmpty ∧ object1≠robot **Pick(object1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Go(s1,s2) %X/s1, la eseguo** At(object1,s2) HandEmpty ∧ object1≠robot Room(s2) ∧ At(robot,s2) ∧ At(object1,s2) ∧ HandEmpty ∧ object1≠robot **Pick(object1)** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1)

	At(object1,s1) ∧ At(object2,s1)
At(robot,s2) At(object2,s1) At(object1,s2) Room(s1) Room(s2) HandEmpty	**Pick(object1)** **% la eseguo** Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s2) At(object2,s1) Room(s1) Room(s2) Holding(object1)	Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object1,s1) ∧ At(object2,s1)
idem	At(robot,s1) Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	Room(X) ∧ Room(s1) ∧ At(robot,X) ∧ X≠s1 **Go(X,s1)** Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Go(s2,s1)** **% X/s2, la eseguo** Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s1) At(object2,s1) Room(s1) Room(s2) Holding(object1)	Holding(object1) Holding(object1) Room(s1) ∧ At(robot,s1) ∧ Holding(object1) **Drop(object1)** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	**Drop(object1)** **% la eseguo** At(object2,s1) At(object1,s1) ∧ At(object2,s1)
At(robot,s1) At(object2,s1) At(object1,s1) Room(s1) Room(s2) Handempty	At(object2,s1) At(object1,s1) ∧ At(object2,s1)
idem	VUOTO

L'and dei goal e' soddisfatto: stack dei goal vuoto.

Esercizio 10

Le tassonomie sono gerarchie (acicliche) di termini, dove l'unica relazione possibile tra concetti e' quella del tipo "padre-figlio". Un classico esempio di tassonomia e' la classificazione degli esseri viventi secondo Linneo. I thesaurus sono tassonomie di origine linguistica, dove oltre alla relazione "padre-figlio" sono specificate alcune relazioni tipiche dello studio dei linguaggi naturali, quali olonimia, sinonimia, iperonimia, etc. etc. Un esempio di thesaurus e' il progetto WordNet.