Fondamenti di Intelligenza Artificiale M (9 crediti)
15 Giugno 2010 – Tempo a disposizione 2h 45min – Risultato 32/32 punti

Esercizio 1 (punti 3)

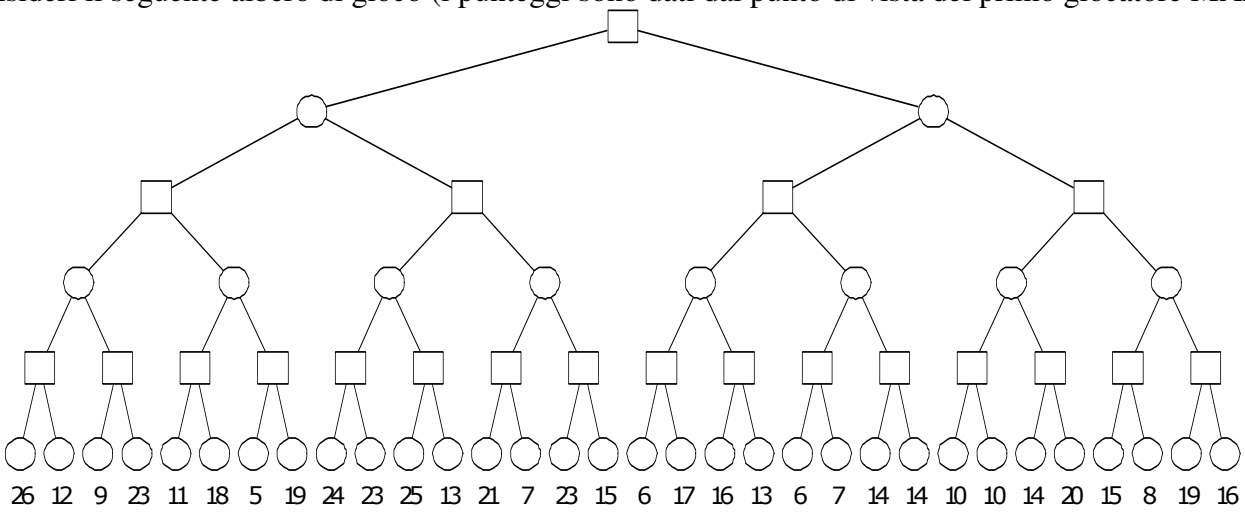
Si formalizzino in logica dei predicati del I ordine le seguenti frasi:

- *Se la TV non funziona, è rotta o non c'è alimentazione*
- *Se esiste qualche dispositivo che funziona, allora c'è corrente*
- *La TV non funziona*
- *Il lettore DVD funziona*

Dimostrare poi applicando il principio di risoluzione che: *La Tv è rotta.*

Esercizio 2 (punti 5)

Si consideri il seguente albero di gioco (i punteggi sono dati dal punto di vista del primo giocatore MAX):



Si mostri come gli algoritmi min-max e alfa-beta risolvono il problema

Esercizio 3 (punti 3)

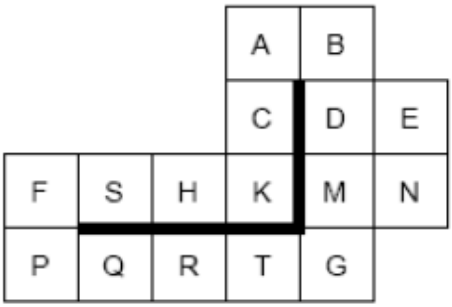
Il seguente predicato verifica che tutti gli elementi di una lista siano diversi:

```
alldiff([]).
alldiff([H|T]):-
    not(membchk(H,T)),
    alldiff(T).
membchk(A,[A|_]):- !.
membchk(A,[_|T]):- membchk(A,T).
```

Si mostri l'albero di derivazione SLDNF relativo al goal `alldiff([p,p(X),p(Y)]).`

Esercizio 4 (punti 5)

Si consideri la seguente scacchiera in cui i successori di ciascuna cella sono tutte le celle adiacenti in direzione Nord, Sud, Ovest e Est (in questo ordine), eccettuato per le celle sui confini della scacchiera o in corrispondenza del confine rappresentato dalla linea in grassetto. Per esempio, successori(M) ={D,G,N}. Si assuma che ogni mossa abbia costo 1.



Si vuole trovare un cammino dalla cella S alla cella G, applicando varie strategie di ricerca. Ad esempio, con la strategia *Breadth-First Search* (senza visite ripetute dello stesso nodo) si otterrebbe il cammino che tocca le celle nel seguente ordine:

SFHPKQCRATBG

Nel caso di scelte multiple data una strategia, si utilizzi l'ordine alfabetico delle celle.
Si descriva l'ordine di espansione dei nodi per le strategie (a-d) seguenti, indicando anche se il nodo goal è raggiunto da ciascuna delle strategie.

* (a) *Depth-First Search*. Si assuma che la strategia identifichi I cicli e li elimini, evitando di espandere un nodo già visitato.
* (b) *Greedy Search*. Si utilizzi come euristica la funzione h(stato) rappresentata dalla distanza di Manhattan da stato a G assumendo che non vi siano barriere. Ad esempio, h(K) = 2 e h(S) = 4.
* (c) *Hill-Climbing Search*. Si utilizzi la stessa euristica del caso (b).
* (d) *A* Search*. Si utilizzi la stessa euristica del caso (b). Si rimuovano gli stati ridondanti.
* (e) h è una funzione ammissibile? Si giustifichi la risposta.

Esercizio 5 (punti 3)

Scrivere un predicato: `pivoting(H,L1,LL,LG)`, che in base al valore di un certo numero `H`, suddivida una lista di partenza `L1` in due liste `LL` e `LG` contenenti, rispettivamente i valori di `L1` <= o > di `H`.
Esempio:

```
?-pivoting(3,[1,9,8,2],LL,LG).
     Yes LL=[1,2]  LG=[9,8]
```

Esercizio 6 (punti 2)

Si discuta la negazione in Prolog, perché non è la negazione classica, i suoi problemi di utilizzo e se ne mostri la sua implementazione in Prolog.

Esercizio 7 (punti 5)

Si scriva un metainterprete per Prolog che all'interno del body di ogni clausola selezioni sempre prima quei sottogoal con un minor numero di parametri. Ad esempio, nel body di

```
p(X,Y,Z):- b(Y, Y),n,  a(X, Y), c(Z).
```

selezionerà prima `n`, poi `c(Z)` e poi, indifferentemente `a(X, Y)` e `b(Y, Y)`.
Si supponga di avere a disposizione un predicato `sort(L,L1,L2)` che fornisce in `L2` la lista `L` ordinata (in senso decrescente) secondo i valori contenuti in `L1`.
Si supponga, inoltre, di avere un programma nella forma `clausola(Head, Body)` dove `Body` è una lista di sottogoal.

Esercizio 8 (punti 2)

Si supponga di utilizzare un semplice sistema di produzioni (basato su regole forward) per ordinare una stringa fatta di "a", "b" e "c".

- *Insieme di regole*:
  1. ba ⇒ ab
  2. ca ⇒ ac
  3. cb ⇒ bc

  Le regole vengono utilizzate nel seguente modo: si considera la stringa attualmente nella memoria di lavoro, e si considerano applicabili tutte le regole che hanno come precondizione una sottostringa contenuta in tale stringa. Ad esempio, se nella memoria di lavoro è presente la stringa "cbad", si possono applicare le regole (1) e (3). Si sceglie una regola da applicare secondo una strategia (vedi sotto): l'applicazione di una regola produce una nuova stringa, in cui la sottostringa presente nella precondizione della regola viene riscritta come specificato dalla regola. La nuova stringa così prodotta sostituisce nella memoria di lavoro la stringa precedente. Ad esempio, supponendo di applicare la regola (1) alla stringa "cbad", si ottiene la nuova stringa "cabd", che va a sostituire la stringa precedente.
- *Memoria di lavoro*: la stringa in una fase intermedia del processo di ordinamento.All'inizio la stringa da ordinare è **"cbaca"**.
- *Strategia*: la prima regola applicabile

Si descrivano i passaggi che portano alla soluzione riempiendo opportunamente la seguente tabella:

Ciclo	Memoria di lavoro	Insieme dei conflitti	Regola selezionata
0	cbaca		

Esercizio 9 (punti 2)

Si spieghi cosa si intende per "frame" problem nella pianificazione e si riportino come esempio in Prolog alcuni degli assiomi che lo trattano nella formulazione (calcolo delle situazioni) del mondo a blocchi di Kowalski.

Esercizio 10 (punti 2)

Nell'ambito dello standard OWL basato su Description Logics, descrivere brevemente cosa si intende quando una relazione (proprietà) è definita come "funzionale". Aggiungere un brevissimo esempio esplicativo.

SOLUZIONE

Esercizio 1

Formalizzazione:

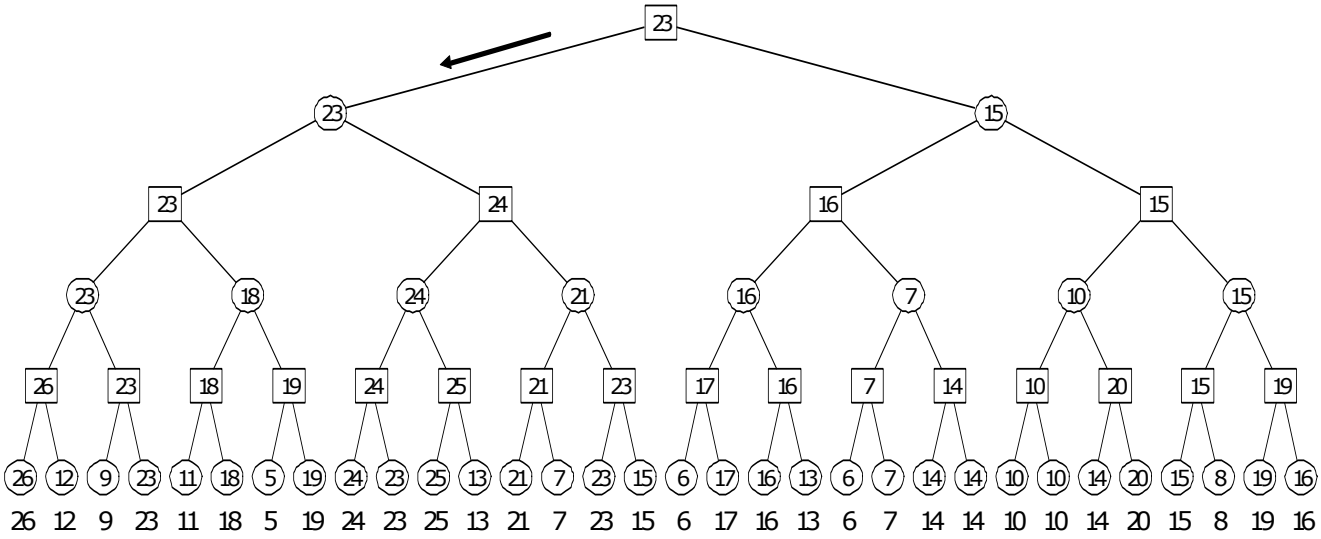
1. -works(tv) → broken(tv) ∨ -pow *in forma a clausole:* works(tv) ∨ broken(tv) ∨ -pow
2. (∃X works(X))→ pow *equivale a:* -∃X works(X) ∨ pow *che diventa:*
   ∀X -works(X) ∨ Pow *come clausola:*
   -works(X) ∨ Pow
3. -works(tv)
4. works(dvd)
5. -broken(tv) (*query negata, goal*)

Risoluzione:

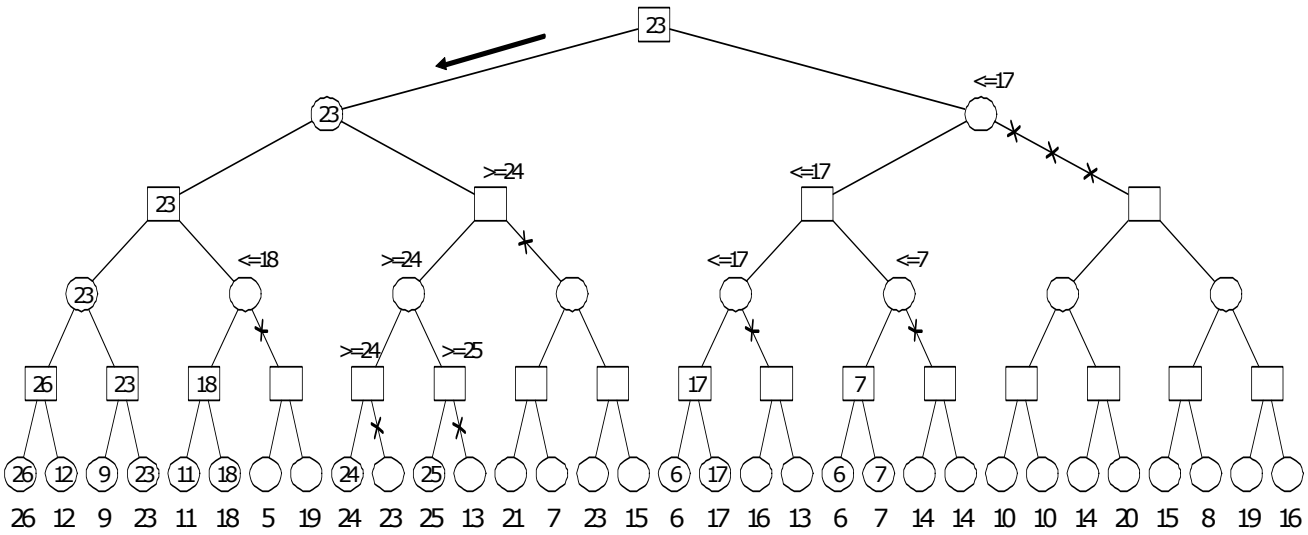
1+3 = 6. broken(tv) ∨ -pow
2+4= 7. pow
6+7= 8. broken(tv)
5+8 = 9. {}

Esercizio 2

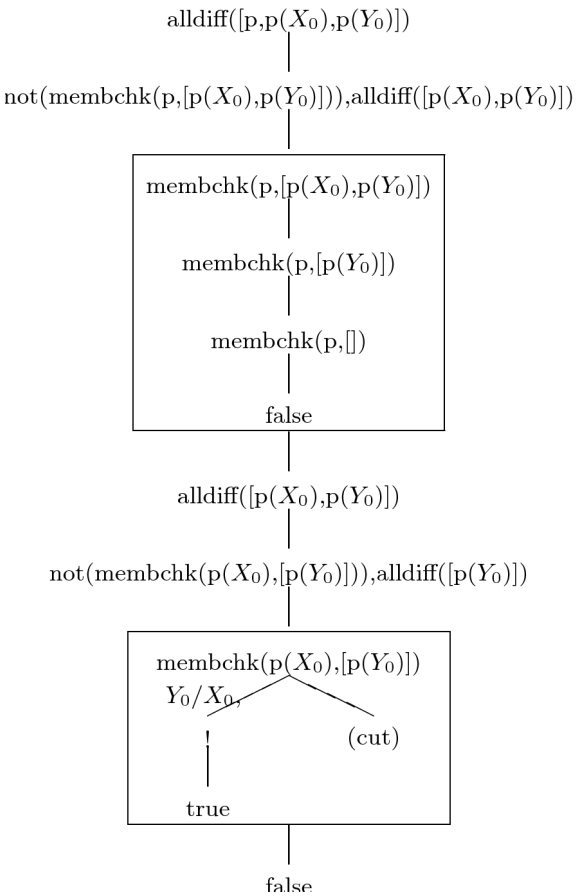
min-max:



alfa-beta:



Esercizio 3



Esercizio 4

* (a) Depth-First Search

SFPQRTG

* (b) Greedy Search

SHKCABDMG

* (c) Hill-Climbing Search

SHK - poi si ferma in un massimo locale

* (d) A* Search

SHKCFPQRTG

* (e) Sì, la funzione è ammissibile poiché non sovrastima mai la distanza dal goal.

Nel caso ci si trovi adiacenti al goal, la funzione euristica calcola la distanza (reale) più breve dal goal.
Nel caso ci sia una barriera tra la posizione raggiunta e il goal, la funzione euristica calcola una stima per difetto della più piccola distanza dal goal, poiché non tiene conto della barriera.

Esercizio 5

```
pivoting(H,[],[],[]).

pivoting(H,[X|T],[X|L],G):- X=<H,!,pivoting(H,T,L,G).

pivoting(H,[X|T],L,[X|G]):- pivoting(H,T,L,G).
```

Esercizio 7

```
solve([]).
solve([Head|Tail]):-!,
      solve(Head),
      solve(Tail).
solve(Goal):-

      clausola(Goal, Body),
      ordina_body(Body, Body_ord),
            solve(Body_ord).

ordina_body(Body, Body_ord):-
      numero_par(Body, Parametri),
      sort(Body, Parametri, Body_rev),
      reverse(Body_rev, Body_ord).

numero_par([],[]).
numero_par([Head|Tail],[Nhead |Ntail]):-
      Head=..[_|Arg],
      conta(Arg, Nhead),
      numero_par(Tail, Ntail).


conta([], 0):- !.
conta([Arg|Tail],  Nout):-
      conta(Tail, Nin), Nout is Nin +1.
```

Esercizio 8

Ciclo	Memoria di lavoro	Insieme dei conflitti	Regola selezionata
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2, 3	2
3	acbac	1, 3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	0	ALT