COMPITO DI INTELLIGENZA ARTIFICIALE (v.o.) – PARTE I
FONDAMENTI DI INTELLIGENZA ARTIFICIALE

21 Giugno 2007 (Tempo a disposizione 2,30h; su 32 punti)

Esercizio 1 (punti 7)
Si rappresentino in logica dei predicati del I ordine le seguenti frasi:

- Tom ama i treni.
- Chi ama i treni è appassionato di locomotive elettriche o a carbone (non esclusivo).
- Gli appassionati di locomotive a carbone non amano la tecnologia.
- Gli appassionati di locomotive elettriche amano la velocità.
- Tom non ama le cose che ama Anna e viceversa.
- Anna ama la velocità.

Le si trasformi in clausole e si dimostri usando la risoluzione che esiste un appassionato di locomotive a carbone che non è appassionato anche di locomotive elettriche.

Esercizio 2 (punti 7)

Il seguente programma Prolog è un riconoscitore sintattico per le espressioni:

```
mar07(L):- espressione(L,[]),!.
espressione(L,R):- termine(L,R).
espressione(L,R):-
    termine(L,R1),
    R1 = [ + |R2],
    espressione(R2,R).
termine(L,R):- fattore(L,R).
termine(L,R):- fattore(L,R1),
    R1 = [ * |R2],
    termine(R2,R).
fattore([N|R],R):- number(N).
```

(dove number/1 è un predicato che ha successo se il suo argomento è un numero). Si mostri l'albero SLD relativo alla invocazione:

```
?- mar07([3, *, 2]).
```

Esercizio 3 (punti 6)

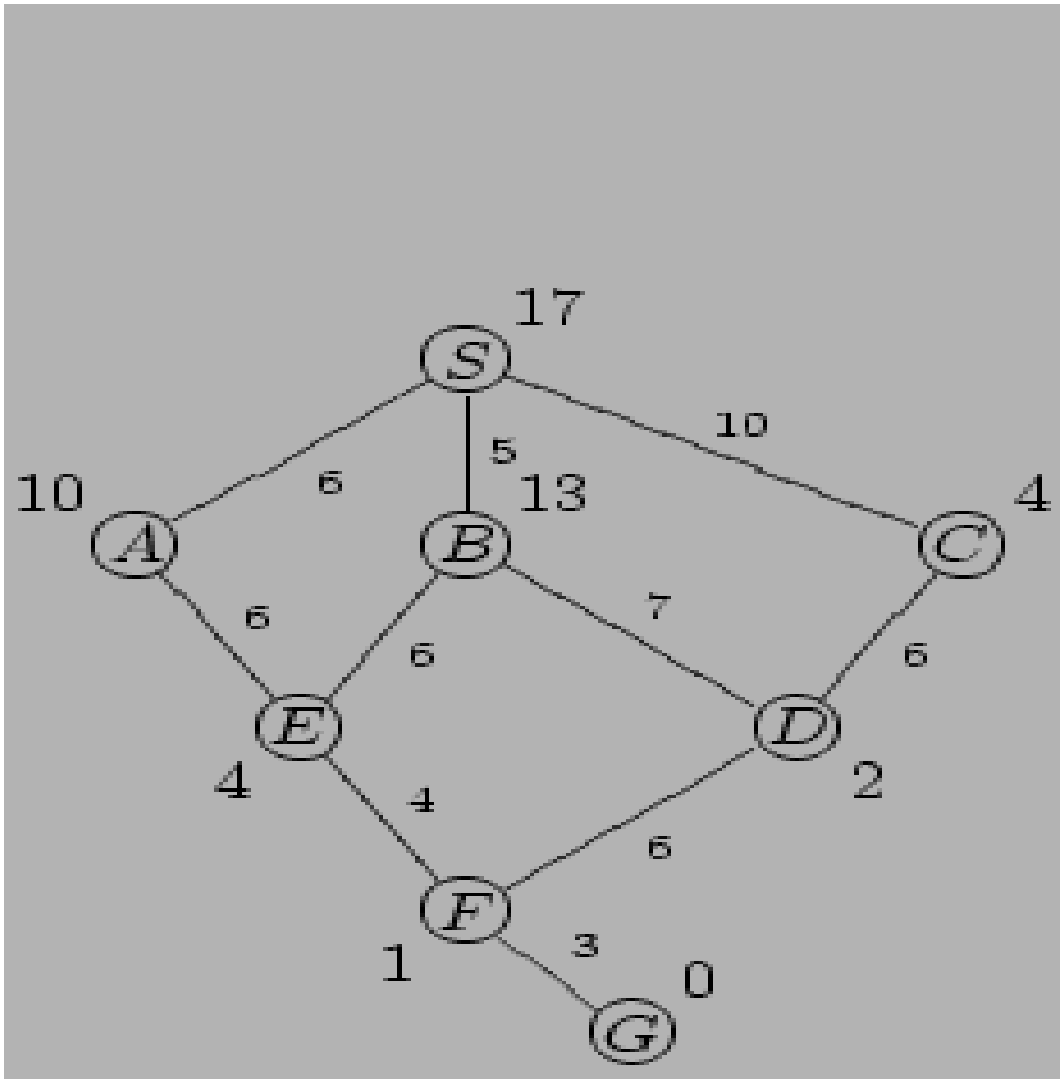
Si scriva un programma Prolog che prende in ingresso una lista L di numeri ed un numero N e produce in uscita una possibile espressione EX (se esiste) tale che
1. EX contiene come argomenti gli elementi della lista, nell'ordine dato
2. la valutazione di EX fornisca come risultato N.
Si considerino come operatori possibili +, -, *, /.
Per semplicità si considerino solo le espressioni in cui vengono eseguiti prima i calcoli più a destra (ad es, 1+2-3 va pensato come 1+(2-3))
Esempio:

```
?- equiv([1,2,3,4,5,6],EX,22).
yes, EX = 1-(2+(3+(4-5*6)))
```

Esercizio 4 (punti 9)

Si esegua l'algoritmo A* (con aggiornamento dei costi quando necessario) sul grafo mostrato nella seguente figura, dove il valore di h e' rappresentato dal numero vicino al nodo. Si parta dal nodo iniziale S e si mostri per passi come crescono le code dei nodi aperti e chiusi con le relative informazioni (cioè per ogni nodo N hiuso o aperto tramite la coppia [N+percorso per attivare a N, valore di f]) fino al goal G.

L'euristica assegnata e' ammissibile? E' consistente? L'algoritmo utilizzato trovera' il percorso ottimo?



Esercizio 5 (punti 3)

Si spieghi in cosa consiste l'occur check e cosa implica il non applicarlo.

SOLUZIONE

Esercizio 1

Logica dei predicati:

- *Tom ama i treni.*
- *Chi ama i treni è appassionato di locomotive elettriche o a carbone (non esclusivo).*
- *Gli appassionati di locomotive a carbone non amano la tecnologia.*
- *Gli appassionati di locomotive elettriche amano la velocità.*
- *Tom non ama le cose che ama Anna e viceversa.*
- *Anna ama la velocità.*

ama(tom, treni)
$\forall$X ama(X, treni) $\Rightarrow$ appassionato(X,elettr) or appassionato(X,carbone)
$\forall$X appassionato(X,carbone) $\Rightarrow$ not ama(X, tecnologia)
$\forall$X appassionato(X,elettr) $\Rightarrow$ ama(X, velocita)
$\forall$X ama(tom,X) $\Rightarrow$ not ama(anna,X)
$\forall$X ama(anna,X) $\Rightarrow$ not ama(tom,X)
ama(anna, velocita)
Query: $\exists$X appassionato(X,carbone) and not appassionato(X,elettr)

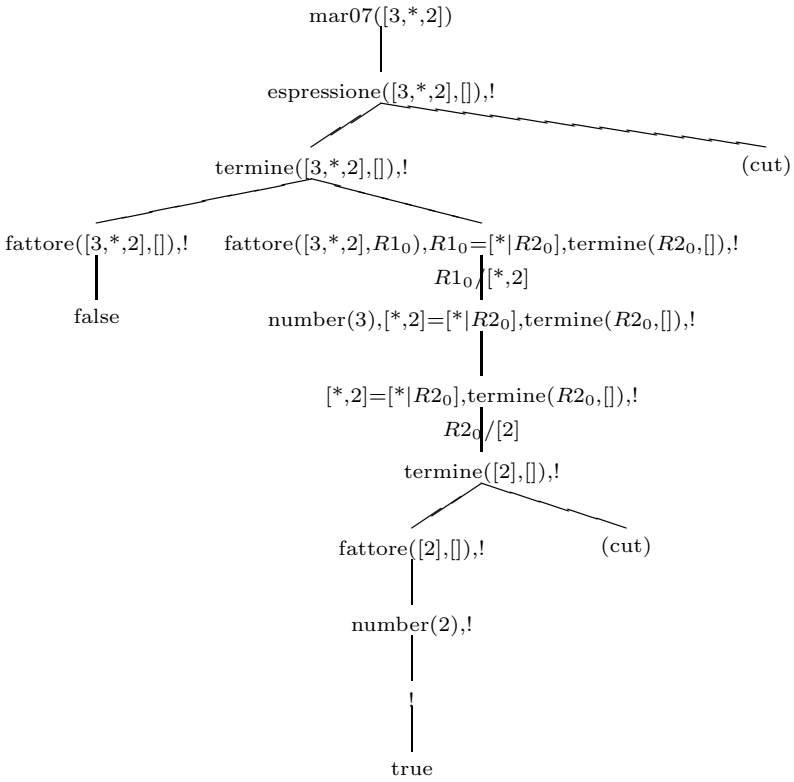
Clausole:

1) ama(tom, treni)
2) not ama(X, treni) or appassionato(X,elettr) or appassionato(X,carbone)
3) not appassionato(X,carbone) or not ama(X, tecnologia)
4) not appassionato(X,elettr) or ama(X, velocita)
5) not ama(tom,X) or not ama(anna,X)
6) not ama(anna,X) or not ama(tom,X)
7) ama(anna, velocita)
8) Goal: not appassionato(X,carbone) or appassionato(X,elettr)

Risoluzione:

9) Da 6+7) not ama(tom, velocita)
10) Da 9+4) not appassionato(tom, elettr)
11) Da 10+8) not appassionato(tom,carbone)
12) Da 10+11+2) not ama(tom, treni)
13) Da 12+1) clausola vuota

Esercizio 2



Esercizio 3

```
equiv(L,Op,N):-
            equation(L,Op),
            N is Op.

equation([X],X) :- !.
equation([X|T],X+Y):-
            equation(T,Y).
equation([X|T],X-Y):-
            equation(T,Y).
equation([X|T],X*Y):-
            equation(T,Y).
equation([X|T],X/Y):-
            equation(T,Y).
```

Esercizio 4:

1) Aperti: [[S,17]]
   Chiusi : []

2) Aperti: [[SC,14], [SA,16], [SB,18]]
   Chiusi : [[S,17]]

3) Aperti: [[SA,16], [SCD,18], [SB,18]]
   Chiusi : [[C,14], [S,17]]

4) Aperti: [[SAE,16], [SCD,18], [SB,18]]

Chiusi : [[C,14], [A,16], [S,17]]

5) Aperti: [[SAEF,17], [SCD,18], [SB,18]] =➔ SAEB NON VIENE INSERITO O VALUTATO IN QUANTO [SAEB,31]
Chiusi : [[E,16], [C,14], [A,16], [S,17]]

6) Aperti: [[SCD,18], [SB,18], [SAEFG,19]] =➔ SAEFD NON VIENE INSERITO O VALUTATO IN QUANTO [D,24]
Chiusi : [[E,16], [C,14], [A,16], [F,17], [S,17]]

7) Aperti: [[SB,18], [SAEFG,19]] =➔ SCDB ED SCDF NON VENGONO INSERITI O VALUTATO IN QUANTO [SCDB,36] E [SCDF ,23].
Chiusi : [[E,16], [C,14], [A,16], [F,17], [S,17], [D,18]]

8) Aperti: [[SBEFG,18]] =➔ [SBD, 14] non viene inserito poiché D è in Chiusi; il costo per giungere a D viene aggiornato (da 18 a 14); [SBE, 15] non viene inserito perché E è già in Chiusi; il costo per giungere ad E viene aggiornato (da 16 a 15); siccome il costo per giungere ad E viene aggiornato, viene aggiornato il costo per giungere da F (da 17 a 16); siccome il costo per giungere ad E è stato aggiornato, viene aggiornato anche [SAEFG,19]: in particolare il percorso diventa [SBEFG,18].
Chiusi : [[E,15], [C,14], [A,16], [F,16], [S,17], [D,14], [B, 18]]

9) GOAL

L'euristica assegnata e' ammissibile? SI'
E' consistente? NO
L'algoritmo utilizzato trovera' il percorso ottimo? SI