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

15 Gennaio 2007 (Tempo a disposizione 2h; su 32 punti)

Esercizio 1 (punti 6)
Dare di ognuna delle seguenti frasi una rappresentazione in logica dei predicati
a) Cavalli, mucche e maiali sono mammiferi.
b) Il figlio di un cavallo è un cavallo.
c) Barbablu è un cavallo.
d) Barbablu è il padre di Charlie.
e) Figlio e padre sono relazioni inverse (ovvero se x e' figlio di y allora y e' padre di x e viceversa).
f) Ogni mammifero ha un padre.
Si dimostri poi tramite risoluzione che Charlie è un cavallo

Esercizio 2 (punti 7)
Considerando il seguente programma Prolog:

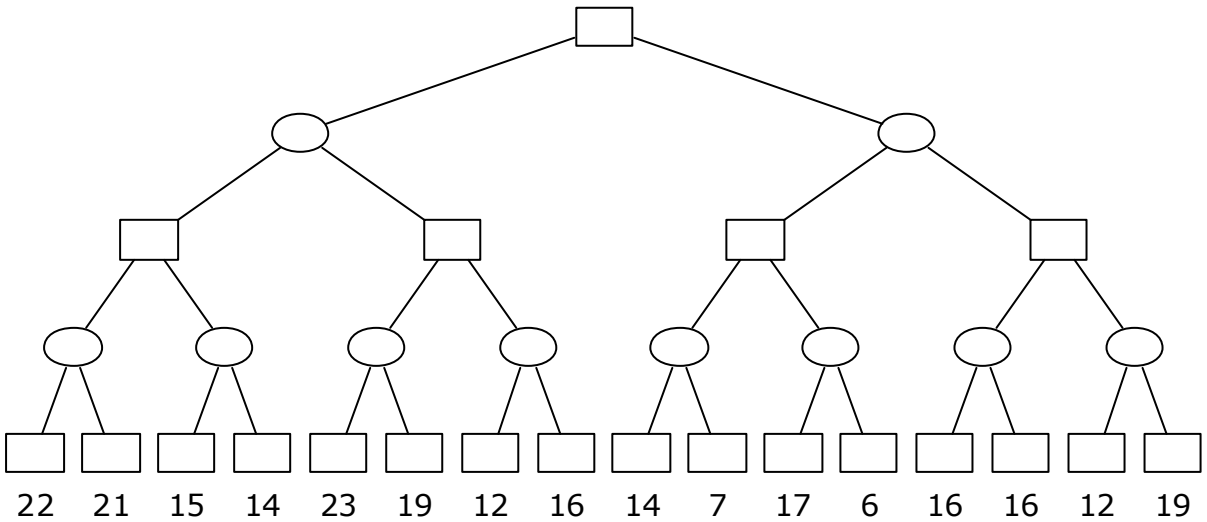
```
setdiff(L,[],L):-!.
setdiff([],_,[]):-!.
setdiff([H|T1],L2,[H|T2]):- not(member(H,L2)),!,
                            setdiff(T1,L2,T2).
setdiff([_|T],L2,T2):- setdiff(T,L2,T2).

member(X,[X|_]):-!.
member(X,[_|T]):-member(X,T).
```

Si rappresenti l'albero di derivazione SLDNF relativo al goal

```
?-setdiff([1,2,3],[2,3],L).
```

e si dica qual è la risposta calcolata.

Esercizio 3 (punti 6)
Si consideri il seguente albero di gioco, dove i punteggi sono tutti dal punto di vista del primo giocatore (MAX).



Si mostri come l'algoritmo min-max risolve il problema e quale mossa viene scelta. Si mostrino poi i tagli alfa-beta.

Esercizio 4 (punti 6)

Si scriva un programma Prolog `liste(Lin,Lout)` che data una lista di liste di interi `Lin`, produca una lista di interi in uscita `Lout` contenente solo gli elementi di `Lin` che hanno lunghezza uguale al proprio **ultimo** elemento. Si scrivano esplicitamente tutti i predicati Prolog usati nella soluzione.

Esempi:

```
?-liste([[2], [3,2], [2,3,3], [], [1,4,5,4]], X)
restituisce
yes   X=[[3,2], [2,3,3], [1,4,5,4]]

?-liste([[9,20]], X)
restituisce
yes   X=[]
```

Esercizio 5 (punti 4)

Si applichi la arc consistenza al seguente problema di soddisfacimento di vincoli:

```
A::[1..50],
B::[20..40],
C::[1..30],
D::[1..30],
A ≤ B,
B+7 < C,
D ≠ B,
D > A
D+10 < C
```

e si verifichi se la rete può essere resa arc consistente.

Esercizio 6 (punti 3)

Si descriva come lavora l'algoritmo di unificazione con particolare riferimento ai legami che crea. Si mostri poi in quale parte si applica l'occur check ed in cosa consiste.

SOLUZIONE

Esercizio 1

Nota: le variabili sono indicate con minuscole e i predicati e costanti con maiuscole. Se non esplicitamente scritto, le variabili sono quantificate universalmente.
a) Cavallo(x) -> Mammifero(x), Mucca(x)->Mammifero(x), Maiale(x)->Mammifero(x)
b) Figlio(x,y) and Cavallo(y) ->Cavallo(x)
c) Cavallo(Barbablu)
d) Padre(Barbablu,Charlie)
e) Figlio(x,y) -> Padre(y,x), Padre(x,y) -> Figlio(y,x)
f) Mammifero(x)-> esiste y Padre(y,x)

goal: Cavallo(Charlie).

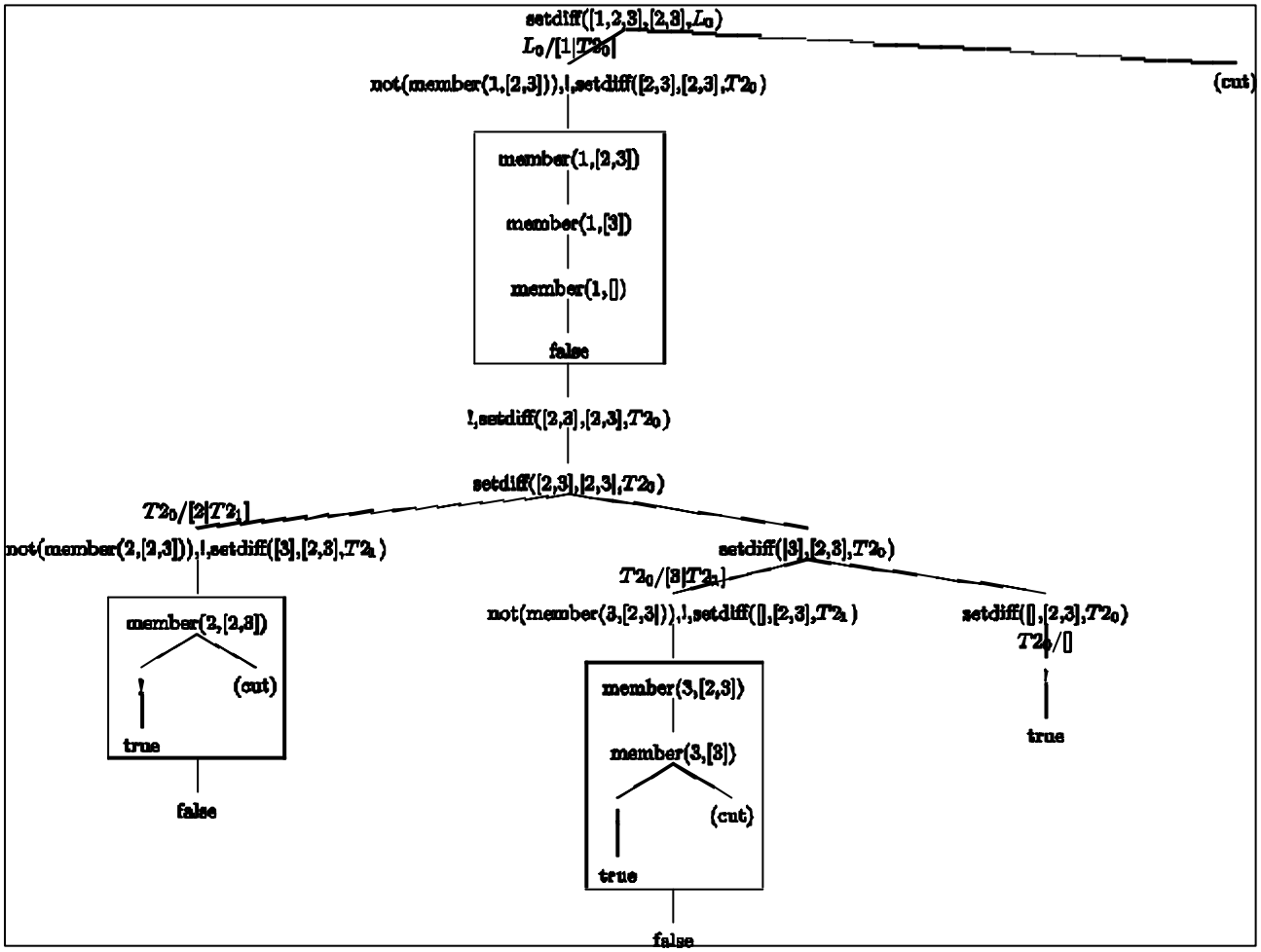
Traduzione:

1. not Cavallo(x) or Mammifero(x),
2. not Mucca(x) or Mammifero(x),
3. not Maiale(x) or Mammifero(x)
4. not Figlio(x,y) or not Cavallo(y) or Cavallo(x)
5. Cavallo(Barbablu)
6. Padre(Barbablu,Charlie)
7. Not Figlio(x,y) or Padre(y,x),
8. Not Padre(x,y) or Figlio(y,x)
9. not Mammifero(x) or Padre(g(x) ,x)
10. Goal negato : not Cavallo (Charlie)

Risoluzione:

10 + 4 = 11 not Figlio(Charlie,y) or not Cavallo(y)
11 +8 = 12 not padre (y, Charlie) or not cavallo (y).
12 + 6 = 13 not cavallo (Barbablu)
13 + 5. contraddizione!

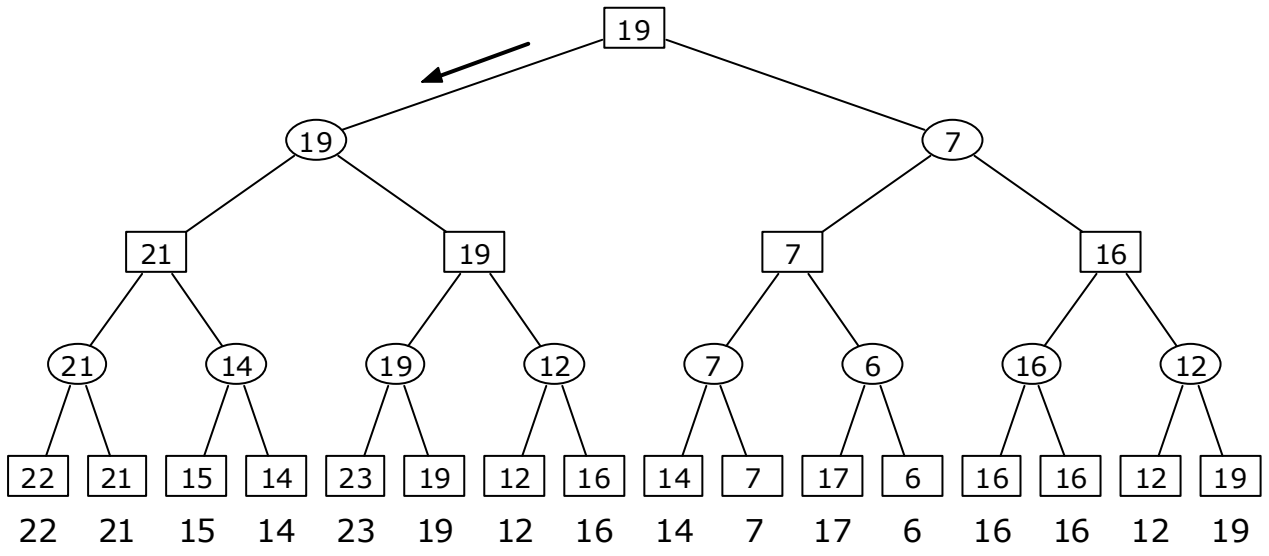
Esercizio 2



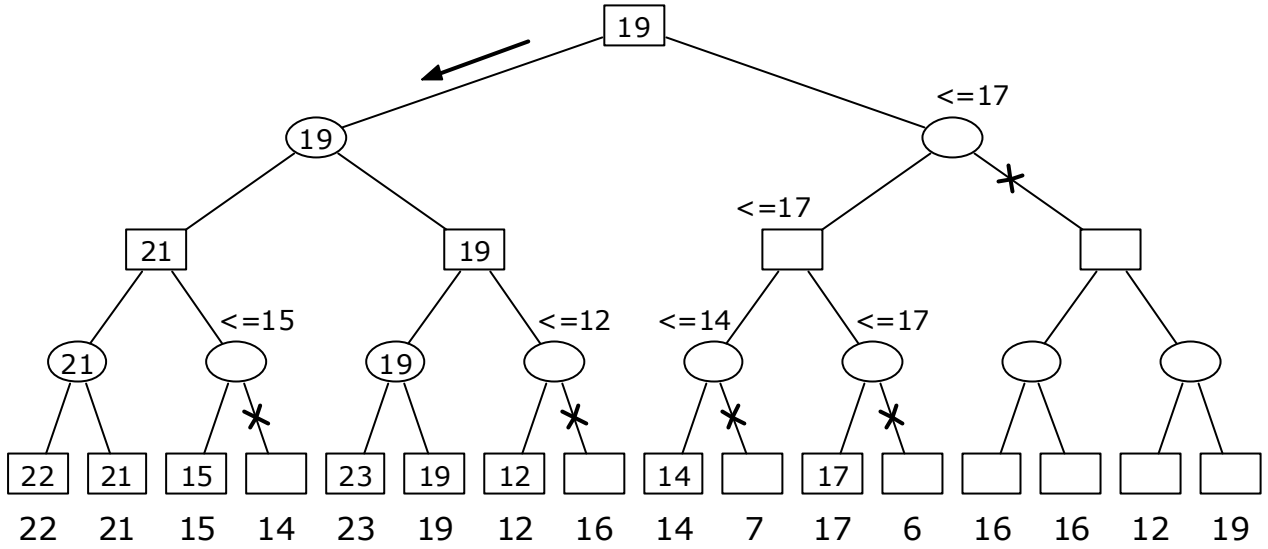
La risposta calcolata è L/[1].

Esercizio 3

min-max



alfa-beta



Esercizio 4:

```
liste([],[]).
liste([H|T],[H|Lout]):-
    lastlen(H,1,_),!,
    liste(T,Lout).
liste([_|T],Lout):-
    liste(T,Lout).


lastlen([X],Y,Y):- !, X=Y.
lastlen([_,H2|T],Nin,Nout):-
    Ntemp is Nin+1,
    lastlen([H2|T],Ntemp,Nout).
```

Oppure:

```
liste([],[]).
liste([H|T],[H|T2]):-last_length(H),!,liste(T,T2).
liste([_|T],L):-liste(T,L).

last_length(L):-last_length(L,1).
last_length([X],X):-!.
last_length([_|T],N):-N2 is N+1,last_length(T,N2).
```

Esercizio 5:

A::[1..50], B::[20..40], C::[1..30], D::[1..30]

Prima iterazione:

	A	B	C	D
A ≤ B	**1..40**	20..40	1..30	1..30
B+7 < C	1..40	**20..22**	**28..30**	1..30
D ≠ B	1..40	20..22	28..30	1..30
D > A	**1..29**	20..22	28..30	**2..30**
D+10 < C	1..29	20..22	28..30	**2..19**

Seconda iterazione:

	A	B	C	D
A ≤ B	**1..22**	20..22	28..30	2..19
B+7 < C	1..22	20..22	28..30	2..19
D ≠ B	1..22	20..22	28..30	2..19
D > A	**1..18**	20..22	28..30	2..19
D+10 < C	1..18	20..22	28..30	2..19

La terza iterazione non cambia i domini riportati nell'ultima riga.
La rete può essere resa arc consistente.