COMPITO DI INTELLIGENZA ARTIFICIALE Parte I
e FONDAMENTI DI INTELLIGENZA ARTIFICIALE
25 Giugno 2003
Durata: 2 h (totale su 32 punti)

Esercizio 1 (punti 6)

Si scriva un programma Prolog `conclun(Lin1, Lin2, Y)` che date due liste di elementi `Lin1` e `Lin2`, produca in uscita il valore per `Y` pari alla somma delle lunghezze delle due liste in ingresso. Si scrivano esplicitamente tutti i predicati Prolog usati nella soluzione.
Esempio:

```
conclun([a,b,c], [3,c,7,9,0], X)
```

restituisce

```
X=8
```

Esercizio 2 (punti 8)

Considerando il seguente programma Prolog:

```
nextto(X,Y,[X,Y|_]).
nextto(X,Y,[_|L]) :- nextto(X,Y,L).
```

si rappresenti l'albero di derivazione SLDNF solo fino alla prima soluzione relativo al goal

```
?- K=[A,2,C,D], nextto(1,2,K), nextto(3,2,K), not nextto(2,3,K).
```

Facoltativo: Considerando la seguente implementazione (2):

```
nextto(X,Y,[X,Y|_]).
nextto(X,Y,[_,H|L]) :- nextto(X,Y,[H|L]).
```

la semantica dichiarativa del predicato cambia? Questa seconda implementazione è più o meno efficiente? Si indichino sull'albero SLDNF quali nodi in più o in meno si sarebbero dovuti esplorare.

Esercizio 3 (punti 8)

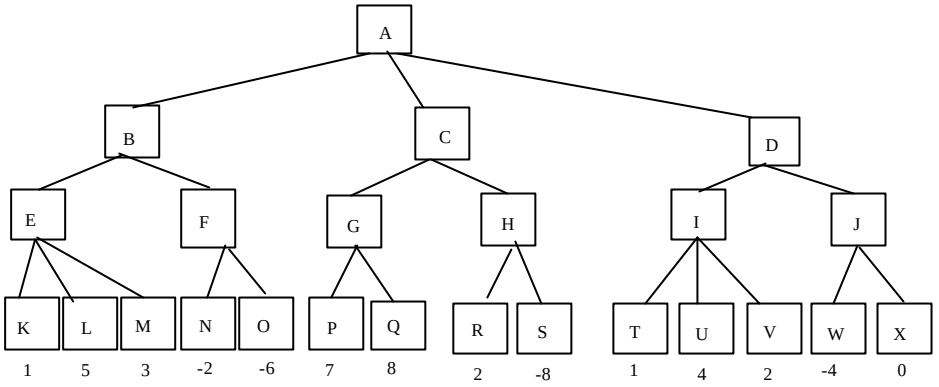
Si traducano in logica dei predicati le frasi seguenti:

- Gli esseri viventi del mondo sono animali o (xor) vegetali.
- Gli animali carnivori vogliono mangiare qualunque altro animale.
- Leo è un essere vivente animale carnivoro.
- Tarzan è un essere vivente animale.

Si trasformino in clausole le formule scritte e si usi questa conoscenza per dimostrare, tramite il principio di risoluzione, che esiste almeno un animale che vuole mangiare Tarzan.

Esercizio 4 (punti 5)

Si consideri il seguente albero di gioco dove i punteggi sono tutti dal punto di vista del primo giocatore:



Supponendo che il primo giocatore sia MIN, quale mossa dovrebbe scegliere? Si risolva l'esercizio tramite l'algoritmo MIN-MAX. Successivamente si mostrino i tagli alfa-beta.

Esercizio 5 (punti 5)

Si spieghi sinteticamente a che cosa serve l'unificazione, si schematizzi il funzionamento dell'algoritmo di unificazione e si spieghi in cosa consiste l'occur check.

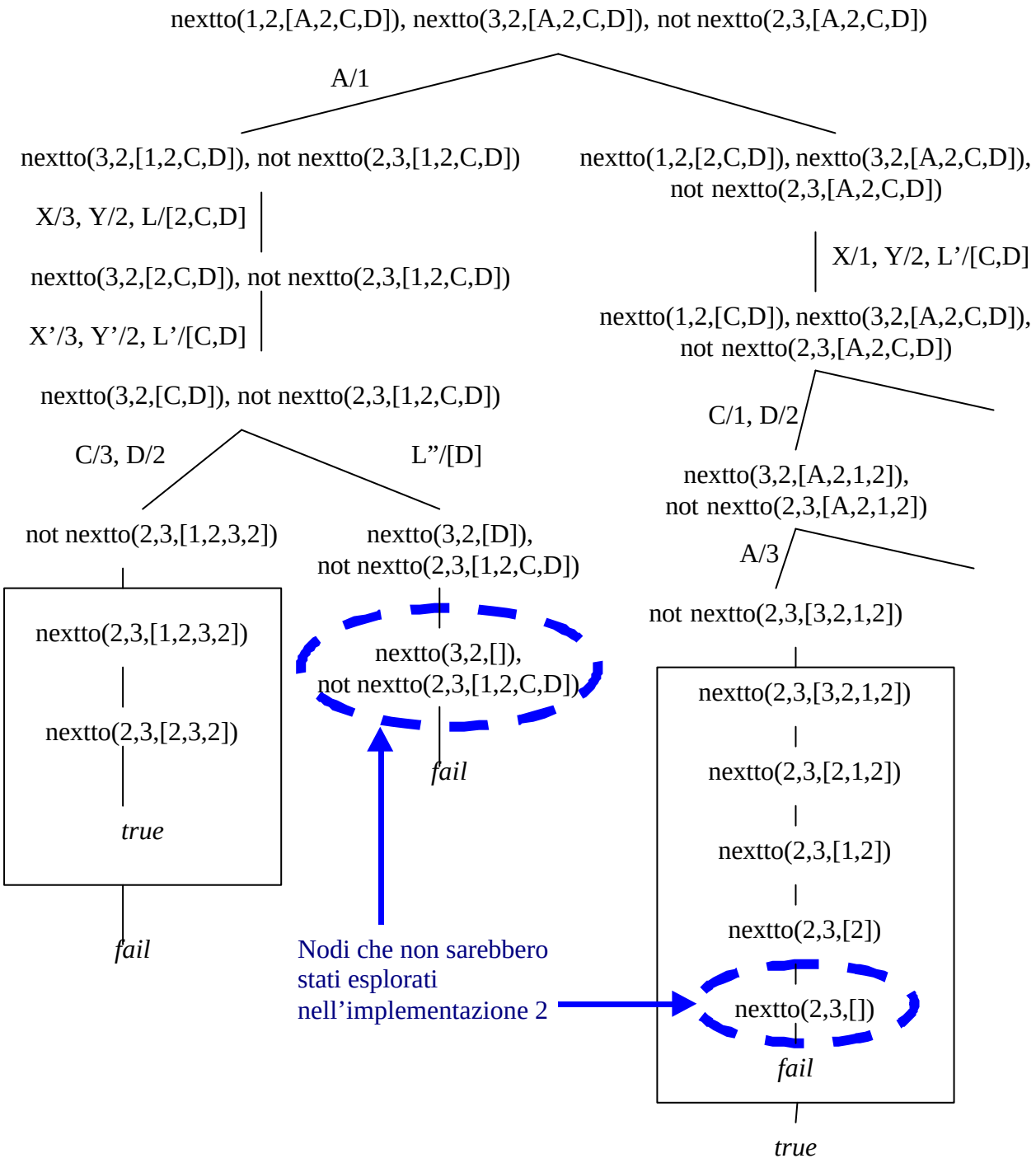
SOLUZIONE

Esercizio 1

```
conclun(Lin1, Lin2, Y):-
     lenght(Lin1,N1), lenght(Lin2,N2), Y is N1+N2.

length([],0).
length([_|T],N) :-
     length(T,N1), N is N+1.
```

Esercizio 2



Nella seconda implementazione la semantica dichiarativa non cambia. L'unica differenza è che la seconda clausola unifica solo con atomi in cui il terzo argomento è una lista con almeno due elementi. Per questo la seconda implementazione risulta più efficiente, in quanto fa fallire immediatamente i rami in cui la lista contiene meno di due elementi.

Esercizio 3

- Gli esseri viventi del mondo sono animali o vegetali.

∀ X vivente(X) => animale(X) xor vegetale(X)

- Gli animali carnivori vogliono mangiare qualunque altro animale.

∀ X ∀ Y animale(X) and carnivoro(X) and animale (Y) => mangia(X,Y)

- Leo è un essere vivente animale carnivoro.

vivente(leo) and animale(leo) and carnivoro(leo)

- Tarzan è un essere vivente animale.

vivente(tarzan) and animale(tarzan)
Goal: ∃ X animale(X) and mangia(X,tarzan)

Forma a clausole:

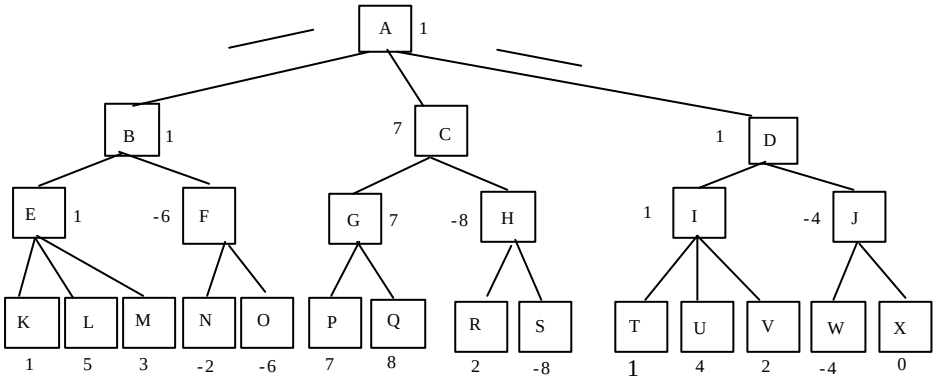
C1: not vivente(X) or animale(X) or vegetale(X)
C2: not vivente(X) or not animale(X) or not vegetale(X)
C3: not animale(X) or not carnivoro(X) or not animale(Y) or mangia(X,Y)
C4: vivente(leo)
C5: animale(leo)
C6: carnivoro(leo)
C7: vivente(tarzan)
C8: animale(tarzan)
Goal: not animale(X) or not mangia(X,tarzan)

Applicando il principio di risoluzione:

C9=Goal e C3: not animale(X) or not carnivoro(X) or not animale(tarzan)
C10=C5 e C9: not carnivoro(leo) or not animale(tarzan)
C11=C6 e C10: not animale(tarzan)
C12=C8 e C11: clausola vuota

Esercizio 4

Min-Max



Tagli alfa-beta