

**COMPITO DI INTELLIGENZA ARTIFICIALE (v.o.) – PARTE I**  
**FONDAMENTI DI INTELLIGENZA ARTIFICIALE**  
**13 Luglio 2005 (Tempo a disposizione 2 h, su 32 punti)**

**Esercizio 1 (punti 8)**

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.

**Esercizio 2 (punti )**

Si consideri il seguente programma Prolog:

```
sumlp([],0).
sumlp([X|T],S):-
    var(X),!,
    sumlp(T,S1), X is S-S1.
sumlp([X|T],S):-
    nonvar(S), X>S,!, fail.
sumlp([X|T],S):-
    sumlp(T,S1), S is S1+X.
```

dove *var* e *nonvar* sono predicati che hanno successo, rispettivamente, se l'argomento è una variabile e se non lo è (se è ground). Si rappresenti l'albero di derivazione SLD relativo al goal:

?- sumlp([A,1],5).

In particolare, si richiede di indicare quali rami sono tagliati per effetto del predicato cut (!).

**Esercizio 3 (punti 6)**

Si risolva il seguente esercizio, che applica l'algoritmo Alfa-beta al gioco del filetto.

Nel gioco del filetto (tic-tac-toe) supponiamo di usare la seguente funzione di valutazione euristica:

$$f(n) = 3X_2 + X_1 - (3O_2 + O_1)$$

dove  $X_n$  = numero di righe, colonne, diagonali con  $n$  X e nessuna O  
 $O_n$  = numero di righe, colonne, diagonali con  $n$  O e nessuna X

a) Espandere 2 livelli a partire dallo stato iniziale (schema vuoto), valutare gli stati risultanti, propagare all'indietro con la regola del MIN e MAX e scegliere la mossa migliore (muove per primo il giocatore X, che è ovviamente MAX).

b) Evidenziare cerchiandoli i nodi a livello 2 che non sarebbero valutati se si usasse la tecnica di potatura alfa-beta

c) Lo stesso del punto b) dopo avere riordinato i successori in modo che siano nell'ordine ottimale per la potatura alfa-beta.

NOTA: la funzione di utilità potrebbe essere:

$$\begin{array}{ll} +1 & \text{se } X_3=1 \\ -1 & \text{se } O_3=1 \end{array}$$

e quindi in accordo con la funzione di valutazione in quanto per vincere si devono prima mettere due simboli X in fila e questi stati sono valutati meglio degli altri.

#### **Esercizio 4 (punti 6)**

Si vuole scrivere un predicato Prolog `uguale(L1, L2)` che dia valore vero se le due liste `L1` ed `L2` contengono gli stessi elementi (anche se con differente ordine).

#### Esempio:

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

```
yes
```

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

```
no
```

#### **Esercizio 5 (punti 3)**

Dire se le seguenti formule sono valide o insoddisfacibili o nè l'una nè l'altra.

1.  $((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire}) \Leftrightarrow ((\text{Smoke} \Rightarrow \text{Fire}) \vee (\text{Heat} \Rightarrow \text{Fire}))$
2.  $(\text{Smoke} \Rightarrow \text{Fire}) \Rightarrow ((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire})$
3.  $((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire}) \Rightarrow (\text{Smoke} \Rightarrow \text{Fire})$
4.  $\text{Big} \vee \text{Dumb} \vee (\text{Big} \Rightarrow \text{Dumb})$
5.  $(\text{Big} \wedge \text{Dumb}) \vee \neg \text{Dumb}$

## SOLUZIONE

### Esercizio 1:

a1. Formalizzazione:

1.  $\forall X \text{ Stud}(X) \Rightarrow \exists Y \text{ Solve}(X, Y) \wedge \exists Z \neg \text{Solve}(X, Z)$
2.  $\exists X \text{ Ins}(X) \wedge \forall Y \text{ Solve}(X, Y)$
3. Da dimostrare:  $\exists X \text{ Ins}(X) \wedge \neg \text{Stud}(X)$

a2. Trasformazione in forma a clausole:

1.  $\forall X \text{ Stud}(X) \Rightarrow \exists Y \text{ Solve}(X, Y) \wedge \exists Z \neg \text{Solve}(X, Z)$   
 $\forall X \neg \text{Stud}(X) \vee (\exists Y \text{ Solve}(X, Y) \wedge \exists Z \neg \text{Solve}(X, Z))$   
[eliminazione  $\Rightarrow$ ]  
 $\forall X \neg \text{Stud}(X) \vee (\text{Solve}(X, p1(X)) \wedge \neg \text{Solve}(X, p2(X)))$   
[skolemizzazione]  
 $\neg \text{Stud}(X) \vee (\text{Solve}(X, p1(X)) \wedge \neg \text{Solve}(X, p2(X)))$   
[eliminazione  $\vee$ ]  
 $(\neg \text{Stud}(X) \vee \text{Solve}(X, p1(X))) \wedge (\neg \text{Stud}(X) \vee \neg \text{Solve}(X, p2(X)))$   
1.1  $\{\neg \text{Stud}(X1), \text{Solve}(X1, p1(X1))\}$   
1.2  $\{\neg \text{Stud}(X2), \neg \text{Solve}(X2, p2(X2))\}$

2.  $\exists X \text{ Ins}(X) \wedge \forall Y \text{ Solve}(X, Y)$

$\text{Ins}(i) \wedge \forall Y \text{ Solve}(i, Y)$

[skolemizzazione]

$\{\text{Ins}(i)\}$

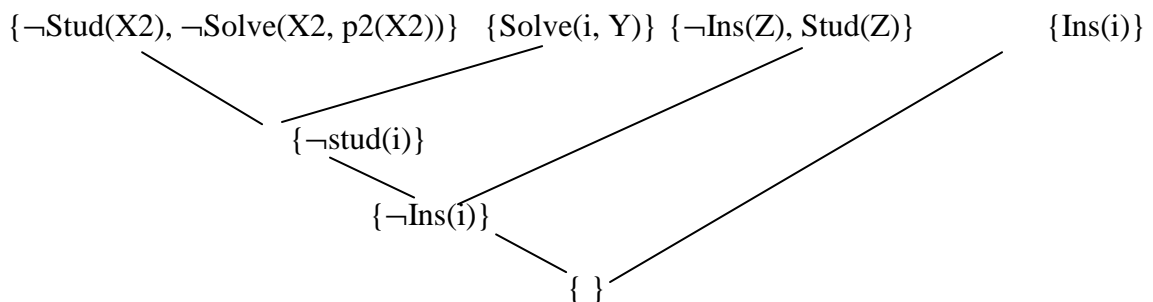
$\{\text{Solve}(i, Y)\}$

Goal negato:  $\neg \exists X \text{ Ins}(X) \wedge \neg \text{Stud}(X)$

$\forall X \neg \text{Ins}(X) \vee \text{Stud}(X)$

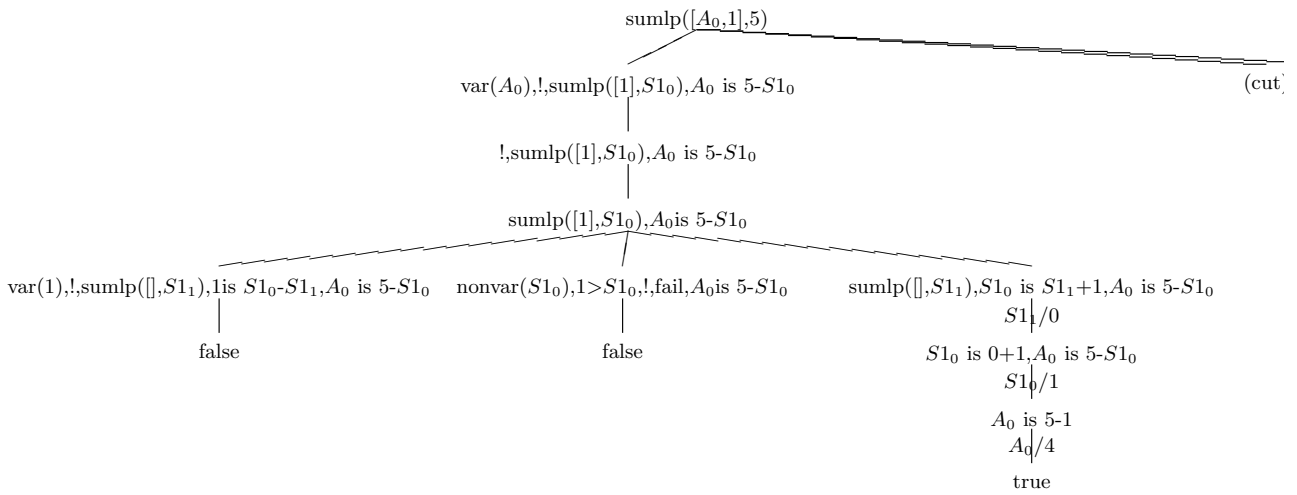
$\{\neg \text{Ins}(Z), \text{Stud}(Z)\}$

a3. Dimostrazione per refutazione:

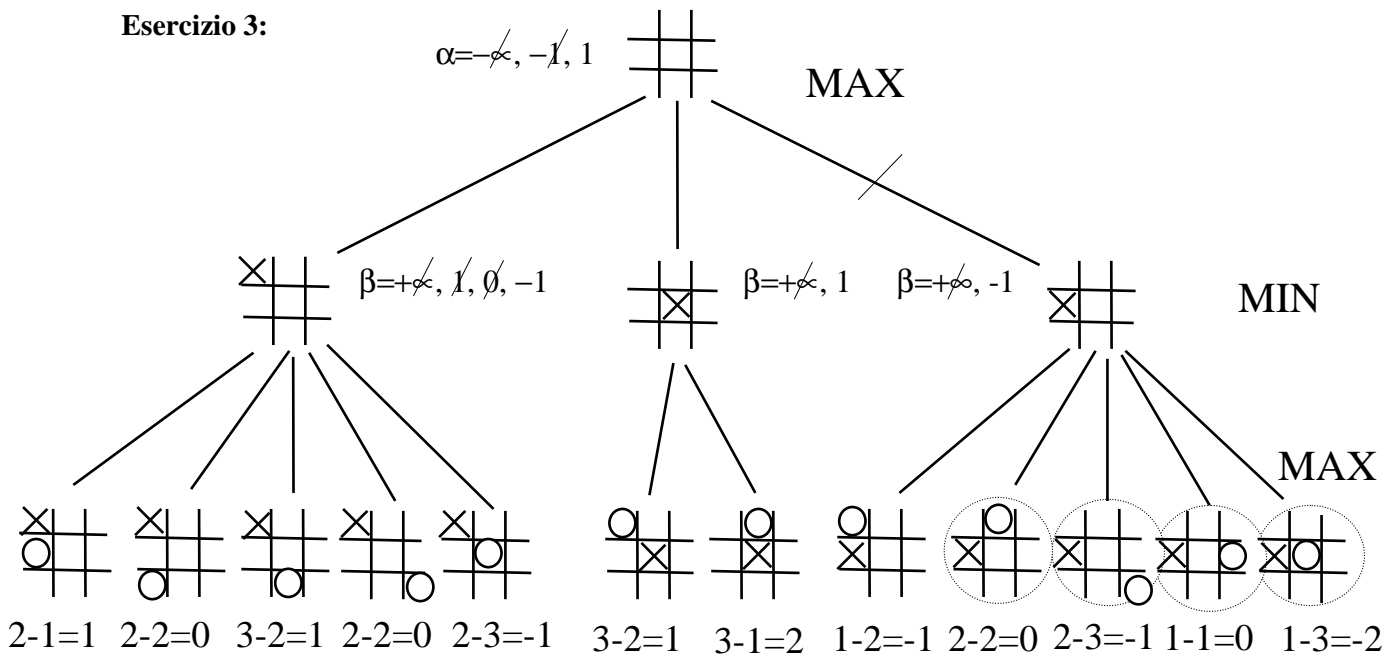


b. Non è possibile rendere come programma logico la KB iniziale in quanto la prima formula, trasformata in forma a clausole, non è una clausola Horn **definita** (la clausola 1.2 non contiene letterali positivi).

### Esercizio 2:



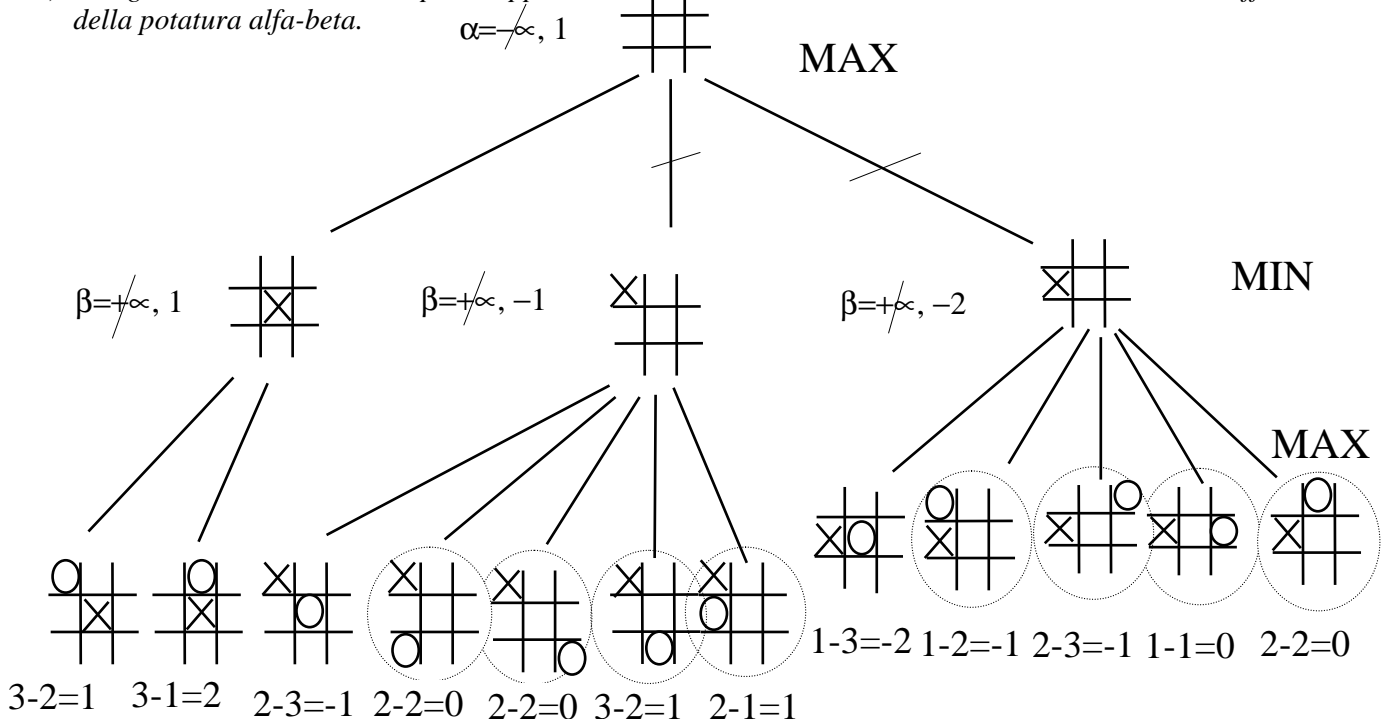
**Esercizio 3:**



a) la mossa scelta da MAX come prima mossa è la seconda, valutata 1.

b) gli stati non visitati con la potatura alfa-beta sono quelli cerchiati.

c) la seguente è la situazione dopo un opportuno riordinamento dei nodi in modo da massimizzare l'effetto della potatura alfa-beta.



**Esercizio 4:**

uguale([],[]).  
uguale([A|B],C):- member-del(A, C, T), uguale(B,T).

member-del(A, [A|C],C):-!.  
member-del(A,[B|C], [B|T]):- member-del(A,C,T).

**Esercizio 5:**

La formula 1 è valida.

La formula 2 è valida.

La formula 3 è altro perché per Smoke = T, Heat = T e Fire = T la formula è vera, mentre per Smoke = T, Heat = F e Fire = F la formula è falsa.

La formula 4 è valida.

La formula 5 è altro perché per dumb = F è vera mentre per Dumb = T e Big = F è falsa.