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

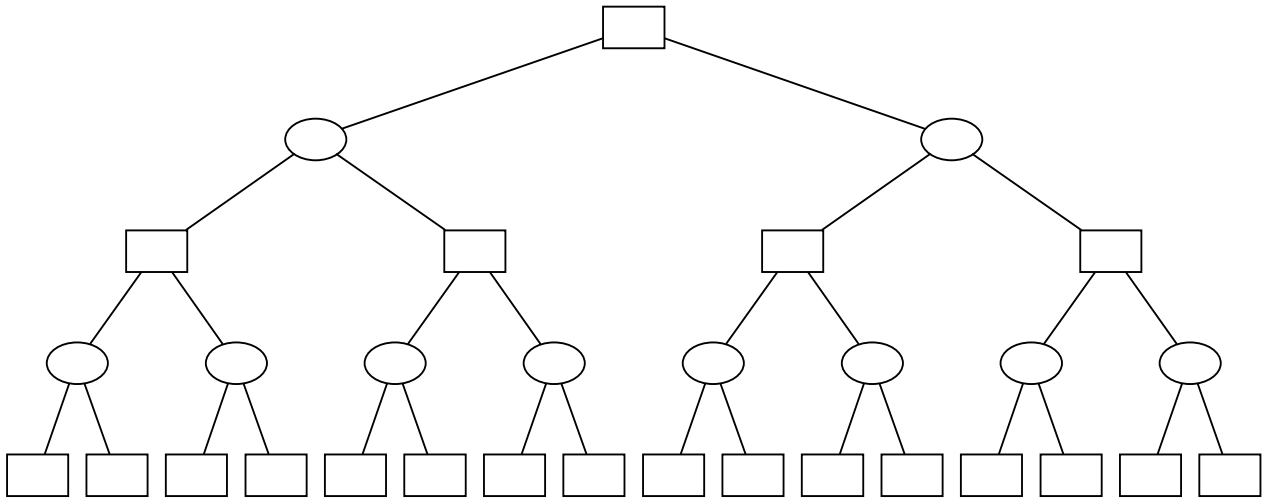
16 Dicembre 2008 (Tempo a disposizione 2h ; su 32 punti)

Esercizio 1 (punti 7)
Si rappresenti la seguente base di conoscenza in logica del primo ordine (si usino solo i seguenti predicati: *heavy*(*X*), *blue*(*X*), *green*(*X*)) e la si converta in forma normale congiuntiva adatta all'applicazione della risoluzione:

1. Se tutti gli oggetti pesanti sono blu, allora tutti quelli non pesanti sono verdi
2. Tutti gli oggetti sono o blu o verdi, ma non di entrambi i colori
3. Se esiste un oggetto non pesante, allora tutti quelli pesanti sono blu
4. L'oggetto o1 è pesante, l'oggetto o2 non è pesante

Si dimostri, per refutazione, tramite l'applicazione della risoluzione, che esiste un oggetto verde.

Esercizio 2 (punti 5)
Si consideri il seguente albero di gioco, dove i punteggi sono dal punto di vista del primo giocatore (Max):



Si mostri come l'algoritmo min-max risolve il problema. Si mostrino poi i tagli alfa-beta.

Esercizio 3 (punti 4)
Si scriva un predicato Prolog `power(X,N,V)` che è vero se `V` è `X` elevato alla `N`-esima potenza.

Esempi:

```
?-power(2,3,8).
yes
?-power(2,3,9).
no
?-power(0,3,0).
yes
?-power(2,0,1).
yes
?-power(3,3,V).
yes, V=27
```

Esercizio 4 (punti 6)
Il seguente programma Prolog verifica se un elemento appartiene ad una difference list:

```
variabile(X):- not(not(X=any)).
memdiff(X,L):- variabile(L),!,fail.
memdiff(X,[X|_]).
memdiff(X,[_|T]):- memdiff(X,T).
```

Si mostri l'albero SLDNF relativo all'invocazione del goal `memdiff(p(X),[Y,2|R]).`

Esercizio 5 (punti 7)
Si consideri il seguente gioco ispirato alla Torre di Hanoi, ma con quattro blocchi delle stesse dimensioni e numero di posizioni libere sul pavimento non limitato. Lo stato obiettivo è quello mostrato in figura nella parte destra. L'unica azione possibile consiste nello spostare un blocco (purché non si trovi sotto un altro blocco) sul pavimento o sopra un altro blocco.

Si risolva questo problema con la strategia di ricerca realizzata con l'algoritmo A*. Si assuma il costo di ogni azione pari a 1. Si utilizzi la seguente funzione euristica:
• 1 (se *A* non si trova su *B*) + 1 (se *B* non si trova su *C*) + 1 (se *C* non si trova su *D*) + 1 (se *D* non si trova sul pavimento).
Nello sviluppo dell'albero, per semplicità, non si generino nuovamente nodi già generati.

Esercizio 6 (punti 3)
Descrivere le tecniche partial e full look-ahead, il loro campo di applicazione e i vantaggi del loro utilizzo.

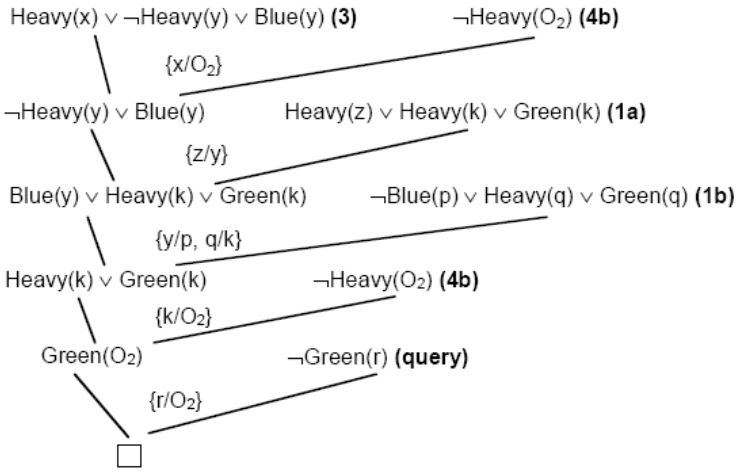
SOLUZIONE

Esercizio 1

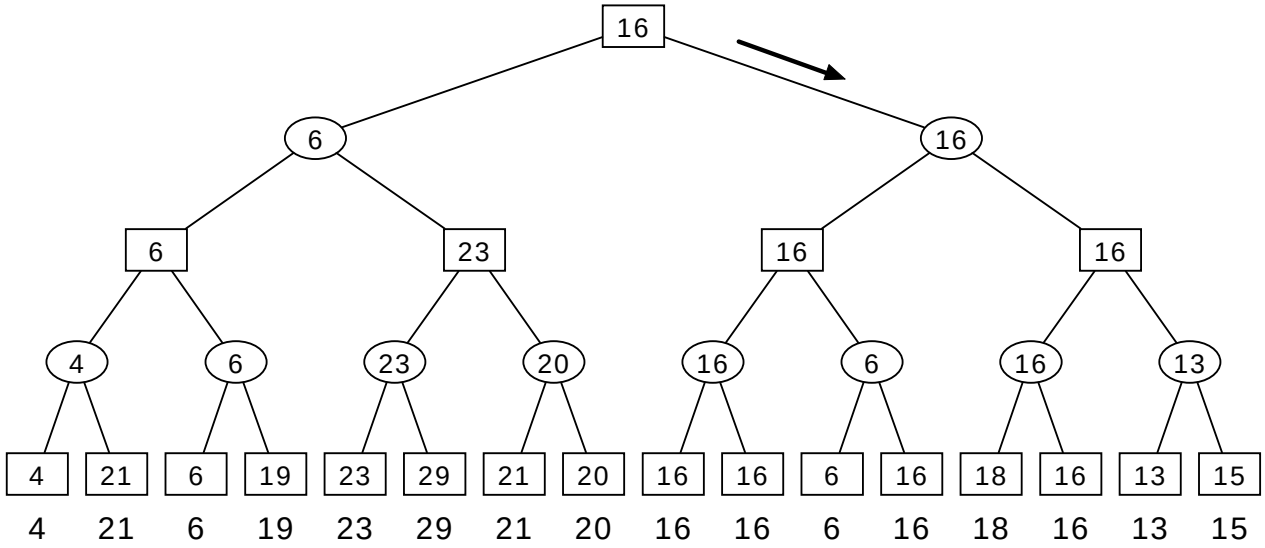
Nella soluzione, i predicati e le costanti sono scritti con la maiuscola e le variabili con la minuscola:

1. $[\forall x\ Heavy(x) \Rightarrow Blue(x)] \Rightarrow [\forall y\ \neg Heavy(y) \Rightarrow Green(y)]$,
   $\neg[\neg Heavy(x) \vee Blue(x)] \vee [Heavy(y) \vee Green(y)]$,
   $[Heavy(x) \wedge \neg Blue(x)] \vee [Heavy(y) \vee Green(y)]$,
   $[Heavy(x) \vee Heavy(y) \vee Green(y)] \wedge [\neg Blue(x) \vee Heavy(y) \vee Green(y)]$,
   $[Heavy(x) \vee Heavy(y) \vee Green(y)]$ (1a);
   $[\neg Blue(x) \vee Heavy(y) \vee Green(y)]$ (1b).
2. **$\forall x\ Blue(x) \vee Green(x)$** (2a);
   $\forall x\ \neg[Blue(x) \wedge Green(x)]$,
   $\neg Blue(x) \vee \neg Green(x)$ (2b).
3. $[\exists x\ \neg Heavy(x)] \Rightarrow [\forall y\ Heavy(y) \Rightarrow Blue(y)]$,
   $\neg[\exists x\ \neg Heavy(x)] \vee [\forall y\ \neg Heavy(y) \vee Blue(y)]$, $[\forall x\ Heavy(x)] \vee [\forall y\ \neg Heavy(y) \vee Blue(y)]$,
   $Heavy(x) \vee \neg Heavy(y) \vee Blue(y)$.
4. **$Heavy(O_1)$** (4a);
   $\neg Heavy(O_2)$ (4b).

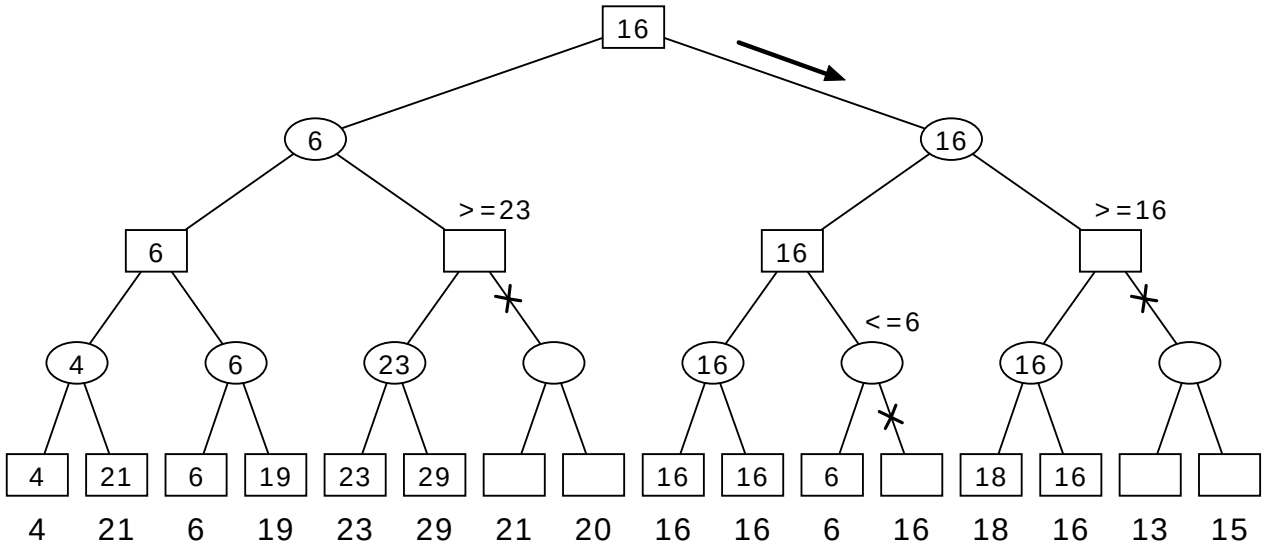
La query è $\exists x\ Green(x)$. Negandola si ottiene **$\forall x\ \neg Green(x)$** (è già in forma canonica).
L'albero di dimostrazione è il seguente.



Esercizio 2
Min-Max:



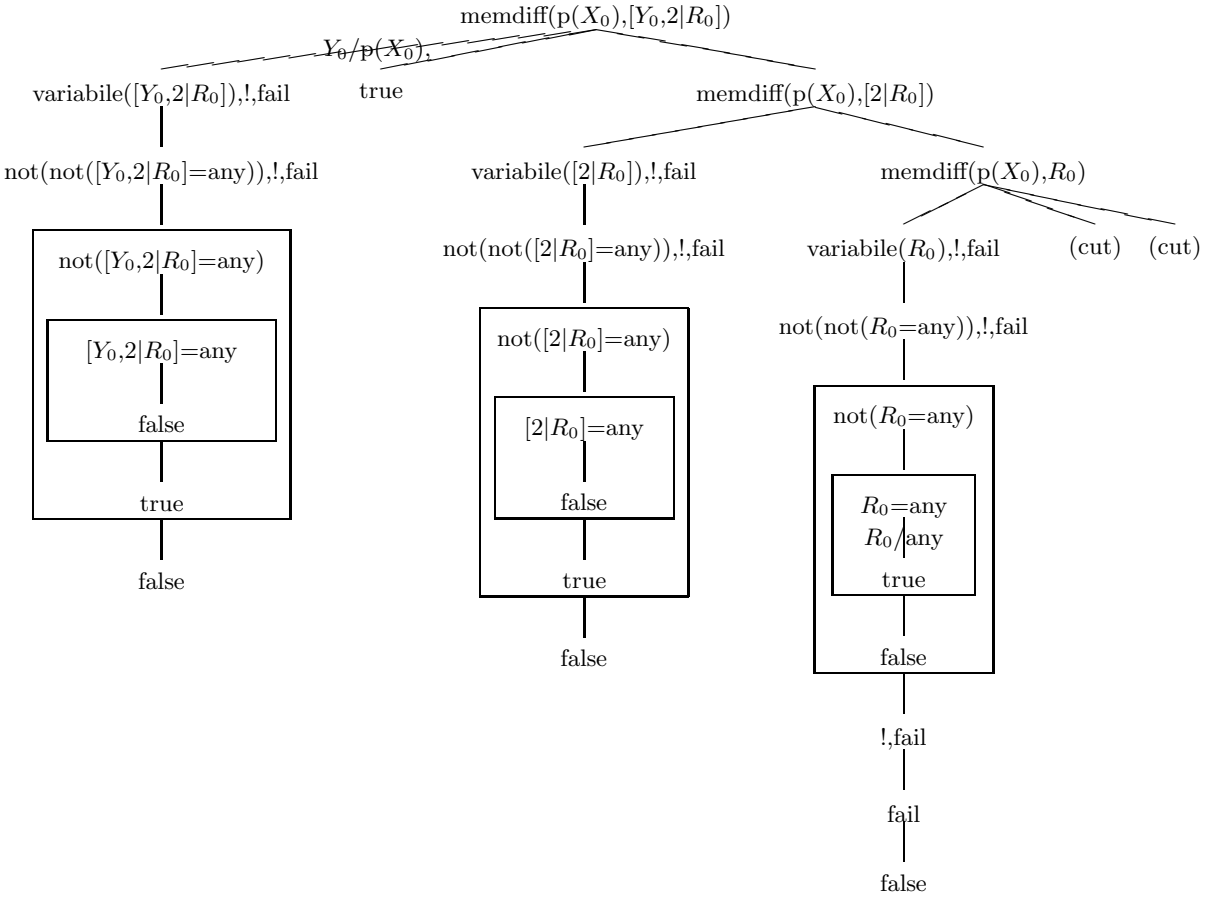
Alfa-Beta:



Esercizio 3

```
/* power(X,N,V) is true if V is X to the Nth power.*/
power(0,N,0):-N>0.
power(X,0,1):- X>0.
power(X,N,V):-X>0, N>0, N1 is N - 1, power(X,N1,V1), V is V1*X.
```

Esercizio 4



Esercizio 5