FONDAMENTI DI INTELLIGENZA ARTIFICIALE (8 CFU)

10 Luglio 2014 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

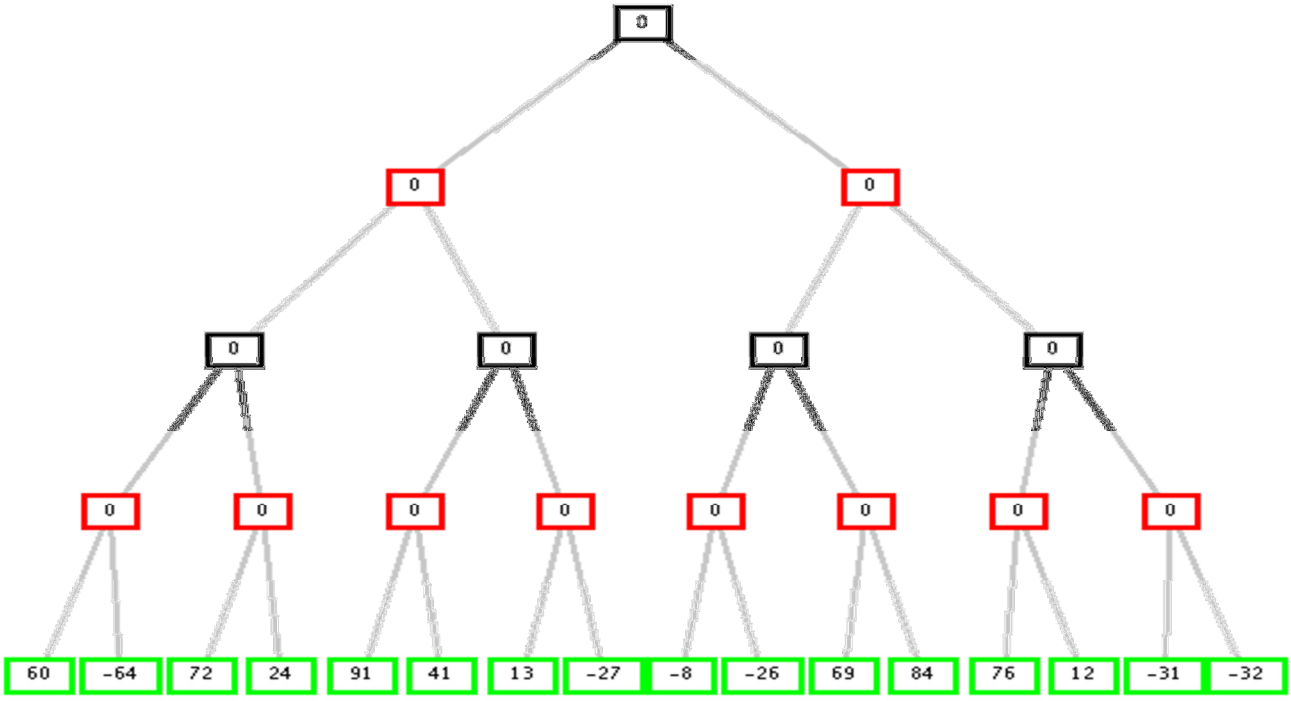
Si esprimano in logica dei predicati del I ordine le seguenti frasi:

1. Lucy è un professore
2. Tutti i professori sono persone
3. Fuchs è il preside
4. I presidi sono professori
5. Tutti i professori considerano il decano un amico o non lo conoscono (or inclusivo)
6. Per ognuno esiste qualcuno che è loro amico (suggerimento per la traduzione: Per ogni X esiste un Y tale che Y è amico di X)
7. Se una persona critica un'altra persona, allora la seconda non è amica della prima
8. Lucy critica Fuchs

Si dimostri, per refutazione, tramite l'applicazione della risoluzione, che Fuchs non è amico di Lucy. Si usi il seguente vocabolario: predicati: prof(X), person(X), dean(X), friend-of(X,Y) con significato "X è amico di Y", knows(X,Y) con significato "X conosce Y", criticize(X,Y); costanti: lucy, fuchs.

Esercizio 2 (4 punti)

Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore (*MAX*). Si mostri come gli algoritmi *min-max* e *alfa-beta* risolvono il problema.



Si mostri come l'algoritmo min-max risolve il problema, indicando con una freccia la strada scelta dall'algoritmo. Si mostrino poi i tagli alfa-beta.

Esercizio 3 (4 punti)

Definire nel linguaggio *Prolog* il predicato `positivi(N,P)` , che risulta vero quando `P` rappresenta la sottolista dei numeri positivi (maggiori di 0) della lista `N`. Esempio:

```
?- positivi([-1,2,4,-5], P).
yes, P=[2,4]

?- positivi([1,5], []).
no
```

Esercizio 4 (6 punti)

Nel seguente programma Prolog è rappresentato un grafo orientato, in cui gli archi che collegano direttamente due nodi sono elencati come insieme di fatti:

```
arc(a,b).  % 1
arc(a,c).  % 2
arc(b,c).  % 3
arc(b,d).  % 4
arc(c,d).  % 5
```

Il predicato `path(X,Y,L)` è vero se esiste un cammino aciclico (rappresentato come lista `L`) dal nodo `X` al nodo `Y`:

```
path(X,Y,L) :- path_aux(X,Y,[X],L).                % 6
path_aux(X, Y, Lin, [X,Y]) :-                      % 7 Il terzo argomento è la
       arc(X,Y), \+(member(Y,Lin)).                %   lista dei nodi visitati
path_aux(X,Y,Lin,[X|R1]) :-                        % 8
       arc(X,Z), \+(member(Z,Lin)),
       path_aux(Z,Y,[Z|Lin],R1).
```

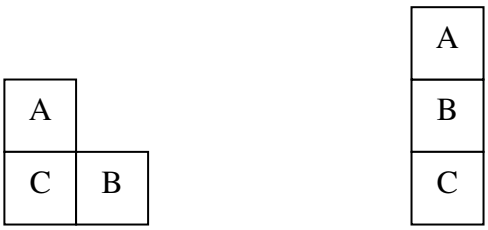
Si disegni l'albero SLD per il goal:

```
?-path(b,d,P).
```

fino alle prime due soluzioni, e si indichino le prime due risposte calcolate dal programma. Per semplicità, si consideri il predicato member/2 come built-in (cioè non si mostri, solo per member/2, l'albero sld). Ovviamente, si mostri l'albero sld relativo all'operatore di NAF.

Esercizio 5 (6 punti)

Si consideri il gioco ispirato della Torre di Hanoi, con tre blocchi (A, B, C) 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 indichino le azioni con *(x,y)*, con il significato *"sposta il blocco x su y", dove x può indicare A, B o C, e y può indicare A, B, C o floor.* 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 *floor*)

Ad esempio, nello stato iniziale il valore di *h* è 2. Nello sviluppo dell'albero, per semplicità, non si generino nuovamente nodi già generati. Si indichi l'ordine di espansione dei nodi, oltre al valore della funzione euristica, a fianco di ciascun nodo.

Esercizio 6 (3 punti)

Descrivere l'algoritmo di Forward Checking, e applicarlo al seguente problema CSP:

X1,X2, X3::[1,2,3,4]
X1+3>X2
X2<X3

dopo avere istanziato la variabile X1 al primo valore del dominio. Non si istanzino le altre due variabili. Come cambia la riduzione dei domini applicando invece il Partial Look Ahead dopo avere istanziato al valore 1 la variabile X1?

Esercizio 7 (3 punti)

Si descriva brevemente il sistema STRIPS e il funzionamento del suo motore di inferenza.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE
10 Luglio 2014 – Soluzioni

Esercizio 1

Formule in FOL:

- prof(lucy)
- $\forall$ x (prof(x) $\rightarrow$ person(x))
- dean(fuchs)
- $\forall$ x (dean(x) $\rightarrow$ prof(x))
- $\forall$ x ($\forall$ y (prof(x) $\wedge$ dean(y) $\rightarrow$ friend-of(y,x) $\vee$ $\neg$ knows(x, y)))
- $\forall$ x ($\exists$ y (friend-of (y, x)))
- $\forall$ x ($\forall$ y (person(x) $\wedge$ person(y) $\wedge$ criticize (x,y) $\rightarrow$ $\neg$ friend-of (y,x)))
- criticize(lucy,fuchs)

Goal:
$\neg$ friend-of(fuchs,lucy)

Traduzione in clausole:

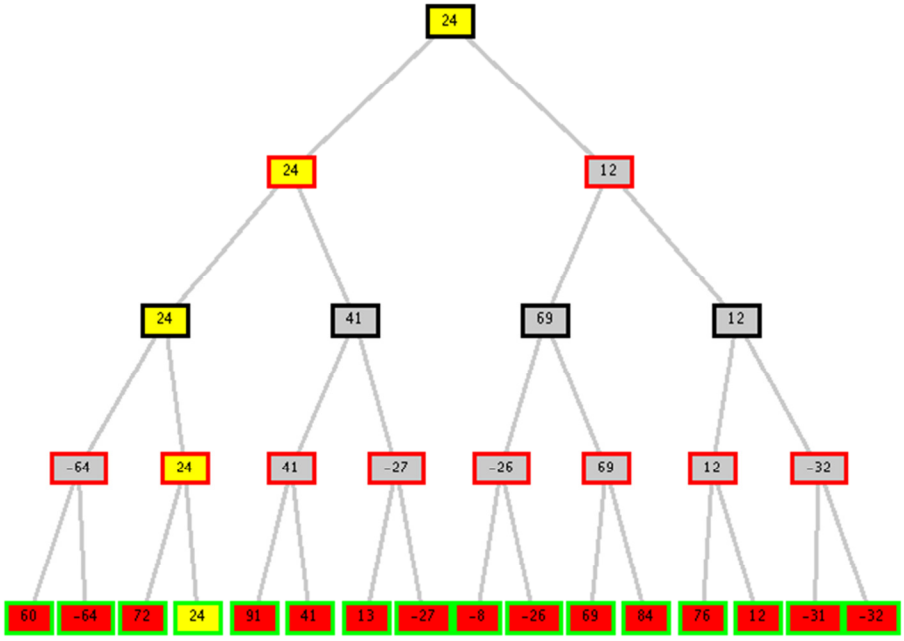
C1: prof(lucy)
C2: $\neg$ prof(x) $\vee$ person(x)
C3: dean(fuchs)
C4: $\neg$ dean(x) $\vee$ prof(x))
C5: $\neg$ prof(x) $\vee$ $\neg$ dean(y) $\vee$ friend-of(y,x) $\vee$ $\neg$ knows(x, y)
C6: friend-of (f(x), x))) Skolem
C7: $\neg$ person(x) $\vee$ $\neg$ person(y) $\vee$ $\neg$ criticize (x,y) $\vee$ $\neg$ friend-of (y,x)))
C8: criticize(lucy,fuchs)
Gneg: friend-of(fuchs,lucy)

Risoluzione:

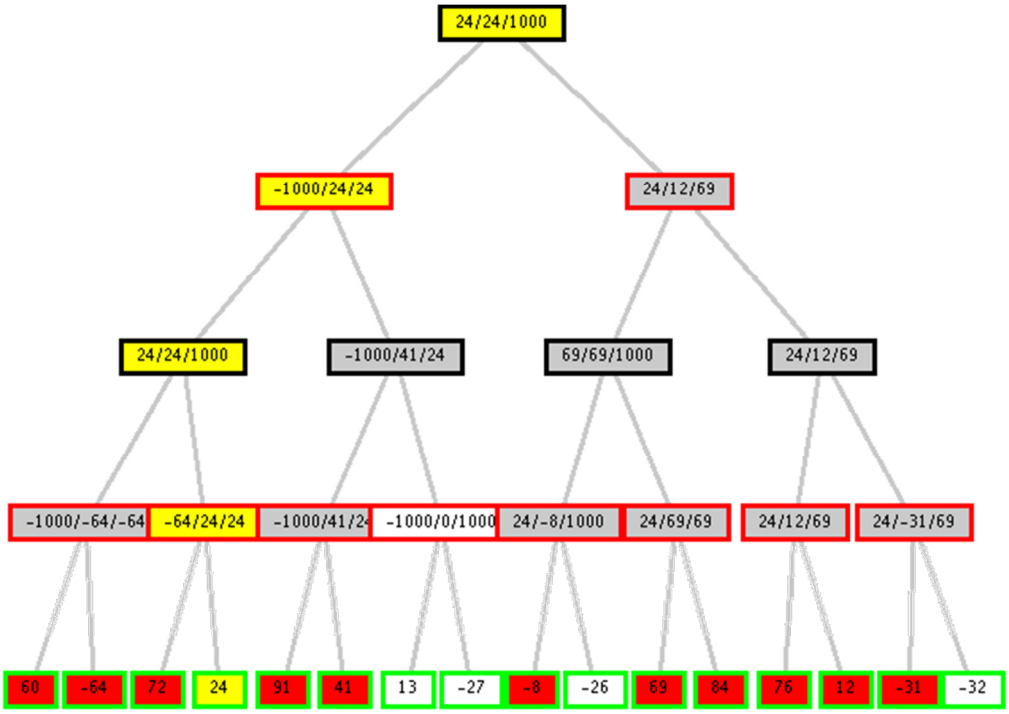
C9: Gneg+C7: $\neg$ person(lucy) $\vee$ $\neg$ person(fuchs) $\vee$ $\neg$ criticize (lucy,fuchs)
C10: C9+C8: $\neg$ person(lucy) $\vee$ $\neg$ person(fuchs)
C11: C10+C2: $\neg$ prof(lucy) $\vee$ $\neg$ person(fuchs)
C12: C11+C1: $\neg$ person(fuchs)
C13: C12+C2: $\neg$ prof(fuchs)
C14: C13+C4: $\neg$ dean(fuchs)
C15: C14+C3: contraddizione!

Esercizio 2

Min-Max:



Alfa-Beta:



I nodi in bianco sono quelli tagliati.

Esercizio 3

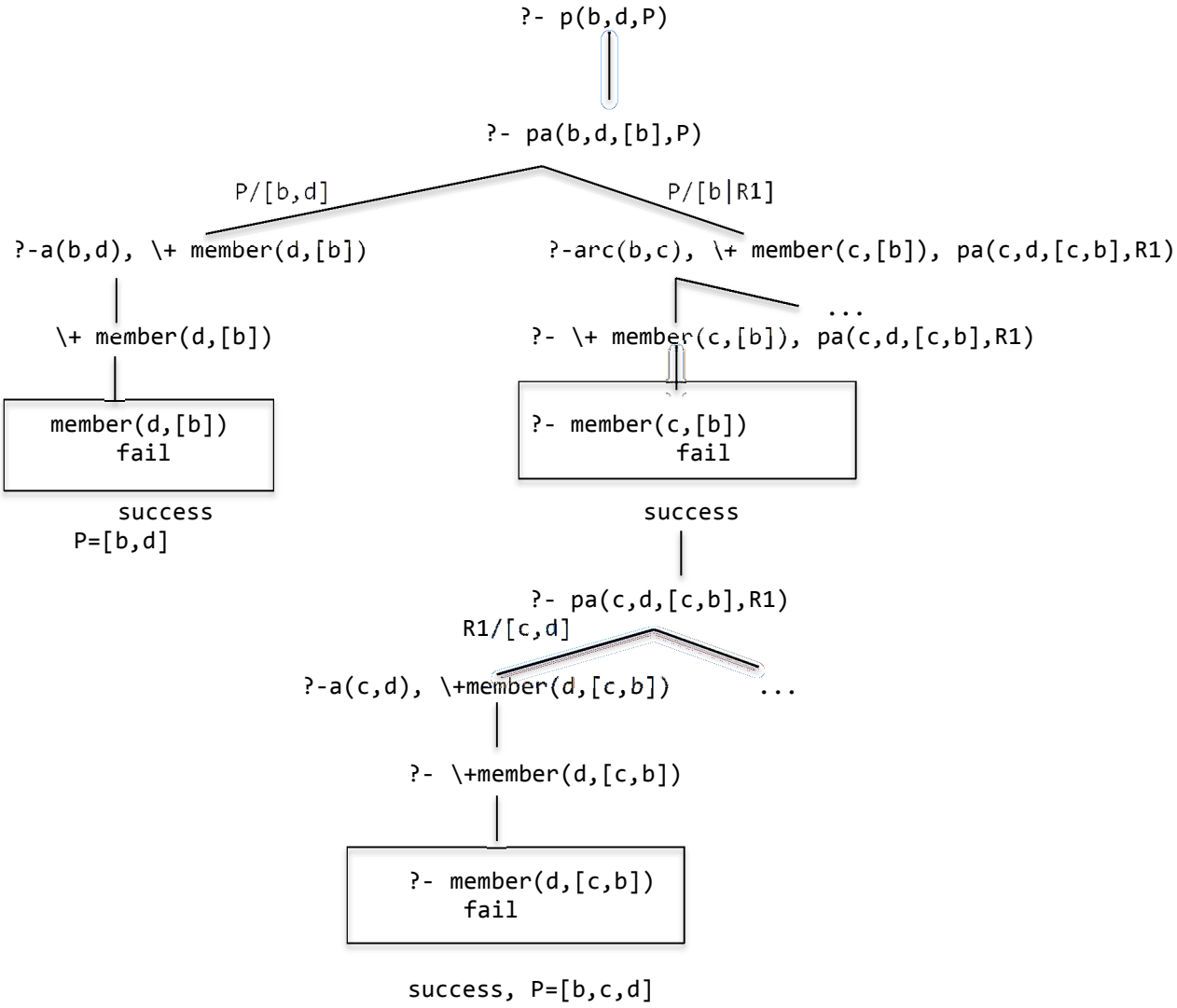
```
positivi([],[]).
positivi([N|Tn],[N|Tp]) :- N> 0, !, positivi(Tn,Tp).
positivi([N|Tn],P) :- positivi(Tn,P).
```

Esercizio 4

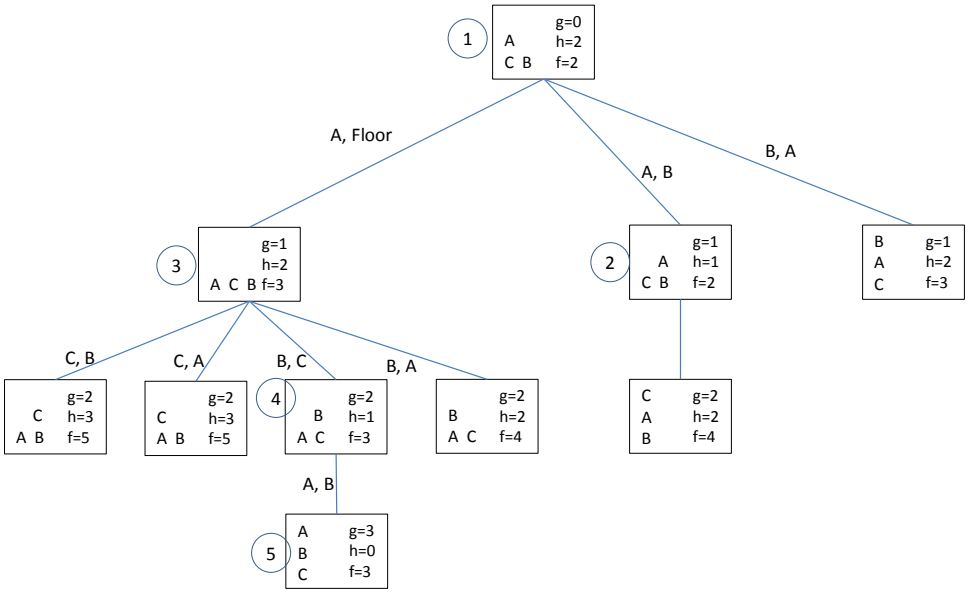
Le prime due risposte sono:

```
?- path(b,d,P).
P = [b, d] ;
P = [b, c, d] ;
```

L'albero SLD è il seguente, dove, per brevità "p" sta per path e "pa" sta per path_aux:



Esercizio 5



Esercizio 6

X1,X2, X3::[1,2,3,4]
X1+3>X2
X2<X3
Forward checking:

	X2	X3
X1=1	[1,2,3]	[1,2,3,4]

Partial look ahead:

	X2	X3
X1=1	[1,2,3]	[1,2,3,4]

Esercizio 7

Vedi slide del corso.