FONDAMENTI DI INTELLIGENZA ARTIFICIALE – PRIMA PARTE (6 CFU)

16 Giugno 2011 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (punti 6)

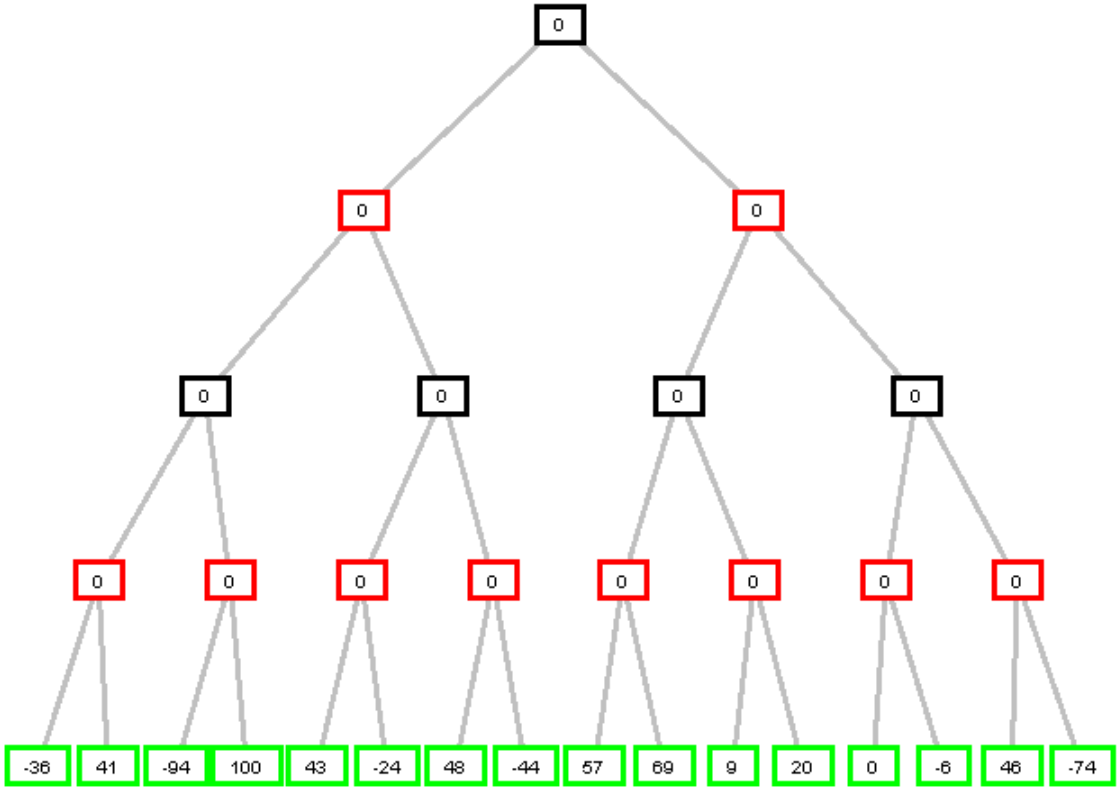
Scrivere le seguenti frasi in logica dei predicati del primo ordine:

- *Ogni persona lavora come infermiere oppure come insegnante* (xor)
- *Tutti gli infermieri sono maschi*
- *Tutte le persone sono maschi o femmine* (xor)
- *Steve è un maschio e Roberta è una femmina, entrambi sono persone.*

Trasformarle poi in clausole e dimostrare tramite risoluzione che Roberta insegna.

Esercizio 2 (punti 5)

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



Esercizio 3 (punti 6)

Il seguente programma Prolog calcola la componente connessa di un grafo:

```
comp([e,d],[e,d]) :- !.
comp(L,C) :- member(N,L), edge(N,X),
    not(membchk(X,L)), !,
    comp([X|L],C).
comp(L,L).
edge(a,b).
edge(b,c).
edge(d,e).
member(X,[X|T]).
member(X,[H|T]) :- member(X,T).
membchk(X,[X|T]) :- !.
membchk(X,[H|T]) :- membchk(X,T).
```

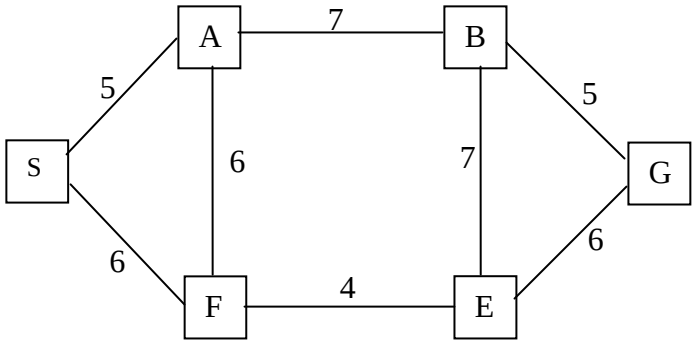
Si mostri l'albero di derivazione SLD relativo alla query

```
?- comp([d],C).
```

Esercizio 4 (punti 4)
Si scriva un predicato Prolog `adiacenti(X,Y,Z)` che è vero se `X` e `Y` sono adiacenti nella lista `Z`:

```
?-adiacenti(3,4,[1,2,3,4,5,6]).
yes
?-adiacenti(3,4,[1,2,3,5,4]).
no
?-adiacenti(3,4,[]).
no
```

Esercizio 5 (punti 7)
Si consideri il seguente grafo che rappresenta uno spazio di stati. `S` è lo stato iniziale, `G` lo stato goal. Gli archi sono etichettati con i costi.



Si può evitare di visitare nodi già visitati nel cammino dal nodo iniziale al nodo attuale. Quando serve, si usi la funzione di valutazione euristica $h(n)$ che segue, che fornisce una stima della distanza del nodo n dal goal. A parità di euristica si utilizzi l'ordine alfabetico.

N	S	A	B	E	F	G
h(*n*)	10	9	4	6	7	0

Per ognuno dei seguenti algoritmi si listino i nodi nell'ordine in cui sono espansi e i cammini soluzione trovati:

a. *Ricerca di costo uniforme*
b. *Ricerca greedy best-first*
c. *Ricerca A**

Esercizio 6 (punti 4)
Si descriva il concetto di *arc-consistenza* per un *constraint graph* possibilmente con almeno un esempio e l'algoritmo per ottenerla (AC-3).

VOTO:

- *Esame da 6 CFU, il voto è determinato da questa I parte*
- *Esame da 9 CFU, è la media pesata della I parte (che vale 2/3) e della II (che vale 1/3) ovvero il voto finale è dato da:* $\frac{2 \times \text{voto I parte} + \text{voto II parte}}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – SECONDA PARTE (3 CFU)

16 Giugno 2011 – Tempo a disposizione: 45 min – Risultato: 32/32 punti

Esercizio 7 (punti 4)

Si illustri brevemente con un esempio il vantaggio di avere annotazioni semantiche all'interno di informazioni disponibili sul Web.

Esercizio 8 (punti 9)

Si scriva un meta-interprete che dato un programma logico proposizionale (senza variabili) ed un goal, a partire da una lista vuota in ingresso, cerchi di risolvere il goal restituendo in uscita la lista dei predicati distinti invocati e risolti con successo.

Esempio: dato il programma

```
p :- q,r.
q :- r.
r.
```

la chiamata:

```
?- solve(p,[],L)
```

restituisce il seguente risultato (tre chiamate distinte; `p`, `q`, `r`, risolte con successo; `r` è chiamato due volte – in `p` e in `q`– ma compare nella lista una sola volta):

```
Yes L=[p,q,r]
```

Esercizio 9 (punti 3)

Lo svolge solo chi non ha partecipato alla lez/esercitazione del 19 Novembre 2010 su Prolog e grammatiche

Data la seguente grammatica:

$G = (V_n, V_t, P, S)$

$V_n = \{E, T\}$

$V_t = \{ +, *, a, b, (,) \}$

P = { E::= T + E | T
T::= F *T | F
F::=a | b| (E) }

S = E

Si scrivano le clausole DCG corrispondenti, e il goal per verificare la correttezza sintattica della frase: **`(a+a)*b`**

VOTO:

- *Esame da 3 CFU, il voto è determinato da questa II parte*
- *Esame da 9 CFU, è la media pesata della I parte (che vale 2/3) e della II (che vale 1/3) ovvero il voto finale è dato da:* $\frac{2 \times \text{voto I parte} + \text{voto II parte}}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

SOLUZIONE PARTE I

Esercizio 1

Traduzione in predicati in logica del primo ordine:

- *Ogni persona lavora o come infermiere oppure come insegnante* (xor).
  `∀X person(X) → works(X,teacher) xor works(X,nurse)`
- *Tutti gli infermieri sono maschi*.
  `∀X works(X,nurse) → male(X)`
- *Tutte le persone sono o maschi o femmine* (xor).
  `∀X person(X) → male(X) xor female(X)`

- *Steve è un maschio e Roberta è una femmina, entrambi sono persone*.
  ```
  male(steve),
  female(roberta),
  person(steve).
  person(roberta).
  ```

- Negazione del Goal: *Roberta non è un'insegnante.*
  `¬works(roberta,teacher)`

Traduzione in clausole:

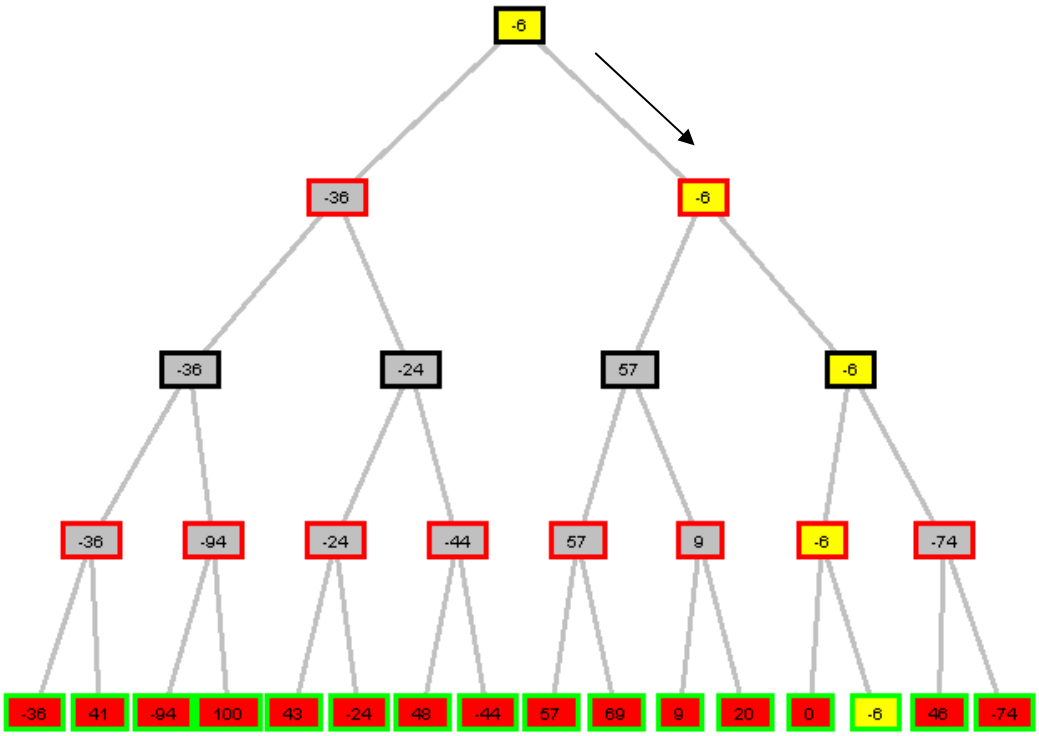
1. `¬person(X) or works(X,teacher) or works(X,nurse).`
2. `¬person(X) or ¬works(X,teacher) or ¬works(X,nurse).`
3. `¬works(X,nurse) or male(X).`
4. `¬person(X) or male(X) or female(X).`
5. `¬person(X) or ¬male(X) or ¬female(X).`
6. `male(steve).`
7. `female(roberta).`
8. `person(steve).`
9. `person(roberta).`
10. `¬works(roberta,teacher)`

Risoluzione:

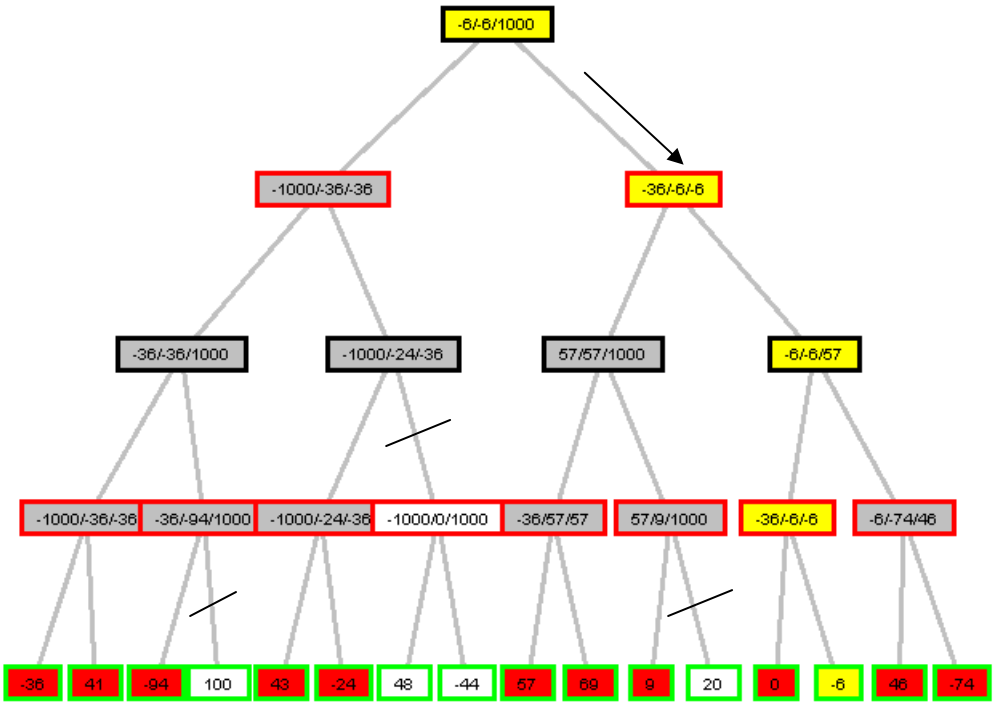
10 + 1 = 11 `¬person(roberta) or works(roberta,nurse).`
11 + 9 = 12 `works(roberta,nurse).`
12 + 3 = 13 `male(roberta).`
13 + 5 = 14 `¬person(roberta) or ¬female(roberta).`
14 + 9 = 15 `¬female(roberta).`
15 + 7 = *Contraddizione*

Esercizio 2

min-max:

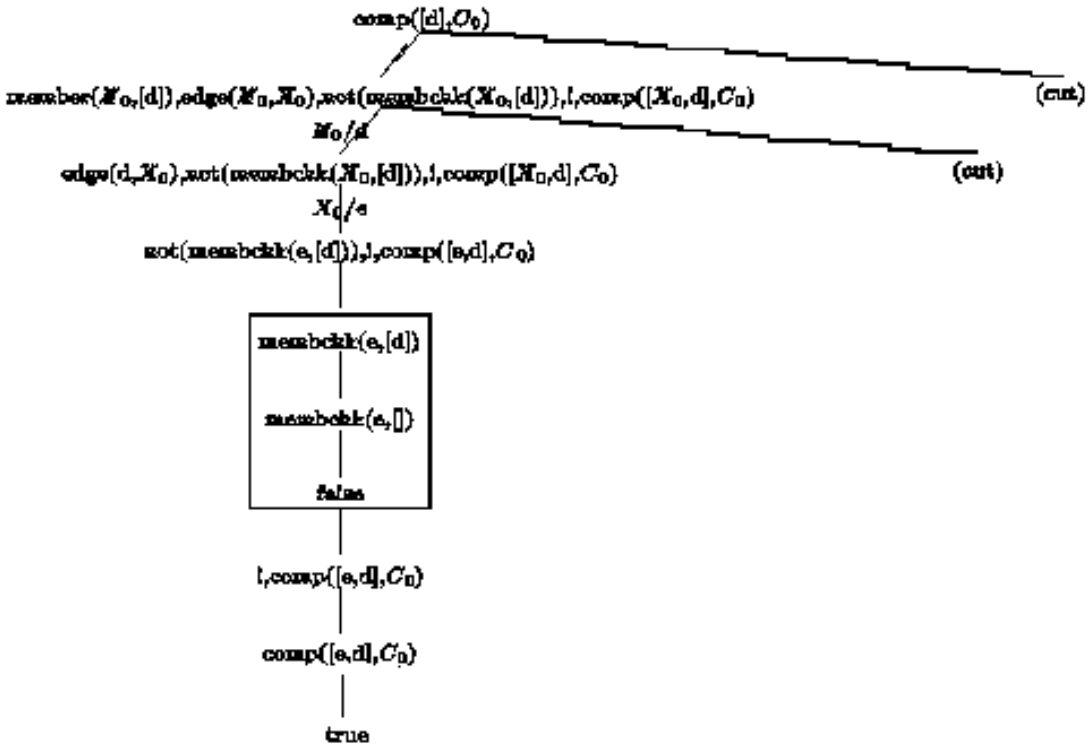


Alfa-beta:



I nodi non colorati sono quelli che vengono tagliati nell'algoritmo alfa-beta.

Esercizio 3



Esercizio 4

```
%% adjacent(X,Y,Zs) is true if the elements X and Y are adjacent in the list Zs.
adjacent(X,Y,[X,Y|Zs]).
adjacent(X,Y,[Z|Zs]):-adjacent(X,Y,Zs).
```

Esercizio 5

- *Ricerca di costo uniforme*
  - Ordine di espansione: S, A, F, E, F, A, B, E, G
  - Cammino soluzione: S, F, E, G
- *Ricerca greedy best-first*
  - Ordine di espansione: S, F, E, G
  - Cammino soluzione: S, F, E, G
- *Ricerca A**
  - Ordine di espansione: S, F, A, B, E, G
  - Cammino soluzione: S, F, E, G

SOLUZIONE PARTE II

Esercizio 7 - Description Logics
Soluzione... vedi slides

Esercizio 8 - Metainterprete

```
solve(GOAL,Lin,Lout)
%% Lout sono i predicati distinti risolti per arrivare al successo di
GOAL
solve(true,L,L).
solve((A,B), Lin, Lout) :- solve(A,Lin,L1),
                           solve(B,L1,Lout).
solve(A,L,L) :- system(A), !, call(A).
solve(A,[A|T],Lout) :- clause(A,B),
solve(A,Lin,Lout) :- clause(A,B),
                     member(A,Lin), !,
                     solve(B,Lin,Lout).
solve(A,Lin,Lout) :- clause(A,B),
                     solve(B,[A|Lin],Lout).
```

Nota: versione con 2 soli parametri (invocabile con `?- solve(p,L)`)

```
solve(GOAL,L)
%% L sono i predicati distinti risolti per arrivare al successo di GOAL
%% predicato "union" x l'unione di due insiemi: vedi slides
solve(true,[]).
solve((A,B), L) :- solve(A, L1),
                   solve(B, L2), union(L1,L2,L).
solve(A,[]):- system(A), !, call(A).
solve(A,L) :- clause(A,B),
              solve(B,L2),
              union([A],L2,L).
```

Esercizio 9 – DCG

```
expr --> term.
expr --> term, [+], expr.
term --> factor.
term --> factor, [*], term.
factor --> [a].
factor --> [b].
factor --> ['('],expr, [')'].
```

Goal:

```
?-  expr([(a+a)*b],[]).
yes
```