FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 1° parte (6 CFU)

15 Febbraio 2013 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (7 punti)

Si considerino le seguenti frasi:

1. *Daria è una donna*
2. *Elena è una donna*
3. *Ciro è genitore di Daria*
4. *Ciro è genitore di Berto*
5. *Anna è genitore di Berto*
6. *Elena è moglie di Berto*
7. *Daria è moglie di Franco*
8. *Se una donna ha un fratello che è sposato con qualcuna, quella donna è sua cognata*
9. *Se qualcuno è genitore di due persone, quelle persone sono fratelli*
10. *Se una persona è moglie di qualcuno, quel qualcuno ha sposato quella persona*
11. *Se una persona è moglie di qualcuno, quella persona ha sposato quel qualcuno*

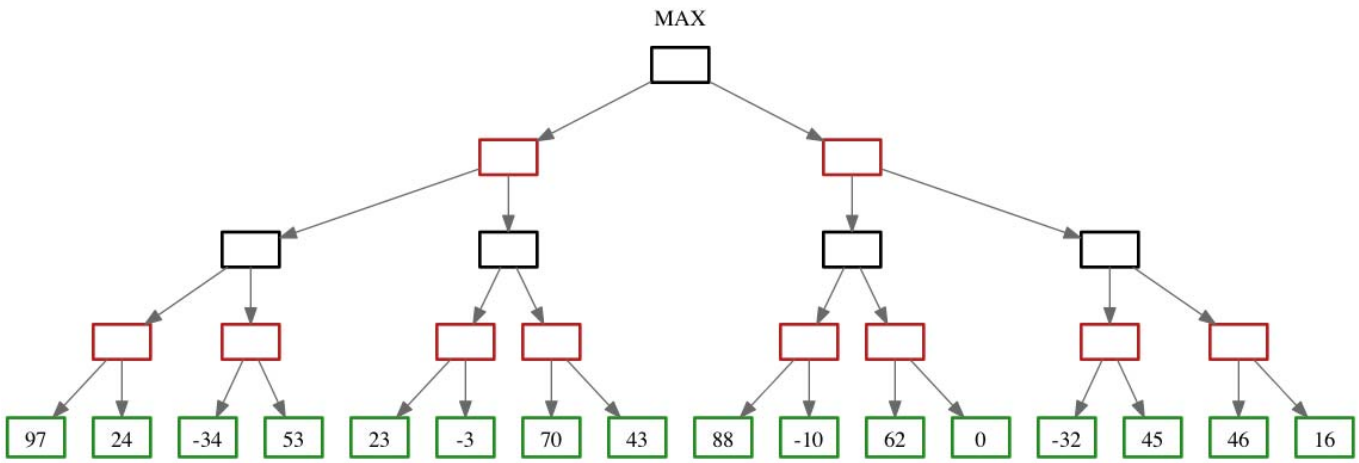
Si formalizzino in *logica dei predicati del I ordine*, utilizzando i seguenti predicati:

- `donna(X)` X è una donna
- `genitore(X,Y)` X è genitore di Y
- `moglie(X,Y)` X è moglie di Y
- `fratello(X,Y)` X è fratello (o sorella) di Y
- `sposato(X,Y)` X è sposato con Y
- `cognata(X,Y)` X è cognata di Y

Infine si trasformino in *clausole* e si dimostri tramite *risoluzione* che *Daria è cognata di Elena*.

Esercizio 2 (5 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.



Esercizio 3 (6 punti)

Il seguente programma *Prolog* restituisce una lista ordinata di termini che è la fusione delle due liste ordinate di termini date.

```
mergesort(X, [], X).
mergesort([], Y, Y).
mergesort([X|TX], [Y|TY], [X|T]) :- X=<Y, !, mergesort(TX, [Y|TY], T).
mergesort([X|TX], [Y|TY], [Y|T]) :- mergesort([X|TX], TY, T).
```

Si mostri l'*albero SLD-NF* relativo al goal:

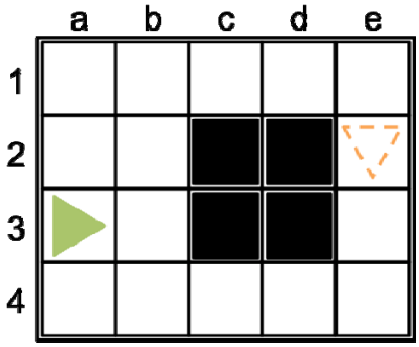
```
?- mergesort([3,7], [4], X).
```

Esercizio 4 (4 punti)

Dato un grafo semplice aciclico, si scriva un predicato `percorso/3` che, dati due nodi `a` e `b`, determini se nel grafo esiste un percorso che partendo da `a` raggiunge `b`, restituendo in una lista i nodi percorsi. Il grafo è descritto da fatti del tipo `arco(x, y)` che indicano che nel grafo i nodi `x` ed `y` sono collegati direttamente da un arco non orientato (che può quindi essere percorso da `x` verso `y`, o viceversa).

Es.:
```
arco(a, c). arco(d, a). arco(d, b). arco(f, e).
?- percorso(a, b, L).   yes   L=[a, d, b]
?- percorso(a, e, L).   no
```

Esercizio 5 (7 punti)



Si consideri l'arena di gioco rappresentata in figura in cui le celle nere sono ostacoli e quelle bianche sono posizioni su cui l'agente rappresentato dal triangolo verde può muoversi. Il doppio bordo esterno è un limite invalicabile. L'agente è inizialmente nella cella `(a,3)` orientato verso destra e deve raggiungere la cella `(e,2)` con orientamento verso il basso. L'agente può compiere due sole azioni:

- Spostarsi nella cella vuota adiacente nella direzione del suo orientamento attuale (costo 1),
- Modificare il proprio orientamento di 90° in senso orario rimanendo nella stessa cella (gli orientamenti possibili sono: nord, est, sud ovest; il costo è 1).

Si risolva il problema utilizzando l'*algoritmo A* con eliminazione degli stati ripetuti*. Come euristica si usi la *distanza di Manhattan* tra la posizione dell'agente e la posizione finale `(e,2)` tenendo conto degli ostacoli: la distanza tra le celle `(b,3)` ed `(e,3)`, ad esempio, è 5. Si riportino i primi 10 nodi dell'albero di ricerca (ad esclusione del nodo radice e dei nodi ripetuti), indicandone l'ordine di esplorazione. Se due o più nodi nella frontiera hanno lo stesso costo totale, si espandano prima i nodi generati per primi.

Esercizio 6 (3 punti)

Si descrivano le strategie CSP di *Generate & Test* (*GT*) e di *Standard Backtracking (SB)* discutendone analogie e differenze.

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:* $\frac{2 \times voto\ I\ parte + voto\ II\ parte}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte (3 CFU)

15 Febbraio 2013 – Tempo: 30 + 15min – Risultato: 16/16 punti

Esercizio 7 (5 punti)

Si introduca cosa sono le *logiche descrittive*, come si differenziano dalla logica del I ordine e perché sono state introdotte.

Esercizio 8 (8 punti)

Si scriva un meta-predicato: `assert_lemma(Goal)` che, dato in ingresso un `Goal`, asserisce le soluzioni della chiamata `Goal` nel database *Prolog* come lemma. Ciascun lemma esclude le clausole successive per quel predicato (di fatto ha un *cut* nel corpo).

Ad esempio, supponendo di avere anche le clausole:

```
p(X, Y):-q(X,Y)
q(1, 2).
q(a, b).
```

l'invocazione seguente ha successo con esito come indicato:

```
?- assert_lemma(p(X, Y).
yes
```

e nel database Prolog sono aggiunti:

```
p(a,b):-!.
p(1,2):-!.
p(X, Y):-q(X,Y)
q(1, 2).
q(a, b).
```

Esercizio 9 (3 punti) – ulteriori 15 min

Nota: lo deve svolgere solamente chi non ha partecipato alle lezioni/esercitazioni su Prolog e grammatiche.

Data la seguente grammatica, che definisce cosa è un identificatore (un identificatore inizia con una lettera, 'a' o 'b' per limitarci a sole due lettere, seguita da cifre – 0 o 1 - o lettere comunque alternate, senza altri simboli; ad esempio, i nomi *a*, *a123*, *a1b, b1* sono identificatori, *1a* non lo è):

```
G = (Vn, Vt, P, S)
Vn = {ident, lettera, cifra, lettCifra}
Vt = {a,b,0,1}
P = { ident ::= lettera | lettera lettCifra
      lettera ::= a | b
      cifra ::= 0 | 1
      lettCifra ::= lettera | lettera lettCifra !
                    cifra  | cifra lettCifra    }
S = ident
```

Si scrivano le corrispondenti clausole in versione DCG estesa che verificano la correttezza di una frase (ad esempio l'identificatore `a123`).

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: 2*(votoParteI+votoParteII)/3 e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 1° parte

15 Febbraio 2013 – Soluzioni

Esercizio 1

Frasi in *logica dei predicati del I ordine* e loro traduzione in *clausole*:

a. *Daria è una donna*
   donna(daria)
   {donna(daria)}

b. *Elena è una donna*
   donna(elena)
   {donna(elena)}

c. *Ciro è genitore di Daria*
   genitore(ciro,daria)
   {genitore(ciro,daria)}

d. *Ciro è genitore di Berto*
   genitore(ciro,berto)
   {genitore(ciro,berto)}

e. *Anna è genitore di Berto*
   genitore(anna,berto)
   {genitore(anna,berto)}

f. *Elena è moglie di Berto*
   moglie(elena,berto)
   {moglie(elena,berto)}

g. *Daria è moglie di Franco*
   moglie(daria,franco)
   {moglie(daria,franco)}

h. *Se una donna ha un fratello che è sposato con qualcuno, quella donna è sua cognata*
   ∀X,Y,Z: donna(X) ∧ fratello(X,Z) ∧ sposato(Z,Y) ⇒ cognata(X,Y)
   {¬donna(X) ∨ ¬fratello(X,Z) ∨ ¬sposato(Z,Y) ∨ cognata(X,Y)}

i. *Se qualcuno è genitore di due persone, quelle persone sono fratelli*
   ∀X,Y,Z: genitore(Z,X) ∧ genitore(Z,Y) ⇒ fratello(X,Y)
   {¬genitore(Z,X) ∨ ¬genitore(Z,Y) ∨ fratello(X,Y)}

j. *Se una persona è moglie di qualcuno, quel qualcuno ha sposato quella persona*
   ∀X,Y: moglie(X,Y) ⇒ sposato(Y,X)
   {¬moglie(X,Y) ∨ sposato(Y,X)}

k. *Se una persona è moglie di qualcuno, quella persona ha sposato quel qualcuno*
   ∀X,Y: moglie(X,Y) ⇒ sposato(X,Y)
   {¬moglie(X,Y) ∨ sposato(X,Y)}

l. *Goal: Daria è cognata di Elena*
   cognata(daria,elena)
   {¬cognata(daria,elena)}

Teoria a clausole:

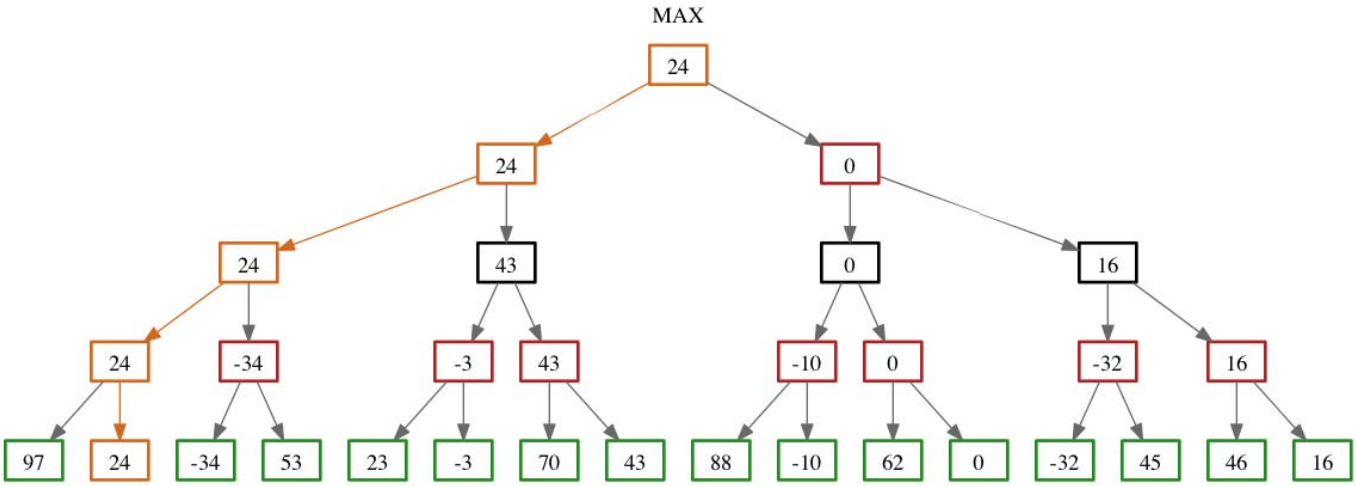
1. donna(daria)
2. donna(elena)
3. genitore(ciro,daria)
4. genitore(ciro,berto)
5. genitore(anna,berto)
6. moglie(elena,berto)

7. moglie(daria,franco)
8. ¬donna(X) ∨ ¬fratello(X,Z) ∨ ¬sposato(Z,Y) ∨ cognata(X,Y)
9. ¬genitore(Z,X) ∨ ¬genitore(Z,Y) ∨ fratello(X,Y)
10. ¬moglie(X,Y) ∨ sposato(Y,X)
11. ¬moglie(X,Y) ∨ sposato(X,Y)
12. ¬cognata(daria,elena)

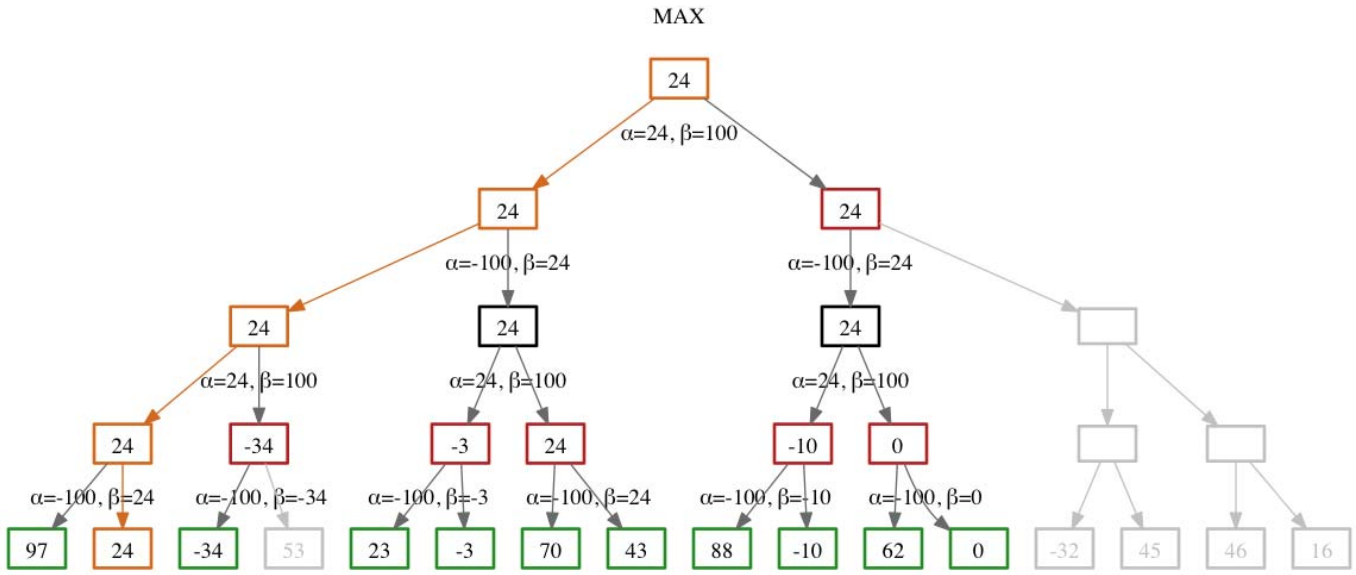
Risoluzione:

13. = 12. ∪ 8.	¬donna(daria) ∨ ¬fratello(d,Z) ∨ ¬sposato(Z,elena)	{X/d,Y/e}
14. = 13. ∪ 1.	¬fratello(d,Z) ∨ ¬sposato(Z,elena)	{}
15. = 14. ∪ 10.	¬fratello(d,Z) ∨ ¬moglie(e,Z)	{X/Z,Y/e}
16. = 15. ∪ 6.	¬fratello(d,Z)	{}
17. = 16. ∪ 9.	¬genitore(Z,daria) ∨ ¬genitore(Z,berto)	{X/d,Y/b}
18. = 17. ∪ 4.	¬genitore(ciro,daria)	{Z/c}
19. = 18. ∪ 3.	□	

Esercizio 2 - *min-max:*

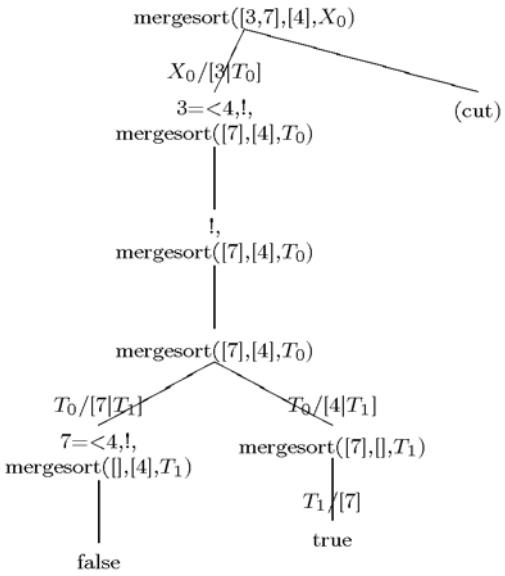


alfa-beta:



I nodi colorati in grigio sono quelli che vengono tagliati dall'algoritmo alfa-beta.

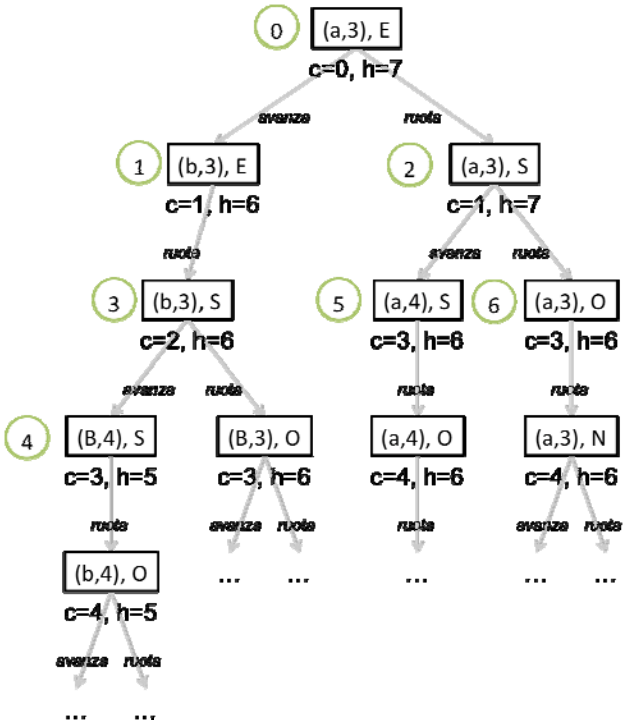
Esercizio 3 - *Albero SLD-NF:*



Esercizio 4

```
percorso(X, Y, [X, Y]) :-
      passo(X, Y).
percorso(X, Y, [X|Percorso]) :-
      passo(X, Z),
      percorso(Z, Y, Percorso).
passo(X, Y) :- arco(X, Y).
passo(X, Y) :- arco(Y, X).
```

Esercizio 5



Esercizio 6

Vedi slides del corso.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte
15 Febbraio 2013 – Soluzioni

Esercizio 7

Vedi slides del corso.

Esercizio 8

```
assert_lemma(Goal) :- call(Goal),
                      asserta((Goal:-!)),
                      fail.
assert_lemma(_).
```

Esercizio 9

```
ident --> lettera.
ident --> lettera, lettCifra.
lettera --> [a].
lettera --> [b].
cifra --> [0].
cifra --> [1].
lettCifra --> lettera.
lettCifra --> cifra.
lettCifra --> lettera, lettCifra.
lettCifra --> cifra, lettCifra.
```