FONDAMENTI DI INTELLIGENZA ARTIFICIALE – (8 CFU)

12 Giugno 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. Trilli e Attila sono gatti
2. Attila mangia le stesse scatolette che mangia Trilli
3. Trilli mangia le stesse scatolette che mangia Attila
4. Un gatto che mangia qualunque scatoletta è "goloso"
5. Attila non è goloso

Si dimostri, per refutazione, tramite l'applicazione della risoluzione, che Trilli non mangia tutte le scatolette (esiste una scatoletta che Trilli non mangia).

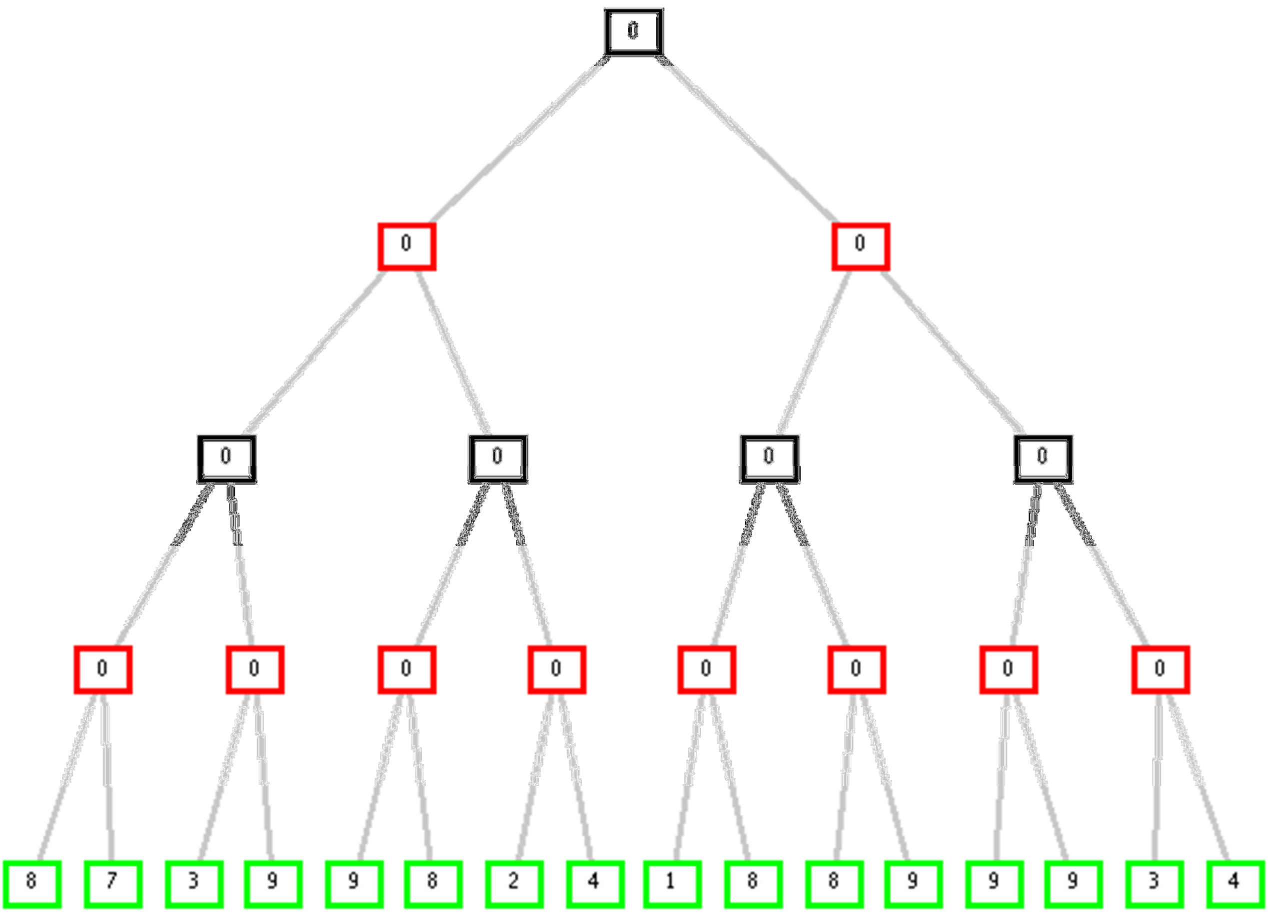
Si usi il seguente vocabolario: predicati: **gatto(X)**, **scatoletta(X)**, **mangia(X,Y)** con significato "X mangia Y", **goloso(X)**; costanti: **trilli**, **attila**.

Si riporta anche la traduzione della frase 4. Qualunque gatto, se mangia qualunque scatoletta, è un gatto "goloso":

$\forall$X [$\forall$Y gatto(X) /\ scatoletta(Y) => mangia(X,Y)] => goloso(X)

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.



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 `semisomma(L,S)` , che risulta vero quando S rappresenta la sommatoria degli elementi della lista L in posizione dispari. Per definizione, la semisomma di una lista vuota è zero.
Esempio:

```
?-semisomma[1,2,3,4,5],X).
yes X=9
```

Esercizio 4 (5 punti)

Si scriva un meta interprete di Prolog `solve(GoalList)` che, data una lista di goal atomici `GoalList` rappresentante una congiunzione di goal, determina se questa è soddisfatta. Il goal true è la lista vuota. Il meta-interprete inoltre deve gestire esplicitamente un predicato speciale, di nome "`abort`": qualora un sotto-goal unifichi con "abort", il programma deve fallire immediatamente (senza esplorare eventuali alternative date dai punti di scelta ancora aperti).

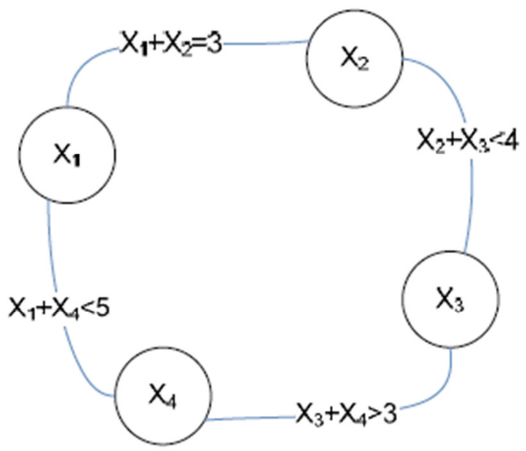
Ad esempio, il programma sia rappresentato come segue, con predicato rule(Head, ListaBody):

```
rule(p(X), [q(X)]).
rule(q(X), [s(X)]).
rule(s(1), []).
rule(r(2), []).
```

Le seguenti query vengono eseguite così:

```
?- solve( [p(A), r(B)]).
yes A=1, B=2
?- solve( []).
yes
?- solve( [p(A), abort] ).
no
```

Esercizio 5 (6 punti)



Sia dato il seguente problema di soddisfacimento di vincoli, dove le variabili X_1, X_2, X_3 e X_4 hanno tutte dominio [1,2,3], ed i vincoli tra esse sono rappresentati nella rete in figura:

Si applichi l'algoritmo di arc-consistenza al problema dato mostrando che l'algoritmo non riesce a determinare un unico valore per tutte le variabili.

Esercizio 6 (3 punti)

Dare il significato di euristica ammissibile, e enunciare le proprietà che ne derivano per l'algoritmo di ricerca A* che utilizzi una tale euristica.

Esercizio 7 (3 punti)

Si discuta come si gestisce la modifica di stato e il frame problem nei sistemi di pianificazione deduttivi basati sulla logica aiutandosi anche con un esempio.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE
12 Giugno 2014 – Soluzioni

Esercizio 1

1. *Trilli e Attila sono gatti*
   gatto(trilli) /\ gatto(attila)

2. *Attila mangia le stesse scatolette che mangia Trilli*
   ∀X scatoletta(X) /\ mangia(trilli, X) => mangia(attila,X)

3. *Trilli mangia le stesse scatolette che mangia Attila*
   ∀X scatoletta(X) /\ mangia(attila, X) => mangia(trilli,X)

4. Qualunque gatto, se mangia qualunque scatoletta, è un gatto "goloso"
   ∀X [∀Y gatto(X) /\ scatoletta(X) => mangia(X,Y)] => goloso(X)

5. Attila non è goloso
   ¬ goloso(attila)

Query:

∃X scatoletta(X) /\ ¬ mangia(trilli,X)

Trasformazione in clausole delle formule 1-3:

gatto(trilli)	(1a)
gatto(attila)	(1b)
¬scatoletta(X) \/ ¬mangia(trilli, X) \/ mangia(attila,X)	(2)
¬scatoletta(X) \/ ¬mangia(attila, X) \/ mangia(trilli,X)	(3)

4. *Qualunque gatto, se mangia qualunque scatoletta, è un gatto "goloso"*

∀X [∀Y gatto(X) /\ scatoletta(Y) => mangia(X,Y)] => goloso(X)
∀X [∀Y ¬gatto(X) \/ ¬scatoletta(Y) \/mangia(X,Y)] => goloso(X)
∀X ¬[∀Y ¬gatto(X) \/ ¬scatoletta(Y) \/mangia(X,Y)] \/ goloso(X)
∀X [∃Y gatto(X) /\ scatoletta(Y) /\ ¬mangia(X,Y)] \/ goloso(X)
Si introduce una funzione di Skolem f(X):
∀X [gatto(X) /\ scatoletta(f(X)) /\ ¬mangia(X,f(X))] \/ goloso(X)
Si distribuisce l'and:
∀X [gatto(X) \/ goloso(X)] /\ [scatoletta(f(X)) \/ goloso(X)] /\ [¬mangia(X,f(X)) \/ goloso(X)]
e si ottengono le clausole:

gatto(X) \/ goloso(X)	(4a)
scatoletta(f(X)) \/ goloso(X)	(4b)
¬mangia(X,f(X)) \/ goloso(X)	(4c)

Inoltre, da 5:

¬ goloso(attila)	(5)

Inoltre, il Goal, da negazione della query:
¬(∃X scatoletta(X) /\ ¬ mangia(trilli,X))
∀X (¬scatoletta(X) \/ mangia(trilli,X))
dà la clausola:

¬scatoletta(X) \/ mangia(trilli,X)	(Goal)

Da Goal e (2) si deriva:

¬scatoletta(X) \/ mangia(attila,X)	(6)

Da (6) e (4c):

¬scatoletta(f(attila)) \/ goloso(attila) (7)

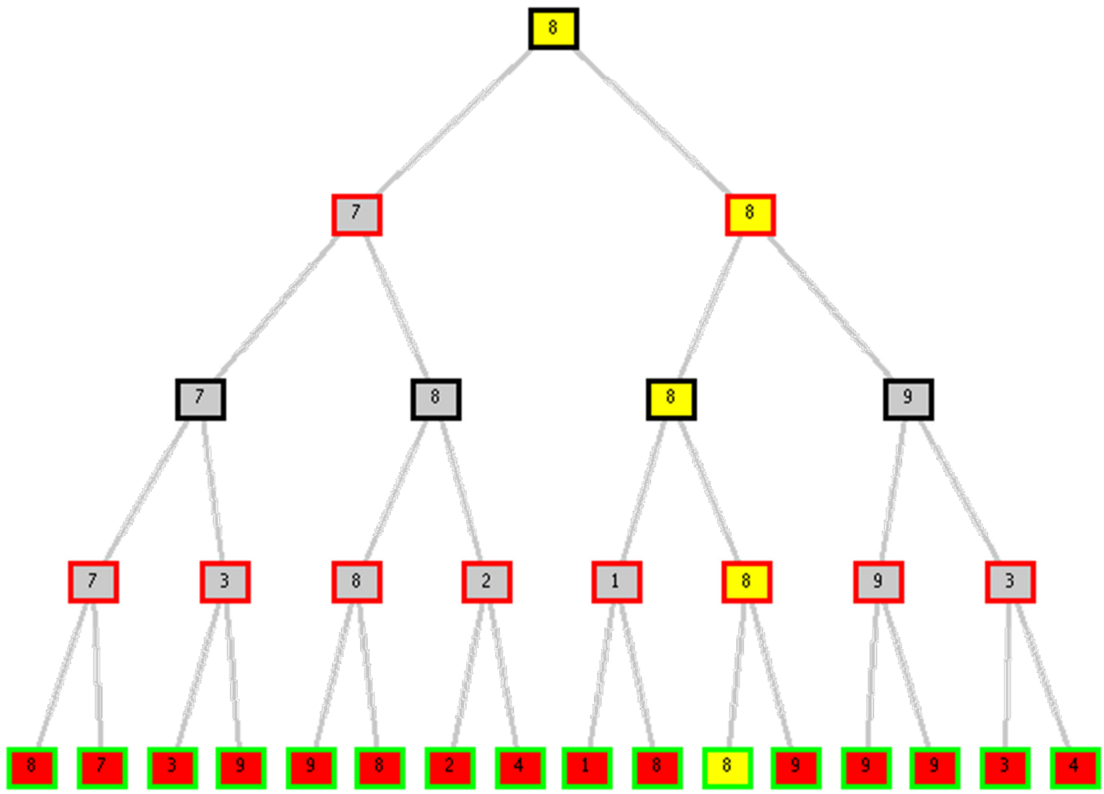
Da (7) e (4b):

goloso(attila) (8)

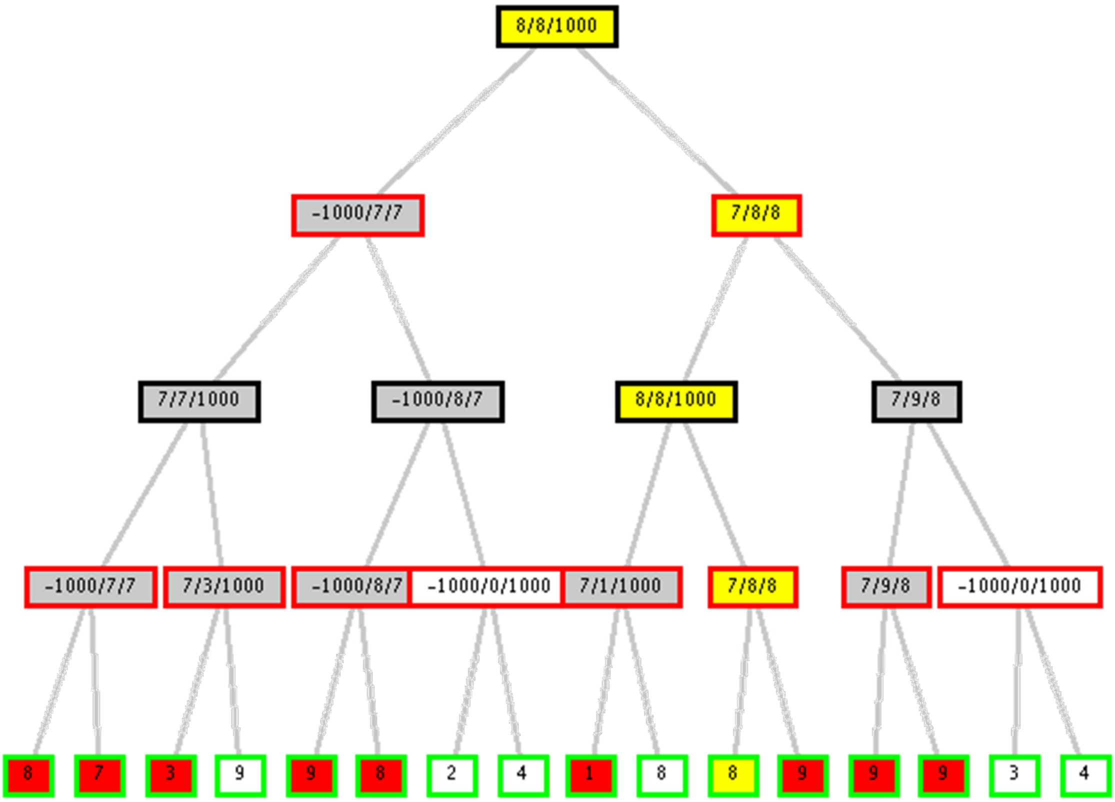
Da (8) e (5) si deriva la contraddizione (claudola vuota).

Esercizio 2

Min-Max:



Alfa-Beta:



I nodi in bianco sono quelli tagliati.

Esercizio 3

```
semisomma([],0).
semisomma([N],N).
semisomma([N1, _ | T], N) :semisomma(T,Nt), N is N1+Nt.
```

Esercizio 4

```
    solve([]).
    solve([abort|_]):-
        !,
        fail.
    solve([A|T]):-
        rule(A,B),
        append(B,T,NT),
        solve(NT).
```

Oppure, partendo dal metainterprete vanilla:

```
    solve([]).
    solve([abort|_]):-
        !,
        fail.
    solve([A|B]):-
        !,
        solve(A), solve(B).
    solve(A):-
        rule(A,B), solve(B).
```

Esercizio 5

Considero la seguente coda di archi: <$A(X_1, X_2)$, $A(X_1, X_4)$, $A(X_2, X_3)$, $A(X_3, X_4)$>. Applico l'algoritmo di consistenza d'arco:

$A(X_1, X_2)$:
- $X_1 \rightarrow X_2$: viene rimosso il valore {3} dal dominio di X_1,
- $X_2 \rightarrow X_1$: viene rimosso il valore {3} dal dominio di X_2,
- viene rimesso in coda l'arco $A(X_1, X_2)$.

$A(X_1, X_4)$:
- $X_1 \rightarrow X_4$: nessun effetto,
- $X_4 \rightarrow X_1$: nessun effetto.

$A(X_2, X_3)$:
- $X_2 \rightarrow X_3$: nessun effetto,
- $X_3 \rightarrow X_2$: viene rimosso il valore {3} dal dominio di X_3,
- viene rimesso in coda l'arco $A(X_2, X_3)$.

$A(X_3, X_4)$:
- $X_3 \rightarrow X_4$: nessun effetto,
- $X_4 \rightarrow X_3$: viene rimosso il valore {1} dal dominio di X_4,
- vengono rimessi in coda gli archi $A(X_1, X_4)$ e $A(X_3, X_4)$.

$A(X_1, X_2)$:
- $X_1 \rightarrow X_2$: nessun effetto,
- $X_2 \rightarrow X_1$: nessun effetto.

$A(X_1, X_4)$:
- $X_1 \rightarrow X_4$: nessun effetto,
- $X_4 \rightarrow X_1$: nessun effetto.

$A(X_2, X_3)$:
- $X_2 \rightarrow X_3$: nessun effetto,
- $X_3 \rightarrow X_2$: nessun effetto.

$A(X_3, X_4)$:
- $X_3 \rightarrow X_4$: nessun effetto,
- $X_4 \rightarrow X_3$: nessun effetto.

I domini risultanti sono: $D(X_1)=D(X_2)=D(X_3)=\{1,2\}$, $D(X_4)=\{2,3\}$.

Esercizio 6

Vedi slide del corso.

Esercizio 7

Vedi slide del corso.