

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

14 Giugno 2013 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (7 punti)

Si considerino le seguenti frasi:

1. *Tutti gli insegnanti sono persone*
2. *Giovanni è il preside*
3. *I presidi sono insegnanti*
4. *Tutti gli insegnanti considerano il preside un amico o non lo conoscono*
5. *Ognuno è amico di qualcuno*
6. *La gente (le persone) critica soltanto le persone che non sono loro amiche*
7. *Marco è un insegnante*
8. *Marco ha criticato Giovanni*

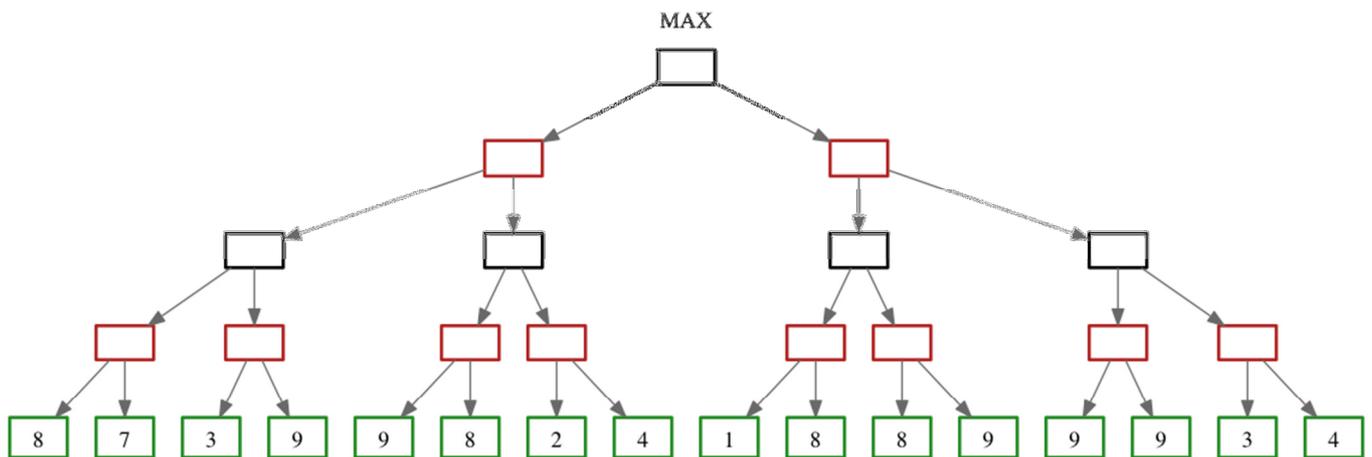
Si formalizzino in *logica dei predicati del I ordine*, utilizzando i seguenti predicati:

- $criticise(X,Y)$ X critica Y
- $knows(X,Y)$ X conosce Y
- $isFriendOf(X,Y)$ X è amico di Y
- $isPerson(X)$ X è una persona
- $isPrincipal(X)$ X è il preside
- $isProfessor(X)$ X è un professore

Infine si trasformino in *clausole* e si dimostri tramite *risoluzione* che *Giovanni non è amico di Marco*.

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 il numero *N* di vocali contenute nella lista *P* passata come parametro.

```
contavocali([], 0).
contavocali([H|T], N) :- vocale(H), contavocali(T, NT), N is NT + 1, !.
contavocali([H|T], N) :- contavocali(T, N).
vocale(a). vocale(e). vocale(i). vocale(o). vocale(u).
```

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

```
?- contavocali([o,k], X).
```

Esercizio 4 (5 punti)

Si scriva un predicato $\text{remove_at}(X, L, K, LR)$, che, data una lista L e un intero positivo K , rimuove il K -esimo elemento di L restituendo la lista residua LR e il K -esimo elemento rimosso X . Il primo elemento della lista ha $K=1$.

Esempio:

?-remove(X , [a,b,c,d,e], 3, L).

Yes $X=c$ $L=[a,b,d,e]$

Esercizio 5 (6 punti)

Si consideri il problema di ordinare in ordine crescente gli elementi di un vettore di interi $V = (4, 3, 2, 1)$. Ad ogni passo si può eseguire un'operazione $\text{swap}(x, y)$, mediante la quale è possibile scambiare di posto gli interi in posizione x ed y . Si risolva il problema utilizzando l'*algoritmo A^* con eliminazione degli stati ripetuti* e avendo cura di *indicare l'ordine di espansione dei nodi*, tenendo presente che:

- ogni operazione di $\text{swap}(x, y)$ ha costo unitario;
- ad ogni passo, gli stati futuri si ottengono applicando ogni possibile operazioni di $\text{swap}(x, y)$ nell'ordine: $\text{swap}(1, 2)$, $\text{swap}(1, 3)$, $\text{swap}(1, 4)$, $\text{swap}(2, 3)$, $\text{swap}(2, 4)$, $\text{swap}(3, 4)$;
- l'euristica di ogni stato è data da $\lceil n/2 \rceil$, dove n è il numero di elementi fuori posto;
- a parità di costo totale si preferisca il nodo a profondità minore ancora da esplorare;
- a parità di profondità si preferisca il nodo da esplorare che è stato generato prima.

Esercizio 6 (3 punti)

Si introduca il concetto di *arc-consistenza*, si mostri un esempio di *rete arc-consistente* e si delinei in pseudocodice l'*algoritmo AC3*.

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 – 2° parte (3 CFU)

14 Giugno 2013 – Tempo a disposizione: 45 min – Risultato: 32/32 punti

Esercizio 7 (10 punti)

Si introducano le *logiche descrittive* sottolineandone le peculiarità rispetto alla logica classica.

Esercizio 8 (10 punti)

Definire un predicato Prolog `count_unifying_terms(T, L, Res)` che, dati una lista L non necessariamente ground ed un termine T non necessariamente ground, restituisca il numero di elementi della lista L che unificano con T.

Esercizio 9 (12 punti)

Si mostri con un esempio la *formalizzazione di Kowalski* di un problema di Planning.

Quali sono vantaggi e svantaggi di tale formalizzazione?

Cosa si intende per *frame problem* e *frame axioms*.

VOTO:

- Esame da 3 CFU, il voto è determinato da questa 2° parte
- Esame da 9 CFU, è la media pesata della 1° parte (che vale 2/3) e della 2° (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 – 1° parte

14 Giugno 2013 – Soluzioni

Esercizio 1

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

1. *Tutti gli insegnanti sono persone*

$\forall X: \text{isProfessor}(X) \Rightarrow \text{isPerson}(X)$
{ $\neg \text{isProfessor}(X) \vee \text{isPerson}(X)$ }

2. *Giovanni è il preside*

$\text{isPrincipal}(\text{gio})$
{ $\text{isPrincipal}(\text{gio})$ }

3. *I presidi sono insegnanti*

$\forall X: \text{isPrincipal}(X) \Rightarrow \text{isProfessor}(X)$
{ $\neg \text{isPrincipal}(X) \vee \text{isProfessor}(X)$ }

4. *Tutti gli insegnanti considerano il preside un amico o non lo conoscono*

$\forall X, Y: \text{isProfessor}(X) \wedge \text{isPrincipal}(Y) \Rightarrow \text{isFriendOf}(X, Y) \vee \neg \text{knows}(X, Y)$
{ $\neg \text{isProfessor}(X) \vee \neg \text{isPrincipal}(Y) \vee \text{isFriendOf}(X, Y) \vee \neg \text{knows}(X, Y)$ }

5. *Ognuno è amico di qualcuno*

$\forall X, \exists Y: \text{isFriendOf}(X, Y)$
{ $\text{isFriendOf}(X, g(X))$ } ($g(X)$ è la funzione di Skolem)

6. *La gente (le persone) critica soltanto le persone che non sono loro amiche*

$\forall X, Y: \text{isPerson}(X) \wedge \text{isPerson}(Y) \wedge \text{criticise}(X, Y) \Rightarrow \neg \text{isFriendOf}(Y, X)$
{ $\neg \text{isPerson}(X) \vee \neg \text{isPerson}(Y) \vee \neg \text{criticise}(X, Y) \vee \neg \text{isFriendOf}(Y, X)$ }

7. *Marco è un insegnante*

$\text{isProfessor}(\text{mar})$
{ $\text{isProfessor}(\text{mar})$ }

8. *Marco ha criticato Giovanni*

$\text{criticise}(\text{mar}, \text{gio})$
{ $\text{criticise}(\text{mar}, \text{gio})$ }

9. *Goal: Giovanni non è amico di Marco*

$\neg \text{isFriendOf}(\text{gio}, \text{mar})$
{ $\text{isFriendOf}(\text{gio}, \text{mar})$ }

Teoria a clausole:

1. $\neg \text{isProfessor}(X) \vee \text{isPerson}(X)$

2. $\text{isPrincipal}(\text{gio})$

3. $\neg \text{isPrincipal}(X) \vee \text{isProfessor}(X)$

4. $\neg \text{isProfessor}(X) \vee \neg \text{isPrincipal}(Y) \vee \text{isFriendOf}(X, Y) \vee \neg \text{knows}(X, Y)$

5. $\text{isFriendOf}(X, g(X))$

6. $\neg \text{isPerson}(X) \vee \neg \text{isPerson}(Y) \vee \neg \text{criticise}(X, Y) \vee \neg \text{isFriendOf}(Y, X)$

7. $\text{isProfessor}(\text{mar})$

8. $\text{criticise}(\text{mar}, \text{gio})$

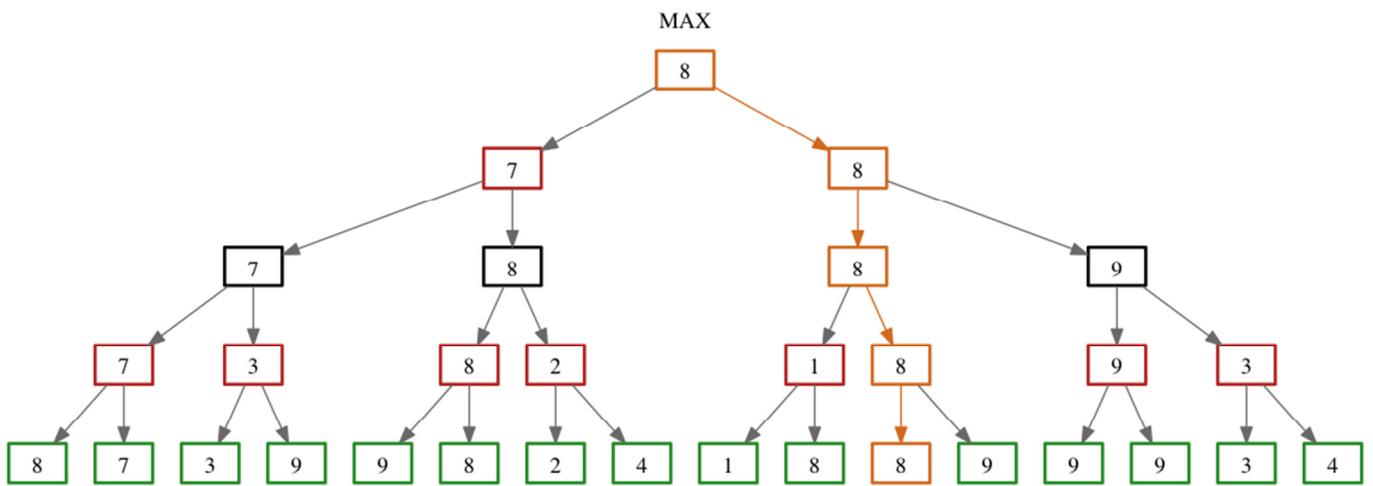
9. $\text{isFriendOf}(\text{gio}, \text{mar})$

Risoluzione:

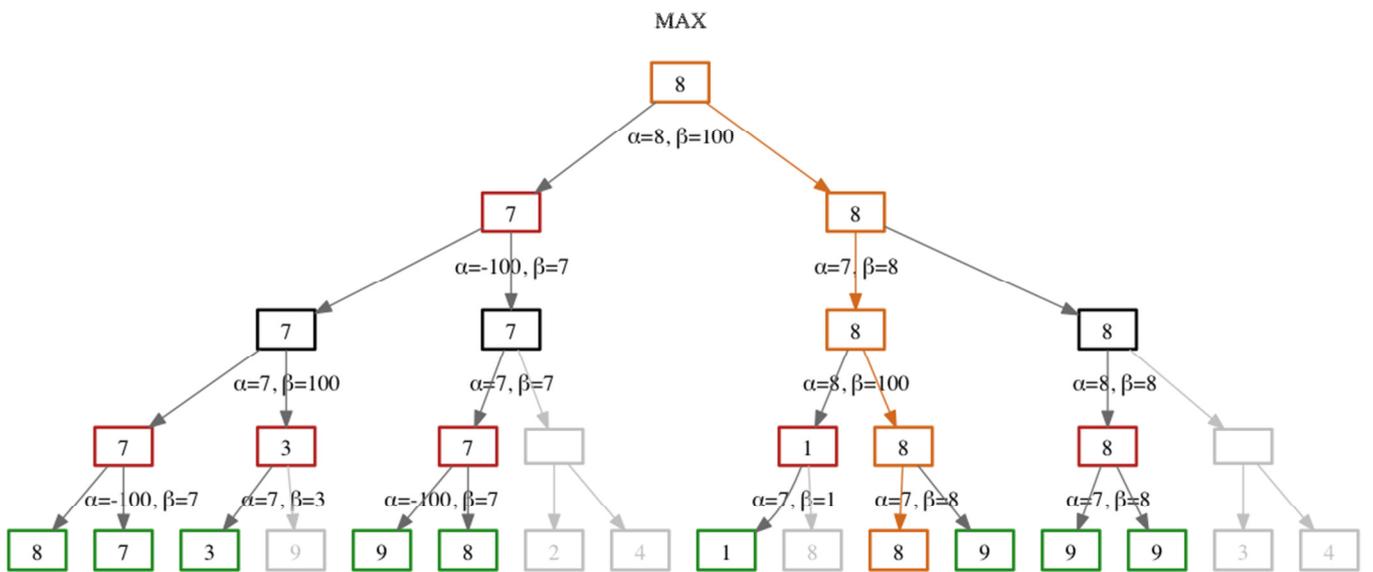
- 10. = 9. \cup 6. $\neg \text{isPerson}(\text{mar}) \vee \neg \text{isPerson}(\text{gio}) \vee \neg \text{criticise}(\text{mar}, \text{gio})$ {X/mar, Y/gio}
- 11. = 10. \cup 8. $\neg \text{isPerson}(\text{mar}) \vee \neg \text{isPerson}(\text{gio})$ {}
- 12. = 11. \cup 1. $\neg \text{isProfessor}(\text{mar}) \vee \neg \text{isPerson}(\text{gio})$ {X/mar}
- 13. = 12. \cup 7. $\neg \text{isPerson}(\text{gio})$ {}
- 14. = 13. \cup 1. $\neg \text{isProfessor}(\text{gio})$ {X/gio}
- 15. = 14. \cup 3. $\neg \text{isPrincipal}(\text{gio})$ {X/gio}
- 16. = 15. \cup 2. \square

Esercizio 2

min-max:



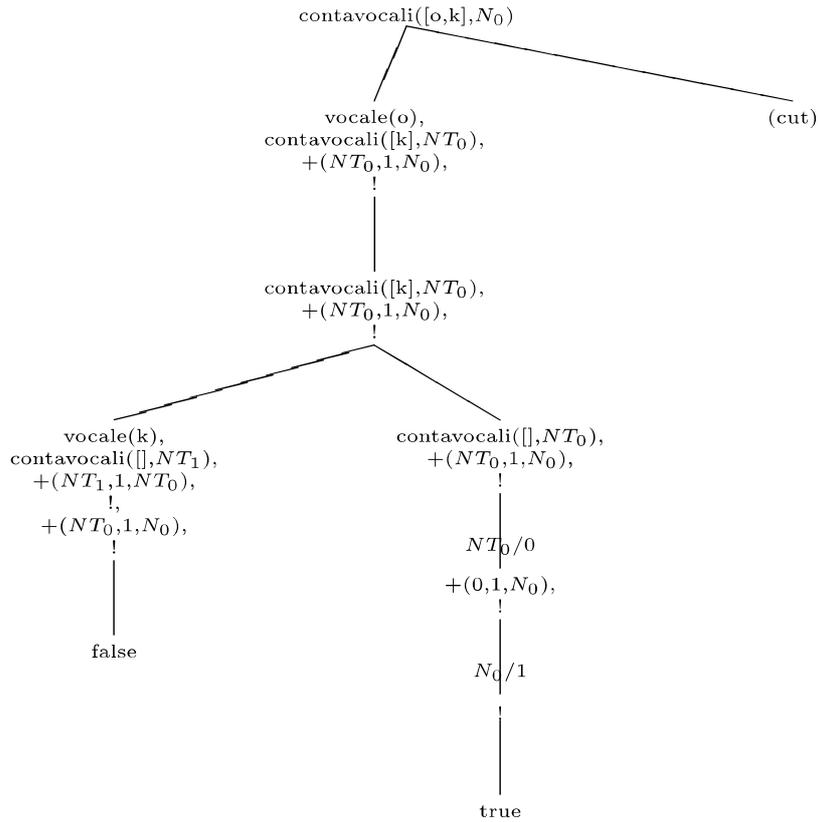
alfa-beta:



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

Esercizio 3

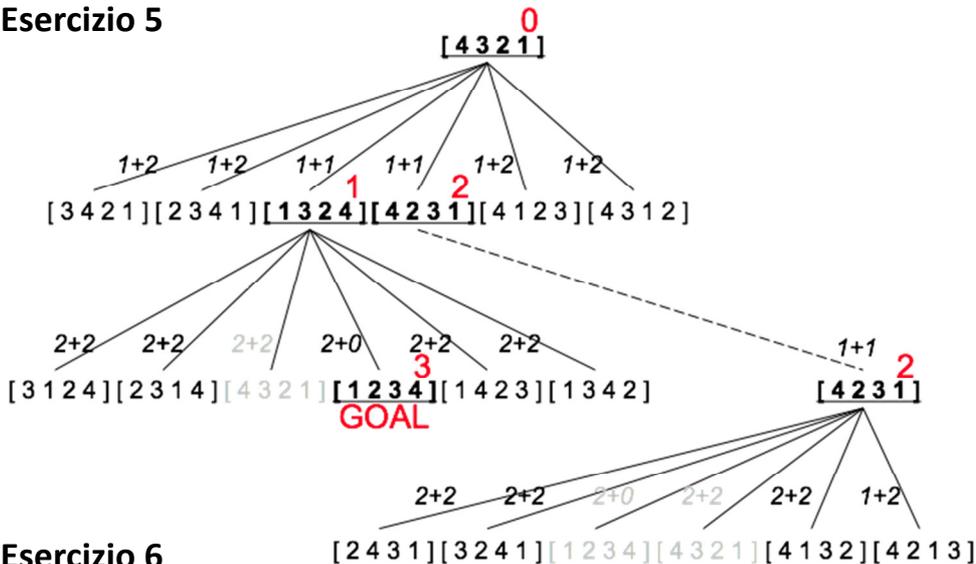
Albero SLD:



Esercizio 4

```
% Remove the K'th element from a list.
% The first element in the list is number 1.
% remove_at(X,L,K,R) :- X is the K'th element of the list L; R is the
% list that remains when the K'th element is removed from L.
remove_at(X, [X|Xs], 1, Xs).
remove_at(X, [_|Xs], K, [Y|Ys]) :- K > 1, K1 is K - 1, remove_at(X,Xs,K1,Ys).
```

Esercizio 5



Esercizio 6

Vedi slides del corso.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte

14 Giugno 2013 – Soluzioni

Esercizio 7

Vedi slides del corso.

Esercizio 8

Una prima soluzione:

```
count_unifying_terms1(Term, List, NUMBER) :-
    aux_count_unifying_terms1(Term, List, NUMBER,0).
aux_count_unifying_terms1(Term, [], NUMBER, NUMBER).
aux_count_unifying_terms1(Term, [Term|Tail], NUMBER, AUX_NUMBER) :-
    !, NEW_AUX_NUMBER is AUX_NUMBER + 1,
    aux_count_unifying_terms1(Term, Tail, NUMBER, NEW_AUX_NUMBER).
aux_count_unifying_terms1(Term, [Head|Tail], NUMBER, AUX_NUMBER) :-
    aux_count_unifying_terms1(Term, Tail, NUMBER, AUX_NUMBER).
```

Tale soluzione non è corretta in alcuni casi, poiché nel caso in cui Term contenga variabili e i termini nella lista contengano costanti o altri termini ground, nell'unificazione la sostituzione delle variabili verrebbe propagata lungo la chiamata ricorsiva. Ad esempio, la soluzione di cui sopra, se invocata con:

```
:- count_unifying_terms1(p(1), [p(X), p(Y)], NUMBER).
```

Restituisce true, X/1, Y/1, NUMBER/2

Se invece invocata con il goal:

```
:- count_unifying_terms1(p(X), [p(1), p(2)], NUMBER).
```

Il predicato restituisce true con X/1, NUMBER/1 (invece il risultato atteso è NUMBER/2)

Soluzioni possibili in tal senso possono essere diverse:

```
%%% CONTROLLA LE NON UNIFICAZIONI, IMPONENDO UN FAIL QUANDO UNIFICANO
```

```
count_unifying_terms(Term, List, NUMBER) :-
    aux_count_unifying_terms(Term, List, NUMBER,0).
aux_count_unifying_terms(Term, [], NUMBER, NUMBER).
aux_count_unifying_terms(Term, [Head|Tail], NUMBER, AUX_NUMBER):-
    not_unifies(Term, Head),
    !,
    aux_count_unifying_terms(Term, Tail, NUMBER, AUX_NUMBER).
aux_count_unifying_terms(Term, [Head|Tail], NUMBER, AUX_NUMBER):-
    NEW_AUX_NUMBER is AUX_NUMBER + 1,
    aux_count_unifying_terms(Term, Tail, NUMBER, NEW_AUX_NUMBER).
```

```
not_unifies(Term, Term) :- !, fail.
```

```
not_unifies( _ , _ ).
```

Oppure:

```
%%% ASSERTISCE TERM COME "TEMPLATE" DI CLAUSOLA
count_unifying_terms1(Term, List, NUMBER) :-
    assert(term(Term)),
    aux_count_unifying_terms1(List, NUMBER,0).

aux_count_unifying_terms1([], NUMBER, NUMBER).
aux_count_unifying_terms1([X|Tail], NUMBER, AUX_NUMBER):-
    term(X),
    !,
    NEW_AUX_NUMBER is AUX_NUMBER + 1,
    aux_count_unifying_terms1(Tail, NUMBER, NEW_AUX_NUMBER).
aux_count_unifying_terms1([_ | Tail], NUMBER, AUX_NUMBER):-
    aux_count_unifying_terms1(Tail, NUMBER, AUX_NUMBER).
```

Oppure:

```
%%% Uso di findall direttamente sulla lista
count_unifying_terms1(Term, List, NUMBER) :-
    findall(Term, member(Term, List), R),
    length(R, NUMBER).
%%% member definita poi come usuale
```

Oppure, altre due soluzioni possibili: (a) effettuando una copia di Term e usare tale copia per testare l'unificazione; (b) ricostruendo l'algoritmo di unificazione e testando i vari casi.

Esercizio 9

Vedi slides del corso.