FONDAMENTI DI INTELLIGENZA ARTIFICIALE
8 Settembre 2016 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

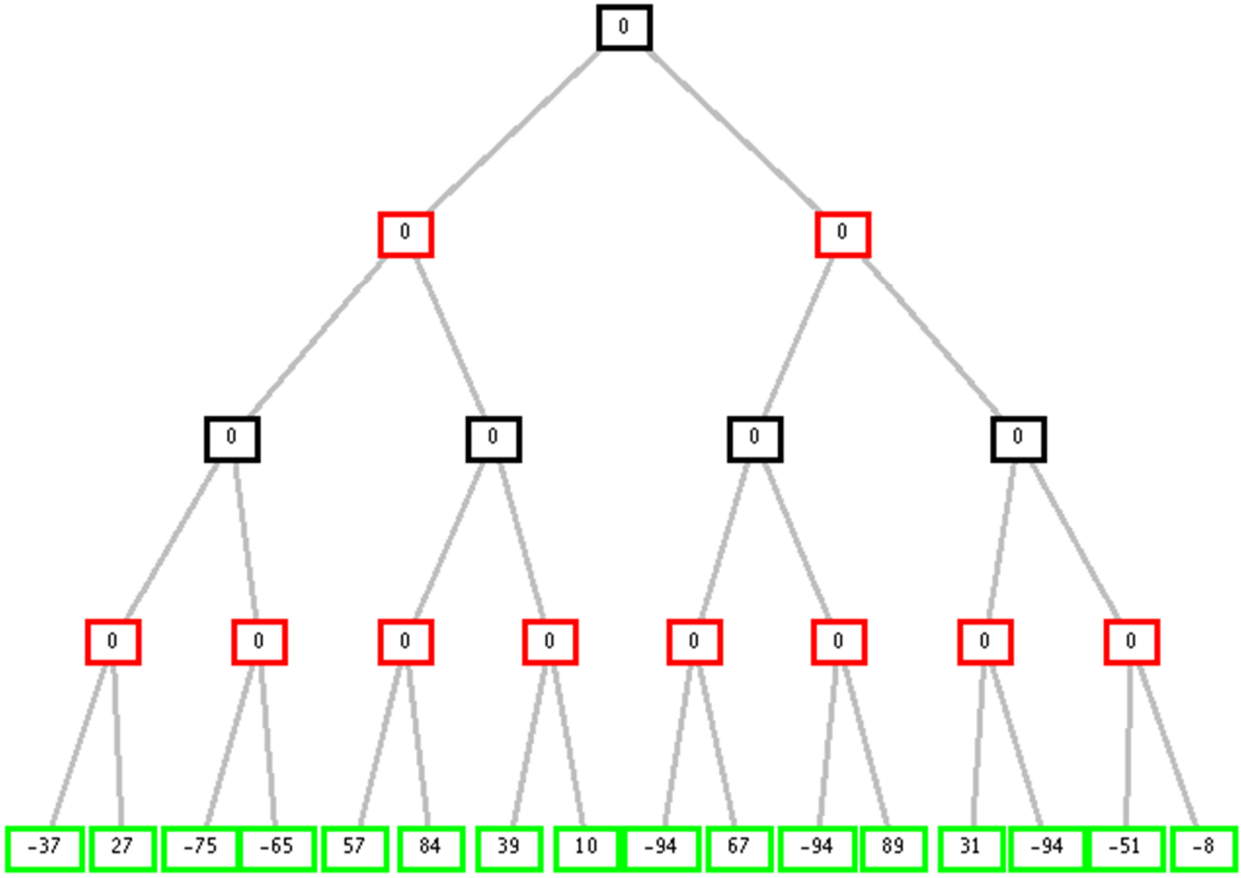
Si formalizzino in logica dei predicati del I ordine le seguenti frasi:

1. Se un membro di una squadra ha violato il doping, tale squadra viene sospesa.
2. Chiunque viola il doping se è risultato positivo al test per EPO, o se è risultato positivo al test per testosterone sintetico. Un membro viola il doping anche qualora non si presenti al test.
3. Un test è considerato positivo ad una sostanza se sia il primo, sia il secondo esame per quella sostanza danno risultato positivo.
4. canvelo (nome di una squadra) ha come membro jim.
5. lightriders (nome di una squadra) ha come membro kim.
6. jim è risultato positivo al primo test per l'EPO.
7. kim non si è presentato al test.

Si utilizzi la risoluzione per dimostrare che esiste almeno una squadra che viene sospesa. Si usino i seguenti predicati dove "…/x" esprime il numero x di parametri richiesti:
`hasMember/2`, **`hasViolation/1`**, **`suspendTeam/1`**, **`testPositive/2`**, **`firstPositive/2`**, **`secondPositive/2`**, **`missedTest/1`**.

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 l'algoritmo *min-max* e l'algoritmo *alfa-beta* risolvono il problema e la mossa selezionata dal giocatore.



Esercizio 3 (6 punti)

Il seguente programma Prolog:

```
mymember(X,[X|_]) :- atomic(X), !.
mymember(X,[Y|_]) :- is_list(Y), !, mymember(X,Y).
mymember(X,[_|L]) :- mymember(X,L).
```

controlla se il primo argomento è un atomo e compare nel secondo (lista di atomi, o lista di liste, a loro volta anche innestate). Ad esempio, tutte le chiamate seguenti hanno successo:

```
?-mymember(a,[b,a]).
?-mymember(a,[1,2,[[a,1,b],3]]).
```

mentre la chiamata:

```
?-mymember([b,a],[[b,a]]).
```

fallisce. Si dica qual è il risultato dell'esecuzione del seguente goal e se ne disegni il relativo albero di derivazione SLD:

```
?-mymember(a,[[1,f,a],b,a]).
```

A tal scopo, si considerino i predicati `is_list/1` e `atomic/1` come pre-definiti, con significato ovvio.

Esercizio 4 (4 punti)

Si scriva un predicato Prolog `remove-list(L1,L2,L3)` che data una lista L1 e una lista L2 restituisca una lista L3 contenente gli elementi di L1 non contenuti in L2. Esempio:

```
?- remove_list([a,b,a,b,b,c],[a,c],L).
     L=[b,b,b].
?- remove_list([a,b], [], L).
     L=[a,b].
```

Si definiscano tutti i predicati utilizzati.

Esercizio 5 (5 punti)

Si risolva il seguente problema di soddisfacimento di vincoli. Si hanno 6 cani e 4 box. Ogni box può contenere al più due cani. I cani hanno le seguenti caratteristiche:

cane1: adulto marrone
cane2: adulto bianco
cane3: cucciolo nero
cane4: adulto nero
cane5: cucciolo bianco
cane6: adulto bianco

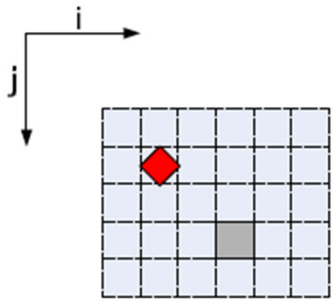
I vincoli sono i seguenti:

- i cani neri adulti devono stare nel box vicino al cane marrone;
- un cane nero adulto non può stare né nello stesso box né in box vicini ad un cane bianco adulto, e viceversa;
- i cuccioli possono solo stare in box contenenti solo altri cuccioli;
- i cuccioli di un certo colore devono stare in box vicini ad almeno un box contenente un adulto dello stesso colore.

Si modelli il problema come un CSP, in cui le variabili rappresentano i cani. Il dominio delle variabili contiene i numeri corrispondenti ai box. Numeri vicini corrispondono a box vicini. Non è necessario rappresentare formalmente anche i vincoli, che però andranno utilizzati nella soluzione.
Si risolva il problema adottando la strategia del Forward Checking. Per la selezione delle variabili si adotti la strategia first fail. In caso di uguale dimensione dei domini si utilizzino le variabili (che rappresentano i cani) nell'ordine definito dal problema (partendo dal numero più basso).

Esercizio 6 (5 punti)

Si consideri il seguente problema, in cui l'obiettivo è spostare una scatola (rombo rosso in figura) in una stanza fino a raggiungere una casella obiettivo (indicata dal quadrato grigio in figura). La stanza è considerata di dimensioni infinite e priva di ostacoli. Si noti quindi che possono essere assunte posizioni con coordinate anche negative.



Dalla posizione di riga i e colonna j, indicata come box(i,j), la scatola può essere spostata:

1. a destra (R, per Right) portandola in box(i+1,j)
2. a sinistra (L, per Left), portandola in box(i-1,j)
3. in alto (U, per Up) portandola in box(i,j-1)
4. in basso (D, per Down) portandola in box(i,j+1).

Sapendo che nello stato iniziale la scatola è nel **box(0,0)** (indicato in figura), e l'obiettivo è di averla nel **box(2,2)**, risolvere il problema di ricerca identificando la sequenza di mosse applicando la ricerca euristica "greedy". Si utilizzi come euristica la distanza di Manhattan della scatola dalla casella obiettivo (2,2). Si applichino le azioni nell'ordine in cui sono indicate nel testo (R,L,U,D). A parità di altro, si preferisca l'espansione del nodo generato prima.

Si riporti l'albero di ricerca, le azioni che portano ai nodi, l'ordine di espansione dei nodi, il valore della funzione euristica, e infine la sequenza di azioni soluzione. Per indicare lo stato si riporti solo la posizione della scatola all'interno della stanza.

Esercizio 7 (1 punti)

Si descrivano i predicati Prolog di assert e retract e il loro utilizzo, eventualmente con anche con l'aiuto di un esempio.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE

8 Settembre 2016 – Soluzioni

Esercizio 1

1. ∀Team, ∀Member hasMember(Team,Member) ∧ hasViolation(Member) ⟶ suspendTeam(Team).
2. ∀Person
   (testPositive(Person, epo) ∨ testPositive(Person, testosterone) ∨ missedTest(Person)) ⟶ hasViolation(Person).
3. ∀Person ∀Substance
   firstPositive(Person, Substance) ∧ secondPositive(Person, Substance) ⟶ testPositive(Person, Substance).
4. hasMember(canvelo, jim).
5. hasMember(lightriders, kim).
6. firstPositive(jim, epo).
7. missedTest(kim).

Query: ∃X suspendTeam(X).

Trasformazione in clausole:

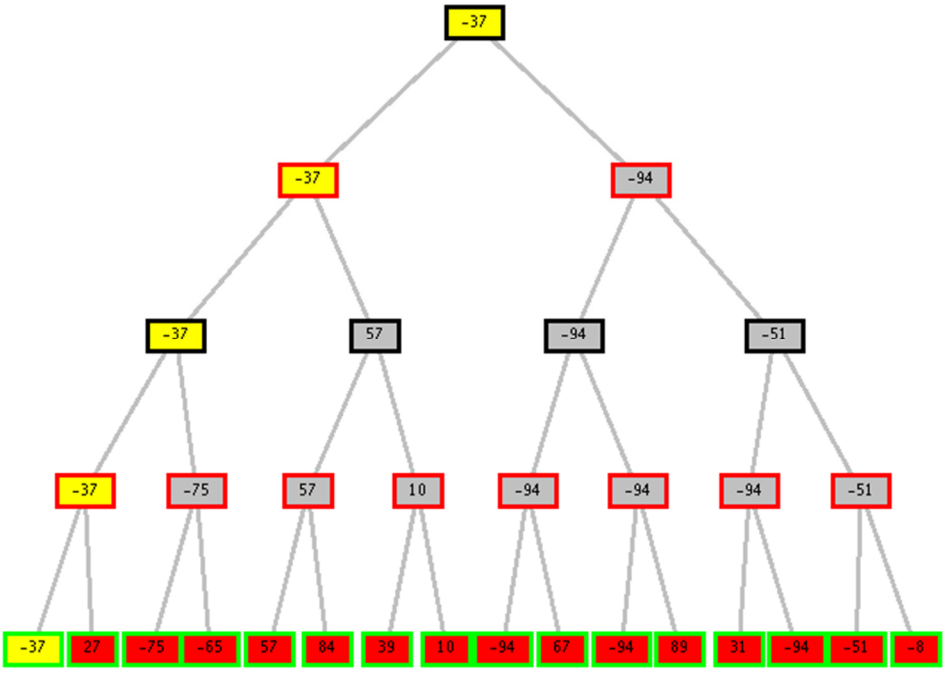
C1: ¬hasMember(Team,Member) ∨ ¬hasViolation(Member) ∨ suspendTeam(Team).
C2a: ¬testPositive(Person, epo) ∨ hasViolation(Person).
C2b: ¬testPositive(Person, testosterone) ∨ hasViolation(Person).
C2c: ¬missedTest(Person) ∨ hasViolation(Person).
C3: ¬firstPositive(Person, Substance) ∨ ¬secondPositive(Person, Substance) ∨ testPositive(Person, Substance).
C4: hasMember(canvelo, jim).
C5: hasMember(lightriders, kim).
C6: firstPositive(jim, epo).
C7: missedTest(kim).
QNeg: ¬suspendTeam(X).

Risoluzione:

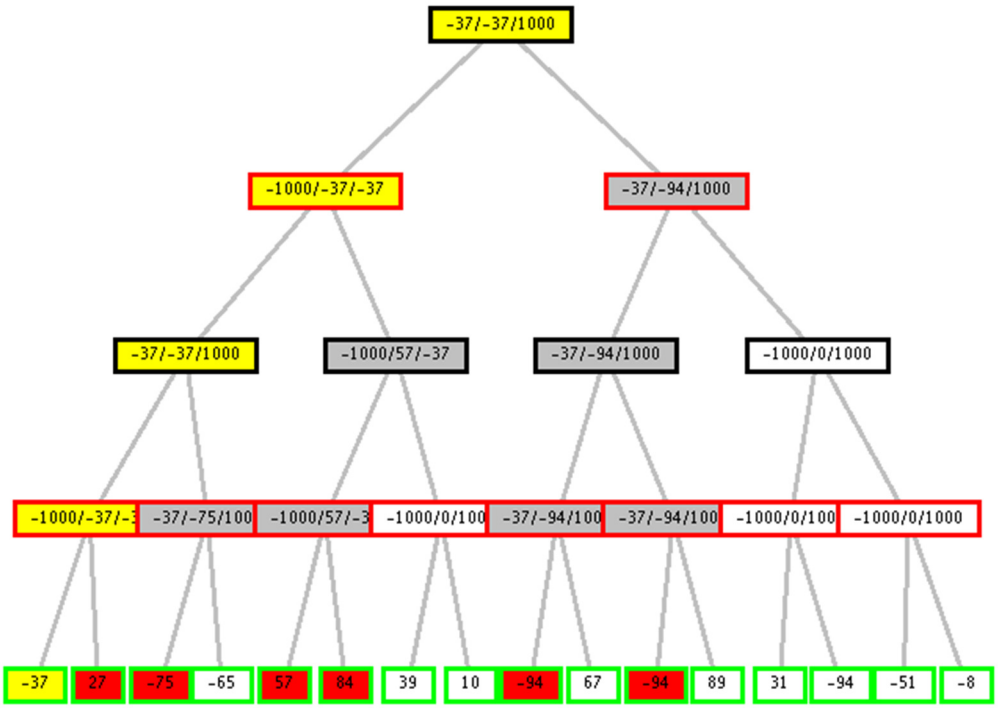
C8: QNeg+C1: ¬hasMember(Team,Member) ∨ ¬hasViolation(Member).
C9: C8+C5: ¬hasViolation(kim).
C10: C9+C2c: ¬missedTest(kim).
C11: C10+C7: contraddizione!

Esercizio 2

Min-Max:

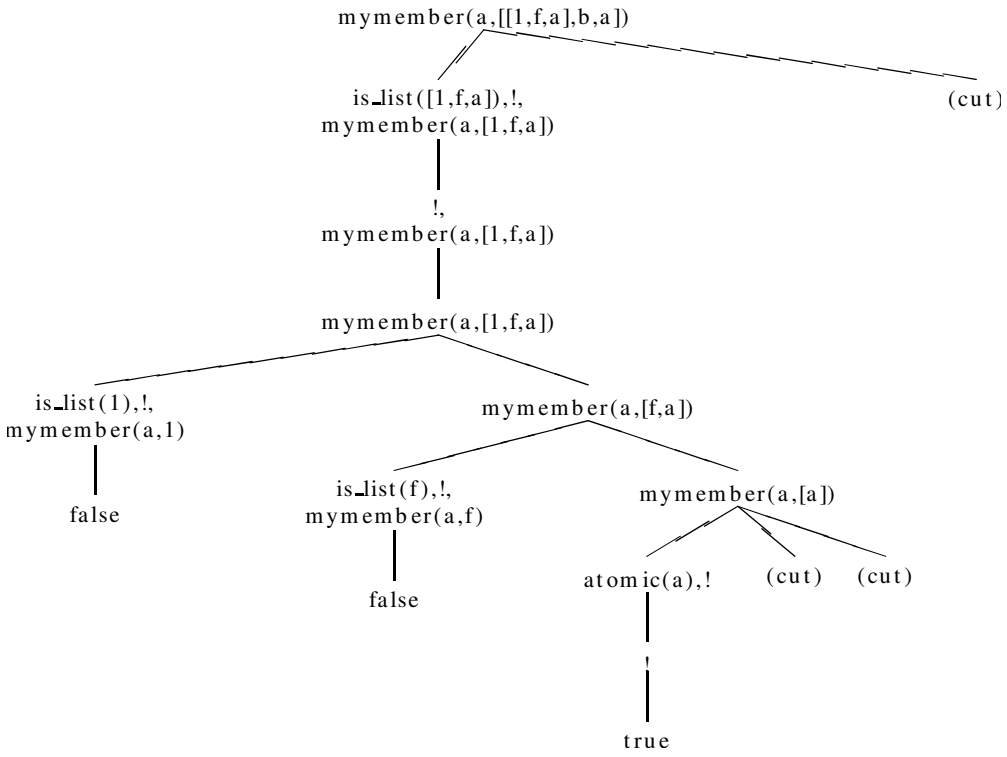


Alfa-beta:



I nodi che portano alla soluzione sono in giallo, quelli tagliati in bianco.

Esercizio 3



Esercizio 4

```
remove_list( [ ], _, [ ]) :- !.
remove_list( [H|T], L2, R) :- member(H, L2), !, remove_list(T, L2, R).
remove_list( [H|T], L2, [H|R]) :-  remove_list(T, L2, R).

member( X, [ X | _]) :- !.
member( X, [ _ | T]) :- member(X,T).
```

Esercizio 5

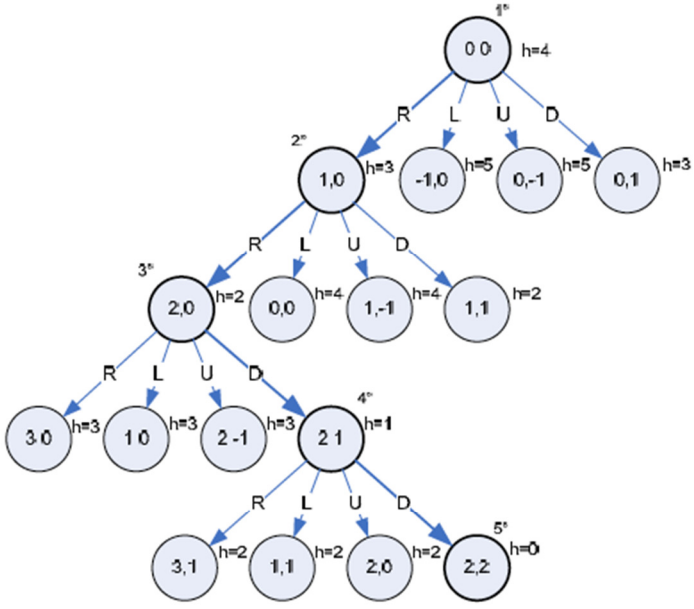
Le variabili rappresentano i cani. Il dominio delle variabili contiene i numeri corrispondenti ai box.
C1, C2, C3, C4, C5, C6::[1, 2, 3, 4]

Istanzio la prima variabile:
C1 = 1
e propago:
C2::[1,2,3,4]
C3::[2,3,4]
C4::[2]
C5::[2,3,4]
C6::[1,2,3,4]

Istanzio la quarta variabile:
C4=2
e propago
C2::[4]
C3::[3,4]
C5::[3,4]
C6::[4]

Istanzio la seconda variabile:
C2=4
e propago
C3::[3]
C5::[3]
C6::[4]
Soluzione.

Esercizio 6



Il piano generato è: R(0,0) – R(1,0) – D(2,0) – D(2,1).

Nota: per semplicità, nella figura sopra le azioni sono state abbreviate omettendo l'indicazione delle coordinate corrispondenti allo stato da cui partono. Per esempio: le quattro azioni dal nodo radice dovrebbero essere: R(0,0), L(0,0), U(0,0) e D(0,0).

Esercizio 7

Vedi slide.