FONDAMENTI DI INTELLIGENZA ARTIFICIALE

14 Gennaio 2016 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

Si rappresentino in logica dei predicati del I ordine, le seguenti affermazioni:

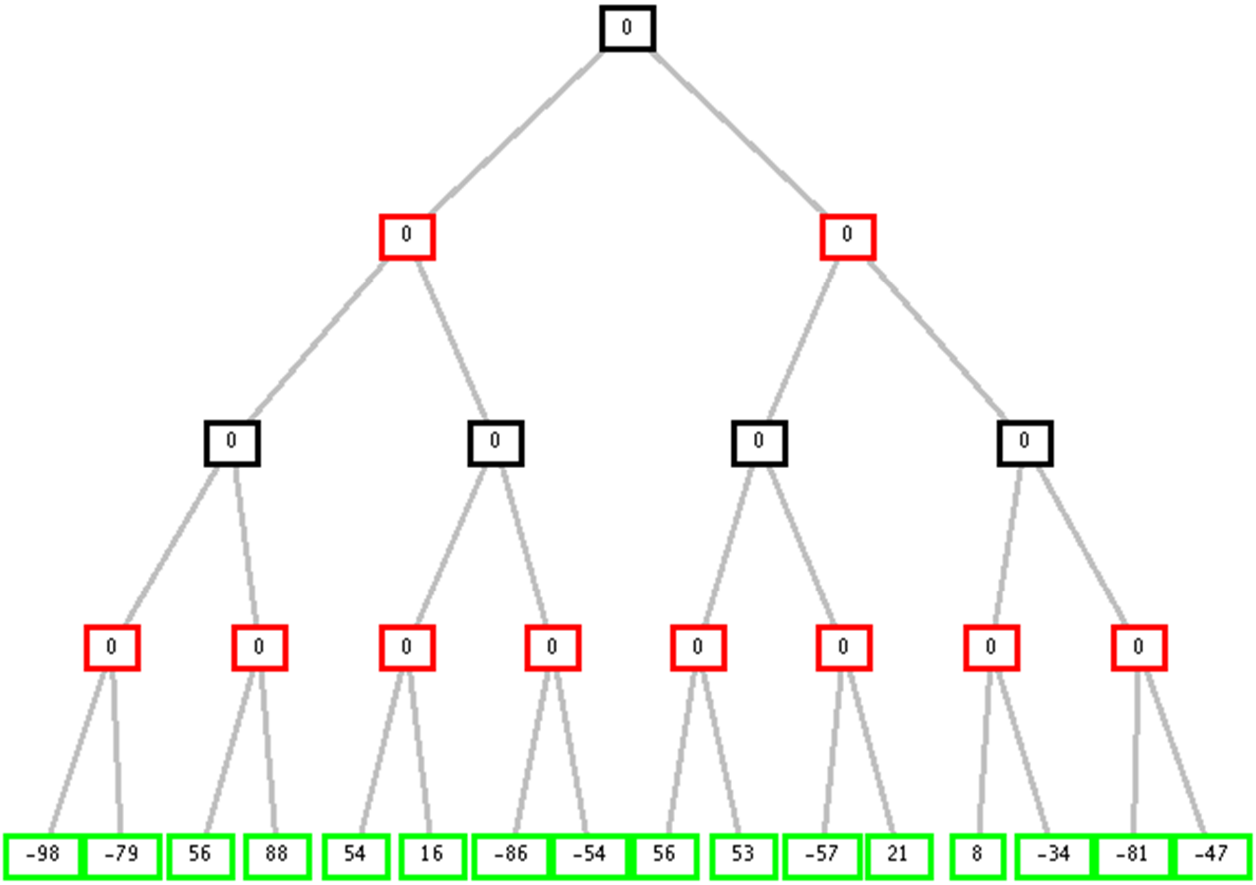
- Ad alcuni pazienti piacciono tutti i medici.
- A nessun paziente piace alcun ciarlatano.

Si applichi la risoluzione alla teoria formata dalle formule ottenute sopra per verificare che:

- Nessun medico è un ciarlatano.
-

Esercizio 2 (4 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 *alfa-beta* risolve il problema.



Esercizio 3 (5 punti)

Si considerino dei termini Prolog che rappresentano le informazioni relative ad uno studente: nome, cognome, matricola, lista esami, in cui lista esami (superati) è una lista di coppie (esame,voto).
Ad esempio la struttura/termine Prolog:
`stud(mario, rossi, 5567, [esame(mah,24), esame(geo, 18), esame(info, 30)]).`
rappresenta uno studente con infomazioni sopra elencate.
Scrivere un programma PROLOG `numstudenti` che, data una lista che contiene tante strutture studente come sopra esemplificate, calcola il numero di studenti che hanno superato un SOLO esame (quindi che hanno un SOLO elemento nella lista esami).
Ad esempio alla query:

```
?-  numstudenti ([stud(mario, rossi, 5567, [esame(math,24),
esame(geo, 18), esame(info, 30)]), stud(giovanna,verdi, 5569,
[esame(math,30)]), stud(paola, gialli, 4569, [])],N).
```

la risposta sarà
`yes N=1`
Perché solo 1 studente ha dato un SOLO esame.

Esercizio 4 (6 punti)

Dato il programma Prolog:

```
nodupl([], []).
nodupl([X|Xs], Ys):-
    member(X, Xs), !,
    nodupl(Xs, Ys).
nodupl([X|Xs], [X|Ys]):-
      nodupl(Xs, Ys).
```

Disegnare l'albero SLD per il goal (fermarsi nello sviluppo dell'albero appena si raggiunge **la prima soluzione e non si espliciti la parte relativa al predicato member**):
`:-nodupl([a,a,b],X).`

Esercizio 5 (6 punti)

Si hanno a disposizione due brocche, una da tre litri e una da cinque, e una sorgente d'acqua. Le due brocche sono inizialmente vuote, e si vuole far sì che una qualsiasi di esse contenga esattamente un litro d'acqua e l'altra sia vuota, attingendo dalla sorgente la minor quantità possibile di acqua. Per ottenere questo è possibile riempire una qualsiasi delle brocche attingendo l'acqua dalla sorgente, oppure svuotare una delle brocche gettando via l'acqua, o versare l'acqua da una brocca qualsiasi all'altra fino a che la prima sia vuota o la seconda sia piena.
Risolvere il problema con l'algoritmo di ricerca a costo uniforme (selezione sempre del nodo per espansione a costo minore), evitando gli stati ripetuti.

Consigli sulla modellazione del problema come ricerca nello spazio degli stati:

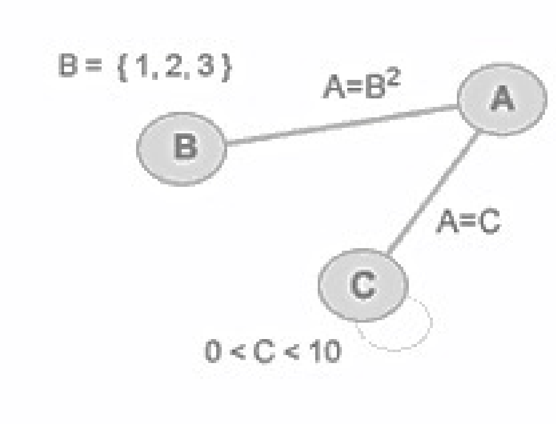
- Uno stato può essere rappresentato da una coppia di numeri (b1,b2) che corrispondono alla quantità di acqua (in litri) pesente nelle due brocche (b1 indica il contenuto della brocca più grande). Tenendo conto delle possibili azioni, lo spazio degli stati è l'insieme delle coppie di interi (b1,b2) tali che b1 ∈ {0,1,...,5} e b2 ∈ {0,1,2,3}. Si noti che non tutte le azioni sono reversibili, quindi lo spazio degli stati sarà un grafo orientato. Un nodo (stato) A è connesso a un altro nodo B (con un arco orientato da A a B), se è possibile raggiungere B a partire da A con una delle azioni permesse (per es., lo stato (5, 3) è connesso a (5, 0), poiché il secondo può essere raggiunto dal primo gettando via l'intero contenuto della seconda brocca). Lo stato iniziale è (0, 0), mentre si hanno due stati obiettivo: (1, 0) e (0, 1).

- Gli operatori possono essere definiti come una funzione successore SF(b1,b2), che restituisce tutte le coppie ((b′1, b′2), a), dove (b′1, b′2) è uno stato raggiungibile da (b1, b2) con l'azione a (a è una qualsiasi descrizione dell'azione, per es. una stringa). In alternativa si possono definire sei operatori (funzioni) fk(b1,b2), corrispondenti alle seguenti azioni:
  f1. riempire la brocca più grande con l'acqua della sorgente (k = 1);
  f2. riempire la brocca più piccola con l'acqua della sorgente (k = 2);
  f3. gettar via l'acqua dalla brocca più grande (k = 3);
  f4. gettar via l'acqua dalla brocca più piccola (k = 4);
  f5. versare nella brocca più piccola l'acqua della brocca più grande fino a svuotare la seconda o a riempire la prima (k = 5);
  f6. versare nella brocca più grande l'acqua della brocca più piccola fino a svuotare la seconda o a riempire la prima (k = 6).

  Notare che ciascun operatore restituisce un singolo stato (b′1,b′2), se l'azione corrispondente è applicabile (per es., f1 non è applicabile allo stato (5, 0), in quanto la brocca più grande è già piena). Tenendo conto del vincolo di dover attingere dalla sorgente la minor quantità possibile di acqua, il costo degli operatori f1 e f2 è pari alla quantità di acqua attinta dalla sorgente, mentre tutti gli altri operatori hanno un costo nullo.

Esercizio 6 (3 punti)

Si definisca cosa si intende per ARC-Consistenza di un constraint graph e si descriva come si ottiene l'arc-consistenza del seguente constaint-graph. Si supponga che la variabile A abbia quale dominio iniziale i numeri interi compresi fra 0 e 100, estremi inclusi.



Esercizio 7 (2 punti)

Si spieghi cosa si intende per pianificazione "classica" in AI e le assunzioni semplificative che si fanno al riguardo.

FONDAMENTI DI INTELLIGENZA ARTIFICIALE

14 Gennaio 2016 – Soluzioni

Esercizio 1

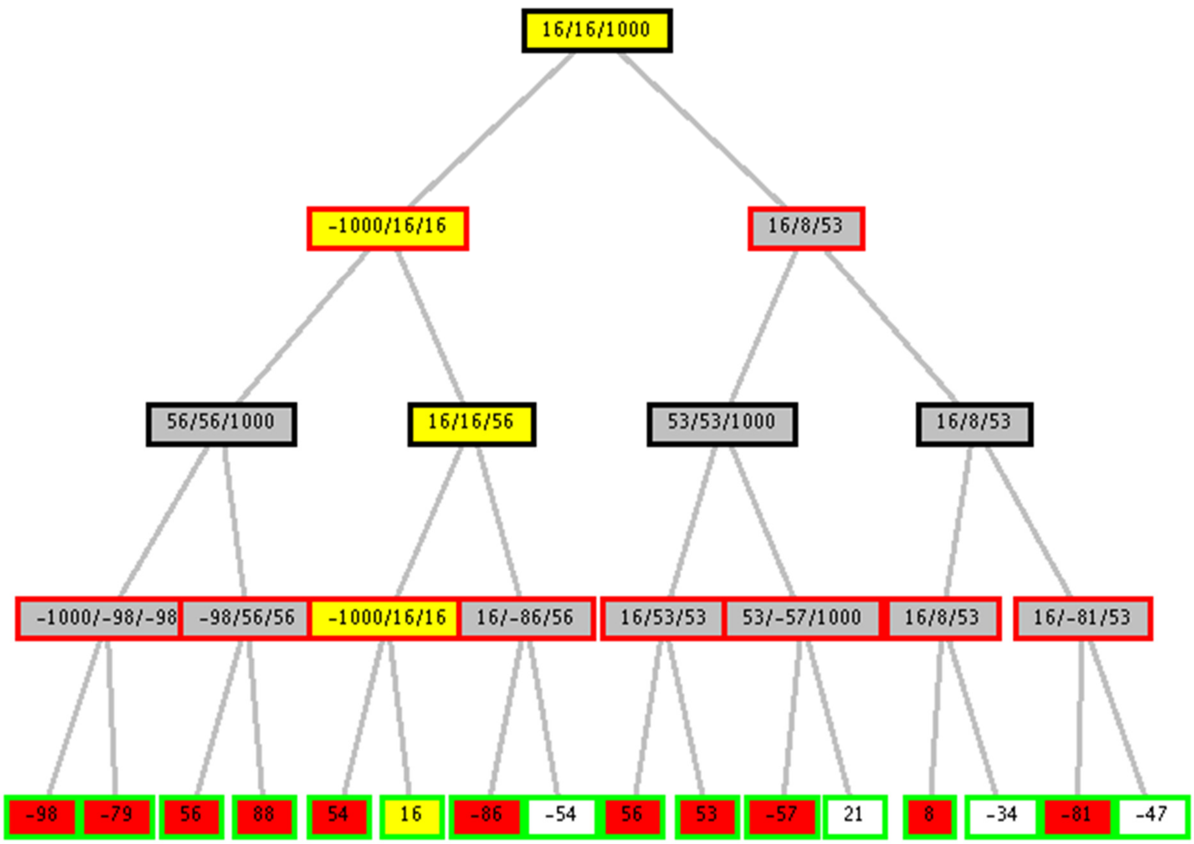
1.: $\exists X\ (\ paz(X) \wedge \forall Y\ (med(Y) \rightarrow piace(X, Y))\ )$
2.: $\neg\exists X\ (\ paz(X) \wedge \exists Y\ (ciar(Y) \wedge piace(X, Y))\ )$
Goal:
$\forall X\ (med(X) \rightarrow \neg ciar(X))$
oppure, in modo equivalente: $\neg\exists X\ (\ med(X) \wedge ciar(X)\ )$
Goal negato: $\neg\forall X\ (med(X) \rightarrow \neg ciar(X))$

Tradotti in clausole:
1.: paz(c)
1a.: $\neg med(Y) \vee piace(c,y)$
2.: $\neg paz(X) \vee \neg ciar(Y) \vee \neg piace(X, Y)$
GNeg1.: med(c1)
GNeg2.: ciar(c1)

Refutazione:

3.: GNeg1+1a. :	piace(c, c1)
4.: 3. + 2. :	$\neg paz(c) \vee \neg ciar(c1)$
5.: 4. + GNeg2.:	$\neg paz(c)$
6.: 5. + 1:	Clausola vuota

Esercizio 2



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

Esercizio 3

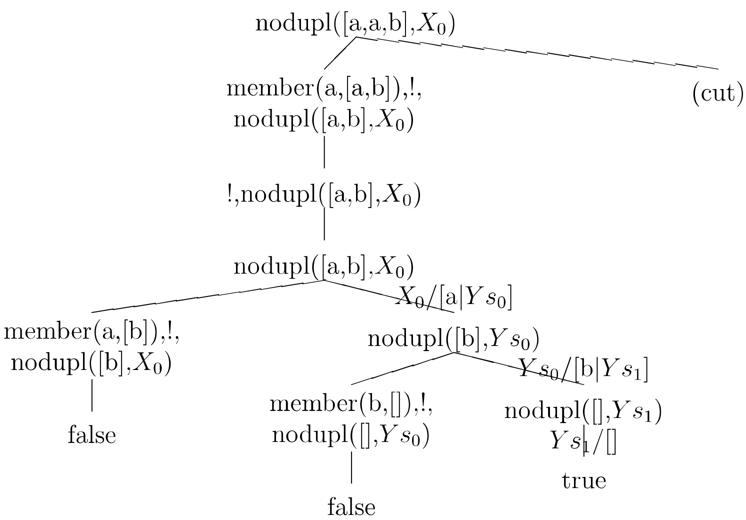
```
unesame(stud(_,_,_,[esame(_,_)])).


numstudenti([],0).
numstudenti([X|Xs],N):-unesame(X),!,
          numstudenti(Xs,P), N is P + 1.

numstudenti([_|Xs],N):- numstudenti(Xs,N).
```

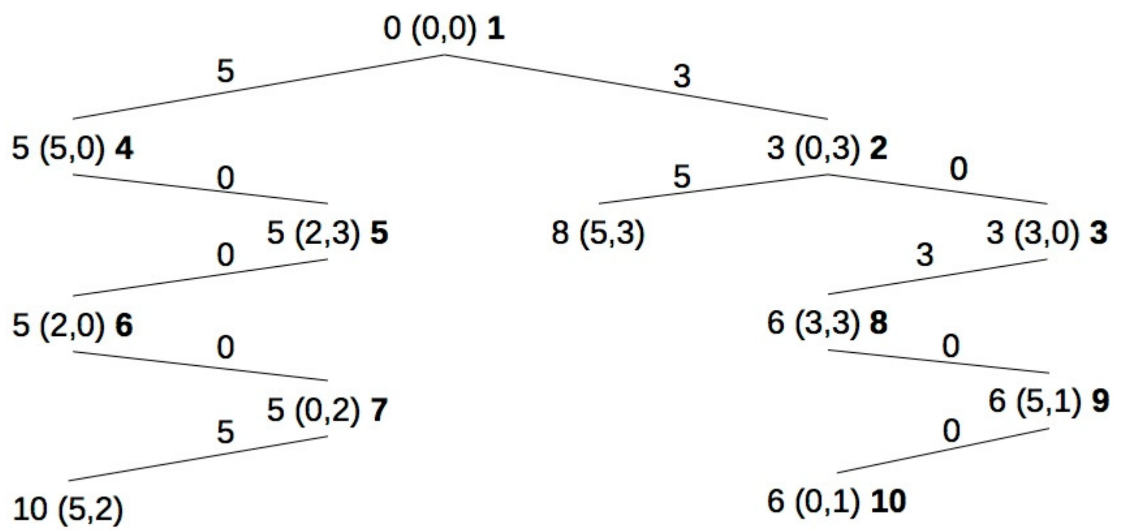
Esercizio 4

Qui si mostra l'intero albero SLD.



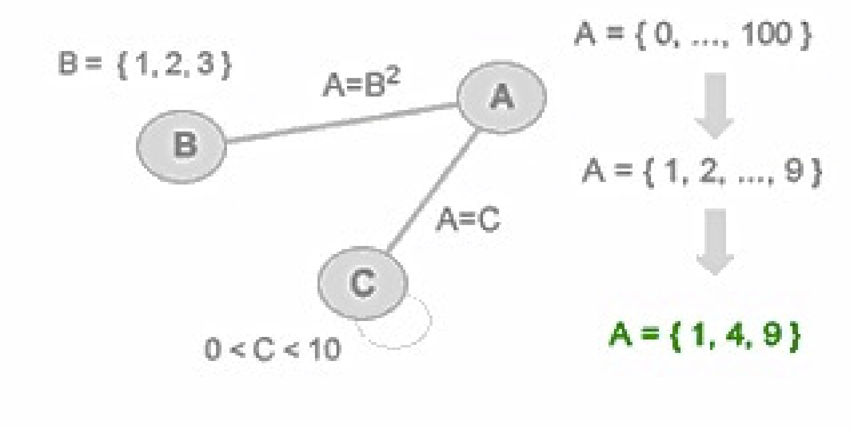
Esercizio 5

L'albero di ricerca con la strategia di ricerca a costo uniforme è riportato in figura. Alla sinistra di ogni stato indicato il path cost, alla destra (in neretto) l'ordine di espansione. Su ogni arco è indicato il costo dell'operatore (non si è indicato esplicitamente l'operatore applicato, che si individua con facilità).



Esercizio 6

Inizialmente il nodo A ha un dominio pari a { 0, ... , 100 } ed è sottoposto a due vincoli binari. Il vincolo binario tra A e C definisce il dominio del nodo A uguale al nodo C (A=C). A sua volta il nodo C è sottoposto a un vincolo unario 0<C<10 che definisce il suo dominio pari a { 1, 2, ... , 9}. In base al vincolo binario (A=C) possiamo ridurre il dominio del nodo A da { 0, ... , 100 } all'insieme dei valori del nodo C { 1, 2, ... , 9 }. L'analisi di consistenza non è ancora finita. Il nodo A è sottoposto anche al vincolo binario con il nodo B che definisce l'uguaglianza tra gli elementi di A con gli elementi di B elevati a potenza (A = B^2). Possiamo ridurre ulteriormente il dominio del nodo A ai soli elementi che soddisfano il vincolo A=B2. Essendo il nodo B composto dagli elementi { 1, 2, 3 } possiamo ridurre il dominio del nodo A da { 1, ..., 9 } a { 1, 4, 9 } i Con il dominio della variabile pari a { 1, 4, 9 } il nodo A è un nodo arco-consistente poiché tutti gli elementi del dominio della variabile A soddisfano i vincoli binari con gli altri nodi collegati (B e C). Il dominio del nodo C deve essere però coerentemente ridotto ai valori per rispettare il vincolo di C=A.



Esercizio 7

Vedi slide del corso.