FONDAMENTI DI INTELLIGENZA ARTIFICIALE – PRIMA PARTE (6 CFU)

19 Gennaio 2011

Tempo a disposizione 2h – Risultato 32/32 punti

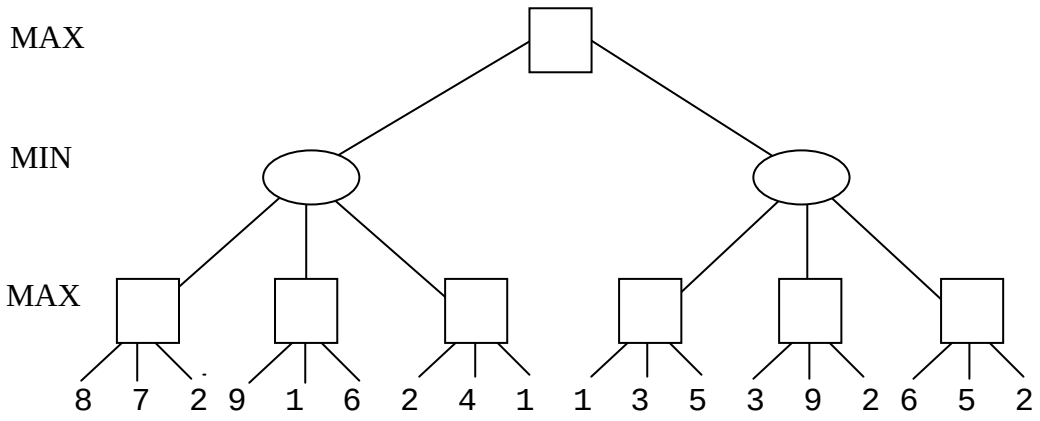
Esercizio 1 (punti 5)

Si formalizzino le seguenti frasi in logica dei predicati del I ordine:
- Gli animali domestici sono grandi o (xor) piccoli
- Gli animali domestici piccoli sono da appartamento
- Gli animali domestici grandi sono da giardino
- Gli elefanti sono animali domestici grandi
- Fuffi è un elefante

Le si trasformi in clausole e si usi la risoluzione per dimostrare che Fuffi è un animale da giardino.

Esercizio 2 (punti 5)

Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore. Si assuma che il primo giocatore sia Max. Si mostri come gli algoritmi min-max e alfa-beta risolvono il problema



Esercizio 3 (punti 4)

Dato il seguente programma Prolog, che stabilisce che “due oggetti che sono a contatto hanno la stessa temperatura” e che “a e b sono a contatto”, “b e c sono a contatto” e “la temperatura di a è 20”.

```
temp(a,20).
temp(X,T):-
    contact(Y,X),
    temp(Y,T).

contact(a,b).
contact(b,c).
```

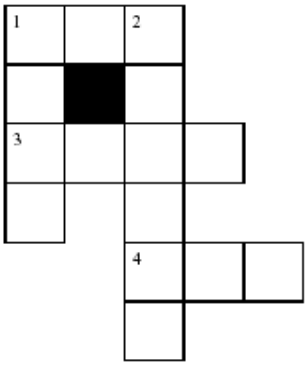
Si disegni l’albero SLD per la query che determina qual è la temperatura di `c` (modellare la query in Prolog). Indicare qual è la riposta calcolata.

Esercizio 4 (punti 6)

Si scriva un programma Prolog e un predicato `p(X)` che è vero se `X` è una lista (non vuota) che contiene tante volte il simbolo `a` quante volte a seguire c’e’ il simbolo `b`.

```
?-p([a,a,a,b,b,b]).
yes
?-p([a,a,a,c,b,b]).
no
```

Esercizio 5 (punti 9)



Si consideri lo schema di parole incrociate raffigurato a fianco.
L'obiettivo è di riempire lo schema con parole selezionate dal seguente vocabolario:
{*ant, ape, big, bus, car, has, bard, book, buys, hold, lane, year, rank, browns, ginger, symbol, syntax*}.
Si formuli il problema come un problema CSP utilizzando le variabili X1h, X1v, X2v, X3h, X4h (dove il pedice numerico indica il numero di riga o colonna, e il pedice h sta per orizzontale e v per verticale)
Si trovi la soluzione utilizzando come euristica per la scelta della variabile da istanziare ad ogni passo il Minimum Remaining Value (MRV) e come propagazione il forward checking (FC). In caso di "parità" rispetto all'euristica MRV, si proceda selezionando la variabile con numero più basso e prima orizzontale di verticale
Per i valori da istanziare si proceda in ordine alfabetico.

Esercizio 6 (punti 3)
Dare la definizione di euristica ammissibile.
Si consideri poi il gioco dell'otto (o gioco del filetto), che consiste nel posizionare 8 tessere numerate da 1 a 8 in una scacchiera 3x3 di 9 caselle, per raggiungere il goal:

1	2	3
4	5	6
7	8	

Dire se l'euristica h = 0,5 × (*numero di caselle fuori posto* + *somma delle distanze Manhattan*) è un'euristica ammissibile per il gioco dell'otto, motivando la risposta.

VOTO:
Esame da 6 CF, il voto è determinato da questa prima 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: ((voto_Iparte +voto_IIparte)/3)*2 e varia quindi da 0 a massimo 32 (equivalente a lode).

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – SECONDA PARTE (3 CFU)
– Tempo a disposizione 30'+15' per DCG – Risultato 16/16 punti

Seminari – Semantic Web (punti 4)
Elencare brevemente gli standard più importanti per la descrizione di basi di conoscenza, definiti nell'ambito dell'iniziativa Semantic Web. Si descriva anche brevemente qual'è lo scopo di ognuno di questi linguaggi.

Metainterprete (punti 9)
Dato un linguaggio proposizionale, con regole "di produzione" del tipo:

```
imply([a,b],d).          % a,b -> d
imply([a,b],e).          % a,b -> e
imply([d],f).            % d -> f
imply([a,f],g).          % a,f -> g
```

Si costruisca un meta-interprete che funziona in modalità forward e che - data una lista di fatti in ingresso priva di ripetizioni - è in grado di derivarne tutte le conseguenze, dirette o indirette.
Ad esempio:

```
?-infer([a,b],X).
Yes X=[d,e,f,g]
```

Si supponga che le implicazioni scritte non siano mai cicliche.
E' dato un predicato che trasforma liste con elementi ripetuti in insiemi (list to set):

```
ltos([],[]).
ltos([H|T],T2):-
     member(H,T),!,
     ltos(T,T2).
     ltos([H|T],[H|T2]):-
     ltos(T,T2).
```

Tale predicato serve per ridurre ad un insieme la lista di fatti (con eventuali fatti ripetuti) inferita nel singolo passo forward.

Prolog e grammatiche (punti 3) – Lo svolge solo chi non ha partecipato alla lez/esercitazione del 19 Novembre 2010
Data la seguente grammatica:
$G = (V_n, V_t, P, S)$
$V_n = \{E, T\}$
$V_t = \{ +, a \}$
P = { E::= T + E | T
T::=a}
S = E

Si scrivano le clausole DCG corrispondenti – come DCG estesa – che producono in uscita l'albero sintattico astratto.

Esempio:

```
?- e(X, [a,+,a,+,a],[]).
yes X = piu(a,piu(a,a)).
```

SOLUZIONE - PARTE I (6 CFU)

Esercizio 1

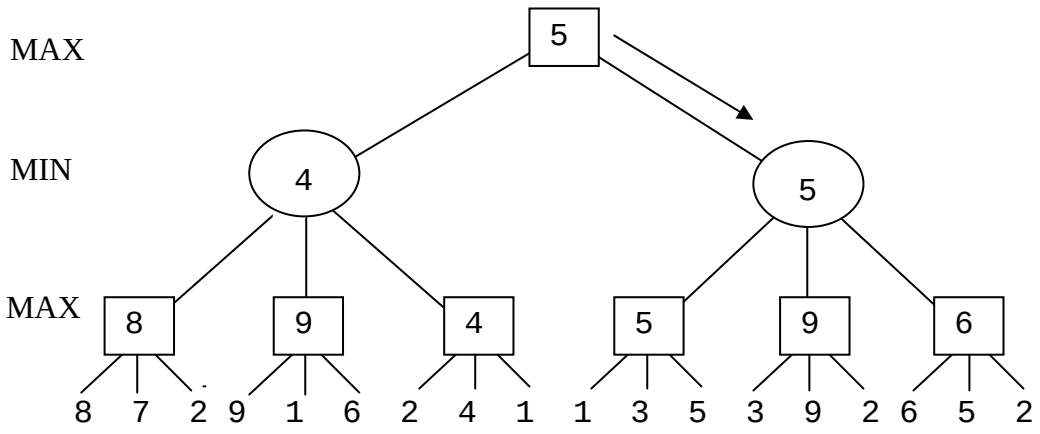
∀ X anim_domestico(X) -> (piccolo(X) ex-or grande(X))
∀ X anim_domestico(X) and piccolo(X) -> appartamento(X)
∀ X anim_domestico(X) and grande(X) -> giardino(X)
∀ X elefante(X) -> grande(X) and anim_domestico(X)
elefante(fuffi)
Query: giardino(fuffi)

Clausole:
C1: not anim_domestico(X) or piccolo(X) or grande(X)
C2: not anim_domestico(X) or not piccolo(X) or not grande(X)
C3: not anim_domestico(X) or not piccolo(X) or appartamento(X)
C4: not anim_domestico(X) or not grande(X) or giardino(X)
C5: not elefante(X) or grande(X)
C6: not elefante(X) or anim_domestico (X)
C7: elefante(fuffi)
Goal: not giardino(fuffi)

Risoluzione:
C8: da C7+C6: anim_domestico(fuffi)
C9: da C7+C5: grande(fuffi)
C10: da C9+C8+Goal in più passi si ottiene la clausola vuota

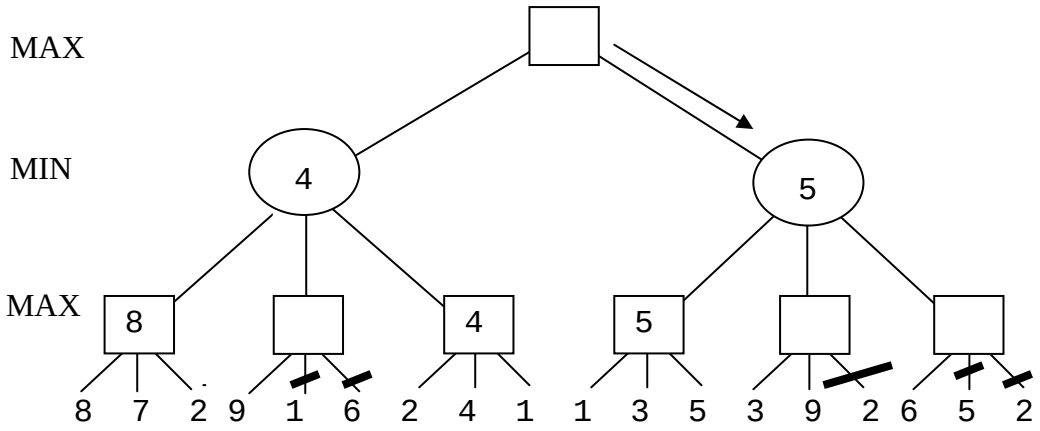
Esercizio 2

min-max:



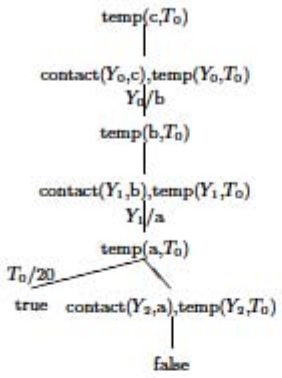
Alfa-beta:

min-max:



Esercizio 3

Albero SLD:



Esercizio 4

```
/* p(Xs) is true if Xs is a list consisting of n a's followed by n b's.    */
p(Xs):-p_1(q0,Xs,[]).

p_1(q1,[],[]).
p_1(q0,[a|Xs],S):-p_1(q0,Xs,[a|S]).
p_1(q0,[b|Xs],[a|S]):-p_1(q1,Xs,S).
p_1(q1,[b|Xs],[a|S]):-p_1(q1,Xs,S).
```

Esercizio 5

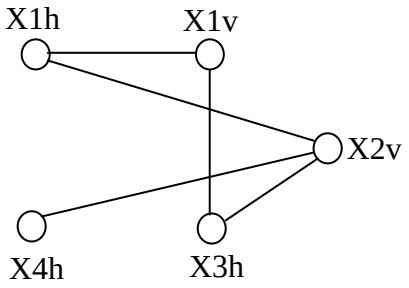
Un modo naturale di formulare il problema è come CSP.

Variabili: X1h, X1v, X2v, X3h, X4h
rispettivamente per le posizioni 1 orizzontale, 1 verticale, 2 verticale, 3 orizzontale, 4 orizzontale.

Domini delle variabili:
X1h = {*ant, ape, big, bus, car, has*}
X1v = {*bard, book, buys, hold, lane, year, rank*}
X2v = {*browns, ginger, symbol, syntax*}
X3h = {*bard, book, buys, hold, lane, year, rank*}
X4h = { *ant, ape, big, bus, car, has*}

Vincoli relativi agli incroci:

Grafo dei vincoli:



X1h[1] = X1v[1]
X1h[3] = X2v[1]
X1v[3] = X3h[1]
X3h[3] = X2v[3]
X2v[5] = X4h[1]

Si noti che nel caso di problemi CSP visti come problemi di ricerca:
- gli stati sono assegnamenti parziali di variabili nel rispetto dei vincoli (a parte in formulazioni a stato completo con riparazioni euristiche)
- lo stato finale è un assegnamento completo che rispetti i vincoli
- le azioni sono assegnare un vincolo ad una nuova variabile non assegnata
- il costo delle azioni tipicamente non è significativo (le variabili vanno comunque assegnate tutte e il costo della soluzione è comunque legato al numero di variabili da assegnare quindi tutte le soluzioni hanno lo stesso costo)

La ricerca di una soluzione che usa le euristiche MRV + forward checking (FC) è riportata nel seguito. L'euristica MRV sceglie la variabile con dominio più piccolo, X2v. Per i valori andiamo in ordine. FC ci autorizza a restringere i domini delle variabili collegate nel grafo dei vincoli.

X2v = browns X1h = { } fail	X2v = ginger X1h = {big} X4h = { } fail	X2v = symbol X1h = {bus, has} X4h = { } fail	X2v = syntax X1h = {bus, has} X3h = {lane, rank} X4h = {ant, ape} X1v = {bard, book, buys, hold, lane, year, rank}

X2v = syntax X1h = bus X3h = {lane, rank} X4h = {ant, ape} X1v = {bard, book, buys}	X2v = syntax X1h = bus X3h = lane X4h = {ant, ape} X1v = { } fail	X2v = syntax X1h = bus X3h = rank X4h = {ant, ape} X1v = {bard}	X2v = syntax X1h = bus X3h = rank X4h = {ant, ape} X1v = bard
X2v = syntax X1h = bus X3h = rank X4h = ant X1v = bard			

Esercizio 6

Noi sappiamo che la somma delle Manhattan distance è un'euristica ammissibile per il gioco dell'otto. Indichiamo con $f(s)$ il numero di caselle fuori posto nello stato s e con $m(s)$ la somma delle Manhattan distance nello stato s. Così abbiamo che per ogni s:

$$h(s) = 0{,}5 \times (f(s) + m(s)) \leq 0{,}5 \times (m(s) + m(s)) = m(s) \quad \text{essendo } f(s) \leq m(s)$$

Quindi per ogni s, $h(s) \leq m(s)$; pertanto anche $h(s)$ è ammissibile.

SOLUZIONE PARTE II (+3 CFU)

Logiche descrittive

Gli standard piu'importanti sono sicuramente URI, XML, RDF e OWL. In particolare RDF e OWL hanno come obiettivo la rappresentazione di basi di conoscenza. RDF permette di descrivere informazioni come triplette della forma "soggetto, verbo, compl. oggetto", o, equivalentemente, "oggetto, proprieta', valore". E' prevalentemente un linguaggio per definire A-Box, cioe' asserzioni di informazioni su oggetti del dominio
del discorso. OWL invece e' proposto per descrivere le relazioni tra le classi (concetti) presenti nel domino del discorso: tipicamente e' utilizzato per definire T-Box.
Tra RDF e OWL si colloca RDF-S, uno Schema (secondo la definizione XML) per RDF, che premette di definire gia' anche informazioni di tipo ontologico, tanto che in buonaparte OWL e RDF-S sono sovrapposti" come capacita' espressiva.

Metainterprete

A) il predicato elimina eventuali duplicati passati in input come fatti. Poi chiama "infer2", che in pratica inferisce tutte le proposizioni vere, compresi quindi i fatti dati in input. Quindi li tolgo dal risultato con subtract.

```
infer(PTemp,R):-
      ltos(PTemp,P),
      infer2(P,RP),
      subtract(RP,P,R).
```

B) Infer2 si "ferma" (dicendo che il risultato equivale alle premesse che ho) quando quello che inferisco è un sottoinsieme (quindi anche equivalente) delle premesse.
Utilizzo "ltos" sul risultato che inferisco dalle premesse perché potrei inferire più volte la stessa cosa (ad esempio con premesse [a,b] e regole a->c, b->c, la findall mi restituisce [c,c]).

```
infer2(P,P):-
      findall(H, (imply(B,H),subset(B,P)), NewRTemp),
      ltos(NewRTemp,NewR),
      subset(NewR,P),!.
```

C) Caso in cui inferisco più proposizioni di quelle che ho fra le premesse. In questo caso aggiungo il risultato inferito alle premesse e richiamo il predicato.

```
infer2(P,R):-
      findall(H, (imply(B,H),subset(B,P)), NewRTemp),
      ltos(NewRTemp,NewR),
      union(P,NewR,NewP),
      infer2(NewP,R).
```

D) list to set era dato nel testo:

```
ltos([],[]).
ltos([H|T],T2):-
      member(H,T),!,
      ltos(T,T2).
      ltos([H|T],[H|T2]):-
      ltos(T,T2).
```

Prolog e grammatiche:

```
% DCG:
e(piu(X,Y)) --> t(X),[+],e(Y).
e(X) --> t(X).
t(a) --> [a].

% Programma Prolog:
e(piu(X,Y),Lin,Lout):-
   t(X,Lin,Lt),piu(Lt,Lt1),
   e(Y,Lt1,Lout).
e(X,Lin,Lout):- t(X,Lin,Lout).
t(a,Lin,Lout):- a(Lin,Lout).
piu([+|L],L).
a([a|L],L).
```