COMPITO SCRITTO DI INTELLIGENZA ARTIFICIALE
14 SETTEMBRE 2001 (Prof. Lamma)

h. 10.30 AULA 13 - Durata 3 ore
(Punteggio 31/; Risultati 21 Settembre)

Esercizio 1: (PUNTI 8)
Si traducano in logica dei predicati del primo ordine le seguenti frasi

- Se un contadino ha due coltivazioni in un anno, queste non sono incompatibili
- I legumi e i cereali sono incompatibili
- I legumi e i tuberi sono incompatibili
- I fagioli sono legumi
- Il grano è un cereale
- Le patate sono tuberi
- Un contadino coltiva nel 2000 fagioli

Si dimostri tramite la risoluzione quali coltivazioni non possono essere fatte nell'anno 2000.

Esercizio 2: (PUNTI 6)
Dato il seguente problema di soddisfacimento di vincoli:

```
X::[1,2,3,4], Y::[1,2,3,4], Z::[1,2,3,4], W::[1,2]
X < Y, X ≠ Z, Z < W, Y ≠ W
```

Si applichi l'arc consistenza alla rete (considerando i vincoli nell'ordine in cui sono scritti) facendo vedere, passo passo, come variano i domini delle variabili del problema. Mostrare poi lo spazio di ricerca generato per trovare tutte le soluzioni utilizzando una strategia forward checking usando l'euristica first fail.

Esercizio 3: (PUNTI 7)
Dato il programma Prolog:

```
contains(t(L,X,R),X).
contains(t(L,V,R),X) :- contains(L,X).
contains(t(L,V,R),X) :- contains(R,X).

depth([],0).
depth(t(L,V,R),D) :-
    depth(L,D1),
    depth(R,D2),
    D1 =< D2,
    D is D2+1.
depth(t(L,V,R),D) :-
    depth(L,D1),
    depth(R,D2),
    D2 =< D1,
    D is D1+1.
```

si rappresenti l'albero SLD relativo alla query:

```
:- depth(A,1), contains(A,1).
```

Si rappresenti l’albero solo fino alla prima soluzione e si fornisca la risposta calcolata.

Esercizio 4: (PUNTI 11)
Scrivere un metainterprete Prolog che selezioni i letterali all'interno del body di una clausola in modo da scegliere a ogni passo della risoluzione sempre quello che ha meno argomenti e, a parità di numero di argomenti, per primi quelli che dopo l'unificazione della testa della clausola hanno il primo argomento legato (quindi successivamente quelli che hanno come primo argomento una variabile non legata). Ad esempio, dato goal:

```
:-r(3,Z).
```

e la clausola:

```
r(X,Y):- q(X,Y), q(Y,X), p(X).
```

la cui testa unifica con `r(3,Z)`, viene invocato per primo il sottogoal `p(X)` perché ha un solo argomento, successivamente `q(X,Y)` perché ha due argomenti di cui il primo non variabile (legato alla costante 3) e infine `q(Y,X)` che ha sempre due argomenti, ma che dopo l'unificazione con la testa della clausola aveva il primo argomento variabile (non legata).

Per semplicità si supponga che le clausole siano memorizzate nel data base nella forma `clausola(Testa,Body)` dove `Testa` è un atomo e `Body` una lista di letterali.
Inoltre, si supponga di avere a disposizione un predicato del tipo `sort(L1,L2,L3)` che restituisce in `L3` il contenuto di `L1` ordinato secondo i valori crescenti contenuti in `L2`.
Esempio `sort([a,b,c],[4,7,1],L3)  L3 = [c,a,b]`

SOLUZIONE

ESERCIZIO 1:

1: ∀X,Y,A coltiva(X,A),coltiva(Y,A)-> not incompatibili(X,Y)
2: ∀X,Y legume(X), cereale(Y) -> incompatibili(X,Y)
3: ∀X,Y legume(X), tubero(Y) -> incompatibili(X,Y)
4: legume(fagioli)
5: cereale(grano)
6: tubero(patata)
7: coltiva(fagioli,2000)
G: ∃ X not coltiva(X,2000)

Forma a clausole

C1: not coltiva(X,A) or not coltiva(Y,A) or not incompatibili(X,Y)
C2: not legume(X) or not cereale(Y) or incompatibili(X,Y)
C3: not legume(X) or not tubero(Y) or incompatibili(X,Y)
C4: legume(fagioli)
C5: cereale(grano)
C6: tubero(patata)
C7: coltiva(fagioli,2000)
Gneg: coltiva(X,2000)

Risoluzione

C8: C1 e C7 not coltiva(Y,2000) or not incompatibili(fagioli,Y)
C9: C8 e Gn not incompatibili(fagioli,X) X/Y
C10: C9 e C2 not legume(fagioli) or not cereale(X)
C11: C10 e C4 not cereale(X)
C12: C11 e C5 clausola vuota con X/grano

ESERCIZIO 2:

Passo 0:

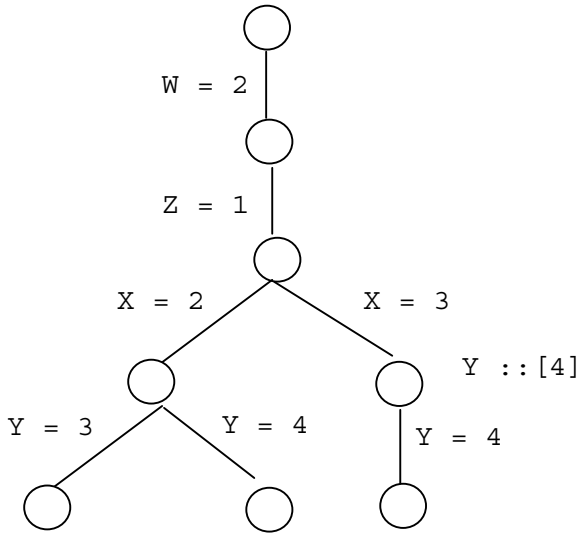
X 1, 2, 3 , 4
Y 1, 2, 3 , 4
W 1, 2,
Z 1, 2, 3 , 4

Passo 1:

X 1, 2, 3
Y 2, 3 , 4
W 2
Z 1

Passo 2:

X 2, 3
Y 3, 4
W 2
Z 1



Ci sono tre soluzioni possibili:
X=2, Y=3, W=2, Z=1
X=2, Y=4, W=2, Z=1
X=3, Y=4, W=2, Z=1

ESERCIZIO 3:
La risposta calcolata è:
:- depth(A,1), contains(A,1).
Yes, A=t([], 1, t([], V, []))

depth(A,2), contains(A,1).

A/t(L,V,R)

depth(L,D1), depth(R,D2), D1 =< D2, 2 is D2+1,
contains(t(L,V,R),1).

L/[], D1/0

depth(R,D2), 0 =< D2, 2 is D2+1, contains(t([],V,R),1).

R/[], D2/0

0 =< 0, 2 is 0+1, contains(t([],V,[]),1).

contains(t([],V,[]),1).

true

ESERCIZIO 4:

```
solve([]):- !.
solve([A|B]) :- !,
      solve(A),
      solve(B).
solve(A) :-
      clausola(A,B),
      ordina(B,NewB),
      solve(NewB).

ordina(B, BodyOrd) :-
      conta(B, NumArg),
      sort(B,NumArg,ListOrd),
      ordinaprimopar(ListOrd,BodyOrd).

ordinaprimopar(ListOrd,ListOrd1):-
      gruppistessipar(ListOrd,Gruppi),
      swapgruppi(Gruppi,Swapped),
      appiattiscilista(Swapped,ListOrd1).

gruppistessipar([],[]).
gruppistessipar ([El1|Altri], [StessoN|AltreL]):-
      contapar(El1,N),
      scandisci(Altri,N,StessoN,AltriN),
      gruppistessipar(AltriN, AltreL).

scandisci([],_,[],[]).
scandisci([El|T],N,[El|T1],L1):-
      contapar(El,N), !,
      scandisci(T,N,T1,L1).
scandisci([El|T],N,[],T).

swapgruppi([],[]).
swapgruppi([Gruppo|Altri],[Swapped|AltriS]):-
      dividi(Gruppo,Primi,Secondi),
      append(Primi,Secondi,Swapped),
      swapgruppi(Altri,AltriS).

dividi([],[],[]).
dividi([El|T],[El|P],S):-
      El=..[_,N|_],
      ground(N), !,
      dividi(T,P,S).
dividi([El|T],P,[El|S]):-
      dividi(T,P,S).

conta([],[]).
conta([El|Altri],[N,AltriN]):-
      contapar(El,N),
      conta(Altri,AltriN).

contapar(A,N):-
      A=..[_|L],
      length(L,N).
```