COMPITO DI APPLICAZIONI DI INTELLIGENZA ARTIFICIALE
16 dicembre 2005 (Punteggio su 30/30; Tempo 2h)

Esercizio 1 (punti 8)
Dato il seguente training set S:

Colore	**Numero di ruote**	**Classe**
Rosso	2	+
Verde	4	+
Blu	2	-
Blu	4	+
Rosso	?	-
Verde	4	+
Blu	2	-
Verde	2	-
Rosso	3	-
Verde	3	+
Rosso	3	-
Rosso	?	+
Blu	2	-

a) Si calcoli l'entropia del training set rispetto all'attributo Classe (punti 1)
b) Si calcoli il rapporto di guadagno dei due attributi rispetto a questi esempi di training (punti 4)
c) si costruisca un albero decisionale ad un solo livello per il training set dato, indicando le etichette delle foglie (numero di esempi finiti nella foglia/numero di esempi finiti nella foglia non appartenenti alla classe della foglia). (punti 1,5)
d) si classifichi l'istanza: (punti 1,5)

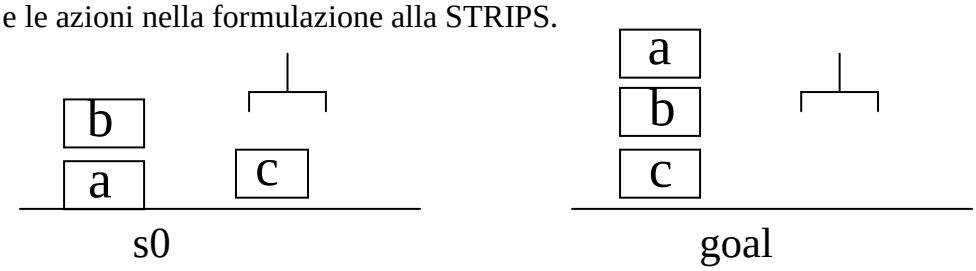
Verde	?

Esercizio 2 (punti 8)
Un mercante aveva tre figli che alla sua morte si dovevano dividere l'eredità. L'unica difficoltà si ebbe nel dividere le scorte di miele: il mercante aveva 21 barili di miele, di cui 7 pieni, 7 a metà e 7 vuoti. Il mercante lasciò nel testamento che ciascuno dei figli avrebbe dovuto avere la stessa quantità di miele e lo stesso numero di barili. Inoltre, non si doveva travasare del miele da un barile all'altro perché si sarebbe sprecato del miele. Sapendo che ciascuno dei figli non voleva avere più di 4 barili dello stesso tipo (pieni, vuoti, metà), si scriva un programma CLP che risolve il problema.
Alcuni predicati utili:
matrix(*+N, +M, -Rows, -Cols*): crea una matrice di variabili, data come lista di liste, organizzata in N righe e M colonne. La matrice è fornita sia come lista di righe sia come lista di colonne (ovvero viene anche fornita la trasposta). Es
?- matrix(2,3,Rows,Cols)
Rows = [[A,B,C],[D,E,F]]
Cols = [[A,D],[B,E],[C,F]]
flatten(*+NestedList, ?FlatList*): data una lista di liste, fornisce una lista "piatta"
?- flatten([A,B,[C,D],E],L)
L = [A,B,C,D,E].

sumlist(+List, ?Sum): Vincolo che lega una lista di variabili con dominio con la variabile che rappresenta la loro somma

Esercizio 3 (punti 8)
Si faccia riferimento al seguente esempio relativo al "mondo a blocchi".
Ci sono tre blocchi, a, b e c, che possono essere sovrapposti o posizionati su un tavolo; attraverso una mano robotica è possibile impilarne uno su un altro (stack(X,Y)) o de-impilarne uno dall'altro (unstack(X,Y)), posizionare un blocco trattenuto dalla mano sul tavolo (putdown(X)) o prelevare un blocco libero dal tavolo (pickup1(X)) o da sopra un altro blocco (pickup2(X)) trattenendolo nella mano robotica.

Si modelli il seguente stato iniziale (s0), il goal corrispondente a uno stato finale in cui a è su b e b su c e le azioni nella formulazione alla STRIPS.



Si descriva poi come l'algoritmo lineare backward STRIPS trova un piano per questo goal a partire da s0. Si descriva lo stato e lo stack dei goal passo passo.

Esercizio 4 (punti 6)
Scrivere un metainterprete per Prolog puro con regola di calcolo right-most-

SOLUZIONE

Esercizio 1:

a) info(S)=-6/13*$\log_2$ 6/13-7/13*$\log_2$ 7/13=0.996

b)

$info_{Colore}$(S)=5/13*(-2/5*$\log_2$2/5-3/5*$\log_2$ 3/5)+4/13*(-3/4*$\log_2$3/4-1/4*$\log_2$ 1/4)+4/13*(-1/4*$\log_2$ 1/4-3/4 * $\log_2$ 3/4)=

=0.385*0.971+0.308*0.811+0.308*0.811=0.873

gain(Colore) =0.996-0.873=0,123

splitinfo(Colore)=-5/13*$\log_2$(5/13)-4/13*$\log_2$(4/13)-4/13*$\log_2$(4/13)=1.577

gainratio(Colore) =0.123/1.577=0.078

Per caclolare il guadagno dell'attributo Dimensione non si usa l'entropia calcolata su tutto il training set ma solo sugli esempi che hanno Dimensione noto (insieme F):

info(F)=-5/11*$\log_2$ 5/11– 6/11*$\log_2$ 6/11=0.994

$info_{Ruote}$(F)=5/11*(-1/5*$\log_2$1/5-4/5*$\log_2$ 4/5)+3/11(-1/3*$\log_2$ 1/3-2/3*$\log_2$ 2/3) +3/11(-3/3*$\log_2$ 3/3-0/3*$\log_2$ 0/3) =

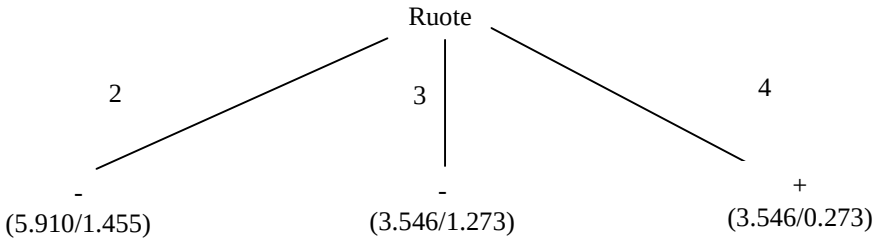
=0.455*0.722+0.273*0.918+0.273*0=0.579

gain(Ruote) =11/13*(0.994-0.579)=0.351

splitinfo(Ruote) =-5/13*$\log_2$(4/13)-3/13*$\log_2$(4/13)-3/13*$\log_2$(3/13)-2/13*$\log_2$(2/13)=1.922

gainratio(Ruote) =0.351/1.922=0.183

c)



d) l'istanza viene divisa in tre parti, di peso rispettivamente 5.910/13=0.455, 3.546/13=0.273 e 3.546/13=0.273. La prima parte viene mandata lungo il ramo 1 e viene classificata come - con probabilità 4.455/5.910 =75.4% e come + con probabilità 1-75.4%=24.6%. La seconda parte viene mandata lungo il ramo 2 e viene classificata come - con probabilità 2.273/3.546=64.1% e come + con probabilità 1-64.1%=35.9%. La terza parte viene mandata lungo il ramo 3 e viene classificata come + con probabilità 3.273/3.546=92.3% e come - con probabilità 1-92.3%=7.7%. Quindi in totale la classificazione dell'istanza è

+: 0.455*24.6%+0.273*35.9%+0.273*92.3%=46.2%

-: 0.455*75.4% + 0.273*64.1%+0.273*7.7%=53.9%

Esercizio 2

```
:- lib(fd).
:- lib(fd_global).
:- lib(matrix_util).

solve(Figli):-
    matrix(3, 3, Figli, Barili),
    flatten(Figli,Flat),
    % ciascun figlio non vuole piu` di 4 barili dello stesso tipo
    Flat :: 0..4,
    % Il totale di miele deve essere uguale per tutti
    totMiele(Figli,TotMiele),
    % Il numero di barili e` uguale per tutti
    somma_uguale(Figli,TotBarili),
    % Per colonne: il totale di barili di ogni tipo e` uguale a 7
    somma_uguale(Barili,7),
    labeling(Flat).

totMiele([],_).
totMiele([[Vuoto,Mezzo,Pieno]|T],Tot):-
    Vuoto*0+Mezzo+Pieno*2 #= Tot,
    totMiele(T,Tot).

% Impone che tutte le liste della lista di liste diano la stessa somma
somma_uguale([],_).
somma_uguale([H|T],Tot):-
    sumlist(H,Tot),
    somma_uguale(T,Tot).
```

Esercizio 3

Stato inziale: **[ontable(a), ontable(c), on(b,a), clear(b), clear(c), handempty]**
Goal: **on(b,c), on(a,b)**

Azioni:

pickup1(X)
PRECOND: ontable(X), clear(X), handempty
DELETE: ontable(X),clear(X), handempty
ADD: holding(X)

pickup2(X)
PRECOND: on(X,Y), clear(X), handempty
DELETE: on(X,Y),clear(X), handempty
ADD: clear(Y), holding(X)

putdown(X)
PRECOND: holding(X)
DELETE: holding(X)
ADD: ontable(X), clear(X), handempty

stack(X,Y)
PRECOND: holding(X), clear(Y)
DELETE: holding(X), clear(Y)
ADD: handempty, on(X,Y), clear(X)

unstack(X,Y)
PRECOND: handempty, on(X,Y), clear(X)
DELETE: handempty, on(X,Y), clear(X)
ADD: holding(X), clear(Y)

Costruzione del piano:

Stato:
ontable(a)
ontable(c)
on(b,a)
clear(b)
clear(c)
handempty

Stack:
on(b,c), on(a,b)

Goal regression con l'azione **stack(X,Y) : X/b,Y/c**

Stato:
ontable(a)
ontable(c)
on(b,a)
clear(b)
clear(c)
handempty

Stack:
holding(b) and clear(c)
stack(b,c)
on(a,b)

Stato:
ontable(a)
ontable(c)
on(b,a)
clear(b)
clear(c)
handempty

Stack:
clear(c)
holding(b)
stack(b,c)

on(a,b)

Stato:
ontable(a)
ontable(c)
on(b,a)
clear(b)
clear(c)
handempty

Stack:
holding(b)
stack(b,c)
on(a,b)

Goal regression con **pickup2(b):**

Stato:
ontable(a)
ontable(c)
on(b,a)
clear(b)
clear(c)
handempty

Stack:
clear(b) and handempty //entrambe vere nello stato corrente
pickup2(b)
stack(b,c)
on(a,b)

Stato:
ontable(a)
ontable(c)
on(b,a)
clear(b)
clear(c)
handempty

Stack:
pickup2(b)
stack(b,c)
on(a,b)

Esecuzione di **pickup2(b)**

Stato:
ontable(a)
ontable(c)
clear(a)
clear(c)
holding(b)

Stack:
stack(b,c)
on(a,b)

Esecuzione di **stack(b,c)**

Stato:
ontable(a)
ontable(c)
clear(a)
on(b,c)
clear(b)
handempty

Stack:
on(a,b)

Goal regression con l'azione **stack(X,Y) : X/a,Y/b**

Stato:
ontable(a)
ontable(c)
clear(a)
on(b,c)
clear(b)
handempty

Stack:
holding(a) and clear(b) //la seconda vera nelo stato corrente
stack(a,b)
on(a,b)

Goal regression con **pickup1(a):**

Stato:
ontable(a)
ontable(c)
clear(a)
on(b,c)
clear(b)
handempty

Stack:
clear(a) and handempty //entrambe vere nello stato corrente
pickup1(a)
stack(a,b)
on(a,b)

Esecuzione di **pickup1(a)**

Stato:
ontable(c)
on(b,c)

clear(b)
holding(a)

Stack:
stack(a,b)
on(a,b)

Esecuzione di **stack(a,b)**

Stato:
ontable(c)
on(b,c)
clear(b)
on(a,b)
clear(a)
handempty

Stack:
on(a,b) // vera nello stato raggiunto