Esercizi su CLP(FD)

Dott. Ric. Marco Gavanelli

Langford

Un bambino ha due collezioni di N blocchi numerati da 1 a N e vuole mettere i blocchi in sequenza in modo che:

- *i due blocchi '1' abbiano 1 blocco che li separa*
- *i due blocchi '2' abbiano 2 blocchi che li separano*
- *...*
- *i due blocchi 'N' abbiano N blocchi che li separano.*

Ad esempio, una soluzione per N=4 è la seguente:

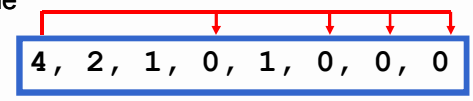
4 1 3 1 2 4 3 2

Si scriva un programma CLP per ECL^{i}PS^e che, dato un numero N, risolve questo problema.

Suggerimento Si noti che non è richiesto che l'output sia fornito nello stesso formato indicato in precedenza: lo studente è libero di mostrare l'output come preferisce.

Sequenza magica

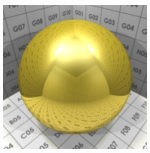
- Dato un insieme di n+1 variabili $X_0,\ldots,X_n$. Ogni X_i deve rispettare i seguenti vincoli :
  - 0 compare X_0 volte in soluzione
  - 1 compare X_1 volte
  - …
  - n appare X_n volte.



- Si scriva un predicato `magic(L,N)` che, dato un numero N, produce una sequenza magica di lunghezza N
- Si provi ad aggiungere vincoli ridondanti: come cambia l'efficienza?

Divisioni

- *Due fratelli trovano un tesoro composto da N sfere d'oro i cui diametri sono* 1, 2, 3, 4,....., *N*-1, *N rispettivamente. Come devono dividersele per ottenere lo stesso peso?*
- *(Il peso di una sfera dipende dal diametro al cubo)*



Es 1: Modello

- *Variabili: per ogni sfera s ho una variabile Xs che vale*
  - *0 se la sfera è assegnata al primo fratello*
  - *1 se la sfera è assegnata al secondo fratello*
- *Vincoli:*
  - *il peso totale delle sfere assegnate al secondo fratello è pari a metà del peso totale*
    - *Nota che se calcolo $\sum_i (X_i\, i^3)$ ottengo il peso delle sfere assegnate al 2 fratello*

Es1: Codice ECLiPSe

```
sfere(L,N) :-
    peso_totale(N,Tot),        % calcolo peso tot (non è un
                               vincolo)
    length(L,N),               % Costruisco una lista di N var
    L :: 0..1,                 % Domini
    sommapesi(L,N,Sum), % Calcolo la somma degli elem
    della lista L moltiplicati per 1..N (peso 2 fratello)
    Sum * 2 #= Tot, % peso tot = 2*peso secondo fratello
    labeling(L).
```

Es 1: Altri predicati

```
% Calcola la somma dei primi N numeri elevati al cubo
peso_totale(1,1) :-!.
peso_totale(N,Tot) :-
    N1 is N-1,
    peso_totale(N1,Tot1),
    Tot is Tot1+N*N*N.
```

```
% Impone che S sia la somma ∑_i (X_i i^3)
sommapesi([X],1,X) :-!.
sommapesi([H|T],N,S) :-
    N1 is N-1,
    sommapesi(T,N1,S1),
    S #= S1+N*N*N*H.
```


Quadrato magico

- *Un quadrato magico di ordine N è una matrice NxN che contiene i numeri interi da 1 a N^2, tale che*
  - *la somma dei numeri su ogni riga*
  - *la somma dei numeri su ogni colonna*

    sia sempre costante
- *Si scriva un predicato che prende in ingresso un numero N e fornisce un quadrato magico di ordine N.*

2	7	6	→ 15
9	5	1	→ 15
4	3	8	→ 15
↓	↓	↓	
15	15	15	

- *Predicati utili:*
  - `matrix(N,M,Righe,Colonne)` *genera una matrice NxM di variabili, data sia per Righe che per Colonne. Usare lib(matrix_util)*

    ES: matrix(2,3,R,Col) fornisce

    R= [[A,B,C], [D,E,F]]

    Col=[[A,D], [B,E], [C,F]]
  - `flatten(ListaDiListe,Lista)` *appiattisce una lista di liste in una lista semplice*

    Es: flatten([A,[1,X],[[3]]],L) fornisce L=[A,1,X,3].
- *Sfida: Cercate di valutare i tempi di calcolo e di generare il quadrato più grande nel minor tempo possibile*

Calcoletti

- *In ciascuna delle seguenti operazioni, le cinque "x" vanno sostituite con 5 cifre consecutive non necessariamente nell'ordine naturale.*

1. *xx = x+x+x*
2. *x*x = x+x+x*
3. *x*x = xx+x*
4. *xx*x = xx*

Scrivere un programma ECLiPSe che trova i valori che risolvono le equazioni

Scheduling

- 6 attività devono essere assegnate a due macchine: m1 e m2. Ogni macchina può eseguire una sola attività alla volta.
- Ogni attività è descritta da un fatto, con la seguente struttura:

```
task(ID,DURATA,MACCHINA).
```

- I dati sulle attività sono i seguenti

```
task(1,3,m1).
task(2,8,m1).
task(3,8,m1).
task(4,6,m2).
task(5,3,m2).
task(6,4,m2).
```

- *Le attività possono cominciare in un istante >=0 e devono essere terminate entro l'istante 20*
- *Si calcoli il tempo di inizio per tutte le attività*
- *Si supponga poi che l'attività 2 possa essere svolta solo dopo che la 4 è terminata.*

Cavalieri & Mascalzoni

- *C'è un'isola abitata esclusivamente da cavalieri e da mascalzoni. I cavalieri dicono sempre la verità, i mascalzoni mentono sempre. Su un autobus ci sono N passeggeri. Alla domanda "quanti cavalieri ci sono in questo autobus?" rispondono così:*
- *primo passeggero : al massimo 1*
- *secondo : al massimo 2*
- *terzo : al massimo 3*
- *...*
- *N-esimo : al massimo N*
- *Si scriva un programma ECLiPSe che, dato N, fornisce una lista con N elementi in cui l'i-esimo elemento assume valore 1 se il corrispondente passeggero è un cavaliere.*

Noci di cocco

- *Cinque uomini e una scimmia fecero naufragio su un'isola deserta e passarono il primo giorno a raccogliere noci di cocco. Poi le ammucchiarono tutte insieme e andarono a dormire. Ma mentre tutti dormivano uno di essi si svegliò e pensando che il mattino dopo ci sarebbero stati dei litigi alla spartizione decise di prendersi la sua parte. Perciò divise le noci in cinque mucchi uguali. Rimaneva una noce che egli dette alla scimmia, poi nascose la sua parte e mise tutto il resto assieme. Subito dopo un secondo uomo si svegliò e fece lo stesso. Anch'egli dette una noce residua alla scimmia. Uno dopo l'altro tutti e cinque gli uomini fecero la stessa cosa, ognuno prendendo un quinto del mucchio che trovava svegliandosi e dando una noce alla scimmia.*
- *Alla mattina seguente i cinque uomini si divisero le noci che restavano e la divisione non diede resto.*
- *Si scriva un programma CLP che calcola quante noci c'erano all'inizio (supponendo che fossero meno di 10.000).*

Cellulare (23 mar 2006)

- *Un cellulare ha un microprocessore che può funzionare a tre frequenze di clock (1, 2 e 3 MHz).*
- *Il cellulare deve eseguire la stessa sequenza di operazioni ogni TD millisecondi.*
- *Ciascuna operazione ha una durata base DB, che è la durata dell'operazione se eseguita a 1MHz (se eseguita a 2MHz la durata sarà DB/2, se eseguita a 3MHz sarà DB/3). Si supponga che le durate base siano multipli di 6ms.*
- *Ciascuna delle operazioni consuma energia dalla batteria. Il consumo per un'operazione di durata DB eseguita alla frequenza F è pari a E = (2*F-1)* DB.*
- *Si scriva un predicato CLP* `cell(ListaDurateBase,TD,ListaFreq)`
- *che prende in ingresso*
  - `ListaDurateBase`*: le durate base delle operazioni da svolgere,*
  - `TD`*: il tempo entro cui tutte le operazioni devono essere svolte*
- *e che fornisce in uscita la* `ListaFreq`*, che contiene le frequenze a cui devono essere svolte le varie operazioni. Il predicato deve calcolare per ogni operazione la frequenza a cui deve essere eseguita, in modo tale che tutte le operazioni vengano svolte entro il tempo TD, minimizzando l'energia consumata.*

N-Regine

- *Si risolva il problema delle N Regine usando il vincolo alldifferent*
- *suggerimento: per evitare che le regine si attacchino in diagonale, si può fare così:*
  - *creo una lista LD in cui*
    - *il primo elemento è la pos della regina 1*
    - *il secondo è la pos della regina 2 a cui aggiungo 1*
    - *…*
    - *l'n-esimo elemento è la pos della regina n a cui aggiungo (n-1)*

    A questo punto impongo il vincolo di alldifferent sulla lista LD
  - *stessa cosa per la diagonale secondaria*

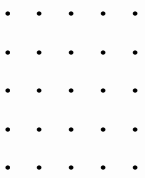
N Regine (2)

- Si utilizzi per assegnare i valori alle variabili il predicato labeling. Si verifichi fino a quante regine è possibile risolvere il problema in mezzo minuto.
- Si utilizzi ora il labeling con 1st fail principle. Fino a quale dimensione si riesce a risolvere il problema?
- Si utilizzi LDS.
- Si trovi ora la soluzione che rende minima la massima distanza delle regine dall'origine degli assi (per semplicità si minimizzi la distanza al quadrato dall'origine: D2=X*X+Y*Y). Si provi con i 3 metodi descritti sopra. Che cosa si può notare?

N Regine (3)

- Si implementi il calcolo della distanza con un nuovo vincolo dist(X,Y,D) che fa in modo

A e B



- Sostituire 14 puntini con sette "A" e sette "B" in modo tale che su ciascuna colonna e ciascuna riga si contino lo stesso numero di A e B.
- In altre parole, ogni allineamento deve contenere una A ed una B, oppure due A e due B.
- Quante soluzioni ci sono?

Which of the following statements are true and which are false?

i) The answers to #6 and #7 are the same.

ii) #1 is false.

iii) The answers to #4 and #20 are different.

iv) The answers to #3 and #20 are different.

v) The answer to this statement is different from the answer to #19.

vi) #2 is true.

vii) #15 is true.

viii) The answers to #11 and #19 are the same.

ix) #10 is true.

x) #13 is false.

xi) Mrs. Jones is allergic to strawberries.

xii) #16 is true.

xiii) #12 is true.

xiv) The answer to this statement is the same as the answer to #11.

xv) At least half the statements in this problem are false.

xvi) At least half the statements in this problem are true.

xvii) The answers to #9 and #4 are the same.

xviii)#7 is true.

ixx) Mrs. Jones first name is Shirley.

xx) The answers to #3 and #4 are different.

Marmellata

- *Di 6 bambini si sa che esattamente 2 hanno rubato la marmellata. Ma chi?*
- *Aldo dice " Dario ed Elio"*
- *Bruno dice "Carlo e Franco"*
- *Carlo dice "Franco ed Elio"*
- *Dario dice "Aldo ed Elio"*
- *Elio dice "Carlo e Bruno"*
- *Franco è assente.*
- *Uno dei 5 bambini ha nominato due innocenti. Gli altri hanno nominato un innocente ed un colpevole.*
- *Si scriva un programma CLP che scopre chi ha rubato la marmellata.*
- *Suggerimento: per ogni bambino, dobbiamo scoprire due cose: 1. se ha mentito sempre o ha detto una verità, 2: se ha rubato la marmellata.*

Sudoku

Il Sudoku è un rompicapo che si gioca su una scacchiera 9x9. In ciascuna casella va scritto un numero da 1 a 9, in modo che

- *in ogni riga non ci siano due numeri uguali*
- *in ogni colonna non ci siano due numeri uguali*
- *la scacchiera è suddivisa in 9 sottoscacchiere 3x3: in ciascuna di queste sottoscacchiere non ci devono essere due numeri uguali*

Si scriva un programma CLP che risolve il rompicapo.

Alcuni predicati utili (vedere manuale di ECLiPSe):

- *matrix(+NRows, +NCols, -Rows, -Cols)*
- *flatten(+NestedList, ?FlatList)*
- *append(A,B,C)*

4	2	1	9	6	7	8	5	3
6	7	5	3	1	8	4	9	2
3	8	9	2	4	5	6	1	7
1	9	8	7	3	4	5	2	6
7	4	2	8	5	6	1	3	9
5	6	3	1	2	9	7	4	8
2	1	6	5	7	3	9	8	4
8	3	7	4	9	1	2	6	5
9	5	4	6	8	2	3	7	1

Es 2: Contenitori

Ci sono 10 contenitori con capacità: 1,2,4,5,6,12,15,22,24,38 litri. Sapendo che

- *uno è pieno di latte,*
- *alcuni sono pieni di vino ed altri pieni di acqua,*
- *uno è completamente vuoto*
- *il vino è il doppio dell'acqua*
- *la quantità di acqua è il doppio della quantità di latte,*

stabilire il contenuto di ciascun recipiente.

Es 2: Modello: Variabili

- *una variabile per ogni contenitore. Ogni variabile rappresenta il tipo di liquido contenuto. Una lista Ltipo. Domini: possibili contenuti [acqua, latte, vino, vuoto]*
- *Inoltre, serve sapere la quantità di acqua, latte, vino per ogni contenitore: per ogni coppia (contenitore,possibile contenuto) creo una variabile. Ho 3 liste: Llatte, Lvino, Lacqua. I valori che possono assumere sono*
  - *0 se il contenitore non contiene il liquido stabilito*
  - *capacità del contenitore se contiene il liquido stabilito*

  Es, Llatte=[0,2,0,5,…] vuol dire che il primo contenitore non contiene latte, il secondo ne ha 2 litri, ecc.
- *Quantità totali: 3 variabili Qlatte, Qvino, Qacqua*

Es 2: Modello: Vincoli

- *"uno è pieno di latte, uno è completamente vuoto"*

  occurrences(latte, Ltipo, 1),

  occurrences(vuoto, Ltipo, 1)
- *Per ogni contenuto, calcolo il totale*

  sumlist(Llatte,Qlatte), sumlist(Lvino,Qvino),

  sumlist(Lacqua,Qacqua),
- *"il vino è il doppio dell'acqua, l'acqua è il doppio del latte"*

  *Qvino #= Qacqua*2, Qacqua #= Qlatte *2*

Es 2: Modello: Vincoli

- *Per ogni contenitore di capacità C, voglio che se contiene latte la corrispondente lista Llatte abbia il corrispondente elemento = C e le altre liste = 0.*
- *Scrivo un predicato*

  quantita(contenuto,Lcapa,Ltipo,Lcontenuto)

  che, per ogni contenitore, considera la coppia (contenitore,contenuto) e mette in Lcontenuto C o 0.
- *Posso quindi invocarlo:*

  quantita(latte,Lcapa,Ltipo,Llatte),

  quantita(vino,Lcapa,Ltipo,Lvino),

  quantita(acqua,Lcapa,Ltipo,Lacqua)

Es 2: Vincolo quantità

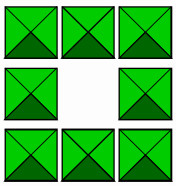
```
quantita(_,[],[],[]).
quantita(X,[Cap|Lcapa],[Tipo|Ltipo],[Q|Lq]):-
    vincolo_seleziona(X,Cap,Tipo,Q),
    quantita(X,Lcapa,Ltipo,Lq).
```

- *vincolo_seleziona: se X=Tipo, allora Cap=Q, altrimenti Q=0*
- *Possiamo implementarlo con Propia:*
  - vincolo_seleziona(X,Cap,Tipo,Q):-
    seleziona(X,Cap,Tipo,Q) infers fd.
  - seleziona(X,C,X,C).
    seleziona(X,_,Y,0):- X #\= Y.

Es 2: Codice ECLiPSe

```
:- lib(fd), lib(fd_global), lib(propia).
litri(Lcapa,Ltipo):-
    length(Lcapa,N),    length(Ltipo,N), % Creo lista della stessa lungh
    Ltipo :: [latte,vino,acqua,vuoto], % Domini
    occurrences(latte, Ltipo, 1),    occurrences(vuoto, Ltipo, 1),
    quantita(latte,Lcapa,Ltipo,Llatte),    quantita(vino,Lcapa,Ltipo,Lvino),
    quantita(acqua,Lcapa,Ltipo,Lacqua),
    sumlist(Llatte,Qlatte),    sumlist(Lvino,Qvino),
    sumlist(Lacqua,Qacqua),
    Qvino #= Qacqua*2,    Qacqua #= Qlatte *2,
    append_lists([Ltipo,Lacqua,Lvino,Llatte],Ltot), % fare per esercizio
    labeling(Ltot).
```

Es 3: Guerrieri



- *Un sultano decide di far risiedere i suoi 24 guerrieri in un accampamento quadrato avente 8 tende, in modo che ne rimanessero 3 per ogni tenda.*
- *Volendo essere sicuro che tutte le sere essi rimanessero nell'accampamento, egli decise di mandare uno schiavo a controllare che ce ne fossero 9 per ogni lato del quadrato dell'accampamento.*

•Nonostante lo schiavo confermasse tutte le sere che vi erano 9 persone per lato, gli scaltri guerrieri erano riusciti ad ingannare il sultano:

•La prima alcuni di loro sono andati a gozzovigliare al villaggio.

•La seconda notte hanno invece avuto ospiti dei vinai e dei coppieri dal villaggio.

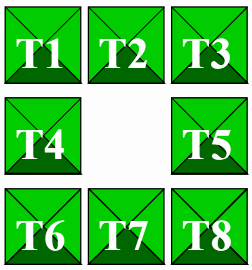
•Quali sono il numero massimo e quello minimo di persone nell'accampamento, senza che il sultano se ne accorga?

Es 3: Modello

- *Variabili: numero di guerrieri per ogni tenda*
- *Vincoli: somma su righe e colonne = 9*
- *Funzione obiettivo: somma di tutti le variabili*

Es 3: Codice ECLiPSe

```
guerrieri(L,Tot):-
    L= [T1,T2,T3,T4,T5,T6,T7,T8],
    L :: 0..72,
    sumlist([T1,T2,T3],9),
    sumlist([T1,T4,T6],9),
    sumlist([T3,T5,T8],9),
    sumlist([T6,T7,T8],9),
    sumlist(L,Tot),
    labeling(L).
minimo(L,Tot):-   minimize(guerrieri(L,Tot),Tot).
massimo(L,Tot):-   MenoTot + Tot #= 0,
    minimize(guerrieri(L,Tot),MenoTot).
```



Es 4: Analisi: $A^2+B^2 <= C$

```
sq(A, B, C) :-
    dvar_domain(A, DomA), dom_range(DomA, MinA, MaxA),
    dvar_domain(B, DomB), dom_range(DomB, MinB, MaxB),
    MiA2 is MinA*MinA, MaB2 is MaxB*MaxB,
    (MiA2 + MaB2 > C
    ->      NewMaxB is fix(sqrt(C - MiA2)),
            dvar_remove_greater(B, NewMaxB)
    ;       NewMaxB = MaxB ),
    MaA2 is MaxA*MaxA, MiB2 is MinB*MinB,
    (MaA2 + MiB2 > C
    ->      NewMaxA is fix(sqrt(C - MiB2)),
            dvar_remove_greater(A, NewMaxA)
    ;       NewMaxA = MaxA ),
    (NewMaxA*NewMaxA + NewMaxB*NewMaxB =< C -> true
    ; suspend(sq(A, B, C), 3, (A, B)->fd:min) ),
    wake.
```

Es 4: Analisi

- *Sapendo che*
  - *sqrt calcola la radice quadrata*
  - *fix calcola il troncamento*
- *dire*
  - *se il vincolo sq(A,B,C) funziona come vincolo ternario*
  - *che livello di consistenza ottiene*
- *Si mostri poi come ECLiPSe risponde alla query*

  [X,Y]::1..5, X+Y #> 5, sq(X,Y,20)

  mostrando i passaggi

Es 4: Soluzione

[X,Y]::1..5, X+Y #> 5, sq(X,Y,20)

- *X+Y #> 5* *X::1..5,* *Y:: 1..5* *sospeso*
- *sq(X,Y,20)* *X::1..4,* *Y::1..4* *sospeso*
- *X+Y #> 5* *X::2..4,* *Y::2..4* *sospeso*
- *sq(X,Y,20)* *nessuna modifica, sospeso*
- *Risposta di ECLiPSe:*
  - *yes,*
  - *X::2..4, Y::2..4*
  - *delayed goals: X+Y#>5, sq(X,Y,20).*

Es 5: Conversioni

- *Un programma di pianificazione di orario utilizza una rappresentazione del tempo per settimane*

- *Si vogliono scrivere vincoli con rappresentazione in Giorno-Ora; è necessario un vincolo che converte fra una rappresentazione e l'altra*

  time_conv(Giorno,Ora,T)

  - *Giorno:: 0..6, Ora:: 0..23, T :: 0..167*
- *Si scriva il vincolo time_conv con propagazione GAC*
  - *(nota che imporre il vincolo T #= Giorno*24+Ore ottiene GBC)*

Es 5: Soluzione

- *Possiamo usare Propia:*

  time_conv(Giorno,Ora,T) :- time_conv_pred(Giorno,Ora,T) infers fd.
- *A questo punto, dobbiamo implementare il predicato nondeterministico time_conv_pred che fornisce tutte le combinazioni valide della tripla <Giorno,Ora,T>*

Es 5

```
time_conv_pred(Ora,Giorno,T):-
   Ora::0..23, Giorno::0..6,
    Giorno*24+Ora #= T,
    labeling([Ora,Giorno]).
```


Es 5.1

- *Con tale rappresentazione dei domini, si impongano vincoli che esprimono il requisito:*

  "Il docente X vuole che le sue lezioni vengano tenute in giorni diversi ma consecutivi"

- *predicato:*

  giorni_consecutivi(ListaStartTimes)

Es 5.1: Soluzione

- *Se il docente deve tenere N lezioni in N giorni diversi e consecutivi, significa che la differenza fra il primo giorno e l'ultimo è N-1*

```
giorni_consecutivi(StartTimes):-
    length(StartTimes,Num),
    converti_giorni(StartTimes,Giorni),
    maxlist(Giorni,MaxDay),      minlist(Giorni,MinDay),
    MinDay + Num -1 #= MaxDay,
    alldifferent(Giorni).
converti_giorni([],[]).
converti_giorni([Time|Times],[Giorno|Giorni]):-
    time_conv(Giorno,_,Time),
    converti_giorni(Times,Giorni).
```


Es 5.1: difetti

- *In questo modo il vincolo time_conv dipende anche da una variabile che rappresenta l'ora del giorno: finché non viene istanziata, il vincolo non esce dal Constraint Store*

```
[eclipse]: T:: 25..30, time_conv(Ora,Giorno,T).
    Yes
    Ora = Ora{[1..6]}
    Giorno = 1
    T = T{[25..30]}
Delayed goals:

    time_conv_pred(Giorno,Ora,T) infers fd
```

Es 5.2

- *Si scriva un vincolo conv_day(Giorno,T) che collega il giorno e tempo settimanale usando le sospensioni*

Soluzione

```
conv_day(Giorno,T):-
    dom(T,TVal), % Estrae la lista di elementi nel dominio di una variabile
    giorni_consistenti(TVal,GDom),
    Giorno :: GDom,
    dom(Giorno,GDomNew),
    cancella_ore_inconsistenti(GDomNew,T),
    (var(Giorno), var(T)
    ->    suspend(conv_day(Giorno,T),4,[Giorno,T]->fd:any)
    ; true).
```

Altri predicati

```
giorni_consistenti([],[]).
giorni_consistenti([X|TVal],[Y|GDom]):-
      Y is X // 24,
    giorni_consistenti(TVal,GDom).
cancella_ore_inconsistenti(GDomNew,T):-
    cancella_ore_inconsistenti(GDomNew,T,0).
cancella_ore_inconsistenti(GDomNew,T,6).
cancella_ore_inconsistenti(GDomNew,T,I):-
    I < 6,
    (memberchk(I,GDomNew)
      -> true
      ;    Start is I*24, End is (I+1)*24-1,
           rimuovi_intervallo(T,Start,End)),
    I1 is I+1,
    cancella_ore_inconsistenti(GDomNew,T,I1).
```

```
rimuovi_intervallo(T,S,E):-
    S>E.
rimuovi_intervallo(T,Start,End):-
    S =< E,
    dvar_remove_element(T,J),
    S1 is Start+1,
    rimuovi_intervallo(T,S1,End).
```


Es 5.3

- *Si scriva un predicato che impone che un docente abbia N giorni liberi*

Es 5.3: Modello

- *Creo una lista ListaOccorrenze di 7 variabili*
  - *il primo elemento dice quante delle var StartTime sono di lunedì*
  - *il secondo quante sono di martedì*
  - *…*
- *poi impongo che nella ListaOccorrenze ci siano almeno N elementi a zero*

Es 5.3: Codice ECLiPSe

```
free_days(StartTimes,N):-
    converti_giorni(StartTimes,Giorni),
% Calcolo quante volte occorre il lun, il mar, ecc.
    calcola_occorrenze(Giorni,ListaOccorrenze),
    occurrences(0,ListaOccorrenze,N).

calcola_occorrenze(Giorni,ListaOccorrenze):-
    calcola_occorrenze(Giorni,ListaOccorrenze,0).
calcola_occorrenze([],[],6):-!.
calcola_occorrenze(Giorni,[Occ|ListaOccorrenze],I):-
      occurrences(I,Giorni,Occ), I1 is I+1,
      calcola_occorrenze(Giorni,ListaOccorrenze,I1).
```

Case

- *In una strada vi sono 5 case dipinte in 5 colori differenti. In ogni casa vive una persona di differente nazionalità. Ognuno dei padroni di casa beve una differente bevanda, fuma una differente marca di sigarette e tiene un animaletto differente.*
- *L'inglese abita nella casa rossa.*
- *Lo Svedese ha un cane*
- *Il Danese beve il The,*
- *La casa Verde è subito prima di quella Bianca,*
- *L'inquilino della casa verde beve caffè*
- *Chi fuma Pall Mall ha un uccellino*
- *Chi abita nella casa gialla fuma Dunhill*
- *Nella terza casa si beve latte*
- *Il Norvegese abita nella prima casa*
- *il vicino di chi fuma Blends ha un gatto*
- *Il vicino di chi fuma Dunhill ha un cavallo*
- *Chi fuma BlueMaster beve Birra*
- *Il Tedesco fuma le Prince,*
- *Il Norvegese ha un vicino con la casa blu*
- *Il vicino di chi fuma le Blends beve Acqua*

Soluzione

```
case(Colori,Nazione,Bevanda,Sigarette,Animale) :-
    Colori = [Rosso,Verde,Bianco,Giallo,Blu],              dominio(Colori),
    Nazionalita=[Inglese,Svedese,Danese,Norvegese,Tedesco],dominio(Nazione),
    Bevanda = [The,Caffe,Latte,Birra,Acqua],               dominio(Bevanda),
    Sigarette=[Dunhill,PallMall,Blends,BlueMaster,Prince], dominio(Sigarette),
    Animale = [Cane,Uccellino,Gatto,Cavallo,Pesce],        dominio(Animale),
    Inglese=Rosso,   Svedese=Cane,          Danese=The,     Verde #= Bianco-1,
    Verde=Caffe,     PallMall=Uccellino,    Giallo=Dunhill, Latte=3,
    Norvegese=1,     vicino(Blends,Gatto), vicino(Cavallo,Dunhill),
    BlueMaster=Birra, Tedesco=Prince,      vicino(Norvegese,Blu),
    vicino(Blends,Acqua),
    append_lists([Colori,Nazionalita,Bevanda,Sigarette,Animale],Sol),
    labeling(Sol).

dominio(L):- L :: 1..5, alldifferent(L).
vicino(A,B) :- A-B #= C, C::[-1,1].
% append_lists da implementare come esercizio
```

Vini e Formaggi

Cinque persone amano accompagnare il loro formaggio preferito con un buon vino rosso d'annata. In base ai dati che vi vengono forniti, cercate di stabilire, per ciascun nominativo, il tipo di formaggio preferito, con quale vino viene gustato e l'annata del vino

- *A Francesca piace il Gruviera, ma non beve Barolo, che è il vino preferito da Anna.*
- *Il vino dell'88 accompagna il formaggio Grana. Enrico ha scelto un vino del '95 per gustare il Pecorino*
- *Il Lambrusco è dell'annata '90. Il Chianti del '94 non viene bevuto da Mario che mangia Fontina*
- *Alessandra beve il Barbera, ma non per accompagnare il Brie*
- *Il Merlot non è del '91*

In libreria

Cinque impiegati di una grossa azienda con sede vicino alla libreria hanno approfittato dei loro momenti liberi per acquistare un certo numero di libri. in base ai dati forniti, cercate di stabilire per ciascuno dei 5 impiegati il numero di volumi comprati, in quale giorno della settimana e in quale momento della giornata.

- *Nicoletta ha acquistato più volumi di chi si è recato in libreria dopo il lavoro.*
- *Chi ha fatto i suoi acquisti, non di lunedì, durante la pausa caffè ha comperato un libro in meno di chi è andato in libreria martedì, ma in un giorno della settimana precedente rispetto a chi ha comperato i libri durante la pausa pomeridiana*
- *Tommaso ha acquistato un libro in meno di chi è andato in libreria prima del lavoro e che ha acquistato i libri due giorni dopo chi ha acquistato 8 volumi*
- *Valentina ha comperato i libri il giorno dopo rispetto a chi ha approfittato della pausa per il pranzo, che ha comperato due libri in pi\`u di chi è andato in libreria venerdì*
- *Gianni ha acquistato piu` libri di chi si è recato in libreria mercoledì.*
- *Impiegati: Gianni, Nicoletta, Roberto, Tommaso, Valentina*
- *Numero Libri: 3, 4, 6, 7, 8*
- *Giorni: Lun, mar, mer, ven, sab*
- *Mementi: dopo il lavoro, pausa caffè, pausa pomeridiana, pausa pranzo, prima del lavoro*