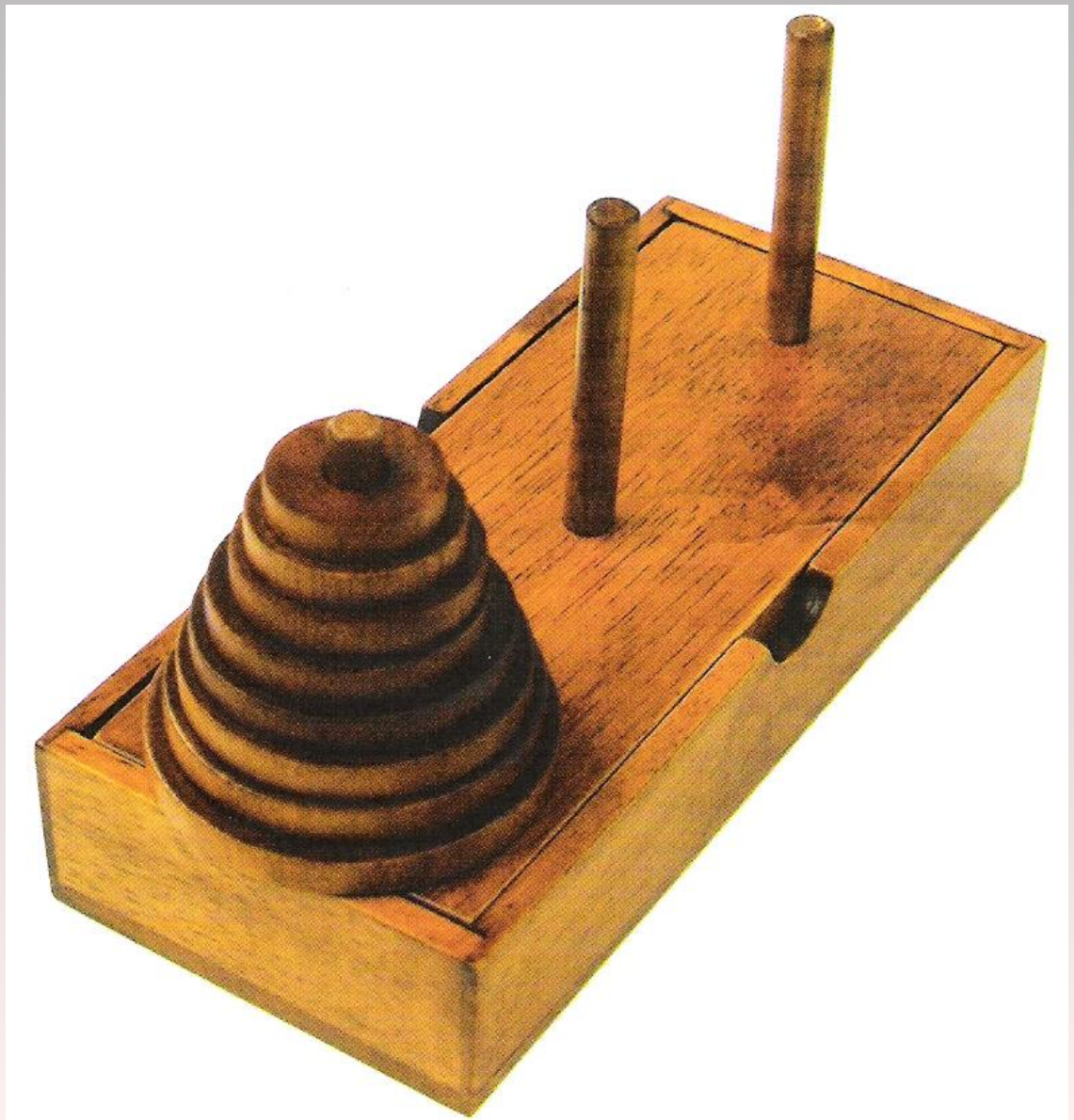
La programmazione logica come metodologia per la codifica e la risoluzione di rompicapi

Agostino Dovier

Dipartimento di Matematica e Informatica
Università di Udine

11 Dicembre 2015

La torre di Hanoi



La torre di Hanoi

Rappresentazione con `inpeg(tempo, disco, peg)` e `on(tempo,disco sopra, disco sotto/floor)`.
Mentre `top(tempo, peg, disco)` viene definito.

La torre di Hanoi

Rappresentazione con `inpeg(tempo, disco, peg)` e `on(tempo,disco sopra, disco sotto/floor).`
Mentre `top(tempo, peg, disco)` viene definito.

```
peg(1..3).
disk(1..n).
tempo(0..t).

% Situazione iniziale
inpeg(0,D,1)   :- disk(D).
on(0,D,D+1)    :- disk(D),disk(D+1).
on(0,n,floor).

% Goal (dico che li voglio tutti nel 2)
goal(D) :- inpeg(t,D,2), disk(D).
:- disk(D), not goal(D).
```

La torre di Hanoi

Azione: `move(tempo,pioloX, pioloY)`

```
% C'e' uno ed uno solo movimento al tempo T
1{ move(T,X,Y): peg(X), peg(Y), X != Y} 1 :-
          tempo(T), T < t.

% Vincolo di disco piccolo sopra
:- on(T,A,B), tempo(T), disk(A), disk(B), A >= B.

% Per fare la mossa il piolo deve contenere un disco
:- move(T,A,B), vuoto(T,A), tempo(T), peg(A), peg(B).
```

La torre di Hanoi

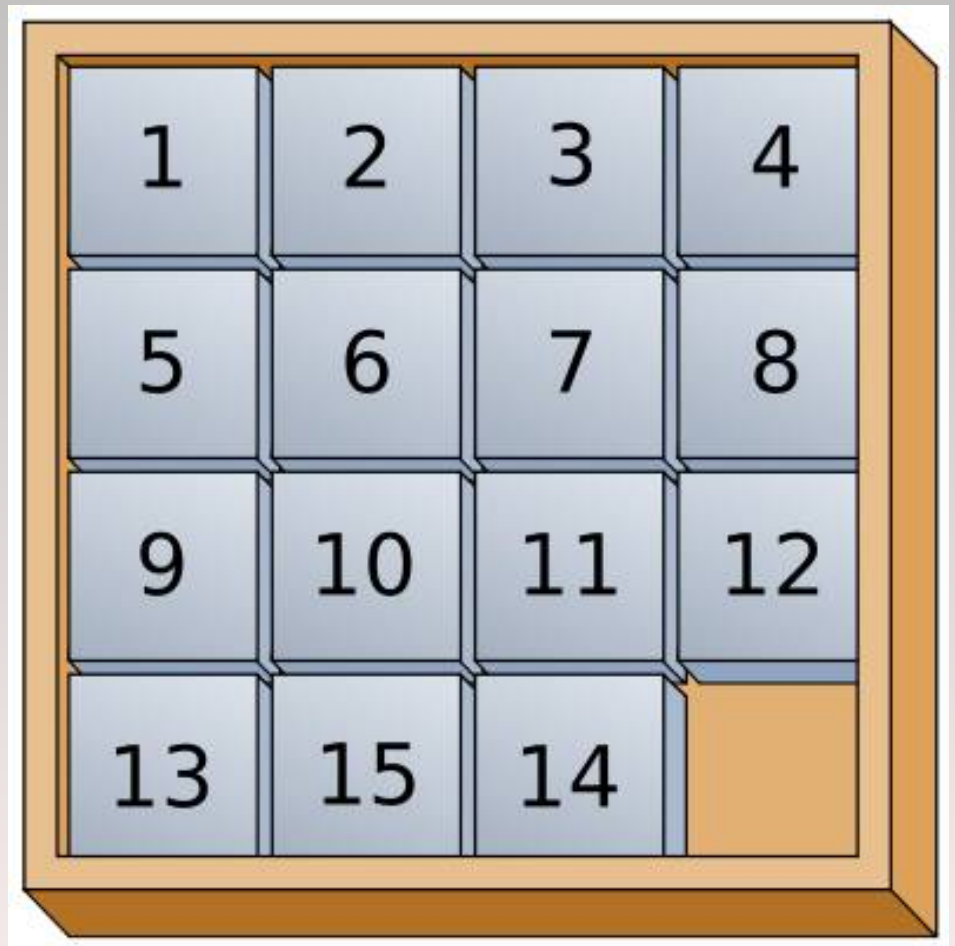
Predicati ausiliari:

```
coperto(T,D2)  :- on(T,D1,D2),
            tempo(T),  disk(D1), disk(D2).

top(T,A,D)     :- inpeg(T,D,A), not coperto(T,D),
            tempo(T), disk(D), peg(A).

nonvuoto(T,A)  :- inpeg(T,D,A),
            tempo(T), disk(D), peg(A).

vuoto(T,A)     :- not nonvuoto(T,A),
            tempo(T),  peg(A).

top(T,A,floor) :- vuoto(T,A),
            tempo(T), peg(A).

moved(T,D)     :- top(T,A,D), move(T,A,B),
            tempo(T), disk(D), peg(A), peg(B).
```

La torre di Hanoi

```
% 1. cambia on del moved
on(T+1,D1,D2) :-
        top(T,A,D1), top(T,B,D2), move(T,A,B),
        tempo(T), tempo(T+1), peg(A), peg(B), disk(D1), disk(D2).
on(T+1,D,floor) :-
        top(T,A,D), move(T,A,B), vuoto(T,B),
        tempo(T), tempo(T+1), peg(A), peg(B), disk(D).
% 2. cambia inpeg del moved
inpeg(T+1,D,B) :-
        top(T,A,D), move(T,A,B),
        tempo(T), tempo(T+1), peg(A), peg(B),disk(D).
% Inerzia sul resto
on(T+1,D,floor) :-
        on(T,D,floor), not moved(T,D),
        tempo(T), tempo(T+1), disk(D).
on(T+1,D1,D2) :-
        on(T,D1,D2), not moved(T,D1),
        tempo(T), tempo(T+1), disk(D1), disk(D2).
inpeg(T+1,D,A) :-
        inpeg(T,D,A), not moved(T,D),
        tempo(T), tempo(T+1), peg(A), disk(D).
```

Sam Lloyd's puzzle



Sam Lloyd's puzzle

```
tempo(0..t).
val(0..8).
range(0..2).

% 0     1     2     %  X=0
% 3     4     5     %  X=1
% 6     7     8     %  X=2
%%%%%%%
% 0     1     2   <- Y
%%%%%%
%% Lo "0" e' la cella vuota (buco)
%% Definisco il predicato
%% cell(tempo T, riga X, colonna Y, valore V)

%%% Input (esempio)
cell(0,0,0,1). cell(0,0,1,2). cell(0,0,2,5).
cell(0,1,0,3). cell(0,1,1,4). cell(0,1,2,8).
cell(0,2,0,6). cell(0,2,1,0). cell(0,2,2,7).
```

Sam Lloyd's puzzle

```
%%% La mossa sposta il buco!
1{ move(T,up); move(T,down); move(T,left); move(T,right)}1 :-
        tempo(T), tempo(T+1).

hole(T,X,Y) :- cell(T,X,Y,0), tempo(T), range(X), range(Y).

% Mosse proibite
:- tempo(T), move(T,up),    range(Y), hole(T,0,Y).
:- tempo(T), move(T,down),  range(Y), hole(T,2,Y).
:- tempo(T), move(T,left),  range(X), hole(T,X,0).
:- tempo(T), move(T,right), range(Y), hole(T,X,2).

% Posizione in cui andra' il buco
moved(T,X-1,Y) :- hole(T,X,Y), move(T,up),    tempo(T),range(X),range(Y).
moved(T,X+1,Y) :- hole(T,X,Y), move(T,down),  tempo(T),range(X),range(Y).
moved(T,X,Y-1) :- hole(T,X,Y), move(T,left),  tempo(T),range(X),range(Y).
moved(T,X,Y+1) :- hole(T,X,Y), move(T,right), tempo(T),range(X),range(Y).
```

Sam Lloyd's puzzle

```
% Nuovi contenuti celle
cell(T+1,X,Y,0)      :-
     moved(T,X,Y), tempo(T), tempo(T+1), range(X), range(Y).
cell(T+1,X1,Y1,V)    :-
    hole(T,X1,Y1), moved(T,X2,Y2), cell(T,X2,Y2,V),
    tempo(T), tempo(T+1), val(V),
    range(X1), range(X2), range(Y1), range(Y2).

% Inerzia

affected(T,X,Y) :- hole(T,X,Y),  tempo(T),  range(X), range(Y).
affected(T,X,Y) :- moved(T,X,Y), tempo(T),  range(X), range(Y).

cell(T+1,X,Y,V) :-  cell(T,X,Y,V),
       not affected(T,X,Y),
       tempo(T), tempo(T+1), range(X), range(Y), val(V).
```

Hamming Codes

Hamming Codes

```
word(1..k).
position(1..n).
dmin(d).

%% Valori
val(0).
val(1).
```

Hamming Codes

Vogliamo definite il predicato `tupla(indice,posizione,valore)`

Hamming Codes

Vogliamo definite il predicato `tupla(indice,posizione,valore)`

```
1{ tupla(I,J,V) : val(V) } 1 :- word(I), position(J).

distanza(I1,I2,S) :-
   word(I1), word(I2), I1 < I2,
   S = #count{ J : tupla(I1,J,V1), tupla(I2,J,V2), position(J), val(V1), v

badcode :-
   word(I1), word(I2), I1 < I2, dmin(d),
   distanza(I1,I2,S),
   S < d.
:- badcode.
```

Hamming Codes

Symmetry breaking

```
%%% Zero is in the code
tupla(1,J,0) :- position(J).

%%% Tuples are lexico ordered
ordered(I1,I2) :-
    word(I1), word(I2), I1 < I2,
    ordered(I1,I2,1).
ordered(I1,I2,J) :-
    position(J),
    word(I1), word(I2), I1 < I2,
    val(V1), val(V2),
    tupla(I1,J,V1), tupla(I2,J,V2),
    V1 < V2.
ordered(I1,I2,J) :-
    position(J), position(J+1),
    word(I1), word(I2), I1 < I2,
    val(V),
    tupla(I1,J,V), tupla(I2,J,V),
    ordered(I1,I2,J+1).
:-    word(I1), word(I2), I1 < I2, not ordered(I1,I2).
```

Hamming Codes

Output

```
% Completo con x i valori senza senso se n < 10
extrapos(n+1..10).
tupla(I,J,x) :- word(I), extrapos(J).

mostra(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10) :-
    word(I),
    tupla(I,1,X1),
    tupla(I,2,X2),
    tupla(I,3,X3),
    tupla(I,4,X4),
    tupla(I,5,X5),
    tupla(I,6,X6),
    tupla(I,7,X7),
    tupla(I,8,X8),
    tupla(I,9,X9),
    tupla(I,10,X10).
#show mostra/10.
```

Sudoku

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

Sudoku

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

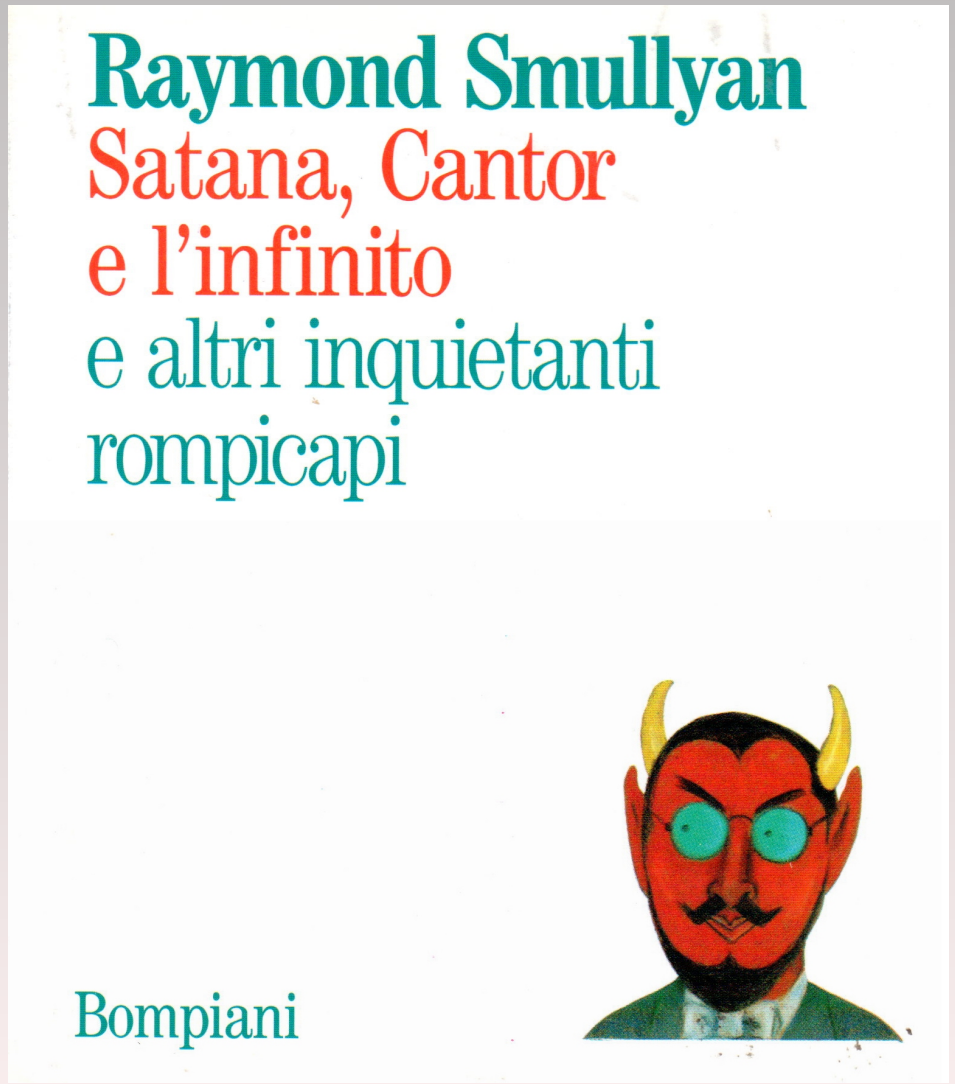
Vogliamo definite il predicato `x(riga,colonna,valore)`

```
x(1,3,1).
x(2,3,2). x(2,5,3). x(2,9,4).
x(3,4,5). x(3,7,6). x(3,9,7).
x(4,1,5). x(4,4,1). x(4,5,2).
x(5,2,7). x(5,8,2).
x(6,5,7). x(6,6,8). x(6,9,9).
x(7,1,8). x(7,3,7). x(7,6,9).
x(8,1,4). x(8,5,6). x(8,7,3).
x(9,7,5).
```

Sudoku

```
coord(1..9).
val(1..9).

% Assegno una coppia (X,Y) ad un sottoquadrato - e viceversa
square(I,X,Y) :-
   coord(X;Y;I),
   I == (X-1) / 3 + 3*((Y-1) / 3) + 1.

% Per ogni cella si assegna esattamente un valore
1 { x(X,Y,N) : val(N) } 1   :- coord(X;Y).

% Ogni valore viene usato esattamente una volta in una colonna
1 { x(X,Y,N) : coord(X) } 1 :- coord(Y), val(N).

% Ogni valore viene usato esattamente una volta in una riga
1 { x(X,Y,N) : coord(Y) } 1 :- coord(X), val(N).

% Ogni valore viene usato esattamente una volta in un sottoquadrato
1 { x(X,Y,N) : square(I,X,Y) } 1  :- val(N), coord(I).
```

Cavalieri e fanti



Cavalieri e fanti

Con una fitta di apprensione come non aveva mai provato prima, un antropologo di nome Abercrombie mise piede sull'isola dei Cavalieri e dei Fanti. Sapeva che quest'isola era popolata da gente quanto mai bizzarra: i cavalieri, che facevano solo affermazioni vere, e i fanti, che facevano solo affermazioni false. "Come farò a sapere qualcosa di quest'isola se non riesco a distinguere chi mente e chi dice la verità?" si chiedeva Abercrombie.

Abercrombie si rendeva conto che, prima di poter scoprire alcunché, avrebbe dovuto farsi un amico, qualcuno di cui potesse essere certo che gli avrebbe sempre detto la verità. Così, quando si imbatté nel primo gruppo di indigeni — tre persone, i cui nomi erano presumibilmente Arturo, Bernardo e Carlo —, Abercrombie pensò tra sé: "Ecco una buona occasione per trovarmi un cavaliere." Anzitutto Abercrombie chiese ad Arturo: "Bernardo e Carlo sono entrambi cavalieri?" Arturo rispose: "Sì." Allora Abercrombie chiese: "Bernardo è un cavaliere?" Con sua grande sorpresa, Arturo rispose: "No."

Carlo è un cavaliere o un fante?

Cavalieri e fanti

```
%%% ABITANTI
indigeno(a; b; c).

%%% AFFERMAZIONI
dice(a, yes, e(cavaliere(b),cavaliere(c))).
dice(a, no, cavaliere(b) ).

%%% Ognuno e' fante o cavaliere
1{fante(X);cavaliere(X)}1 :- indigeno(X).
```

Cavalieri e fanti

Regole per stabilire conseguenze:

```
%%% PARLA IL CAVALIERE:

fante(Y) :-
     dice(X,no,cavaliere(Y)),  cavaliere(X), indigeno(X), indigeno(Y).
cavaliere(Y) :-
     dice(X,yes,e(cavaliere(Y),cavaliere(Z))),
     cavaliere(X), indigeno(X), indigeno(Y), indigeno(Z).
cavaliere(Z) :-
     dice(X,yes,e(cavaliere(Y),cavaliere(Z))),
     cavaliere(X), indigeno(X), indigeno(Y), indigeno(Z).

%%% PARLA IL FANTE:

cavaliere(Y) :-
     dice(X, no, cavaliere(Y)),  fante(X), indigeno(X), indigeno(Y).
:-   dice(X,yes,e(cavaliere(Y),cavaliere(Z))),  fante(X),
     indigeno(X), indigeno(Y), indigeno(Z),
     cavaliere(Y), cavaliere(Z).
```

Torneo di calcetto

Calcetto serale

Torneo all'italiana

```
squadra(a;b;c;d; e;f;g;h).
giorno(1..7).   %% 1 lunedi', ..., 7 domenica
feriale(1..5).  %%
settimana(1..s).
orario(17;19;21).

slot(G,S,O)     :- giorno(G),  settimana(S), orario(O).
squadreord(X,Y) :- squadra(X), squadra(Y), X < Y.
```

Calcetto serale

```
testaserie(a;b;c;d).
testeserie(A,B) :- testaserie(A),testaserie(B), A < B.

% Per ogni coppia di squadre, si incontrano una ed una volta sola
1 { gioca(X,Y,G,S,O): slot(G,S,O) } 1 :- squadreord(X,Y).
% Per ogni orario, c'e' al piu' una partita
0 { gioca(X,Y,G,S,O): squadreord(X,Y)} 1 :- slot(G,S,O).
```

Calcetto serale

Predicati ausiliari:

```
oggi(X,S,X,S)              :- giorno(X),settimana(S).
domani(X,S,X+1,S)          :- giorno(X),giorno(X+1),settimana(S).
domani(7,S,1,S+1)          :- settimana(S),settimana(S+1).
dopodomani(X,S,X+2,S)      :- giorno(X),giorno(X+2),settimana(S).
dopodomani(7,S,2,S+1)      :- settimana(S),settimana(S+1).
dopodomani(6,S,1,S+1)      :- settimana(S),settimana(S+1).
good(X1,S1,X2,S2) :-
      giorno(X1),giorno(X2),
      settimana(S1),settimana(S2),
      not oggi(X1,S1,X2,S2),
      not domani(X1,S1,X2,S2),
      not domani(X2,S2,X1,S1),
      not dopodomani(X1,S1,X2,S2),
      not dopodomani(X2,S2,X1,S1).
```

Calcetto serale

Vincoli:

```
% Almeno tre giorni tra una partita e l'altra

:- squadreord(X,Y), squadreord(X,Z), Y < Z,
   slot(G1,S1,O1), slot(G2,S2,O2),
   gioca(X,Y,G1,S1,O1),gioca(X,Z,G2,S2,O2),
   not good(G1,S1,G2,S2).
:- squadreord(X,Y), squadreord(Z,X), Y != Z,
   slot(G1,S1,O1), slot(G2,S2,O2),
   gioca(X,Y,G1,S1,O1),gioca(Z,X,G2,S2,O2),
   not good(G1,S1,G2,S2).
:- squadreord(Y,X), squadreord(Z,X), Y < Z,
   slot(G1,S1,O1), slot(G2,S2,O2),
   gioca(Y,X,G1,S1,O1),gioca(Z,X,G2,S2,O2),
   not good(G1,S1,G2,S2).
```

Calcetto serale

Vincoli:

```
%%%% Di martedi' e mercoledi' non si gioca alle 21 (c'e' la champions)
:- squadreord(X,Y), settimana(S), gioca(X,Y,2,S,21).
:- squadreord(X,Y), settimana(S), gioca(X,Y,3,S,21).
%%%% Di sabato e domenica non si gioca alle 21 (c'e' l'anticipo ed il post
:- squadreord(X,Y), settimana(S), gioca(X,Y,6,S,21).
:- squadreord(X,Y), settimana(S), gioca(X,Y,7,S,21).

%%%% Le testadiserie non giocano tra loro nelle prima settimana
:- testeserie(X,Y), slot(G,S,O), gioca(X,Y,G,S,O), S < 2.
%%%% Non ci sono due sfide tra teste di serie nello stesso giorno
:- testeserie(X1,X2), testeserie(X3,X4),
   slot(G,S,O1), slot(G,S,O2), O1 < O2,
   gioca(X1,X2,G,S,O1), gioca(X3,X4,G,S,O2).

%%% Non ci sono giorni settimanali buchi (sabato o domenica ok)
slotoccupato(G,S,O) :-
     slot(G,S,O),squadreord(X,Y),gioca(X,Y,G,S,O).
giorno_occupato(G,S) :-
     slot(G,S,O), slotoccupato(G,S,O).
:- feriale(G), settimana(S), not giorno_occupato(G,S).
```

Calcetto serale

La direttiva minimize

Vincoli:

```
giorno_singolo(G,S) :-
   giorno(G),settimana(S), orario(O1), orario(O2), orario(O3),
   O1 != O2, O1 != O3, O2 < O3,
   slotoccupato(G,S,O1),
   not slotoccupato(G,S,O2),
   not slotoccupato(G,S,O3).

%%% Se singolo si gioca alle 19
:- giorno(G), settimana(S), giorno_singolo(G,S), slotoccupato(G,S,17).
:- giorno(G), settimana(S), giorno_singolo(G,S), slotoccupato(G,S,21).

%%% Se non singolo non deve avere buco
:- giorno(G), settimana(S), not giorno_singolo(G,S),
   slotoccupato(G,S,17), not slotoccupato(G,S,19), slotoccupato(G,S,21).

%%% MINIMIZZIAMO I GIORNI CON SINGOLO MATCH
contagiornisingoli(N) :- N = #count{G,S : giorno_singolo(G,S)}.
#minimize {N:contagiornisingoli(N)}.
```

Calcetto serale

Vincoli “speciali”:

```
%%% Richieste speciali:
%% La squadra "a" non gioca di sabato
%% (alcuni giocatori giocano a basket)
:- settimana(S), orario(O), squadra(Y), gioca(a,Y,6,S,O).
:- settimana(S), orario(O), squadra(Y), gioca(Y,a,6,S,O).
%% La squadra "b" non gioca di lunedi'
%% (alcune fidanzate hanno il negozio chiuso)
:- settimana(S), orario(O), squadra(Y), gioca(b,Y,1,S,O).
:- settimana(S), orario(O), squadra(Y), gioca(Y,b,1,S,O).
%% Nella prima e ultima  giornata ci devono essere almeno 2 partite.
:-  orario(O1), orario(O2), O1 < O2,
    not slotoccupato(1,1,O1), not slotoccupato(1,1,O2).
:-  orario(O1), orario(O2), O1 < O2,
    not slotoccupato(7,s,O1), not slotoccupato(7,s,O2).
```

Calcetto serale

Vincoli “speciali”:

```
day(A,X1,X2,X3,X4,X5,X6) :-
    giorno(G),
    settimana(S),
    A = G + 7*(S-1),
    completamento(X1,X2,G,S,17),
    completamento(X3,X4,G,S,19),
    completamento(X5,X6,G,S,21).
completamento(X,Y, G,S,O) :-
    slot(G,S,O), squadreord(X,Y), gioca(X,Y,G,S,O).
completamento(nil,nil, G,S,O) :-
    slot(G,S,O), not slotoccupato(G,S,O).
```

Conclusioni

1. La torre di Hanoi
2. Sam Lloyd's puzzle
3. Hamming Codes
4. Sudoku
5. Cavalieri e fanti
6. Torneo di calcetto

Abbiamo visto due problemi di planning, due problemi combinatorici "statici", un problema di KR e infine un problema "reale" di scheduling.