/home/shawn/schoolwork/CS4216/PS3/n-hybrid.pl

```
:-lib(suspend).
queers(QueerStruct,Number) :-
   dim(QueerStruct,[Number]),
   constraints(QueerStruct,Number),
   search(QueerStruct).
constraints(QueerStruct,Number) :-
   ( for(I,1,Number),
      param(QueerStruct,Number)
      do
      QueerStruct[I] :: 1..Number,
      ( for(J,1,I-1),param(I,QueerStruct) do
         QueerStruct[I] $\= QueerStruct[J]
      ),
      (I > 1 ->
         QueerStruct[I] - QueerStruct[I-1] $> 2 or QueerStruct[I] - QueerStruct[I-1] $< -2;
         true
      ),
      (I > 2 ->
         QueerStruct[I] - QueerStruct[I-2] $> 1 or QueerStruct[I] - QueerStruct[I-2] $< -1;
         true
      )

   ).
search(QueerStruct) :-
   dim(QueerStruct,[N]),
   ( foreacharg(Col,QueerStruct),
      param(N)
      do
      select_val(1,N,Col)
   ).
select_val(1,N,Col) :-
   (
      fromto(
         fail,
         C,
         (C;(Col=I)),
         Q
      ),
         for(I,1,N),param(Col) do
            true
   ),Q.

```

/home/shawn/schoolwork/CS4216/PS3/tiling.pl

```
:-lib(ic).
solve(N,Blanks,Grid):-
   constraints(N,Blanks,TileCount,Grid1,VarList),
   search(TileCount,VarList),
   (
      foreacharg(R,Grid1),foreach(RL,Grid) do
      (
         foreacharg(V,R),foreach(V1,RL) do
            (V = 0 ->
               V1 = x;
               V1 = V
            )
      )
   ).

constraints(N,Missing,TileCount,Grid,VarList):-
   length(Missing,M),
   CellCount is (N*N)-M,
   T is CellCount/2,
   integer(T,TileCount),
   length(VarList,CellCount),
   dim(Grid,[N,N]),
   (
      multifor([I,J],1,N),param(Grid,Missing,N,VarList,TileCount),fromto(VarList,InVars,OutVars,[]),param(VarList) do
      (
         subscript(Grid,[I,J],C),
         Isuc is I+1,Ipre is I-1,Jsuc is J+1,Jpre is J-1,
         ((I>1,not(member(Ipre-J,Missing))) -> subscript(Grid,[Ipre,J],L);L is -1),
         ((I<N,not(member(Isuc-J,Missing))) -> subscript(Grid,[Isuc,J],R);R is -2),
         ((J>1,not(member(I-Jpre,Missing))) -> subscript(Grid,[I,Jpre],U);U is -3),
         ((J<N,not(member(I-Jsuc,Missing))) -> subscript(Grid,[I,Jsuc],D);D is -4),
         (member((I-J),Missing) ->
            (
               OutVars = InVars,
               C = x
            );(
               C :: 1..TileCount,
               InVars = [C|OutVars],
               C #= L or C #= R or C #= U or C #= D,
               diff_others(I,J,Missing,Grid)
            )
         ),
         alldifferent([L,R,U,D])
         %L #\= R,L $\= U,L $\= D, %all different
         %R #\= U,R $\= D,
         %U #\= D
      )
```

```
   ),
   distinct(TileCount,VarList).

diff_others(I,J,Missing,Grid):-
   dim(Grid,[N,N]),
   (
      multifor([X,Y],1,N),param(I,J,Missing,Grid) do
      (
         not(((X=:=I),(Y-J > -2,Y-J < 2));((Y=:=J),(X-I> -2,X-I < 2))) ->
            (
               not(member((X-Y),Missing))->
                  (subscript(Grid,[I,J],A),
                  subscript(Grid,[X,Y],B),
                  A $\= B);
                  true
            );true
      )
   ).

search(TileCount,VarList) :-
   search(VarList,0,first_fail,indomain,complete,[]).

%search(TileCount,VarList) :-
%   (
%     foreach(V,VarList),param(TileCount,VarList) do
%     (
%        not(ground(V)) ->
%           select_val(1,TileCount,V),
%           true
%     )
%   ).


select_val(1,N,Col) :-
   (
      fromto(fail,C,(C;(Col=I)),Q),for(I,1,N),param(Col) do true
   ),Q.

sus_member(E,L) :- sus_member(E,L,0).
sus_member(_,[],C):- C.
sus_member(E,[H|T],C):- sus_member(E,T,C or (E #= H)).

memberlist([],_).
memberlist([H|T],L) :- sus_member(H,L), memberlist(T,L).

sorted([]).
sorted([_]).
sorted([H1,H2|T]) :- H1 #< H2, sorted([H2|T]).

```

```
distinct(K,L) :-
   length(M,K),
   (for(I,1,K),foreach(A,M) do A=I),
   memberlist(M,L),
   memberlist(L,M).

%:-constraints(3,[2-2],G,V),
%   nl,nl,
%   write(G),
%   subscript(G,[2,1],1),nl,
%   write(G),
%   nl,nl.

```