Uso das Restrições de Prolog na resolução de puzzles Outside Sudoku*

João Carlos Fonseca Pina de Lemos[up201000660] & Luís Guilherme da Costa Castro Neves[up201306485]

Faculdade de Engenharia da Universidade do Porto - Rua Dr. Roberto Frias, s/n
4200-465 Porto PORTUGAL feup@fe.up.pt
https://sigarra.up.pt/feup/pt/web_page.inicial

Resumo Este trabalho apresenta-se com o objetivo principal de entender a abordagem necessária para a resolução de problemas de satisfação de restrições, de forma a aplicá-la em casos concretos e práticos como problemas de otimização ou decisão combinatória. Foi abordado o puzzle Outside Sudoku, uma variante do Sudoku original, onde se utilizou programação em lógica com restrições para a sua resolução.

Keywords: Prolog · SICStus · CLP(FD) · Outside Sudoku · FEUP · PLOG.

1 Introdução

Durante a cadeira de Programação e Lógica do 3º ano do Mestrado Integrado em Engenharia Informatíca e de Computadores da Faculdade de Engenharia da Universidade do Porto, foi-nos proposto o desenvolvimento de um projeto que resolvesse, de forma eficiente, o puzzle Outside Sudoku, uma variante do Sudoku normal.
Sendo este puzzle pouco conhecido e existindo apenas alguns exemplos do mesmo, a principal fonte de dados foi a página gmpuzzles.com conhecida como *The Art of Puzzles*, da qual retirámos todos os puzzles Outside Soduku existentes, que podem ser testados na nossa resolução.
A implementação da solução do puzzle é feita em SICStus Prolog, usando a biblioteca *clpfd* que expõe um conjunto de predicados para restrições de números inteiros.
Ao longo do artigo, são feitas várias referências ao código desenvolvido com o formato [<ficheiro>:L<linha>], que indica o ficheiro e a linha do ficheiro. O artigo está organizado primeiramente pela descrição do problema, onde é estudado e explicado o problema em questão. Seguidamente é apresentada a abordagem ao problema de satisfação de restrições, subdividindo-se em variáveis de decisão e restrições. De seguida, é apresentada a solução e, para finalizar, os resultados e conclusões.

* Afiliação FEUP-PLOG, Turma 3MIEIC01, Grupo Outside Sudoku_2

2 Descrição do Problema

O Puzzle em questão, (Outside Sudoku) é um problema similar ao Sudoku standard, com a diferença de não existirem quaisquer valores dentro da matriz do puzzle, no início da resolução. Ao invés disto, aparecem números nas arestas da matriz, números estes que terão que ser colocados nas primeiras 3 células adjacentes às celulas da matriz em questão. Usamos a Figura 1 como demonstração do problema, tendo esta figura sido usada como referência para a resolução do projeto. Esta mesma versão do puzzle pode ser testada correndo o predicado solve(1), aquando a compilação do projeto.

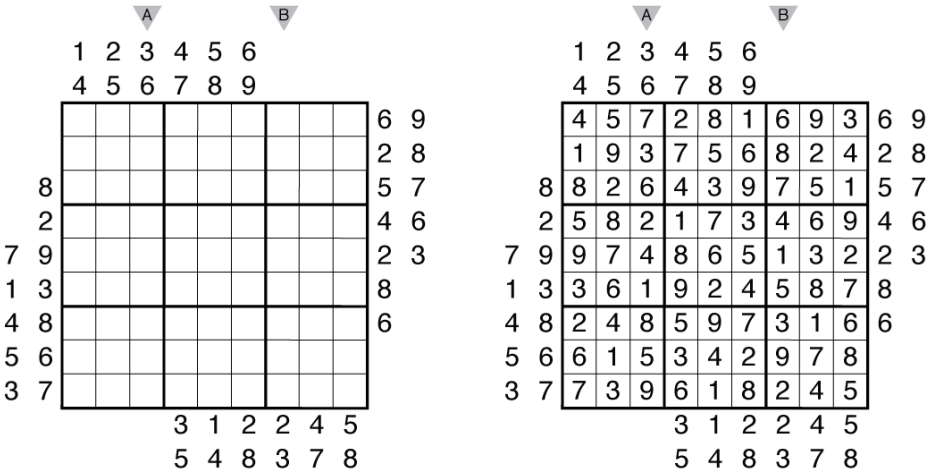

Figura 1. Outside Sudoku R&I - Exemplo base de Outside_Sudoku retirado do website gmpuzzles.com. Os números que aparecem nas arestas do puzzle têm que aparecer numa das 3 células adjacentes da matriz na resolução final.

3 Abordagem

Em seguida, é descrito a nossa resolução do puzzle Outside Sudoku, modelado como um PSR (Problema de Satisfação de Restrições).
Como primeiro teste para a resolução do problema, decidimos começar pela criação de predicados que resolvessem o problema de sudoku normal. Estes predicados ainda estão disponíveis no ficheiro final, assim como os puzzles de Sudoku de teste. O predicado para testar a nossa primeira resolução de sudoku é *solve_sudoku(P)* e pode ser consultado em [main.pl:L27].
A nossa resolução passa por correr o predicado *solve(P)* [main.pl:L15] no qual P irá buscar ao ficheiro *outside_sudoku_cases.pl* usando o predicado

get_out_sudoku_entries(P,Entries) um dos 5 puzzles implementados nos nossos testes e retirados da nossa fonte, estes testes vêm descritos sob forma de uma estrura brevemente descrita. Seguidamente, o programa mostra ao utilizador os inputs presentes na estrutura Entries para facilitar a verificação do estado inicial do puzzle, isto é feito através do predicado *display_outside_sudoku_entries(Entries)*. Estatísticas à parte, o programa segue com a resolução do puzzle através do predicado *outside_sudoku(Puzzle,Entries)*, resolução essa que é depois vizualizada com o predicado *display_sudoku(Puzzle)*.

3.1 Variáveis de Decisão

Como acabámos de apresentar o predicado de resolução do puzzle é Outside Sudoku *outside_sudoku(Puzzle,Entries)*.

Este predicado recebe a estrutura *Entries* e devolve ao predicado principal a matriz *Puzzle*. A estrutura Entries é definida como uma lista de quatro elementos e pode ser exemplificado como:

```
Puzzle = [Lista1,Lista2,Lista3,Lista4]
Lista = Ponto_de_entrada-Lista_de_entradas
Lista_de_entradas = X(de 1 a 9)-[Valores de entrada]
```

Assim sendo e usando os valores da Figura1 temos a seguinte estrutura:

```
%Outside Sudoku R&I
get_out_sudoku_entries(1,Puzzle):-
    Puzzle = [
        up-[1-[1,4],2-[2,5],3-[3,6],4-[4,7],
            5-[5,8],6-[6,9]],
        left-[3-[8],4-[2],5-[7,9],6-[1,3],
              7-[4,8],8-[5,6],9-[3,7]],
        right-[1-[6,9],2-[2,8],3-[5,7],
               4-[4,6],5-[2,3],6-[8],7-[6]],
        down-[4-[3,5],5-[1,4],6-[2,8],
              7-[2,3],8-[4,7],9-[5,8]]
    ].
```

A matriz *Puzzle* é depois definida como:

```
Puzzle=[[A1,B1,C1,D1,E1,F1,G1,H1,I1],
        [A2,B2,C2,D2,E2,F2,G2,H2,I2],
        [A3,B3,C3,D3,E3,F3,G3,H3,I3],
        [A4,B4,C4,D4,E4,F4,G4,H4,I4],
        [A5,B5,C5,D5,E5,F5,G5,H5,I5],
        [A6,B6,C6,D6,E6,F6,G6,H6,I6],
        [A7,B7,C7,D7,E7,F7,G7,H7,I7],
        [A8,B8,C8,D8,E8,F8,G8,H8,I8],
        [A9,B9,C9,D9,E9,F9,G9,H9,I9]],
```

Estes valores de A1 a I9 são preenchidos segundo os valores de entrada da estrutura e as restrições programadas. Estes valores são restringidos ao domínio de 1 a 9 através do código: ([*outside_ sudoku_ solver.pl:L13*])

```
Vars = [A1,B1,C1,D1,E1,F1,G1,H1,I1 ,
        A2,B2,C2,D2,E2,F2,G2,H2,I2 ,
        A3,B3,C3,D3,E3,F3,G3,H3,I3 ,
        A4,B4,C4,D4,E4,F4,G4,H4,I4 ,
        A5,B5,C5,D5,E5,F5,G5,H5,I5 ,
        A6,B6,C6,D6,E6,F6,G6,H6,I6 ,
        A7,B7,C7,D7,E7,F7,G7,H7,I7 ,
        A8,B8,C8,D8,E8,F8,G8,H8,I8 ,
        A9,B9,C9,D9,E9,F9,G9,H9,I9] ,
  domain(Vars,1,9)
```

3.2 Restrições

A solução do puzzle tem que obdecer a 4 restrições:

- **Outside:** Os valores que aparecem fora da matriz têm que aparecer numa das 3 células adjacentes na direção correspondente;
- **Linhas:** O mesmo valor não se pode repetir na mesma linha;
- **Colunas:** O mesmo valor não se pode repetir na mesma coluna;
- **Quadrado:** O mesmo valor não se pode repetir no mesmo quadrado de 9 elementos.

As restrições de **Outside** são feitas através do predicado *solve_ outside(Puzzle,Entries).* Este predicado percorre a estrutura até chegar à análize das coordenadas e dos valores de entrada, procedendo depois à restrição definida, usando a função *count(+Val,+List,+RelOp,?Count)* da biblioteca CLP(FD). Esta função passa o valor a entrar para o puzzle em Val, insere-o nas 3 células que lhe dizem respeito em List, usam a operação '#=' como RelOp e 'X in 1..9' para Count.

```
solve_outside(Puzzle,Entries):-
        Entries=[L|Ls],
        get_entries(Puzzle,L),
        solve_outside(Puzzle,Ls,2).
solve_outside(_,_,5).
solve_outside(Puzzle, Entries, Aux):-
        Entries=[L|Ls],
        get_entries(Puzzle,L),
        Aux2 is Aux+1,
        solve_outside(Puzzle,Ls,Aux2).
solve_outside(_,_,_).

get_entries(Puzzle,Entries):-
        Entries = Entry_point-List,
```

```
        %Entry_point is up, down, left or right
        %List is (coordenate - input list)
        solve_from_entry_point(Puzzle, Entry_point, List).

solve_from_entry_point(Puzzle, Entry_point, List):-
        List=[Index-Values|Ls],
        %Index is Coordenate
        %Values is input list
        length(Values, Length),
        get_value(Puzzle, Entry_point, Index, Values, Length, 1),
        solve_from_entry_point(Puzzle, Entry_point, Ls).
solve_from_entry_point(_,_,_).
```

Devido às diversas diferenças de entradas possíveis, foi criado o ficheiro *get_ value.pl* com todos os 9*4 casos possíveis para que o count funcionasse de forma perfeita na nossa execussão.
Estando tratadas as restrições de **Outside**, as restantes restrições são tratadas com um solver normal de sudoku desenvolvido anteriormente aos testes.

```
sudoku(Rows) :-
        maplist(all_distinct, Rows),
        transpose(Rows, Columns),
        maplist(all_distinct, Columns),
        Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
        blocks(As, Bs, Cs),
        blocks(Ds, Es, Fs),
        blocks(Gs, Hs, Is).

blocks([], [], []).
blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
        all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
        blocks(Ns1, Ns2, Ns3).
```

4 Visualização da Solução

Para visualização da Solução, foi criado o predicado *display_ sudoku(Puzzle)* no ficheiro *interface.pl*, onde estão todos os predicados que retornam informação para o utilizador.
Este predicado começa por desenhar a aresta de topo do puzzle (uma linha horizontal), desenhando seguidamente as 3 linhas seguintes com os 9 valores das células de cada uma, linhas verticais laterais e interiores nos casos respetivos seguida de outra linha horizontal e repetindo este processo mais duas vezes, para desenhar toda a matriz do Outside Sudoku. Através do predicado *get_ cell(X, Y, Board)* [interface.pl:L50], cada célula é desenhada com o valor correto.
Todo este código foi já desenvolvido para o estudo inicial de realização de um sudoku normal (que pode ser testado através de *solve_ sudoku(P)* com P de 1 a

7), onde desenhava o puzzle inicial colocando-se asteríscos nas células vazias e o puzzle final já terminado.

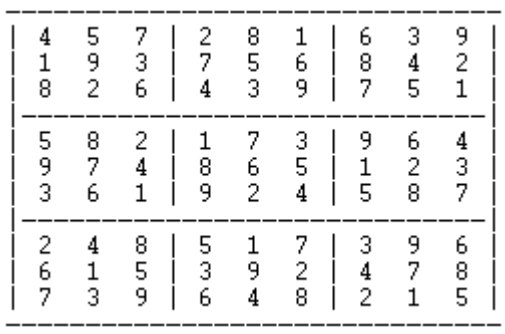

Figura 2. Vizualização da solução do puzzle Outside Sudoku da Figura 1

Para além da vizualização da solução final, foi também criado o predicado *display_ outside_ sudoku_ entries(Puzzle)* [interface.pl:L69] que apresenta, ates da resolução, os inputs que entram no puzzle pela estrutura. O resultado deste predicado pode ser visto na Figura 3.

5 Resultados

Para testar a nossa resolução, colocámos no ficheiro *outside_ sudoku_ cases.pl* os 5 casos existentes na nossa fonte pelo que os resultados de tempo de resolução e estatísticas são colocados na seguinte tabela:

Tabela 1. Resultados da realização dos puzzles existentes em gmpuzzles.com

Id	Puzzles	Time	Resumptions	Entailments	Prunings	Backtracks	Constraints Created
1	R&I	3.27s	5789209	1771404	3567399	87286	2214
2	#36	0.37s	596118	155219	312707	5626	2226
3	J. Bulten	0.26s	216479	64017	107813	1741	2618
4	#32	0.17s	4869	1180	2624	50	278
5	#34	0.26s	408679095	82813195	182844732	5884390	11519258

Apesar dos puzzles não terem um nível de dificuldade evidenciado no site, o puzzle de Id 4 conhecido como **Dr. Sudoku Prescribes #32 – Outside Sudoku** é tido como o exemplo com grau de dificuldade mais baixo dos existentes, algo que se consegue confirmar pelos dados obtidos

```
| ?- solve(1).

Input Values:
up variables:
     1 - 1,4;
     2 - 2,5;
     3 - 3,6;
     4 - 4,7;
     5 - 5,8;
     6 - 6,9;

left variables:
     3 - 8;
     4 - 2;
     5 - 7,9;
     6 - 1,3;
     7 - 4,8;
     8 - 5,6;
     9 - 3,7;

right variables
     1 - 6,9;
     2 - 2,8;
     3 - 5,7;
     4 - 4,6;
     5 - 2,3;
     6 - 8;
     7 - 6;

down variables:
     4 - 3,5;
     5 - 1,4;
     6 - 2,8;
     7 - 2,3;
     8 - 4,7;
     9 - 5,8;
```

Figura 3. Vizualização dos dados de entrada para a resolução apresentada na Figura 2

6 Conclusões e Trabalho Futuro

Este projeto mostrou-se essencial para uma melhor compreensão da técnica de programação por restrições, nomeadamente programação em Lógica com restrições de inteiros sobre SICStus Prolog, estudado na unidade curricular PLOG, pois foi possível, de uma forma pratica, estudar e testar conceitos, através da solução do puzzle Outside Sudoku. Foi-nos também possível confirmar a boa capacidade de resolver problemas de puzzles de lógica com a linguagem de programação Prolog.
A nossa solução tem duas desvantagens muitos próprias, a primeira é a necessidade de uma estrutura de entrada tão complexa que é propícia a erros, aquando a escrita à mão da mesma (algo que nos aconteceu, algumas vezes, dando origem a ciclos infinitos no programa devido à criação de puzzles sem solução); a segunda e a mais evidente foi a necessidade de criação de um ficheiro *get_value.pl* tão extenso, devido a não termos chegado a uma solução simples de selecionar a células da matriz respetiva para todos os casos necessários, sendo este o caso considerado mais necessário de alteração em desenvolvimento futuro.

Referências

1. Snyder, T.,Huang, W.: Mutant Sudoku. Puzzlewright , USA (2009)
2. Outside Sudoku R&I https://www.gmpuzzles.com/blog/sudoku-rules-and-info/outside-sudoku-rules-and-info/. Acedido a 23 de Dezembro de 2018

7 Anexo

7.1 main.pl

```
 :- use_module(library(clpfd)).
:- use_module(library(lists)).
:- use_module(library(aggregate)).
:- use_module(library(system)).


:-[sudoku_cases].
:-[interface].
:-[solver].
:-[outside_sudoku_cases].
:-[outside_sudoku_solver].
:-[stats].
:-[get_value].

solve(P):-
    get_out_sudoku_entries(P,Entries),nl,
    write('Input Values:'),nl,
```

```
    display_outside_sudoku_entries(Entries),
    reset_timer,
    outside_sudoku(Puzzle,Entries),
    display_sudoku(Puzzle),nl,
    print_time,
        fd_statistics.


%normal sudoku solver used for first testing
solve_sudoku(P):-
    get_sudoku(P, Puzzle),
    write('Problem: '),nl,
    display_sudoku(Puzzle),nl,
    reset_timer,
    sudoku_old(Puzzle),
    print_time,
    write('Solution: '),nl,
    display_sudoku(Puzzle),nl,
    print_time,
    fd_statistics.
```

7.2 interface.pl

```
 display_sudoku(Puzzle):-
    draw_line,
    draw_interior(0,Puzzle),
    draw_line.

draw_interior(3, Puzzle):-
    draw_line_int,
    Aux is 4,
    write('|'),
    write_cells(0, 3, Puzzle),
    draw_interior(Aux, Puzzle).

draw_interior(6, Puzzle):-
    draw_line_int,
    Aux is 7,
    write('|'),
    write_cells(0, 6, Puzzle),
    draw_interior(Aux, Puzzle).

draw_interior(9, _).

draw_interior(I, Puzzle):-
    Aux is I+1,
```

```
    write('|'),
    write_cells(0, I, Puzzle),
    draw_interior(Aux, Puzzle).
draw_interior(_,_).


write_cells(9, _, _):-
    write('|'), nl.

write_cells(2, Y, Puzzle):-
    get_cell(2, Y, Puzzle),
    write('|'),
    X2 is 3,
    write_cells(X2, Y, Puzzle).

write_cells(5, Y, Puzzle):-
    get_cell(5, Y, Puzzle),
    write('|'),
    X2 is 6,
    write_cells(X2, Y, Puzzle).

write_cells(X, Y, Puzzle):-
    get_cell(X, Y, Puzzle),
    X2 is X+1,
    write_cells(X2, Y, Puzzle).

get_cell(X, Y, Board) :-
        nth0(Y, Board, Column),
        nth0(X, Column, Cell),
    integer(Cell),
    write(' '),
    write(Cell),
    write(' ').

get_cell(X, Y, Board) :-
        nth0(Y, Board, Column),
        nth0(X, Column, _),
    write(' * ').

draw_line:-
    write('-----------------------------'),nl.

draw_line_int:-
    write('|---------------------------|'),nl.
```

```
display_outside_sudoku_entries(Puzzle):-
    Puzzle=[L|Ls],
    display_entries(L),nl,
    display_outside_sudoku_entries(Ls,2).

display_outside_sudoku_entries(_,5).
display_outside_sudoku_entries(Puzzle, Aux):-
    Puzzle=[L|Ls],
    display_entries(L),nl,
    Aux2 is Aux+1,
    display_outside_sudoku_entries(Ls,Aux2).

display_entries(L):-
    L=Entry_point-List,
    write(Entry_point), write(' variables:'),nl,
    display_list(List,1),nl.
display_entries(_).

display_list(List,Count):-
    List=[Index-Lx|Ls],
    length(Lx, Length),
    write('      '),
    write(Index),write(' - '), display_values(Lx,1,Length),nl,
    C is Count+1,
    display_list(Ls,C).
display_list([]).

display_values(Values,Max,Max):-
    Values = [A],
    write(A),
    write(';').
display_values(Values,X,Max):-
    X \= Max,
    Values = [A|B],
    write(A),
    write(','),
    NewX is X+1,
    display_values(B,NewX,Max).
```

7.3 stats.pl

```
reset_timer :- statistics(walltime,_).
print_time :-
        statistics(walltime,[_,T]),
        TS is ((T//10)*10)/1000,
        nl, write('Time: '), write(TS), write('s'), nl, nl.
```

7.4 outside_sudoku_solver.pl

```
:- use_module(library(clpfd)).

outside_sudoku(Puzzle,Entries):-
        Puzzle=[[A1,B1,C1,D1,E1,F1,G1,H1,I1],
        [A2,B2,C2,D2,E2,F2,G2,H2,I2],
        [A3,B3,C3,D3,E3,F3,G3,H3,I3],
        [A4,B4,C4,D4,E4,F4,G4,H4,I4],
        [A5,B5,C5,D5,E5,F5,G5,H5,I5],
        [A6,B6,C6,D6,E6,F6,G6,H6,I6],
        [A7,B7,C7,D7,E7,F7,G7,H7,I7],
        [A8,B8,C8,D8,E8,F8,G8,H8,I8],
        [A9,B9,C9,D9,E9,F9,G9,H9,I9]],
    Vars = [A1,B1,C1,D1,E1,F1,G1,H1,I1,
            A2,B2,C2,D2,E2,F2,G2,H2,I2,
            A3,B3,C3,D3,E3,F3,G3,H3,I3,
            A4,B4,C4,D4,E4,F4,G4,H4,I4,
            A5,B5,C5,D5,E5,F5,G5,H5,I5,
            A6,B6,C6,D6,E6,F6,G6,H6,I6,
            A7,B7,C7,D7,E7,F7,G7,H7,I7,
            A8,B8,C8,D8,E8,F8,G8,H8,I8,
            A9,B9,C9,D9,E9,F9,G9,H9,I9],
    domain(Vars,1,9),
    solve_outside(Puzzle,Entries),
    sudoku(Puzzle),
    labeling([],Vars).

solve_outside(Puzzle,Entries):-
        Entries=[L|Ls],
        get_entries(Puzzle,L),
        solve_outside(Puzzle,Ls,2).
solve_outside(_,_,5).
solve_outside(Puzzle, Entries, Aux):-
        Entries=[L|Ls],
        get_entries(Puzzle,L),
        Aux2 is Aux+1,
        solve_outside(Puzzle,Ls,Aux2).
solve_outside(_,_,_).

get_entries(Puzzle,Entries):-
        Entries = Entry_point-List,
        %Entry_point is up, down, left or right
        %List is (coordenate - input list)
        solve_from_entry_point(Puzzle, Entry_point, List).
```

```
solve_from_entry_point(Puzzle, Entry_point, List):-
        List=[Index-Values|Ls],
        %Index is Coordenate
        %Values is input list
        length(Values, Length),
        get_value(Puzzle, Entry_point, Index, Values, Length, 1),
        solve_from_entry_point(Puzzle, Entry_point, Ls).
solve_from_entry_point(_,_,_).
```

7.5 get_value.pl

```
get_value(_,_,_,_,Length,I):- I>Length.
get_value(Puzzle, up, 1, Values, Length, I):-
        I=<Length,
        Puzzle=[[A1|_],[A2|_],[A3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A1,A2,A3]),#=,X),
        get_value(Puzzle, up, 1, Ls, Length, NextI).
get_value(Puzzle, up, 2, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,B1|_],[_,B2|_],[_,B3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([B1,B2,B3]),#=,X),
        get_value(Puzzle, up, 2, Ls, Length, NextI).
get_value(Puzzle, up, 3, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,C1|_],[_,_,C2|_],[_,_,C3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([C1,C2,C3]),#=,X),
        get_value(Puzzle, up, 3, Ls, Length, NextI).
get_value(Puzzle, up, 4, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,_,D1|_],[_,_,_,D2|_],[_,_,_,D3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([D1,D2,D3]),#=,X),
        get_value(Puzzle, up, 4, Ls, Length, NextI).
get_value(Puzzle, up, 5, Values, Length, I):-
        I=<Length,
```

```
        Puzzle=[[_,_,_,_,E1|_],[_,_,_,_,E2|_],[_,_,_,_,E3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([E1,E2,E3]),#=,X),
        get_value(Puzzle, up, 5, Ls, Length, NextI).
get_value(Puzzle, up, 6, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,_,_,_,F1|_],[_,_,_,_,_,F2|_],[_,_,_,_,_,F3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([F1,F2,F3]),#=,X),
        get_value(Puzzle, up, 6, Ls, Length, NextI).
get_value(Puzzle, up, 7, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,_,_,_,_,G1|_],[_,_,_,_,_,_,G2|_],[_,_,_,_,_,_,G3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G1,G2,G3]),#=,X),
        get_value(Puzzle, up, 7, Ls, Length, NextI).
get_value(Puzzle, up, 8, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,_,_,_,_,_,H1|_],[_,_,_,_,_,_,_,H2|_],[_,_,_,_,_,_,_,H3|_]|_
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([H1,H2,H3]),#=,X),
        get_value(Puzzle, up, 8, Ls, Length, NextI).
get_value(Puzzle, up, 9, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,_,_,_,_,_,_,I1],[_,_,_,_,_,_,_,_,I2],[_,_,_,_,_,_,_,_,I3]|_
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([I1,I2,I3]),#=,X),
        get_value(Puzzle, up, 9, Ls, Length, NextI).
get_value(Puzzle, left, 1, Values, Length, I):-
        I=<Length,
        Puzzle=[[A1,B1,C1|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A1,B1,C1]),#=,X),
```

```
        get_value(Puzzle, left, 1, Ls, Length, NextI).
get_value(Puzzle, left, 2, Values, Length, I):-
        I=<Length,
        Puzzle=[_,[A2,B2,C2|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A2,B2,C2]),#=,X),
        get_value(Puzzle, left, 2, Ls, Length, NextI).
get_value(Puzzle, left, 3, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,[A3,B3,C3|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A3,B3,C3]),#=,X),
        get_value(Puzzle, left, 3, Ls, Length, NextI).
get_value(Puzzle, left, 4, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,[A4,B4,C4|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A4,B4,C4]),#=,X),
        get_value(Puzzle, left, 4, Ls, Length, NextI).
get_value(Puzzle, left, 5, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,[A5,B5,C5|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A5,B5,C5]),#=,X),
        get_value(Puzzle, left, 5, Ls, Length, NextI).
get_value(Puzzle, left, 6, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,[A6,B6,C6|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A6,B6,C6]),#=,X),
        get_value(Puzzle, left, 6, Ls, Length, NextI).
get_value(Puzzle, left, 7, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,[A7,B7,C7|_]|_],
        Values=[Num|Ls],
```

```
        NextI is I+1,
        X in 1..9,
        count(Num,([A7,B7,C7]),#=,X),
        get_value(Puzzle, left, 7, Ls, Length, NextI).
get_value(Puzzle, left, 8, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,[A8,B8,C8|_]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A8,B8,C8]),#=,X),
        get_value(Puzzle, left, 8, Ls, Length, NextI).
get_value(Puzzle, left, 9, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,_,[A9,B9,C9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A9,B9,C9]),#=,X),
        get_value(Puzzle, left, 9, Ls, Length, NextI).
get_value(Puzzle, right, 1, Values, Length, I):-
        I=<Length,
        Puzzle=[[_,_,_,_,_,_,G1,H1,I1]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G1,H1,I1]),#=,X),
        get_value(Puzzle, right, 1, Ls, Length, NextI).
get_value(Puzzle, right, 2, Values, Length, I):-
        I=<Length,
        Puzzle=[_,
                [_,_,_,_,_,_,G2,H2,I2]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G2,H2,I2]),#=,X),
        get_value(Puzzle, right, 2, Ls, Length, NextI).
get_value(Puzzle, right, 3, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,
                [_,_,_,_,_,_,G3,H3,I3]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G3,H3,I3]),#=,X),
```

```
        get_value(Puzzle, right, 3, Ls, Length, NextI).
get_value(Puzzle, right, 4, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,
                [_,_,_,_,_,_,G4,H4,I4]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G4,H4,I4]),#=,X),
        get_value(Puzzle, right, 4, Ls, Length, NextI).
get_value(Puzzle, right, 5, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,
                [_,_,_,_,_,_,G5,H5,I5]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G5,H5,I5]),#=,X),
        get_value(Puzzle, right, 5, Ls, Length, NextI).
get_value(Puzzle, right, 6, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,
                [_,_,_,_,_,_,G6,H6,I6]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G6,H6,I6]),#=,X),
        get_value(Puzzle, right, 6, Ls, Length, NextI).
get_value(Puzzle, right, 7, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,
                [_,_,_,_,_,_,G7,H7,I7]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G7,H7,I7]),#=,X),
        get_value(Puzzle, right, 7, Ls, Length, NextI).
get_value(Puzzle, right, 8, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,
                [_,_,_,_,_,_,G8,H8,I8]|_],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G8,H8,I8]),#=,X),
```

```
        get_value(Puzzle, right, 8, Ls, Length, NextI).
get_value(Puzzle, right, 9, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,_,_,
                [_,_,_,_,_,_,G9,H9,I9]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G9,H9,I9]),#=,X),
        get_value(Puzzle, right, 9, Ls, Length, NextI).
get_value(Puzzle, down, 1, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,
                [A7|_],[A8|_],[A9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([A7,A8,A9]),#=,X),
        get_value(Puzzle, down, 1, Ls, Length, NextI).
get_value(Puzzle, down, 2, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,
                [_,B7|_],[_,B8|_],[_,B9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([B7,B8,B9]),#=,X),
        get_value(Puzzle, down, 2, Ls, Length, NextI).
get_value(Puzzle, down, 3, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,
                [_,_,C7|_],[_,_,C8|_],[_,_,C9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([C7,C8,C9]),#=,X),
        get_value(Puzzle, down, 3, Ls, Length, NextI).
get_value(Puzzle, down, 4, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,_,
                [_,_,_,D7|_],[_,_,_,D8|_],[_,_,_,D9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([D7,D8,D9]),#=,X),
```

```
        get_value(Puzzle, down, 4, Ls, Length, NextI).
get_value(Puzzle, down, 5, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,
                [_,_,_,_,E7|_],[_,_,_,_,E8|_],[_,_,_,_,E9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([E7,E8,E9]),#=,X),
        get_value(Puzzle, down, 5, Ls, Length, NextI).
get_value(Puzzle, down, 6, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,
                [_,_,_,_,_,F7|_],[_,_,_,_,_,F8|_],[_,_,_,_,_,F9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([F7,F8,F9]),#=,X),
        get_value(Puzzle, down, 6, Ls, Length, NextI).
get_value(Puzzle, down, 7, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,
                [_,_,_,_,_,_,G7|_],[_,_,_,_,_,_,G8|_],[_,_,_,_,_,_,G9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([G7,G8,G9]),#=,X),
        get_value(Puzzle, down, 7, Ls, Length, NextI).
get_value(Puzzle, down, 8, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,
                [_,_,_,_,_,_,_,H7|_],[_,_,_,_,_,_,_,H8|_],[_,_,_,_,_,_,_,H9|_]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([H7,H8,H9]),#=,X),
        get_value(Puzzle, down, 8, Ls, Length, NextI).
get_value(Puzzle, down, 9, Values, Length, I):-
        I=<Length,
        Puzzle=[_,_,_,_,_,_,
                [_,_,_,_,_,_,_,_,I7],[_,_,_,_,_,_,_,_,I8],[_,_,_,_,_,_,_,_,I9]],
        Values=[Num|Ls],
        NextI is I+1,
        X in 1..9,
        count(Num,([I7,I8,I9]),#=,X),
```

```
        get_value(Puzzle, down, 9, Ls, Length, NextI).
get_value(_,_,_,_,Length,I):- I>Length.
get_value(_,_,_,_,_,_).
```

7.6 outside_sudoku_cases.pl

```
%-------Outside Sudoku puzzles
%    All puzzles from gmpuzzles.com titles in up comments

%Outside Sudoku R&I
get_out_sudoku_entries(1,Puzzle):-
    Puzzle = [
        up-[
            1-[1,4],
            2-[2,5],
            3-[3,6],
            4-[4,7],
            5-[5,8],
            6-[6,9]],
        left-[
            3-[8],
            4-[2],
            5-[7,9],
            6-[1,3],
            7-[4,8],
            8-[5,6],
            9-[3,7]],
        right-[
            1-[6,9],
            2-[2,8],
            3-[5,7],
            4-[4,6],
            5-[2,3],
            6-[8],
            7-[6]],
        down-[
            4-[3,5],
            5-[1,4],
            6-[2,8],
            7-[2,3],
            8-[4,7],
            9-[5,8]]
    ].

%Dr. Sudoku Prescribes #36      Outside Sudoku
get_out_sudoku_entries(2,Puzzle):-
```

```
Puzzle = [
    up-[
        1-[1,2],
        2-[3,4],
        3-[5,6],
        4-[7,8],
        5-[9,1],
        6-[2,3],
        7-[4,5],
        8-[6,7],
        9-[8,9]
    ],
    left-[
        1-[8,9],
        2-[6,7],
        3-[4,5],
        4-[2,3],
        5-[9,1],
        6-[7,8],
        7-[5,6],
        8-[3,4],
        9-[1,2]
    ],
    right-[
        1-[1],
        2-[2],
        3-[3],
        4-[4],
        5-[5],
        6-[6],
        7-[7],
        8-[8],
        9-[9]
    ],
    down-[
        1-[3],
        2-[1],
        3-[8],
        4-[4],
        5-[2],
        6-[9],
        7-[7],
        8-[5],
        9-[7]
    ]
```

```
    ].

%Outside Sudoku by John Bulten
get_out_sudoku_entries(3,Puzzle):-
    Puzzle = [
        up-[
            1-[3],
            2-[1,4,1],
            3-[5,9],
            4-[2,6],
            5-[5,3,5],
            6-[8,9,7],
            7-[9],
            8-[3,2,3],
            9-[8,4]],
        left-[
            1-[7,8,1],
            2-[3],
            4-[5,9,2],
            6-[4,4],
            7-[9],
            8-[7,4]],
        right-[
            1-[6,2,6],
            2-[4],
            3-[3,3],
            4-[8,3],
            5-[2,7],
            6-[9],
            7-[5,2],
            8-[8,8],
            9-[4,1,9]],
        down-[
            1-[8,2,9],
            2-[5],
            3-[1],
            4-[3,7,5],
            5-[9,9],
            7-[3],
            8-[9],
            9-[7,1,6]]
    ].

%Dr. Sudoku Prescribes #32      Outside Sudoku
get_out_sudoku_entries(4,Puzzle):-
```

```
    Puzzle = [
        up-[
            2-[1,4,7],
            3-[2,5,8],
            4-[3,6,9],
            6-[1,4,7],
            7-[2,5,8],
            8-[3,6,9]],
        left-[
            2-[1,2,3],
            3-[4,5,6],
            4-[7,8,9],
            6-[1,2,3],
            7-[4,5,6],
            8-[7,8,9]],
        right-[
            1-[3],
            3-[1,2],
            4-[3,4],
            6-[6,7],
            7-[8,9],
            9-[5]],
        down-[
            1-[7],
            3-[3,4],
            4-[1,2],
            6-[5,8],
            7-[3,6],
            9-[9]]
    ].

%Dr. Sudoku Prescribes #34      Outside Sudoku
get_out_sudoku_entries(5,Puzzle):-
    Puzzle = [
        up-[
            1-[5],
            2-[8,9],
            3-[1],
            4-[2],
            6-[4],
            7-[8],
            8-[1,6],
            9-[3,2]],
        left-[
            1-[5],
```

```
                2-[3,4],
                3-[2,1],
                4-[1,3],
                5-[8],
                6-[5],
                7-[3],
                8-[2],
                9-[1]],
        right-[
                1-[6,4],
                2-[1,2,8],
                4-[2],
                5-[5,6],
                6-[3,1,4],
                7-[1],
                8-[8],
                9-[2,7]],
        down-[
                1-[9],
                2-[5,1,2],
                3-[3],
                4-[3,4],
                6-[2,1,6],
                7-[1,2,5],
                9-[6,4]]
    ].
```

7.7 solver.pl

```
:- use_module(library(clpfd)).
sudoku(Rows) :-
        maplist(all_distinct, Rows),
        transpose(Rows, Columns),
        maplist(all_distinct, Columns),
        Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
        blocks(As, Bs, Cs),
        blocks(Ds, Es, Fs),
        blocks(Gs, Hs, Is).

blocks([], [], []).
blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
        all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
        blocks(Ns1, Ns2, Ns3).

sudoku_old([[A1,B1,C1,D1,E1,F1,G1,H1,I1],
        [A2,B2,C2,D2,E2,F2,G2,H2,I2],
```

```
      [A3,B3,C3,D3,E3,F3,G3,H3,I3],
      [A4,B4,C4,D4,E4,F4,G4,H4,I4],
      [A5,B5,C5,D5,E5,F5,G5,H5,I5],
      [A6,B6,C6,D6,E6,F6,G6,H6,I6],
      [A7,B7,C7,D7,E7,F7,G7,H7,I7],
      [A8,B8,C8,D8,E8,F8,G8,H8,I8],
      [A9,B9,C9,D9,E9,F9,G9,H9,I9]]):-
Vars = [A1,B1,C1,D1,E1,F1,G1,H1,I1,
        A2,B2,C2,D2,E2,F2,G2,H2,I2,
        A3,B3,C3,D3,E3,F3,G3,H3,I3,
        A4,B4,C4,D4,E4,F4,G4,H4,I4,
        A5,B5,C5,D5,E5,F5,G5,H5,I5,
        A6,B6,C6,D6,E6,F6,G6,H6,I6,
        A7,B7,C7,D7,E7,F7,G7,H7,I7,
        A8,B8,C8,D8,E8,F8,G8,H8,I8,
        A9,B9,C9,D9,E9,F9,G9,H9,I9],
domain(Vars,1,9),
% Rows
all_different([A1,B1,C1,D1,E1,F1,G1,H1,I1]),
all_different([A2,B2,C2,D2,E2,F2,G2,H2,I2]),
all_different([A3,B3,C3,D3,E3,F3,G3,H3,I3]),
all_different([A4,B4,C4,D4,E4,F4,G4,H4,I4]),
all_different([A5,B5,C5,D5,E5,F5,G5,H5,I5]),
all_different([A6,B6,C6,D6,E6,F6,G6,H6,I6]),
all_different([A7,B7,C7,D7,E7,F7,G7,H7,I7]),
all_different([A8,B8,C8,D8,E8,F8,G8,H8,I8]),
all_different([A9,B9,C9,D9,E9,F9,G9,H9,I9]),

% Columns
all_different([A1,A2,A3,A4,A5,A6,A7,A8,A9]),
all_different([B1,B2,B3,B4,B5,B6,B7,B8,B9]),
all_different([C1,C2,C3,C4,C5,C6,C7,C8,C9]),
all_different([D1,D2,D3,D4,D5,D6,D7,D8,D9]),
all_different([E1,E2,E3,E4,E5,E6,E7,E8,E9]),
all_different([F1,F2,F3,F4,F5,F6,F7,F8,F9]),
all_different([G1,G2,G3,G4,G5,G6,G7,G8,G9]),
all_different([H1,H2,H3,H4,H5,H6,H7,H8,H9]),
all_different([I1,I2,I3,I4,I5,I6,I7,I8,I9]),

%Squares
all_different([A1,A2,A3,B1,B2,B3,C1,C2,C3]),
all_different([D1,D2,D3,E1,E2,E3,F1,F2,F3]),
all_different([G1,G2,G3,H1,H2,H3,I1,I2,I3]),

all_different([A4,A5,A6,B4,B5,B6,C4,C5,C6]),
```

```
    all_different([D4,D5,D6,E4,E5,E6,F4,F5,F6]),
    all_different([G4,G5,G6,H4,H5,H6,I4,I5,I6]),

    all_different([A7,A8,A9,B7,B8,B9,C7,C8,C9]),
    all_different([D7,D8,D9,E7,E8,E9,F7,F8,F9]),
    all_different([G7,G8,G9,H7,H8,H9,I7,I8,I9]),

    labeling([],Vars).
```

7.8 sudoku_cases.pl

```
get_sudoku(1,Puzzle):-
    Puzzle = [
    [2,_,_,_,9,_,_,_,1],
    [3,_,9,_,_,7,_,_,_],
    [_,_,1,_,4,_,_,7,_],
    [_,6,_,_,_,_,_,_,_],
    [_,_,_,_,_,3,_,_,_],
    [_,_,8,6,_,_,7,9,_],
    [6,_,_,7,_,_,8,_,_],
    [1,2,3,_,_,8,_,_,_],
    [_,8,7,_,_,4,3,_,_]
    ].

get_sudoku(2,Puzzle):-
    Puzzle = [
    [1,2,3,4,5,6,7,8,9],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_]
    ].

%Hard sudoku from https://www.websudoku.com/?level=3
get_sudoku(3,Puzzle):-
    Puzzle = [
    [8,_,_,_,1,_,_,_,_],
    [_,_,_,_,_,7,_,5,_],
    [_,7,6,_,_,2,_,_,9],
    [_,3,_,_,_,_,_,7,5],
    [2,_,_,1,4,5,_,_,3],
    [6,8,_,_,_,_,_,2,_],
```

```
    [1,_,_,8,_,_,2,4,_],
    [_,6,_,3,_,_,_,_,_],
    [_,_,_,_,5,_,_,_,7]
    ].

%World's Hardest Sudoku https://www.telegraph.co.uk/news/science/science-news/93
get_sudoku(4,Puzzle):-
    Puzzle = [
    [8,_,_,_,_,_,_,_,_],
    [_,_,3,6,_,_,_,_,_],
    [_,7,_,_,9,_,2,_,_],
    [_,5,_,_,_,7,_,_,_],
    [_,_,_,_,4,5,7,_,_],
    [_,_,_,1,_,_,_,3,_],
    [_,_,1,_,_,_,_,6,8],
    [_,_,8,5,_,_,_,1,_],
    [_,9,_,_,_,_,4,_,_]
    ].

get_sudoku(5,Puzzle):-
    Puzzle = [
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [4,5,2,_,_,_,_,_,_],
    [3,1,_,_,_,_,_,_,_],
    [2,3,5,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [1,4,_,_,_,_,_,_,_]
    ].

get_sudoku(6,Puzzle):-
    Puzzle = [
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [2,4,5,_,_,_,_,_,_],
    [1,3,_,_,_,_,_,_,_],
    [3,5,2,_,_,_,_,_,_],
    [_,_,_,_,_,_,_,_,_],
    [4,1,_,_,_,_,_,_,_]
    ].
```

```
get_sudoku(7,Puzzle):-
    Puzzle = [
    [1,2,3,4,5,6,7,8,9],
    [4,5,6,7,8,9,1,2,3],
    [7,8,9,1,2,3,4,5,6],
    [2,1,4,3,6,5,8,9,7],
    [3,6,5,8,9,7,2,1,4],
    [8,9,7,2,1,4,3,6,5],
    [5,3,1,6,4,2,9,7,8],
    [6,4,2,9,7,8,5,3,1],
    [_,_,_,_,_,_,_,_,_]
    ].
get_sudoku(_,_).
```