Resolução de um Problema de um Puzzle em Prolog com Restrições: Crazy Pavement

Inês Ferreira (up201305866) e **Tomás Caldas**(up201404990)

FEUP-PLOG, CrazyPavement3

Resumo O artigo tem como objetivo complementar o segundo projeto da unidade curricular de Programação em Lógica. O puzzle Crazy Pavement consiste em pintar certas regiões, respeitando as restrições dadas, relacionadas com a soma de algumas linhas e algumas colunas.

Keywords: Crazy Pavement, restrições, regiões, soma, linhas, colunas

1 Introdução

O objetivo deste trabalho é implementar a resolução de um problema de decisão ou otimização utilizando Programação em Lógica com restrições.

Para tal, foi escolhido o puzzle Crazy Pavement. O tabuleiro está dividido por regiões, sendo que o utilizador tem de printar um certo número de peças em determinadas colunas ou linhas, sem esquecer que ao pintar uma peça tem de pintar todas da sua região.

O presente artigo irá, numa primeira parte, descrever com mais detalhe este problema de decisão. Posteriormente, irá descrever a modelação do problema como Programação em Lógica com Restrições. Por fim, serão abordadas as soluões, bem como resultados, seguindo-se as conclusões.

2 Descrição do Problema

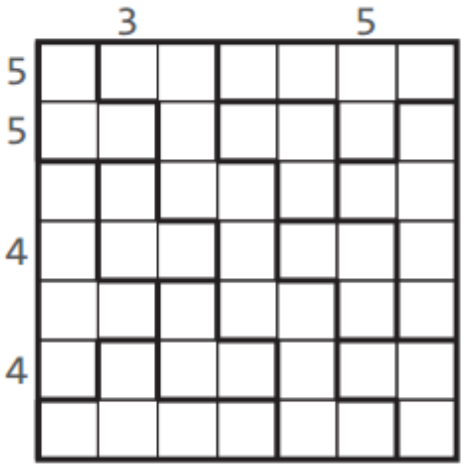

Figura 1: Exemplo de um puzzle

Na imagem anterior, podemos ver um exemplo de um tabuleiro do Crazy Pavement. Com foi direito anteriormente, o tabuleiro está dividido em secções, delimitadas pelas linhas mais grossas. Os números de lado e em cima do tabuleiro dizem respeito à soma da linha ou coluna relativa, respetivamente. Por exemplo, sabe-se que a primeira linha tem de ter cinco casas pintadas e a segunda coluna terá de três. A conjugação destas restrições resultam, neste caso, para uma única solução.

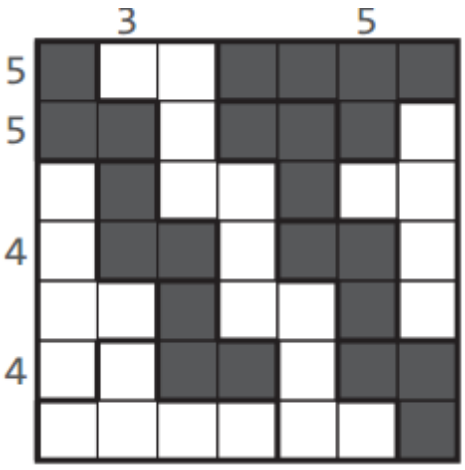

Figura 2: Solução do exemplo anterior

3 Abordagem

O puzzle é representado por uma lista de listas, em que cada lista representa uma linha do puzzle. Os elementos das listas representam as peças, em que estas são identificadas por variáveis iguais se pertencerem a uma mesma região. Pensamos ser a melhor maneira de abordar o problema, visto que as peças de uma região terão de ser todas pintadas ou não pintadas.

Na secção anterior foi apresentado um tabuleiro, com dimensão 7x7. Para assegurar que o programa funcionava com tabuleiros de dimensões maiores, desenhamos dois tabuleiros de maiores dimensões, um de 10x10 e outro de 15x15.

3.1 Variáveis de decisão

As variáveis de decisão são todas as peças do tabuleiro, pois no início estão todas vazias e não temos conhecimento do valor de nenhuma delas. O domínio destas variáveis é 0 ou 1, em que 0 significa que não foi pintada e, por outro lado, 1 significa que foi pintada.

3.2 Restrições

As restrições estão relacionadas com os números de lado e em cima do tabuleiro, dizendo respeito às linhas e colunas, respetivamente.

Para cada um dos tabuleiros, definimos uma lista para as restrições de cima e do lado. Por exemplo, para o exemplo mostrado anteriormente,temos

horizontal_1([5,5,-1,4,-1,4,-1]).
vertical_1([-1,3,-1,-1,-1,5,-1]).

Para implementarmos estas restrições, definimos duas funções, uma que trata as restrições horizontais e outra que trata as restrições verticais.

Relativamente às restrições horizontais, implementamos uma função recursiva n vezes, sendo o n a dimensão do tabuleiro. Quando o elemento n da lista de restrições horizontais é diferente de -1, definimos que a soma da linha n do tabuleiro tem que ser esse valor.

Relativamente às restrições verticais, foi decidido criar uma lista com o elemento n de todas as linhas do tabuleiro. Deste modo, podemos fazer uma restrição semelhante à anterior, em que a soma dos elementos dessa lista, que representa a coluna, tem de ser igual ao valor n, no caso de este ser diferente de -1.

3.3 Estratégia de pesquisa

De modo a ser possível utilizar a função *labelling*, implementamos a função *flatten([LH|LT], Flattened)*, que transforma o tabuleiro (lista de listas) numa só

lista. Procedemos, então, à etiquetagem.

4 Visualização da Solução

Assim que o programa acaba de executar e a solução do puzzle é encontrada, esta é apresentada com a seguinte estrutura:

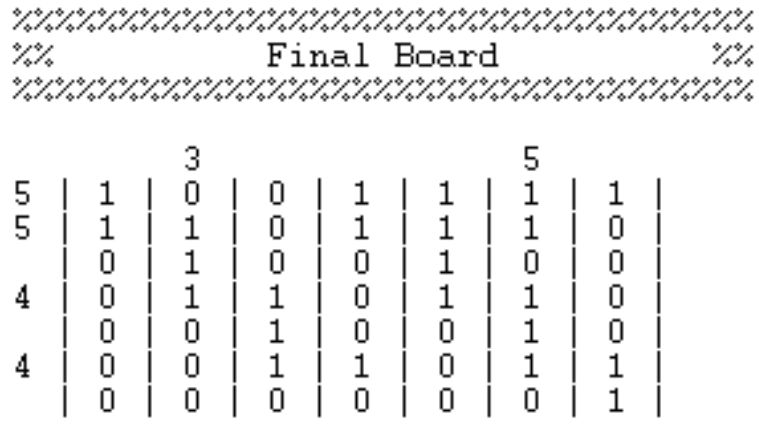

Figura 3: Resolução final do tabuleiro de dificuldade fácil.

Os primeiros números que aparecem tanto na horizontal como na vertical servem para apresentar o número de casas coloridas que cada coluna ou linha

deve ter preenchida. Os 0's e os 1's são respectivamente uma casa vazia e uma casa preenchida.

O processamento desta informação começa no predicado *show_final_board (+Board,+V,+H,+N,+N1,+Lstart,+Llen,+Total,+Counter)* que altera dependendo do nível do puzzle. Este por sua vez invoca o *display_vertical_final (+V,+N,+N1)* para mostrar as restrições a nível vertical e o *display_final_board (+Board,+N,+N1,+H,+Total,+Counter,+Lstart,+Llen)* para as restrições horizontais e o a solução final.

5 Resultados

Nesta secção, podemos verificar o resultado obtido pelos predicados *print_time* e *fd_statistics*. No nosso caso verificamos que o tempo de resolução é automático em todos os níveis, resultando num tempo de execução de 0.00s. No entanto, encontramos algumas incoerências nas estatísticas oferecidas pelo segundo predicado, isto pode se dever ao facto de que não só o tamanho do tabuleiro define a deficuldade, mas também as restrições e a maneira como as peças estão distruibuídas.

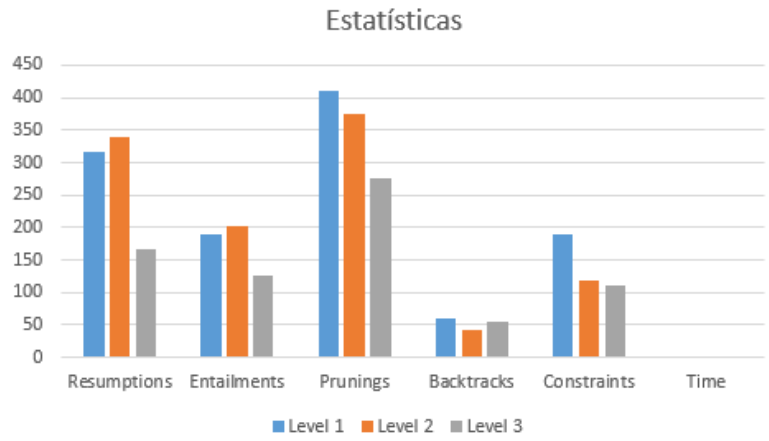

Figura 4: Gráfico com as estatísticas obtidas para os diversos níveis.

6 Conclusões

O grupo conseguiu perceber e interpretar facilmente o puzzle escolhido. Implementar a resolução do tabuleiro 7x7 foi relativamente fácil, mas houve problemas nos tabuleiros seguintes. Não conseguámos encontrar uma solução, devido

a um erro mínimo de implementação das restrições verticais, que não se manifestava no tabuleiro anterior, pois este poderia ser resolvido apenas com as restrições horizontais. Após percebermos o erro, foi fácil corrigi-lo e verificar que solucionava os tabuleiros com maiores dimensões.

Juntamente com o que foi referido anteriormente e a dificuldade em perceber como gerar regiões aleatoriamente, não fomos capazes de gerar problemas aleatoriamente.

Por este motivo, pensamos que com mais um pouco de tempo, seria algo a fazer no futuro.

7 Bibliografia

– http://logicmastersindia.com/lmitests/dl.asp?attachmentid=564

8 Anexos

Ficheiro cli.pl

```
:- use_module(library(clpfd)).
:- use_module(library(lists)).
:-include('board.pl').
:-include('displays.pl').

solve_easy:-board_1(B),initial_1(Bi),vertical_1(V),horizontal_1(H),show_initi
solve_medium:-board_2(B),initial_2(Bi),vertical_2(V),horizontal_2(H),show_ini
solve_hard:-board_3(B),initial_3(Bi),vertical_3(V),horizontal_3(H),show_initi

clearScreen:-
        printBlank(65).

printBlank(X):-
        X > 0,
        nl,
        X1 is X-1,
        printBlank(X1).

printBlank(_).

mainMenu:-
        clearScreen,
        printMainMenu,
        get_char(In),
        (
          In = '1' -> solve_easy;
          In = '2' -> solve_medium;
          In = '3' -> solve_hard;
          In = '4';
          mainMenu

        ).

printMainMenu:-
        write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n'),
        write('%%          Crazy Pavement         %%\n'),
        write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n'),
        write('%%     Choose board dimensions     %%\n'),
        write('%%             1 - 7x7             %%\n'),
        write('%%            2 - 10x10            %%\n'),
        write('%%            3 - 15x15            %%\n'),
        write('%%                                 %%\n'),
```

```
        write('%%              4 - Exit                %%\n'),
        write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n').

start_game:-mainMenu.
```

Ficheiro board.pl

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


board_1([
          [P1,P2,P2,P3,P3,P3,P3],
          [P1,P1,P2,P7,P7,P3,P8],
          [P4,P5,P2,P2,P7,P8,P8],
          [P4,P5,P5,P2,P9,P9,P8],
          [P4,P4,P6,P2,P2,P9,P8],
          [P4,P11,P6,P6,P2,P10,P10],
          [P11,P11,P11,P11,P2,P2,P10]
          ]).


board_2([
         [P1,P1,P2,P2,P2,P3,P3,P4,P4,P4],
         [P1,P1,P6,P2,P8,P8,P3,P4,P9,P4],
         [P5,P6,P6,P7,P8,P3,P3,P9,P9,P9],
         [P5,P10,P7,P7,P8,P8,P14,P14,P9,P16],
         [P10,P10,P11,P12,P13,P13,P14,P15,P15,P16],
         [P10,P11,P11,P12,P13,P13,P14,P15,P16,P16],
         [P10,P17,P12,P12,P20,P20,P22,P22,P16,P24],
         [P17,P17,P17,P19,P19,P20,P22,P23,P23,P24],
         [P18,P17,P18,P19,P20,P20,P21,P23,P25,P25],
         [P18,P18,P18,P19,P19,P21,P21,P21,P25,P25]
         ]).

board_3([
         [P1,P1,P2,P2,P3,P3,P3,P3,P3,P4,P4,P5,P5,P5,P5],
         [P1,P2,P2,P6,P6,P7,P8,P8,P9,P4,P4,P4,P5,P5,P5],
         [P1,P12,P12,P6,P6,P7,P7,P8,P9,P9,P10,P4,P4,P4,P4],
         [P1,P13,P12,P7,P7,P7,P8,P8,P9,P9,P10,P10,P10,P4,P4],
         [P13,P13,P12,P12,P14,P15,P16,P16,P16,P16,P10,P10,P11,P11,P4],
         [P13,P13,P14,P14,P14,P15,P16,P18,P18,P18,P18,P18,P11,P11,P11],
         [P13,P13,P25,P24,P15,P15,P15,P17,P17,P38,P20,P20,P20,P20,P21],
         [P27,P26,P25,P24,P24,P15,P17,P17,P38,P38,P22,P22,P20,P21,P21],
         [P27,P26,P25,P25,P25,P15,P17,P17,P38,P22,P22,P21,P21,P21,P23],
         [P27,P26,P26,P26,P30,P15,P15,P17,P38,P38,P22,P22,P22,P21,P23],
         [P27,P28,P29,P30,P30,P30,P30,P17,P38,P39,P22,P22,P22,P22,P23],
```

```
        [P27,P28,P29,P30,P31,P31,P31,P39,P39,P39,P22,P23,P23,P23,P23],
        [P28,P28,P29,P29,P31,P32,P32,P40,P40,P40,P40,P36,P36,P37,P37],
        [P28,P28,P29,P29,P32,P32,P32,P32,P34,P35,P35,P35,P35,P42,P42],
        [P28,P28,P29,P29,P33,P33,P33,P32,P34,P41,P41,P41,P35,P42,P42]
        ]).


horizontal_1([5,5,-1,4,-1,4,-1]).
vertical_1([-1,3,-1,-1,-1,5,-1]).

horizontal_2([5,5,6,6,-1,5,5,-1,5,5]).
vertical_2([6,6,-1,-1,-1,-1,-1,-1,4,4]).

horizontal_3([9,8,10,6,4,6,9,6,6,6,4,12,7,4,10]).
vertical_3([-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]).




board_horizontal(_,_,0).

board_horizontal(Board,Vertical,N):-
  element(N,Vertical,LineSum),
  LineSum #= -1,
  N1 is N - 1,
  board_horizontal(Board,Vertical, N1).


board_horizontal(Board,Vertical,N):-
  select_list(Board, 1, N, List),
  element(N, Vertical, LineSum),
  domain(List,0,1),
  sum(List,#=,LineSum),
  N1 is N - 1,
  board_horizontal(Board,Vertical,N1).

select_list([Head|Tail], N1, N, List):-
  N1 #\= N,
  N2 is N1 + 1,
  select_list(Tail, N2, N, List).

select_list([Head|_], N, N, Head).
```

```
select_list([],_,_,[]).

board_vertical(_,_,0).

board_vertical(Board, Horizontal, N):-
  element(N, Horizontal, ColSum),
  ColSum #= -1,
  N1 is N - 1,
  board_vertical(Board, Horizontal, N1).


board_vertical(Board, Horizontal, N):-
  get_col(Board, N, Col),
  element(N, Horizontal, ColSum),
  domain(Col, 0, 1),
  sum(Col, #=, ColSum),
  N1 is N - 1,
  board_vertical(Board, Horizontal, N1).


  get_col([],_,[]).
  get_col([Head|Tail], N, [ColHead|ColTail]):-
    element(N, Head, Val),
    ColHead = Val,
    get_col(Tail,N,ColTail).



flatten([],[]).
flatten([LH|LT], Flattened) :-
        is_list(LH),
        flatten(LH, FlattenedTemp),
        append(FlattenedTemp, LT2, Flattened),
        flatten(LT, LT2).

flatten([LH | LT], [LH | FlattenedT]) :-
        \+is_list(LH),
        flatten(LT, FlattenedT).

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

```
solve(Board, Vertical, Horizontal, BoardFlat):-
  length(Board, Length),
  board_vertical(Board, Vertical, Length),
  board_horizontal(Board, Horizontal, Length),
  flatten(Board, BoardFlat),
  reset_timer,
  labeling([], BoardFlat),
  print_time,
  fd_statistics.
```

Ficheiro displays.pl

```
:-use_module(library(lists)).
:-use_module(library(clpfd)).
:-include('board.pl').


initial_1([
          [p1 ,p2 ,p2 ,p3 ,p3 ,p3 ,p3 ],
          [p1 ,p1 ,p2 ,p7 ,p7 ,p3 ,p8 ],
          [p4 ,p5 ,p2 ,p2 ,p7 ,p8 ,p8 ],
          [p4 ,p5 ,p5 ,p2 ,p9 ,p9 ,p8 ],
          [p4 ,p4 ,p6 ,p2 ,p2 ,p9 ,p8 ],
          [p4 ,p11,p6 ,p6 ,p2 ,p10,p10],
          [p11,p11,p11,p11,p2 ,p2 ,p10]
          ]).


initial_2([
          [p1,p1,p2,p2,p2,p3,p3,p4,p4,p4],
          [p1,p1,p6,p2,p8,p8,p3,p4,p9,p4],
          [p5,p6,p6,p7,p8,p3,p3,p9,p9,p9],
          [p5,p10,p7,p7,p8,p8,p14,p14,p9,p16],
          [p10,p10,p11,p12,p13,p13,p14,p15,p15,p16],
          [p10,p11,p11,p12,p13,p13,p14,p15,p16,p16],
          [p10,p17,p12,p12,p20,p20,p22,p22,p16,p24],
          [p17,p17,p17,p19,p19,p20,p22,p23,p23,p24],
          [p18,p17,p18,p19,p20,p20,p21,p23,p25,p25],
          [p18,p18,p18,p19,p19,p21,p21,p21,p25,p25]
          ]).


initial_3([
         [p1,p1,p2,p2,p3,p3,p3,p3,p3,p4,p4,p5,p5,p5,p5],
         [p1,p2,p2,p6,p6,p7,p8,p8,p9,p4,p4,p4,p5,p5,p5],
         [p1,p12,p12,p6,p6,p7,p7,p8,p9,p9,p10,p4,p4,p4,p4],
```

```
        [p1,p13,p12,p7,p7,p7,p8,p8,p9,p9,p10,p10,p10,p4,p4],
        [p13,p13,p12,p12,p14,p15,p16,p16,p16,p16,p10,p10,p11,p11,p4],
        [p13,p13,p14,p14,p14,p15,p16,p18,p18,p18,p18,p18,p11,p11,p11],
        [p13,p13,p25,p24,p15,p15,p15,p17,p17,p38,p20,p20,p20,p20,p21],
        [p27,p26,p25,p24,p24,p15,p17,p17,p38,p38,p22,p22,p20,p21,p21],
        [p27,p26,p25,p25,p25,p15,p17,p17,p38,p22,p22,p21,p21,p21,p23],
        [p27,p26,p26,p26,p30,p15,p15,p17,p38,p38,p22,p22,p22,p21,p23],
        [p27,p28,p29,p30,p30,p30,p30,p17,p38,p39,p22,p22,p22,p22,p23],
        [p27,p28,p29,p30,p31,p31,p31,p39,p39,p39,p22,p23,p23,p23,p23],
        [p28,p28,p29,p29,p31,p32,p32,p40,p40,p40,p40,p36,p36,p37,p37],
        [p28,p28,p29,p29,p32,p32,p32,p32,p34,p35,p35,p35,p35,p42,p42],
        [p28,p28,p29,p29,p33,p33,p33,p32,p34,p41,p41,p41,p35,p42,p42]
        ]).

display_vertical(V,N,N1):-
  N#>=N1,
  element(N1,V,Number),
  (
  Number = -1 -> write('      ');
  write(Number), write('    ')
  ),
  N2 is N1 +1,
  display_vertical(V,N,N2).

display_vertical(V,N,N1):-
  nl.

display_vertical([],_,_).



display_vertical_final(V,N,N1):-
  N#>=N1,
  element(N1,V,Number),
  (
  Number = -1 -> write('    ');
  write(Number), write('   ')
  ),
  N2 is N1 +1,
  display_vertical_final(V,N,N2).

display_vertical_final(V,N,N1):-
  nl.

display_vertical_final([],_,_).
```

```
display_initial_board(Board,N,N1,H):-
  N#>=N1,
  element(N1,H,Number),
  (
  Number = -1 -> write('  '), write(' | ');
  Number >= 10 ->write(Number), write('| ');
  write(Number),write(' | ')
  ),
  select_list(Board,1,N1,Row),
  display_line(Row),
  nl,
  N2 is N1+1,
  display_initial_board(Board,N,N2,H).

display_initial_board(Board,N,N1,H):-
  nl.
display_initial_board([],_,_,[]).


display_line([E1|ES]) :- translate(E1,V), write(V), write(' | '), display_lin

display_line([]).


display_line2([E1|ES]) :-  write(E1), write(' | '), display_line2(ES).

display_line2([]).

display_final_board(Board,N,N1,H,Total,Counter,Lstart,Llen):-
  Total#>=Counter,
  element(N1,H,Number),
  (
  Number = -1 -> write('  '), write(' | ');
  Number >= 10 ->write(Number), write('| ');
  write(Number),write(' | ')
  ),
  sublist(Board,Row,Lstart,Llen),
  display_line2(Row),
  nl,
  N2 is N1+1,
  Counter1 is Counter +Llen,
  Lstart1 is Lstart +Llen,
```

```
  display_final_board(Board,N,N2,H,Total,Counter1,Lstart1,Llen).

  display_final_board(Board,N,N1,H,Total,Counter,Lstart,Llen):-
    nl.
  display_final_board([],_,_,[],_,_,_,_).



show_initial_board(Board,N,N1,V,H):-
  nl,
  nl,
  nl,
  write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n'),
  write('%%              Initial Board           %%\n'),
  write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n'),
  nl,
  write('       '),
  display_vertical(V,N,N1),
  display_initial_board(Board,N,N1,V).

show_final_board(Board,V,H,N,N1,Lstart,Llen,Total,Counter):-
  nl,
  nl,
  nl,
  write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n'),
  write('%%               Final Board            %%\n'),
  write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n'),
  nl,
  write('       '),
  display_vertical_final(V,N,N1),
  display_final_board(Board,N,N1,H,Total,Counter,Lstart,Llen).




translate(p1,'01').
translate(p2,'02').
translate(p3,'03').
translate(p4,'04').
translate(p5,'05').
translate(p6,'06').
translate(p7,'07').
translate(p8,'08').
translate(p9,'09').
```

```
translate(p10,'10').
translate(p11,'11').
translate(p12,'12').
translate(p13,'13').
translate(p14,'14').
translate(p15,'15').
translate(p16,'16').
translate(p17,'17').
translate(p18,'18').
translate(p19,'19').
translate(p20,'20').
translate(p21,'21').
translate(p22,'22').
translate(p23,'23').
translate(p24,'24').
translate(p25,'25').
translate(p26,'26').
translate(p27,'27').
translate(p28,'28').
translate(p29,'29').
translate(p30,'30').
translate(p31,'31').
translate(p32,'32').
translate(p33,'33').
translate(p34,'34').
translate(p35,'35').
translate(p36,'36').
translate(p37,'37').
translate(p38,'38').
translate(p39,'39').
translate(p40,'40').
translate(p41,'41').
translate(p42,'42').
```