Pointing At The Crowd

Resolução de problema de decisão usando programação em lógica com restrições

Leonor Mendes de Freitas, up201207603@fe.up.pt
Marisa Oliveira, up201308594@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

Resumo: Este artigo complementa o segundo projecto da Unidade Curricular de Programação em Lógica constante no plano de estudos do Mestrado Integrado em Engenharia Informática e de Computação. O projecto consiste num programa escrito em prolog capaz de resolver qualquer tabuleiro do problema de decisão Point At the Crowd.

Palavras-chave: sicstus, prolog, pointing at the crowd

1. Introdução

O objetivo deste trabalho consiste em implementar a resolução de um problema de decisão ou otimização em Prolog com restrições. Foi dada a oportunidade aos estudantes de escolherem um dos dois tipos de problema. O nosso grupo decidiu implementar o problema de decisão **Pointing At The Crowd.** O puzzle consiste num tabuleiro **hexagonal**, onde o jogador tem de colocar pontos nas células do tabuleiro de forma a que na direção das setas, previamente inseridas no tabuleiro, fique um número maior de pontos do que nas outras direções em que a seta poderia apontar.

Este artigo descreve detalhadamente este problema e está dividido nas seguintes secções:

- **Descrição do problema** onde será explicitado, de uma forma detalhada, no que consiste o puzzle;
- **Abordagem** escolhida para a resolução do problema, suas respectivas variáveis de decisão e restrições;
- **Visualização e resultados** onde será exposto os resultados obtidos e suas conclusões.

2. Descrição do Problema

Pointing At The Crowd consiste em colocar pontos num tabuleiro **hexagonal**, onde existe um número dado de setas. As setas vão ter uma determinada direção e nessa direção terá de existir um maior número de pontos do que nas outras possíveis direções da seta.

As regiões sem setas podem servir para inserir os pontos com determinadas restrições que irão ser abordadas mais adiante neste relatório.

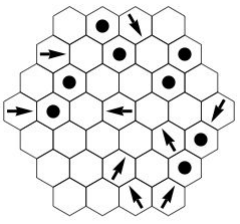

Figura 1. Exemplo de um tabuleiro de *Pointing At The Crowd* depois da solução

3. Abordagem

Na implementação deste puzzle em Prolog foi optado por representar as setas, como uma lista de listas, tendo em conta a sua posição e direção no tabuleiro.

```
18  setas1([[3,1,2], [1,2,1], [1,4,1], [4,4,4], [7,4,3], [5,5,5], [3,6,6], [3,7,5], [4,7,6]]). %setas do primeiro tabuleiro
19  setas2([[2,1,3], [5,3,4], [4,4,1], [7,4,5], [1,6,6], [3,6,5], [4,7,6]]). %setas do 2 tabuleiro (do exemplo)
```

Cada lista contém listas com as posições da seta, e a sua respetiva orientação.

Em relação à orientação, considerámos:

Para representação de um tabuleiro que pudesse ter um tamanho variado o grupo decidiu pedir ao utilizador o tamanho do lado do hexágono, ou seja, o tamanho da linha mais pequena. A partir deste tamanho foi possível calcular o tamanho da linha maior do hexágono (a do meio). Com isto foi possível criar listas com o tamanho certo das linhas, mas para o fazer foi dividida a implementação em duas partes: uma que abordasse a parte superior do tabuleiro e outra a parte inferior.

3.1 Variáveis de decisão

Neste jogo as variáveis de de decisão correspondem às células livres do jogo (sem setas). O objectivo será decidir, entre as células livres, onde colocar os pontos.
O domínio da variável vai ser o tamanho do tabuleiro menos o número de setas que existem, ou seja, se tivermos um tabuleiro de lado 4, significa que temos 2*(6+5+4) + 7 = 37 células e caso tenhamos 5 setas, significa que temos 32 células livres, sendo que o domínio vai ser de 0 a 37.

3.2 Restrições

Só irá existir uma restrição para a resolução do jogo:

- Uma seta aponta para pelo menos um ponto, e essa direção vai ter que ter mais pontos do que nas outras direções para as quais a seta pode apontar.

3.3 Função de avaliação

Para avaliar a solução obtida utilizando o SICStus Prolog recorreu-se aos predicados ***fd_statistics*** e ***statistics*** que permitem aceder estatísticas de execução.
No caso do predicado ***fd_statistics*** foi invocado no final da execução do programa, mostrando:

```
Resumptions: 444
Entailments: 0
Prunings: 375
Backtracks: 0
Constraints created: 251
```

Resumptions: número de vezes que uma instrução foi concluída;
Entailments: número de vezes que uma consequência foi detetada por uma restrição;
Prunings: número de opções inviáveis;
Backtracks: número de vezes que uma contradição foi encontrada e teve de encontrar novas soluções;
Constraints created: Número de restrições criada;

O predicado ***statistics*** foi usado para medir o tempo da execução, dando o resultado em milisegundos.

3.4 Estratégia de pesquisa

A estratégia de pesquisa está estruturada em duas partes: nos predicados **case** do ficheiro *Cases.pl* e no predicado **pointing** do ficheiro *PointingAtTheCrowd.pl*.

Os predicados case vão guardar numa lista os elementos que se encontram na mesma direcção de uma determinada seta. Desta forma é possível, para qualquer que seja a direcção da seta, obter todos os elementos de maneira a saber quais os pontos que já se encontram nessa linha.

O predicado **pointing**, que será o principal predicado para construir o tabuleiro, irá utilizar o predicado **checkSetas** (que utiliza o predicado **comparaDir**) para fazer a verificação das direções das setas do tabuleiro e o predicado **labeling**/2 da biblioteca clpfd do SICStus Prolog que permite avaliar e escolher quais os locais mais adequados para colocar os pontos.

4. Visualização da solução

Os predicados de visualização do tabuleiro encontram-se no ficheiro ***PointingAtTheCrowd.pl***. A visualização do tabuleiro foi dividida em duas partes: um predicado permite visualizar a metade de cima do tabuleiro e outro permite visualizar a metade de baixo. Desta forma, sendo que o tabuleiro tem a forma de um hexágono aproximadamente, cada parte do tabuleiro irá ser simétrica da outra e terá a forma aproximada de um trapézio, conseguidas através de predicados que calculam o espaçamento necessário para obter esta forma. Os predicados baseiam-se maioritariamente em recursividade.

- **espacamento(Size):**

Calcula e imprime recursivamente o espaçamento no início de cada linha, de forma a obtermos a forma hexagonal do tabuleiro.

- **parteCima(Size, [H|Tail]):**

Escreve cada elemento, espaçado do elemento seguinte, das linhas da metade superior do tabuleiro. Este predicado é chamado recursivamente enquanto não se tiver chegado ao fim da lista da linha respectiva (fim conhecido através do contador Size).

- **verParteCima([H|Tail], Size1, Size2) :**

Permite, chamando os predicados espacamento e parteCima, calcular e visualizar a metade superior do tabuleiro. O predicado é chamado recursivamente até todas as linhas desta metade estarem impressas no ecrã.

- **parteBaixo(Size, [H|Tail]):**

Escreve cada elemento, espaçado do elemento seguinte, das linhas da metade inferior do tabuleiro. Este predicado é chamado recursivamente enquanto não se tiver chegado ao fim da lista da linha respectiva (fim conhecido através do contador Size).

- **verParteBaixo([H|Tail], Size1, Size2):**

Permite, chamando os predicados espacamento e parteBaixo, calcular e visualizar a metade inferior do tabuleiro. O predicado é chamado recursivamente até todas as linhas desta metade estarem impressas no ecrã.

- **divide([H|Tail], N, List1, List2, Aux1):**

Este predicado permite dividir a lista que compõe o tabuleiro em duas listas, de forma a que uma das listas seja usada para imprimir a parte superior do tabuleiro e a outra a parte inferior.

- **mostraTab(Board):**

Este predicado chama os predicados, dos predicados descritos acima, necessários para a visualização do tabuleiro completo.

5. Resultados

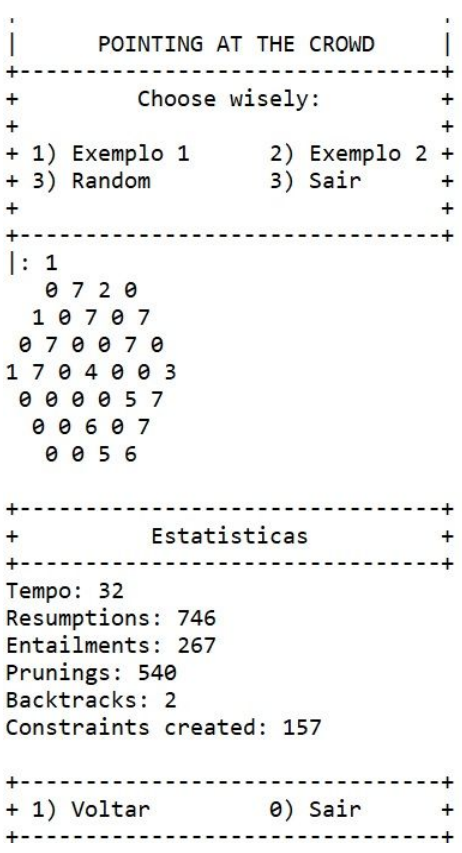

Figura 3. Exemplo de Tabuleiro pré-definido Pointing At The Crowd de dimensão 4 e as respetivas estatísticas de resolução, geradas de acordo com a dimensão e com o número de setas presentes no tabuleiro (9).

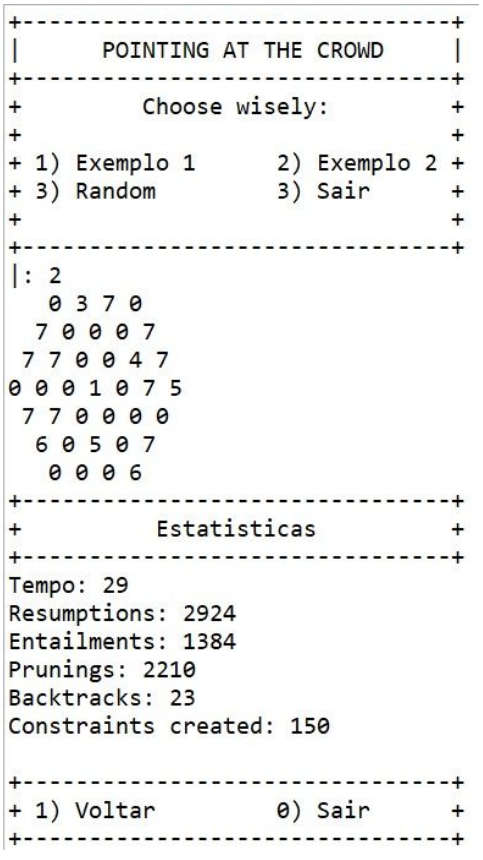

Figura 4. Exemplo de Tabuleiro pré-definido Pointing At The Crowd de dimensão 4 e as respetivas estatísticas de resolução, geradas de acordo com a dimensão e com o número de setas presentes no tabuleiro (7).

```
+--------------------------------+
|: 3
Lado do hexagono(maior do que 4):
|: 7
      0 3 0 7 0 0 0
     0 7 0 7 7 0 0 0
    7 7 0 0 4 7 0 0 0
   7 0 0 1 7 0 5 7 7 0
  0 0 0 0 0 0 0 0 0 0 0
 6 0 5 0 0 0 0 0 0 0 0 0
0 0 0 6 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0
     0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0
+--------------------------------+
+          Estatisticas          +
+--------------------------------+
Tempo: 4079
Resumptions: 8317
Entailments: 5462
Prunings: 7413
Backtracks: 249
Constraints created: 426
```

Figura 5. Exemplo de Tabuleiro Pointing At The Crowd gerado aleatoriamente de dimensão 7 e as respetivas estatísticas de resolução, geradas de acordo com a dimensão e com o número de setas presentes no tabuleiro (7).

Analisando as estatísticas obtidas da resolução do problema apresentado, podemos concluir, tal como seria esperado, a complexidade da resolução aumenta tanto com o aumento da dimensão do tabuleiro como com o aumento do número de setas no tabuleiro.

Este aumento pode ser observado, por exemplo a nível temporal, do primeiro exemplo para o segundo tabuleiro com o aumento do número de setas (quanto maior o número de setas mais pontos terão que ser colocados no tabuleiro e mais linhas terão que ser analisadas) ou do primeiro tabuleiro para o tabuleiro gerado aleatoriamente, em que tanto o número de setas como a dimensão do tabuleiro aumentam.

6. Conclusões

Este projecto mostrou-se fundamental para uma melhor compreensão da programação em lógica com restrições. Através deste tipo de lógica de resolução de problemas e da biblioteca do SICStus Prolog, é possível o desenvolvimento de programas complexos com poucas linhas de código, sendo esta uma ferramenta poderosa neste tipo de situações.

Apesar de Prolog se mostrar extremamente útil em resoluções desta natureza, esta linguagem pode também comportar-se como um entrave ao desenvolvimento eficiente de programas uma vez que uma tarefa à primeira vista simples, como imprimir um tabuleiro na consola, se pode tornar numa tarefa verdadeiramente complexa.

Referências

- http://www2.stetson.edu/~efriedma/puzzle/champ/World2014/

- Material disponibilizado na página Moodle da disciplina

Anexos

Código Fonte

Cases.pl

```
%Caso da seta apontar para a direita
case(1, Line, Col, N, Board,List, Final) :-
        NewLine is Line + 1,
        getPos(NewLine, Col, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(1, NewLine, Col, N, Board, [Cell|List], Final);
        case(1, NewLine, Col, N, Board, List, Final))).

%Caso da seta apontar para a diagonal direita inferior
case(2, Line, Col, N, Board, List, Final) :-
        NewCol is Col + 1,
        ((Col >= N,
        getPos(Line, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(2,Line, NewCol, N, Board, [Cell|List], Final);
        case(2, Line, NewCol, N, Board, List, Final))));

        (NewLine is Line + 1,
        getPos(NewLine, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(2, NewLine, NewCol, N, Board, [Cell|List], Final);
        case(2, NewLine, NewCol, N, Board, List, Final))))).

%Caso da seta apontar para a diagonal esquerda inferior
case(3, Line, Col, N, Board, List, Final) :-
        NewCol is Col + 1,
        ((Col < N,
        getPos(Line, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(3, Line, NewCol, N, Board, [Cell|List], Final);
        case(3, Line, NewCol, N, Board, List, Final))));

        (NewLine is Line-1,
        getPos(NewLine, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(3, NewLine, NewCol, N, Board, [Cell|List], Final);
        case(3, NewLine, NewCol, N, Board, List, Final))))).
```

```
%Caso da seta apontar para a esquerda
case(4, Line, Col, N, Board, List, Final) :-
        NewLine is Line - 1,
        getPos(NewLine, Col, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(4, NewLine, Col, N, Board, [Cell|List], Final);
        case(4, NewLine, Col, N, Board, List, Final))).

%Caso da seta apontar para a diagonal superior esquerda
case(5, Line, Col, N, Board, List, Final) :-
        NewCol is Col - 1,
        ((Col > N,
        getPos(Line, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(5, Line, NewCol, N, Board, [Cell|List], Final);
        case(5, Line, NewCol, N, Board,List, Final))));

        (NewLine is Line - 1,
        getPos(NewLine, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(5, NewLine, NewCol, N, Board,[Cell|List], Final);
        case(5, NewLine, NewCol, N, Board, List, Final))))).

%Caso da seta apontar para a diagonal superior direita
case(6, Line, Col, N, Board,List, Final) :-
        NewCol is Col - 1,
        ((Col =< N,
        getPos(Line, NewCol, Board, Cell),
        (Cell = -1, reverse(List, Final);
        var(Cell),
        case(6, Line, NewCol, N, Board,[Cell|List], Final);
        case(6, Line, NewCol, N, Board, List, Final)));

        (NewLine is Line + 1,
        getPos(NewLine, NewCol, Board, Cell),
        (Cell = -1 -> (reverse(List, Final), true);
        (var(Cell) -> case(6, NewLine, NewCol, N, Board,[Cell|List], Final);
        case(6, NewLine, NewCol, N, Board, List, Final))))).
```

PointingAtTheCrowd.pl

```
:- use_module(library(lists)).
:- use_module(library(clpfd)).
:- use_module(library(random)).
:- ensure_loaded('Cases.pl').
```

```
% posicao e direcao das setas

% direita - 1
% baixoDir - 2
% baixoEsq - 3
% esq - 4
% cimaEsq - 5
% cimaDir - 6

% pontos - 7

% setas ([X1,Y1,Dir],...).
setas1([[3,1,2], [1,2,1], [1,4,1], [4,4,4], [7,4,3], [5,5,5], [3,6,6], [3,7,5], [4,7,6]]). %setas do primeiro tabuleiro
setas2([[2,1,3], [5,3,4], [4,4,1], [7,4,5], [1,6,6], [3,6,5], [4,7,6]]). %setas do 2 tabuleiro (do exemplo)
setas3([[2,1,3], [5,3,4], [4,5,1], [3,3,5], [4,6,6], [7,7,5], [8,8,6], [4,8,3],[3,5,3]]).
setas4([[3,1,2], [1,2,1], [1,4,1], [7,4,5], [1,6,6], [3,6,5], [4,7,6],[7,1,3], [8,3,4], [9,4,1], [10,4,5], [1,8,6], [9,6,5], [9,7,6]]).
setas5([[2,1,3], [5,3,4], [4,4,1], [3,1,2], [1,2,1], [1,4,1],[4,7,6],[7,1,3], [3,3,4], [9,8,1], [10,9,5], [7,8,6], [9,8,5], [9,8,6]]).
setas6([[2,1,3], [5,3,4], [4,4,1], [7,4,5], [2,1,3], [5,3,4], [4,7,6],[4,4,3], [5,5,4]]).
setas7([[2,4,3], [6,3,4], [6,4,1], [8,1,2], [1,7,1], [1,8,1],[4,7,6],[7,1,3], [3,3,4], [7,8,1], [10,9,5], [7,8,6], [9,8,5], [9,8,6],[2,1,3], [5,3,4], [4,4,1], [3,1,2],[1,2,1], [1,4,1],[10,1,3]]).

setas(1,Arrows):- setas1(Arrows).
setas(2,Arrows):- setas2(Arrows).
setas(3,Arrows):- setas3(Arrows).
setas(4,Arrows):- setas3(Arrows).
setas(5,Arrows):- setas5(Arrows).
setas(6,Arrows):- setas6(Arrows).
setas(7,Arrows):- setas7(Arrows).

% Visualizacao do tabuleiro

espacamento(0).
espacamento(Size1):-
        write(' '),
        NewSize is Size1 -1,
        espacamento(NewSize).

parteCima(0, _) :-
        write('\n').
parteCima(Size, [H|Tail]):-
        NewSize is Size-1,
```

```
        write(H),write(' '),
        parteCima(NewSize, Tail).

verParteCima([], _).
verParteCima([H|Tail], Size1) :-
        length(H, Size),
        Size2 is Size1 - 1,
        espacamento(Size2),
        parteCima(Size, H),
        verParteCima(Tail, Size2).

parteBaixo(0,_):-
        write('\n').
parteBaixo(Size, [H|Tail]):-
        NewSize is Size - 1,
        write(H), write(' '),
        parteBaixo(NewSize, Tail).

verParteBaixo([],_).
verParteBaixo([H|Tail], Size1):-
        length(H, Size),
        Size2 is Size1 + 1,
        espacamento(Size2),
        parteBaixo(Size,H),
        verParteBaixo(Tail, Size2).

divideTab(H, 0, Aux1, H, Aux1).
divideTab([H|Tail], N, List1, List2, Aux1):-
        NewN is N-1,
        divideTab(Tail, NewN, List1, List2, [H|Aux1]).

parteCima(0, Board, Size, SizeTotal) :-
        N is round(Size/2),
        NewSize is N * 2 - 2,
        NewN is N - 1,
        parteBaixo(NewN, Board, NewSize,SizeTotal).

parteCima(N, [Line|Tail], Size, SizeTotal) :-
        NewN is N - 1,
        NewSize is Size + 1,
        length(Line, Size),
        domain(Line, 0, SizeTotal),
        parteCima(NewN, Tail, NewSize,SizeTotal).

parteBaixo(0, _, _,_).
parteBaixo(N, [Line|Tail], Size, SizeTotal) :-
```

```
        NewN is N - 1,
        NewSize is Size - 1,
        length(Line, Size),
        domain(Line, 0, SizeTotal),
        parteBaixo(NewN, Tail, NewSize,SizeTotal).

%cria um tabuleiro hexagonal de lado N
criaTab(N, Board) :-
        Size is N * 2 - 1, %tamanho da linha maior (a do meio)
        length(Board, Size),
        parteCima(N, Board, N, Size).

%a parte de baixo vai ser simetrica da parte de cima
mostraTab(Board):-
        length(Board, Size),
        N is round((Size + 1)/2),
        divideTab(Board, N, List1, List2, Aux1),
        reverse(List1, NewList1),
        verParteCima(NewList1, N),
        verParteBaixo(List2,0).

% restricao
% A soma dos pontos na direcao da seta tem de ser maior do que a soma dos pontos nas
outras direcoes para quais a seta podia virar

% caso a celula seja 0 ou 7 (pontos) coloca numa lista
insere([], Points, Points).
insere([H|Tail], List, Points) :-
        (var(H), H #= 0 #\/ H #= 7, insere(Tail, [H|List], Points)) ;
        insere(Tail, [H|List], Points).

%para obter a lista com os pontos
getPoints([], Points, Points).
getPoints([H|Tail], List, Points) :-
        insere(H, List, NewVars),
        getPoints(Tail, NewVars, Points).


%obter a celula da linha Lin e coluna Col
getPos(Lin, Col, Board, Cell) :-
        nth1(Col, Board, Line),
        nth1(Lin, Line, Cell).
getPos(_, _, _, -1).

%inserir as setas no tabuleiro formado
colocaSetas([], _).
```

```
colocaSetas([[Lin,Col, Dir]|Tail], Board) :-
        getPos(Lin, Col, Board, Cell),
        Cell is Dir,
        colocaSetas(Tail, Board).


%Aplicar a restricao:
%1º compara a direcao da seta
%2º verifica os pontos na direcao da seta, se tiver , adiciona-os a uma lista
%3º se nao, inverte a lista e verifica na direcao contraria, caso nao tenha tambem, vai ver nas diagonais (colunas)
% restricao: a soma dos pontos na direcao da seta tem de ser maior do que nas outras direcoes possiveis

comparaDir(Lin, Col, N, Board, Dir, Dir).
comparaDir(Lin, Col, N, Board, Dir, Dir2) :-
        case(Dir,Lin, Col, N, Board, [], List),
        case(Dir2,Lin, Col, N, Board, [], List2),
        sum(List, #=, Sum),
        sum(List2, #<, Sum).

checkSetas([], _, _).
checkSetas([[Lin,Col, Dir]|Tail], N, Board) :-
        comparaDir(Lin, Col, N, Board, Dir, 1),
        comparaDir(Lin, Col, N, Board, Dir, 2),
        comparaDir(Lin, Col, N, Board, Dir, 3),
        comparaDir(Lin, Col, N, Board, Dir, 4),
        comparaDir(Lin, Col, N, Board, Dir, 5),
        comparaDir(Lin, Col, N, Board, Dir, 6),
        checkSetas(Tail, N, Board).

%funcao principal
pointing(N, Arrows, Board) :-
        criaTab(N, Board),
        colocaSetas(Arrows, Board),
        checkSetas(Arrows, N, Board),
        getPoints(Board, [], Points),
        labeling([], Points).

%Interface      ----------
% Ler caracteres introduzidos pelo utilizador
le(Linha):-
        get_code(Ch),
        leTodos(Ch,[H|T]),
        name(Linha,[H|T]), !.
```

```
leTodos(13,[]).
leTodos(10,[]).

leTodos(Ch,[Ch|Chars]):-
        Ch >= 48,
        Ch =< 57,
        !,
        get_code(NewCh),
        leTodos(NewCh,Chars).

main:-
        nl, write('+--------------------------------+'),
        nl, write('|      POINTING AT THE CROWD     |'),
        nl, write('+--------------------------------+'),
        nl, write('+         Choose wisely:         +'),
        nl, write('+                                +'),
        nl, write('+ 1) Exemplo 1      2) Exemplo 2 +'),
        nl, write('+ 3) Random         3) Sair      +'),
        nl, write('+                                +'),
        nl, write('+--------------------------------+'),nl,
        repeat, le(C),
        analize(C).

analize(1):-
        statistics(walltime, [Start,_]),
        setas1(Arrows),
        pointing(4, Arrows, Board),
        mostraTab(Board),nl,
        write('+--------------------------------+'),nl,
        write('+          Estatisticas          +'),nl,
        write('+--------------------------------+'),nl,
        statistics(walltime, [End,_]) ,
        Time is End - Start ,
        write('Tempo: '),
        write(Time),nl,
        fd_statistics,
        nl, write('+--------------------------------+'),
        nl, write('+ 1) Voltar         0) Sair      +'),
        nl, write('+--------------------------------+'),nl,
        repeat, le(C),
        analize2(C).


analize(2):-
        statistics(walltime, [Start,_]),
        setas2(Arrows),
```

```
        pointing(4, Arrows, Board),
        mostraTab(Board),
        write('+--------------------------------+'),nl,
        write('+          Estatisticas          +'),nl,
        write('+--------------------------------+'),nl,
        statistics(walltime, [End,_]) ,
        Time is End - Start ,
        write('Tempo: '),
        write(Time),nl,
        fd_statistics,
        nl, write('+--------------------------------+'),
        nl, write('+ 1) Voltar        0) Sair      +'),
        nl, write('+--------------------------------+'),nl,
        repeat, le(C),
        analize2(C).

analize(0):- !.
analize2(0):- !.

analize2(1):- !, main.

analize(3):-
        statistics(walltime, [Start,_]),
        write('Lado do hexagono(maior do que 4):'), nl,
        repeat, le(C),C >= 4,
        random(1,8, S), %escolhe aleatoriamente o grupo das setas TODO: tem um bug para os casos de ter um tabuleiro em que o numero maximo de colunas e 7 e coloca-se uma 8, se se voltar a escolher entretanto formara o tabuleiro, mas como e random a escolha, as vezes pode demorar
        %todo: corrigir isto
        setas(S, Arrows),
        pointing(C, Arrows, Board),
        mostraTab(Board),
        write('+--------------------------------+'),nl,
        write('+          Estatisticas          +'),nl,
        write('+--------------------------------+'),nl,
        statistics(walltime, [End,_]) ,
        Time is End - Start ,
        write('Tempo: '),
        write(Time),nl,
        fd_statistics,
        nl, write('+--------------------------------+'),
        nl, write('+ 1) Voltar        0) Sair      +'),
        nl, write('+--------------------------------+'),nl,
        repeat, le(C2),
        analize2(C2).
```