Recorridos de valor mínimo en tableros

Julia Serrano Arrese. C200119

Table of Contents

code

Usage and interface

- **Library usage:**
  use_module('code.pl')
- **Exports:**
  - *Predicates:*
    `mueve/3`, `check_pos/2`, `my_select/3`, `direccion_valida/2`, `my_reverse/2`, `reverse_aux/3`, `obtener_valor_minimo/2`, `obtener_valor_minimo_aux/3`, `recorridos_min/3`, `recorridos_min_aux/4`, `efectuar_movimiento/3`, `movimiento_valido/3`, `select_cell/4`, `select_dir/3`, `aplicar_op/3`, `generar_recorrido/6`, `gen_rec_aux/8`, `generar_recorridos/5`, `tablero/5`.
  - *Properties:*
    `author_data/4`.
  - *Multifiles:*
    `Σcall_in_module/2`.

Documentation on exports

author_data/4: PROPERTY

Usage:

Nombre y matrícula del autor de la práctica.

```
author_data('Serrano','Arrese','Julia','200119').
```

mueve/3: PREDICATE

Usage: `mueve(Dir,Pos,NewPos)`

`NewPos` es la posición resultante de moverse desde `Pos` en la dirección marcada por `Dir`, que puede ser una de las siguientes:

- Norte (n)
- Sur (s)
- Este (e)
- Oeste (o)
- Noroeste (no)
- Noreste (ne)
- Suroeste (so)
- Sureste (se)

```
mueve(n,pos(Row,Col),pos(RowNew,Col)) :-
    RowNew is Row-1.
mueve(s,pos(Row,Col),pos(RowNew,Col)) :-
    RowNew is Row+1.
mueve(e,pos(Row,Col),pos(Row,ColNew)) :-
```

```
    ColNew is Col+1.
mueve(o,pos(Row,Col),pos(Row,ColNew)) :-
    ColNew is Col-1.
mueve(no,pos(Row,Col),pos(RowNew,ColNew)) :-
    RowNew is Row-1,
    ColNew is Col-1.
mueve(ne,pos(Row,Col),pos(RowNew,ColNew)) :-
    RowNew is Row-1,
    ColNew is Col+1.
mueve(so,pos(Row,Col),pos(RowNew,ColNew)) :-
    RowNew is Row+1,
    ColNew is Col-1.
mueve(se,pos(Row,Col),pos(RowNew,ColNew)) :-
    RowNew is Row+1,
    ColNew is Col+1.
```

Other properties:

Test: `mueve(Dir,Pos,NewPos)`

– *If the following properties hold at call time:*

`Dir=n` (`= /2`)

`Pos=pos(2,2)` (`= /2`)

then the following properties should hold upon exit:

`NewPos=pos(1,2)` (`= /2`)

then the following properties should hold globally:

All the calls of the form `mueve(Dir,Pos,NewPos)` do not fail. (`not_fails/1`)

Test: `mueve(Dir,Pos,NewPos)`

– *If the following properties hold at call time:*

`Dir=s` (`= /2`)

`Pos=pos(2,2)` (`= /2`)

then the following properties should hold upon exit:

`NewPos=pos(3,2)` (`= /2`)

then the following properties should hold globally:

All the calls of the form `mueve(Dir,Pos,NewPos)` do not fail. (`not_fails/1`)

check_pos/2: PREDICATE

Usage: `check_pos(Pos,N)`

Verifica si la posición `Pos` es válida en un tablero de `NxN`.

```
check_pos(pos(Row,Col),N) :-
    integer(Row),
    integer(Col),
    integer(N),
    Row>0,
    Col>0,
    Row=<N,
    Col=<N.
```

Other properties:

Test: `check_pos(Pos,N)`

– *If the following properties hold at call time:*
`Pos=pos(2,3)` (= /2)
`N=3` (= /2)
then the following properties should hold upon exit:
`R=yes` (= /2)
then the following properties should hold globally:
All the calls of the form `check_pos(Pos,N)` do not fail. (`not_fails/1`)

Test: `check_pos(Pos,N)`
– *If the following properties hold at call time:*
`Pos=pos(2,3)` (= /2)
`N=2` (= /2)
then the following properties should hold upon exit:
`R=no` (= /2)
then the following properties should hold globally:
Calls of the form `check_pos(Pos,N)` fail. (`fails/1`)

my_select/3: PREDICATE

Usage: `my_select(X,List,NewList)`

Elimina la primera aparición del elemento `X` en la lista `List`, generando la lista resultante `NewList`.

```
my_select(X,[X|Tail],Tail).
my_select(X,[Y|Tail],[Y|NewTail]) :-
    my_select(X,Tail,NewTail).
```

Other properties:

Test: `my_select(X,List,NewList)`
– *If the following properties hold at call time:*
`X=2` (= /2)
`List=[1,2,3]` (= /2)
`NewList=[1,3]` (= /2)
then the following properties should hold upon exit:
`R=yes` (= /2)
`NewList=[1,3]` (= /2)
then the following properties should hold globally:
All the calls of the form `my_select(X,List,NewList)` do not fail. (`not_fails/1`)

Test: `my_select(X,List,NewList)`
– *If the following properties hold at call time:*
`X=4` (= /2)
`List=[1,2,3]` (= /2)
`NewList=[1,2,3]` (= /2)
then the following properties should hold upon exit:
`R=no` (= /2)
then the following properties should hold globally:
Calls of the form `my_select(X,List,NewList)` fail. (`fails/1`)

Test: `my_select(X,List,NewList)`

– *If the following properties hold at call time:*

`X=2` (= /2)

`List=[3,4,5,2]` (= /2)

then the following properties should hold upon exit:

`NewList=[3,4,5]` (= /2)

then the following properties should hold globally:

All the calls of the form `my_select(X,List,NewList)` do not fail. (`not_fails/1`)

direccion_valida/2: PREDICATE

Usage: `direccion_valida(Board,Pos)`

Verifica si la posición `Pos` pertenece al tablero `Board`. `Board` es una lista de celdas representadas como: `cell(pos(Row,Col),op(Operador,Operando))`.

```
direccion_valida(Board,Pos) :-
    member(cell(Pos,_1),Board).
```

Other properties:

Test: `direccion_valida(Board,Pos)`

– *If the following properties hold at call time:*

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(2,2),op(+,20)),cell(pos(3,3),op(//,-23))]` (= /2)

`Pos=pos(1,1)` (= /2)

then the following properties should hold upon exit:

`R=yes` (= /2)

then the following properties should hold globally:

All the calls of the form `direccion_valida(Board,Pos)` do not fail. (`not_fails/1`)

Test: `direccion_valida(Board,Pos)`

– *If the following properties hold at call time:*

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(2,2),op(+,20)),cell(pos(3,3),op(//,-23))]` (= /2)

`Pos=pos(1,2)` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `direccion_valida(Board,Pos)` fail. (`fails/1`)

my_reverse/2: PREDICATE

Usage: `my_reverse(List,Reversed)`

Invierte una lista `List`, devolviendo la lista invertida en `Reversed`.

```
my_reverse(List,Reversed) :-
    reverse_aux(List,[],Reversed).
```

Other properties:

Test: `my_reverse(List,Reversed)`

– *If the following properties hold at call time:*

`List=[1,2,3,4,5]` (`= /2`)

`Reversed=[5,4,3,2,1]` (`= /2`)

then the following properties should hold upon exit:

`R=yes` (`= /2`)

then the following properties should hold globally:

All the calls of the form `my_reverse(List,Reversed)` do not fail. (`not_fails/1`)

Test: `my_reverse(List,Reversed)`

– *If the following properties hold at call time:*

`List=[1,2,3]` (`= /2`)

`Reversed=[2,3,1]` (`= /2`)

then the following properties should hold upon exit:

`R=no` (`= /2`)

then the following properties should hold globally:

Calls of the form `my_reverse(List,Reversed)` fail. (`fails/1`)

Test: `my_reverse(List,Reversed)`

– *If the following properties hold at call time:*

`List=[1,2,3,4,5]` (`= /2`)

then the following properties should hold upon exit:

`Reversed=[5,4,3,2,1]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `my_reverse(List,Reversed)` do not fail. (`not_fails/1`)

reverse_aux/3: PREDICATE

Usage: `reverse_aux(List,Acc,Reversed)`

Predicado auxiliar utilizado por 'my_reverse/2' para realizar la recursividad. Realiza la inversión de una lista mediante recursión, acumulando los elementos invertidos en la variable de acumulación `Acc` y devolviendo la lista invertida final en `Reversed`.

```
reverse_aux([],Acc,Acc).
reverse_aux([H|T],Acc,Reversed) :-
    reverse_aux(T,[H|Acc],Reversed).
```

obtener_valor_minimo/2: PREDICATE

Usage: `obtener_valor_minimo(Recorridos,ValorMinimo)`

Predicado utilizado para encontrar el valor mínimo en una lista de recorridos. Toma una lista de recorridos representados como tuplas de la forma `(Recorrido, Valor)` y devuelve en `ValorMinimo` el valor mínimo encontrado en dicha lista.

```
obtener_valor_minimo([(_Recorrido,Valor)|RestoRecorridos],ValorMinimo) :-
    obtener_valor_minimo_aux(RestoRecorridos,Valor,ValorMinimo).
```

Other properties:

Test: `obtener_valor_minimo(Recorridos,ValorMinimo)`

– *If the following properties hold at call time:*
`Recorridos=[]` (= /2)
`ValorMinimo=0` (= /2)
then the following properties should hold upon exit:
`R=yes` (= /2)
then the following properties should hold globally:
All the calls of the form `obtener_valor_minimo(Recorridos,ValorMinimo)` do not fail. (`not_fails/1`)

Test: `obtener_valor_minimo(Recorridos,ValorMinimo)`
– *If the following properties hold at call time:*
`Recorridos=[(_Recorrido1,10),(_Recorrido2,5),(_Recorrido3,8)]` (= /2)
`ValorMinimo=5` (= /2)
then the following properties should hold upon exit:
`R=yes` (= /2)
then the following properties should hold globally:
All the calls of the form `obtener_valor_minimo(Recorridos,ValorMinimo)` do not fail. (`not_fails/1`)

Test: `obtener_valor_minimo(Recorridos,ValorMinimo)`
– *If the following properties hold at call time:*
`Recorridos=[(_Recorrido1,10),(_Recorrido2,5),(_Recorrido3,8)]` (= /2)
`ValorMinimo=0` (= /2)
then the following properties should hold upon exit:
`R=no` (= /2)
then the following properties should hold globally:
Calls of the form `obtener_valor_minimo(Recorridos,ValorMinimo)` fail. (`fails/1`)

Test: `obtener_valor_minimo(Recorridos,ValorMinimo)`
– *If the following properties hold at call time:*
`Recorridos=[(_Recorrido1,8),(_Recorrido2,-5),(_Recorrido3,0)]` (= /2)
then the following properties should hold upon exit:
`ValorMinimo= -5` (= /2)
then the following properties should hold globally:
All the calls of the form `obtener_valor_minimo(Recorridos,ValorMinimo)` do not fail. (`not_fails/1`)

obtener_valor_minimo_aux/3: PREDICATE

Usage: `obtener_valor_minimo_aux(Recorridos,ValorActual,ValorMinimo)`

Predicado auxiliar utilizado por 'obtener_valor_minimo/2' para encontrar el valor mínimo en una lista de recorridos. Realiza la comparación de valores y la recursividad necesaria para encontrar el valor mínimo.

```
obtener_valor_minimo_aux([],ValorMinimo,ValorMinimo).
obtener_valor_minimo_aux([(_Recorrido,Valor)|RestoRecorridos],ValorActual,ValorM
    Valor<ValorActual,
    obtener_valor_minimo_aux(RestoRecorridos,Valor,ValorMinimo).
obtener_valor_minimo_aux([(_Recorrido,_1)|RestoRecorridos],ValorActual,ValorMini
    obtener_valor_minimo_aux(RestoRecorridos,ValorActual,ValorMinimo).
```

recorridos_min/3: PREDICATE

Usage: `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)`

Cuenta el número de rutas en la lista de `Recorridos` que tienen el `ValorMinimo` especificado. El resultado se unifica con el `NumRecorridosMin`.

```
recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin) :-
    recorridos_min_aux(Recorridos,ValorMinimo,0,NumRecorridosMin).
```

Other properties:

Test: `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)`

– *If the following properties hold at call time:*

`Recorridos=[[(pos(1,1),2),(pos(1,2),0),(pos(2,1),-3)],[(pos(1,1),-3),(pos(1,2),0),(pos(2,1),-6)],[(pos(1,1),5),(pos(1,2),-1),(pos(2,1),5)]]` (`=/2`)

`ValorMinimo= -6` (`=/2`)

`NumRecorridosMin=1` (`=/2`)

then the following properties should hold upon exit:

`R=yes` (`=/2`)

then the following properties should hold globally:

All the calls of the form `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)` do not fail. (`not_fails/1`)

Test: `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)`

– *If the following properties hold at call time:*

`Recorridos=[[(pos(1,1),2),(pos(1,2),0),(pos(2,1),-10)],[(pos(1,1),-3),(pos(1,2),0),(pos(2,1),-6)],[(pos(1,1),5),(pos(1,2),-1),(pos(2,1),-10)]]` (`=/2`)

`ValorMinimo= -10` (`=/2`)

`NumRecorridosMin=2` (`=/2`)

then the following properties should hold upon exit:

`R=yes` (`=/2`)

then the following properties should hold globally:

All the calls of the form `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)` do not fail. (`not_fails/1`)

Test: `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)`

– *If the following properties hold at call time:*

`Recorridos=[[(pos(1,1),2),(pos(1,2),0),(pos(2,1),-10)],[(pos(1,1),-3),(pos(1,2),0),(pos(2,1),-6)],[(pos(1,1),5),(pos(1,2),-1),(pos(2,1),-10)]]` (`=/2`)

`ValorMinimo=0` (`=/2`)

`NumRecorridosMin=2` (`=/2`)

then the following properties should hold upon exit:

`R=no` (`=/2`)

then the following properties should hold globally:

Calls of the form `recorridos_min(Recorridos,ValorMinimo,NumRecorridosMin)` fail. (`fails/1`)

recorridos_min_aux/4: PREDICATE

Usage:
`recorridos_min_aux(Recorridos,ValorMinimo,ContadorActual,NumRecsMin)`

Predicado auxiliar utilizado por 'recorridos_min/4' para contar el úmero de recorridos (`Recorridos`) que tienen el valor mínimo especificado por `ValorMinimo`. Realiza la recursividad sobre la lista de `Recorridos` y lleva el `ContadorActual` de recorridos mínimos encontrados, obteniendo finalmente el número total de recorridos mínimos encontrados unificado en `NumRecsMin`.

```
recorridos_min_aux([],_ValMin,NumRecsMin,NumRecsMin).
recorridos_min_aux([(_Recorrido,Val)|RestoRec],ValMin,ContAct,NumRecsMin) :-
    Val=:=ValMin,
    NuevoCont is ContAct+1,
    recorridos_min_aux(RestoRec,ValMin,NuevoCont,NumRecsMin).
recorridos_min_aux([_Recorrido|RestoRec],ValMin,ContAct,NumRecsMin) :-
    recorridos_min_aux(RestoRec,ValMin,ContAct,NumRecsMin).
```

efectuar_movimiento/3: PREDICATE

Usage: `efectuar_movimiento(Pos,Dir,Pos2)`

Predicado que realiza un movimiento desde la posición `Pos` en la dirección `Dir`, obteniendo la nueva posición `Pos2`.

```
efectuar_movimiento(Pos,Dir,Pos2) :-
    mueve(Dir,Pos,Pos2).
```

Other properties:

Test: `efectuar_movimiento(Pos,Dir,Pos2)`

– *If the following properties hold at call time:*

`Pos=pos(1,1)` (`= /2`)

`Dir=e` (`= /2`)

`Pos2=pos(1,2)` (`= /2`)

then the following properties should hold upon exit:

`R=yes` (`= /2`)

then the following properties should hold globally:

All the calls of the form `efectuar_movimiento(Pos,Dir,Pos2)` do not fail. (`not_fails/1`)

Test: `efectuar_movimiento(Pos,Dir,Pos2)`

– *If the following properties hold at call time:*

`Pos=pos(1,4)` (`= /2`)

`Dir=n` (`= /2`)

`Pos2=pos(0,1)` (`= /2`)

then the following properties should hold upon exit:

`R=no` (`= /2`)

then the following properties should hold globally:

Calls of the form `efectuar_movimiento(Pos,Dir,Pos2)` fail. (`fails/1`)

movimiento_valido/3: PREDICATE

Usage: `movimiento_valido(N,Pos,Dir)`

Predicado que verifica si el movimiento desde la posición `Pos` en la dirección `Dir` es válido en un tablero de tamaño `NxN`.

```
movimiento_valido(N,Pos,Dir) :-
    efectuar_movimiento(Pos,Dir,Pos2),
    check_pos(Pos2,N).
```

Other properties:

Test: `movimiento_valido(N,Pos,Dir)`

– *If the following properties hold at call time:*

`N=3` (`= /2`)

`Pos=pos(1,1)` (`= /2`)

`Dir=e` (`= /2`)

then the following properties should hold upon exit:

`R=yes` (`= /2`)

then the following properties should hold globally:

All the calls of the form `movimiento_valido(N,Pos,Dir)` do not fail. (`not_fails/1`)

Test: `movimiento_valido(N,Pos,Dir)`

– *If the following properties hold at call time:*

`N=3` (`= /2`)

`Pos=pos(1,1)` (`= /2`)

`Dir=n` (`= /2`)

then the following properties should hold upon exit:

`R=no` (`= /2`)

then the following properties should hold globally:

Calls of the form `movimiento_valido(N,Pos,Dir)` fail. (`fails/1`)

select_cell/4: PREDICATE

Usage: `select_cell(IPos,Op,Board,NewBoard)`

Extrae la celda con la posición `IPos` del tablero `Board`, obteniendo `NewBoard` sin dicha celda y unificando `Op` con la operación asociada a la respectiva celda.

```
select_cell(IPos,Op,Board,NewBoard) :-
    my_select(cell(IPos,Op),Board,NewBoard).
```

Other properties:

Test: `select_cell(IPos,Op,Board,NewBoard)`

– *If the following properties hold at call time:*

`IPos=pos(2,2)` (`= /2`)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(2,2),op(+,-6)),cell(pos(3,3),op(//,-23))]` (`= /2`)

then the following properties should hold upon exit:

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(3,3),op(//,-23))]` (`= /2`)

`Op=op(+,-6)` (`= /2`)

then the following properties should hold globally:

All the calls of the form `select_cell(IPos,Op,Board,NewBoard)` do not fail. (`not_fails/1`)

Test: `select_cell(IPos,Op,Board,NewBoard)`

– *If the following properties hold at call time:*

`IPos=pos(4,4)` (`= /2`)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(2,2),op(+,-6)),cell(pos(3,3),op(//,-23))]` (`= /2`)

then the following properties should hold upon exit:

`R=no` (`= /2`)

then the following properties should hold globally:

Calls of the form `select_cell(IPos,Op,Board,NewBoard)` fail. (`fails/1`)

select_dir/3: PREDICATE

Usage: `select_dir(Dir,Dirs,NewDirs)`

Resta una dirección `Dir` de las direcciones permitidas en `Dirs`, obteniendo `NewDirs` con la dirección restada.

```
select_dir(Dir,Dirs,NewDirs) :-
    my_select(dir(Dir,Num),Dirs,DirsWithoutDir),
    ( Num>1 ->
        NewNum is Num-1,
        NewDirs=[dir(Dir,NewNum)|DirsWithoutDir]
    ; NewDirs=DirsWithoutDir
    ).
```

Other properties:

Test: `select_dir(Dir,Dirs,NewDirs)`

– *If the following properties hold at call time:*

`Dir=n` (`= /2`)

`Dirs=[dir(n,3),dir(e,2),dir(s,1),dir(so,4)]` (`= /2`)

then the following properties should hold upon exit:

`NewDirs=[dir(n,2),dir(e,2),dir(s,1),dir(so,4)]+not_fails` (`= /2`)

Test: `select_dir(Dir,Dirs,NewDirs)`

– *If the following properties hold at call time:*

`Dir=s` (`= /2`)

`Dirs=[dir(n,3),dir(e,2),dir(s,1),dir(so,4)]` (`= /2`)

then the following properties should hold upon exit:

`NewDirs=[dir(n,3),dir(e,2),dir(so,4)]+not_fails` (`= /2`)

Test: `select_dir(Dir,Dirs,NewDirs)`

– *If the following properties hold at call time:*

`Dir=s` (`= /2`)

`Dirs=[dir(n,3),dir(e,2),dir(s,1),dir(so,4)]` (`= /2`)

`NewDirs=[dir(n,4),dir(e,2),dir(s,1),dir(so,4)]` (`= /2`)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `select_dir(Dir,Dirs,NewDirs)` fail. (`fails/1`)

aplicar_op/3: PREDICATE

Usage: `aplicar_op(Op,Valor,Valor2)`

Dada una `Op` representada por `op(Operador,Operando)`, se aplica la operación especificada por el operador de la siguiente forma:

- Operando izquierdo -> `Valor`
- Operando derecho -> Operando
- Resultado -> `Valor2`

```
aplicar_op(op(+,Operando),Valor,Valor2) :-
    Valor2 is Valor+Operando.
aplicar_op(op(-,Operando),Valor,Valor2) :-
    Valor2 is Valor-Operando.
aplicar_op(op(*,Operando),Valor,Valor2) :-
    Valor2 is Valor*Operando.
aplicar_op(op(//,Operando),Valor,Valor2) :-
    Operando\=0,
    Valor2 is Valor//Operando.
```

Other properties:

Test: `aplicar_op(Op,Valor,Valor2)`

– *If the following properties hold at call time:*

`Op=op(+,5)` (= /2)

`Valor=10` (= /2)

then the following properties should hold upon exit:

`Valor2=15` (= /2)

then the following properties should hold globally:

All the calls of the form `aplicar_op(Op,Valor,Valor2)` do not fail. (`not_fails/1`)

Test: `aplicar_op(Op,Valor,Valor2)`

– *If the following properties hold at call time:*

`Op=op(+,5)` (= /2)

`Valor=10` (= /2)

`Valor2=7` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `aplicar_op(Op,Valor,Valor2)` fail. (`fails/1`)

Test: `aplicar_op(Op,Valor,Valor2)`

– *If the following properties hold at call time:*

`Op=op(*,3)` (= /2)

`Valor=8` (= /2)

then the following properties should hold upon exit:

`Valor2=24` (= /2)

then the following properties should hold globally:

All the calls of the form `aplicar_op(Op,Valor,Valor2)` do not fail. (`not_fails/1`)

Test: `aplicar_op(Op,Valor,Valor2)`

– *If the following properties hold at call time:*

`Op=op(//,4)` (= /2)

`Valor=27` (= /2)

then the following properties should hold upon exit:

`Valor2=6` (= /2)

then the following properties should hold globally:

All the calls of the form `aplicar_op(Op,Valor,Valor2)` do not fail. (`not_fails/1`)

Test: `aplicar_op(Op,Valor,Valor2)`

– *If the following properties hold at call time:*

`Op=op(//,0)` (= /2)

`Valor=10` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `aplicar_op(Op,Valor,Valor2)` fail. (`fails/1`)

generar_recorrido/6: PREDICATE

Usage: `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)`

Obtiene un `Recorrido` del tablero `Board`, que tiene el tamaño `NxN`, iniciado en la posición `Ipos`, teniendo en cuenta las direcciones permitidas en `DirPerm`, y obteniendo un valor final `Valor`. Realiza la llamada al predicado auxiliar 'gen_rec_aux/8' para generar el recorrido.

```
generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor) :-
    gen_rec_aux(Ipos,N,Board,DirPerm,[],Recorrido,0,Valor).
```

Other properties:

Test: `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)`

– *If the following properties hold at call time:*

`Ipos=pos(1,2)` (= /2)

`N=2` (= /2)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000))]` (= /2)

`DirPerm=[dir(n,5),dir(s,6),dir(e,7),dir(o,4)]` (= /2)

then the following properties should hold upon exit:

`Recorrido=[(pos(1,2),-1),(pos(2,2),1999),(pos(2,1),1996),(pos(1,1),-5988)]` (= /2)

`Valor= -5988` (= /2)

then the following properties should hold globally:

All the calls of the form `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)` do not fail. (`not_fails/1`)

Test: `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)`

– *If the following properties hold at call time:*

`Ipos=pos(4,2)` (= /2)

`N=2` (= /2)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000))]` (= /2)

`DirPerm=[dir(n,5),dir(s,6),dir(e,7),dir(o,4)]` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)` fail. (`fails/1`)

Test: `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)`

– *If the following properties hold at call time:*

`N=3` (= /2)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000))]` (= /2)

`DirPerm=[dir(n,1),dir(s,6),dir(e,7),dir(o,3)]` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `generar_recorrido(Ipos,N,Board,DirPerm,Recorrido,Valor)` fail. (`fails/1`)

gen_rec_aux/8: PREDICATE

Usage: `gen_rec_aux(Pos,N,Board,DirPerm,Visitadas,Recorrido,ValorActual,ValorFinal)`

Predicado recursivo auxiliar utilizado por 'generar_recorrido/6' para generar un `Recorrido` en un `Board` de tamaño `NxN`.

El predicado realiza la recursividad sobre el `Board` y las `DirPerm`, manteniendo una lista de celdas visitadas, `Visitadas`, y el valor actual del recorrido, `ValorActual`.

Al finalizar, unifica el recorrido obtenido en la variable `Recorrido` y el valor final en `ValorFinal`.

```
gen_rec_aux(Pos,_N,[Last],_DirPerm,Visit,Rec,ValorActual,ValorFinal) :-
    direccion_valida([Last],Pos),
    select_cell(Pos,op(Operador,Operando),[Last],_NewBoard),
    aplicar_op(op(Operador,Operando),ValorActual,ValorFinal),
    my_reverse([(Pos,ValorFinal)|Visit],Rec).
gen_rec_aux(Pos,N,Board,DirPerm,Visit,Rec,ValorActual,ValorFinal) :-
    movimiento_valido(N,Pos,Dir),
    direccion_valida(Board,Pos),
    efectuar_movimiento(Pos,Dir,NewP),
    select_dir(Dir,DirPerm,NewDirs),
    select_cell(Pos,op(Operador,Operando),Board,NewB),
    aplicar_op(op(Operador,Operando),ValorActual,NewVal),
    gen_rec_aux(NewP,N,NewB,NewDirs,[(Pos,NewVal)|Visit],Rec,NewVal,ValorFinal)
```

generar_recorridos/5: PREDICATE

Usage: `generar_recorridos(N,Board,DirPerm,Recs,Valor)`

Genera todos los recorridos (`Recs`) posibles en un tablero de tamaño `NxN`. El predicado recibe el tamaño del tablero `N`, el tablero `Board` que contiene las celdas del tablero, la lista de direcciones permitidas `DirPerm`, y unifica los recorridos generados en `Recs` y el valor final en `Valor`. Para generar los recorridos, se obtiene una posición del tablero utilizando el predicado 'member/2' y luego se utiliza el predicado 'generar_recorrido/6' para generar un recorrido iniciando desde esa posición.

```
generar_recorridos(N,Board,DirPerm,Recs,Valor) :-
    member(cell(Pos,_Valor),Board),
    generar_recorrido(Pos,N,Board,DirPerm,Recs,Valor).
```

Other properties:

Test: `generar_recorridos(N,Board,DirPerm,Recs,Valor)`

– *If the following properties hold at call time:*

`N=3` (`= /2`)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000)),cell(pos(2,3),op(-,3)),cell(pos(3,1),op(+,2000)),cell(pos(3,2),op(+,2000)),cell(pos(3,3),op(+,2000)` (`= /2`)

`DirPerm=[dir(n,2),dir(s,2),dir(e,2),dir(o,6)]` (`= /2`)

then the following properties should hold upon exit:

`Recorridos=[(pos(1,3),-1),(pos(2,3),-4),(pos(3,3),1996),(pos(3,2),3996),(pos(3,1),5996),(pos(2,1),5993),(pos(2,2),7993-23976)]` (`= /2`)

`Valor= -23976` (`= /2`)

then the following properties should hold globally:

All the calls of the form `generar_recorridos(N,Board,DirPerm,Recs,Valor)` do not fail. (`not_fails/1`)

Test: `generar_recorridos(N,Board,DirPerm,Recs,Valor)`

– *If the following properties hold at call time:*

`N=3` (`= /2`)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000)),cell(pos(2,3),op(-,3)),cell(pos(3,1),op(+,2000)),cell(pos(3,2),op(+,2000)),cell(pos(3,3),op(+,2000)` (`= /2`)

`DirPerm=[dir(n,2),dir(s,2),dir(e,2),dir(o,6)]` (`= /2`)

`Valor= -23` (`= /2`)

then the following properties should hold upon exit:

`R=no` (`= /2`)

then the following properties should hold globally:

Calls of the form `generar_recorridos(N,Board,DirPerm,Recs,Valor)` fail. (`fails/1`)

Test: `generar_recorridos(N,Board,DirPerm,Recs,Valor)`

– *If the following properties hold at call time:*

`N=2` (= /2)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000)),cell(pos(2,3),op(-,3)),cell(pos(3,1),op(+,2000)),cell(pos(3,2),op(+,2000)),cell(pos(3,3),op(+,2000)` (= /2)

`DirPerm=[dir(n,2),dir(s,2),dir(e,2),dir(o,6)]` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `generar_recorridos(N,Board,DirPerm,Recs,Valor)` fail. (`fails/1`)

Test: `generar_recorridos(N,Board,DirPerm,Recs,Valor)`

– *If the following properties hold at call time:*

`N=3` (= /2)

`Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,1)),cell(pos(2,1),op(-,3)),cell(pos(2,2),op(+,2000)),cell(pos(2,3),op(-,3)),cell(pos(3,1),op(+,2000)),cell(pos(3,2),op(+,2000)),cell(pos(3,3),op(+,2000)` (= /2)

`DirPerm=[dir(so,2),dir(n,3),dir(s,1),dir(no,1),dir(e,2),dir(o,6)]` (= /2)

then the following properties should hold upon exit:

`Recs=[(pos(1,2),-1),(pos(1,3),-2),(pos(2,3),-5),(pos(3,2),1995),(pos(3,3),3995),(pos(2,2),5995),(pos(3,1),7995),(pos(2,1),7992 23976)]` (= /2)

`Valor= -23976` (= /2)

then the following properties should hold globally:

All the calls of the form `generar_recorridos(N,Board,DirPerm,Recs,Valor)` do not fail. (`not_fails/1`)

tablero/5: PREDICATE

Usage: `tablero(N,Tablero,DirPerm,ValorMinimo,NumeroDeRutasConValorMinimo)`

Predicado que utiliza el predicado anterior 'generar_recorridos/5' para generar todos los recorridos posibles del `Tablero` de tamaño `N`, teniendo en cuenta las direcciones permitidas `DirPerm`.

Luego, obtiene el `ValorMinimo` entre todos los recorridos y cuenta el número de rutas que tienen dicho valor. Los resultados se unifican con los argumentos `ValorMinimo` y `NumeroDeRutasConValorMinimo` respectivamente.

```
tablero(N,Tablero,DirPerm,ValorMinimo,NumeroDeRutasConValorMinimo) :-
    findall((Rec,Valor),generar_recorridos(N,Tablero,DirPerm,Rec,Valor),Recs),
    obtener_valor_minimo(Recs,ValorMinimo),
    recorridos_min(Recs,ValorMinimo,NumeroDeRutasConValorMinimo).
```

Other properties:

Test: `tablero(N,Tablero,DirPerm,ValorMinimo,NumeroDeRutasConValorMinimo)`

– *If the following properties hold at call time:*

`N=4` (= /2)

`Tablero=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,4)),cell(pos(1,4),op(-,555)),cell(pos(2,1),op(-,3)),cell(pos(2,4),op(-,444)),cell(pos(3,1),op(*,0)),cell(pos(3,4),op(+,20)),cell(pos(4,1),op(-,2)),cell(pos(4,2),op(-,1000)),cell(pos(4,3),op(-,9)),cell(pos(4,4),op(*,4))]` (= /2)

`DirPerm=[dir(n,2),dir(s,2),dir(e,2),dir(o,6)]` (= /2)

then the following properties should hold upon exit:

`R=no` (= /2)

then the following properties should hold globally:

Calls of the form `tablero(N,Tablero,DirPerm,ValorMinimo,NumeroDeRutasConValorMinimo)` fail. (`fails/1`)

Test: `tablero(N,Tablero,DirPerm,ValorMinimo,NumeroDeRutasConValorMinimo)`

– *If the following properties hold at call time:*

`N=4` (= /2)

`Tablero=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,4)),cell(pos(1,4),op(-,555)),cell(pos(2,1),op(-,3)),cell(pos(2,4),op(-,444)),cell(pos(3,1),op(*,0)),cell(pos(3,4),op(+,20)),cell(pos(4,1),op(-,2)),cell(pos(4,2),op(-,1000)),cell(pos(4,3),op(-,9)),cell(pos(4,4),op(*,4))]` (= /2)

`DirPerm=[dir(n,5),dir(s,6),dir(e,7),dir(o,4)]` (= /2)

then the following properties should hold upon exit:

`ValorMinimo= -5028` (= /2)

`NumeroDeRutasConValorMinimo=1` (= /2)

Documentation on multifiles

Σcall_in_module/2: PREDICATE

No further documentation available for this predicate. The predicate is *multifile*.

Documentation on imports

This module has the following direct dependencies:

– *Application modules:*

`operators`, `dcg_phrase_rt`, `datafacts_rt`, `dynamic_rt`, `classic_predicates`.

– *Internal (engine) modules:*

`term_basic`, `arithmetic`, `atomic_basic`, `basiccontrol`, `exceptions`, `term_compare`, `term_typing`, `debugger_support`, `hiord_rt`, `stream_basic`, `io_basic`, `runtime_control`, `basic_props`.

– *Packages:*

`prelude`, `initial`, `condcomp`, `classic`, `runtime_ops`, `dcg`, `dcg/dcg_phrase`, `dynamic`, `datafacts`, `assertions`, `assertions/assertions_basic`.