Práctica2 Programación declarativa

Miguel Hernando Padilla

26 May 2023

1 Documentación generada por CIAO

TOC

code

Usage and interface

Library usage:
`:- use_module(/home/miguel/Escritorio/Universidad/3ero_Carrera/2do_semestre/programacion_declarativa/CiaoTask/code.pl).`
Exports:
- *Predicates:*
  `author_data/4`, `board1/1`, `board2/1`, `efectuar_movimiento/3`, `movimiento_valido/3`, `aplicar_op/3`, `select_cell/4`, `select_dir/3`, `generar_recorrido/6`, `generar_recorrido_aux/7`, `generar_recorridos/5`, `minimum_value/3`, `tablero/5`.
- *Multifiles:*
  `call_in_module/2`.

Documentation on exports

PREDICATE **author_data/4**

No further documentation available for this predicate.

PREDICATE **board1/1**

No further documentation available for this predicate.

PREDICATE **board2/1**

No further documentation available for this predicate.

PREDICATE **efectuar_movimiento/3**

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

Recibe en `Dir` una dirección, que puede ser norte(n), sur, esto(e), oeste(o) y las combinaciones de estas cuatro direcciones. El predicado se encarga de comprobar que si `Pos` se mueve en la dirección marcada en `Dir`, se mueve a la casilla `Pos2`.

Other properties:

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

- *If the following properties hold at call time:*
  `Pos=pos(2,3)` (`= /2`)
  `Pos2=pos(1,3)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=n` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(3,3)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=s` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(2,4)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=e` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(2,2)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=o` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(1,2)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=no` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(1,4)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=ne` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(3,2)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=so` (`= /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(2,3)` (`= /2`)
  `Pos2=pos(3,4)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=se` (`= /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`)

PREDICATE **movimiento_valido/3**

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

Comprueba que en un tablero de tamaño `N` x `N` , desde la posicion indicada en la variable `Pos` , se pueda mover en la dirección indicada en `Dir` .

Other properties:

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

- *If the following properties hold at call time:*
  `N=3` (`= /2`)
  `Pos=pos(1,1)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=s;Dir=e;Dir=se` (undefined property)
  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,P,Dir)`

Caso 1 efectuar movimientos

- *If the following properties hold at call time:*
  `N=6` (`= /2`)
  `P=pos(2,6)` (`= /2`)
  then the following properties should hold upon exit:
  `Dir=n;Dir=s;Dir=o;Dir=e;Dir=no;Dir=ne;Dir=so;Dir=se` (undefined property)
  then the following properties should hold globally:
  All the calls of the form `movimiento_valido(N,P,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=(1,1)` (`= /2`)
  then the following properties should hold globally:
  Calls of the form `movimiento_valido(N,Pos,Dir)` fail. (`fails/1`)

PREDICATE aplicar_op/3

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

En `Op` se recibe un valor y un operador. En `Valor2` esta el resultado de la operación del primer valor de `Op` y `Valor` , usando el operador recibido en el segundo argumento de `Op` .

Other properties:

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

- *If the following properties hold at call time:*
  `Op=op(+,2)` (`= /2`)
  `Valor=3` (`= /2`)
  then the following properties should hold upon exit:
  `Valor2=5` (`= /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(-,2)` (`= /2`)
  `Valor=3` (`= /2`)
  then the following properties should hold upon exit:
  `Valor2=1` (`= /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(*,2)` (`= /2`)
  `Valor=3` (`= /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(//,2)` (`= /2`)
  `Valor=4` (`= /2`)
  then the following properties should hold upon exit:
  `Valor2=2` (`= /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(//,2)` (= /2)
  `Valor=0` (= /2)
  then the following properties should hold upon exit:
  `Valor2=0` (= /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(//,2)` (= /2)
  `Valor=t` (= /2)
  then the following properties should hold globally:
  Calls of the form `aplicar_op(Op,Valor,Valor2)` fail. (fails/1)

PREDICATE select_cell/4

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

Recorre la lista de celdas `Board` hasta que haya una celda cuya pos y op coincidan con `IPos` y `Op` respectivamente. En la variable `NewBoard` se devuelve la `Board` sin la celda coincidente

Other properties:

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

- *If the following properties hold at call time:*
  `IPos=pos(1,2)` (= /2)
  `Op=op(-,1)` (= /2)
  `Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,4))]` (= /2)
  then the following properties should hold upon exit:
  `NewBoard=[cell(pos(1,1),op(*,-3)),cell(pos(1,3),op(-,4))]` (= /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(1,2)` (= /2)
  `Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,4))]` (= /2)
  then the following properties should hold upon exit:
  `NewBoard=[cell(pos(1,1),op(*,-3)),cell(pos(1,3),op(-,4))]` (= /2)
  `Op=op(-,1)` (= /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(1,2)` (= /2)
  `Op=op(-,2)` (= /2)
  `Board=[cell(pos(1,1),op(*,-3)),cell(pos(1,2),op(-,1)),cell(pos(1,3),op(-,4))]` (= /2)
  then the following properties should hold globally:
  Calls of the form `select_cell(IPos,Op,Board,NewBoard)` fail. (fails/1)

PREDICATE select_dir/3

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

En `Dir` hay una dirección, puede ser n, s, o, e, además de las comninaciones entre estas. En `Dirs` es donde hay una lista de direcciones permitidas, que es una lista de estructuras dir(A,B), con A siendo una dirección y B el numero de veces que se puede ir en esa dirección. En `NewDirs` esta la misma lista, pero para `Dir` un valor menos en el número de movimientos que permiten realizar, o no aparecer si solo podía realizar un movimiento.

Other properties:

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

- *If the following properties hold at call time:*
  `Dir=n` (= /2)

`Dirs=[dir(n,3),dir(s,4),dir(o,2),dir(se,10)]` (`= /2`)

then the following properties should hold upon exit:

`NewDirs=[dir(n,2),dir(s,4),dir(o,2),dir(se,10)]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `select_dir(Dir,Dirs,NewDirs)` do not fail. (`not_fails/1`)

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

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

  `Dirs=[dir(n,3),dir(s,4),dir(o,2),dir(se,10)]` (`= /2`)

  `NewDirs=[dir(n,2),dir(s,4),dir(o,2),dir(se,10)]` (`= /2`)

  then the following properties should hold upon exit:

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

  then the following properties should hold globally:

  All the calls of the form `select_dir(Dir,Dirs,NewDirs)` do not fail. (`not_fails/1`)

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

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

  `Dir=no` (`= /2`)

  `Dirs=[dir(n,3),dir(s,4),dir(o,2),dir(se,10)]` (`= /2`)

  then the following properties should hold globally:

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

PREDICATE **generar_recorrido/6**

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

Este predicado genera un recorrido a partir de la posición indicada en `Ipos` . En el tablero indicado en `Board` , de tamaño `N` , debe recorrerlo entero usando solo direcciones incluidas en `DireccionesPermitidas` , devolver una lista de pares la posición a la que va y el valor del recorrido en ese momento en `Recorrido` , y el valor final en `Valor` . El valor se consigue realizando sobre el valor de cada posición la operación asociada a la celda seleccionada en el tablero.

Other properties:

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

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

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

  `DireccionesPermitidas=[dir(n,3),dir(s,4),dir(o,2),dir(e,10)]` (`= /2`)

  then the following properties should hold upon exit:

  `Recorrido=[(pos(1,1),0),(pos(2,1),-3),(pos(2,2),1997),(pos(1,2),1996)],Valor=1996;Recorrido=[(pos(1,1),0),(pos(1,2),-1),(pos(2,2),1999),(pos(2,1),1996)],Valor=1996` (undefined property)

  then the following properties should hold globally:

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

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

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

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

  `DireccionesPermitidas=[dir(n,3),dir(s,4),dir(o,2),dir(se,10)]` (`= /2`)

  then the following properties should hold globally:

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

PREDICATE **generar_recorrido_aux/7**

Usage: `generar_recorrido_aux(Ipos,N,Board,DireccionesPermitidas,Recorrido,ValorActual,ValorF)`

Igual que generar_recorrido/6, pero en `ValorActual` esta el valor que tiene en cada momento el recorrido

PREDICATE **generar_recorridos/5**

Usage: `generar_recorridos(N,Board,DireccionesPermitidas,Recorrido,Valor)`

`N`, `Board` y `DireccionesPermitidas` son variables que se usan para llamar al predicado generar_recorridos/6. Se encuentran todas sus posibles soluciones con findall y se devuelven en `Recorrido` el recorrido con el menor valor, que se almacena en `Valor`.

Other properties:

Test: `generar_recorridos(N,Board,DireccionesPermitidas,Recorrido,Valor)`

- *If the following properties hold at call time:*
  `N=4` (= /2)
  `Board=` (= /2)
  `[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`
  `DireccionesPermitidas=[dir(n,5),dir(s,6),dir(e,7),dir(o,4)]` (= /2)
  then the following properties should hold upon exit:
  `Recorrido=[(pos(4,1),-2),(pos(3,1),0),(pos(2,1),-3),(pos(2,2),1997),(pos(2,3),265601),(pos(3,3),132800),` (= /2)
  `(pos(3,2),20584000),(pos(4,2),20583000),(pos(4,3),20582991),(pos(4,4),82331964),(pos(3,4),82331984),(pos(2,4),82331540),`
  `(pos(1,4),82330985),(pos(1,3),82330981),(pos(1,2),82330980),(pos(1,1),-246992940)]`
  `Valor= -246992940` (= /2)
  then the following properties should hold globally:
  All the calls of the form `generar_recorridos(N,Board,DireccionesPermitidas,Recorrido,Valor)` do not fail. (`not_fails/1`)

PREDICATE **minimum_value/3**

Usage: `minimum_value(Recorridos,Recorrido,Valor)`

`Recorridos` una lista de pares, donde el primer par es una lista de pares posición-valor. En `Recorrido` se guarda la lista cuyo valor sea más pequeño, y en `Valor` el valor más más pequeño entre todos los que había en `Recorridos`.

PREDICATE **tablero/5**

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

Encuentra el valor mínimo y el número de rutas con ese valor mínimo en `Tablero`, con tamaño `N`, utilizando las `DireccionesPermitidas`, usando generar_recorridos/5. En `ValorMinimo` se devuelve la el valor de la ruta más pequeña, y en `NumeroDeRutasConValorMinimo` cuantas rutas con ese valor hay

Other properties:

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

- *If the following properties hold at call time:*
  `N=4` (= /2)
  `Tablero=` (= /2)
  `[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`
  `DireccionesPermitidas=[dir(n,5),dir(s,6),dir(e,7),dir(o,4)]` (= /2)
  then the following properties should hold upon exit:
  `ValorMinimo= -246992940` (= /2)
  `NumeroDeRutasConValorMinimo=1` (= /2)
  then the following properties should hold globally:
  All the calls of the form `tablero(N,Tablero,DireccionesPermitidas,ValorMinimo,NumeroDeRutasConValorMinimo)` do not fail. (`not_fails/1`)

Documentation on multifiles

PREDICATE **call_in_module/2**

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, regtypes.
```

2 Métodos

2.1 efectuar_movimiento/3

```
:-pred efectuar_movimiento(Pos,Dir,Pos2)
  #"Recibe en @var{Dir} una dirección, que puede ser norte(n), sur,
    esto(e), oeste(o) y las combinaciones de estas cuatro direcciones.
    El predicado se encarga de comprobar que si @var{Pos} se mueve en
    la dirección marcada en @var{Dir}, se mueve a la casilla @var{Pos2}.".

efectuar_movimiento(pos(X1,Y1), n, pos(X2,Y1)):-
    X2 is X1-1.

efectuar_movimiento(pos(X1,Y1), s, pos(X2,Y1)):-
    X2 is X1+1.

efectuar_movimiento(pos(X1,Y1), o, pos(X1,Y2)):-
    Y2 is Y1-1.

efectuar_movimiento(pos(X1,Y1), e,pos(X1,Y2)):-
    Y2 is Y1+1.


efectuar_movimiento(pos(X1,Y1), no, pos(X2,Y2)):-
    efectuar_movimiento(pos(X1,Y1), n, pos(X2,_)),
    efectuar_movimiento(pos(X1,Y1), o,pos(_,Y2)).

efectuar_movimiento(pos(X1,Y1), ne, pos(X2,Y2)):-
    efectuar_movimiento(pos(X1,Y1), n, pos(X2,_)),
    efectuar_movimiento(pos(X1,Y1), e,pos(_,Y2)).

efectuar_movimiento(pos(X1,Y1), so, pos(X2,Y2)):-
    efectuar_movimiento(pos(X1,Y1), s, pos(X2,_)),
    efectuar_movimiento(pos(X1,Y1), o,pos(_,Y2)).

efectuar_movimiento(pos(X1,Y1), se, pos(X2,Y2)):-
    efectuar_movimiento(pos(X1,Y1), s, pos(X2,_)),
    efectuar_movimiento(pos(X1,Y1), e,pos(_,Y2)).

:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(1,3)) => (Dir = n) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(3,3)) => (Dir = s) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(2,4)) => (Dir = e) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(2,2)) => (Dir = o) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(1,2)) => (Dir = no) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(1,4)) => (Dir = ne) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(3,2)) => (Dir = so) + not_fails.
:- test efectuar_movimiento(Pos,Dir,Pos2) : (Pos=pos(2,3),Pos2=pos(3,4)) => (Dir = se) + not_fails.
```

Figure 1: Código del predicado efectuar_movimiento/3.

2.2 movimiento_valido/3

```
:- pred movimiento_valido(N,Pos,Dir)
   #"Comprueba que en un tablero de tamaño @var{N}x@var{N}, desde la posicion indicada
     en la variable @var{Pos}, se pueda mover en la dirección indicada en @var{Dir}.".

movimiento_valido(N,pos(X,Y),n):-
    X > 1,
    Y =< N.

movimiento_valido(N,pos(X,Y),s):-
    X < N,
    Y =< N.

movimiento_valido(N,pos(X,Y),o):-
    Y > 1,
    X =< N.

movimiento_valido(N,pos(X,Y),e):-
    Y < N,
    X =< N.

movimiento_valido(N,pos(X,Y),no):-
    movimiento_valido(N,pos(X,Y),n),
    movimiento_valido(N,pos(X,Y),o).

movimiento_valido(N,pos(X,Y),ne):-
    movimiento_valido(N,pos(X,Y),n),
    movimiento_valido(N,pos(X,Y),e).

movimiento_valido(N,pos(X,Y),so):-
    movimiento_valido(N,pos(X,Y),s),
    movimiento_valido(N,pos(X,Y),o).

movimiento_valido(N,pos(X,Y),se):-
    movimiento_valido(N,pos(X,Y),s),
    movimiento_valido(N,pos(X,Y),e).

:-test movimiento_valido(N,Pos,Dir): (N=3,Pos=pos(1,1)) => (Dir = s; Dir = e; Dir = se) + not_fails.

:- test movimiento_valido(N,P,Dir) : (N = 6, P = pos(2,6) )
   =>(Dir = n;Dir = s;Dir = o;Dir = e;Dir = no;Dir = ne;Dir = so;Dir = se) + not_fails #"Caso 1 efect

:-test movimiento_valido(N,Pos,Dir): (N=3,Pos=(1,1)) + fails.
```

Figure 2: Código del predicado movimiento_valido/3.

2.3 select_cell/4

```
:- pred select_cell(IPos,Op,Board,NewBoard)
  #"Recorre la lista de celdas @var{Board} hasta que haya una celda cuya pos y op coincidan con
    @var{IPos} y @var{Op} respectivamente. En la variable @var{NewBoard} se devuelve la
   @var{Board} sin la celda coincidente".

select_cell(IPos, Op, [cell(IPos,Op)|Board], Board).

select_cell(IPos, Op, [cell(Pos,Ope)|Board], [cell(Pos,Ope)|NewBoard]) :-
    select_cell(IPos, Op, Board, NewBoard).

:- test select_cell(IPos,Op,Board,NewBoard) : (IPos = pos(1,2), Op=op(- ,1),
       Board= [cell(pos(1,1) ,op(*,-3)),cell(pos(1,2),op(-,1)), cell(pos(1,3) ,op(- ,4))])
   => (NewBoard = [cell(pos(1,1) ,op(* ,-3)), cell(pos(1,3) ,op(- ,4))]) + not_fails.

:- test select_cell(IPos,Op,Board,NewBoard) : (IPos = pos(1,2),
        Board= [cell(pos(1,1) ,op(*,-3)),cell(pos(1,2),op(-,1)), cell(pos(1,3) ,op(- ,4))])
   => (NewBoard = [cell(pos(1,1) ,op(* ,-3)), cell(pos(1,3) ,op(- ,4))], Op=op(- ,1)) + not_fails.

:- test select_cell(IPos,Op,Board,NewBoard) : (IPos = pos(1,2), Op=op(- ,2),
        Board= [cell(pos(1,1) ,op(*,-3)),cell(pos(1,2),op(-,1)), cell(pos(1,3) ,op(- ,4))])
   + fails.
```

Figure 3: Código del predicado select_cell/4.

2.4 select_dir/3

```
:- pred select_dir(Dir,Dirs,NewDirs)
   #"En @var{Dir} hay una dirección, puede ser n, s, o, e, además de las comninaciones entre estas. En
     @var{Dirs} es donde hay una lista de direcciones permitidas, que es una lista de estructuras dir(A,B)
     con A siendo una dirección y B el numero de veces que se puede ir en esa dirección.
     En @var{NewDirs} esta la misma lista, pero para @var{Dir} un valor menos en el número de movimientos
     que permiten realizar, o no aparecer si solo podía realizar un movimiento.".

%select_dir(_,[],[]).

%select_dir(Dir, [dir(Dir,X)|Dirs],[dir(Dir,Y)|NewDirs]):-
select_dir(Dir, [dir(Dir,X)|Dirs],[dir(Dir,Y)|Dirs]):-
    X > 1,
    Y is X - 1.

select_dir(Dir, [dir(Dir,1)|Dirs],Dirs).

select_dir(Dir, [N|Dirs],[N|NewDirs]):-
    select_dir(Dir,Dirs,NewDirs).

:- test select_dir(Dir,Dirs,NewDirs) : (Dir=n, Dirs=[dir(n,3), dir(s,4), dir(o,2), dir(se,10)])
   => (NewDirs=[dir(n,2), dir(s,4), dir(o,2), dir(se,10)]) + not_fails.

:- test select_dir(Dir,Dirs,NewDirs) : (Dirs=[dir(n,3), dir(s,4), dir(o,2), dir(se,10)],
        NewDirs=[dir(n,2), dir(s,4), dir(o,2), dir(se,10)])
   => (Dir=n) + not_fails.

:- test select_dir(Dir,Dirs,NewDirs) : (Dir=no, Dirs=[dir(n,3), dir(s,4), dir(o,2), dir(se,10)]) + fails.
```

Figure 4: Código del predicado select_dir/3.

2.5 aplicar_op/3

```
:- pred aplicar_op(Op, Valor, Valor2)
   #"En @var{Op} se recibe un valor y un operador. En @var{Valor2} esta el resultado de la operación
     del primer valor de @var{Op} y @var{Valor}, usando el operador recibido en el segundo argumento
    de @var{Op}.".

aplicar_op(op(+, Op1),Op2, Resultado) :-
    Resultado is Op2 + Op1.

aplicar_op(op(-, Op1),Op2, Resultado) :-
    Resultado is Op2 - Op1.

aplicar_op(op(*,Op1),Op2, Resultado) :-
    Resultado is Op2 * Op1.

aplicar_op(op(//, Op1),0,0).

aplicar_op(op(//, Op1),Op2, Resultado) :-
    Resultado is Op2 //  Op1.

:- test aplicar_op(Op, Valor, Valor2): (Op=op(+,2) ,Valor=3) => (Valor2=5) + not_fails.
:- test aplicar_op(Op, Valor, Valor2): (Op=op(-,2) ,Valor=3) => (Valor2=1) + not_fails.
:- test aplicar_op(Op, Valor, Valor2): (Op=op(*,2) ,Valor=3) => (Valor2=6) + not_fails.
:- test aplicar_op(Op, Valor, Valor2): (Op=op(//,2) ,Valor=4) => (Valor2=2) + not_fails.
:- test aplicar_op(Op, Valor, Valor2): (Op=op(//,2) ,Valor=0) => (Valor2=0) + not_fails.
:- test aplicar_op(Op, Valor, Valor2): (Op=op(//,2) ,Valor=t) + fails.
```

Figure 5: Código del predicado aplicar_op/3.

2.6 generar_recorrido/6

```
:- pred generar_recorrido(Ipos,N,Board,DireccionesPermitidas,Recorrido,Valor)
   #" Este predicado genera un recorrido a partir de la posición indicada en @var{Ipos}.
      En el tablero indicado en @var{Board}, de tamaño @var{N}, debe recorrerlo entero usando
      solo direcciones incluidas en @var{DireccionesPermitidas}, devolver una lista de
      pares la posición a la que va y el valor del recorrido en ese momento en @var{Recorrido},
      y el valor final en @var{Valor}. El valor se consigue realizando sobre el valor de cada
      posición la operación asociada a la celda seleccionada en el tablero. ".

generar_recorrido(Ipos,N,Board,DireccionesPermitidas,Recorrido,Valor):-
    generar_recorrido_aux(Ipos,N,Board,DireccionesPermitidas,Recorrido,0,Valor).

generar_recorrido_aux(Ipos,_,[cell(Ipos,Op)],_,[(Ipos,ValorF)],ValorActual,ValorF):-
    select_cell(Ipos,Op,Board,[]), %sacas la celda que has comprobado
    aplicar_op(Op,ValorActual,ValorF).

generar_recorrido_aux(Ipos,N,Board,DireccionesPermitidas,[(Ipos,V)|Recorrido],ValorActual,ValorF):-
    select_cell(Ipos,Op,Board,NewBoard), %sacas la celda que has comprobado
    aplicar_op(Op,ValorActual,V), %se comprueba que el valor actual realizando la operación que toca sea
    movimiento_valido(N,Ipos,Dir),%Comprueba que se pueda realizar el movimiento pedido
    efectuar_movimiento(Ipos,Dir,Pos2), %dirección a la que tiene que ir para el siguiente movimiento
    select_dir(Dir,DireccionesPermitidas,NewDireccionesPermitidas), %saca la dirección utilizada
    generar_recorrido_aux(Pos2,N,NewBoard,NewDireccionesPermitidas,Recorrido,V,ValorF).

:- pred generar_recorrido_aux(Ipos,N,Board,DireccionesPermitidas,Recorrido,ValorActual,ValorF)
   #"Igual que generar_recorrido/6, pero en @var{ValorActual} esta el valor que tiene en cada
     momento el recorrido".

:- test generar_recorrido(Ipos,N,Board,DireccionesPermitidas,Recorrido,Valor) :
   (Ipos=pos(1,1),N=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(+
    DireccionesPermitidas=[dir(n,3), dir(s,4), dir(o,2), dir(e,10)])
   => (Recorrido=[(pos(1,1),0),(pos(2,1),-3),(pos(2,2),1997),(pos(1,2),1996)], Valor=1996;
       Recorrido=[(pos(1,1),0),(pos(1,2),-1),(pos(2,2),1999),(pos(2,1),1996)], Valor=1996) + not_fails.

:- test generar_recorrido(Ipos,N,Board,DireccionesPermitidas,Recorrido,Valor) :
   (Ipos=pos(1,1),N=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(+
    DireccionesPermitidas=[dir(n,3), dir(s,4), dir(o,2), dir(se,10)]) + fails.
```

Figure 6: Código del predicado generar_recorrido/6.

2.7 AUXILIAR: minimum_value/3

```
:-pred minimum_value(Recorridos, Recorrido, Valor)
  #"@var{Recorridos} una lista de pares, donde el primer par es una lista de pares posición-valor.
    En @var{Recorrido} se guarda la lista cuyo valor sea más pequeño, y en @var{Valor} el valor más
    más pequeño entre todos los que había en @var{Recorridos}.".

minimum_value([(R, V)], R, V).
minimum_value([(R1, V1), (R2, V2)|Recorridos], MinRecorrido, MinValor) :-
    V1 < V2,
    minimum_value([(R1, V1)|Recorridos], MinRecorrido, MinValor).
minimum_value([(R1, V1), (R2, V2)|Recorridos], MinRecorrido, MinValor) :-
    V2 =< V1,
    minimum_value([(R2, V2)|Recorridos], MinRecorrido, MinValor).
```

Figure 7: Código del predicado minimum_value/3.

2.8 generar_recorridos/5

```
:-pred generar_recorridos(N, Board, DireccionesPermitidas, Recorrido, Valor)
  #"@var{N}, @var{Board} y @var{DireccionesPermitidas} son variables que se usan para llamar al predicado
    generar_recorridos/6. Se encuentran todas sus posibles soluciones con findall y se devuelven en
    @var{Recorrido} el recorrido con el menor valor, que se almacena en @var{Valor}.".

generar_recorridos(N, Board, DireccionesPermitidas, Recorrido, Valor) :-
    findall((R, V),generar_recorrido(_,N, Board, DireccionesPermitidas,  R, V), Recorridos),
    minimum_value(Recorridos, Recorrido, Valor).

:- test generar_recorridos(N, Board, DireccionesPermitidas, Recorrido, Valor) :
  (N=4,Board=[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(2,2) ,op(+ ,2000)),
        cell(pos(2,3) ,op(* ,133)),
        cell(pos(2,4) ,op(- ,444)),
        cell(pos(3,1) ,op(* ,0)),
        cell(pos(3,2) ,op(* ,155)),
        cell(pos(3,3) ,op(// ,2)),
        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))],
    DireccionesPermitidas = [dir(n,5),dir(s,6),dir(e,7),dir(o,4)])
   => (Recorrido = [
        (pos(4, 1), -2),
        (pos(3, 1), 0),
        (pos(2, 1), -3),
        (pos(2, 2), 1997),
        (pos(2, 3), 265601),
        (pos(3, 3), 132800),
        (pos(3, 2), 20584000),
        (pos(4, 2), 20583000),
        (pos(4, 3), 20582991),
        (pos(4, 4), 82331964),
        (pos(3, 4), 82331984),
        (pos(2, 4), 82331540),
        (pos(1, 4), 82330985),
        (pos(1, 3), 82330981),
        (pos(1, 2), 82330980),
        (pos(1, 1), -246992940)
    ],
    Valor = -246992940)
   + not_fails.
```

Figure 8: Código del predicado generar_recorridos/5.

2.9 tablero/5

```
:-pred tablero(N, Tablero, DireccionesPermitidas, ValorMinimo, NumeroDeRutasConValorMinimo)
  #"Encuentra el valor mínimo y el número de rutas con ese valor mínimo en @var{Tablero}, con tamaño @var{
    utilizando las @var{DireccionesPermitidas}, usando generar_recorridos/5. En @var{ValorMinimo} se devue
    el valor de la ruta más pequeña, y en @var{NumeroDeRutasConValorMinimo} cuantas rutas con ese valor ha
tablero(N, Tablero, DireccionesPermitidas, ValorMinimo, NumeroDeRutasConValorMinimo) :-
    findall((Recorrido, Valor), generar_recorridos(N, Tablero, DireccionesPermitidas, Recorrido, Valor), R
    minimum_value(Recorridos,_, ValorMinimo),
    findall(Recorrido, member((Recorrido, ValorMinimo), Recorridos), RutasMinimas),
    length(RutasMinimas, NumeroDeRutasConValorMinimo).

:- test tablero(N, Tablero, DireccionesPermitidas, ValorMinimo, NumeroDeRutasConValorMinimo) :
   (N=4,
    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(2,2) ,op(+ ,2000)),
         cell(pos(2,3) ,op(* ,133)),
         cell(pos(2,4) ,op(- ,444)),
         cell(pos(3,1) ,op(* ,0)),
         cell(pos(3,2) ,op(* ,155)),
         cell(pos(3,3) ,op(// ,2)),
         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))],
    DireccionesPermitidas=[dir(n,5),dir(s,6),dir(e,7),dir(o,4)])
   => ( ValorMinimo = -246992940,
       NumeroDeRutasConValorMinimo=1) + not_fails.
```

Figure 9: Código del predicado tablero/5.