UTF-8

Autómatas Celulares Reversibles

Alejandro Moreno Ferreiro, A180413

Table of Contents

Documentación acerca de los predicados realizados.

0.1 Usage and interface

- **Library usage:**
  `:- use_module(/home/amf/prolog/code.pl).`
- **Exports:**
  - *Predicates:*
    `apply_rule/3`, `own_append/3`, `cells/3`, `first/2`, `ultimo/2`, `evol/3`, `steps/2`, `ruleset/2`.

0.2 Documentation on exports

apply_rule/3: PREDICATE

Usage: `apply_rule(List,Rule,Res)`

Predicado auxiliar que aplica las reglas proporcionadas dado un estado inicial y genera una lista con el estado resultante de aplicar la regla sucesivamente. Argumentos: [X,Y,Z|R]: Lista con el estado inicial con al menos 3 elementos. Rule: Regla a aplicar. [Res|Resto]: Lista con el resultado, el primer elemento viene de aplicar una regla y el resto de hacer la llamada recusriva con la lista del primer argumento, quitando el primer elemento. La recursividad acaba cuando la lista tiene 2 o menos elementos devolviendo como resultado en el tercer argumento la lista vaca.

```
apply_rule([X,Y,Z|R],Rule,[Res|Resto]) :-
    rule(X,Y,Z,Rule,Res),
    apply_rule([Y,Z|R],Rule,Resto).
apply_rule([],_1,[]).
apply_rule([_2],_1,[]).
apply_rule([_2,_3],_1,[]).
```

Other properties:

Test: `apply_rule(List,Rule,Res)`

Test apply_rule1

- *If the following properties hold at call time:*
  `List=[o,x,x,x,o,o,o,x,o,o,x,x,o,x,x,x,x,o,x,x,o]` (`= /2`)
  `Rule=r(x,x,o,x,o,x,o)` (`= /2`)
  then the following properties should hold upon exit:
  `Res=[x,o,o,x,o,o,x,x,o,x,o,x,x,o,o,o,x,x,o]` (`= /2`)
  then the following properties should hold globally:
  All the calls of the form `apply_rule(List,Rule,Res)` do not fail. (`not_fails/1`)

Test: `apply_rule(List,Rule,Res)`

Test apply_rule2

- *If the following properties hold at call time:*
  `List=[o,x,o,o,o,x,o]` (`= /2`)
  `Rule=r(x,x,o,x,o,o,x)` (`= /2`)
  then the following properties should hold upon exit:

`Res=[x,x,o,o,x]` (= /2)

then the following properties should hold globally:

All the calls of the form `apply_rule(List,Rule,Res)` do not fail. (`not_fails/1`)

Test: `apply_rule(List,Rule,Res)`

Test apply_rule3

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

`List=[o,x,o]` (= /2)

`Rule=r(x,x,x,o,o,x,o)` (= /2)

then the following properties should hold upon exit:

`Res=[x]` (= /2)

then the following properties should hold globally:

All the calls of the form `apply_rule(List,Rule,Res)` do not fail. (`not_fails/1`)

own_append/3: PREDICATE

Usage: `own_append(List,List,Res)`

Implementación propia del append, que concatena dos listas.

```
own_append([],W,W).
own_append([X|V],W,[X|R]) :-
    own_append(V,W,R).
```

cells/3: PREDICATE

Usage: `cells(List,Rule,Result)`

Predicado cells, Se utiliza el own_append para introducir un 0 al principio de la lista y un 0 al final cuando tengas el resultado. Comprueba que el primer y el último elemento de la lista es 0 y aplica la regla para generar el estado final. Argumentos: L: Lista con el estado inicial. Rule: Regla a aplicar. [o|R]: Lista resultado de aplicar la regla con el predicado aply_rule.

```
cells(L,Rule,[o|R]) :-
    first(L,o),
    ultimo(L,o),
    own_append(L,[o],LL),
    apply_rule([o|LL],Rule,RR),
    own_append(RR,[o],R).
```

Other properties:

Test: `cells(List,Rule,Result)`

Test cells1

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

`List=[o,x,o]` (= /2)

`Rule=r(x,x,o,x,x,o,x)` (= /2)

then the following properties should hold upon exit:

`Res=[o,o,x,x,o]` (= /2)

then the following properties should hold globally:

All the calls of the form `cells(List,Rule,Result)` do not fail. (`not_fails/1`)

Test: `cells(List,Rule,Result)`

Test cells2

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

`List=[o,x,o,o,o,o,x,x,x,o,o,x,o,x,o]` (`= /2`)

`Rule=r(x,o,o,x,x,x,o)` (`= /2`)

then the following properties should hold upon exit:

`Res=[o,o,o,x,o,o,o,x,o,x,x,o,o,x,o,x,o]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `cells(List,Rule,Result)` do not fail. (`not_fails/1`)

Test: `cells(List,Rule,Result)`

Test cells3

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

`List=[o,x,x,x,o,o,o,x,o,o,x,x,o,x,x,x,x,o,x,x,o]` (`= /2`)

`Rule=r(x,x,o,x,o,o,x)` (`= /2`)

then the following properties should hold upon exit:

`Res=[o,o,o,x,o,x,o,o,x,x,o,o,o,x,o,x,x,o,x,o,o,x,o]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `cells(List,Rule,Result)` do not fail. (`not_fails/1`)

first/2: PREDICATE

Usage: `first(List,N)`

Compara el primer valor de una lista dada.

```
first([X|_1],X).
```

ultimo/2: PREDICATE

Usage: `ultimo(List,N)`

Compara el último valor de la lista

```
ultimo([X],X).
ultimo([_1|L],X) :-
    ultimo(L,X).
```

evol/3: PREDICATE

Usage: `evol(Sucesor,Rule,EF)`

Aplica N paso de evolucion desde un estado dado usando el predicado cells sucesivamente. Argumentos: Numero de pasos. Regla a aplicar. Estado final que se alcanza.

```
evol(0,_1,[o,x,o]).
evol(s(N),Rule,Cells) :-
    evol(N,Rule,EI),
    cells(EI,Rule,Cells).
```

Other properties:

Test: `evol(Sucesor,Rule,EF)`

Test evol1

– *If the following properties hold at call time:*
`Sucesor=s(s(s(s(s(0))))))` (= /2)
`Rule=r(x,x,o,x,x,o,x)` (= /2)
then the following properties should hold upon exit:
`EF=[o,o,o,o,o,o,o,o,o,o,o,o,x,x,o]` (= /2)
then the following properties should hold globally:
All the calls of the form `evol(Sucesor,Rule,EF)` do not fail. (`not_fails/1`)

Test: `evol(Sucesor,Rule,EF)`

Test evol2

– *If the following properties hold at call time:*
`Sucesor=s(s(s(s(s(0))))))` (= /2)
`Rule=r(x,o,o,x,x,x,o)` (= /2)
then the following properties should hold upon exit:
`EF=[o,o,o,o,o,o,o,o,o,o,o,o,o,x,o]` (= /2)
then the following properties should hold globally:
All the calls of the form `evol(Sucesor,Rule,EF)` do not fail. (`not_fails/1`)

Test: `evol(Sucesor,Rule,EF)`

Test evol3

– *If the following properties hold at call time:*
`Sucesor=s(0)` (= /2)
`Rule=r(o,x,o,x,x,x,o)` (= /2)
then the following properties should hold upon exit:
`EF=[o,o,x,o,o]` (= /2)
then the following properties should hold globally:
All the calls of the form `evol(Sucesor,Rule,EF)` do not fail. (`not_fails/1`)

steps/2: PREDICATE

Usage: `steps(List,Sucesor)`

Nmero de pasos necesarios para llegar un estado a partir de una configuración inicial de tres élulas. Como en cada paso se aaden dos células al estado anterior, se construye la lista de 3,5,7 miembros. Argumentos: Estado que se alcanza tras N pasos. Nmero de pasos.

```
steps([_1,_2,_3],0).
steps([_1,_2|L],s(N)) :-
    steps(L,N).
steps(Goal,_1) :-
    call(Goal).
```

Other properties:

Test: `steps(List,Sucesor)`

Test steps1

– *If the following properties hold at call time:*
`List=[o,o,o,o,o,o,o,o,o,o,o,o,o,x,x,o]` (= /2)
`Sucesor=N` (= /2)
then the following properties should hold upon exit:

`N=s(s(s(s(s(0))))))` (= /2)

then the following properties should hold globally:

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

Test: `steps(List,Sucesor)`

Test steps2

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

`List=[o,x,x,x,o,o,o,x,o,o,x,x,o,x,x,x,x,o,x,x,o]` (= /2)

`Sucesor=N` (= /2)

then the following properties should hold upon exit:

`N=s(s(s(s(s(s(s(s(0))))))))` (= /2)

then the following properties should hold globally:

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

Test: `steps(List,Sucesor)`

Test steps3

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

`List=[o,o,x,x,o]` (= /2)

`Sucesor=N` (= /2)

then the following properties should hold upon exit:

`N=s(0)` (= /2)

then the following properties should hold globally:

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

ruleset/2: PREDICATE

Usage: `ruleset(Rule,FS)`

Descubre si un estado final es alcanzable y con que regla. Argumentos: Regla a aplicar. Estado final.

```
ruleset(Rule,FS) :-
    steps(FS,N),
    evol(N,Rule,FS).
```

Other properties:

Test: `ruleset(Rule,FS)`

Test ruleset1

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

`Rule=RuleSet` (= /2)

`FS=[o,x,x,o,o,x,o,o,o,o,x,o,o,x,o]` (= /2)

then the following properties should hold upon exit:

`RuleSet=r(x,x,x,o,o,x,o)` (= /2)

then the following properties should hold globally:

All the calls of the form `ruleset(Rule,FS)` do not fail. (`not_fails/1`)

Test: `ruleset(Rule,FS)`

Test ruleset2

– *If the following properties hold at call time:*
`Rule=RuleSet` (`= /2`)
`FS=[o,x,x,o,o,x,o,o,o,o,x,o,x,x,o]` (`= /2`)
then the following properties should hold globally:
Calls of the form `ruleset(Rule,FS)` fail. (`fails/1`)

0.3 Documentation on imports

This module has the following direct dependencies:

– *Internal (engine) modules:*
`term_basic`, `arithmetic`, `atomic_basic`, `basiccontrol`, `exceptions`, `term_compare`, `term_typing`, `debugger_support`, `basic_props`.

– *Packages:*
`prelude`, `initial`, `condcomp`, `assertions`, `assertions/assertions_basic`, `regtypes`.