UTF-8

Practica 1 - Programacion logica pura

ALVARO CABO CIUDAD

Table of Contents

0.1 Predicados auxiliares 1
0.1.1 Color 1
0.1.2 Reglas 1
0.2 Testing 1
0.2.1 Running the tests 1
0.2.2 cells/3 tests 2
0.2.3 evol/3 tests 2
0.3 Usage and interface 3
0.4 Documentation on exports 3
author_data/4 (pred) 3
color/1 (pred) 3
rule/5 (pred) 3
cells/3 (pred) 3
evol/3 (pred) 3
steps/2 (pred) 4
ruleset/2 (pred) 4
0.5 Documentation on imports 4

Modelado de un autmata celular 1D en Ciao-Prolog

0.1 Predicados auxiliares

0.1.1 Color

Stablishes valid colors/states for the cells

```
color(o).
color(x).
```

0.1.2 Reglas

Defines valid rules and returns the result of the iteration

```
Example usage:
  ?- R = r(x,o,x,x,x,x,o), rule(o,x,o,R,Y).
  R = r(x,o,x,x,x,x,o),
  Y = o.
```

0.2 Testing

Included at the end of the document all the `:- test` assertions.

Note: Every unit test uses `r(x,x,o,x,o,x,o)` as its ruleset, just as Deliverit

0.2.1 Running the tests

Testing can be run using the ciao console or using the integrated Ciao debugger on Emacs

```
Example of usage:
?-  use_module(library(unittest)).
    yes
?- run_tests_in_module('/home/varo/UPM/3ero/ProDec/Pr_1/code.pl').

PASSED: (lns 123-126) cells/3.
PASSED: (lns 127-128) cells/3.
PASSED: (lns 129-129) cells/3.
PASSED: (lns 130-130) cells/3.
PASSED: (lns 131-135) evol/3.
PASSED: (lns 136-136) evol/3.
FAILED: (lns 137-139) steps/2.
(lns 137-139) steps(_1,_) run-time check failure
Requires in *success*:
    _1=[_,_,_,_,_,_].
But instead:
    _1=[_,o,o,o,o,o]
    _=_2
    _=_1
    _=_3
    _=_4
    _=_5
```

```
    FAILED: (lns 140-140) steps/2.
    (lns 140-140) steps(_1,_) run-time check failure.
    Requires in *success*:
        _1=[_,_,_,_,_,_,_,_].
    But instead:
        _1=[_,o,o,o,o,o,o,o]
        _=_2
        _=_1
        _=_3
        _=_4
        _=_5
        _=_6
        _=_7

Note: {Total:
Passed: 6 (75.00'%') Failed: 2 (25.00'%') Precond Failed: 0 (0.00'%') Aborted: 0 (0.0
}

yes
?-
```

0.2.2 cells/3 tests

Test basico

```
:- test cells(I,R,F) : (I = [o,x,o], R=r(x,x,o,x,o,x,o)) => (C=[o,o,x,x,o])
```

Test largo

```
:- test cells(I,R,F) : (I = [o,x,x,x,o,o,o,x,o,o,x,x,o,x,x,x,x,o,x,x,o],
                        R=r(x,x,o,x,o,x,o))
                        => (C=[o,o,x,o,o,x,o,o,x,x,o,x,o,x,x,o,o,o,x,x,o,x,o])
```

Test inverso

```
:- test cells(I,R,F) : (C = [o,o,x,x,o,o,o,x,o,o,x,o,x,x,x,x,o], R=r(x,x,o,x,o,x,o))
```

Test inferencia de ruleset

```
:- test cells(I,R,F) : (I = [o,x,o,o,o,o,x,x,x,o,o,x,o,x,o],
    F= [o,o,x,x,o,o,o,x,o,o,x,o,x,x,x,x,o]) => R=r(x,x,o,x,o,x,o)
```

0.2.3 evol/3 tests

Test basico

```
:- test evol(N,R,C) : (N = 0, R=r(x,x,o,x,o,x,o)) => (C=[o,x,o])
```

0.3 Usage and interface

- **Library usage:**
  `:- use_module(/home/varo/UPM/3ero/ProDec/Pr_1/code.pl).`
- **Exports:**
  - *Predicates:*
    `author_data/4`, `color/1`, `rule/5`, `cells/3`, `evol/3`, `steps/2`, `ruleset/2`.

0.4 Documentation on exports

author_data/4: PREDICATE

No further documentation available for this predicate.

color/1: PREDICATE

Usage: `color(X)`

Binary representation where `X` is either x or o.

```
color(o).
color(x).
```

rule/5: PREDICATE

Usage: `rule(+Cell1,+Cell2,+Cell3,+Rules,-ResultCell)`

This predicate is used to consult a specific rule given by the `Rules` list and the pattern of `Cell1`, `Cell2`, and `Cell3` cells. It returns the `ResultCell` that corresponds to the pattern of cells based on the rules in the `Rules` list.

cells/3: PREDICATE

Usage: `cells(+InitialState,+Rules,-FinalState)`

Verifies whether `InitialState` is a valid list of cells that can be evolved according to the given `Rules`. If so, the predicate binds `FinalState` to the resulting evolved state.

```
cells([o,X|Rest],Rules,[o,S|FinalState]) :-
    rule(o,o,X,Rules,S),
    evolve([o,X|Rest],Rules,FinalState).
```

evol/3: PREDICATE

Usage: `evol(+N,+Rules,-Cells)`

Aplies `N` steps of the evolution starting at `[o,x,o]`

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

steps/2: PREDICATE

Usage: `steps(+Cells,-N)`

Returns the `N` steps necessary to get from the intial state `[o,x,o]` to `Cells`

ruleset/2: PREDICATE

Usage: `ruleset(RuleSet,Cells)`

Returns valid `Cells` using `RuleSet` starting at the intial state `[o,x,o]` to

0.5 Documentation on imports

This module has the following direct dependencies:

– *Application modules:*

`unittest`.

– *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`.