UTF-8

Practica 1 - Programacion logica pura

ALVARO CABO CIUDAD

Table of Contents

Modelado de un autmata celular 1D en Ciao-Prolog

0.1 Predicados auxiliares dados

0.1.1 Color

Establece los valores validos del color de la clula con color/1:

0.1.2 Reglas

Establece el conjunto de reglas posibles para crear el automata:

```
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 Tests

Se incluyen aserciones que empiezan por `:- test` al final del documento.

0.2.1 basic_building(X) tests

```
:- test basic_building(X) : (X = [[1,1],[s(0)]] )

:- test basic_building(X) : (X = [[s(0),1],[s(0)]] )

:- test basic_building(X) : (X = [[],[s(0)]])

:- test basic_building(X) : (X = [])
```

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`.

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.

```
rule(o,o,o,_1,o).
rule(x,o,o,r(A,_1,_2,_3,_4,_5,_6),A) :-
    color(A).
rule(o,x,o,r(_1,B,_2,_3,_4,_5,_6),B) :-
    color(B).
rule(o,o,x,r(_1,_2,C,_3,_4,_5,_6),C) :-
    color(C).
rule(x,o,x,r(_1,_2,_3,D,_4,_5,_6),D) :-
    color(D).
rule(x,x,o,r(_1,_2,_3,_4,E,_5,_6),E) :-
    color(E).
rule(o,x,x,r(_1,_2,_3,_4,_5,F,_6),F) :-
    color(F).
rule(x,x,x,r(_1,_2,_3,_4,_5,_6,G),G) :-
    color(G).
```

cells/3: PREDICATE

Usage: `cells(::(InitialState,in),::(Rules,in),::(FinalState,out))`

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.

evol/3: PREDICATE

Usage: `evol(N,Rules,Evolution)`

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

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`.