codigo

Usage and interface

Library usage:
`:- use_module(/run/media/dani/SSD Toshiba/OneDrive/Estudios/GII 3º Curso/6º Semestre/PDLR - Programación Declarativa Logica-Restricciones/codigo/P2/codigo.pl).`
Exports:

- *Predicates:*
  `comprimir/2`, `memo/2`, `limpia_memo/0`, `compresion_recursiva/2`, `mejor_compresion_memo/2`, `partir/3`, `parentesis/3`, `repeat_list/3`, `se_repite/4`, `repeticion/2`, `division/2`, `compresion/2`, `mejor_compresion/2`, `choose_smallest/2`, `choose_smallest/3`.
- *Properties:*
  `alumno_prode/4`.
- *Multifiles:*
  `call_in_module/2`.

Documentation on exports

PROPERTY alumno_prode/4

Usage:

```
alumno_prode('Tomas','Sanchez','Daniel',a180428).
```

PREDICATE comprimir/2

Usage: `comprimir(Initial,Compressed)`

`Compressed` is the compressed version with the shortest length of the list `Initial`. The compression process uses memoization. First of all, clean all previous results memoizated.

```
comprimir(Initial,Compressed) :-
    limpia_memo,
    compresion_recursiva(Initial,Compressed).
```

PREDICATE memo/2

No further documentation available for this predicate. The predicate is of type *dynamic*.

PREDICATE limpia_memo/0

Usage:

Clean all results previously memoizated.

```
limpia_memo :-
    retractall(memo(_1,_2)).
```

PREDICATE compresion_recursiva/2

Usage: `compresion_recursiva(Initial,Compressed)`

`Compressed` is the compressed version with the shortest length of the list `Initial`. The compression process uses memoization.

```
compresion_recursiva(Initial,Compressed) :-
    mejor_compresion_memo(Initial,Compressed).
```

PREDICATE mejor_compresion_memo/2

Usage: `mejor_compresion_memo(Initial,Compressed)`

`Compressed` is the compressed version with the shortest length of the list `Initial`. The compression process uses memoization.

```
mejor_compresion_memo(Initial,Compressed) :-
    memo(Initial,Compressed),
    !.
mejor_compresion_memo(Initial,Compressed) :-
    mejor_compresion(Initial,Compressed),
    assert(memo(Initial,Compressed)).
```

PREDICATE partir/3

Usage: `partir(Whole,Part1st,Part2nd)`

`Part1st` and `Part2nd` are non empty subsequences of the list `Whole`.

```
partir(Whole,Part1st,Part2nd) :-
    append(Part1st,Part2nd,Whole),
    Part1st\=[],
    Part2nd\=[].
```

PREDICATE parentesis/3

Usage: `parentesis(Part,Ocur,R)`

`R` is the result of appending `Ocur` to `Part`. If `Part` has two elements or more `Part` is surrounded by brackets.

```
parentesis([X,Y|Part],Ocur,R) :-
    integer(Ocur),
    append(['('],[X,Y|Part],L1),
    append(L1,[')',Ocur],R),
    !.
parentesis([X],Ocur,R) :-
    integer(Ocur),
    append([X],[Ocur],R).
```

PREDICATE repeat_list/3

Usage: `repeat_list(L,N,R)`

`R` is the result of repeating `N` times the list `L`.

```
repeat_list(_1,0,[]).
repeat_list(L,N,R) :-
    N>0,
    N1 is N-1,
    repeat_list(L,N1,R1),
    append(L,R1,R).
```

PREDICATE se_repite/4

Usage: `se_repite(Cs,Part,N0,Num)`

`Num` is the result of adding *N* to `N0`, being *N* the number of repetitions of `Part` which form `Cs` .

```
se_repite([],_1,_2,0) :- !.
se_repite(Cs,Part,N0,Num) :-
    length(Cs,CsN),
    length(Part,PartN),
    0 is CsN mod PartN,
    Reps is CsN//PartN,
    repeat_list(Part,Reps,Cs),
    Num is N0+Reps.
```

PREDICATE repeticion/2

Usage: `repeticion(Initial,Compressed)`

`Compressed` is the compressed by repetition version of the list `Initial`.

```
repeticion(Initial,Compressed) :-
    partir(Initial,Part1st,_1),
    se_repite(Initial,Part1st,0,R),
    compresion_recursiva(Part1st,Comp),
    parentesis(Comp,R,Compressed).
```

PREDICATE division/2

Usage: `division(Initial,Compressed)`

`Compressed` is the compressed by division version of the list `Initial`.

```
division(Initial,Compressed) :-
    partir(Initial,Part1st,Part2nd),
    compresion_recursiva(Part1st,Comp1st),
    compresion_recursiva(Part2nd,Comp2nd),
    append(Comp1st,Comp2nd,Compressed),
    Initial\=Compressed.
```

PREDICATE compresion/2

Usage: `compresion(Initial,Compressed)`

`Compressed` is the compressed version of the list `Initial`.

```
compresion(Initial,Compressed) :-
    repeticion(Initial,Compressed),
    length(Initial,InitN),
    length(Compressed,CompN),
    CompN<InitN.
compresion(Initial,Compressed) :-
    division(Initial,Compressed),
    length(Initial,InitN),
    length(Compressed,CompN),
    CompN<InitN.
compresion(Initial,Initial).
```

PREDICATE mejor_compresion/2

Usage: `mejor_compresion(Initial,Compressed)`

`Compressed` is the compressed version with the shortest length of the list `Initial`.

```
mejor_compresion(Initial,Compressed) :-
    findall(Comp,compresion(Initial,Comp),L),
    L\=[],
    choose_smallest(L,Compressed),
    !.
mejor_compresion(Initial,Initial).
```

PREDICATE choose_smallest/2

Usage: `choose_smallest(L,Compressed)`

`Compressed` is the sublist with the shortest length of the list `L`.

```
choose_smallest(L,Compressed) :-
    choose_smallest(L,_1,Compressed).
```

PREDICATE **choose_smallest/3**

Usage: `choose_smallest(L,SmCompN,SmComp)`

`SmComp` is the sublist with length `SmCompN` and the shortest length of the list `L`.

```
choose_smallest([],10**10,[]).
choose_smallest(L,SmCompN,SmComp) :-
    L=[H|T],
    length(H,HN),
    choose_smallest(T,SmCompAuxN,SmCompAux),
    ( HN<SmCompAuxN ->
        SmComp=H,
        SmCompN=HN
    ; SmComp=SmCompAux,
      SmCompN=SmCompAuxN
    ).
```

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`.