Sumas de pares de listas y cuadrados

Xiao Peng Ye, A180321

Table of Contents

codigo

Usage and interface

- **Library usage:**
  use_module('codigo.pl')
- **Exports:**
  - *Predicates:*
    `alumno_prode/4`, `sum/3`, `mul/3`, `append/3`, `length/2`, `split_lists/3`, `sumlist_matrix/2`, `less/2`, `nums/2`, `sumlist/2`, `choose_one/3`, `perm/2`, `split/3`, `sumlists/4`, `square_lists/3`.
  - *Properties:*
    `nat/1`.

Documentation on exports

alumno_prode/4: PREDICATE

No further documentation available for this predicate.

nat/1: PROPERTY

Usage:

Número natural.

```
nat(0).
nat(s(X)) :-
    nat(X).
```

sum/3: PREDICATE

Usage: `sum(A,B,C)`

`C` es el resultado de aplicar la suma de los valores `A` y `B` en formato Peano.

```
sum(0,X,X) :-
    nat(X).
sum(s(X),Y,s(Z)) :-
    sum(X,Y,Z).
```

Other properties:

Test: `sum(A,B,C)`

Caso base

- *If the following properties hold at call time:*
  `A=0` (`=/2`)
  `B=s(0)` (`=/2`)
  then the following properties should hold upon exit:

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

then the following properties should hold globally:

All the calls of the form `sum(A,B,C)` do not fail. (`not_fails/1`)

Test: `sum(A,B,C)`

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

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

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

then the following properties should hold upon exit:

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

then the following properties should hold globally:

All the calls of the form `sum(A,B,C)` do not fail. (`not_fails/1`)

mul/3: PREDICATE

Usage: `mul(X,Y,Z)`

`Z` es el resultado de multiplicar los números Peano `X` y `Y`.

```
mul(0,X,0) :-
    nat(X).
mul(s(X),Y,Z1) :-
    mul(X,Y,Z),
    sum(Y,Z,Z1).
```

Other properties:

Test: `mul(X,Y,Z)`

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

`X=0` (= /2)

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

then the following properties should hold upon exit:

`Z=0` (= /2)

then the following properties should hold globally:

All the calls of the form `mul(X,Y,Z)` do not fail. (`not_fails/1`)

Test: `mul(X,Y,Z)`

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

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

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

then the following properties should hold upon exit:

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

then the following properties should hold globally:

All the calls of the form `mul(X,Y,Z)` do not fail. (`not_fails/1`)

append/3: PREDICATE

Usage: `append(L1,L2,L3)`

`L3` es la lista resultado de concatenar las listas `L1` y `L2`.

```
append([],L,L).
append([X|L1],L2,[X|L3]) :-
    append(L1,L2,L3).
```

Other properties:

Test: `append(L1,L2,L3)`

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

`L1=[b,c]` (`= /2`)

`L2=[a]` (`= /2`)

then the following properties should hold upon exit:

`L3=[b,c,a]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `append(L1,L2,L3)` do not fail. (`not_fails/1`)

Test: `append(L1,L2,L3)`

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

`L1=[[a,b]]` (`= /2`)

`L2=[[c],[d]]` (`= /2`)

then the following properties should hold upon exit:

`L3=[[a,b],[c],[d]]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `append(L1,L2,L3)` do not fail. (`not_fails/1`)

length/2: PREDICATE

Usage: `length(L,N)`

`N` es la logitud de la lista `L`.

```
length([],0).
length([_1|L],s(N)) :-
    length(L,N).
```

Other properties:

Test: `length(L,N)`

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

`L=[a,b,c,d]` (`= /2`)

then the following properties should hold upon exit:

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

then the following properties should hold globally:

All the calls of the form `length(L,N)` do not fail. (`not_fails/1`)

split_lists/3: PREDICATE

Usage: `split_lists(N,L,M)`

`M` es la matriz resultante de dividir la lista `L` en sublistas de `N` elementos.

```
split_lists(_1,[],[]).
split_lists(N,L,[X|M]) :-
    length(X,N),
    append(X,L1,L),
    split_lists(N,L1,M).
```

Other properties:

Test: `split_lists(N,L,M)`

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

`L=[a,b,c,d]` (`= /2`)

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

then the following properties should hold upon exit:

`M=[[a,b],[c,d]]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `split_lists(N,L,M)` do not fail. (`not_fails/1`)

sumlist_matrix/2: PREDICATE

No further documentation available for this predicate.

less/2: PREDICATE

Usage: `less(X,Y)`

`X` y `Y` son números peanos tal que `X` es menor o igual uqe `Y`.

```
less(0,X) :-
    nat(X).
less(s(X),s(Y)) :-
    less(X,Y).
```

Other properties:

Test: `less(X,Y)`

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

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

`Y=0` (`= /2`)

then the following properties should hold globally:

Calls of the form `less(X,Y)` fail. (`fails/1`)

Test: `less(X,Y)`

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

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

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

then the following properties should hold globally:

All the calls of the form `less(X,Y)` do not fail. (`not_fails/1`)

nums/2: PREDICATE

Usage: `nums(N,L)`

`L` es la lista de números naturales en orden descendente de `N` a 1).

```
nums(0,[]).
nums(s(N),[s(N)|L]) :-
    nums(N,L).
```

Other properties:

Test: `nums(N,L)`

– *If the following properties hold at call time:*
`L=[0,s(0)]` (`= /2`)
then the following properties should hold globally:
Calls of the form `nums(N,L)` fail. (`fails/1`)

Test: `nums(N,L)`

– *If the following properties hold at call time:*
`N=s(s(s(0)))` (`= /2`)
then the following properties should hold upon exit:
`L=[s(s(s(0))),s(s(0)),s(0)]` (`= /2`)
then the following properties should hold globally:
All the calls of the form `nums(N,L)` do not fail. (`not_fails/1`)

sumlist/2: PREDICATE

Usage: `sumlist(L,S)`

`S` es la suma de todos los números de la lista `L`.

```
sumlist([],0).
sumlist([X|L],Z) :-
    sumlist(L,S),
    sum(X,S,Z).
```

Other properties:

Test: `sumlist(L,S)`

– *If the following properties hold at call time:*
`S=s(s(0))` (`= /2`)
`L=[s(s(0))]` (`= /2`)
then the following properties should hold globally:
All the calls of the form `sumlist(L,S)` do not fail. (`not_fails/1`)

Test: `sumlist(L,S)`

– *If the following properties hold at call time:*
`L=[s(0),s(0),s(s(0))]` (`= /2`)
then the following properties should hold upon exit:
`S=s(s(s(s(0))))` (`= /2`)
then the following properties should hold globally:
All the calls of the form `sumlist(L,S)` do not fail. (`not_fails/1`)

choose_one/3: PREDICATE

Usage: `choose_one(E,L,R)`

Dada una lista `L` devuelve un elemento `E` siendo `R` la lista de los restos de elementos.

```
choose_one(E,[E|R],R).
choose_one(E,[X|L],[X|R]) :-
    choose_one(E,L,R).
```

Other properties:

Test: `choose_one(E,L,R)`

– *If the following properties hold at call time:*
`L=[a,b]` (= /2)
`E=a` (= /2)
`R=[b]` (= /2)
then the following properties should hold globally:
All the calls of the form `choose_one(E,L,R)` do not fail. (`not_fails/1`)

Test: `choose_one(E,L,R)`
– *If the following properties hold at call time:*
`L=[a,b,c,d,e,f,g]` (= /2)
`E=b` (= /2)
`R=[a,c,d,e,f,g]` (= /2)
then the following properties should hold globally:
All the calls of the form `choose_one(E,L,R)` do not fail. (`not_fails/1`)

perm/2: PREDICATE

Usage: `perm(L,LP)`

`LP` es una permutación de los elementos de la lista `L`.

```
perm([],[]).
perm(L,[E|LP]) :-
    choose_one(E,L,R),
    perm(R,LP).
```

Other properties:

Test: `perm(L,LP)`
– *If the following properties hold at call time:*
`L=[a]` (= /2)
then the following properties should hold upon exit:
`LP=[a]` (= /2)
then the following properties should hold globally:
All the calls of the form `perm(L,LP)` do not fail. (`not_fails/1`)

Test: `perm(L,LP)`
– *If the following properties hold at call time:*
`L=[a,b,c,d]` (= /2)
`LP=[b,c,a,d]` (= /2)
then the following properties should hold globally:
All the calls of the form `perm(L,LP)` do not fail. (`not_fails/1`)

split/3: PREDICATE

Usage: `split(L,L1,L2)`

`L` es una lista de longitud `N`, `N` es par, `L1` contiene los `N`/2 elementos en posicin impar de `L` y `L2` los en posición par.

```
split([],[],[]).
split([X,Y|L],[X|L1],[Y|L2]) :-
    split(L,L1,L2).
```

Other properties:

Test: `split(L,L1,L2)`

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

`L=[a,b,c,d]` (= /2)

then the following properties should hold upon exit:

`L1=[a,c]` (= /2)

`L2=[b,d]` (= /2)

then the following properties should hold globally:

All the calls of the form `split(L,L1,L2)` do not fail. (`not_fails/1`)

Test: `split(L,L1,L2)`

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

`L1=[f,g]` (= /2)

`L2=[a,i]` (= /2)

then the following properties should hold upon exit:

`L=[f,a,g,i]` (= /2)

then the following properties should hold globally:

All the calls of the form `split(L,L1,L2)` do not fail. (`not_fails/1`)

sumlists/4: PREDICATE

Usage: `sumlists(N,L1,L2,S)`

`N` es par, `L1` y `L2` son dos listas de longitud `N`/2, que contienen entre ellas todos los números de Peano de 1 a `N`, y `L1` y `L2` suman lo mismo. `S` debe ser el valor de dicha suma.

```
sumlists(0,[],[],0).
sumlists(N,L1,L2,S) :-
    nums(N,L),
    perm(L,LP),
    split(LP,L1,L2),
    sumlist(L1,S),
    sumlist(L2,S).
```

Other properties:

Test: `sumlists(N,L1,L2,S)`

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

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

then the following properties should hold globally:

Calls of the form `sumlists(N,L1,L2,S)` fail. (`fails/1`)

Test: `sumlists(N,L1,L2,S)`

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

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

`L1=[s(0),s(s(s(s(0))))]` (= /2)

`L2=[s(s(0)),s(s(s(0)))]` (= /2)

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

then the following properties should hold globally:

All the calls of the form `sumlists(N,L1,L2,S)` do not fail. (`not_fails/1`)

square_lists/3: PREDICATE

Usage: `square_lists(N,SQ,S)`

`N` es el número, `SQ` la matriz de `N` listas con `N` elementos (números de peanos desde 1 a `N` al cuadrado sin repetir) cada una y `S` el valor que suma cada fila de la matriz `SQ` y que sean iguales.

```
square_lists(N,SQ,S) :-
    less(s(0),N),
    mul(N,N,N2),
    nums(N2,L),
    perm(L,LP),
    split_lists(N,LP,SQ),
    sumlist_matrix(SQ,S).
```

Other properties:

Test: `square_lists(N,SQ,S)`

N no puede ser 0

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

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

then the following properties should hold globally:

Calls of the form `square_lists(N,SQ,S)` fail. (`fails/1`)

Test: `square_lists(N,SQ,S)`

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

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

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

`SQ=[[s(s(s(s(0)))),s(0)],[s(s(s(0))),s(s(0))]]` (`= /2`)

then the following properties should hold globally:

All the calls of the form `square_lists(N,SQ,S)` do not fail. (`not_fails/1`)

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`.

References

(this section is empty)