varlistp

List of variables protocol.

author:
`Paulo Moura`
version:
`1.0`
date:
`2009/4/29`

compilation:
`static, context_switching_calls`

(no dependencies on other files)

Public interface

append/3

Appends two lists.

compilation:
`static`

template:
`append(List1,List2,List)`

mode - number of solutions:
`append(?list,?list,?list) - zero_or_more`

delete/3

Deletes from a list all ocurrences of an element returning the list of remaining elements.

compilation:
`static`

template:
`delete(List,Element,Remaining)`

mode - number of solutions:
`delete(@list,@term,?list) - one`

empty/1

True if the argument is an empty list.

compilation:
`static`

template:
`empty(List)`

mode - number of solutions:
`empty(@list) - zero_or_one`

flatten/2

Flattens a list of lists into a list.

compilation:
`static`

template:
```
flatten(List,Flatted)
```
mode - number of solutions:
```
flatten(@list,-list) - one
```

last/2

List last element (if it exists).

compilation:
```
static
```
template:
```
last(List,Last)
```
mode - number of solutions:
```
last(@list,@var) - zero_or_one
```

length/2

List length.

compilation:
```
static
```
template:
```
length(List,Length)
```
mode - number of solutions:
```
length(@list,?integer) - zero_or_one
```

memberchk/2

Checks if a variable is a member of a list.

compilation:
```
static
```
template:
```
memberchk(Element,List)
```
mode - number of solutions:
```
memberchk(@var,@list) - zero_or_one
```

nextto/3

X and Y are consecutive elements in List.

compilation:
```
static
```
template:
```
nextto(X,Y,List)
```
mode - number of solutions:
```
nextto(@var,@var,?list) - zero_or_more
```

nth0/3

Nth element of a list (counting from zero).

compilation:
```
static
```
template:
```
nth0(Nth,List,Element)
```

mode - number of solutions:
```
nth0(?integer,+list,@var) - zero_or_more
```

nth0/4

Nth element of a list (counting from zero).

compilation:
```
static
```
template:
```
nth0(Nth,List,Element,Residue)
```
mode - number of solutions:
```
nth0(?integer,+list,@var,?list) - zero_or_more
```

nth1/3

Nth element of a list (counting from one).

compilation:
```
static
```
template:
```
nth1(Nth,List,Element)
```
mode - number of solutions:
```
nth1(?integer,+list,@var) - zero_or_more
```

nth1/4

Nth element of a list (counting from zero).

compilation:
```
static
```
template:
```
nth1(Nth,List,Element,Residue)
```
mode - number of solutions:
```
nth1(?integer,+list,@var,?list) - zero_or_more
```

permutation/2

The two lists are a permutation of the same list.

compilation:
```
static
```
template:
```
permutation(List,Permutation)
```
mode - number of solutions:
```
permutation(@list,@list) - zero_or_one
```

prefix/2

Prefix is a prefix of List.

compilation:
```
static
```
template:
```
prefix(Prefix,List)
```
mode - number of solutions:
```
prefix(?list,@list) - zero_or_more
```

reverse/2

Reverses a list.

compilation:
```
static
```
template:
```
reverse(List,Reversed)
```
mode - number of solutions:
```
reverse(@list,?list) - zero_or_one
reverse(?list,@list) - zero_or_one
reverse(-list,-list) - one_or_more
```

same_length/2

The two lists have the same length.

compilation:
```
static
```
template:
```
same_length(List1,List2)
```
mode - number of solutions:
```
same_length(@list,?list) - zero_or_one
same_length(?list,@list) - zero_or_one
same_length(-list,-list) - one_or_more
```

select/3

Selects an element from a list, returning the list of remaining elements.

compilation:
```
static
```
template:
```
select(Element,List,Remaining)
```
mode - number of solutions:
```
select(@var,?list,?list) - zero_or_more
```

sublist/2

The first list is a sublist of the second.

compilation:
```
static
```
template:
```
sublist(Sublist,List)
```
mode - number of solutions:
```
sublist(?list,@list) - zero_or_more
```

subtract/3

Removes all elements in the second list from the first list, returning the list of remaining elements.

compilation:
```
static
```
template:
```
subtract(List,Elements,Remaining)
```
mode - number of solutions:
```
subtract(@list,@list,-list) - one
```

suffix/2

Suffix is a suffix of List.

compilation:
`static`

template:
`suffix(Suffix,List)`

mode - number of solutions:
`suffix(?list,@list) - zero_or_more`

Protected interface

(none)

Private predicates

(none)