listp

List protocol.

author:
    `Paulo Moura`
version:
    `1.7`
date:
    `2011/5/14`

compilation:
    `static, context_switching_calls`

(no dependencies on other files)

Public interface

append/2

Appends all lists in a list of lists.

compilation:
    `static`

template:
    `append(Lists,Concatenation)`

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

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 occurrences 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`

delete_matches/3

Deletes all matching elements from a list, returning the list of remaining elements.

compilation:
    `static`

template:
```
    delete_matches(List,Element,Remaining)
```
mode - number of solutions:
```
    delete_matches(@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
```

keysort/2

Sorts a list of key-value pairs in ascending order.

compilation:
```
    static
```
template:
```
    keysort(List,Sorted)
```
mode - number of solutions:
```
    keysort(+list,-list) - one
```

last/2

List last element (if it exists).

compilation:
```
    static
```
template:
```
    last(List,Last)
```
mode - number of solutions:
```
    last(?list,?term) - zero_or_more
```

length/2

List length.

compilation:
```
    static
```
template:
```
    length(List,Length)
```

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

max/2

Determines the list maximum value using standard order. Fails if the list is empty.

compilation:
```
static
```
template:
```
max(List,Maximum)
```
mode - number of solutions:
```
max(+list,-term) - zero_or_one
```

member/2

Element is a list member.

compilation:
```
static
```
template:
```
member(Element,List)
```
mode - number of solutions:
```
member(?term,?list) - zero_or_more
```

memberchk/2

Checks if a term is a member of a list.

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

min/2

Determines the minimum value in a list using standard order. Fails if the list is empty.

compilation:
```
static
```
template:
```
min(List,Minimum)
```
mode - number of solutions:
```
min(+list,-term) - zero_or_one
```

msort/2

Sorts a list in ascending order (duplicated elements are not removed).

compilation:
```
static
```
template:
```
msort(List,Sorted)
```
mode - number of solutions:
```
msort(+list,-list) - one
```

msort/3

Sorts a list using a user-specified comparison predicate modeled on the standard compare/3 predicate (duplicated elements are not removed).

compilation:
`static`

template:
`msort(Closure,List,Sorted)`

meta-predicate template:
`msort(3,*,*)`

mode - number of solutions:
`msort(+callable,+list,-list) - one`

nextto/3

X and Y are consecutive elements in List.

compilation:
`static`

template:
`nextto(X,Y,List)`

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

partition/5

Partitions a list in lists with values less, equal, and greater than a given value (using standard order).

compilation:
`static`

template:
`partition(List,Value,Less,Equal,Greater)`

mode - number of solutions:
`partition(+list,+number,-list,-list,-list) - one`

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_more`

prefix/2

Prefix is a prefix of List.

compilation:
`static`

template:
`prefix(Prefix,List)`

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

proper_prefix/2

Prefix is a proper prefix of List.

compilation:
`static`

template:
`proper_prefix(Prefix,List)`

mode - number of solutions:
`proper_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
```

same_length/3

The two lists have the same length.

compilation:

```
static
```

template:

```
same_length(List1,List2,Length)
```

mode - number of solutions:

```
same_length(+list,?list,?integer) - zero_or_one
same_length(?list,+list,?integer) - zero_or_one
same_length(-list,-list,-integer) - 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(?term,?list,?list) - zero_or_more
```

selectchk/3

Checks that an element can be selected from a list, returning the list of remaining elements.

compilation:

```
static
```

template:

```
selectchk(Element,List,Remaining)
```

mode - number of solutions:

```
selectchk(?term,?list,?list) - zero_or_one
```

select/4

Selects an element from a list, replacing it by a new element and returning the resulting list.

compilation:
`static`

template:
`select(Old,OldList,New,NewList)`

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

selectchk/4

Checks that an element from a list can be replaced by a new element, returning the resulting list.

compilation:
`static`

template:
`selectchk(Old,OldList,New,NewList)`

mode - number of solutions:
`selectchk(?term,?list,?term,?list) - zero_or_one`

sort/2

Sorts a list in ascending order (duplicated elements are removed).

compilation:
`static`

template:
`sort(List,Sorted)`

mode - number of solutions:
`sort(+list,-list) - one`

sort/3

Sorts a list using a user-specified comparison predicate modeled on the standard compare/3 predicate (duplicated elements are removed).

compilation:
`static`

template:
`sort(Closure,List,Sorted)`

meta-predicate template:
`sort(3,*,*)`

mode - number of solutions:
`sort(+callable,+list,-list) - one`

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`

subsequence/3

List is an interleaving of Subsequence and Remaining. Element order is preserved.

compilation:
`static`

template:
`subsequence(List,Subsequence,Remaining)`

mode - number of solutions:
`subsequence(?list,?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`

proper_suffix/2

Suffix is a proper suffix of List.

compilation:
`static`

template:
`proper_suffix(Suffix,List)`

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

Protected interface

(none)

Private predicates

(none)