setp

Set protocol.

author:
`Paulo Moura`
version:
`1.3`
date:
`2011/2/16`

compilation:
`static, context_switching_calls`

(no dependencies on other files)

Public interface

delete/3

Deletes an element from a set returning the set of remaining elements.

compilation:
`static`

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

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

disjoint/2

True if the two sets have no element in common.

compilation:
`static`

template:
`disjoint(Set1,Set2)`

mode - number of solutions:
`disjoint(+set,+set) - zero_or_one`

equal/2

True if the two sets are equal.

compilation:
`static`

template:
`equal(Set1,Set2)`

mode - number of solutions:
`equal(+set,+set) - zero_or_one`

empty/1

True if the set is empty.

compilation:
`static`

template:
```
empty(Set)
```
mode - number of solutions:
```
empty(+set) - zero_or_one
```

insert/3

Inserts an element in a set, returning the resulting set.

compilation:
```
static
```
template:
```
insert(In,Element,Out)
```
mode - number of solutions:
```
insert(+set,+term,?set) - one
```

insert_all/3

Inserts a list of elements in a set, returning the resulting set.

compilation:
```
static
```
template:
```
insert_all(List,In,Out)
```
mode - number of solutions:
```
insert_all(+list,+set,?set) - one
```

intersect/2

True if the two sets have at least one element in common.

compilation:
```
static
```
template:
```
intersect(Set1,Set2)
```
mode - number of solutions:
```
intersect(+set,+set) - zero_or_one
```

intersection/3

Returns the intersection of Set1 and Set2.

compilation:
```
static
```
template:
```
intersection(Set1,Set2,Intersection)
```
mode - number of solutions:
```
intersection(+set,+set,?set) - zero_or_one
```

intersection/4

True if Intersection is the intersection of Set1 and Set2 and Difference is the difference between Set2 and Set1.

compilation:
```
static
```
template:
```
intersection(Set1,Set2,Intersection,Difference)
```

mode - number of solutions:

```
intersection(+set,+set,?set,?set) - zero_or_one
```

length/2

Number of set elements.

compilation:

```
static
```

template:

```
length(Set,Length)
```

mode - number of solutions:

```
length(+set,?integer) - zero_or_one
```

member/2

Element is a member of set Set.

compilation:

```
static
```

template:

```
member(Element,Set)
```

mode - number of solutions:

```
member(+term,+set) - zero_or_one
member(-term,+set) - zero_or_more
```

memberchk/2

Checks if a term is a member of a set.

compilation:

```
static
```

template:

```
memberchk(Element,Set)
```

mode - number of solutions:

```
memberchk(+term,+set) - zero_or_one
```

powerset/2

Returns the power set of a set, represented as a list of sets.

compilation:

```
static
```

template:

```
powerset(Set,Powerset)
```

mode - number of solutions:

```
powerset(+set,-list) - one
```

product/3

Returns the cartesian product of two sets.

compilation:

```
static
```

template:

```
product(Set1,Set2,Product)
```

mode - number of solutions:

```
product(+set,+set,-set) - one
```

select/3

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

compilation:
`static`

template:
`select(Element,Set,Remaining)`

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

selectchk/3

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

compilation:
`static`

template:
`selectchk(Element,Set,Remaining)`

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

subset/2

True if Subset is a subset of Set.

compilation:
`static`

template:
`subset(Subset,Set)`

mode - number of solutions:
`subset(+set,+set) - zero_or_one`

subtract/3

True when Difference contains all and only the elements of Set1 which are not also in Set2.

compilation:
`static`

template:
`subtract(Set1,Set2,Difference)`

mode - number of solutions:
`subtract(+set,+set,?set) - zero_or_one`

symdiff/3

True if Difference is the symmetric difference of Set1 and Set2.

compilation:
`static`

template:
`symdiff(Set1,Set2,Difference)`

mode - number of solutions:
`symdiff(+set,+set,?set) - zero_or_one`

union/3

True if Union is the union of Set1 and Set2.

compilation:
`static`

template:
`union(Set1,Set2,Union)`

mode - number of solutions:
`union(+set,+set,?set) - zero_or_one`

union/4

True if Union is the union of Set1 and Set2 and Difference is the difference between Set2 and Set1.

compilation:
`static`

template:
`union(Set1,Set2,Union,Difference)`

mode - number of solutions:
`union(+set,+set,?set,?set) - zero_or_one`

Protected interface

(none)

Private predicates

(none)