

AIT Semantic and Declarative Technologies Course

Class practice: Prolog data structures

In an integer list an element is a local maximum (or local minimum) if both of its neighbors are smaller (or bigger) than the element itself. If an element is a local maximum or a local minimum, we call it local optimum.

1. Checking for local optimum

In this task we call a list a zigzag list, if all elements of the list (except for the first and last) are local optima.

```
% zigzag(+L): true, if L is a zigzag list

| ?- zigzag([1,2]).
yes
| ?- zigzag([1,3,2,4,3,5,4,6]).
yes
| ?- zigzag([1,3,2,4,8,5,4,6]).
no
```

2. Counting the local optima in a list

```
% zigzag_count(+L, ?N): true, if in the integer list L there are N local optima

| ?- zigzag_count([1,2], N).
N = 0 ? ; no
| ?- zigzag_count([1,3,2,4,3,5,4,6], N).
N = 6 ? ; no
| ?- zigzag_count([1,3,2,4,8,5,4,6], 6).
no
| ?- zigzag_count([1,3,2,4,8,5,4,6], N).
N = 4 ? ; no
```

3. Union of bags

A bag is a datastructure, that contains elements - like a set - but it can contain the same element multiple times - unlike a set. The number an element appears in the bag is called the multiplicity of that element.

In Prolog a bag can be represented by a list containing E-M pairs, where E is the element in the bag and M is the multiplicity of that element. Each element must be unique, but the order of the elements is arbitrary.

The union of two bags A and B is a bag C, where an element is in C if it is in A or B (or both). The multiplicity of the element in C is the larger of the multiplicity of the same element in A and B. If the element only appears in one of the bags, then that will be the multiplicity in C. The order of the elements in C is arbitrary.

```
% union(+A, +B, -C): A, B and C are bags, C is the union of A and B

| ?- union([], [], B).
B = [] ? ; no
| ?- union([], [111-10], B).
B = [111-10] ? ; no
| ?- union([11-10,33-30], [22-2,44-4], B).
B = [11-10,33-30,22-2,44-4] ? ; no
| ?- union([11-10,22-2,33-30], [11-1,22-20], B).
B = [11-10,22-20,33-30] ? ; no
```

4. Intersection of bags

The intersection of two bags A and B is a bag C, where an element is in C if it is in both A and B. The multiplicity of the element in C is the smaller of the multiplicity of the same element in A and B.

```
% intersection(+A, +B, -C): A, B and C are bags, C is the intersection of A and B

| ?- intersection([], [], B).
B = [] ? ; no
| ?- intersection([], [111-10], B).
B = [] ? ; no
| ?- intersection([11-10,33-30], [22-2,44-4], B).
B = [] ? ; no
| ?- intersection([11-10,22-2,33-30], [11-1,22-20,44-4], B).
B = [11-1,22-2] ? ; no
```

5. Checking if a tree is ordered

A binary tree is ordered, if from the leftmost leaf to the rightmost leaf the values contained in the leaves form a strictly increasing series.

```
%ordered_tree(+Tree): true, if Tree is ordered

| ?- ordered_tree(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3)))).
no
| ?- ordered_tree(node(node(leaf(1),leaf(3)),node(leaf(5),node(leaf(6),leaf(9))))).
yes
| ?- ordered_tree(node(node(node(leaf(1),leaf(2)),node(leaf(3),leaf(4))),
    node(leaf(5),node(node(leaf(6),leaf(7)),leaf(8)))).
yes
```

Hint: you can use a predicate `ordered_tree/3` with the following head comment:

```
% ordered_tree(Tree,M1,M2): Tree is an ordered tree, where
% the leftmost value is greater than M1, and the rightmost value is M2.
```