Heinrich Heine Universität Düsseldorf

Software Engineering and Programming Languages
Prof. Dr. Michael Leuschel
David Geleßus, Fabian Vu

Introduction to Logic Programming – WS 2023
Exercise Sheet 11

1 Exercises

Exercise 1 (Lecture – Inverted Classroom)
Watch the lecture video *20 DCGs DiffLists*[1] in the HHU Mediathek. The corresponding slides are uploaded in ILIAS: 12_DifferenceLists.pdf

The complete playlist is available at: `https://mediathek.hhu.de/playlist/691`.

Note: you have to log in with your HHU account (Uni-Kennung) to see the lecture videos!

The exercises will be discussed on 16th January 2024.

Exercise 2 (Dynamic Predicates and Backtracking)
Predicates can be dynamically added to the knowledge base using `asserta/1` (insert top) or `assertz/1` (insert bottom). Dynamic predicates can be removed using `retract/1` or `retractall/1`.

- Check the behavior of `asserta/1` and `retract/1` when backtracking.
- Implement the predicates `btasserta/1` and `btretract/1` which behave like `asserta/1` and `retract/1` but revert their effect when backtracking.

Exercise 3 (Accumulators)
Implement a predicate `mysum/2` which sums the elements of a list using an accumulator.

The predicate should return 0 for the empty list.

Examples:

```
?- mysum([1,2,3,4,5], S).
S = 15

?- mysum([1], S).
S = 1

?- mysum([], S).
S = 0
```

[1] `https://mediathek.hhu.de/watch/f6ffc36a-bd2d-479d-b6d0-44639146e257`

Exercise 4 (Higher-Order Predicates and Accumulators)
Implement a predicate `reduce/3` which receives a predicate in the first argument and a list in the second argument. The first element of the list should be used as an initial accumulator. The elements of the list are accumulated by applying the passed predicate.

The predicate should fail for the empty list.

Examples:

```
plus(A, B, C) :- C is A + B.

?- reduce(plus, [1,2,3,4], Res).
Res = 10.

?- reduce(plus, [1], Res).
Res = 1.

?- reduce(plus, [], Res).
false.

?- reduce(append, [[1,2],[3],[4]], L).
L = [1, 2, 3, 4].
```

Exercise 5
Implement a predicate `interleave/2` which receives a list of lists as input and collects all elements in the sublist at the same index. The length of the smallest sublist determines the length of the output (see fourth example below).

Examples:

```
?- interleave([[a,b],[1,2]],L).
L = [a, 1, b, 2].

?- interleave([[a,b,c],[1,2,3],[x,y,z]],I).
I = [a, 1, x, b, 2, y, c, 3, z].

?- interleave([],L).
L = [].

?- interleave([[1,2],[3]],L).
L = [1,3].
```