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 4

1 Exercises

The exercises will be discussed on 14th November 2023.

Exercise 1 (Lecture – Inverted Classroom)
Watch the lecture videos *7 LP DPLL Prolog*[1], *8 LP SLD*[2], and *9 LP FOL Syntax*[3] in the HHU Mediathek. The corresponding slides are uploaded in ILIAS: 5_DPLL_SLD.pdf (slides 16-72) and 6_FOL.pdf (slides 1-14)

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!

Exercise 2 (Proof by Contradiction, Resolution)
Let `K := {{A,C},{¬C,B},{¬B,¬C,A}}` be a set of clauses. Prove by contradiction and resolution that the statement A holds.

Exercise 3 (Compressing Prolog Lists)
Implement a predicate `compress(+L, -CL)` which removes consecutive duplicates in a list. The order of the elements in the list should not be changed.

There are unit tests for this exercise in the ILIAS.

Example:

```
?- compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e], CL).
CL = [a,b,c,a,d,e]
```

Exercise 4 (Binary Trees)
Implement the following predicates processing binary trees:

- `inorder(+Tree, -L)` collects the elements of a binary tree in in-order.
- `postorder(+Tree, -L)` collects the elements of a binary tree in post-order.
- `preorder(+Tree, -L)` collects the elements of a binary tree in pre-order.

[1] `https://mediathek.hhu.de/watch/9a9c36f5-42ff-4508-a7e3-e2e38a9c79e1`
[2] `https://mediathek.hhu.de/watch/1a8fa70b-296b-4480-af4e-9f7ec27720cd`
[3] `https://mediathek.hhu.de/watch/cef6a064-502a-42a9-ad40-4ad400e33893`

We represent a binary tree as a term `node(Value, LeftSubTree, RightSubTree)` and the empty tree as the atom `nil`.

Exercise 5 (Interpreting Prolog Code)
Briefly describe the semantics of the following Prolog predicate without implementing it. State the result of `L` for the example call.

```
1 t(L, NL) :-
2     t(L, [], NL).
3 t([], L, L).
4 t([H|T], A, NL) :-
5     t(T, [H|A], NL).
```

Example:

```
1 ?- t([1,2,3,4], L).
```

Exercise 6 (Greatest Common Divisor and Coprime)
a) Implement a predicate `gcd(+X, +Y, -Gcd)` which calculates the greatest common divisor of two natural numbers `X` and `Y`. Use the following imperative pseudocode algorithm as a reference:

```
1 euclid_recursive(x,y):
2     if y = 0:
3         return x
4     if x = 0:
5         return y
6     if x > y:
7         return euclid_recursive(x - y, y)
8     return euclid_recursive(x, y - x)
```

Example:

```
1 ?- gcd(36, 63, G).
2 G = 9
```

b) Implement a predicate `coprime(+X, +Y)` which is true if the two numbers `X` and `Y` are coprime. That means, their greatest common divisor is equal to 1.

Example:

```
1 ?- coprime(35,64).
2 true
```