Kiel University
Department of Computer Science
Programming Languages and Compiler Construction

Priv.-Doz. Dr. Frank Huch, Sandra Dylus, B.Sc. Jan-Hendrik Matthes, Rajib Chandra Das

Institut für Informatik

12. Exercise Sheet on „Advanced Programming (pre-masters)“

Logic Programming 2

Submit by Monday, 01. July 2019 - 14:00

Exercise 1 - Horses and Riders — 2 Points

A family walks past a riding stable on their Sunday walk. While the children watch the horses, the father counts diligently and proudly announces: “I counted 8 heads and 20 legs. How many horses and riders are on the farm?”

Solve the puzzle with a Prolog program using the Peano numbers from the lecture. Define a predicate `rider` that relates the number of heads and legs to the number of horses and riders.

Exercise 2 - Substitution — 2 Points

Given are the following substitutions.

$\sigma = \{X_1 \mapsto t_1, \ldots, X_k \mapsto t_k\} \quad (k \geq 0)$
$\theta = \{Y_1 \mapsto u_1, \ldots, Y_l \mapsto u_l\} \quad (l \geq 0)$

1. Specify the substitution $\theta \circ \sigma$ in set notation (see above).
2. Define the application of the substitution $(\theta \circ \sigma)(X)$ to a variable X by case distinction over X.

Exercise 3 - Unification — 2 Points

Use the algorithm to determine a most general unifier to check the following pairs of terms for unifiability. If the terms can be unified, specify a most general unifier. Otherwise, why did the unification fail?

1. `f(a,g(X,f(Z)),h(Z))` and `f(Y,g(Y,f(b)),h(Y))`
2. `f(f(a,X),b)` and `f(Y,b)`
3. `f(g(X,Y),Z,h(Z))` and `f(Z,g(Y,X),h(g(a,b)))`
4. `f(X,g(X))` and `f(g(Y),Y)`
5. `f(B,C,D)` and `f(g(A,A),g(B,B),g(C,C))`

Exercise 4 - Occurs Check — 2 Points

We consider the following, flawed Haskell program.

```
1 f [x] = [x]
2 f x   = [x]
```

When trying to compile the program, we get the following error message.

```
OccursCheck.hs:2:10: error:
    • Occurs check: cannot construct the infinite type: a ~ [a]
    • In the expression: x
      In the expression: [x]
      In an equation for ‘f’: f x = [x]
    • Relevant bindings include
        x :: [a] (bound at OccursCheck.hs:2:3)
        f :: [a] -> [a] (bound at OccursCheck.hs:1:1)
  |
2 | f x   = [x]
  |
```

Haskell's type inference uses unification to infer the most common type of a function. For each equation of a function, the most general type is determined and the types of the individual equations are then unified to obtain the most general type of all function equations.

Put this error message in relation to the unification algorithm from the lecture. Which type terms are unified with each other and why is there a positive occurs check?

Exercise 5 - House of Nicholas

2 Points

The *generate-and-test* technique can be used to find solutions for a search problem. Use this technique to find a solution for the house of Nicholas.

> The house of Nicholas is a drawing game and puzzle for children. The goal is to draw a "house" in a line of exactly eight routes without having to go through a course twice. Accompanying the drawing is the simultaneous spoken rhyme of eight syllables: "This is the house of Ni-cho-las."

Represent the five corners of the house by the numbers 1 to 5. An edge can then be represented as a tuple of two numbers (corners). Now define a predicate `edges/1` that checks whether the given list contains all the eight possible edges of the house.

Next, define a predicate `flip/2` that flips the tuples in a given list of edges non-deterministically.

Finally, define a predicate `nicholas/1` that checks whether the given sequence of edges represents a valid solution with no interruptions.

After that, you can use the following predicate to find solutions for the house of Nicholas.

```
1 houseOfNicholas(N) :- edges(Es),
2                       permutation(Es, EsP),
3                       flip(EsP, N),
4                       nicholas(N).
```