Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion

Sandra Dylus, Priv.-Doz. Dr. Frank Huch, Niels Bunkenburg

Institut für Informatik

4. Klausur zur Vorlesung „Advanced Programming (pre-masters)“

WS 18/19

You can obtain 28 points within two assignments. To pass the test, you have to reach at least 14 points.

You have 90 minutes to complete the test. It is not allowed to use any material other than a pen. Electronic devices have to be turned off.

Hold your student card nearby, we will check it during the examination.

You will be informed about the results on Wednesday, February 27 (2:00 to 3:00 PM) in room number 715 in CAP 4.

Aufgabe 1 - Questions and Multiple Choice

12 Punkte

Answer the following questions directly on this sheet of paper.

Questions

1. (1P) What is the result for the query `?- swap([1,8,1,1],X).`?

```
1 swap([A,B|L], [B,A|L]) :- B < A.
2 swap([A|L], [A|N]) :- swap(L, N).
```

2. (2P) Give the most general unifier (if it exists) for the terms t_1 = `member(f(A),[A|[A|[]]])` and t_2 = `member(X,[1|Xs])`. Otherwise, explain why no MGU exists.

3. (1.5P) Give all solutions of the goal `?- append(Xs,[2|Ys],[1,2,3,2,4]).`.

For the next five questions, simply mark all correct answers with an X. **If you want to change your answer after marking a statement, fill the original square and draw a new one, as shown in the example.** Note that any number of answers can be correct. Each question is worth 1.5 points. For each incorrect answer 0.5 points are deducted; negative scores are not possible.

Example

1. Which names are associated with Advanced Programming?

   a) ☒ Sandra

   b) ☐ Peter

   c) ☒ Frank

   d) ■ ☐ Rebecca

   e) ☒ Niels

Multiple Choice

1. Let σ be an MGU for the terms t_1 and t_2. Which of the following statements hold?

   a) ☐ $t_1 = t_2$

   b) ☐ $\sigma(t_1) = \sigma(\sigma(t_2))$

   c) ☐ $\sigma(t_1) = \sigma(t_2)$

   d) ☐ $\sigma(t_1) = (\sigma \circ \sigma)(t_2)$

   e) ☐ $t_1 \neq t_2$

2. Which of the following statements about the unification algorithm are true?

   a) ☐ The algorithm always terminates.

   b) ☐ The algorithm does not terminate if no MGU exists.

   c) ☐ Unification is used in Haskell's type inference algorithm.

   d) ☐ The occurs check avoids non-termination.

   e) ☐ The algorithm returns an MGU for every input.

3. Which of the following queries are answered with `true` or a binding for the occuring variables.

   a) ☐ `?- 42 is 7 * 6.`

   b) ☐ `?- 1 + 7 is 1 + 7.`

   c) ☐ `?- 3 + 1 = 5 - 1.`

   d) ☐ `?- 3 * 4.`

   e) ☐ `?- 7 is X.`

4. Which of the following sentences is correct?

   a) ☐ `?- findall(Xs,append(Xs,Ys,[1,2]),L).` has exactly one solution.

   b) ☐ σ is a most general unifier if for all unifiers σ' there exists a substitution ϕ such that $\sigma' = \phi \circ \sigma$.

   c) ☐ Prolog uses the selection strategy FIRST in the SLD resolution.

d) ☐ $ds(g(X, f(Y)), g(a, b)) = \{X, a, f(Y), b\}$

e) ☐ `?- X = f(X).` has a solution in swi-Prolog.

5. Which of the following list patterns unify with the expression `[1,2]`?

a) ☐ `[X | [2]]`

b) ☐ `[1 | X]`

c) ☐ `[1 | 2]`

d) ☐ `[[1] | [2]]`

e) ☐ `[1, 2 | []]`

Aufgabe 2 - Programming in Prolog 16 Punkte

In this excercise we use Peano numbers. The structure should be clear from the following predicate, which can be used to check whether a given value is a Peano number.

```
1 isPeano(o).
2 isPeano(s(N)) :- isPeano(N).
```

1. (4P) Define a predicate `max(P1,P2,Max)` that computes the maximum `Max` of two Peano numbers `P1` and `P2`. Implement this predicate directly, that is, without using a helper predicate like `less` from the lecture. Your solution should not generate correct answers multiple times.

   Note that your program should also produce solutions for the goal `?- max(N,M,s(o))`.

2. (4P) Define a predicate `take(P,Xs,Ys)` that checks whether the first `P` elements of the list `Xs` are identical with the list `Ys`. Note that `P` is a Peano number.

3. **See back side!**

1. (8P) We consider the following Prolog program.

```
1 app([],Ys,Ys).
2 app([X|Xs],Ys,[X|Zs]) :- app(Xs,Ys,Zs).
```

and the goal `?- app(Xs,Ys,[1|[Y|[]]]), [Y] = Xs.`

Construct the SLD tree for this goal **according to the format introduced in the lectures and exercises**.