Test-Exam

June 28, 2023

Lastname: ______________________________

Firstname: ______________________________

Matriculation Number: ______________________________

Exercise	Points	Score
Single Choice	30	
Well-Definedness of Functional Programs	22	
Verification of Functional Programs	30	
Verification of Imperative Programs	18	
$\sum$	100	

- The time for the exam is 100 minutes, so 1 point = 1 minute.
- The available points per exercise are written in the margin.
- Write on the printed exam and use extra blank sheets if more space is required.
- Your answers can be written in English or German.

Exercise 1: Single Choice $\boxed{30}$

For each statement indicate whether it is true (✓) or false (✗). Giving the correct answer is worth 3 points, giving no answer counts 1 point, and giving the wrong answer counts 0 points (for that statement).

1. ____ Both R^+, the transitive closure of a binary relation R, and R^*, the reflexive-transitive closure of R, can be specified as inductively defined sets.
2. ____ Whenever R is an inductively defined set, then membership in R is decidable.
3. ____ Consider some functional program P and its one step relation $\hookrightarrow$. P is pattern disjoint if and only if $\hookrightarrow$ is locally confluent.
4. ____ Let $\mathcal{E}$ be the universally quantified defining equations of a well-defined functional program. Whenever $\mathcal{M} \models \varphi$ for the standard model $\mathcal{M}$, then $\mathcal{E} \models \varphi$.
5. ____ The proof of Ramsey's theorem is performed by induction on the number of colors.

Consider a well-defined functional program P including some datatype definitions and defining equations. Now assume that all datatype definitions are merged into a single one as follows. One creates a new type τ and each n-ary constructor c of the functional program now gets type $c : \tau \times \ldots \times \tau \to \tau$. Afterwards, all previous datatypes are thrown away, but the equations are untouched, only the type of the defined symbols is changed accordingly. Let us call the result of this process P'.

Example: Instead of List and Nat one would have one datatype Tau and the constructors would be $\mathsf{Zero} : \mathsf{Tau}$, $\mathsf{Succ} : \mathsf{Tau} \to \mathsf{Tau}$, $\mathsf{Nil} : \mathsf{Tau}$ and $\mathsf{Cons} : \mathsf{Tau} \times \mathsf{Tau} \to \mathsf{Tau}$. Furthermore, the type of functions plus and append would both be changed to $\mathsf{Tau} \times \mathsf{Tau} \to \mathsf{Tau}$, and the reverse function would get type $\mathsf{Tau} \to \mathsf{Tau}$.

Now answer whether the following statements always hold, i.e., whether they are true whenever P is well-defined.

6. ____ P' is a functional program, i.e., in particular all defining equations are well-typed.
7. ____ P' is pattern-complete.
8. ____ P' is pattern-disjoint.
9. ____ Any termination proof via dependency pairs, usable equations and polynomial interpretations for P can be used without changes for P'.
10. ____ The axioms for equality of constructors for P' are a subset of the axioms for equality of constructors in P (when identifying the equality-signs for the new type τ and the various old types).

Exercise 2: Well-Definedness of Functional Programs $\boxed{22}$

Consider the following functional program for computing the maximal element of a list of natural numbers.

$$\begin{aligned}
\mathsf{max}(\mathsf{Succ}(x), \mathsf{Succ}(y)) &= \mathsf{Succ}(\mathsf{max}(y, x)) && (1)\\
\mathsf{max}(\mathsf{Zero}, y) &= y && (2)\\
\mathsf{list\_max}(\mathsf{Cons}(x, \mathsf{Cons}(y, ys))) &= \mathsf{list\_max}(\mathsf{Cons}(\mathsf{max}(x, y), ys)) && (3)\\
\mathsf{list\_max}(\mathsf{Cons}(x, \mathsf{Nil})) &= x && (4)
\end{aligned}$$

(a) Determine if the program is pattern-disjoint and pattern-complete. If this is not the case, add equations or modify the existing equations in a sensible way so that the resulting program P is both pattern-disjoint and pattern-complete. (4)

(b) Compute all dependency pairs of P and remove as many of these using the subterm criterion or the size-change criterion. In the former case provide the argument position, in the latter case provide the set of multigraphs. (5)

(c) For all remaining dependency pairs, determine the set of usable equations, and write down the constraints that you get when trying to prove termination with the help of some reduction pair $(\succ, \succsim)$. (5)

(d) Solve the constraints. To this end you can choose to either (8)

- provide a concrete polynomial interpretation so that the constraints are satisfied (by guessing a suitable one), or
- use a symbolic linear polynomial interpretation and show which constraints you get after applying the absolute positiveness criterion; here, you should use the symbolic interpretation where each symbol is abbreviated by its first letter, e.g.:

$$[\![\mathsf{max}]\!](x, y) = \mathsf{m}_0 + \mathsf{m}_1 x + \mathsf{m}_2 y$$
$$[\![\mathsf{Cons}]\!](x, y) = \mathsf{C}_0 + \mathsf{C}_1 x + \mathsf{C}_2 y$$

Exercise 3: Verification of Functional Programs $\boxed{30}$

Consider the following functional program for reversing lists.

$$\begin{aligned}
\mathsf{append}(\mathsf{Cons}(x, xs), ys) &= \mathsf{Cons}(x, \mathsf{append}(xs, ys)) \\
\mathsf{append}(\mathsf{Nil}, ys) &= ys \\
\mathsf{rev}(\mathsf{Cons}(x, xs)) &= \mathsf{append}(\mathsf{rev}(xs), \mathsf{Cons}(x, \mathsf{Nil})) \\
\mathsf{rev}(\mathsf{Nil}) &= \mathsf{Nil}
\end{aligned}$$

(a) Complete the following property so that it is a theorem in the standard model. (3)

$$\forall xs, ys.\ \mathsf{rev}(\mathsf{append}(xs, ys)) =_{\mathsf{List}} \mathsf{append}(\qquad\qquad\qquad\qquad)$$

(b) Prove the specified property of the previous part by using induction and equational reasoning via $\rightsquigarrow$. (13)

- Briefly state on which variable(s) you perform induction, and which induction scheme you are using.
- Write down each case explicitly and also write down the IH that you get, including quantifiers.
- Write down each single $\rightsquigarrow$-step in your proof.
- You may use the following two well-known properties from the lecture as axioms.
  - $\forall xs.\ \mathsf{append}(xs, \mathsf{Nil}) =_{\mathsf{List}} xs$
  - $\forall xs, ys, zs.\ \mathsf{append}(\mathsf{append}(xs, ys), zs) =_{\mathsf{List}} \mathsf{append}(xs, \mathsf{append}(ys, zs))$

  If you require further auxiliary properties, then write them down, but don't prove them.

$$\begin{aligned}
\mathsf{append}(\mathsf{Cons}(x, xs), ys) &= \mathsf{Cons}(x, \mathsf{append}(xs, ys)) \\
\mathsf{append}(\mathsf{Nil}, ys) &= ys \\
\mathsf{rev}(\mathsf{Cons}(x, xs)) &= \mathsf{append}(\mathsf{rev}(xs), \mathsf{Cons}(x, \mathsf{Nil})) \\
\mathsf{rev}(\mathsf{Nil}) &= \mathsf{Nil}
\end{aligned}$$

(c) Prove $\forall xs.\ \mathsf{rev}(\mathsf{rev}(xs)) =_{\mathsf{List}} xs$ by using induction and equational reasoning via $\rightsquigarrow$ as in the previous part. You can use the property of the previous parts as axiom. (14)

Exercise 4: Verification of Imperative Programs $\boxed{18}$

The Fibonacci-numbers are defined as follows:

$$\begin{aligned} fib(0) &= 0 \\ fib(1) &= 1 \\ fib(n+2) &= fib(n+1) + fib(n) \qquad \text{for } n \in \mathbb{N} \end{aligned}$$

The following program P calculates the Fibonacci-numbers.

```
x = 0;
y = 1;
while (n != 0) {
  z := x + y;
  x := y;
  y := z;
  n := n - 1;
}
```

(a) Write down a specification in form of a Hoare triple that P computes the Fibonacci-numbers. (2)

(b) For which inputs does the program terminate? Provide a suitable variant e that can be used to prove termination. (It suffices to provide e in this part, a proof tableau is not required.) (3)

(c) Construct a proof tableau for proving partial correctness. (13)

```
x = 0;

y = 1;

while (n != 0) {

  z := x + y;

  x := y;

  y := z;

  n := n - 1;

}
```