universität innsbruck

Program Verification	SS 2023	LVA 703083+703084
Sheet 4		Deadline: April 19, 2023, 10am

- Prepare your solutions on paper.
- Mark the exercises in OLAT before the deadline.
- Upload your Haskell solution in OLAT.
- Marking an exercise means that a significant part of that exercise has been treated.

Exercise 1 *Matching Algorithm* **6 p.**

The matching algorithm has been proven correct in the lecture. However, the algorithm itself is only pseudo-code.

1. Implement the matching algorithm in Haskell. A template-file is given. (3 points)
2. Implement an algorithm in Haskell which evaluates a term t one step, i.e., either some term s such that $t \hookrightarrow s$ should be returned, or it should be indicated that there is no such term. A template-file is given. (3 points)

Exercise 2 *Preservation of Groundness* **9 p.**

The aim of this exercise is to prove that $\hookrightarrow^*$ preserves groundness, cf. slide 3/35.

1. Adjust the proof structure given on slide 3/34 to groundness. To this end in this first part you should just *specify* the property of groundness-preservation for the various operations. For instance, the property "$\hookrightarrow$ preserves groundness" can be expressed with existing notions:

   $$t \hookrightarrow s \longrightarrow t \in \mathcal{T}(\Sigma) \longrightarrow s \in \mathcal{T}(\Sigma)$$

   For other operations you first need to define a notion which connects groundness and substitutions.

   (3 points)
2. Prove that matching preserves groundness. Explicitly state the property on that you perform induction or state which invariant you are using. If necessary, adjust or generalize your notion of groundness of substitutions. (3 points)
3. Prove that $\hookrightarrow$ preserves groundness, where you can assume groundness-preservation for matching and substitution application. Explicitly state the property on that you perform induction or state which invariant you are using. (3 points)

Exercise 3 *Pattern Disjointness* **5 p.**

Consider the definition of pattern disjointness on slide 3/39. Testing whether a program is pattern disjoint is not directly possible based on this definition, since it involves a quantification over infinitely many terms. In this exercise, the aim is to develop an algorithm to decide whether a program is pattern disjoint, based on unification.
Unification of two terms s and t is the question whether there exists a substitution σ such that $s\sigma = t\sigma$ and deliver such a substitution in case it exists. So in contrast to matching, here the substitution is applied on both terms.

A concrete unification algorithm is due to Martelli and Montanari, cf. https://en.wikipedia.org/wiki/Unification_(computer_science)#A_unification_algorithm, and its structure is quite similar to the matching algorithm.
Consider the following algorithm to decide pattern disjointness of a program:

check for each pair of distinct equations $\ell_1 = r_1$ and $\ell_2 = r_2$ that ℓ_1 and ℓ_2 do not unify.

Argue that this algorithm is not correct with the help of the following functional program (datatype definitions omitted). Here, you don't have to perform the unification algorithm step-by-step.

$$\mathsf{append}(\mathsf{Cons}(x, xs), ys) = \mathsf{Cons}(x, \mathsf{append}(xs, ys)) \tag{1}$$
$$\mathsf{append}(\mathsf{Nil}, \mathsf{Cons}(y, ys)) = \mathsf{Cons}(y, ys) \tag{2}$$
$$\mathsf{append}(xs, \mathsf{Nil}) = xs \tag{3}$$

How would you correct the algorithm? (5 points)