universität innsbruck

Program Verification	SS 2024	LVA 703083+703084
Sheet 5		Deadline: April 23, 2024, 3pm

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

Exercise 1 *Correctness of Implementation of Unification* **6 p.**

Study the proof given on slides 4/36–37.

1. Perform the proof of case 3, i.e., where the arguments are $(f(ts), x) : u$ and v. (2 points)
2. In case 4 with arguments $(x, t) : u$ and v the algorithm deviates from the abstract algorithm in the following sense: the abstract algorithm only applies (eliminate) if x occurs in U, but such a condition is not tested in the implementation.

   Prove that this difference does not cause a problem, i.e., prove $P((x, t) : u, v, U)$ where $x \neq t$, $x \notin \mathcal{V}ars(t)$ and x does not occur in $set\ u \cup set\ v$, where of course you may assume an IH for the recursive invocation of *unifyMain*. (4 points)

Exercise 2 *Pattern Completeness* **14 p.**

Consider the algorithm for pattern completeness on slide 4/44.

1. The output of the algorithm is just a Boolean, i.e., the result is either $\bot$ (not pattern complete) or $\varnothing$ (pattern complete).

   Note that the fully expanded semantics of completeness of a set of pattern problems P is as follows:

   $$P \text{ is complete iff } \forall pp \in P.\ \forall \sigma : \mathcal{V} \to \mathcal{T}(\mathcal{C}).\ \underbrace{\exists mp \in pp.\ \exists \gamma.\ \forall (t, \ell) \in mp.\ t\sigma = \ell\gamma}_{=:\varphi(pp,\sigma)}$$

   Hence, if P is not pattern complete there must be some witness pattern problem $pp \in P$ and witness substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{C})$ such that $\varphi(pp, \sigma)$ is not satisfied.

   - Modify the algorithm for pattern completeness in a way that witnesses can be obtained instead of just returning $\bot$ for incomplete P.
   - Illustrate the modified algorithm on the example input on slide 4/46.
   - You do NOT have to prove correctness of the modified algorithm.

   Hint: only $\rightharpoonup\!\!\!\!\rightharpoonup$ needs to be modified, $\rightharpoonup$ does not need to be altered.

   (7 points)

2. Design an implementation of the pattern completeness algorithm in Haskell, and, optionally, try to define a refinement relation and a partial correctness statement.

   Hint: it might be useful to design several sub-algorithms, working on matching problems, pattern problems and lists of pattern problems. Note that for the optional task, each sub-algorithm can have its own refinement relation and partial correctness statement. (7 points)