universität innsbruck

Program Verification	SS 2021	LVA 703083+703084
Sheet 3		Deadline: March 23, 2021, 8am

- Prepare your solutions on paper.
- Marking an exercise in OLAT means that a significant part of that exercise has been treated.
- Upload your solution in OLAT as a single PDF file.

Exercise 1 *Type-Checking of Formulas* **7 p.**

Consider the type-checking algorithm for formulas from the (solution of the) previous exercise sheet. Prove soundness of the type-checking algorithm as in slides 2/39 – 2/41.

$$\mathit{type\_check\_formula}\ \Sigma\ \mathcal{V}\ \mathcal{P}\ \varphi = \mathit{return}\ () \longrightarrow \varphi \in \mathcal{F}(\Sigma, \mathcal{P}, \mathcal{V})$$

Be precise when applying induction: what kind of induction? on which property $P(\ldots)$? on which variables? which variables are arbitrary?

Exercise 2 *Data Type Definitions* **5 p.**

Consider slides 3/3 and 3/7.

1. What would go wrong if one drops distinctness of the constructor names? Provide a concrete data type definition where something goes wrong, i.e., where all conditions except for the distinctness of constructor names are satisfied, but where in the definitions of $\mathcal{T}y$, Σ, $\mathcal{P}$ and $\mathcal{M}$ some problem occurs. (2 points)
2. Consider the following sequence of datatypes that define rose trees, i.e., trees where each node may have arbitrarily many children. (3 points)

   $$\textsf{data Nat} = \textsf{Zero} : \textsf{Nat} \mid \textsf{Succ} : \textsf{Nat} \to \textsf{Nat}$$
   $$\textsf{data Tree} = \textsf{Node} : \textsf{Nat} \times \textsf{Tree\_List} \to \textsf{Tree}$$
   $$\textsf{data Tree\_List} = \textsf{Nil} : \textsf{Tree\_List} \mid \textsf{Cons} : \textsf{Tree} \times \textsf{Tree\_List} \to \textsf{Tree\_List}$$

   - Describe the universes of trees and tree-lists as inductive sets.
   - Are all universes non-empty? For each non-empty universe provide an element that is in the universe.
   - Is the definition allowed wrt. slide 3/3? If not, give a short description why it is not allowed.

Exercise 3 *Functional Programming* **8 p.**

Consider slides 3/14 – 3/20.

1. Specify an algorithm for subtraction of two natural numbers within the functional programming language defined in the slides and evaluate "3 − 2" and "2 − 3" step-by-step on paper. (2 points)
2. Specify an algorithm for the division of two natural numbers within the functional programming language defined in the slides. Evaluate "2/2" step-by-step on paper. How does your algorithm handle division-by-zero? How does your algorithm handle non-exact division, e.g., dividing 1 by 2. (3 points)
3. Function definitions on slide 3/15 are quite restricted, e.g., no mutual recursion, no if-then-else, no built-in integers, etc. (3 points)
   - Try to modify the definition of function definitions on slide 3/15 in a way that allows mutual recursion.
   - Ensure that the even-odd definitions on slide 3/17 are accepted.