Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inf. P. Michel
Dipl.-Inf. C. Feller

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Exercise Sheet 5: Specification and Verification with Higher-Order Logic (Summer Term 2012)

Date: 09.05.2012

Exercise 1 Methods and Rules in Isabelle/HOL

In this exercise we want to practice the use of different methods (like `rule`, `erule` or `frule`) to prove properties in propositional and predicate logic.

You should only use the rules of the first exercise sheet, together with the following additional rules: `conjE`, `impE`, `iffI`, `iffE` and `classical`.

Hint: You can always invoke `C-c C-v` *to enter a command like* `thm impI` *and see the concrete definition of the rule in Isabelle/HOL.*

a) (Prepare!) Apply the rule

$$[\![(?a, ?b) \in ?r^*; \bigwedge x.\ ?P\ x\ x; \bigwedge x\ y\ z.\ [\![(x, y) \in ?r^*;\ ?P\ x\ y;\ (y, z) \in ?r]\!] \Longrightarrow ?P\ x\ z]\!] \Longrightarrow ?P\ ?a\ ?b$$

with the method `erule` to the following subgoal by hand (i.e. on paper):

$$(i, j) \in s^* \Longrightarrow 0 \leq (dist\ i\ j)$$

Hint: Don't be distracted by unknown function names; you don't have to know anything about their meaning. Just apply the rule syntactically.

b) Prove or disprove the following theorems.

- $A \longrightarrow A$
- $A \wedge B \longrightarrow B \wedge A$
- $(A \wedge B) \longrightarrow (A \vee B)$
- $((A \vee B) \vee C) \longrightarrow A \vee (B \vee C)$
- $A \longrightarrow B \longrightarrow A$
- $(A \vee A) = (A \wedge A)$
- $(A \longrightarrow B \longrightarrow C) \longrightarrow (A \longrightarrow B) \longrightarrow A \longrightarrow C$
- $(A \longrightarrow B) \longrightarrow (B \longrightarrow C) \longrightarrow A \longrightarrow C$
- $\neg\neg A \longrightarrow A$
- $A \longrightarrow \neg\neg A$
- $(\neg A \longrightarrow B) \longrightarrow (\neg B \longrightarrow A)$
- $((A \longrightarrow B) \longrightarrow A) \longrightarrow A$
- $A \vee \neg A$
- $(\neg(A \wedge B)) = (\neg A \vee \neg B)$

- $(\exists x.\ \forall y.\ P\ x\ y) \longrightarrow (\forall y.\ \exists x.\ P\ x\ y)$
- $(\forall x.\ P\ x \longrightarrow Q) = ((\exists x.\ P\ x) \longrightarrow Q)$
- $((\forall x.\ P\ x) \wedge (\forall x.\ Q\ x)) = (\forall x.\ (P\ x \wedge Q\ x))$
- $((\forall x.\ P\ x) \vee (\forall x.\ Q\ x)) = (\forall x.\ (P\ x \vee Q\ x))$
- $((\exists x.\ P\ x) \vee (\exists x.\ Q\ x)) = (\exists x.\ (P\ x \vee Q\ x))$
- $(\forall x.\ \exists y.\ P\ x\ y) \longrightarrow (\exists y.\ \forall x.\ P\ x\ y)$
- $(\neg(\forall x.\ P\ x)) = (\exists x.\ \neg P\ x)$

Exercise 2 Language Semantics, Specification and Correctness

In this exercise we look at the compiler from Section 3.3 of the Isabelle/HOL tutorial.

a) (Prepare!) Make yourself familiar with the involved languages, the compiler and the definition of its correctness. In particular:

   - Create an Isabelle/HOL theory with all the datatype and function definitions of the two languages and the compiler.
   - Define two constants for source programs, representing the two expressions `((a + 1) + b)` and `(5 + (2 * (3 + 6)))`.
   - Evaluate the expressions, execute their compiled counterparts and compare the results.
   - Add the correctness theorem (and auxiliary lemma) of the section to your theory and complete their proofs using the hints given in the tutorial.

b) (Prepare!) Add unary operators to the source and target language. Adjust the proofs accordingly.

c) At the moment, programs of the source language are just expressions. We now want to extend the language with assignments. A program is then a sequence of assignments. As seen in the tutorial, the semantics of expressions are just values. The semantics of a statement is the state after executing the statement.

   - Define a datatype for statements, which are either sequences of statements or assignments.
   - Define a function to ”run” statements.

d) We want to extend the compiler to statements. We therefore need a store instruction in the target language.

   Extend the target language with a store instruction and adjust the compiler accordingly.

e) To show the correctness of the new compiler, adjust the semantics of the target language and add a fitting correctness theorem. Prove the theorem.