Prof. Dr. A. Poetzsch-Heffter
M.Sc. Peter Zeller
Dipl.-Inf. C. Feller

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Exercise Sheet 1: Specification and Verification with Higher-Order Logic (Summer Term 2014)

Please prepare the marked tasks for the exercise on Wednesday, April 30, 2014

Exercise 1 Calculus of Natural Deduction

We consider the *Genzten-Calculus*, also known as calculus of *natural deduction*. The calculus uses *sequents* (German: *Sequenzen*) of the form $\Gamma \vdash A$. They state that the formula A can be syntactically derived from the set of formulas Γ. If it is possible to derive such a sequent using only the *rules* of the calculus, starting from the *axioms*, we also know that A is a semantic conclusion from Γ (as the calculus is *correct*).

The calculus has only one axiom, which states that every formula can be derived from itself: $A \vdash A$, for all formulas A. Additionally, there are various rules to derive new sequents from existing ones:

Conjunction, Disjunction and Implication (Binary Relations)

$$\frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\ {\scriptstyle(\wedge I)}\ \texttt{conjI} \qquad \frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\ {\scriptstyle(\vee I_l)}\ \texttt{disjI1} \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\ {\scriptstyle(\vee I_r)}\ \texttt{disjI2}$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q}\ {\scriptstyle(\to I)}\ \texttt{impI} \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}\ {\scriptstyle(\wedge E_l)}\ \texttt{conjunct1} \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}\ {\scriptstyle(\wedge E_r)}\ \texttt{conjunct2}$$

$$\frac{\Gamma \vdash P \to Q \qquad \Gamma \vdash P}{\Gamma \vdash Q}\ {\scriptstyle(\to E)}\ \texttt{mp} \qquad \frac{\Gamma \vdash P \vee Q \qquad \Gamma, P \vdash R \qquad \Gamma, Q \vdash R}{\Gamma \vdash R}\ {\scriptstyle(\vee E)}\ \texttt{disjE}$$

Truth Values (Constants), Negation (Unary Relation) and Weakening

$$\frac{\Gamma \vdash \texttt{False}}{\Gamma \vdash P}\ {\scriptstyle(\texttt{False}E)}\ \texttt{FalseE} \qquad \frac{\Gamma, P \vdash \texttt{False}}{\Gamma \vdash \neg P}\ {\scriptstyle(\neg I)}\ \texttt{notI} \qquad \frac{\Gamma \vdash \neg P \qquad \Gamma \vdash P}{\Gamma \vdash \texttt{False}}\ {\scriptstyle(\neg E)}\ \texttt{notE} \qquad \frac{\Gamma \vdash Q}{\Gamma, P \vdash Q}\ {\scriptstyle(W)}$$

Universal and Existential Quantifiers

$$\frac{\Gamma \vdash \{a_{new}/x\}P}{\Gamma \vdash \forall x.P}\ {\scriptstyle(\forall I)}\ \texttt{allI} \qquad \frac{\Gamma \vdash \forall x.P}{\Gamma \vdash \{t/x\}P}\ {\scriptstyle(\forall E)}\ \texttt{spec}$$

$$\frac{\Gamma \vdash \{t/x\}P}{\Gamma \vdash \exists x.P}\ {\scriptstyle(\exists I)}\ \texttt{exI} \qquad \frac{\Gamma \vdash \exists x.P \qquad \Gamma, \{a_{new}/x\}P \vdash Q}{\Gamma \vdash Q}\ {\scriptstyle(\exists E)}\ \texttt{exE}$$

The names of the rules are given on the right side in parenthesis. The name of the corresponding Isabelle/HOL rules are given below in `typewriter` font. The I is an abbreviation of *Introduction*, E of *Elimination* and W of *Weakening*. The syntax $\{y/x\}A$ denotes that all unbound occurrences of x in A are replaced by y. You have to choose a completely new variable for each a_{new}, i.e. it must not appear in any term or formula yet. t on the other hand is allowed to be an arbitrary term.

A proof in the calculus is a tree of rule applications, whose leaves are axioms and whose root is the theorem you want to prove. Usually such a proof is done *backwards*, starting with the theorem and trying to reach the axioms.

a) (Prepare!) Prove the following sequent using the Gentzen-Calculus:

$$\vdash a \rightarrow (a \vee b)$$

b) (Prepare!) Prove the following sequent using the Gentzen-Calculus:

$$\vdash (a \vee (b \wedge c)) \rightarrow ((a \vee b) \wedge (a \vee c))$$

c) (Prepare!) Prove the following sequent using the Gentzen-Calculus:

$$\vdash \exists x.\forall y.P(x,y) \rightarrow \forall y.\exists x.P(x,y)$$

d) Write an Isabelle/HOL theory for your proofs from a), b) and c).

Exercise 2 Functions in Isabelle/HOL

Please do not use the append operator '`op @`' or any other predefined functions on lists for this exercise.

a) Write a function `swap : 'a * 'b => 'b * 'a`, which swaps the two components of a pair.

b) Write a function `listSwap : ('a * 'b) list => ('b * 'a) list`, which swaps all pairs of a list.

c) Write a function `map : ('a => 'b) => 'a list => 'b list`, which applies a function to all elements of a list.

d) Write a function `listSwap2 : ('a * 'b) list => ('b * 'a) list`, with the same behavior as `listSwap`, using the `map` function instead of recursion.

e) Write a function `findL : 'a list => 'a => bool`, which determines if a value is contained in a list.

Exercise 3 Datatypes in Isabelle/HOL (Hand in!)

a) Define a datatype `'a tree` to represent binary trees. Leaves should be `Empty` and internal nodes should store a value of type `'a`.

b) Write a function `findT : 'a tree => 'a => bool`, which determines if a value is contained in a tree.

c) Define the functions `preOrder`, `postOrder` and `inOrder` that traverse and convert a binary tree to a list in the respective order.

d) Define a function `mapT : ('a => 'b) => 'a tree => 'b tree` that returns a tree where all markings of the original tree have been replaced according to the given function.