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 $\Gamma$. 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 $\Gamma$ (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)

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \land Q} \quad (\&I) \quad \frac{\Gamma \vdash P}{\Gamma \vdash P \lor Q} \quad (\lor I) \quad \frac{\Gamma \vdash Q}{\Gamma \vdash P \lor Q} \quad (\lor E)$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \rightarrow Q} \quad (\rightarrow I) \quad \frac{\Gamma \vdash P \land Q}{\Gamma \vdash P} \quad (\land I) \quad \frac{\Gamma \vdash P \land Q}{\Gamma \vdash Q} \quad (\land E)$$

$$\frac{\Gamma \vdash P \rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q} \quad (\rightarrow E) \quad \frac{\Gamma \vdash P \land Q \quad \Gamma \vdash P}{\Gamma \vdash P} \quad (\land E) \quad \frac{\Gamma \vdash P \land Q \quad \Gamma \vdash Q}{\Gamma \vdash P} \quad (\land E)$$

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

$$\frac{\Gamma \vdash \text{False}}{\Gamma \vdash P} \quad \text{FalseE} \quad \frac{\Gamma, P \vdash \text{False}}{\Gamma \vdash \neg P} \quad (\neg I) \quad \frac{\Gamma \vdash \neg P}{\Gamma \vdash \text{False}} \quad \text{notI} \quad \frac{\Gamma \vdash \text{False}}{\Gamma \vdash P} \quad \text{notE} \quad \frac{\Gamma \vdash \text{False}}{\Gamma \vdash \neg P} \quad \text{notE} \quad \frac{\Gamma \vdash Q}{\Gamma \vdash \text{False}} \quad (W)$$

Universal and Existential Quantifiers

$$\frac{\Gamma \vdash \{a_{\text{new}}/x\}P}{\Gamma \vdash \forall x.P} \quad (\forall I) \quad \frac{\Gamma \vdash \forall x.P}{\Gamma \vdash \{t/x\}P} \quad \text{spec}$$

$$\frac{\Gamma \vdash \{t/x\}P}{\Gamma \vdash \exists x.P} \quad (\exists I) \quad \frac{\Gamma \vdash \exists x.P \quad \Gamma, \{a_{\text{new}}/x\}P \vdash Q}{\Gamma \vdash \{a_{\text{new}}/x\}P \vdash Q} \quad (\exists E)$$

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 $a_{\text{new}}$, 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:

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

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

$$\vdash (a \lor (b \land c)) \rightarrow ((a \lor b) \land (a \lor c))$$

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.