UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Development of Safety-Critical Systems (Sichere Software) Winter Term 2018
Martin Leucker, Karam Kharraz, Felix Lange December 8, 2018

Exercise Sheet 6

Task 6.1 Propositional Logic

Recall the proof system of natural deduction defined by the four inference rules. Here is a proof that inconsistent assumptions p and $\neg p$ imply any q.

$$\dfrac{\dfrac{\dfrac{\dfrac{p \to \bot \in \{...\}\ \dfrac{}{p, p \to \bot, q \to \bot \vdash p \to \bot}\,\textsc{Assump} \qquad p \in \{...\}\ \dfrac{}{p, p \to \bot, q \to \bot \vdash p}\,\textsc{Assump}}{p, p \to \bot, q \to \bot \vdash \bot}\,\textsc{ImpElim}}{p, p \to \bot \vdash q}\,\textsc{DoubleNeg}}{p \vdash (p \to \bot) \to q}\,\textsc{ImpIntro}}{\vdash p \to (p \to \bot) \to q}\,\textsc{ImpIntro}$$

a) Use the rules to construct proof trees for the following formulas. Prove them, and also $p \to (p \to \bot) \to q$, in Coq using tactics. Finally, prove them by giving explicit proof terms. Use the given template file for the Coq proofs.

   (i) $p \to (p \to \bot) \to \bot$

   (ii) $(p \to q) \to (q \to r) \to (p \to r)$

b) Extend the syntax of propositional logic by conjunction and disjunction. Extend the semantics (evaluation function). Extend the inference rules of the proof system. *Hint:* For both operators, you will need introduction and elimination rule(s), analogously to the two rules ImpIntro, ImpElim. If you get stuck, you may get inspiration from the internet.

c) Prove the following formulas in your extended proof system. Then prove them in Coq both by using tactics and by giving a proof term (use again the template file).

   (i) $(p \wedge q) \to (q \wedge p)$ *“$\wedge$ is commutative”*

   (ii) $p \wedge (q \vee r) \to (p \wedge q) \vee (p \wedge r)$ *“$\wedge$ is distributive over $\vee$”*

Task 6.2 Programming in Coq

Consider the following `option` data type that provides two kinds of values, either `none` or `some v` for some value `v`. Like `list`, it is parameterized over a type `A`.

```
Inductive option (A: Type) : Type :=
| some : A → option A
| none : option A.
```

Write the following programs, using the provided template file.

(i) `pred_opt: nat → option nat`
This function returns the predecessor of a natural number, or `none` if it does not exist.

(ii) `pred: nat → nat`
This function returns the predecessor of a natural number, or `0` if it does not exist.

(iii) `minus: nat → nat → option nat`
This function computes $x - y$ for two natural numbers. If the result is not a natural number, `none` is returned.

(iv) `is_none: ∀(X: Type), option X → bool`
This polymorphic function takes an optional value and returns `true` if the value is `none` and `false` otherwise.

(v) `head: ∀(X: Type), list X → option X`
This function returns the first element of a list, or `none` if the list is empty.

Task 6.3 Proving in Coq

Prove the following propositions in Coq, using the provided template file.

(i) `∀ (X: Type) (x: X) (xs: list X), head (cons x xs) = some x`

(ii) `∀ (n: nat), S (pred n) = n ∨ n = 0`

(iii) `∀ (x y z: nat), (x + y) + z = x + (y + z)`

(iv) `∀ (m n: nat), minus m n = none ∨ minus n m = none ∨ m = n`