EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät

Programmiersprachen und Softwaretechnik

Prof. Klaus Ostermann

Leitung des Seminars
Julian Jabs
David Binder

Interactive Theorem Proving

Hausaufgabenblatt 1 – WS18/19

Tübingen, 15. Oktober 2018

Aufgabe 1: Setup your Development Environment

1.1 Choose an IDE

There are currently two workable options for writing proof scripts with Coq. You can use either use CoqIDE, a graphical environment for interactively developing proof scripts, or you can use Emacs with the Proof General and company-coq extensions.

CoqIDE is simpler to start with, but the combination of emacs with its extensions is considerably more powerful and provides more features. For the purposes of this lecture, both options are adequate.

1.2 Download and install Coq

1.2.1 Windows

Get the installer for Coq 8.8.1 from https://github.com/coq/coq/releases/tag/V8.8.1. This version comes bundled with the latest version of CoqIDE.

1.2.2 Mac

You can either use the installer for MacOS from https://github.com/coq/coq/releases/tag/V8.8.1 which comes bundled with CoqIDE, or install Coq from the sources via OPAM (cp. the next section).

1.2.3 Linux

If the package manager of your distribution provides current versions of Coq (preferably version 8.8.1) and CoqIDE, you can use those.

If your package manager does not provide a current version of Coq, it is recommended to use OPAM, the package manager for the programming language OCaml (the language Coq is implemented in). In addition to being the package manager for OCaml, OPAM is also used to distribute libraries and extensions for Coq.

A guide to install Coq and CoqIDE via OPAM is provided on: https://coq.inria.fr/opam/www/using.html

1.3 (Optional) Configure Emacs

If you are comfortable using Emacs, you can use Emacs with ProofGeneral and company-coq.

Proof General (https://proofgeneral.github.io/) is an extension for emacs which works with several different proof assistants. Proof General is necessary to interactively develop proofs with Coq and Emacs.

company-coq is an additional extension which requires Proof General to be installed. company-coq provides a lot of additional convenience features when working with Emacs and Proof General.

Aufgabe 2: First assignment

2.1 Get the code for your first assignment

You will be using GitHub Classroom in order to get the code for your assignments, upload your homework and get feedback on your submissions. For this to work you need to create a GitHub account, if you haven't done so already.

Once you have a Github Account, follow the invitation link:

https://classroom.github.com/a/n_5GF8nv

You will be asked to choose your codename, which has been assigned to you in the first lecture, from a list. This is so that we can match your real name (e.g. Peter Mueller) with your GitHub name (e.g. 1337h4ck3r). A repository with the name „assignment-1-<YOUR NAME>“ will be created for you, and you will be granted the necessary privileges to push to that repository. Clone that repository and open the file.

2.2 Write a function in Coq

For a review of the basics of Coq, refer to the following chapter of „Software Foundations“:
https://softwarefoundations.cis.upenn.edu/lf-current/Basics.html

```
Inductive shape :=
| Square : nat -> shape
| Rectangle : nat -> nat -> shape.

Fixpoint isSquare (s : shape) : bool :=
  match s with
  | Square _ => true
  | Rectangle _ _ => false
  end.

Definition ex_square := Square 2.
Definition ex_rectangle := Rectangle 2 3.

Compute (isSquare ex_square).
Compute (isSquare ex_rectangle).

Lemma ex_square_is_square : isSquare ex_square = true.
Proof.
  simpl. reflexivity.
Qed.
```

Extend the file with a function `circumference` which takes a shape and returns the circumference of the object. Compute the circumferences of the given examples. Formulate and prove two lemmas which state that the `circumference` function applied to `ex_square` and `ex_rectangle` yield the correct results.

2.3 Use Github to upload your homework

Check that your Code runs through, commit your changes and push them to the repository. The last commit before the handin deadline will be your final submission. (You will be able to push after the deadline is over, but these changes will not be graded.)