Developing a Web-Based Hoare Logic Proof Assistant

-Bachelor Project Description-

Flavio Goldener

Supervisors:
Malte Schwerhoff, Peter Müller
ETH Zürich, Switzerland

25.11.2015

1 Motivation

Hoare logic (also known as axiomatic semantics) are a family of formal systems for reasoning about properties of programs, for example, correctness and termination. Hoare logic is a derivation system, where proofs are represented as proof trees, but when proving program properties on paper, such proofs are usually represented as proof outlines (see Figure 1), which is more convenient to work with.

$\{true\}$
$\models$
$\{42 = 42\}$
$y := 42$
$\{y = 42\}$
skip;
$\{y = 42\}$
x:=y;
$\{x = 42\}$
$\models$
$\{x \geq 0\}$

Figure 1: Proof outline

Working with proof outlines can be quite involved, though:

- Due to the possibility of doing both forward and backward reasoning, it is not always obvious, what the next proof step is. Doing a step which is not correct might only later turn out to be wrong and to undo all the work done in between is quite cumbersome, especially when the proof outline is done on paper.
- Where to place an entailment, i.e. when to apply the rule of consequence, is not trivial. In the example, the first entailment could be forgotten and therefore the proof outline would not be correct.

At ETH, computer science students first hear about Hoare logic in the lecture "Formal Methods and Functional Programming", which is part of the Bachelor's degree programme. The learning experience of the students could be improved by developing a web-based Hoare logic proof assistant, which is easily accessible and provides immediate feedback, which this bachelor's thesis intends to do.

2 Core Goals

The core goal of this bachelor's thesis is to develop a web-based Hoare logic proof assistant for teaching purposes, which supports students in becoming familiar with Hoare logic proof outlines. The main focus of the proof assistant should be on usability and feedback, rather than expressiveness and completeness.

The project should address the following tasks:

- Develop a web application which is intuitively usable and provides immediate feedback, such as warnings in case of incorrectly applied rules.
- Implement two modes of constructing proof outlines:
  - Select a statement and a rule and let the tool apply the rule to get the correct pre-/ postcondition, depending on which one is missing.
  - Let the user create the proof outline and let a checker verify, whether or not the rules were correctly used.
- Develop the functionality necessary for parsing the proof outling, storing it as an abstract syntax tree and pretty-printing it in the browser.
- Choose a suitable data structure for representing partial proof outlines in a way that supports both forward and backward reasoning, where applicable.
- Support partial and total correctness.
- Support the functionality for downloading the proof outline as a PDF document.
- Implement a history view in which the user can review previous steps of the proof outline.

3 Extension goals

- Save and load (unfinished) proofs (including the undo stacks).
- Record usage statistics, e.g. number of users, usage time, error messages, undo patterns.
- Language extensions (such as method calls and local variables).
- Support non-linear undoing, e.g. undo only the work done in the if-branch, although it was done before working on the else-branch.
- Look into ways how entailments could be verified automatically.