CS 6390 Final: Imperative Programs, Semantics, and Types

Prof. William Harris   Ms. Sulekha Kulkarni

December 5, 2015

1 Instructions

Each problem is either a selected exercise from the text or a custom problem written especially for the final. The selected exercises are chosen from the following chapters: `Equiv`, `Hoare`, `Smallstep`, `Stlc`. Please submit a version of each of these chapters with your answer for each of the selected problems listed in section 2. You can choose to leave as admitted any other definitions and proofs in each chapter, just as long as the chapter module is accepted by Coq.

Answers for each of the custom problems should be included in a fresh Coq module named `Final.v`. You can write `Final.v` to include any other modules from the text that you choose. Instructions for each of the custom problems are given in the section 3.

The entire exam is out of 175 pts, with 10 bonus pts available. The number of points possible per problem is given next to each problem.

2 Selected Exercises (75 pts; 10 bonus pts)

Equiv (15 pts)

- `himp_ceval` (5 pts)
- `havoc_swap` (5 pts)
- `havoc_copy` (5 pts)

Hoare (15 pts)

- `himp_hoare` (15 pts)

Smallstep (45 pts)

- `eval__multistep` (15 pts)
- `step_eval` (15 pts)
- `multistep__eval` (15 pts)

Stlc (10 bonus pts)

- `substi` (bonus: 10 pts)

3 Custom Problems (100 pts)

Imperative Summation (50 pts)

Please write a decorated IMP command `c` such that the following Hoare triple is valid:

$$\{\{\texttt{X} = n\}\}\ \texttt{c}\ \{\{2 * \texttt{Y} = n * (\texttt{S}\ n)\}\}$$

All arithmetic expressions in `c` must be constructed from only variables, constants, and the plus operation. You can simply write `c` in plaintext as a comment in `Final.v`: just make sure to use the informal syntax for decorated programs given in `Hoare2.v`.

You can get 15 pts for writing a command that is correct and 35 pts for proving its correctness.

For each implication between assertions $\texttt{P} \longrightarrow \texttt{Q}$ in `c`, include $\texttt{P} \longrightarrow \texttt{Q}$ in `Final.v` as a lemma with a proof.

Types (50 pts)

In this problem, you'll extend the language of terms `tm` defined in `Types.v` with a construct for representing a sum of terms.

Syntax (5 pts): Extend the syntax of `tm` so that for all terms $t_0, t_1 \in \texttt{tm}$, $\texttt{tplus}\ t_0\ t_1 \in \texttt{tm}$.

Small-step semantics (5 pts): Extend the small-step semantics of `tm` defined by `step` so that:

- For each numeric value $v \in \texttt{nvalue}$, $\texttt{tplus}\ \texttt{0}\ v \Rightarrow v$.
- For all numeric values $v_0, v_1 \in \texttt{nvalue}$, $\texttt{tplus}\ (\texttt{tsucc}\ v_0)\ v_1 \Rightarrow \texttt{tplus}\ v_0\ (\texttt{tsucc}\ v_1)$.

Types (40 pts): Extend the typing relation of `tm` defined by `has_type` so that for all terms $t_0, t_1 \in \texttt{Nat}$, $\texttt{tplus}\ t_0\ t_1 \in \texttt{Nat}$. Prove progress and preservation for `has_type` extended for `tplus`.

You can write the proofs any way you choose, but one manageable workflow would be to first finish the proofs of progress and preservation for `tm` included in `Types`, and then extend each proof with cases for terms constructed from `tplus`.

You can get 5 pts for writing correct type inference rules and 35 pts for proving their correctness.