Capstone Proposal

Student: Jeremy Yew, A0156262H Supervisor: Prof Olivier Danvy Title: Editorial Support for the Coq Proof Assistant Subject Areas: Functional programming, mathematical logic and proofs, data structures, language parsing.

Challenges:

To build a tool that applies user-specified syntax rules to a Coq script, in order to build muscle memory for good programming habits. This will involve:

- Programming a command in the Emacs editor, using emacs-lisp or other languages.
- Programming a language parser that can apply both abstract and concrete syntax to a Coq script.
- Programming a grammar parser that can accept a user-provided input grammar, and apply the grammar to a Coq script.
- If writing a program external to Emacs, ensuring the command works across operating systems, i.e. calls the appropriate system command to run the external program outside of Emacs.

Scope:

- Learning how to assist student's CS education by instilling good programming habits.
- Learning how to program Emacs commands.
- Learning how to write a language parser that applies specific syntax rules.
- Learning how to write a grammar parser that interprets user-specified syntax rules and applies them.

Expectations associated with grade achievement:

Note: Unless otherwise mentioned, all goals below are aimed to be completed by Semester 1 unless otherwise stated, with Semester 2 focused on additional features and report-writing/presentation prep. While it is a little ambitious to aim to complete all core code by Semester 1, it is important to design some buffer time to accommodate unpredictable technical challenges.

To fulfill the following tasks, to document their implementation and associated design decisions, and to gather feedback on the tool from the users (Professor and students):

- Program an emacs command that takes a pointer to a Coq script as input, and tells the user whether or not the script adheres to the syntax of the Coq logical language.
- Program the command to apply abstract syntax rules on the script, such as: "Proofs should only use a specified subset of libraries and tactics", and warn the user of transgressions.
- Program the command to apply concrete syntax rules on the script, such as: "Rewrite commands should specify the direction of rewrite, and rewrite rules should have all their arguments supplied", and warn the user of transgressions.
- Program the command to parse one or more input grammars that specify syntax rules.

Optionally (Sem 2):

- Program the command to be more interactive, for example:

- providing intermediate feedback at every step of the proof
- highlighting transgressions inside the script (as opposed to messages in the response buffer)
- providing suggested corrections

Time allocation:

Meetings with Prof Danvy: 1-1.5 hour a week of one-to-one discussion of project progress. Thursdays at 1pm.