Automatic Library Compilation and Proof Tree Visualisation for Coq Proof General

Hendrik Tews

Technische Universität Dresden

Third Coq Workshop, August 26, 2011

Outline

Automatic Library Compilation

Goals

- make `coq_makefile` obsolete for the ordinary Coq user
- Proof General keeps a consistent Coq session, while the user
  - freely changes arbitrary files
  - synchronises files with co-workers
    (possibly through some version control system)
  - asserts text in arbitrary buffers

See `http://askra.de/software/multiple-coq/`

Demonstration

(using the contribution *ConCat* from Amokrane Saïbi)

New Features

- save files and compile libraries before `Require` commands are processed
- Lock ancestors
- Restart `coqtop` when the active scripting buffer changes
- two compilation modes:

  Internal Proof General calls `coqdep` and `coqc` itself

  External Proof General starts compilation
  e.g., `make -C dir lib.vo`
- Implementation is rather straightforward
- current cvs head, will be released with Proof General 4.1.

Problems

- Incompatible with current customisation habbits
  - put nonstandard load path into `coq-load-path`
  - delete `-I` options from `coq-prog-name` and `coq-prog-args`
  - enable `coq-compile-before-require`
- Impossible to map `Required` modules to file names
  - `Locate Library` fails if `.vo` file is missing
  - `Locate File` fails for modules with logical path'
    e.g., `Coq.Init.Wf` vs. `Coq.Program.Wf`
  - `coqdep` does not process `Add LoadPath`

Limitations

- Currently no support for `Declare ML Module` and `Require "x.vo"`
- Synchronous compilation in internal mode locks Emacs
- a change in an ancestor will change the current scripting buffer
- file modification times are compared with a precision of 1 second

Outline

Demonstration

Features

- proof tree visualisation in external window
- expands and shrinks when commands are asserted or undone
- update sequents with instantiated existentials (as far as possible)
- branches are marked with different colours: proved, current, open, admitted
- full sequent or proof command text can be displayed in separate windows
- cloned windows freeze a snapshot of current proof tree

The Need for Patching Coq

Problems

- very difficult or impossible to reconstruct the proof tree externally (which goals are new and which are old?)
- no information about which existential variables got instantiated and which sequents changed because of that
- no information about what to delete after retract/Backtrack

Need a patched version of Coq that

- annotates sequents with unique tags (`make_evar`, `evar_declare` and `mk_goal` increment a counter and store its value in the created `evar_info`.)
- print tags and the list of instantiated existentials for option `-emacs` or `-emacs-U`

Implementation

- divided between Proof General and the program **Prooftree**
- Information flow is one-way: From Proof General to Prooftree

Proof General part

- parse prover output for new sequents and existentials
- keeps as internal state
  - hash of known sequent tags
  - mapping from existentials to sequent tags
  - complete undo information for the previous two items
- sends text messages to Prooftree
- code is divided in a general and a Coq specific part
- available in branch `ProofTreeBranch` in Proof General cvs

Prooftree

- builds and draws tree
- keeps complete undo information
- about 5000 lines of Ocaml + Lablgtk code

Installation

- see `http://askra.de/software/prooftree/`
- need consistent versions of
  - Coq ID patches
  - `ProofTreeBranch` version of Proof General
  - Prooftree
- compile Coq and Prooftree manually
- configure Emacs to use `ProofTreeBranch` and the patched Coq version

Outline

Wish List

- a tool mapping logical path' to file names, working as filter (stdin $\longrightarrow$ stdout)
- unique sequent ID's
- print instantiated existentials
- `Show ID` *num* for displaying the subgoal with ID *num*
- fix `coqdep` (`Add LoadPath ...`)

Thanks for your attention

Comments / Questions are welcome at

proofgeneral-devel@inf.ed.ac.uk

and

coq-club@inria.fr