CoqPie: A new GUI for the COQ Theorem Prover

Kenneth Roe http://www.cs.jhu.edu/~roe The Johns Hopkins University

Motivation

Developing complex theorems using COQ is extremely tedious

The design of CoqPIE is based on an informal analysis of pain points in proof development.

(1) One often finds errors in the statement of a theorem while developing a proof. Changing the statement often requires the proof script to be updated
(2) Often it is useful to quickly review earlier goal/hypothesis states. Existing IDEs require the Coq prover itself to backup or move forward in a script. On a complex proof each step can take 30 seconds to backup or reevaluate.
(3) LTac--the scripting langauge for Coq has many holes. Many simple rules cannot be expressed
(4) Navigating to specific definition declarations can be difficult if there is a large amount of proof script code.

Overview of implementation

* Implemented using Python and Tk

* Parser developed for Coq syntax
  CoqPIE parses all source files
  Enables implementation of intelligent replay and tactics

* CoqPIE preprocesses Coq files.
  (1) All intermediate goals and hypotheses saved.
  (2) Data collected for an entire project and saved in a single .pp file
  (3) CoqPIE automatically updates data cache as edits are performed

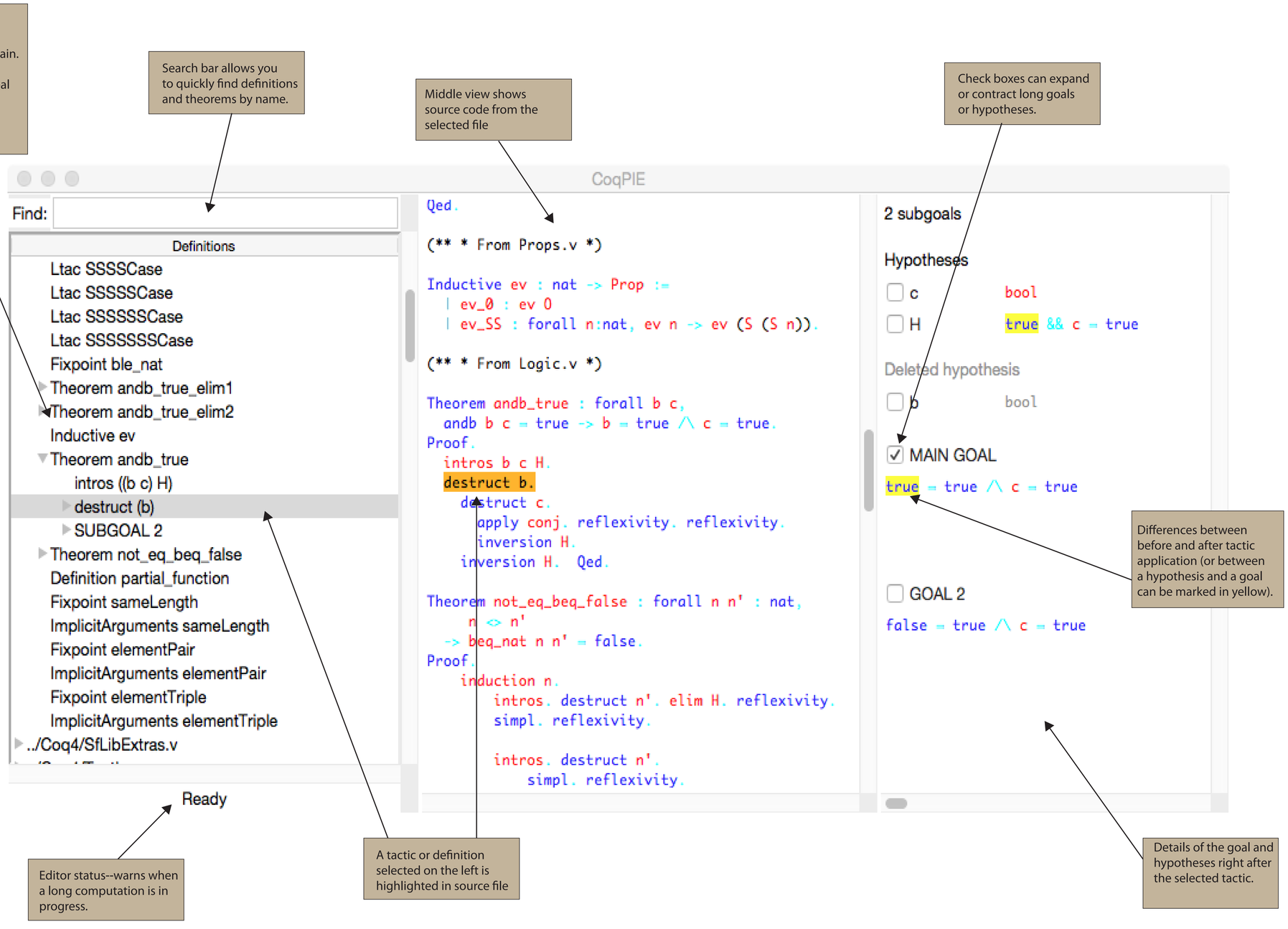
* CoqPIE maintains a relationship between its AST representation and the source file text
  (1) AST used to create definition/theorem browser at the left
  (2) AST incrementally updated as the user makes edits

* User free to edit Coq source files using another editor. However, the .pp file will have to be regenerated which might be tedious

* CoqPIE implements intelligent replay and many editing tactics that use information in the AST

The shortfalls of CoqIDE and Proof general

(1) Both based on the concept of giving control over the current position of CoqTop in the source file
(2) This means that skimming a proof is tedious as this requires Coq to change its state
(3) Replay requires remembering the goals and hypotheses of the proof before editing. This often makes updating confusing



Dependency information

When a definition or lemma is edited, it can impact the validity of other declarations that depend on it.

* If the definition or the statement portion of the theorem is still valid, then later definitions can still be edited
* CoqPIE automatically marks declarations that have been invalidated.
* Intelligent replay can be used to fix the proof scripts of any proof that is marked "invalid"
* One can use Coq to check proofs and definitions even when an earlier proof script is invalid.

Mitigating Coq performance issues

One of the biggest sources of productivity problems is the speed of the Coq theorem prover. Complex proofs can take hours (or even days) to fully verify. We do not make Coq faster but we do minimize the need for Coq to do work

(1) All intermediate goals are cached. Simply reviewing a proof does not invoke Coq.
(2) CoqPIE provides tactics to break up large proofs into lemmas--this often improves the performance of Coq
(3) CoqPIE will replace a proof script with admit if you are simply jumping over an entire theorem to get to something much later

Replay

Often the process of developing a theorem reveals errors in the statement of a theorem. When that statement is changed, the script needs to be adapted. Changes may involve the following:

(1) Removing proofs for subgoals that vanish
(2) Creating stubs for new subgoals
  Often the error discovered in a theorem declaration is a missing antecedant
(3) Updating hypothesis names in tactics.
  Adding an antecedant may shift down hypothesis numbers. For example, "inversion H10" may need to become "inversion H11"

Replay works by stepping through a script. At each step, both the AST representation of the old goal/hypotheses and the new goal/hypotheses is available for analysis. This enables CoqPIE to automatically figure out how hypothesis names have shifted.

The process is semi-automatic. User intervention may be needed to prove a newly added subgoal or to update the expression inside an "assert"

Difference highlighting

Because a goal can be very complex. At each step differences are highlighted in yellow to show what changed.

Project Status

A prototype of the Coq parser and a tool that can read and display the definitions in a Coq file is complete. The tool has facilities to preprocess the scripts and save intermediate data. Work is now being done to allow editing of the proof scripts in CoqPIE. It is anticipated that by Summer 2015 a functional version of the tool will be available.