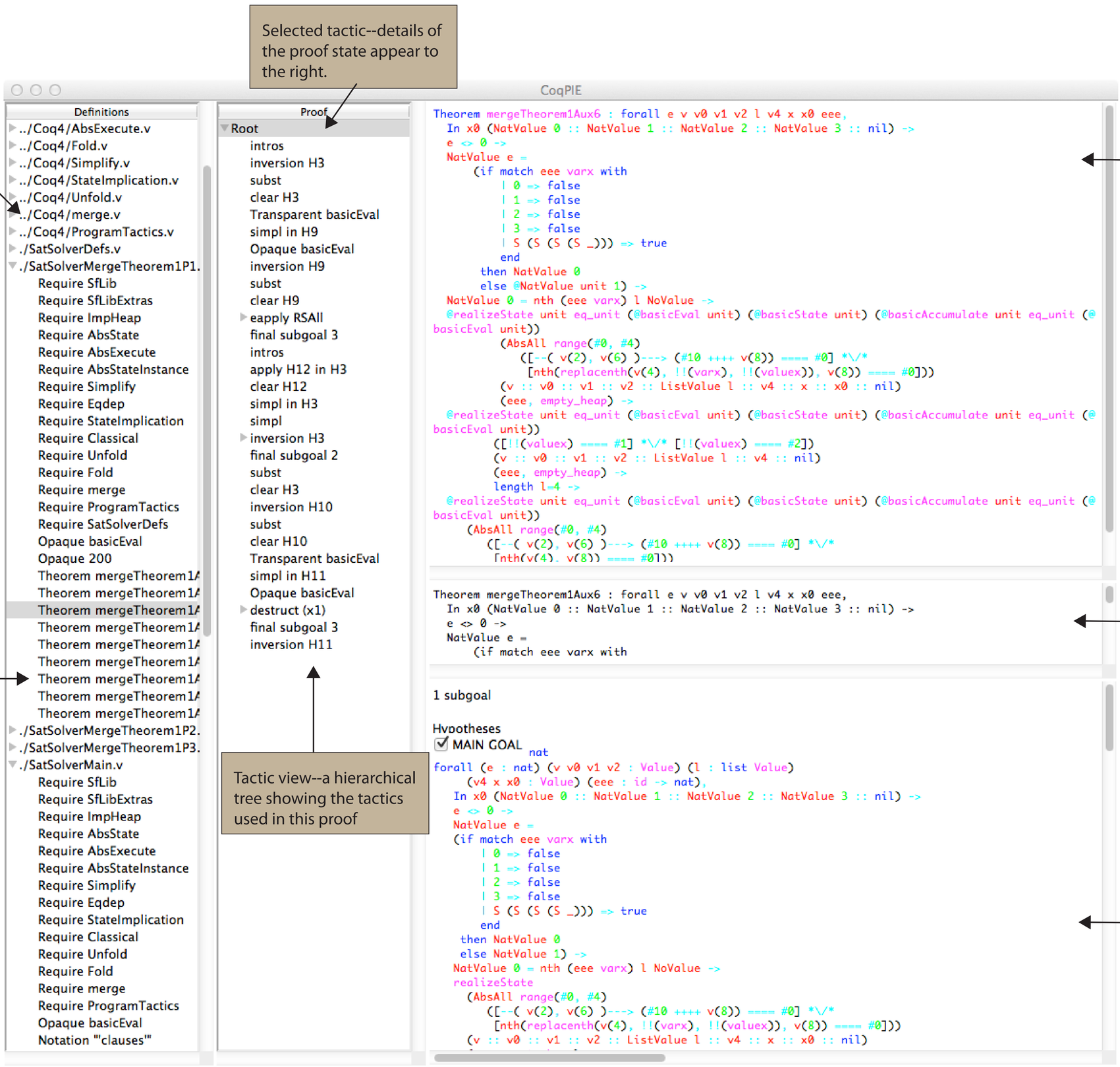
CoqPIE: A new type of UI for Coq

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



Motivation

Developing complex theorems using COQ is extremely tedious

CoqPIE based on an analysis of pain points in proof development:

(1) Errors in the statement of a theorem while developing a proof
-- Changing the statement often requires the proof script to be updated

(2) Useful to quickly review earlier goal/hypothesis states.
-- Existing IDEs require the Coq prover itself to backup for move forward in a script.
-- On complex proof each step can take 30 seconds to backup or reevaluate.

(3) LTac--the scripting langauge for Coq--is incomplete

(4) Navigating to specific definition declarations is difficult if the text in the script file is long.

Overview of implementation

* Implemented using Python and Tk

* Parser developed for Coq syntax
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) Edits done in CoqPIE only change the relevant portions of the user's source file
(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

Dependency information

* CoqPIE parses all source files

* Dependency information between proofs and definitions is generated and maintained

* When a definition or proof is modified, everything that depends on the entity is marked as "invalid"
(1) If the definition or the statement portion of the theorem is still valid, then later definitions can sttill be edited
(2) Intelligent replay can be used to fix the proof scripts an any proof that is marked "invalid"
(3) One can delay fixing proof scripts as CoqPIE automatically replaces the script with "admit" if the user is fixing a later proof

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

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

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 discoveredin 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.

The process is semi-automatic. User intervention may be needed to prove a newly added subgoal or to update the lemma 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

As of the time of this submission, 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 POPL there will be functional version of this tool.

Acknowledgements

The author would like to thank Scott Smith for his feedback.