Proviola
A Tool for Proof Re-animation

Carst Tankink
Joint work with Herman Geuvers,
James McKinna and Freek Wiedijk
Supported by NWO project: “MathWiki”

Institute for Computing and Information Science
Faculty of Science, Radboud University Nijmegen
and
Eindhoven University of Technology
The Netherlands

15th July 2010
(** Having made the assertion, we can also ask Coq to verify it, like this: *)

Proof.
simpl.
reflexivity.
Qed.
(** Having made the assertion, we can also ask Coq to verify it, like this: *)

Proof.
simpl.
reflexivity.
Qed.
(** Having made the assertion, we can also ask Coq to verify it, like this: *)

Proof.
  simpl.
  reflexivity.
Qed.

= Proviola!

Picture by J&R Moviola, CC BY-SA
Demo

Resources, examples:
http://mws.cs.ru.nl/proviola
Motivation

- Mathematics should be explained, not just presented.
- Repository of formal proof should include explanation.
Assumptions

- Two roles: Author and Reader.
- Tactic-based prover: there is a notion of state.
How is formal proof communicated?

1. Author writes formalization using a prover → proof script.
2. Reader obtains script.
3. Reader reads script using local installation of prover.
How is formal proof communicated?

1. Author writes formalization using a prover → proof script.
2. Reader obtains script.
3. Reader reads script using local installation of prover.
How is formal proof communicated?

1. Author writes formalization using a prover $\rightarrow$ proof script.
2. Reader obtains script.
3. Reader reads script using local installation of prover.
What’s the problem?

- Author used to the prover, reader (in general) not.
- Reader needs to see state transformations, requiring:
  1. installation of a prover.
  2. computation of state based on script.
- Possible solution for installation: online provers (*e.g.* ProofWeb)
  - Still has a computational overhead.
What’s the problem?

- Author used to the prover, reader (in general) **not**.
- Reader needs to see state transformations, requiring:
  1. installation of a prover.
  2. computation of state based on script.
- Possible solution for installation: online provers (*e.g.* ProofWeb)
  - Still has a computational overhead.
What’s the problem?

- Author used to the prover, reader (in general) not.
- Reader needs to see state transformations, requiring:
  1. installation of a prover.
  2. computation of state based on script.
- Possible solution for installation: online provers (e.g. ProofWeb)
  - Still has a computational overhead.
What do we need?

In summary: the reader should **zip through the proof**:

- Fast access to proof state.
- No prover necessary for viewing.
- Usable over the web.

Additionally: No overhead for the author.
Proof movies

- Our solution: Proof Movies
- Stores proof states with script, in frames.
  - Trades space for time.
  - A few KB, on average: blowup $6 \times$
- Reader’s access to state becomes lookup, not computation.
- Construction is automated: no overhead for author.
Proof movies

- Movie is implemented as XML.
- Prototype implementation based on Coq/CoqIDE.

```xml
<frame frameNumber="25">
  <command>
    simpl.
  </command>
  <response>
    1 subgoal
  </response>
</frame>
```
Watching a movie: Proviola

- ‘Tool’ for reviewing: Proviola (transformation to HTML).
- Simplifies the reader’s part of the use case.
- Instead of obtaining a proof script, obtain a movie.
- Watch, not recompute: point to reveal state.
Building Movies: Camera

- Making movies is recording prover input and output.
- Input: a proof script, cut into commands.
- Output: captured directly from the prover, after each command.
  - The prover is still a part of the workflow.
Movies Extended for the Web

- Just XML: presentation in HTML is easy.
- Also possible to add other (meta)data: pretty-printing.

In general

- Extend frames with additional data.
- Programs can read the movie, and add data.
- In effect: services based on movies.
Movies Extended for the Web

- Just XML: presentation in HTML is easy.
- Also possible to add other (meta)data: pretty-printing.

In general
- Extend frames with additional data.
- Programs can read the movie, and add data.
- In effect: services based on movies.
Towards a Movie-based MathWiki

- Show movies with documentation to readers of Wiki.
- Movies provide a fast overview of files in the repository.

Roadmap:

- Support other provers:
  - For the author: implement camera.
  - For the reader: rendering.
- Generate pages: just invoke camera.
- Add commentary track to narrate movie.
  - Refer to (and display) previous, future items.
  - Simple implementation: Coqdoc in movies.
  - More complex: give authors UI.
- Editing the movie.
Towards a Movie-based MathWiki

- Show movies with documentation to readers of Wiki.
- Movies provide a fast overview of files in the repository.

Roadmap:

- Support other provers:
  - For the author: implement camera.
  - For the reader: rendering.
- Generate pages: just invoke camera.
- Add commentary track to narrate movie.
  - Refer to (and display) previous, future items.
  - Simple implementation: Coqdoc in movies.
  - More complex: give authors UI.
- Editing the movie.
Movies in MathWiki — Dynamic movie editing

- Author is *also* a reader, also benefits from fast lookup.
- Author writes movie, prover works in the background.
  - Prover is a service.
- Similar to document-centered approaches & ProofGeneral
- Not just a proof script.

- A true Wiki: make authors out of readers.
Author is also a reader, also benefits from fast lookup.

Author writes movie, prover works in the background.
  - Prover is a service.

Similar to document-centered approaches & ProofGeneral

Not just a proof script.

A true Wiki: make authors out of readers.