

Teaching logic using a web interface for Coq

October 31

Cezary Kaliszyk

Radboud University Nijmegen

cek@cs.ru.nl

Presentation Plan

● Presentation Plan

- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Web Interface
 - ◆ Motivation
 - ◆ Requirements
 - ◆ Architecture
 - ◆ Efficiency and Security
- Teaching Logic
 - ◆ Tactics
 - ◆ Graphical presentation of proofs
 - ◆ Problem set

Why the Web?

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Popular - No installation or configuration
- e-mail clients, calendars, maps, chats, word processing, ...
- wikis and Wikipedia
- some tools for proofs

Web Technologies

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Plugins: Java, Flash
- JavaScript
- DOM
- XmlHttp
- Asynchronous DOM modifications
 - ◆ sometimes called *AJAX* or *Web Application*

Proof Assistants

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Often complicated to install
- Proofs are developed locally
 - ◆ Versioning systems
- Static web pages are generated to display proofs on the web
 - ◆ tactic-mode proofs

Architecture (1/2)

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

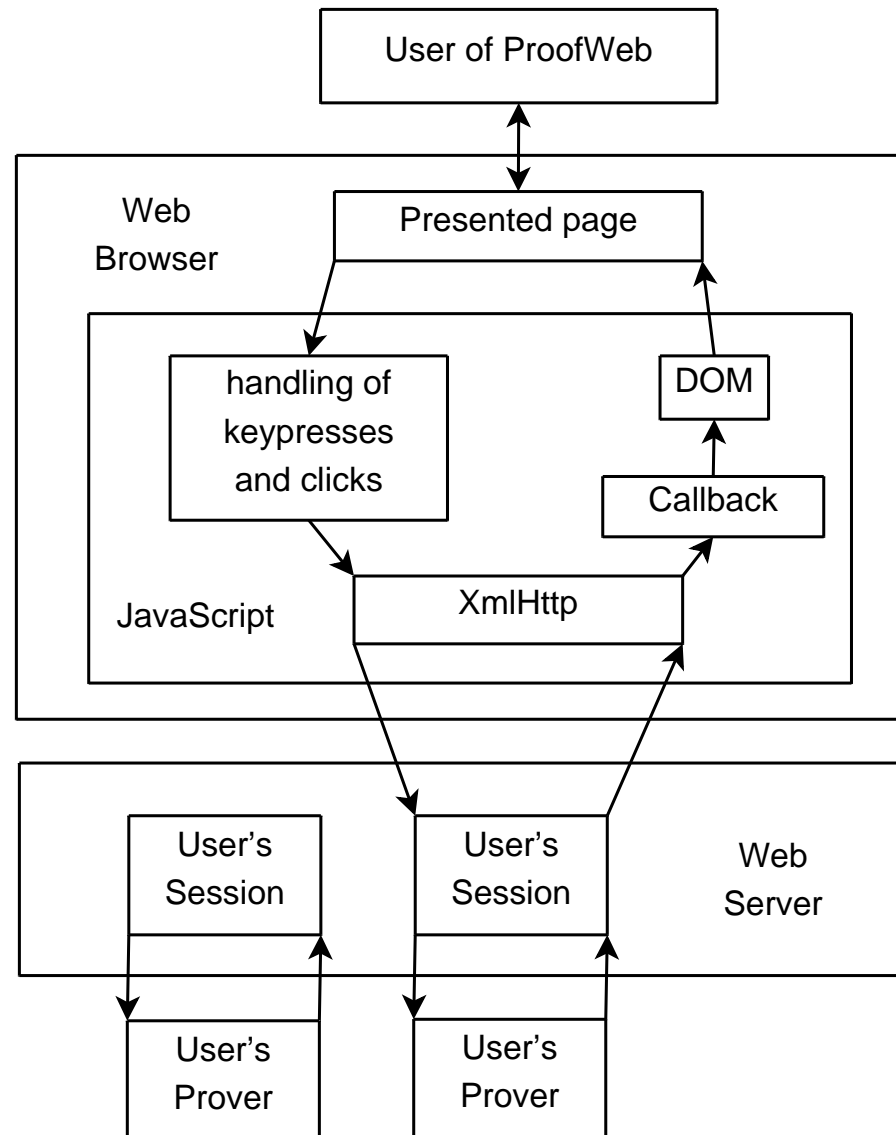
Teaching logic

- Lightweight client part in browser
 - ◆ User does not need to install anything
- Specialized web-server
 - ◆ Prover sub-processes
- Minimal communication

Architecture (2/2)

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic



Implementation of a prototype

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Client part: 30kB of JavaScript and HTML
- Server part: 800 lines of OCaml code, uses OCamlHttpd runs prover subprocesses
- Tested with Mozilla based browsers, Internet Explorer and Opera
- On any platform/architecture one can easily access the interface
 - ◆ No java, plugins installations or privileges are required

User Security and Efficiency

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- System and browser errors
- Efficiency of an interpreted language
 - ◆ Browser efficiency
- Network latency
 - ◆ TCP Ping time

Server Security

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Crackers, “Too-powerful” provers
- Availability of work and files and equal sharing of all resources
- The communication mechanism
- Compilation and dependencies

Server Security

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Crackers, “Too-powerful” provers
 - ◆ Sandboxing
- Availability of work and files and equal sharing of all resources
 - ◆ disk quota, CPU quota, memory quota
- The communication mechanism
 - ◆ HTTPS
- Compilation and dependencies

Server Efficiency

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Depends on:
 - ◆ Number of users, server configuration, provers, usage of automated techniques, . . .
- Possible to start provers on different machines
- We expect compilation of dependencies to be the main bottleneck in bigger projects (more in Pierre's talk)

Project parts

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- Example of a tactic
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- Proofweb in practice
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

- programming the interface
- tactics for first-order logic
- graphical presentation of proofs
- a problem set
- course notes / manual

Example of a tactic

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- **Example of a tactic**
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- Proofweb in practice
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

disjunction elimination:

```
Ltac dis_el X H1 H2 :=
  match X with
  | ( _ \/ _ ) =>
    assert X;
    [ idtac |
      match goal with
      | x : X |- _ =>
        elim x; [intro H1 | intro H2]; clear x
      end
    ]
  | _ => fail "The first argument is not a disjunction"
end.
```

Example of an exercise

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- Example of a tactic
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- Proofweb in practice
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

```
Theorem exercise_024 : (A \\/ B) /\ ~A -> B.
```

```
Proof.
```

```
imp_in z.
```

```
dis_el (A \\/ B) y1 y2.
```

```
con_el1 (~A).
```

```
ass z.
```

```
neg_el A.
```

```
con_elr (A \\/ B).
```

```
ass z.
```

```
ass y1.
```

```
ass y2.
```

```
Qed.
```


Example of a Fitch proof tree

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- Example of a tactic
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- Proofweb in practice
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

1	H: $(\forall x:D, P x \rightarrow Q x) \wedge (\exists x:D, P x)$	assumption
	x	
2	HP: $P x$	assumption
3	$\forall y:D, P y \rightarrow Q y$	$\wedge e_1 1$
4	$P x \rightarrow Q x$	$\forall e 3$
5	$Q x$	$\rightarrow e 4,2$
6	$\exists x:D, Q x$	$\exists i 5$
7	$P x \rightarrow \exists x_0:D, Q x_0$	$\rightarrow i 2-6$
8	$\forall x:D, P x \rightarrow \exists x_0:D, Q x_0$	$\forall i 2-7$
	...	
9	$(\forall x:D, P x \rightarrow Q x) \wedge (\exists x:D, P x)$	
10	$\exists x:D, P x$	$\wedge e_2 9$
11	$\exists x:D, Q x$	$\exists e 8,10$
12	$(\forall x:D, P x \rightarrow Q x) \wedge (\exists x:D, P x) \rightarrow \exists x:D, Q x$	$\rightarrow i 1-11$

Proofweb in practice

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- Example of a tactic
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- **Proofweb in practice**
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

The system has been and will be used in two types of courses so far:

■ Graduate courses:

- ◆ Logical Verification (Amsterdam)
- ◆ Type Theory (Nijmegen)
- ◆ Master Class on Type Theory and Proof Assistants (Nijmegen)

■ Undergraduate courses:

- ◆ Beweren en Bewijzen (Nijmegen)
- ◆ Introduction to logic (Amsterdam)

Coq (with Proofweb) vs. Jape (or cousins)

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- Example of a tactic
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- Proofweb in practice
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

- Prepares for later use of Coq
- Message to students: no plaything
- A broader system (CNF, satisfiability, logic with numbers, modal logic)
- No random proving by clicking
- Saving (incomplete) proofs is possible
- Centralized architecture

Work in progress

- Presentation Plan
- Why the Web?
- Web Technologies
- Proof Assistants
- Architecture (1/2)
- Architecture (2/2)
- Implementation of a prototype
- User Security and Efficiency
- Server Security
- Server Efficiency

Teaching logic

- Project parts
- Example of a tactic
- Example of an exercise
- Example of a proof tree
- Example of a Fitch proof tree
- Proofweb in practice
- Coq (with Proofweb) vs. Jape (or cousins)
- Work in progress

The needed infrastructure for creating a Wiki.

- Many provers and versions of provers
- Security Policy
- Teacher interface vs side utils in Wikis
- Standard protocol