Integrating Graphical Proofs in Coq

A description of the coq-actema system

Pablo Donato    Benjamin Werner    Kaustuv Chaudhuri

Partout team — LIX

CoqPL 2023

Boston
Goal: Make proof assistants *easier* to use

- **Intuitive** and **discoverable** for newcomers
- **Productive** and **beautiful** for experts
Goal: Make proof assistants easier to use

- Intuitive and discoverable for newcomers
- Productive and beautiful for experts

For now, focus on common logical heart:

*Intuitionistic First-Order Logic (iFOL)*
Goal: Make proof assistants easier to use

- Intuitive and discoverable for newcomers
- Productive and beautiful for experts

For now, focus on common logical heart:

*Intuitionistic First-Order Logic (iFOL)*

Disclaimer: WIP, still at an experimental stage...
GRAPHICAL PROOFS
“A demo is worth a thousand words...”
Paradigm

- Fully graphical: no textual proof language
- Both spatial and temporal:

  proof = gesture sequence

- Different modes of reasoning with a single “syntax”:

  Click ⇔ introduction/elimination
  Drag-and-Drop ⇔ backward/forward

Sound and complete for iFOL!
INTEGRATION WITH COQ
$\mathcal{G}$: goal list

Kernel -> Plugin

Source
\( \mathcal{G} : \text{goal list} \rightarrow \text{translate}(\mathcal{G}) : \text{agoal list} \)

1. Kernel
2. Plugin
3. Source
Protocol

\( \mathcal{G} : \) goal list

\( \text{translate(} \mathcal{G} \text{)} : \) agoal list

\( \mathcal{A} : (\text{action} \times \text{int}) \text{ list} \)
\( G \) : goal list

\( \text{translate}(G) \) : agoal list

\( \mathcal{A} \) : (action * int) list

\( \text{compile}(\mathcal{A}) \) : tactic
Protocol

```
translate(\mathcal{G}) : \text{agoal list}
```

```
A : (\text{action} \times \text{int}) \text{ list}
```

Plugin

Source
Protocol (non-interactive)

Plugin

Graphical Proof Database
Plugin

(G, id): aggoal * string

Graphical Proof Database

6/15
Protocol (non-interactive)

(G, id) : agoal * string
𝒜 : (action * int) list

Plugin

1

2

Graphical Proof Database

𝒜 : (action * int) list
Protocol (non-interactive)

\( (G, \text{id}) : \text{aggoal} \times \text{string} \)

\( \mathcal{A} : (\text{action} \times \text{int}) \text{ list} \)

Plugin

Graphical Proof Database

\( \mathcal{A} : (\text{action} \times \text{int}) \text{ list} \)
Protocol (interactive)

Plugin

Actema

User
Protocol (interactive)

$\mathcal{G}$ : goal list

Plugin → Actema

User

1
Protocol (interactive)

$\mathcal{G}$ : goal list

render($\mathcal{G}$) : HTMLDom

1. Plugin
2. Actema

User
Protocol (interactive)

Plugin → Actema

\[ G : \text{goal list} \]

render(\( G \)) : HTMLDom

User

S : gesture start

1 2 3
Protocol (interactive)

$G : \text{goal list}$

$\text{render} (\text{actions} (S)) : \text{HTMLDom}$

1. Plugin
2. Actema
3. User
4. User

$S : \text{gesture start}$
Protocol (interactive)

$\mathcal{G}$ : goal list

render(actions(S)) : HTMLDom

User

Plugin

Actema

1

4

5

E : gesture end
Protocol (interactive)

\[ \mathcal{G} : \text{goal list} \quad \text{render}(\text{actions}(S)) : \text{HTMLDom} \]

Plugin

1 2

Actema

4 5

User

(A, n) : action * int

E : gesture end

7/15
Protocol (interactive)

\( G : \text{goal list} \quad \text{render}(\text{actions}(S)) : \text{HTMLDom} \)

Plugin \quad \text{Actema} \quad \text{User}

\( (A, n) : \text{action} \times \text{int} \quad E : \text{gesture end} \)
• **Click** actions: standard Coq tactics

• **Drag-and-Drop** actions: \(\sim 3000\) lines of Coq/Ltac
  
  • Deep embedding of goal \(\Gamma \vdash C\) in FOL
  
  • Subterm selection as **paths**, i.e. **list** **nat**

• **Computational reflection** for *deep inference* semantics
  
  [Donato et al. (2022)]
  
  • Backward: new conclusion \(C'\)
  
  • Forward: new hypothesis \(A\)

• Final tactic = apply **soundness** theorem
  
  • Backward: \(\Gamma \Rightarrow C' \Rightarrow C\)
  
  • Forward: \(\Gamma \Rightarrow A\)
CONCLUSION
What are the most useful usecases of Actema?

- Proof exploration
- Educational setting
What were the *infrastructure* challenges/solutions?

- Interaction protocol that can handle **arbitrary goals and tactics** (still a WIP, because of FOL and notations)
- Generic protocol **independent of the specifics of Coq** (simpler with FOL)
- **Portable API with reusable boilerplate** for serialization on both sides (atdgen)
- **Linking external libraries** in Coq plugin, for serialization/HTTP (currently falls out of dune capabilities, need coq_makefile)
Related works (non-exhaustive)

- **Proof-by-Pointing** [Bertot et al. (1994)]
- **Subformula linking** [Chaudhuri (2013), Chaudhuri (2021)]
- **ProofWidgets** [Ayers et al. (2021)]
  - Framework for user-defined graphical notations
  - PA serves the GUI, instead of requesting from it
  - Relies on Lean’s metaprogramming capabilities
Future works

For more complex theories:

• Support arbitrary Coq notations (and more?)
• Selection-based lemma search
• Extend to HOL

For proof evolution:

• Translate graphical proof into readable and reusable tactic invokations (avoid paths)
• Replay/Edit graphical proof through animations
Future works

For more complex theories:

• Support arbitrary Coq notations (and more?)
• Selection-based lemma search
• Extend to HOL

For proof evolution:

• Translate graphical proof into readable and reusable tactic invokations (avoid paths)
• Replay/Edit graphical proof through animations

Thank you!
