Compiling graphical actions with deep inference

Pablo Donato   Benjamin Werner   Kaustuv Chaudhuri

Partout team — LIX

Scalp 2023

CIRM

Context

Goal: Make proof assistants *easier* to use

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

Context

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)

Context

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

`coq-actema`

"A demo is worth a thousand words…"

Paradigm

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

$$\text{proof} \quad = \quad \text{gesture sequence}$$

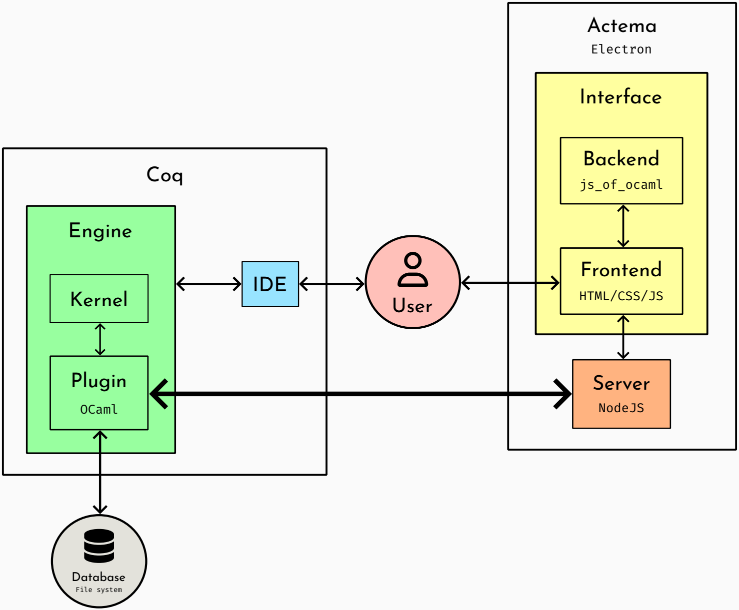
- **Different modes** of reasoning with a **single "syntax"**:

$$\text{Click} \quad \Longleftrightarrow \quad \text{introduction/elimination}$$
$$\text{Drag-and-Drop} \quad \Longleftrightarrow \quad \text{backward/forward}$$

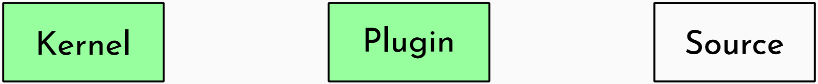
*Sound and **complete** for iFOL!*

Integration with Coq

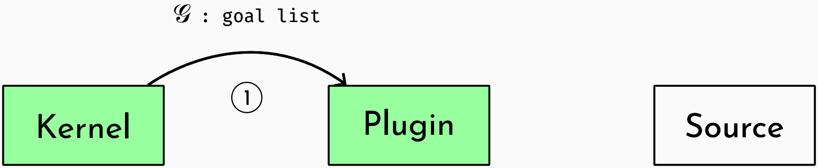
Architecture



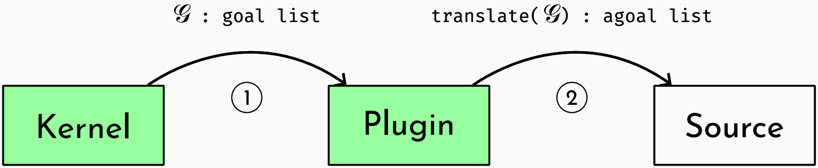
Protocol



Protocol

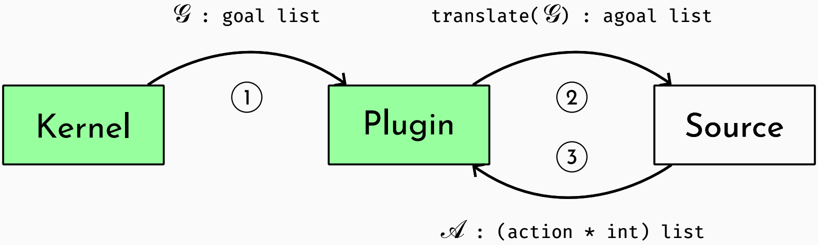


Protocol

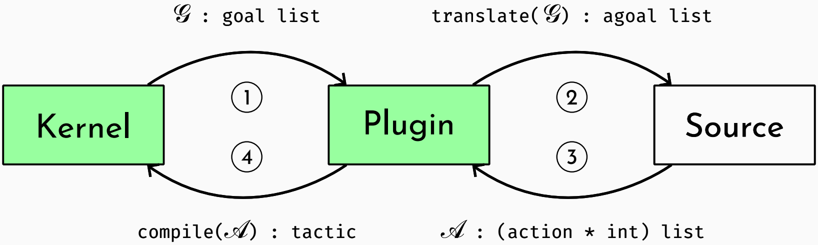


$\mathscr{G}$: goal list
translate($\mathscr{G}$) : agoal list
1
2
Kernel
Plugin
Source

Protocol

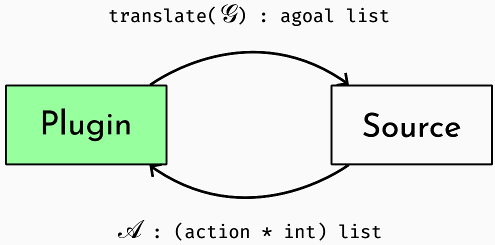

$\mathscr{G}$: goal list
translate($\mathscr{G}$) : agoal list
1
2
3
Kernel
Plugin
Source
$\mathscr{A}$: (action * int) list

Protocol

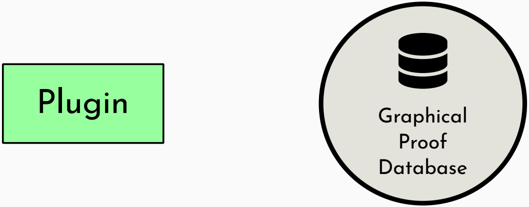


𝒢 : goal list
translate(𝒢) : agoal list
Kernel
1
4
Plugin
2
3
Source
compile(𝒜) : tactic
𝒜 : (action * int) list

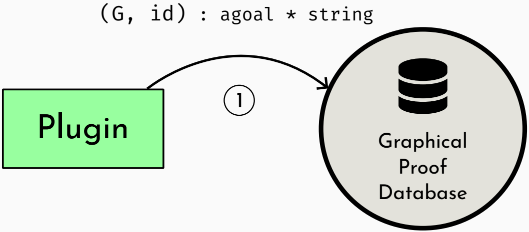
Protocol


translate($\mathscr{G}$) : agoal list
Plugin
Source
$\mathscr{A}$: (action * int) list

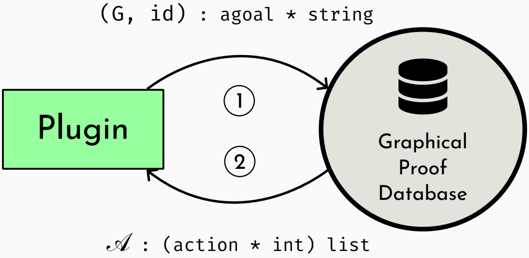
Protocol (non-interactive)



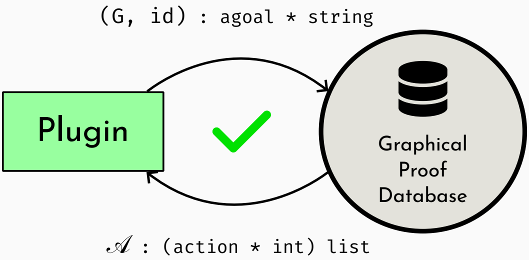
Protocol (non-interactive)



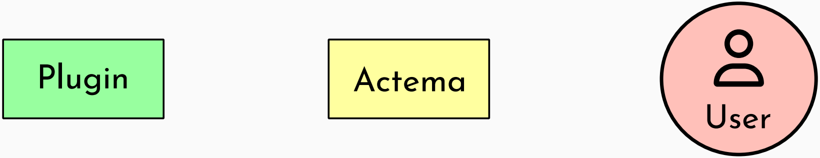
Protocol (non-interactive)



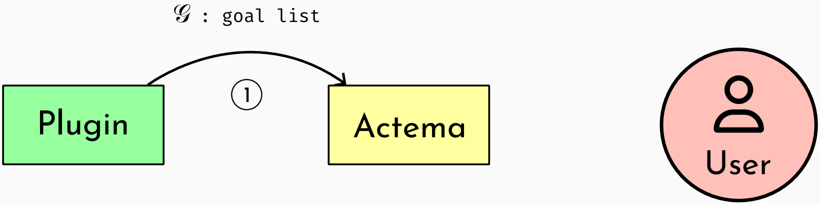
Protocol (non-interactive)



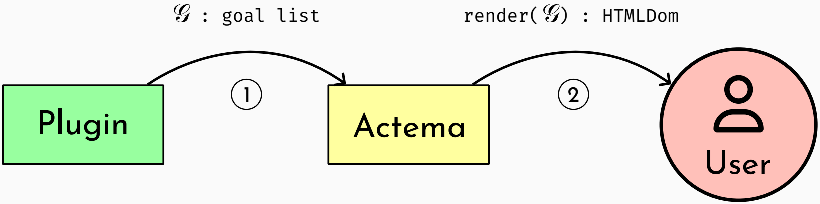
Protocol (interactive)



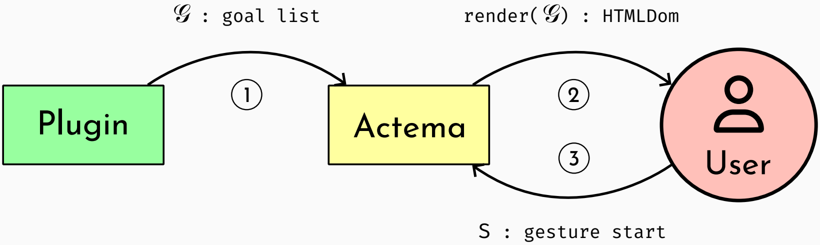
Protocol (interactive)



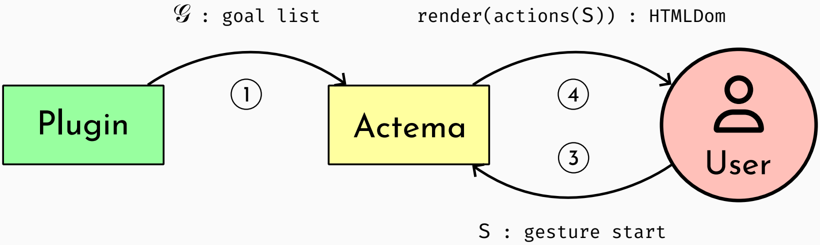
Protocol (interactive)



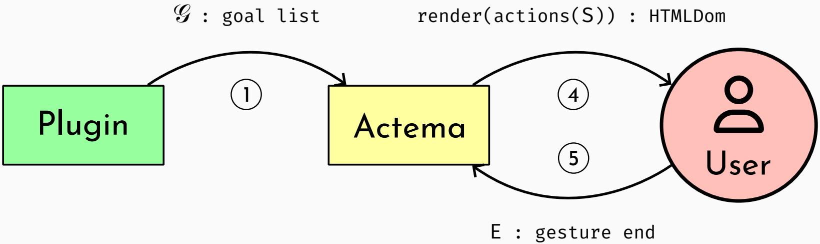
Protocol (interactive)



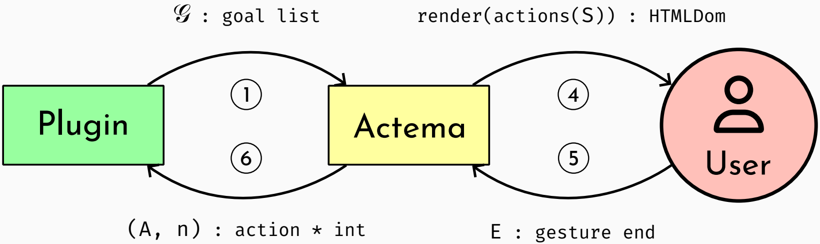
Protocol (interactive)



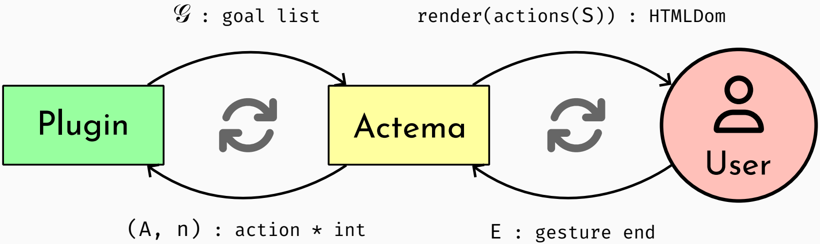
Protocol (interactive)



Protocol (interactive)



Protocol (interactive)



Deep inference semantics

Proof-by-Pointing — Bertot et al. (1994)

- Socrates example:

$$\text{Backward} \iff \texttt{apply H1}$$

$$\text{Forward} \iff \texttt{apply H1 in H2}$$

- $\underline{A} \wedge B \ \vdash \ B \wedge (\underline{A} \vee C) \wedge D$ is trickier...

$$\dfrac{\dfrac{A, B \vdash B \qquad \dfrac{\dfrac{\dfrac{}{A, B \vdash A}}{A, B \vdash A \vee C}\,\vee R_1 \qquad A, B \vdash D}{A, B \vdash (A \vee C) \wedge D}\,\wedge R}{A, B \vdash B \wedge (A \vee C) \wedge D}\,\wedge R}{A \wedge B \vdash B \wedge (A \vee C) \wedge D}\,\wedge L$$

```
destruct H as [HA HB].
split.
* admit.
* split.
  - left. assumption.
  - admit.
```

Idea: instead of *destroying* connectives, switch them

switch:

$$\underline{A} \wedge B \vdash \boxed{B \wedge} (\underline{A} \vee C) \wedge D$$

$$\rhd\ B \wedge (\underline{A} \wedge B \vdash (\underline{A} \vee C) \boxed{\wedge D})$$

$$\rhd\ B \wedge (\underline{A} \wedge B \vdash \underline{A} \boxed{\vee C}) \wedge D$$

$$\rhd\ B \wedge ((\underline{A} \boxed{\wedge B} \vdash \underline{A}) \vee C) \wedge D$$

identity:

$$\rhd\ B \wedge ((B \Rightarrow (\boxed{\underline{A} \vdash \underline{A}})) \vee C) \wedge D$$

unit elimination:

$$\rhd\ B \wedge ((\boxed{B \Rightarrow \top}) \vee C) \wedge D$$

$$\rhd\ B \wedge (\boxed{\top \vee C}) \wedge D$$

$$\rhd\ B \wedge \boxed{\top \wedge} D$$

$$\rhd\ B \wedge D$$

1. Unify linked subformulas
2. **Instantiate** unified variables
3. **Switch** uninstantiated quantifiers

$$\exists y.\ \forall x. R(x, y) \vdash \forall x'. \exists y'. R(x', y')$$

$$\triangleright \quad \forall y. (\forall x. R(x, y) \vdash \forall x'. \exists y'. R(x', y'))$$

$$\triangleright \quad \forall y. \forall x'. (\forall x. R(x, y) \vdash \exists y'. R(x', y'))$$

$$\triangleright \quad \forall y. \forall x'. (\forall x. R(x, y) \vdash R(x', y))$$

$$\triangleright \quad \forall y. \forall x'. (R(x', y) \vdash R(x', y))$$

$$\triangleright \quad \forall y. \forall x'. \top$$

$$\triangleright^* \quad \top$$

$$x \longmapsto x'$$

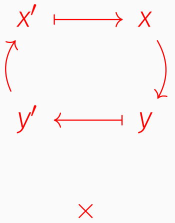
$$y \longleftarrow\!\dashv y'$$

✓

Linking under quantifiers — Donato et al. (2022a)

1. Unify linked subformulas
2. **Instantiate** unified variables
3. **Switch** uninstantiated quantifiers

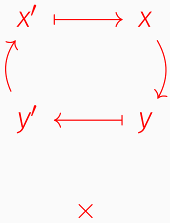
$$\forall x'.\exists y'.\underline{R(x',y')} \vdash \exists y.\forall x.\underline{R(x,y)}$$



Linking under quantifiers — Donato et al. (2022a)

1. Unify linked subformulas
2. **Check** for ∀∃ dependency cycles
3. **Instantiate** unified variables
4. **Switch** uninstantiated quantifiers

$$\forall x'.\exists y'.\underline{R(x', y')} \vdash \exists y.\forall x.\underline{R(x, y)}$$



Add 4 rules $\Longrightarrow$ `rewrite` for free!

$$\underline{t} = u \vdash \underline{A} \quad \rhd \quad A\{t := u\} \qquad\qquad t = \underline{u} \vdash \underline{A} \quad \rhd \quad A\{u := t\}$$

$$\underline{t} = u \ast \underline{A} \quad \rhd \quad A\{t := u\} \qquad\qquad t = \underline{u} \ast \underline{A} \quad \rhd \quad A\{u := t\}$$

Compositional with semantics of **connectives**:

- **Quantifiers:** rewrite modulo *unification*
- **Implication:** *conditional* rewrite
- **Arbitrary** combinations are possible:

$$\forall x. x \neq 0 \Rightarrow \underline{f(x)} = g(x) \vdash \exists y. A(\underline{f(y)}) \vee B(y)$$

$$\rhd^* \quad \exists y. (y \neq 0 \wedge A(g(y))) \vee B(y)$$

Compiling actions (B. Werner, K. Chaudhuri)

- **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. (2022b)]
    - Backward: new conclusion C'
    - Forward: new hypothesis A
  - Final tactic = apply **soundness** theorem
    - Backward: $\Gamma \Rightarrow C' \Rightarrow C$
    - Forward: $\Gamma \Rightarrow A$

Conclusion

FAQ

What are the most useful usecases of Actema?

- Proof **exploration**
- **Educational** setting

FAQ

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!

References

Ayers, E. W., Jamnik, M., and Gowers, W. T. (2021). A Graphical User Interface Framework for Formal Verification. In Cohen, L. and Kaliszyk, C., editors, *12th International Conference on Interactive Theorem Proving (ITP 2021)*, volume 193 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:16, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Bertot, Y., Kahn, G., and Théry, L. (1994). Proof by pointing. In Hagiya, M. and Mitchell, J. C., editors, *Theoretical Aspects of Computer Software*, volume 789, pages 141–160. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

Chaudhuri, K. (2013). Subformula linking as an interaction method. In Blazy, S., Paulin-Mohring, C., and Pichardie, D., editors, *Interactive Theorem Proving*, volume 7998, pages 386–401. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

Chaudhuri, K. (2021). Subformula linking for intuitionistic logic with application to type theory. In Platzer, A. and Sutcliffe, G., editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 200–216. Springer.

Donato, P., Strub, P.-Y., and Werner, B. (2022a). A drag-and-drop proof tactic. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2022, page 197–209, New York, NY, USA. Association for Computing Machinery.

Donato, P., Strub, P.-Y., and Werner, B. (2022b). A drag-and-drop proof tactic. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2022, page 197–209, New York, NY, USA. Association for Computing Machinery.