Actema: Integrating Graphical Proofs into ITPs via Plugin-Driven Gestural Interaction

Pablo Donato
D3S, Charles University
pablo.donato@tuta.io

Benjamin Werner
LIX, INRIA
benjamin.werner@inria.fr

Abstract

We present **Actema**, a graphical user interface for interactive theorem proving that integrates with the Rocq proof assistant through a client-server plugin architecture. Actema enables users to construct formal proofs via direct manipulation actions such as click and drag-and-drop upon logical and mathematical expressions in the current proof state, offering an expressive gestural language that extends the Proof-by-Pointing paradigm introduced by Bertot et al. in the 90s [2]. In particular, the semantics of its drag-and-drop actions exploits recent advances in deep inference proof theory to generalize the behavior of traditional `apply` and `rewrite` tactics [4].

Unlike previous browser-based interfaces such as jsCoq [1], Actema is not a full IDE but an interactive replacement for Rocq's goal view. Figure 1 shows a screenshot of the system in action. Its design emphasizes the separation of interface and kernel responsibilities: a JavaScript frontend for user interaction, an OCaml backend for proof processing, and an HTTP-based communication layer with a custom-made Rocq plugin. This modularity enables Actema to act as an enhanced proof view for any existing or future IDE. While currently focused on Rocq, the design makes it possible to integrate Actema with any other goal-directed ITP, as long as it provides a plugin implementing Actema's interactive proving protocol. Our approach also differs from that of ProofWidgets [3], a framework which (unlike us) supports arbitrary user-programmed GUIs for domain-specific interactive notations in the goal view, but is tightly coupled to the Lean proof assistant.

To bridge the gap between the ITP's internal logic and Actema's HOL-based engine, we introduce a translation layer in the plugin. This layer compiles user gestures performed in Actema's interface into corresponding proof tactics that can be executed to generate certified proof terms, but also inlined to create a static proof script representation of the user's actions. By decoupling the graphical action trace from the tactic script, we stay agnostic to the particular statement and proof languages of the ITP, so that in principle it should be possible to reuse graphical proofs in previously non-interoperable tools. Recorded graphical traces could also be replayed dynamically in Actema's interface in order to review existing proofs, although this feature is not yet implemented.

This architecture supports a hybrid workflow where users can freely switch between textual and graphical proof modes (see Fig. 1), letting them gradually opt into either paradigm depending on their level of expertise. In particular, beginners can start with the intuitive drag-and-drop actions to learn how to build proofs, while advanced users can switch to textual mode for complex domain-specific manipulations and automations.

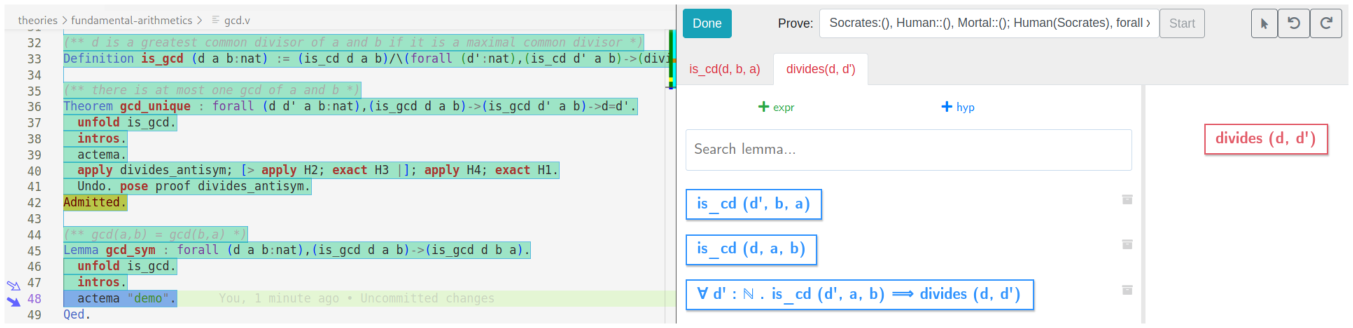
We also reflect on the challenges of statically representing graphical actions in a durable, human-readable way. Time-based proof visualization — where the evolution of a proof is shown dynamically — is especially helpful for understanding the reasoning process on a local scale, but it is not best suited to grasp the higher-level structure of a proof nor to support modification and maintenance of existing proofs, i.e. *proof evolution.* We are currently experimenting with the possibility of exploiting the underlying structure of graphical traces to generate static proof texts in (controlled) natural language, while also researching proof-theoretic frameworks that could support a more uniform representation of proof traces and proof objects.

By integrating Actema with Rocq through a plugin-driven architecture and focusing on direct manipulation, we address key HATRA themes of human-centricity, accessibility, and modularity in proof engineering tools. This work contributes to the HATRA agenda by demonstrating how proof assistants can be extended with human-centric interfaces grounded in a solid theoretical model. We see Actema as a step toward more transparent and embodied forms of formal reasoning.

References

[1] Emilio J. Gallego Arias, Benoît Pin, and Pierre Jouvelot. jsCoq: Towards hybrid theorem proving interfaces. *Electronic Proceedings in Theoretical Computer Science*, 239:15–27, 2017. Available at: `https://doi.org/10.4204/eptcs.239.2`

[2] Yves Bertot, Gilles Kahn, and Laurent Théry. Proof by pointing. In *Theoretical Aspects of Computer Soft-*


theories > fundamental-arithmetics > gcd.v

```
(** d is a greatest common divisor of a and b if it is a maximal common divisor *)
Definition is_gcd (d a b:nat) := (is_cd d a b)/\(forall (d':nat),(is_cd d' a b)->(divi

(** there is at most one gcd of a and b *)
Theorem gcd_unique : forall (d d' a b:nat),(is_gcd d a b)->(is_gcd d' a b)->d=d'.
  unfold is_gcd.
  intros.
  actema.
  apply divides_antisym; [> apply H2; exact H3 |]; apply H4; exact H1.
  Undo. pose proof divides_antisym.
Admitted.

(** gcd(a,b) = gcd(b,a) *)
Lemma gcd_sym : forall (d a b:nat),(is_gcd d a b)->(is_gcd d b a).
  unfold is_gcd.
  intros.
  actema "demo".        You, 1 minute ago • Uncommitted changes
Qed.
```

Figure 1: Developing proofs graphically within a textual setting. On the left, the usual interactive view of the proof script, in the VsCoq IDE. On the right, the graphical proof view of Actema. Blue items correspond to hypotheses, the red item corresponds to the conclusion, and tabs allow to switch focus between subgoals.

ware, pages 141–160. Springer, 1994. Available at: `https://doi.org/10.1007/3-540-57887-0_94`

[3] Edward Ayers, Mateja Jamnik, and W.T. Gowers. A graphical user interface framework for formal verification. In *ITP 2021, LIPIcs*, volume 193, pages 4:1–4:16. Schloss Dagstuhl, 2021. Available at: `https://doi.org/10.4230/LIPIcs.ITP.2021.4`

[4] Pablo Donato, Pierre-Yves Strub, and Benjamin Werner. A drag-and-drop proof tactic. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2022)*, pages 197–209. ACM, 2022. Available at: `https://doi.org/10.1145/3497775.3503692`