Stage de Master Informatique (printemps/été 2026)
Master Internship (spring/summer 2026)
Source-to-source Transformations of Coq/Rocq Proof Scripts: Removing, Inlining, Expanding Tactics

Alexandre Jean (`alexandre.jean@unistra.fr`) and Nicolas Magaud (`magaud@unistra.fr`)

The Coq/Rocq system [1] is a proof assistant dedicated to both mathematics and computer science [1]. It not only allows to formally define mathematical theories, but also to state and prove properties on these theories. It works interactively. The user *interactively* constructs what he believes to be a proof of the theorem, and the system *automatically* checks that the constructed proof does indeed demonstrate the theorem under consideration. We have recently developed some tools to automate transformations of Coq/Rocq proof scripts [5, 4, 3]. The first transformation we implemented is transforming a multi-step proof into an equivalent single-step proof (see Figure 1). We then developped a general framework named `Rocq-ditto` to carry out various source-to-source transformations of Coq/Rocq proof scripts. Such transformations could help making the proof scripts clearer, faster to run, etc. They could also make upgrading formal developments from one version of Coq/Rocq to a more recent one smoother.

```
Lemma foo : forall A B C : Prop, A \/ (B /\ C) -> (A\/B)/\(A\/C).
```

```
Proof.
  intros; destruct H.
  split.
  left; assumption.
  left; assumption.
  destruct H.
  split.
  right; assumption.
  right; assumption.
  Qed.
```

```
Proof.
intros; destruct H;
  [ split;
    [ left; assumption
    | left; assumption ]
  | destruct H ;
    split;
    [ right; assumption
    | right; assumption ] ].
Qed.
```

Figure 1: A regular proof in Coq/Rocq (left) and its compact version (right)

The main objective of this work is to extend this tool to carry out new transformations and to investigate how to combine them. We suggest to start by adding a transformation to remove all occurrences of the tactics `elim` and `case` and replace them by their counterparts `induction` and `destruct`. We can also study how to replace the applications of user-defined tactics (written using Ltac [2] or Ltac2 [6]) by the actual tactics they use. Finally, we should consider whether some proof scripts could be inlined into other ones.

[1] `https://rocq-prover.org/`

References

[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*. Springer, 2004.

[2] David Delahaye. A tactic language for the system coq. In Michel Parigot and Andrei Voronkov, editors, *Logic for Programming and Automated Reasoning, 7th International Conference, LPAR ,2000, Proceedings*, volume 1955 of *LNCS*, pages 85–95. Springer, 2000.

[3] Alexandre Jean and Nicolas Magaud. Transformations automatisées de preuves Coq. In Fabien Dagnat and Olga Kouchnarenko, editors, Approches Formelles pour l'Assistance au Développement de Logiciels 2025 (AFADL 2025) , Pau, France, June 2025, 2025.

[4] Titouan Lozac'h and Nicolas Magaud. Post-processing Coq Proof Scripts to Make Them More Robust. In *2nd Workshop on the development, maintenance, refactoring and search of large libraries of proofs , September 13-14, 2024, Tbilissi, Georgia*, 2024.

[5] Nicolas Magaud. Towards Automatic Transformations of Coq Proof Scripts. In Pedro Quaresma and Zoltán Kovács, editors, Proceedings 14th International Conference on *Automated Deduction in Geometry,* Belgrade, Serbia, 20-22th September 2023, volume 398 of *EPTCS*, pages 4–10. Open Publishing Association, 2024.

[6] Pierre-Marie Pédrot. Ltac2: Tactical Warfare. In Robbert Krebbers and Ilya Sergey, editors, *Proceedings of the CoqPL workshop 2019*, 2019.