A Generalized Search Command for Coq-Actema

Offered by: Benjamin Werner `benjamin.werner@inria.fr`
Location: Inria-X team Partout, LIX, Ecole polytechnique

Context

Coq is a well-known proof system, based on type theory whose development is organized by Inria. Actema is a prototype of a novel user interface for building formal proofs, developed at LIX; it builds on theoretical tools coming from Deep Inference, and is based on the idea that the user can easily point to subterms of either the goal or hypotheses. A description can be found in [1].

A first version of Actema, restricted to first-order logic can be tested online (http://actema.xyz/). A new version, not yet publicly available, acts as a front-end for Coq.

Goal

Standard textual Coq (meaning without Actema) offers a useful Search command. For instance, `Search _ + (S _) = _.` will find all the lemmas ending with the given pattern, like:

```
Nat.add_1_r: forall n : nat, n + 1 = S n
Nat.add_succ_r: forall n m : nat, n + S m = S (n + m)
odd_even_lem: forall p q : nat, 2 * p + 1 <> 2 * q
```

The Actema paradigm, based on interaction between subterms of the goal and/or hypotheses, suggests an extension of this command:

- user selects a subterm of either the goal or an hypothesis,
- the system will then search the base for lemmas able to interact with this subterm (that is either use it or prove it, depending upon whether this subterm is in a positive or negative position).

On the one hand this should make Coq-Actema much more usable for real-life developments. On the other hand it would be a much more powerful version of the Search tactic because, among other:

- It allows Search for positive and negative subterms,
- it builds on the ability of Actema to use subterms easily (for instance Searching for a pattern `P _` will also return lemmas proving `(P x) /\ (Q x)`)

Various generalizations and follow-ups are possible, like including associative-commutative unification.

Assessment

There should be a fair balance between theory and implementation in the work to be done. Knowledge of Coq, logic, and a taste for functional programming are mandatory. There also may be some more algorithmic questions to tackle.

Bibliography

[1] Kaustuv Chaudhuri. Subformula Linking for Intuitionistic Logic with Application to Type Theory *CADE 2021.*

[2] Pablo Donato, Pierre-Yves Strub, Benjamin Werner. A Drag-and-Drop Proof Tactic. CPP 2022. https://hal.science/hal-03823357v2

[3] Pablo Donato, Benjamin Werner, Kaustuv Chaudhuri. Integrating Graphical Proofs in Coq. Talk at CoqPL 2023. https://www.lix.polytechnique.fr/Labo/Pablo.DONATO/abstracts/coqpl23.pdf