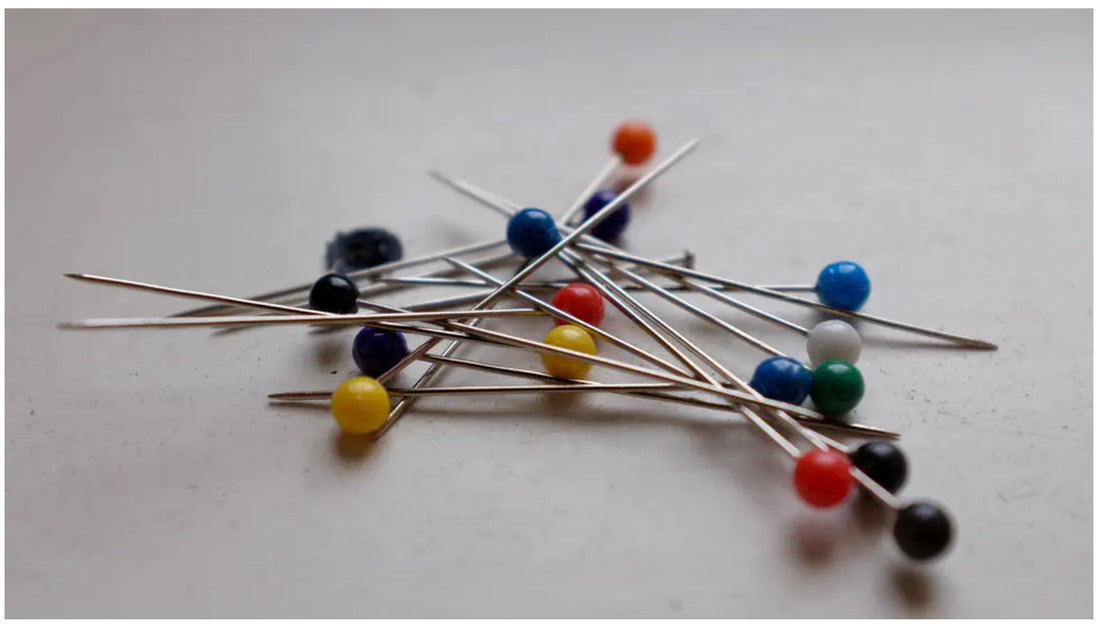
Finding that lemma: Coq search commands

2016-04-19



No hay in this needlestack.

One of the hurdles in using Coq is finding the suitable lemmas from the standard library. There are lots of them and while the naming is consistent, it's hard to remember all of them. Luckily Coq has search commands to help you out.

Note: The following commands work only on modules you have required. If a lemma exists, but you haven't required its module, you're out of luck. Also, before Coq 8.5 `Search` was called `SearchAbout` and `SearchHead` was called `Search`.

The simplest way to search is to search by name. This is one of the things `Search` command does:

```
Coq < Search "len".
length: forall A : Type, list A -> nat
```

You can also search for theorems (or other objects) whose statement contains a given identifier.

```
Coq < Search False.
False_rect: forall P : Type, False -> P
False_ind: forall P : Prop, False -> P
(* ... *)

Coq < Search 0.
nat_rect:
  forall P : nat -> Type,
  P 0 -> (forall n : nat, P n -> P (S n)) -> forall n : nat, P n
nat_ind:
  forall P : nat -> Prop,
  P 0 -> (forall n : nat, P n -> P (S n)) -> forall n : nat, P n
(* ... *)
```

Another thing you can do is to search for patterns with holes _:

```
Coq < Search (S _ <= _).
le_S_n: forall n m : nat, S n <= S m -> n <= m
le_n_S: forall n m : nat, n <= m -> S n <= S m
```

When searching for a pattern, `Search` matches anywhere in the statement. If you only want to search for the pattern in the conclusion, use `SearchPattern`:

```
Coq < SearchPattern (S _ <= _).
le_n_S: forall n m : nat, n <= m -> S n <= S m
```

If you're looking for a rewrite, there's `SearchRewrite`. It finds conclusions of type `_ = _` where one of the sides matches the given pattern.

```
Coq < SearchRewrite (_ + 0).
plus_n_O: forall n : nat, n = n + 0
```

As always, see the manual for details. Coq's manual looks intimidating, but it does contain a lot of good information.

Comments or questions? Send me an e-mail.

- Previous post: How to divide by zero?
- Next post: We're in early days of software engineering
- Filed under: Coq