The proof system Coq

Guillaume Claret

April 21st, 2015
The proof system Coq

A language to:
- state theorems
- write proofs verified by computer
- write algorithms
Fixpoint fact (n:nat) : nat :=
  match n with
  | O => 1
  | S n => S n * fact n
  end.

Lemma lt_0_fact n : 0 < fact n.
Proof.
  induction n; simpl; auto with arith.
Qed.
Why is it important to check proofs?

- mathematical proofs can hundreds of pages long with human mistakes
- software without proofs almost always contains bugs
The proof system Coq

An environment to:

- reason interactively
- organize and distribute proofs
IDE

The proof system Coq

Guillaume Claret

The proof system Coq
Install a package:

```
opam install -j4 package
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coq:color</td>
<td>1.1.0</td>
<td>A library on rewriting theory and termination.</td>
</tr>
<tr>
<td>coq:compcert</td>
<td>2.4.0</td>
<td>The CompCert C compiler.</td>
</tr>
<tr>
<td>coq:concurrency:pluto</td>
<td>1.0.0</td>
<td>A web server written in Coq.</td>
</tr>
<tr>
<td>coq:concurrency:proxy</td>
<td>1.0.0</td>
<td>A proxy to interface concurrent Coq programs with the operating system.</td>
</tr>
<tr>
<td>coq:concurrency:system</td>
<td>1.0.0</td>
<td>Experimental library to write concurrent applications in Coq.</td>
</tr>
<tr>
<td>coq:constructors</td>
<td>1.0.0</td>
<td>An example Coq plugin, defining a tactic to get the constructors of an inductive type in a list.</td>
</tr>
<tr>
<td>coq:coqeq:refinements</td>
<td>0.9.1</td>
<td>A refinement framework (for algebra).</td>
</tr>
<tr>
<td>coq:coqeq:theory</td>
<td>0.9.1</td>
<td>The theory needed by the refinement framework library.</td>
</tr>
<tr>
<td>coq:coquelicot</td>
<td>2.0.1</td>
<td>A Coq formalization of real analysis compatible with the standard library.</td>
</tr>
<tr>
<td>coq:corn</td>
<td>1.0.0</td>
<td>The CoRN library.</td>
</tr>
</tbody>
</table>
Usage

Maths:
- four colors theorem (Gontier 04)
- Feit – Thompson theorem (odd order theorem) (Gontier & all, 12)

Software written and proven in Coq:
- certified C compiler CompCert (Xavier Leroy & all)
- Bedrock library for low-level programs (Adam Chlipala & all)
Many conference papers with Coq proofs in annex.
Based on type theory instead of set theory:

<table>
<thead>
<tr>
<th>set theory</th>
<th>type theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>{n \in \mathbb{N}</td>
<td>\phi(n) \in \mathbb{P}}</td>
</tr>
<tr>
<td>3 + 4 : \text{nat}</td>
<td></td>
</tr>
</tbody>
</table>

- we can mix proofs and programs
- reasoning on programming languages made simpler
quick demo to give you a taste of Coq
we hope it can be useful
Questions