Intro to your 2\textsuperscript{nd} TA.
Using Coq for CS6110 assignments

Abhishek Anand (your 1\textsuperscript{st} TA)

February 12, 2015
Why use Proof Assistants (PA)?

Doing Math (including PL Theory) requires

- Creativity
- Extreme Carefulness
- Mechanical Work
- Good Memory
Why use Proof Assistants (PA)?

With Proof Assistants, doing Math (including PL Theory) mostly requires

- Creativity
- Extreme Carefulness
- Mechanical Work
- Good Memory
Why use Proof Assistants (PA)?

With Proof Assistants, doing Math (including PL Theory) mostly requires:

- Creativity
- Extreme Carefulness
- Mechanical Work
- Good Memory

As a 2nd TA, a PA give immediate feedback often forces you to have a deeper understanding of your proofs.
PAs are already sufficiently mature

- 2 PL-oriented books on Coq, written in Coq: SF, CPDT
- Already captured a vast amount of human knowledge \(^1\) C compiler, variable bindings, real analysis, abstract algebra ...
- Vibrant mailing lists (coq-club, agda); Your question might get answered by a field medalist!

\(^1\)For a more comprehensive list, visit [http://www.lix.polytechnique.fr/coq/pylons/coq/pylons/contribs/bycat/v8.4?cat1=None&cat2=None](http://www.lix.polytechnique.fr/coq/pylons/coq/pylons/contribs/bycat/v8.4?cat1=None&cat2=None)
How Proof Assistants work

Most proofs are composed of a few primitive axioms (e.g. Peano Arith).

- 0 is a number
- \( \forall \text{ number } n, (S \ n) \text{ is a number.} \)
- \( S \) is injective
- \( \forall \text{ number } n, n = n. \) Also, = is symmetric and transitive
- \( (0 + m) = m \)
- \( ((S \ n) + m) = S \ (n + m) \)
- \ldots
- Natural Induction
How Proof Assistants work

Most proofs are composed of a few primitive axioms (e.g. Peano Arith).

- 0 is a number
- ∀ number \( n \), \((S \ n)\) is a number.
- \( S \) is injective
- ∀ number \( n \), \( n = n \). Also, \( = \) is symmetric and transitive
- \((0 + m) = m\)
- \(((S \ n) + m) = S (n + m)\)
- ...
- Natural Induction

\[
\text{Fixpoint } \text{plus} (n \ m : \text{nat}) : \text{nat} := \\
\text{match } n \text{ with } \\
| O \Rightarrow m \\
| S n' \Rightarrow S (\text{plus } n' \ m) \\
\text{end.}
\]
How Proof Assistants work

Most proofs are composed of a few primitive axioms (e.g. Peano Arith).

- 0 is a number
- \( \forall \) number \( n \), \((S \ n)\) is a number.
- \( S \) is injective
- \( \forall \) number \( n \), \( n = n \). Also, = is symmetric and transitive
- \((0 + m) = m\)
- \(((S \ n) + m) = S (n + m)\)
- ... 
- Natural Induction

\[\text{Inductive} \ \text{nat} : \ Type := \]
\[\mid 0 \]
\[\mid S \ (n : \ nat).\]

\[\text{Fixpoint} \ plus \ (n \ m : \ nat) : \ nat := \]
\[\text{match} \ n \ \text{with} \]
\[\mid 0 \Rightarrow \ m \]
\[\mid S \ n' \Rightarrow S \ (\text{plus} \ n' \ m) \]
\[\text{end}.\]

Coq definitions are often quite close to ordinary mathematics
How Proof Assistants work

Most proofs are composed of a few primitive axioms (e.g. Peano Arith).

- 0 is a number
- $\forall$ number $n$, $(S n)$ is a number.
- $S$ is injective
- $\forall$ number $n$, $n = n$. Also, $=$ is symmetric and transitive
- $(0 + m) = m$
- $((S n) + m) = S (n + m)$
- ...
- Natural Induction

\[
\text{Inductive nat : Type :=}
\begin{cases}
  O \\
  S (n : nat)
\end{cases}
\]

\[
\text{Fixpoint plus (n m : nat) : nat :=}
\begin{cases}
  \text{match n with} \\
  O \Rightarrow m \\
  S n' \Rightarrow S (\text{plus } n' \text{ m})
\end{cases}
\]

Coq definitions are often quite close to ordinary mathematics
How Proof Assistants work

Most proofs are composed of a few primitive axioms (e.g. Peano Arith).

- 0 is a number
- ∀ number $n$, $(S \ n)$ is a number.
- $S$ is injective
- ∀ number $n$, $n = n$. Also, $=$ is symmetric and transitive
- $(0 + m) = m$
- $((S \ n) + m) = S \ (n + m)$
- ...
- Natural Induction

\[
\text{Inductive} \ \text{nat} : \ \text{Type} := \\
| \ \text{O} \\
| \ S \ (n : \ \text{nat}).
\]

\[
\text{Fixpoint} \ \text{plus} \ (n \ m : \ \text{nat}) : \ \text{nat} := \\
\text{match} \ n \ \text{with} \\
| \ \text{O} \ \Rightarrow \ m \\
| \ S \ n' \ \Rightarrow \ S \ (\text{plus} \ n' \ m) \\
\text{end}.
\]

Coq definitions are often quite close to ordinary mathematics

Abhishek Anand | Intro to your 2\textsuperscript{nd} TA.
How Proof Assistants work

Most proofs are composed of a few primitive axioms (e.g. Peano Arith).

- 0 is a number
- $\forall$ number $n$, $(S\ n)$ is a number.
- $S$ is injective
- $\forall$ number $n$, $n = n$. Also, $=$ is symmetric and transitive
- $(0 + m) = m$
- $((S\ n) + m) = S\ (n + m)$
- ...
- Natural Induction

**Inductive** nat : Type :=

| O
| $S\ (n : \text{nat})$.

**Fixpoint** plus ($n\ m : \text{nat}$) : \text{nat} :=

match $n$ with

| O ⇒ $m$
| $S\ n' ⇒ S\ (\text{plus}\ n'\ m)$

end.

Coq definitions are often quite close to ordinary mathematics

Abhishek Anand

Intro to your 2$^{nd}$ TA.
$$\textbf{Inductive} \ \text{nat} : \ Type ::=$$
$$\ | \ O$$
$$\ | \ S \ (n : \ nat).$$

$$\textbf{Fixpoint} \ \text{plus} \ (n \ m : \ nat) : \ nat ::=$$
$$\text{match} \ n \ \text{with}$$
$$\ | \ O \ \Rightarrow \ m$$
$$\ | \ S \ n' \ \Rightarrow \ S \ (\text{plus} \ n' \ m)$$
$$\text{end.}$$

Which of these is not a number

- O
- S (S O)
- O (S O)
Inductive nat : Type :=
| O
| S (n : nat).

Fixpoint plus (n m : nat) : nat :=
match n with
| O => m
| S n' => S (plus n' m)
end.

Which of these is not a number
- O
- S (S O)
- O (S O)

Which of these is NOT in a normal form
- O
- plus (S O) (S (S O))
- S (S O)
Inductive nat : Type :=
| O
| S (n : nat).

Fixpoint plus (n m : nat) : nat :=
match n with
| O ⇒ m
| S n' ⇒ S (plus n' m)
end.

Which of these is not a number
- O
- S (S O)
- O (S O)

Which of these is NOT in a normal form
- O
- plus (S O) (S (S O))
- S (S O)
www.cs.cornell.edu/~aa755/CS6110/CoqLecDemo.v

Recommended Tutorials:

PAs can often make proofs easier

- omega, lia, lra, nia...
  \( \forall (n \, m \, k : \text{nat}), n + m \leq m + k + n. \)

- congruence
  \( \forall (n \, m \, k : \text{nat}), n = m \Rightarrow m = k \Rightarrow (n \times m) = (m \times k) \)

- Proofs by computation
  - \( \Omega \) reduces to \( \Omega \)
  - \( \ldots \, [\ldots/\times] \) is equal to \( \ldots \)
  - \( \sqrt{(\cos\frac{1}{2})} < \exp(\cos(sin(arctan(\Pi)))) \)

Abhishek Anand

Intro to your 2\textsuperscript{nd} TA.
PAs can often make proofs easier

- \( \omega, \text{lia}, \text{ira}, \text{nia} \ldots \)
  \[ \forall (n \ m \ k : \text{nat}), \ n + m \leq m + k + n. \]
- congruence
  \[ \forall (n \ m \ k : \text{nat}), \ n = m \Rightarrow m = k \Rightarrow (n \ast m) = (m \ast k) \]
- Proofs by computation
  - \( \Omega \) reduces to \( \Omega \)
  - \( \ldots \ [\ldots/x] \) is equal to \( \ldots \)
  - \( \sqrt{(\cos \frac{1}{2})} < \exp(\cos(\sin(\arctan(\Pi)))) \)
- \( \text{tauto}, \text{ring}, \text{field} \ldots \)
- Custom Hint databases
- Custom Proof Search Algorithms

If you get stuck while doing CS6110 related work in Coq, feel free to ask on Piazza