Intro to your 2^{nd} TA.

Using Coq for CS6110 assignments

Abhishek Anand (your 1^{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~~

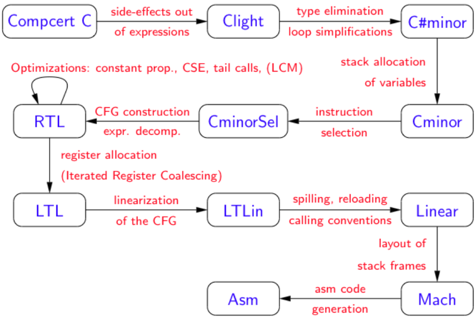
Why use Proof Assistants (PA)?

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

- Creativity
- ~~Extreme Carefulness~~
- ~~Mechanical Work~~
- ~~Good Memory~~

As a 2^{nd} 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

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
- $(0 + m) = m$
- $((S\ n) + m) = S\ (n + m)$
- $\ldots$
- Natural Induction

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 : nat).
```

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

```
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.
```

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

```
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.
```

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

```
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.
```

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

```
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.
```

Coq definitions are often quite close to ordinary mathematics

Quiz

```
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)

Quiz

```
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)

Quiz

```
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)

Demo

www.cs.cornell.edu/~aa755/CS6110/CoqLecDemo.v

Reccommended Tutorials:

- http://www.cis.upenn.edu/~bcpierce/sf/current/Basics.html
- http://www.cis.upenn.edu/~bcpierce/sf/current/Induction.html

PAs can often makes proofs easier

- omega, lia, lra, nia ...
  $\forall (n\ m\ k : nat), n + m \leq m + k + n.$
- congruence
  $\forall (n\ m\ k : nat), n = m \Rightarrow m = k \Rightarrow (n * m) = (m * k)$
- Proofs by computation
  - Ω reduces to Ω
  - ... [.../x] is equal to ...
  - $\sqrt{(cos \frac{1}{2})} < exp(cos(sin(arctan(\Pi))))$

PAs can often makes proofs easier

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

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