SOFTWARE VERIFICATION AND COMPUTER PROOF (lesson 2)

Enrico Tassi – Inria Sophia-Antipolis

Lessons

1. Writing programs in Coq
2. Writing proofs in Coq
3. Proving your recursive programs correct
4. Proofs in arithmetic and more
5. More data types

Today: writing proofs in Coq

- Morning: proofs with simple connectives
- Afternoon: equality and quantifiers

Propositions

- Propositions do have a type too: Prop
- For this morning we assume P Q R are predicates

  Variables P Q R : Prop.
- We focus on the logical connectives

Connectives

- Conjunction of P and Q is written P /\ Q
- Disjunction of P and Q is written P \/ Q
- Negation of P is written ~ P
- P implies Q is written P → Q

Connectives and related commands

connective	use	prove
P /\ Q	destruct	split
P \/ Q	destruct	left, right
P → Q	apply	intros
~ P	case	

Proof commands

1. I'll present them with a demo
2. You will have a short reference for the exercises
3. You will find more examples on the course notes
4. The full documentation can be found in the user manual of Coq https://coq.inria.fr/distrib/current/refman/

Demo

Equality

- The most useful predicate is equality, E.g. x = 3
- Equality means:

  "the two expressions *compute* the same value".

- E.g.

  3 + 7 = 10   (* nothing to prove *)

  x=y → y=z → x=z   (* something to prove *)

Commands for equality

predicate	use	prove
a = b	rewrite → … rewrite ← …	reflexivity

Quantifiers

- Universal quantification is written forall (n : T), ...
- Existential quantification is written exists (n : T), ...
- E.g. forall (n : nat), exists (m : nat), leb n (n + m) = true.

More on quantifiers

- Quantifiers can talk about almost everything, not just numbers, lists, etc. E.g.

  forall (P : Prop) (Q : Prop), ((P → Q) /\ P) → Q

- Moreover we can use some syntactic sugar. E.g.

  forall (P Q : Prop), (P → Q) /\ (P → Q)

  A <-> B instead of (A → B) /\ (B → A)

Commands for quantifiers

quantifier	use	prove
forall n, P	apply apply … with (n := v)	intros
exists n, P	destruct	exists

Searching

- To complete the exercises *you must search* the library for some theorems about arithmetic
- They are too "hard" to be proved by you (for today)
- The command SearchAbout pattern finds all theorems whose statement matches pattern. E.g.
  SearchAbout (_ * _).
  SearchAbout (_ + _ = _).
  SearchAbout (_ + _ * _).

Demo

That is all for today!