

What is Coq's logical framework called?

The Calculus of Inductive Constructions.

What is Coq's specification language called?

Gallina.

Formal development in Gallina consists of
what?

A sequence of *declarations* and *definitions*.

What is a command?

An information request or service routine invocation. It is not part of formal development.

Which command terminates the current session?

Quit.

What does a declaration do?

Associates a *name* with a *specification*.

What is a name?

Roughly speaking, an identifier in a programming language. A string of letters, digits, underscores, primes, starting with a letter.

What is a specification?

A formal expression which classifies the notion which is being declared.

What kinds of specifications are there?

- Logical propositions (`Prop`)
- Mathematical collections (`Set`)
- Abstract types (`Type`)

Prop, Set, **and** Type are what?

Atomic abstract types.

Every expression e in Gallina is associated
with what?

A specification, itself a valid expression, called its *type*, $t(E)$.

How can you get the type of an expression?

Using the `Check` **command:**

```
Coq < Check 0.
```

```
0
```

```
      : nat
```


What is the role of sections?

- Modularize by limiting scope of parameters, hypotheses, and definitions.
- Give a convenient way to reset part of the development environment.

How do you enter a section?

```
Coq < Section Declaration.
```

Translate *let n be a natural number* into Coq.

```
Coq < Variable n : nat.  
n is assumed
```

Translate *let n be a positive natural number*
into Coq.

Coq < Hypothesis Pos_n : (gt n 0) .

Pos_n is assumed

What is the type of `gt`?

`nat -> nat -> Prop`

`nat -> nat -> Prop` is an abbreviation for
what?

`nat -> (nat -> Prop)`

``(gt n 0)` is an abbreviation for what?

((gt n) 0)

Unlike in Scala, types in Coq are ...

... values

```
Coq < Check (nat -> Prop) .
```

```
nat -> Prop
```

```
      : Type
```


What are the types of `nat`, `O`, `S`, and `plus`?

Set, nat, nat \rightarrow nat, nat \rightarrow nat \rightarrow nat

Create several variables in one declaration.

Coq < Variables A B C : Prop.

A is assumed

B is assumed

C is assumed

What are the meanings of \rightarrow ?

- Type constructor, e.g. `nat -> nat`.
- The propositional connective *implication*.

How do you enter the proof engine?

With the command `Goal` followed by the conjecture you want to verify.