# The Calculus of Inductive Constructions.

Gallina.

A sequence of *declarations* and *definitions*.

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

Quit.

Associates a *name* with a *specification*.

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

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

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

# Atomic abstract types.

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

Using the `Check` command:

```
Coq < Check O.
O
     : nat
```

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

```
Coq < Section Declaration.
```

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

```
Coq < Hypothesis Pos_n : (gt n 0).
Pos_n is assumed
```

```
nat -> nat -> Prop
```

```
nat -> (nat -> Prop)
```

```
((gt n) 0)
```

... values

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

```
Set, nat, nat->nat, nat -> nat -> nat
```

```
Coq < Variables A B C : Prop.
A is assumed
B is assumed
C is assumed
```

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

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