# Reflections on Prismatic Constructions

## Marc Coiffier

## Contents

The Calculus of Prismatic Constructions, upon which this platform is based, is an extension of the standard CoC with a mechanism for discriminating inductive constructors.

## Inductive Types

Inductive types can be described as enumerations of constructors. In Coq (and similarly in other proof assistants), an inductive type must be declared along with its constructors, using a syntax like :

```
Inductive T : forall A..., Type :=
| t0 : forall x0..., T (f0... x0...)
...
| tn : forall xn..., T (fn... xn...)
.
```

Here, we declare the inductive type $T : \forall A..., Type$, and its constructors called $t_i$ ($i \in \{0..n\}$).

As a more concrete example, here is how the type of Booleans can be defined inductively :

```
Inductive Boolean : Type := true : Boolean | false : Boolean.
```

The above definition is essentially a formal statement of the following description of Booleans : a Boolean can have one of two shapes, $true$ or $false$, and cannot be any other thing.

This means that, if we want to prove a property $P\,x$ for some unknown Boolean $x$, all we need is to prove $P\,true$ and $P\,false$.

This information is summed up in what we call the *induction principle* for Booleans. In Coq, it will be given the name `Boolean_rect`, and have the type $\forall(P : Boolean \rightarrow Type),\ P\,true\ \rightarrow\ P\,false\ \rightarrow\ \forall(b : Boolean),\ P\,b$.