



5th Asian-Pacific Summer School on Formal Methods
August 5-10, 2013, Tsinghua University, Beijing, China

Inductive data types (I)

jean-jacques.levy@inria.fr

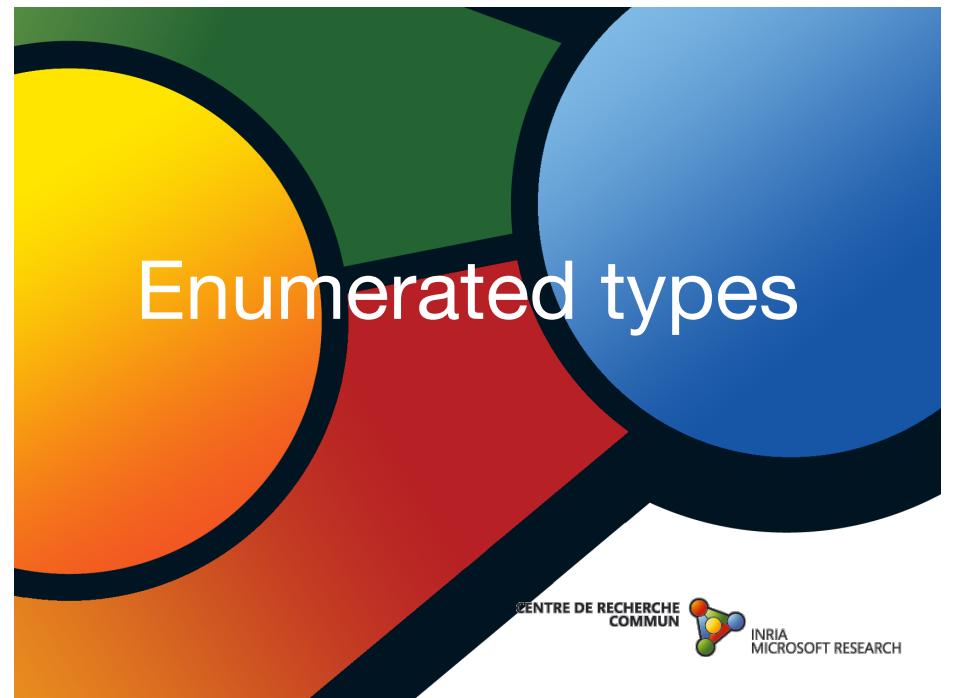
August 7, 2013



<http://sts.thss.tsinghua.edu.cn/Coqschool2013>



Notes adapted from
Assia Mahboubi
(coq school 2010, Paris) and
Benjamin Pierce (software
foundations course, UPenn)



Plan

- easy proofs by simplification and reflexivity
- recursive types
- recursive definitions
- structural induction
- example1: lists
- example2: trees

Recap

- lambda notation for definition of functions
- Coq only allows typed lambda terms



Inductive declarations



An arbitrary type as assumed by:

Variable `T : Type.`

gives no a priori information on the nature, the number, or the properties of its inhabitants.

Inductive declarations

Inductive types in *Coq* can be seen as the generalization of similar type constructions in more common programming languages.

They are in fact an extremely rich way of defining data-types, operators, connectives, specifications,...

They are at the core of powerful programming and reasoning techniques.

Inductive declarations

An **inductive** type declaration explains how the inhabitants of the type are built, by giving **names** to each construction rule:

Enumerative types (1/5)

Enumerated types are types which list and name exhaustively their inhabitants.

`Inductive bool : Set := true : bool | false : bool.`

Set is deprecated. Now use Type.

`Inductive color : Type := black : color | white : color.`



Enumeratives types (2/5)

Enumerated types are types which list and name exhaustively their inhabitants.

A new enumerated type:

```
Inductive day : Type :=
| monday | tuesday | wednesday |
| thursday | friday | saturday | sunday : day.
```



Enumeratives types (4/5)

```
Definition andb (b1:bool) (b2:bool) : bool :=
  match b1 with true => b2 | false => false end.
```

```
Definition orb (b1:bool) (b2:bool) : bool :=
  match b1 with true => true | false => b2 end.
```



Enumeratives types (3/5)

Inspect the enumerated type inhabitants and assign values:

```
Definition negb (b : bool) :=
  match b with true => false | false => true end.
```



Enumeratives types (5/5)

Exercice Give definitions of predicates `work_day` and `weekend_day`.

Exercice Give definitions of predicates `black_if_workday` and `white` for weekends.

