The Coq proof assistant : principles and practice

J.-F. Monin

Université Grenoble Alpes

2016

Lecture 7

Outline

Polymorphism

Outline

Outline

Polymorphism

Lists

Polymorphism

A type can be a parameter of a function

Example: the identity function

Definition ide := fun (X: Type) => fun (x: X) => x.

Definition ide (X: Type) (x: X) := x.

Implicit arguments

When using the identity function, the first argument can be automatically inferred from the second

Example

id nat 3

id _ 3

Local declaration

Definition id {X: Type} (x: X) := x.

Simplified application

id 3

Implicit arguments

When using the identity function, the first argument can be automatically inferred from the second

Example

id nat 3

id _ 3

Local declaration

Definition id {X: Type} (x: X) := x.

Simplified application

id 3

Recovering explicit application

@id nat

id (X:=nat)

Implicit arguments

When using the identity function, the first argument can be automatically inferred from the second

Example

id nat 3
id _ 3

Local declaration

Definition id {X: Type} (x: X) := x.

Simplified application

id 3

Recovering explicit application

@id nat
id (X:=nat)

Global declaration

Set Implicit Arguments.

Outline

Lists

Polymorphic inductive definition

```
Inductive list (X: Set) : Set :=
  | nil : list X
  | cons : X -> list X -> list X.
```

On Type

Can be used in more situations (e.g., lists of predicates)

```
Inductive list (X: Type) : Type :=
  | nil : list X
  | cons : X -> list X -> list X.
```

Basic important properties

app : for appending two lists

Basic important properties

app : for appending two lists

nil is neutral on the left and on the right for app

- left : by reflexivity
- right : by induction

Basic important properties

app : for appending two lists

nil is neutral on the left and on the right for app

- left : by reflexivity
- right : by induction

app is associative

- app (app u v) w = app u (app v w)
  just by induction on u

Additional material

See coq files Lecture07_lists