Polymorphic types

jean-jacques.levy@inria.fr
2013-8-8

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

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

Plan

• polymorphic lists
• polymorphic functions
• implicit arguments
• induction on polymorphic lists
• polymorphic trees, products, options
• higher-order functions

一日为师，终生为父

Yi ri wei shi, zhirusheng wei fu
- Testing
- Static analysis
- Formal methods

Polymorphic lists (1/5)

Lists of any type $X$.

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

Exercice 14 Check list, nil, cons.
Exercice 15 Check cons nat 1 (cons nat 2 (nil nat)).

Definition daylist := list (day).
Definition natlist := list (nat).

Check (cons day monday (cons day tuesday (nil day))).
Check (cons nat 2 (cons nat 3 (nil nat))).

Check (cons _ monday (cons _ tuesday (nil _)�)

Polymorphic lists (2/5)

Fixpoint app (X:Type) (l1 l2 : list X) {struct l1} :=
| list X :=
match l1 with
| nil => l2
| cons h t => cons X h (app X t l2)
end.

Exercice 16 Associativity of append. Etc..

Fixpoint rev (X:Type) (l:list X) {struct l} : list X :=
| match l with
| nil => nil X
| cons h t => app X (rev X t) (cons X h (nil X))
end.
Synthetizing arguments (1/4)

Fixpoint length (X:Type) (l:list X) {struct l} : nat := 
match l with 
| nil => 0 
| cons h t => S (length _ t) 
end.

Example test_length2 :
  length _ (cons _ 1 (cons _ 2 (nil _))) = 2.
  Proof. reflexivity. Qed.

Synthetizing arguments (2/4)

Arguments nil [X].
Arguments cons [X] _ _.

Check cons 2 nil.
Check cons monday nil.

or simply with argument in braces at function definition.

Fixpoint length (X:Type) (l:list X) {struct l} : nat := 
match l with 
| nil => 0 
| cons h t => S (length t) 
end.

Example test_length3 :
  length _ (cons 1 (cons 2 (nil ))) = 2.
  Proof. reflexivity. Qed.

@length is notation for function with all arguments.

Synthetizing arguments (3/4)

Notation "x :: l" := (cons x l) (at level 60, right associativity).

Notation "[ ]" := nil.
Notation "[ x , .. , y ]" := (cons x .. (cons y nil) ..).

Check 3 :: 4 :: nil.
Check monday :: tuesday :: nil.
Check {3, 4, 5}.

Synthetizing arguments (4/4)

Also decreasing argument is implicit when clear from definition.

Fixpoint length {X:Type} (l:list X) : nat := 
match l with 
| nil => 0 
| cons h t => S (length t) 
end.

Fixpoint app {X : Type} (l1 l2 : list X) : (list X) := 
match l1 with 
| nil => l2 
| cons h t => cons h (app l2 t) 
end.

Exercice 17 Write definition of rev with implicit arguments.
Polymorphic lists (4/5)

Let iterative reverse be:

```coq
Fixpoint irev {X: Type} (l1 l2: list X) : list X :=
match l1 with
| [] => l2
| v1 :: l1' => irev l1' (v1 :: l2)
end.
```

**Exercice 18** Show for any lists \( \ell_1, \ell_2, \ell_3 \):

\[
\ell_1 ++ (\ell_2 ++ \ell_3) = (\ell_1 ++ \ell_2) ++ \ell_3
\]

\[
\text{length}(\ell_1 ++ \ell_2) = (\text{length}\ \ell_1) + (\text{length}\ \ell_2)
\]

\[
\text{rev}\ \ell_1 = \text{irev}\ \ell_1 []
\]

\[
\ell ++ [] = \ell
\]

\[
\text{rev}(\ell_1 ++ \ell_2) = (\text{rev}\ \ell_2) ++ (\text{rev}\ \ell_1)
\]

\[
\text{rev}(\text{rev}\ \ell) = \ell
\]

\[
\ell = \text{rev}\ \ell' \Rightarrow \ell' = \text{rev}\ \ell
\]

Polymorphic binary trees (1/2)

Inductive binTree (X : Type) :=
| leaf : X -> binTree X
| node : X -> binTree X -> binTree X -> binTree X.

Fixpoint count_leaves (X: Type) (t : binTree X) :=
match t with
| leaf _ => 1
| node _ t1 t2 => (count_leaves t1) + (count_leaves t2)
end.

Polymorphic Option and Product

A polymorphic non recursive option type:

Inductive option (X : Type) : Type :=
| Some : X -> option X | None : option X.

Use it for default value:

Fixpoint last (X : Type) (l : list X) : option X :=
match l with
| [] => None
| v :: nil => Some v
| _ :: l' => last l'
end.

We also define polymorphic product.

Inductive prod (X Y : Type) : Type :=
| pair : X -> Y -> prod X Y.

The notation \( X \times Y \) denotes \( \text{prod}\ X Y \).
The notation \( (x, y) \) denotes \( \text{pair}\ x\ y \) (implicit argument).

Polymorphic binary trees (2/2)

Lemma height_le_size : forall (X: Type) (t : binTree X),

\[ \text{height}\ t \leq \text{size}\ t. \]

Proof.

intros X t. induction t as [l x t1 IHt1 t2 IHt2].

- reflexivity.
- simpl. apply Le.le_n_S.
  - apply Max.max_case.
    + apply (Le.le_trans _ (size t1) _).
      * apply IHt1. apply Plus.le_plus_l.
    + apply IHt2. apply Plus.le_plus_r.

Qed.
Higher order functions

```
Fixpoint map {X : Type} (f : X -> Y) (l : list X) : list Y :=
match l with
| [] => []
| x :: l' => (f x) :: map f l'
end.

Example map_negb : map negb [true, false] = [false, true].
Example map_next_weekday :
  map next_weekday [monday, friday] = [tuesday, monday].

Exercice 19 Show
  map f (rev `) = rev (map f `)
  map f (l1 ++ l2) = (map f l1) ++ (map f l2)```