Aula 7 - Listas de Naturais

Rafael Castro - rafaelcgs10.github.io/coq

16/05/2018

Listas

- Listas, diferente de duplas, são uma sequência (neste caso finita) de elementos de um mesmo tipo.
- Ex: *[1, 2, 3, 4]*.

Lista de Naturais em Coq

- Uma lista de naturais ou é vazia...
- Ou é número natural inserido no início de outra lista.

```
Inductive natlist : Type :=
  | nil  : natlist
  | cons : nat -> natlist -> natlist.

(** For example, here is a three-element list: *)

Definition mylist := cons 1 (cons 2 (cons 3 nil)).
```

Algumas notações para listas

O operador é associativo para a direita.

Todos os exemplos de listas abaixo significam a mesma coisa.

```
Notation "x :: l" := (cons x l)
     (at level 60, right associativity).
Notation "[ ]" := nil.
Notation "[ x ; .. ; y ]" := (cons x .. (cons y nil) ..).

Definition mylist1 := 1 :: (2 :: (3 :: nil)).
Definition mylist2 := 1 :: 2 :: 3 :: nil.
Definition mylist3 := [1;2;3].
```

Função *repeat*

- A função *repeat* retorna uma lista de tamanho *count* onde cada elemento é *n*.

```
Fixpoint repeat (n count : nat) : natlist :=
  match count with
  | O => nil
  | S count' => n :: (repeat n count')
  end.
```

Função *length*

- Uma lista vazia em tamanho *0*.
- A função *length* aninha um sucessor para cada elemento na lista.

```
Fixpoint length (l:natlist) : nat :=
  match l with
  | nil => O
  | h :: t => S (length t)
  end.
```

Função *append*

- A função *app* concatena duas listas

```
Fixpoint app (l1 l2 : natlist) : natlist :=
  match l1 with
  | nil    => l2
  | h :: t => h :: (app t l2)
  end.

(** Actually, [app] will be used a lot in some parts of what
    follows, so it is convenient to have an infix operator f

Notation "x ++ y" := (app x y)
     (right associativity, at level 60).

Example test_app1:              [1;2;3] ++ [4;5] = [1;2;3;4;5
Proof. reflexivity.  Qed.
Example test_app2:                   nil ++ [4;5] = [4;5].
```

Função *head* e *tail*

- A função head retorna o primeiro elemento de uma lista
- A função tail retorna a mesma lista sem o primeiro elemento.

```
Definition hd (default:nat) (l:natlist) : nat :=
  match l with
  | nil => default
  | h :: t => h
  end.

Definition tl (l:natlist) : natlist :=
  match l with
  | nil => nil
  | h :: t => t
  end.

Example test_hd1:                hd 0 [1;2;3] = 1.
Proof. reflexivity.  Qed.
```