Chapter 1

Library c01_basics

Capítulo 1 - Functional Programming in Coq (Basics)

Booleano

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

Negação

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

```
Notation "~ x" := (negb x).
```

Tabela Verdade - Negação

```
Example test_negb1 : (negb true) = false.
```

```
Example test_negb2 : (negb false) = true.
```

Conjunção

```
Definition andb (x y: bool) : bool :=
  match (x, y) with
    | (true, true) => true
    | _ => false
  end.
```

```
Notation "x /\ y" := (andb x y).
```

Tabela Verdade - Conjunção

```
Example test_andb1 : (andb true true) = true.
```

```
Example test_andb2 : (andb true false) = false.
```

```
Example test_andb3 : (andb false true) = false.

Example test_andb4 : (andb false false) = false.
```

Disjunção

```
Definition orb (x y: bool) : bool :=
  match (x, y) with
    | (false, false) => false
    | _ => true
  end.

Notation "x \/ y" := (orb x y).
```

Tabela Verdade - Disjunção

```
Example test_orb1 : (orb true true) = true.

Example test_orb2 : (orb true false) = true.

Example test_orb3 : (orb false true) = true.

Example test_orb4 : (orb false false) = false.
```

Implicação

```
Definition implb (x y: bool) : bool :=
  match (x, y) with
    | (true, false) => false
    | _ => true
  end.

Notation "x -> y" := (implb x y).
```

Tabela Verdade - Implicação

```
Example test_implb1 : (implb true true) = true.

Example test_implb2 : (implb true false) = false.

Example test_implb3 : (implb false true) = true.

Example test_implb4 : (implb false false) = true.
```

Bi-implicação

```
Definition biimplb (x y: bool) : bool :=
  match (x, y) with
    | (true, true) => true
    | (false, false) => true
    | _ => false
  end.

Notation "x <-> y" := (biimplb x y).
```

Tabela Verdade - Bi-implicação

```
Example test_biimplb1 : (biimplb true true) = true.

Example test_biimplb2 : (biimplb true false) = false.

Example test_biimplb3 : (biimplb false true) = false.

Example test_biimplb4 : (biimplb false false) = true.
```

Exercise: 1 star, standard (nandb)

```
Definition nandb (x y: bool) : bool :=
  match (x, y) with
    | (true, true) ⇒ false
    | _ ⇒ true
  end.

Example test_nandb1: (nandb true false) = true.

Example test_nandb2: (nandb false false) = true.

Example test_nandb3: (nandb false true) = true.

Example test_nandb4: (nandb true true) = false.
```

Exercise: 1 star, standard (andb3)

```
Definition andb3 (x y z: bool) : bool :=
  match (x, y, z) with
    | (true, true, true) ⇒ true
    | _ ⇒ false
  end.

Example test_andb31: (andb3 true true true) = true.

Example test_andb32: (andb3 false true true) = false.

Example test_andb33: (andb3 true false true) = false.

Example test_andb34: (andb3 true true false) = false.
```

Naturais

```
Definition minustwo (x : nat) : nat :=
  match x with
    | O ⇒ O
    | S O ⇒ O
    | S (S n') ⇒ n'
  end.
```

Par

```
Fixpoint evenb (x: nat) : bool :=
  match x with
    | O ⇒ true
    | S O ⇒ false
    | S (S x') ⇒ evenb x'
  end.
```

```
Example test_evenb1 : (evenb 2) = true.
```

```
Example test_evenb2 : (evenb 3) = false.
```

Ímpar

```
Fixpoint oddb (x: nat) : bool :=
  match x with
    | O ⇒ false
    | S O ⇒ true
    | S (S x') ⇒ oddb x'
  end.
```

```
Example test_oddb1 : (oddb 3) = true.
```

```
Example test_oddb2 : (oddb 2) = false.
```

Adição

```
Fixpoint plus (x y : nat) : nat :=
  match x with
    | O ⇒ y
    | S x' ⇒ S (plus x' y)
  end.
```

```
Notation "x + y" := (plus x y) (at level 50, left associativity) : nat_scope.
```

```
Example test_plus1 : (plus 2 3) = 5.
```

Subtração

```
Fixpoint minus (x y:nat) : nat :=
  match x, y with
    | O, _ ⇒ O
    | S _ , O ⇒ x
    | S x', S y' ⇒ minus x' y'
  end.
```

```
Notation "x - y" := (minus x y) (at level 50, left associativity) : nat_scope.
```

```
Example test_minus1 : (minus 5 2) = 3.
```

Multiplicação

```
Fixpoint mult (x y : nat) : nat :=
```

```
  match x with
    | O ⇒ O
    | S x' ⇒ plus y (mult x' y)
  end.

Notation "x * y" := (mult x y) (at level 40, left associativity) : nat_scope.

Example test_mult1: (mult 3 4) = 12.
```

Potenciação

```
Fixpoint exp (b p : nat) : nat :=
  match p with
    | O ⇒ S O
    | S p' ⇒ mult b (exp b p')
  end.

Notation "x ^ y" := (exp x y).

Example test_exp1 : (exp 3 2) = 9.
```

Exercise: 1 star, standard (factorial)

```
Fixpoint factorial (x: nat) : nat :=
  match x with
    | O ⇒ 1
    | S x' ⇒ x × (factorial x')
  end.

Example test_factorial1: (factorial 3) = 6.

Example test_factorial2: (factorial 5) = (mult 10 12).
```

Função 'igual que'

```
Fixpoint eqb (x y : nat) : bool :=
  match x with
    | O ⇒ match y with
      | O ⇒ true
      | S y' ⇒ false
    end
    | S x' ⇒ match y with
      | O ⇒ false
      | S y' ⇒ eqb x' y'
    end
  end.

Notation "x = y" := (eqb x y) (at level 70) : nat_scope.

Example test_eqb1 : (eqb 3 3) = true.
```

```
Example test_eqb2 : (eqb 2 3) = false.
```

Função 'menor ou igual que'

```
Fixpoint leb (x y : nat) : bool :=
  match x with
    | O ⇒ true
    | S x' ⇒ match y with
      | O ⇒ false
      | S y' ⇒ leb x' y'
      end
  end.
Notation "x <= y" := (leb x y) (at level 70) : nat_scope.
Example test_leb1 : (leb 2 2) = true.
Example test_leb2 : (leb 2 4) = true.
Example test_leb3 : (leb 4 2) = false.
```

Exercise: 1 star, standard (ltb) - Função 'menor que'

```
Definition ltb (x y : nat) : bool :=
  (andb (leb x y) (negb (eqb x y))).
Notation "x < y" := (ltb x y) (at level 70) : nat_scope.
Example test_ltb1: (ltb 2 2) = false.
Example test_ltb2: (ltb 2 4) = true.
Example test_ltb3: (ltb 4 2) = false.
```

Exemplo rewrite

```
Theorem plus_id_example : ∀ x y:nat,
  x = y →
  x + x = y + y.
Theorem plus_id_example' : ∀ x y:nat,
  x = y →
  x + x = y + y.
```

Exercise: 1 star, standard (plus_id_exercise)

```
Theorem plus_id_exercise :
  ∀ x y z : nat,
  x = y →
  y = z →
  x + y = y + z.
```

Exercise: 2 stars, standard (mult_S_1)

Theorem *mult_S_1* :
  $\forall\ x\ y\ :\ nat,$
  $y = S\ x \rightarrow$
  $y \times (1 + x) = y \times y.$

Dupla Negação

Theorem *negb_involutive* :
  $\forall\ x\ :\ bool,$
  $negb\ (negb\ x) = x.$

Theorem *negb_involutive'* :
  $\forall\ x\ :\ bool,$
  $negb\ (negb\ x) = x.$

Theorem *negb_involutive''* :
  $\forall\ x\ :\ bool,$
  $negb\ (negb\ x) = x.$
Comutação Theorem *andb_commutative* :
  $\forall\ x\ y,$
  $andb\ x\ y = andb\ y\ x.$

Theorem *andb_commutative'* :
  $\forall\ x\ y,$
  $andb\ x\ y = andb\ y\ x.$

Theorem *andb_commutative''* :
  $\forall\ x\ y,$
  $andb\ x\ y = andb\ y\ x.$
Exercise: 2 stars, standard (andb_true_elim2) Theorem *andb_true_elim2* :
  $\forall\ x\ y\ :\ bool,$
  $andb\ x\ y = true \rightarrow$
  $y = true.$

Theorem *andb_true_elim2'* :
  $\forall\ x\ y\ :\ bool,$
  $andb\ x\ y = true \rightarrow$
  $y = true.$

Exercise: 1 star (zero_nbeq_plus_1)

Theorem *zero_nbeq_plus_1* :
  $\forall\ x\ :\ nat,$
  $(0 = x + 1) = false.$

Theorem *zero_nbeq_plus_1'* :
  $\forall\ x\ :\ nat,$
  $(0 = x + 1) = false.$
Exercise: 2 stars (boolean_functions) Theorem *identity_fn_applied_twice* :
  $\forall\ (f\ :\ bool \rightarrow bool),$

```
  (∀ (x : bool), f x = x) →
  ∀ (b : bool), f (f b) = b.
```

Exercise: 1 star, standard (negation_fn_applied_twice)

```
Theorem negation_fn_applied_twice :
  ∀ (f : bool → bool),
  (∀ (x : bool), f x = negb x) →
  ∀ (b : bool), f (f b) = b.
```

Exercise: 2 stars (andb_eq_orb)

```
Theorem andb_eq_orb :
  ∀ (x y : bool),
  (andb x y = orb x y) →
  x = y.
```

Exercise: 3 stars, standard (binary)

```
Inductive bin : Type :=
  | Z : bin
  | A : bin → bin
  | B : bin → bin.

Fixpoint incr (x : bin) : bin :=
  match x with
  | Z ⇒ B Z
  | A x' ⇒ B x'
  | B x' ⇒ A (incr x')
  end.

Fixpoint bin_to_nat (x : bin) : nat :=
  match x with
  | Z ⇒ O
  | A x' ⇒ mult 2 (bin_to_nat x')
  | B x' ⇒ S (mult 2 (bin_to_nat x'))
  end.

Example inc_three_four: (bin_to_nat (incr (B (B Z)))) = 4.

Example inc_nine_ten: (bin_to_nat (incr (B (A (A (B Z)))))) = 10.

Example zero_is_zero: (bin_to_nat Z) = 0.

Example five_is_five: (bin_to_nat (B (A (B Z)))) = 5.

Fixpoint incN (n:nat) (m:bin) :=
  match n with
  | 0 ⇒ m
  | S n' ⇒ incN n' (incr m)
  end.

Example SanityCheck: bin_to_nat (incN 15 Z) = 15.
```