Functional Programming in Coq (Basics.v)

Keywords

- `Inductive` : define a type of type `Type`
- `Definition` : define functions
- `Compute` : evaluate expressions
- `Example` : example with a name and an assertion
- `Proof. Qed.` : proof
- `Check` : print the type info
- `Module X. End X.`
- `Fixpoint` : define recursive functions

Tactics

- `simpl`
- `reflexivity`
- `intros n.` Move `n` from the quantifier in the goal to a context of current assumptions
- `rewrite -> H` Prove by Rewrite. Rewrite the current goal by replacing the left side of the equality `H` with the right side
- `destruct n as [ | ] eqn:E.` Prove by Case Analysis.
- `intros [ | ]` Perform case analysis on a variable when introducing it
- `intros []` No arguments to name

List of Types, Definitions, Notations, (Useful) Examples

- `Inductive day : Type`
- `Defintion next_weekday (d: day) : day`
-
- `Inductive bool : Type`

- Defintion negb (b: bool) : bool
- Definition andb (b: bool) : bool
- Definition orb (b: bool) : bool
- Notation "x && y"
- Notation "x || y"
- Definition andb3
-
- Inductive rgb : Type
- Inductive color : Type
- Definition monochrome (c: color) : bool
- Definition isred (c: color) : bool
-
- Inductive bit : Type
- Inductive nybble : Type
- Definition all_zero (nb : nybble) : bool
-
- Module NatPlayground.
- Inductive nat : Type
- Definition pred (n : nat) : nat
- End NatPlayground.
-
- Definition minustwo (n : nat) : nat
- Fixpoint evenb (n : nat) : bool
- Definition oddb (n : nat) : bool
-
- Module NatPlayground2.
- Fixpoint plus (n : nat) (m : nat) : nat
- Fixpoint mult (n m : nat) : nat
- Fixpoint minus (n m : nat) : nat
- End NatPlayground2.
-
- Fixpoint exp (base power : nat) : nat
- Fixpoint factorial (n : nat) : nat
-
- Notation "x + y"
- Notation "x - y"
- Notation "x * y"

-
- `Fixpoint eqb (n m : nat) : bool`
- `Fixpoint leb (n m : nat) : bool`
-
- `Notation "x =? y"`
- `Notation "x <=? y"`
- `Definition ltb (n m : nat) : bool`
- `Notation "x <? y"`
-
- `Theorem plus_O_n : forall n : nat, O + n = n.` : `O` is the left identity
- `Theorem plus_1_l : forall n : nat, 1 + n = S n.`
- `Theorem mult_0_l : forall n : nat, 0 * n = 0.`
- `Theorem plus_id_example : forall n,m : nat, n = m -> n + n = m + m.`
- `Theorem prove_id_exercise`
- `Theorem multi_0_plus`: `(0 + n) * m = n * m`
-
- `Theorem plus_1_neq_0`
- `Theorem negb_involutive`
- `Theorem andb_commutative`
- `Theorem andb3_exchange`
-
- `Theorem addb_true_elim2`
- `Theorem zero_nbeq_plus_1`
- `Theorem identify_fn_applied_twice`
- `Theorem negation_fn_applied_twice`
- `Theorem andb_eq_orb`
-
- `Inductive bin : Type`
- `Fixpoint incr (m : bin) : bin`
- `Fixpoint bin_to_nat (m : bin) : nat`

Problems

- `Theorem addb_true_elim2` Simple proof?