IndProp: Inductively Defined Propositions (`IndProp.v`)

Keywords

- `Inductive` Inductive declaration, where each constructor corresponds to an inference rule

Tactics

- `inversion E as [| n' E'].`
- `inversion`
    - applied to equalities: `discriminate` and `injection` (also does necessary `intros + rewrite`)
    - applied to analyze evidence for inductively defined propositions
- `induction E as [|n' E' IH].` induction on `E` which is an inductively defined proposition

Defintions and Theorems

Inductively Defined Propositions

- `Inductive even : nat -> Prop := | ev_0 : even 0 | ev_SS (n : nat) (H : even n) : even (S (S n)).`
- `Theorem even_4 : even 4.` Use `apply` with the rule names
- `Theorem even_4' : even 4.` Use function application syntax
- `Theorem ev_plus4 : forall n, even n -> even (4 + n).`
- `Theorem ev_double : forall n, even (double n).`

Using Evidence in Proofs

- Theorem ev_inversion : forall (n : nat), even n -> (n = 0) \/ (exists n', n = S (S n') /\ even n').
- Theorem ev_minus2 : forall n, even n -> even (pred (pred n)).
- Theorem evSS_ev : forall n, even (S (S n)) -> even n.
- Theorem evSS_ev' : forall n, even (S (S n)) -> even n.
- Theorem one_not_even : ~ even 1.
- Theorem one_not_even' : ~ even 1.
- Theorem SSSSev_even : forall n, even (S (S (S (S n)))) -> even n.
- Theorem even5_nonsense : even 5 -> 2 + 2 = 9.

Induction on Evidence

- Lemma ev_even : forall n, even n -> exists k, n = double k.
- Theorem ev_even_iff : forall n, even n <-> exists k, n = double k.
- Theorem ev_sum : forall n m, even n -> even m -> even (n + m).
- Inductive even' : nat -> Prop :=
- Theorem even'_ev : forall n, even' n <-> even n.
- Theorem ev_ev__ev : forall n m, even (n + m) -> even n -> even m.
- Theorem ev_plus_plus : forall n m p, even (n + m) -> even (n + p) -> even (m + p).

Inductive Relations

- Module Playground.
- Inductive le : nat -> nat -> Prop := | le_n n : le n n | le_S n m (H: le n m) : le n (S m).
- Notation "m <= n" := (le m n).
- End Playground.
- Definition lt (n m:nat) := le (S n) m.
- Notation "m < n" := (lt m n).
- Inductive square_of : nat -> nat -> Prop := | sq n : square_of n (n * n).
- Inductive next_nat : nat -> nat -> Prop := | nn n : next_nat n (S n).
- Inductive next_even : nat -> nat -> Prop := | ne_1 n : even (S n) -> next_even n (S n) | ne_2 n (H : even (S (S n))) : next_even n (S (S n)).
-

Problems

- Theorem `ev_ev__ev : forall n m, even(n + m) -> even n -> even m.` `induction En as [| n' Hn' IHn']` Why is `IHn' : even(n' + m) -> even m` ?

> Written with [StackEdit](https://stackedit.io/).