

Definicje indukcyjne w Coqu (bliżej praktyki)

17 marca 2009

- taktyki wprowadzania - `constructor`, `left`, `right`, `split`, `exists`
- taktyki destrukcji - `destruct`, `case`
- generowanie lematów indukcyjnych
- taktyki indukcyjne - `induction`, `elim`
- typy indukcyjne i równość - `discriminate`, `injection`, `simplify_eq`
- zaawansowana destrukcja - `inversion`
- definicja równości - `eq`
- dowodzenie i używanie równości - `reflexivity`, `rewrite`

Taktyki wprowadzania

- apply c_i
- constructor i
- constructor
- left, right = constructor 1, constructor 2
(o ile są tylko 2 konstruktory)
- split = constructor 1 = constructor
(o ile jest tylko 1 konstruktor)
- exists $t = \text{split with } t$
(o ile jest tylko 1 konstruktor)

Taktyki destrukcji

- `destruct`
- `simple destruct`
- `case` (podstawowa)

- `intros`

Lematy indukcyjne

Dla (przyzwoitej) definicji postaci

Inductive $I (\overrightarrow{r : R}) : \forall \overrightarrow{z : t}, s := c_1 : C_1 \mid \dots \mid c_n : C_n$

tworzone są lematy:

I_rect, I_rec oraz I_ind o następujących typach:

I_ind : forall (r:R) (P : forall z:t, I r z -> Prop),
 $\hat{C}_1 \rightarrow \dots \rightarrow \hat{C}_n \rightarrow \text{forall } (z:t) (x:I r z), P z x$

Lematy indukcyjne

Dla (przyzwoitej) definicji postaci

Inductive $I (\overrightarrow{r : R}) : \forall \overrightarrow{z : t}, s := c_1 : C_1 \mid \dots \mid c_n : C_n$

tworzone są lematy:

I_rect , I_rec oraz I_ind o następujących typach:

$I_ind : forall (r : R) (P : forall z : t, I r z -> Prop),$

$\hat{C}_1 -> \dots -> \hat{C}_n -> forall (z : t) (x : I r z), P z x$

gdzie \hat{C}_i to następująca modyfikacja C_i :

jeśli $C_i = \forall \overrightarrow{y : U}, Irq$ to:

- końcowe wystąpienie Irq zastąpione jest przez $Pq(c_i \vec{y})$
- jeśli $U_j = Irq'$, po U_j dopisywany jest jeszcze jeden argument $Pq' y_j$
- a jeśli $U_j = \forall \overrightarrow{z : \vec{U}}, Irq'$, to po U_j dopisywany jest argument $\forall \overrightarrow{z : \vec{U}}, Pq'(y_j \vec{z})$.

Dowód przez fix i match...

Lematy indukcyjne (c.d.)

Inductive $I(\overrightarrow{r : R}) : \forall \overrightarrow{z : t}, s := c_1 : C_1 \mid \dots \mid c_n : C_n$

$I_ind : \text{forall } (r : R) (P : \text{forall } z : t, I\ r\ z \rightarrow \text{Prop}),$
 $\hat{C}_1 \rightarrow \dots \rightarrow \hat{C}_n \rightarrow \text{forall } (z : t) (x : I\ r\ z), P\ z\ x$

Pozostałe lematy (o ile są)

- I_rec — **Set**
- I_rect — **Type**

Jak $s = \text{Prop}$, to tylko I_ind , w dodatku niezależny
(czyli P ma typ $\text{forall } z : t, \text{Prop}$)

Taktyki indukcyjne

- induction
- simple induction
- elim (podstawowa)
- elimtype

Dla typów wzajemnie rekurencyjnych automatyczne lematy są do niczego.
Aby wygenerować dobre, należy użyć:

```
Scheme id1 := Induction for  $I_i$  Sort  $s_1$   
with ...  
idn := Induction for  $I_n$  Sort  $s_n$ 
```

lub w wersji niezależnej:

```
Scheme id1 := Minimality for  $I_i$  Sort  $s_1$   
with ...  
idn := Minimality for  $I_n$  Sort  $s_n$ 
```

Typy indukcyjne i równość

- $0=1 \rightarrow ?$ `discriminate`
- $S\ x = S\ y \rightarrow x = y ?$ `injection`
- jedno lub drugie: `simplify_eq`

Jak to się mogło stać ?

- `inversion`
(wnioski z (Even n) oraz discriminate i injection)
- `inversion_clear`
(j.w. ale trochę czyści)
- `dependent inversion`
(jeśli dane założenie występuje w „celu”)

Typ indukcyjny: równość

- równość Leibniza `eq`
- dowodzenie: `reflexivity` lub `split` lub `constructor` lub `apply`
`eq_refl`
- dowodzenie: `symmetry`, `transitivity`
- używanie: `rewrite`, `rewrite <-`, `subst`, `replace`