Heq: A Coq library for Heterogeneous Equality

Chung-Kil Hur

09 / Jul / 2010

@ Coq Workshop 2010

Dependent types

- One of the main strengths of Coq
  : Dependent types
- However, sometimes dealing with dependent types is a big overhead due to intentional type theory.
- We develop Heq to reduce such an overhead.

Motivating example

- ```
  Inductive vector : nat → Type :=
  | vnil : vector 0
  | vcons (hd:nat) n (tl:vector n) : vector (S n)
  ```

- ```
  ++ : ∀n m, vector n → vector m → vector (n+m)
  ```

Problem with equality

? $\forall n, (v : \text{vector } n),\quad v +\!\!+ \text{vnil} = v$

$v +\!\!+ \text{vnil} : \text{vector } (n+0)$; $v : \text{vector } n$

? $\forall n, (v : \text{vector } n),\quad v +\!\!+ \text{vnil} = \text{eq\_rect}\ \_\ (\text{vector } v)\ \_\ (\text{plus\_n\_O } n)$

$\text{vector } v : \text{vector } n$; $\text{plus\_n\_O } n : \text{proof of } n+0 = n$

? $\forall n, (v : \text{vector } n),\quad \text{JMeq}\ (v +\!\!+ \text{vnil})\ v$

$\simeq\ (\text{vector } n, v) = (\text{vector } (n+0), v +\!\!+ \text{vnil}) : \coprod_{ty \in \text{Type}} ty$

? $\forall n, (v : \text{vector } n),\quad \text{eq\_dep}\ \text{vector}\ (v +\!\!+ \text{vnil})\ v$

$\simeq\ (n, v) = (n+0, v +\!\!+ \text{vnil}) : \coprod_{n \in \mathbb{N}} (\text{vector } n)$

Heq : equality

We use dependent product.

Notation : $\{\{ \text{vector } \# \; V == V' \}\}$

$\overset{\text{def}}{=\!=} \quad (n, v) = (n', v') \; : \coprod_{n \in \mathbb{N}} \text{vector } n$

? $\forall n, v : \text{vector } n, \quad \{\{ \text{vector } \# \; v \; +\!+ \; \text{vnil} == v \}\}$

this can be inferable using type classes

"Matthieu Sozeau"

Heq : casts

Notation : «vector #C» v $\stackrel{\text{def}}{=}$ eq_rect _ vector v _ C

E.g.) Program Fixpoint vrev n (v: vector n) : vector n

= match v with

| vnil ⇒ vnil

| vcons hd tl ⇒ «vector # _» rev tl ++ (hd :: vnil)

end.

Next obligation. --- . Defined.

rewriting of heterogeneous equality

$$\{\{\ \text{vector}\ \#\ v \mathbin{++} \text{vnil} == v\ \}\}$$

- Inductive Case

$$\frac{\begin{array}{c}\vdots \\ IHv\ :\ \{\{\ \text{vector}\ \#\ v \mathbin{++} \text{vnil} == v\ \}\}\end{array}}{\{\{\ \text{vector}\ \#\ hd :: (v \mathbin{++} \text{vnil}) == hd :: v\ \}\}}$$

Demonstration

Algorithm

{{ v ++ vnil == v }}

v ++ vnil = «pf» v : vector (n+o)

pf : n+o=n

rewrite

hd :: (v++vnil) == hd :: v → hd :: «pf» v == hd :: v

↓ generalize

elimination of casts

∀ c : n+o=n , hd :: «c» v == hd :: v

↓ rewrite n+o = n

∀ c : n=n , hd :: «c» v == hd :: v

||

v

Elimination & relocation of casts

Demonstration

hpattern

```
n, m : nat
P : ∀ n m l, vector n → vector m → Prop
H : S n = m
```

```
∀ (v : vector (S n)) (w : vector n), P (S n) (S n) (S n) v (0 :: w)
```

v : vector (S n)

0 :: w : vector (S n)

rewrite H ⇝ fail

pattern (S n) ⇝ fail

pattern (S n) at 1, 2 → succeed

hrewrite H at 1 → succeed

or hpattern (S n) at 1

→ sound & complete & linear & easy

hpattern : algorithm

hpattern (S n) at 1

∀ (v : vector (S n)) (w : vector n), P (S n) (S n) (S n) v (0 :: w)

↓

∀ (v : vector k) (w : vector n), P _ _ _ v (0 :: w)

↓ type check : fail — cannot infer

∀ (v : vector k) (w : vector n), P _ _ _ v (0 :: w)

↑ fill (S n)

↓ type check

∀ (v : vector k) (w : vector n), P k (S n) (S n) v (0 :: w)

↑ infer ↑ infer ↑ I provided

Summary

Heq :- simple notations for heterogeneous equality & casts

- elimination, relocation of casts ⇝ rewriting of heq
- hpattern
- iterated dependent pairs

See http://www.pps.jussieu.fr/~gil/Heq for more details