Coq-GT

Enrico Tassi

5 July 2011 — Paris

Roadmap

Simplify

Contextual rewrite patterns

simpl: my patch in a nutshell

Simpl unfolds a constant if it contains a fixpoint and the recursive argument evaluates to a constructor.

```
Eval simpl in (S x - y)
  match y with 0 => S x | S y' => x - y end
```

With my patch you can tell simpl to behave as follows:

```
Eval simpl in (S x - y)
  (S x - y)
Eval simpl in (S x - S y)
  (x - y)
Eval simpl in ((f \o g) x)
  f (g x)
Eval simpl in (f \o g)
  f \o g
```

In short, you choose how many arguments must be passed and which must evaluato to a constructor for a constant to be unfolded by simpl.

Attaching meta-data to definition's arguments

Right now:

```
Definition fcomp A B C (f:B -> C) (g:A -> B) x := f (g x).
Infix "\o" := fcomp (at level 50, left associativity).
Implicit Arguments fcomp [A B C].
Arguments Scope fcomp [type_scope type_scope _ _ _ _].
Recursive Arguments fcomp [] 6.
```

What would be nice for the Math-Comp library

```
Arguments fcomp [[A%type B%type C%type]] f g x /.
```

Other examples:

```
Arguments nth [S%type] def !s%seq !i%nat.
Arguments foo [T%type] !n%nat !t / m g.
```

Because:

```
> grep '^Implicit Arguments' *.v  | wc -l    #528
> grep '^Arguments Scope' *.v  | wc -l       #158
```

Why occurrence numbers are bad

```
...
g := [morphism of sdprodm defXA phiAiM] : {morphism
    joing_group A X >-> gT}
kerg : 'ker g = 'Mho^1(A)
skk : 'ker (coset ('ker g)) \subset 'ker g
nkA : joing_group A X \subset 'N('ker g)
fact_g := factm skk nkA : coset_groupType ('ker g) -> gT
imgX : X = fact_g @* (X / 'ker g)
nAA1 : A \subset 'N('Mho^1(A))
nXA1 : X \subset 'N('Mho^1(A))
============================
 minnormal (fact_g @* (A / 'ker g)) X ->
 minnormal (A / 'ker g) (X / 'ker g)

rewrite {1}imgX
```

Why occurrence numbers are bad

```
...
g := [morphism of sdprodm defXA phiAiM] : {morphism
    joing_group A X >-> gT}
kerg : 'ker g = 'Mho^1(A)
skk : 'ker (coset ('ker g)) \subset 'ker g
nkA : joing_group A X \subset 'N('ker g)
fact_g := factm skk nkA : coset_groupType ('ker g) -> gT
imgX : X = fact_g @* (X / 'ker g)
nAA1 : A \subset 'N('Mho^1(A))
nXA1 : X \subset 'N('Mho^1(A))
============================
 minnormal (fact_g @* (A / 'ker g)) X ->
 minnormal (A / 'ker g) (X / 'ker g)

rewrite {1}imgX

rewrite {29}imgX
```

Why occurrence numbers are bad

```
is_true (@minnormal gT (@morphim (@coset_groupType gT (@ker gT gT (@gval gT (@joing_group gT
(@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT
(@gval gT A) (@gval gT X))) g)))) gT (@morphim gT (@coset_groupType gT (@ker gT gT (@gval gT
(@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT
(@joing_group gT (@gval gT A) (@gval gT X))) g)))) (@gval gT (@normaliser_group gT (@ker gT gT
(@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT
(@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g))))) (@coset_morphism gT (@ker gT gT
(@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT
(@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g)))) (@MorPhantom gT (@coset_groupType
gT (@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT
(@mfun gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g)))) (@mfun gT
(@coset_groupType gT (@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g
(@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g))))
(@gval gT (@normaliser_group gT (@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT
X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT
X))) g))))) (@coset_morphism gT (@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT
X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT
X))) g)))))) (@gval gT (@joing_group gT (@gval gT A) (@gval gT X)))) (@factm_morphism gT
(@coset_groupType gT (@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g
(@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g))))
gT (@joing_group gT (@gval gT A) (@gval gT X)) (@normaliser_group gT (@ker gT gT (@gval gT
(@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT
(@joing_group gT (@gval gT A) (@gval gT X))) g)))) g (@coset_morphism gT (@ker gT gT (@gval gT
(@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT
(@joing_group gT (@gval gT A) (@gval gT X))) g)))) skk nkA) (@MorPhantom (@coset_groupType gT
(@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun
gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g)))) gT fact_g) (@quotient gT
(@gval gT A) (@ker gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g (@MorPhantom
gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A) (@gval gT X))) g))))) (@gval gT
X)) ->
is_true (@minnormal (@coset_groupType gT (@ker gT gT (@gval gT (@joing_group gT (@gval gT A)
(@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A)
(@gval gT X))) g)))) (@quotient gT (@gval gT A) (@ker gT gT (@gval gT (@joing_group gT (@gval
gT A) (@gval gT X))) g (@MorPhantom gT gT (@mfun gT gT (@gval gT (@joing_group gT (@gval gT A)
```

Why occurrence numbers are bad

Occurrence numbers are bad for the following reasons:

- can be hard to write
- scripts are less informative when they break

SSR 1.3 contextual patterns:

- specify the occurrences looking at their context

```
minnormal (fact_g @* (A / 'ker g)) X ->
  minnormal (A / 'ker g) (X / 'ker g)
```

One gives the context

```
rewrite [      minnormal (_ @* _) _]imgX
```

Why occurrence numbers are bad

Occurrence numbers are bad for the following reasons:

- can be hard to write
- scripts are less informative when they break

SSR 1.3 contextual patterns:

- specify the occurrences looking at their context

```
minnormal (fact_g @* (A / 'ker g)) X ->
  minnormal (A / 'ker g) (X / 'ker g)
```

One gives the context and names a hole

```
rewrite [R in minnormal (_ @* _) R]imgX
```

Rewrite (contextual) patterns

Terminology

matching head constant driven

```
addnC : forall a b, a + b = b + a
```

redex the term being rewritten, identified with a given patter or a pattern inferred looking at the rule

Rewrite pattern syntax

```
rewrite rule
rewrite [t]rule
rewrite [in t]rule
rewrite [X in t]rule
rewrite [in X in t]rule
rewrite [e in X in t]rule
rewrite [e as X in t]rule
```

Rewrite patterns — example 1

The rule

```
addnC : _ + _ = _ + _
```

The tactic invocation

```
rewrite addnC.
```

The goal

```
(x + y) + f x (x + y).+1 = 0
```

Rewrite patterns — example 2

The rule

```
(addnC x.+1) : x.+1 + _ = _ + x.+1
```

The tactic invocation

```
rewrite [_.+1](addnC x.+1).
```

The goal

```
(x + y) + f x (x + y).+1 = 0
```

Because `(x + y).+1` $=$ `x.+1 + _`

Contextual rewrite patterns — example 3

The rule

```
addnC : _ + _ = _ + _
```

The tactic invocation

```
rewrite [in f _ _]addnC.
```

The goal

```
(x + y) + f x (x + y).+1 = 0
```

Contextual rewrite patterns — example 4

The rule

```
(addnC x.+1) : x.+1 + _ = _ + x.+1
```

The tactic invocation

```
rewrite [R in f _ R](addnC x.+1).
```

The goal

```
(x + y) + f (x.+1 + y) (x + y).+1 = 0
```

Because `R` captured `(x + y).+1` $=$ `x.+1 + _`

Contextual rewrite patterns — example 5

The rule

```
(addnC x) : x + _ = _ + x
```

The tactic invocation

```
rewrite [in R in f _ R](addnC x).
```

The goal

```
(x + y) + f x (z + (x + y).+1) = 0
```

Because `R` captured `z + (x + y).+1`

Contextual rewrite patterns — example 6

The rule

```
(addnC x.+1) : x.+1 + _ = _ + x.+1
```

The tactic invocation

```
rewrite [_.+1 in R in f _ R](addnC x.+1).
```

The goal

```
(x + y) + f x (z + (x + y).+1) = 0
```

Because `R` captured `z + (x + y).+1` and `_.+1` matched `(x + y).+1` = `x.+1 + _`

Contextual rewrite patterns — example 7

The rule

```
addnC : _ + _ = _ + _
```

The tactic invocation

```
rewrite [x.+1 + y as R in f _ (_ + R)]addnC.
```

The goal

```
(x + y) + f x (z + (x + y).+1) = 0
```

Because `R` captured `(x + y).+1` $=$ `x.+1 + y` $=$ `_ + _`

Contextual patterns — the future

Patterns everywhere:

```
set t := {3}(a + _).
set t := (a + _ in R in _ = R).
```

With user defined shortcuts

```
Notation RHS := (X in _ = X)%pattern.
set t := (a + _ in RHS).
move: (a + _ in RHS).
rewrite {2}[in RHS]addnC.
elim: (n in RHS).
```

Questions:

- Do you like this idea?
- Could this feature be part of Coq?

I've a branch in which the ssreflect plugin is split into an ssrmatching plugin and the ssreflect plugin builds on top of it.