using reify to derive a formula for NATMUL

Johan G. F. Belinfante
2003 January 23

```
<< goedel52.r07; << tools.m
```

```
:Package Title: goedel52.r07          2003 January 21 at 11:15 p.m.

It is now:  2003 Jan 23 at 8:25

Loading Simplification Rules

TOOLS.M                      Revised 2002 December 27

weightlimit = 40
```

■ summary

A new formula relating **power** to **invar** is derived with applications to powers of inverse functions. This formula is used to discover a formula for **NATMUL** that is slightly more compact than what was previously known. The entire theory of multiplication of natural numbers could be built up starting with this result. The discovered formula for **NATMUL** is verified by an independent derivation.

■ the new invar formula

The following conditional rewrite rule holds for inverse functions:

```
subvar[inverse[ff]]
```

```
intersection[invar[ff], P[domain[ff]]]
```

Because of this formula, one needs a new rewrite formula for powers of inverse functions. This is the new result:

```
SubstTest[U, subvar[
  union[cross[composite[id[omega], SUCC], cross[Id, z]], id[cart[singleton[0], Id]]]],
    z → inverse[ff]]
```

```
U[intersection[invar[cross[composite[inverse[SUCC], id[omega]], cross[Id, ff]]],
   P[union[cart[intersection[omega, complement[singleton[0]]], cart[V, domain[ff]]],
     cart[singleton[0], Id]]]]] == composite[SWAP, power[ff]]
```

To facilitate pattern matching, we modify this equation slightly, and reformulate it as a new conditional rewrite rule:

```
U[intersection[invar[cross[composite[inverse[SUCC], id[omega]], cross[Id, f_]]],
   P[union[cart[intersection[omega, complement[singleton[0]]], cart[V, x_]],
     cart[singleton[0], Id]]]]] :=
 composite[SWAP, power[f]] /; FUNCTION[f] && x == domain[f]
```

■ plus[x]

In particular, since the integer **plus[x] = composite[NATADD,RIGHT[x]]** is a bijection, the new formula applies to powers of **plus[x]**. One also needs to make use of the following result, which says that **power[inverse[plus[x]]]** is equal to **composite[SWAP,power[plus[x]]]**.

```
SubstTest[power, inverse[y], y → plus[x]]
```

```
power[composite[inverse[RIGHT[x]], inverse[NATADD]]] ==
 composite[SWAP, power[composite[NATADD, RIGHT[x]]]]
```

```
power[composite[inverse[RIGHT[x_]], inverse[NATADD]]] :=
 composite[SWAP, power[composite[NATADD, RIGHT[x]]]]
```

■ reify argument

The use of reify to derive a formula for **NATMUL** is speeded up by turning off the **simplify** flag.

```
simplify = False;
```

Here is how **reify** is used to discover a formula for the binary function **NATMUL** which multiplies natural numbers:

```
Map[rotate[inverse[#]] &,
 SubstTest[reify, x, composite[F[x], power[G[x]]], {F[x] → composite[
     id[image[V, intersection[omega, singleton[x]]]], inverse[LEFT[0]]], G → plus}]]
```

```
NATMUL ==
 composite[rotate[composite[complement[composite[complement[composite[rotate[NATADD],
        SWAP, RIF, cross[SECOND, composite[SWAP, SECOND]], id[composite[
          id[cart[omega, V]], inverse[FIRST], SUCC, FIRST]], cross[inverse[E], Id]]],
      id[composite[inverse[E], IMAGE[id[complement[cart[singleton[0], Id]]]]]],
      inverse[FIRST]]],
    inverse[IMAGE[cross[Id, inverse[LEFT[0]]]]], E]], id[cart[omega, V]]]
```

■ independent verification

The formula for **NATMUL** can be verified without appeal to the theory of reification. The catch is that one needs to know what this formula is.

```
SubstTest[assert, forall[x, equal[composite[u, LEFT[x]], composite[v, LEFT[x]]]],
 {u → NATMUL,
  v -> composite[rotate[composite[complement[composite[complement[composite[rotate[
           NATADD], SWAP, RIF, cross[SECOND, composite[SWAP, SECOND]],
          id[composite[id[cart[omega, V]], inverse[FIRST], SUCC, FIRST]],
          cross[inverse[e], Id]]], id[composite[inverse[e],
          IMAGE[id[complement[cart[singleton[0], Id]]]]]], inverse[FIRST]]],
      inverse[IMAGE[cross[Id, inverse[LEFT[0]]]]], e]], id[cart[omega, V]]]}]
```

```
True == equal[NATMUL,
  composite[rotate[composite[complement[composite[complement[composite[rotate[NATADD],
         SWAP, RIF, cross[SECOND, composite[SWAP, SECOND]], id[composite[
           id[cart[omega, V]], inverse[FIRST], SUCC, FIRST]], cross[inverse[E], Id]]],
       id[composite[inverse[E], IMAGE[id[complement[cart[singleton[0], Id]]]]]],
       inverse[FIRST]]],
     inverse[IMAGE[cross[Id, inverse[LEFT[0]]]]], E]], id[cart[omega, V]]]]
```

■ comments

In practice one only needs to use the formula for **NATMUL** once, to derive a simpler formula for **composite[NATMUL, LEFT[x]]**. The latter quantity is much simpler to work with, and is what is actually needed in practice to derive the properties of multiplication. So the only role for the complicated formula for **NATMUL** itself is just to prove that it exists. Of course, if one is willing to simply add the existence of **NATMUL** as an additional axiom of set theory, then the formula would not be needed at all. That is, if one wants to be lazy, one could just add a new axiom that there exists a function satisfying the following equation:

```
composite[NATMUL, LEFT[x]] == composite[
  id[image[V, intersection[omega, singleton[x]]]], iterate[plus[x], singleton[0]]]
```

```
True
```

The identity function in this formula is needed to combine the cases that **x** is a natural number and the case that it is not a natural number. One could of course simply write out separately these two cases:

```
implies[member[x, omega],
 equal[composite[NATMUL, LEFT[x]], iterate[plus[x], singleton[0]]]]
```

```
True
```

```
implies[not[member[x, omega]], equal[composite[NATMUL, LEFT[x]], 0]]
```

```
True
```