

division of natural numbers

Johan G. F. Belinfante
2002 October 6

```
<< goedel52.p75; << tools.m

:Package Title: goedel52.p75          2002 October 5 at 11:30 p.m.

It is now: 2002 Oct 6 at 9:34

Loading Simplification Rules

TOOLS.M                      Revised 2002 September 16

weightlimit = 40
```

■ summary

This notebook contains a derivation of the fact that division of natural numbers corresponds to a binary function obtained by rotating `NATMUL`, and restricting denominators to be nonzero. The main tool is the rule that left-multiplication by a nonzero number is one-to-one. Two general facts are deduced which help to motivate the derivation, but the application to division requires an additional tool, namely the following observation:

```
implies[equal[0, fix[composite[u, v]]], equal[0, fix[composite[v, u]]]
True
```

This observation is the key to cleaning up the expressions that are encountered in the derivation of the result about division.

■ generalities about binary functions obtained by rotation

The present section provides some general insight into functions obtained by rotation. We begin with two lemmas.

```
SubstTest[assert, forall[z, FUNCTION[composite[w, RIGHT[z]]], w -> rotate[x]]

equal[0, fix[composite[FIRST,
  intersection[composite[inverse[x], FIRST], composite[inverse[SECOND], Di, SECOND]],
  inverse[rotate[composite[x, SWAP]]]]] == FUNCTION[rotate[x]]

equal[0, fix[composite[FIRST,
  intersection[composite[inverse[x_], FIRST], composite[inverse[SECOND], Di, SECOND]],
  inverse[rotate[composite[x_, SWAP]]]]] := FUNCTION[rotate[x]]

SubstTest[assert, forall[z, FUNCTION[composite[w, RIGHT[z]]], w -> rotate[flip[x]]]

equal[0, fix[composite[SECOND,
  intersection[composite[inverse[x], FIRST], composite[inverse[FIRST], Di, SECOND]],
  inverse[rotate[x]]]]] == FUNCTION[rotate[composite[x, SWAP]]]
```

```

equal[0, fix[composite[SECOND,
  intersection[composite[inverse[x_], FIRST], composite[inverse[FIRST], Di, SECOND]],
  inverse[rotate[x_]]]]] := FUNCTION[rotate[composite[x, SWAP]]]

```

With these two lemmas in place, one discovers the following general facts:

```

assert[forall[x, FUNCTION[composite[inverse[LEFT[x]], inverse[w]]]]]
FUNCTION[rotate[w]]

assert[forall[x, FUNCTION[composite[inverse[RIGHT[x]], inverse[w]]]]]
FUNCTION[rotate[composite[w, SWAP]]]

```

Although these facts are not directly applicable to the case at hand, they do provide the motivation for the derivation carried out below.

■ the case of interest is more complicated

The starting point will be this fact:

```

FUNCTION[composite[id[image[V, x]], inverse[LEFT[x]], inverse[NATMUL]]]
True

```

Lemma.

```

and[not[equal[0, x]],
  not[FUNCTION[composite[inverse[LEFT[x]], inverse[NATMUL]]]]] // NotNotTest
and[not[equal[0, x]],
  not[FUNCTION[composite[inverse[LEFT[x]], inverse[NATMUL]]]]] == False
and[not[equal[0, x_]],
  not[FUNCTION[composite[inverse[LEFT[x_]], inverse[NATMUL]]]]] := False

```

The lemma is needed to cope with the negation built into the universal quantifier in the following application:

```

SubstTest[assert, forall[x,
  FUNCTION[composite[id[image[V, x]], inverse[LEFT[x]], inverse[w]]]], w -> NATMUL]
True == subclass[fix[composite[FIRST, intersection[composite[inverse[NATMUL], FIRST],
  composite[inverse[SECOND], Di, SECOND]], inverse[rotate[NATMUL]]]], singleton[0]]
subclass[
  fix[composite[FIRST, intersection[composite[inverse[NATMUL], FIRST], composite[
    inverse[SECOND], Di, SECOND]], inverse[rotate[NATMUL]]]], singleton[0]] := True

```

■ reformulations

Note that this rule can be rewritten in the following homogeneous form:

```

equal[0, fix[composite[FIRST, intersection[
  composite[inverse[NATMUL], FIRST], composite[inverse[SECOND], Di, SECOND]],
  inverse[rotate[NATMUL]], id[complement[singleton[0]]]]]]

```

True

This observation is the basis for the following reformulation:

```

SubstTest[implies, equal[0, fix[composite[u, v]], equal[0, fix[composite[v, u]]],
  {u -> composite[FIRST, intersection[
    composite[inverse[NATMUL], FIRST], composite[inverse[SECOND], Di, SECOND]],
    v -> composite[inverse[rotate[NATMUL]], id[complement[singleton[0]]]]}]
equal[0, fix[composite[SWAP,
  cross[id[complement[singleton[0]]], Di], inverse[NATMUL], NATMUL]]] == True
fix[
  composite[SWAP, cross[id[complement[singleton[0]]], Di], inverse[NATMUL], NATMUL]] := 0

```

A second application of this idea removes the **SWAP**.

```

SubstTest[implies, equal[0, fix[composite[u, v]], equal[0, fix[composite[v, u]]],
  {u -> composite[SWAP, cross[id[complement[singleton[0]]], Di], inverse[NATMUL]],
  v -> NATMUL}]
equal[0, fix[
  composite[NATMUL, cross[id[complement[singleton[0]]], Di], inverse[NATMUL]]] == True
fix[composite[NATMUL, cross[id[complement[singleton[0]]], Di], inverse[NATMUL]]] := 0

```

The final step is to rewrite this in a more recognizable form:

```

SubstTest[equal, 0, fix[composite[w, cross[Id, Di], inverse[w]],
  w -> composite[NATMUL, cross[id[complement[singleton[0]]], Id]] // Reverse
FUNCTION[composite[rotate[NATMUL], id[cart[V, complement[singleton[0]]]]]] == True
FUNCTION[composite[rotate[NATMUL], id[cart[V, complement[singleton[0]]]]]] := True

```