associative law for natural multiplication

Johan G. F. Belinfante
2002 August 29

```
<< goedel52.p18; << tools.m

:Package Title: goedel52.p18        2002 August 28 at 8:45 p.m.

It is now:  2002 Aug 29 at 14:39

Loading Simplification Rules

TOOLS.M                    Revised 2002 August 22

weightlimit = 40
```

■ Summary

The associative law for multiplication for natural numbers is derived in this notebook. The main ingredients in the derivation are the distributive law and the uniqueness of iteration.

■ rewriting the distributive law

The first step will be to rewrite the distributive law in terms of left−multiplicationand right−addition:

```
symdif[composite[NATMUL, LEFT[x], NATADD, RIGHT[y]],
    composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[x]]] // VSNormality

union[intersection[composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[x]],
   composite[complement[NATMUL], LEFT[x], NATADD, RIGHT[y]]],
  intersection[composite[NATMUL, LEFT[x], NATADD, RIGHT[y]],
   composite[complement[NATADD], RIGHT[natmul[x, y]], NATMUL, LEFT[x]]]] == 0

union[intersection[composite[NATADD, RIGHT[natmul[x_, y_]], NATMUL, LEFT[x_]],
   composite[complement[NATMUL], LEFT[x_], NATADD, RIGHT[y_]]],
  intersection[composite[NATMUL, LEFT[x_], NATADD, RIGHT[y_]],
   composite[complement[NATADD], RIGHT[natmul[x_, y_]], NATMUL, LEFT[x_]]]] := 0

SubstTest[equal, 0, symdif[u, v],
    {u -> composite[NATMUL, LEFT[x], NATADD, RIGHT[y]],
    v -> composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[x]]}]

True == equal[composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[x]],
  composite[NATMUL, LEFT[x], NATADD, RIGHT[y]]]
```

The following orientation of this equation as a rewrite rule is adopted to facilitate pattern matching.

```
composite[NATMUL, LEFT[x_], NATADD, RIGHT[y_]] :=
 composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[x]]
```

■ using the uniqueness theorem for iteration

An important consequence of the version of the distributive law that was derived in the preceding section is that the functions **composite[NATMUL,LEFT[natmul[x,y]]]** and **composite[NATMUL,LEFT[x],NATMUL,LEFT[y]]** satisfy the same recursion relation:

```
composite[NATMUL, LEFT[natmul[x, y]], SUCC]

composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[natmul[x, y]]]
```

```
composite[NATMUL, LEFT[x], NATMUL, LEFT[y], SUCC]

composite[NATADD, RIGHT[natmul[x, y]], NATMUL, LEFT[x], NATMUL, LEFT[y]]
```

The proof that these two functions are equal requires two applications of the uniqueness of iteration, one for each function.

```
SubstTest[implies, equal[composite[u, w], composite[w, SUCC]],
    equal[composite[w, id[omega]], iterate[u, image[w, singleton[0]]]],
    {u -> composite[NATADD, RIGHT[natmul[x, y]]],
        w -> composite[NATMUL, LEFT[natmul[x, y]]]}]

equal[composite[NATMUL, LEFT[natmul[x, y]]],
  composite[id[intersection[image[V, intersection[omega, singleton[x]]],
     image[V, intersection[omega, singleton[y]]]]],
   iterate[composite[NATADD, RIGHT[natmul[x, y]]], singleton[0]]] == True
```

We turn this around and add it as a temporary rewrite rule:

```
composite[id[intersection[image[V, intersection[omega, singleton[x_]]],
    image[V, intersection[omega, singleton[y_]]]]],
  iterate[composite[NATADD, RIGHT[natmul[x_, y_]]], singleton[0]]] :=
 composite[NATMUL, LEFT[natmul[x, y]]]
```

The second application of uniqueness now gives us the desired result:

```
SubstTest[implies, equal[composite[u, w], composite[w, SUCC]],
    equal[composite[w, id[omega]], iterate[u, image[w, singleton[0]]]],
    {u -> composite[NATADD, RIGHT[natmul[x, y]]],
        w -> composite[NATMUL, LEFT[x], NATMUL, LEFT[y]]}]

equal[composite[NATMUL, LEFT[natmul[x, y]]],
  composite[NATMUL, LEFT[x], NATMUL, LEFT[y]]] == True
```

The orientation of this equation as a rewrite rule adopted here is this:

```
composite[NATMUL, LEFT[x_], NATMUL, LEFT[y_]] := composite[NATMUL, LEFT[natmul[x, y]]]
```

This orientation not only facilitates pattern matching, but it also has the advantage that it implies that left−multiplicationby **x** commutes with left−multiplicationby **y**:

```
commute[composite[NATMUL, LEFT[x]], composite[NATMUL, LEFT[y]]]

True
```

■ a simplification rule

The following simplification rule will be needed below.

```
equal[union[complement[image[V, intersection[omega, singleton[x]]]],
  natmul[z, natmul[x, y]]], natmul[z, natmul[x, y]]]
```

```
True
```

```
union[complement[image[V, intersection[omega, singleton[x_]]]],
  natmul[z_, natmul[x_, y_]]] := natmul[z, natmul[x, y]]
```

■ setting attributes of natmul

With the above simplifcation rule in place, we obtain:

```
Map[A, ImageComp[composite[NATMUL, LEFT[x]], composite[NATMUL, LEFT[y]], singleton[z]]]
```

```
natmul[z, natmul[x, y]] == natmul[x, natmul[y, z]]
```

This equation cannot be made into a rewrite rule, but can be added as a fact:

```
Map[equal[natmul[natmul[x, y], z], #] &, %] // Reverse
```

```
equal[natmul[x, natmul[y, z]], natmul[z, natmul[x, y]]] == True
```

```
equal[natmul[x_, natmul[y_, z_]], natmul[z_, natmul[x_, y_]]] := True
```

Since the commutative law has already been proved, this is equivalent to the associative law:

```
equal[natmul[natmul[x, y], z], natmul[x, natmul[y, z]]]
```

```
True
```

Mathematica has a special mechanism for dealing with associative binary operations which permits the removal of parentheses, namely to set the **Flat** attribute:

```
SetAttributes[natmul, Flat]
```

This has the effect of removing some brackets:

```
{natmul[natmul[x, y], z], natmul[x, natmul[y, z]]}
```

```
{natmul[x, y, z], natmul[x, y, z]}
```

■ Another variant of the associative law

The following preliminary is needed to derive a variable−freeformulation of the associative law.

```
ImageComp[NATMUL, id[cart[V, V]], singleton[x]]
```

```
image[NATMUL, singleton[x]] == singleton[natmul[first[x], second[x]]]
```

```
image[NATMUL, singleton[x_]] := singleton[natmul[first[x], second[x]]]
```

The rest of the derivation is a standard **symdif** argument:

```
symdif[composite[NATMUL, cross[Id, NATMUL], ASSOC],
    composite[NATMUL, cross[NATMUL, Id]]] // VSNormality
```

```
union[intersection[composite[NATMUL, cross[NATMUL, Id]],
   composite[complement[NATMUL], cross[Id, NATMUL], ASSOC]],
  intersection[composite[complement[NATMUL], cross[NATMUL, Id]],
   composite[NATMUL, cross[Id, NATMUL], ASSOC]]] == 0
```

```
union[intersection[composite[NATMUL, cross[NATMUL, Id]],
   composite[complement[NATMUL], cross[Id, NATMUL], ASSOC]],
  intersection[composite[complement[NATMUL], cross[NATMUL, Id]],
   composite[NATMUL, cross[Id, NATMUL], ASSOC]]] := 0
```

```
SubstTest[equal, 0, symdif[u, v],
    {u -> composite[NATMUL, cross[Id, NATMUL], ASSOC],
     v -> composite[NATMUL, cross[NATMUL, Id]]}]
```

```
True == equal[composite[NATMUL, cross[NATMUL, Id]],
  composite[NATMUL, cross[Id, NATMUL], ASSOC]]
```

This is the variable−freeformulation of the associative law:

```
composite[NATMUL, cross[Id, NATMUL], ASSOC] := composite[NATMUL, cross[NATMUL, Id]]
```