divisors of 0

Johan G. F. Belinfante
2002 September 3

```
<< goedel52.p32; << tools.m
```

```
:Package Title: goedel52.p32        2002 September 3 at 3:55 a.m.
```

```
It is now:  2002 Sep 3 at 13:38
```

```
Loading Simplification Rules
```

```
TOOLS.M                   Revised 2002 August 30
```

```
weightlimit = 40
```

■ summary

The goal in this notebook is to derive the elementary fact that every natural number divides **0**, and the related fact that any factorization of **0** must be have **0** as one of its factors.

■ divisors of 0

Start with the obvious:

```
SubstTest[subclass, image[inverse[z], x], domain[z], z -> DIV]
```

```
subclass[image[inverse[DIV], x], omega] == True
```

```
subclass[image[inverse[DIV], x_], omega] := True
```

The reverse inclusion is almost as easy:

```
SubstTest[subclass, composite[NATMUL, LEFT[x]], DIV, x -> 0]
```

```
subclass[cart[omega, singleton[0]], DIV] == True
```

```
subclass[cart[omega, singleton[0]], DIV] := True
```

```
SubstTest[implies, subclass[u, v],
 subclass[image[inverse[u], w], image[inverse[v], w]],
    {u -> cart[omega, singleton[0]], v -> DIV, w -> singleton[0]}]
```

```
subclass[omega, image[inverse[DIV], singleton[0]]] == True
```

```
subclass[omega, image[inverse[DIV], singleton[0]]] := True
```

We finally use the fact that two classes are equal if each is contained in the other:

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> omega, v -> image[inverse[DIV], singleton[0]]}]
```

```
True == equal[omega, image[inverse[DIV], singleton[0]]]
```

```
image[inverse[DIV], singleton[0]] := omega
```

■ factoring 0

To prove the result about factorizations of **0** just one new (temporary) result is needed:

```
SubstTest[implies, and[equal[u, v], member[v, V]], member[u, V],
    {u -> natmul[x, y], v -> 0}]
```

```
or[and[member[x, omega], member[y, omega]], not[equal[0, natmul[x, y]]]] == True
```

```
or[and[member[x_, omega], member[y_, omega]], not[equal[0, natmul[x_, y_]]]] := True
```

We now draw on previously proven facts, reasoning as follows:

```
Map[not, SubstTest[and, implies[p1, or[p2, p3]], implies[p3, p4],
    implies[and[p4, p5], p6], implies[p6, p7], implies[p5, and[p1, p8]],
        not[implies[p5, or[p2, p7]]],
    {p1 -> member[x, omega],
        p2 -> equal[0, x],
        p3 -> member[0, x],
        p4 -> subclass[y, natmul[x, y]],
        p5 -> equal[0, natmul[x, y]],
        p6 -> subclass[y, 0],
        p7 -> equal[0, y],
        p8 -> member[y, omega]}]]
```

```
or[equal[0, x], equal[0, y], not[equal[0, natmul[x, y]]]] == True
```

```
or[equal[0, x_], equal[0, y_], not[equal[0, natmul[x_, y_]]]] := True
```

The converse is not valid unless one adds hypotheses that **x** and **y** are natural numbers; **natmul[0,x]** is **0** when **x** is a natural number, but it is equal to **V** otherwise.

■ facts about inverse images of NATMUL

The theorem about factoring zero has a clean restatement without variables in terms of inverse images of **NATMUL**. We begin studying such inverse images with some easy facts:

```
image[image[inverse[NATMUL], singleton[0]], singleton[0]] // Normality
```

```
image[image[inverse[NATMUL], singleton[0]], singleton[0]] == omega
```

```
image[image[inverse[NATMUL], singleton[0]], singleton[0]] := omega
```

The commutativity of multiplication yields:

```
IminComp[NATMUL, SWAP, x] // Reverse
```

```
inverse[image[inverse[NATMUL], x]] == image[inverse[NATMUL], x]
```

```
inverse[image[inverse[NATMUL], x_]] := image[inverse[NATMUL], x]
```

■ the hard part: skirting around the membership rule for NATMUL

The membership rule for **NATMUL** causes some temporary difficulties for us because it introduces iterated iterates. One possibility might be to remove the definition temporarily. The approach below is not to remove the definition, but instead to use **SubstTest** to allow it to restate some results.

```
SubstTest[member, x, image[image[inverse[z], singleton[0]], singleton[y]],
z -> NATMUL] // Reverse
```

```
and[member[x, V], member[y, omega], member[pair[x, 0],
   iterate[iterate[SUCC, singleton[y]], singleton[0]]]] == equal[0, natmul[x, y]]
```

```
and[member[x_, V], member[y_, omega], member[pair[x_, 0],
   iterate[iterate[SUCC, singleton[y_]], singleton[0]]]] := equal[0, natmul[x, y]]
```

Here is another such restatement:

```
SubstTest[implies, member[x, image[image[inverse[z], singleton[0]], singleton[y]]],
    or[equal[0, x], equal[0, y]], z -> NATMUL] // Reverse
```

```
or[equal[0, x], equal[0, y], not[member[x, V]], not[member[y, omega]],
  not[member[pair[x, 0], iterate[iterate[SUCC, singleton[y]], singleton[0]]]]] == True
```

```
or[equal[0, x_], equal[0, y_], not[member[x_, V]], not[member[y_, omega]],
  not[member[pair[x_, 0], iterate[iterate[SUCC, singleton[y_]], singleton[0]]]]] := True
```

With these two rather ugly steps out of the way, the coast is clear for the rest of the derivation, which mainly amounts to this:

```
Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class, pair[x, y],
  implies[member[x, image[image[inverse[z], singleton[0]], singleton[y]]],
    or[equal[0, x], equal[0, y]]], z -> NATMUL]]
```

```
True == subclass[
  image[image[inverse[NATMUL], singleton[0]], complement[singleton[0]]], singleton[0]]
```

```
subclass[image[image[inverse[NATMUL], singleton[0]], complement[singleton[0]]],
  singleton[0]] := True
```

The final step uses the fact that two classes are equal if each is contained in the other:

```
SubstTest[equal, 0, symdif[u, v],
    {u -> union[cart[omega, singleton[0]], cart[singleton[0], omega]],
  v -> image[inverse[NATMUL], singleton[0]]}]
```

```
True == equal[image[inverse[NATMUL], singleton[0]],
  union[cart[omega, singleton[0]], cart[singleton[0], omega]]]
```

```
image[inverse[NATMUL], singleton[0]] :=
 union[cart[omega, singleton[0]], cart[singleton[0], omega]]
```

Note that this relational form of the theorem about factoring zero is an equation. It goes beyond the statement involving variables that was obtained earlier, adding the converse part. The needed assumption that the variables must refer to natural numbers is reflected in the two explicit occurrences of **omega** in this formula.