the set Z of integers

Johan G. F. Belinfante
2002 November 15

```
<< goedel52.q28; << tools.m
```

```
:Package Title: goedel52.q28          2002 November 14 at 4:55 p.m.

It is now:  2002 Nov 15 at 15:6

Loading Simplification Rules

TOOLS.M                      Revised 2002 November 15

weightlimit = 40
```

■ summary

Integers are equivalence classes of the equivalence relation **EQUIDIFF**. Some basic facts about the set **Z** of integers are derived in this notebook.

■ first rules

The class **Z** of integers.

```
Z // Normality // Reverse
```

```
image[VERTSECT[EQUIDIFF], cart[omega, omega]] == Z
```

```
image[VERTSECT[EQUIDIFF], cart[omega, omega]] := Z
```

The negative of an integer is an integer.

```
Map[image[#, cart[omega, omega]] &,
 SubstTest[VERTSECT, composite[x, SWAP], x -> EQUIDIFF]]
```

```
image[IMAGE[SWAP], Z] == Z
```

```
image[IMAGE[SWAP], Z] := Z
```

The empty set is not an integer. This fact will be needed below.

```
member[0, Z] // AssertTest
```

```
member[0, Z] == False
```

Here is another way to show this:

```
SubstTest[member, 0, image[VERTSECT[x], domain[x]], x -> EQUIDIFF]
```

```
member[0, Z] == False
```

```
member[0, Z] := False
```

■ sethood rule

```
ImageComp[inverse[E], VERTSECT[EQUIDIFF], cart[omega, omega]] // Reverse
```

```
U[Z] == cart[omega, omega]
```

```
U[Z] := cart[omega, omega]
```

As a corollary, it follows that the class of integers is a set.

```
SubstTest[member, U[x], V, x → Z] // Reverse
```

```
member[Z, V] == True
```

```
member[Z, V] := True
```

■ integers

```
member[composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], V] // AssertTest
```

```
member[composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], V] == True
```

```
member[composite[inverse[RIGHT[x_]], inverse[NATADD], NATADD, RIGHT[y_]], V] := True
```

The following is a temporary rule; the converse will be proved later in this notebook.

```
SubstTest[implies, member[u, v], subclass[image[w, singleton[u]], image[w, v]],
    {u -> PAIR[x, y], v -> cart[omega, omega], w -> VERTSECT[EQUIDIFF]}]
```

```
or[member[composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], Z],
  not[member[x, omega]], not[member[y, omega]]] == True
```

```
or[member[composite[inverse[RIGHT[x_]], inverse[NATADD], NATADD, RIGHT[y_]], Z],
  not[member[x_, omega]], not[member[y_, omega]]] := True
```

■ some reasoning

The following is a consequence of the fact that the empty set is not an integer:

```
SubstTest[implies, and[equal[0, z], member[z, y]], member[0, y], y -> Z]
```

```
or[not[equal[0, z]], not[member[z, Z]]] == True
```

```
or[not[equal[0, z_]], not[member[z_, Z]]] := True
```

Some more temporary lemmas are needed. Here is one:

```
SubstTest[implies, equal[u, z], equal[composite[u, v], composite[z, v]], z -> 0]

or[equal[0, intersection[domain[u], range[v]]], not[equal[0, u]]] == True

or[equal[0, intersection[domain[u_], range[v_]]], not[equal[0, u_]]] := True
```

Here is another:

```
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
        {p1 -> and[member[x, omega], member[y, omega]],
            p2 -> member[union[x, y], omega],
            p3 -> member[union[x, y], image[inverse[S], omega]]}]]

or[member[union[x, y], image[inverse[S], omega]],
  not[member[x, omega]], not[member[y, omega]]] == True

or[member[union[x_, y_], image[inverse[S], omega]],
  not[member[x_, omega]], not[member[y_, omega]]] := True
```

This fact ...

```
equiv[or[not[member[x, omega]], not[member[y, omega]],
  not[member[union[x, y], image[inverse[S], omega]]]],
    or[not[member[x, omega]], not[member[y, omega]]]]

True
```

... justifies the following temporary simplification rule:

```
or[not[member[x_, omega]], not[member[y_, omega]],
  not[member[union[x_, y_], image[inverse[S], omega]]]] :=
 or[not[member[x, omega]], not[member[y, omega]]]
```

The following technical lemma is also needed:

```
SubstTest[equal, composite[u, v], 0,
    {u -> composite[inverse[RIGHT[x]], inverse[NATADD]],
  v -> composite[NATADD, RIGHT[y]]}]

or[not[member[x, V]], subclass[V,
   union[intersection[complement[image[V, intersection[omega, singleton[y]]]],
     image[V, singleton[x]]], intersection[
     complement[image[image[inverse[NATADD], image[S, singleton[y]]], singleton[x]]],
     image[V, singleton[x]]]]]] ==
 or[not[member[x, omega]], not[member[y, omega]]]

or[not[member[x_, V]], subclass[V,
   union[intersection[complement[image[V, intersection[omega, singleton[y_]]]],
     image[V, singleton[x_]]], intersection[
     complement[image[image[inverse[NATADD], image[S, singleton[y_]]], singleton[x_]]],
     image[V, singleton[x_]]]]]] :=
 or[not[member[x, omega]], not[member[y, omega]]]
```

The converse part of the theorem in the preceding section now follows:

```
SubstTest[implies, member[z, Z], not[equal[0, z]],
    z -> composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]]]

or[and[member[x, omega], member[y, omega]], not[
   member[composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], Z]]] == True
```

```
or[and[member[x_, omega], member[y_, omega]], not[
   member[composite[inverse[RIGHT[x_]], inverse[NATADD], NATADD, RIGHT[y_]], Z]]] := True
```

The theorem and its converse can be combined to obtain a permanent rewrite rule:

```
equiv[and[member[x, omega], member[y, omega]],
 member[composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], Z]]
```

```
True
```

```
member[composite[inverse[RIGHT[x_]], inverse[NATADD], NATADD, RIGHT[y_]], Z] :=
 and[member[x, omega], member[y, omega]]
```

■ some particular integers

This is the integer zero:

```
SubstTest[member, composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], Z,
    {x -> 0, y -> 0}]
```

```
member[id[omega], Z] == True
```

```
member[id[omega], Z] := True
```

This is the integer one:

```
SubstTest[member, composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], Z,
    {x -> 0, y -> singleton[0]}]
```

```
member[composite[id[omega], SUCC], Z] == True
```

```
member[composite[id[omega], SUCC], Z] := True
```

■ positive and negative integers in general

```
SubstTest[member, composite[inverse[RIGHT[y]], inverse[NATADD], NATADD, RIGHT[x]], Z,
    y -> 0]
```

```
member[composite[NATADD, RIGHT[x]], Z] == member[x, omega]
```

```
member[composite[NATADD, RIGHT[x_]], Z] := member[x, omega]
```

```
SubstTest[member, composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], Z,
    y -> 0]
```

```
member[composite[inverse[RIGHT[x]], inverse[NATADD]], Z] == member[x, omega]
```

```
member[composite[inverse[RIGHT[x_]], inverse[NATADD]], Z] := member[x, omega]
```