positive and negative integers

Johan G. F. Belinfante
2002 November 30

```
<< goedel52.q56; << tools.m
```

```
:Package Title: goedel52.q56          2002 November 30 at 6:07 p.m

It is now:  2002 Nov 30 at 19:47

Loading Simplification Rules

TOOLS.M                       Revised 2002 November 28

weightlimit = 40
```

■ summary

It is shown in this notebook that the set **Z** of all integers is the union of the set **range[PLUS]** of non−negativeintegers and the set of non−positiveintegers. By definition, the non−negativeintegers are the functions of the form

```
plus[x_] := composite[NATADD, RIGHT[x]]
```

and the non−positiveintegers are the inverses of these functions. The set of integers is defined as the set of all composites of non−positiveand non−negativeintegers.

```
image[EQUIDIFF, singleton[PAIR[x, y]]] == composite[inverse[plus[x]], plus[y]]
```

```
True
```

■ the positive integers

In this section it is shown that **range[PLUS]** is the set of non−negativeintegers. This is scarcely surprising since **PLUS** is the function which takes each natural number to the corresponding non−negativeinteger.

```
member[x, PLUS]
```

```
and[equal[composite[NATADD, RIGHT[first[x]]], second[x]], member[first[x], omega]]
```

The idea is to bring **range[PLUS]** into play by using

```
composite[PLUS, inverse[PLUS]]
```

```
id[range[PLUS]]
```

The starting point is:

```
ImageComp[inverse[PLUS], PLUS, singleton[x]]
```

```
intersection[omega, singleton[x]] ==
 intersection[image[V, intersection[omega, singleton[x]]],
  image[inverse[PLUS], singleton[composite[NATADD, RIGHT[x]]]]]
```

From this one derives an equation full of conditions:

```
Map[image[PLUS, #] &, %]
```

```
intersection[image[V, intersection[omega, singleton[x]]],
  singleton[composite[NATADD, RIGHT[x]]]] ==
 intersection[image[V, intersection[omega, singleton[x]]],
  range[PLUS], singleton[composite[NATADD, RIGHT[x]]]]
```

The conditions are made manifest as follows:

```
Map[equal[First[%], #] &, %]
```

```
True == or[member[composite[NATADD, RIGHT[x]], range[PLUS]], not[member[x, omega]]]
```

```
or[member[composite[NATADD, RIGHT[x_]], range[PLUS]], not[member[x_, omega]]] := True
```

To obtain the reverse implication, one needs this fact:

```
member[0, range[PLUS]] // AssertTest
```

```
member[0, range[PLUS]] == False
```

```
member[0, range[PLUS]] := False
```

From this one derives:

```
SubstTest[implies, and[equal[u, v], member[v, w]], member[u, w],
    {u -> 0, v -> plus[x], w -> range[PLUS]}]
```

```
or[member[x, omega], not[member[composite[NATADD, RIGHT[x]], range[PLUS]]]] == True
```

```
or[member[x_, omega], not[member[composite[NATADD, RIGHT[x_]], range[PLUS]]]] := True
```

It is now a matter of puting the two implications together:

```
equiv[member[plus[x], range[PLUS]], member[x, omega]]
```

```
True
```

This fact is added as an unconditional rewrite rule:

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

■ the key equation

The **Assoc** test provides the key equation needed to derive the description of the set of integers as the union of positives and negatives.

```
key = Assoc[inverse[plus[x]], plus[x], plus[natsub[y, x]]]
```

```
composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y],
  id[intersection[complement[image[V, intersection[x, complement[y]]]],
    image[V, intersection[omega, singleton[x]]],
    image[V, intersection[omega, singleton[y]]]]]] ==
 composite[id[image[V, intersection[omega, singleton[x]]]], NATADD, RIGHT[natsub[y, x]]]
```

The key equation is used to derive two corollaries. This is the first:

```
Map[subclass[#, composite[inverse[plus[x]], plus[y]]] &, key] // Reverse
```

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

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

Here is the second corollary:

```
Map[subclass[#, plus[natsub[y, x]]] &, key]
```

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

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

The next step is to replace the inclusions with an equation:

```
SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
    {p -> and[member[x, omega], subclass[x, y]],
        u -> composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]],
        v -> composite[NATADD, RIGHT[natsub[y, x]]]}] // Reverse
```

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

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

■ some reasoning

The following will be used to show that the composite of a neagative integer with a greater positive integer is a positive integer.

```
SubstTest[implies, and[equal[u, v], member[v, w]], member[u, w],
    {u -> composite[inverse[plus[x]], plus[y]],
  v -> plus[natsub[y, x]], w -> range[PLUS]}]
```

```
or[member[composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]], range[PLUS]],
  not[equal[composite[NATADD, RIGHT[natsub[y, x]]],
    composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]]]],
  not[member[x, omega]], not[member[y, omega]], not[subclass[x, y]]] == True
```

```
or[member[composite[inverse[RIGHT[x_]], inverse[NATADD], NATADD, RIGHT[y_]],
   range[PLUS]], not[equal[composite[NATADD, RIGHT[natsub[y_, x_]]],
    composite[inverse[RIGHT[x_]], inverse[NATADD], NATADD, RIGHT[y_]]]],
  not[member[x_, omega]], not[member[y_, omega]], not[subclass[x_, y_]]] := True
```

The above result can be cleaned up with a little reasoning:

```
Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
    {p1 -> and[member[x, omega], member[y, omega], subclass[x, y]],
       p2 -> equal[composite[NATADD, RIGHT[natsub[y, x]]],
     composite[inverse[RIGHT[x]], inverse[NATADD], NATADD, RIGHT[y]]],
       p3 -> member[composite[inverse[RIGHT[x]], inverse[NATADD],
      NATADD, RIGHT[y]], range[PLUS]]}]]
```

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

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

The next step is to eliminate all the variables in the above statement to obtain an equation:

```
SubstTest[class, pair[x, y],
 implies[member[pair[x, y], u], member[image[v, cart[singleton[x], singleton[y]]], w]],
    {u -> composite[id[omega], S, id[omega]], v -> EQUIDIFF, w -> range[PLUS]}]
```

```
cart[V, V] == union[cart[V, complement[omega]], cart[complement[omega], V],
  composite[Id, image[inverse[VERTSECT[EQUIDIFF]], range[PLUS]]],
  composite[inverse[E], id[omega]]]
```

This looks better when one takes the complement.

```
Map[equal[0, composite[Id, complement[#]]] &, %]
```

```
True == subclass[cart[omega, omega],
  union[E, inverse[image[inverse[VERTSECT[EQUIDIFF]], range[PLUS]]]]]
```

```
subclass[cart[omega, omega],
  union[E, inverse[image[inverse[VERTSECT[EQUIDIFF]], range[PLUS]]]]] := True
```

The occurrence of the membership relation **E** is eliminated by using dichotomy. This is accomplished by simply transposing **E** to get it negated:

```
SubstTest[subclass, intersection[u, complement[w]], v,
    {u -> cart[omega, omega],
  v -> inverse[image[inverse[VERTSECT[EQUIDIFF]], range[PLUS]]], w -> E}]
```

```
subclass[image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]], range[PLUS]] ==
 True
```

```
subclass[
  image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]], range[PLUS]] := True
```

■ positive integers

In this section the inclusion found in the preceding section is sharpened to an equation. To speed up the normality tests, the **simplify** flag is turned off.

```
simplify = False;
```

The idea is that any positive integer can be written as the composite of zero (= **id[omega]**) with itself.

```
composite[EQUIDIFF, id[cart[singleton[0], omega]]] // VSTriNormality
```

```
composite[EQUIDIFF, id[cart[singleton[0], omega]]] ==
 composite[SWAP, inverse[rotate[NATADD]], inverse[LEFT[0]]]
```

```
composite[EQUIDIFF, id[cart[singleton[0], omega]]] :=
 composite[SWAP, inverse[rotate[NATADD]], inverse[LEFT[0]]]
```

This yields a formula for **range[PLUS]** as an image of **VERTSECT[EQUIDIFF]**.

```
Map[image[#, cart[singleton[0], omega]] &, SubstTest[VERTSECT,
   composite[x, id[cart[singleton[0], omega]]], x -> EQUIDIFF]] // Reverse
```

```
image[VERTSECT[EQUIDIFF], cart[singleton[0], omega]] == range[PLUS]
```

```
image[VERTSECT[EQUIDIFF], cart[singleton[0], omega]] := range[PLUS]
```

An immediate corollary is the reverse of the inclusion found in the preceding section.

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
    {u -> cart[singleton[0], omega],
  v -> composite[id[omega], S, id[omega]], w -> VERTSECT[EQUIDIFF]}]
```

```
subclass[range[PLUS],
  image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]]] == True
```

```
subclass[range[PLUS],
  image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]]] := True
```

Since the inclusion goes in both directions, one obtains an equation:

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> range[PLUS],
  v -> image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]]}]
```

```
True == equal[image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]], range[PLUS]]
```

```
image[VERTSECT[EQUIDIFF], composite[id[omega], S, id[omega]]] := range[PLUS]
```

■ negatives

The negative integers are the inverses of the positive ones. Since **SWAP** commutes with **EQUIDIFF**, the same holds when wrapped with **IMAGE**.

```
Map[VERTSECT, Assoc[EQUIDIFF, SWAP, inverse[E]]]
```

```
composite[IMAGE[EQUIDIFF], IMAGE[SWAP]] == composite[IMAGE[SWAP], IMAGE[EQUIDIFF]]
```

Taking the composite with **SINGLETON** yields a corresponding result involving **VERTSECT[EQUIDIFF]**:

```
Map[composite[#, SINGLETON] &, %]
```

```
union[cart[complement[cart[V, V]], singleton[0]],
  composite[VERTSECT[EQUIDIFF], SWAP]] == composite[IMAGE[SWAP], VERTSECT[EQUIDIFF]]
```

From this one obtains a formula for the set of non−positiveintegers analogous to the formula obtained for the non−negative integers in the preceding section.

```
Map[image[#, composite[id[omega], S, id[omega]]] &, %]
```

```
image[VERTSECT[EQUIDIFF], composite[id[omega], inverse[S], id[omega]]] ==
 image[INVERSE, range[PLUS]]
```

```
image[VERTSECT[EQUIDIFF], composite[id[omega], inverse[S], id[omega]]] :=
 image[INVERSE, range[PLUS]]
```

Taking the union of these two formulas yields the final result:

```
SubstTest[image, w, union[x, y],
    {w -> VERTSECT[EQUIDIFF], x -> composite[id[omega], S, id[omega]],
  y -> composite[id[omega], inverse[S], id[omega]]}]
```

```
Z == union[image[INVERSE, range[PLUS]], range[PLUS]]
```

The union of the set of non−positiveintegers and the set of non−negativeintegers is the set of all integers.

```
union[image[INVERSE, range[PLUS]], range[PLUS]] := Z
```