image[COMPOSE,cart[Z,Z]]

Johan G. F. Belinfante
2003 July 25

```
In[1]:= << goedel52.s61; << tools.m

        :Package Title: goedel52.s61       2003 July 23 at 8:55 p.m.

        It is now:  2003 Aug 15 at 10:0

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 July 25

        weightlimit = 40
```

summary

In this notebook, one should think of an integer as a function whose graph is a line with slope one in the natural number plane **cart[omega,omega]**. Composites of positive integers are integers, composites of negative integers are integers, and the composite of a negative integer with a positive integer is an integer. But the composite of a positive integer with a negative integer could be just a subclass of an integer, a partial line if you will. It is shown that the composite of integers in all cases is nonempty.

reduction of the general case to the case of partial lines

Lemma:

```
In[2]:= equiv[or[not[member[u, x]], not[member[v, y]], not[member[composite[u, v], V]]],
         or[not[member[u, x]], not[member[v, y]]]]

Out[2]= True

In[3]:= or[not[member[u_, x_]], not[member[v_, y_]], not[member[composite[u_, v_], V]]] :=
         or[not[member[u, x]], not[member[v, y]]]
```

This lemma is needed to relate **composite** to **COMPOSE**.

```
In[4]:= SubstTest[implies, subclass[s, t], subclass[image[z, s], image[z, t]],
            {s -> cart[singleton[u], singleton[v]], t -> cart[x, y], z -> COMPOSE}]

Out[4]= or[member[composite[u, v], image[COMPOSE, cart[x, y]]],
          not[member[u, x]], not[member[v, y]]] == True

In[5]:= or[member[composite[u_, v_], image[COMPOSE, cart[x_, y_]]],
          not[member[u_, x_]], not[member[v_, y_]]] := True
```

The above lemma is applied to the case of the composite of a positive integer **plus[x]** with **zero = id[omega]**.

```
In[6]:= SubstTest[implies, and[member[u, w], member[v, z]],
         member[composite[u, v], image[COMPOSE, cart[w, z]]],
           {u -> plus[x], v -> id[omega], w -> range[PLUS], z -> image[INVERSE, range[PLUS]]}]

Out[6]= or[member[composite[NATADD, RIGHT[x]], image[COMPOSE,
           cart[range[PLUS], image[INVERSE, range[PLUS]]]]], not[member[x, omega]]] == True

In[7]:= or[member[composite[NATADD, RIGHT[x_]], image[COMPOSE,
           cart[range[PLUS], image[INVERSE, range[PLUS]]]]], not[member[x_, omega]]] := True
```

The variable **x** is eliminated:

```
In[8]:= Map[equal[V, #] &, union[complement[omega], image[inverse[PLUS],
           image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]]] // Normality]

Out[8]= subclass[omega, image[inverse[PLUS],
          image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]]] == True

In[9]:= subclass[omega, image[inverse[PLUS],
          image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]]] := True
```

The inverse images are eliminated from this statement:

```
In[10]:= Map[equal[#, range[PLUS]] &, ImageComp[PLUS, inverse[PLUS],
           image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]]]

Out[10]= subclass[range[PLUS],
           image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] == True

In[11]:= subclass[range[PLUS],
           image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] := True
```

Lemma.

```
In[12]:= ImageComp[INVERSE, COMPOSE, cart[x, y]] // Reverse

Out[12]= image[INVERSE, image[COMPOSE, cart[x, y]]] ==
          image[COMPOSE, cart[image[IMAGE[SWAP], y], image[IMAGE[SWAP], x]]]

In[13]:= image[INVERSE, image[COMPOSE, cart[x_, y_]]] :=
          image[COMPOSE, cart[image[IMAGE[SWAP], y], image[IMAGE[SWAP], x]]]
```

The case of negative integers follows from the lemma and the result for positive integers. Negative integers are just the inverses of positive integers.

```
In[14]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
            {u -> range[PLUS],
           v -> image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]], w -> INVERSE}]

Out[14]= subclass[image[INVERSE, range[PLUS]],
           image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] == True

In[15]:= subclass[image[INVERSE, range[PLUS]],
           image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] := True
```

Since the set of integers is the union of the set of positive integers and negative integers, the above results can be combined into a single statement: this amounts to the statement that complete lines are a special case of partial lines.

```
In[16]:= SubstTest[subclass, union[u, v], w,
           {u -> range[PLUS], v -> image[INVERSE, range[PLUS]],
          w -> image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]}]

Out[16]= subclass[Z, image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] == True

In[17]:= subclass[Z, image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] := True
```

Corollary 1.

```
In[18]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u -> cart[range[PLUS], image[INVERSE, range[PLUS]]], v -> cart[Z, Z], w -> COMPOSE}]

Out[18]= subclass[image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]],
          image[COMPOSE, cart[Z, Z]]] == True

In[19]:= subclass[image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]],
          image[COMPOSE, cart[Z, Z]]] := True

In[20]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> Z, v -> image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]],
          w -> image[COMPOSE, cart[Z, Z]]}]

Out[20]= subclass[Z, image[COMPOSE, cart[Z, Z]]] == True

In[21]:= subclass[Z, image[COMPOSE, cart[Z, Z]]] := True
```

Corollary 2.

```
In[22]:= equal[union[Z, image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]],
          image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]]

Out[22]= True

In[23]:= union[Z, image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]] :=
          image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]]
```

The various other cases of composites of integers are integers, and one can combine all the possible cases:

```
In[24]:= ImageComp[INVERSE, COMPOSE, cart[range[PLUS], range[PLUS]]]

Out[24]= image[COMPOSE, cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]] ==
          image[INVERSE, range[PLUS]]

In[25]:= image[COMPOSE, cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]] :=
          image[INVERSE, range[PLUS]]

In[26]:= SubstTest[image, COMPOSE, union[u, v, x, y],
           {u -> cart[range[PLUS], range[PLUS]],
          v -> cart[range[PLUS], image[INVERSE, range[PLUS]]],
               x -> cart[image[INVERSE, range[PLUS]], range[PLUS]],
          y -> cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]}] // Reverse

Out[26]= image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]] ==
          image[COMPOSE, cart[Z, Z]]
```

This result allows one to reduce the general case of composites of integers to the one special problematic case where the result need not be a complete integer.

```
image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]] ==
 image[COMPOSE, cart[Z, Z]]
```

```
In[27]:= image[COMPOSE, cart[range[PLUS], image[INVERSE, range[PLUS]]]] :=
          image[COMPOSE, cart[Z, Z]]
```

the main argument

In this section it is shown that the composite of integers can not be empty.

```
In[28]:= intersection[omega, image[inverse[PLUS], P[cart[complement[omega], V]]]] // Normality

Out[28]= intersection[omega, image[inverse[PLUS], P[cart[complement[omega], V]]]] == 0

In[29]:= intersection[omega, image[inverse[PLUS], P[cart[complement[omega], V]]]] := 0

In[30]:= Map[range, SubstTest[intersection, cart[omega, omega], composite[inverse[PLUS],
             INVERSE, image[inverse[COMPOSE], z], PLUS], z -> singleton[0]]] // Reverse

Out[30]= image[inverse[PLUS], P[cart[complement[omega], V]]] == 0

In[31]:= image[inverse[PLUS], P[cart[complement[omega], V]]] := 0

In[32]:= ImageComp[PLUS, inverse[PLUS], P[cart[complement[omega], V]]]

Out[32]= intersection[P[cart[complement[omega], V]], range[PLUS]] == 0

In[33]:= intersection[P[cart[complement[omega], V]], range[PLUS]] := 0
```

The main result:

```
In[34]:= Map[not[equal[0, #]] &,
          ImageComp[composite[COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]],
           composite[cross[inverse[PLUS], composite[inverse[PLUS], INVERSE]],
            inverse[COMPOSE]], singleton[0]]]

Out[34]= member[0, image[COMPOSE, cart[Z, Z]]] == False

In[35]:= member[0, image[COMPOSE, cart[Z, Z]]] := False
```