card[cart[omega,omega]]

Johan G. F. Belinfante
2006 August 27

```
In[1]:= SetDirectory["l:"]; << goedel84.24a; << tools.m

        :Package Title: goedel84.24a      2006 August 24 at 4:45 p.m.

        It is now:  2006 Aug 27 at 22:2

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 August 22

        weightlimit = 40
```

summary

The set **omega** of natural numbers has the same cardinality as its cartesian square. This can be shown by embedding **cart[omega,omega]** into **omega** in a one-to-one fashion. In this notebook it is shown that the function **f(x, y) = $(x + y)^2$ + x** does the job. A temporary abbreviation is introduced for a variable-free formula for this function:

```
In[2]:= F := composite[NATADD, cross[FIRST, composite[NATMUL, DUP, NATADD]], DUP]
```

In addition to this temporary abbreviation, it is convenient to introduce two others. These are used for input, but are eliminated from all output.

```
In[3]:= natsq[x_] := natmul[x, x]

In[4]:= f[x_, y_] := natadd[natsq[natadd[x, y]], x]
```

simplification rules

Lemma.

```
In[5]:= ImageComp[id[omega], NATMUL, x] // Reverse
Out[5]= intersection[omega, image[NATMUL, x]] == image[NATMUL, x]

In[6]:= intersection[omega, image[NATMUL, x_]] := image[NATMUL, x]
```

Lemma. The set of all squares of natural numbers is:

```
In[7]:= ImageComp[NATMUL, id[cart[omega, omega]], Id] // Reverse
Out[7]= image[NATMUL, id[omega]] == image[NATMUL, Id]
```

```
In[8]:= image[NATMUL, id[omega]] := image[NATMUL, Id]
```

card[range[F]]

In this section it is shown that the subset **range[F]** of **omega** is countably infinite. The starting point is the observation that **f[x, 0] = natsq[x]**.

Lemma.

```
In[9]:= composite[NATMUL, DUP, NATADD, id[cart[set[0], omega]], inverse[FIRST]] // VSNormality

Out[9]= composite[NATMUL, DUP, NATADD, id[cart[set[0], omega]], inverse[FIRST]] ==
         cart[set[0], image[NATMUL, Id]]

In[10]:= % /. Equal → SetDelayed
```

Theorem.

```
In[11]:= SubstTest[subclass, image[x, y], range[x], {x → F, y → cart[set[0], omega]}]

Out[11]= subclass[image[NATMUL, Id], image[NATADD, composite[NATMUL, DUP, S, id[omega]]]] == True

In[12]:= % /. Equal → SetDelayed
```

It follows that **range[F]** is not finite:

```
In[13]:= Map[not, SubstTest[implies, and[subclass[u, v], member[v, FINITE]], member[u, FINITE],
           {u -> image[NATMUL, Id], v → range[F]}]]

Out[13]= member[image[NATADD, composite[NATMUL, DUP, S, id[omega]]], FINITE] == False

In[14]:= % /. Equal → SetDelayed
```

Corollary. Since any infinite subset of **omega** is countable, one has **card[range[F]] = omega**.

```
In[15]:= SubstTest[implies, and[subclass[x, omega], not[member[x, FINITE]]],
          equal[card[x], omega], x → range[F]]

Out[15]= equal[omega, card[image[NATADD, composite[NATMUL, DUP, S, id[omega]]]]] == True

In[16]:= card[image[NATADD, composite[NATMUL, DUP, S, id[omega]]]] := omega
```

conditional dichotomy rules

For natural numbers, the statement **x ≤ y** is equivalent to the negation of **y < x**. The following conditional rewrite rules serve to eliminate ≤ in favor of <.

```
In[17]:= implies[and[member[x, omega], member[y, omega]],
            equiv[subclass[x, y], not[member[y, x]]]] // not // not

Out[17]= True

In[18]:= subclass[x_, y_] := not[member[y, x]] /; and[member[x, omega], member[y, omega]]

In[19]:= implies[and[member[x, omega], member[y, omega]],
            equiv[or[equal[x, y], member[x, y]], not[member[y, x]]]] // not // not

Out[19]= True

In[20]:= or[equal[x_, y_], member[x_, y_]] :=
          not[member[y, x]] /; and[member[x, omega], member[y, omega]]
```

---

inequalities for f

Cancellation lemma.

```
In[21]:= SubstTest[member, nat[x], natadd[nat[x], nat[w]], w → natadd[nat[x], nat[y], nat[y]]]

Out[21]= member[nat[x], natadd[nat[x], nat[x], nat[y], nat[y]]] ==
          or[not[equal[0, nat[x]]], not[equal[0, nat[y]]]]

In[22]:= ((First[%] /. {u → u_, v → v_, x → x_, y → y_}) == Last[%]) /. Equal → SetDelayed
```

This is a consequence of the lemma.

```
In[23]:= member[f[nat[x], nat[y]], natsq[succ[natadd[nat[x], nat[y]]]]]

Out[23]= True
```

Lemma. (Squaring is strictly monotone.)

```
In[24]:= Map[not, SubstTest[member, natsq[nat[s]], natsq[nat[t]],
            {s → natadd[nat[x], nat[y]], t -> succ[natadd[nat[u], nat[v]]]}]]

Out[24]= member[natadd[nat[u], nat[u], nat[v], nat[v], natmul[nat[u], nat[u]],
             natmul[nat[u], nat[v]], natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
            natadd[natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
             natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]] ==
           member[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]

In[25]:= ((First[%] /. {u → u_, v → v_, x → x_, y → y_}) == Last[%]) /. Equal → SetDelayed
```

Lemma. If **u + v < x + y**, then **f[u, v] < f[x, y]**.

```
In[26]:= SubstTest[implies,
          and[subclass[nat[u], nat[v]], member[nat[v], nat[w]]], member[nat[u], nat[w]],
          {u -> natsq[natadd[nat[x], nat[y]]], v -> f[nat[u], nat[v]],
           w → natsq[succ[natadd[nat[u], nat[v]]]]}]

Out[26]= or[member[natadd[nat[u], natmul[nat[u], nat[u]], natmul[nat[u], nat[v]],
             natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]], natadd[natmul[nat[x], nat[x]],
             natmul[nat[x], nat[y]], natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]],
           not[member[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]] == True

In[27]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[28]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → equal[f[nat[x], nat[y]], f[nat[u], nat[v]]],
            p2 → subclass[natsq[natadd[nat[x], nat[y]]], f[nat[u], nat[v]]],
            p3 → not[member[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]]}]]

Out[28]= or[not[equal[natadd[nat[u], natmul[nat[u], nat[u]],
             natmul[nat[u], nat[v]], natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
            natadd[nat[x], natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
             natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]],
           not[member[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]] == True

In[29]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If **f[u, v] = f[x, y]**, then **u + v = x + y**.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 -> equal[f[nat[x], nat[y]], f[nat[u], nat[v]]],
            p2 -> not[member[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]],
            p3 -> not[member[natadd[nat[x], nat[y]], natadd[nat[u], nat[v]]]],
            p4 → equal[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]}]]

Out[30]= or[equal[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]],
           not[equal[natadd[nat[u], natmul[nat[u], nat[u]],
             natmul[nat[u], nat[v]], natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
            natadd[nat[x], natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
             natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]]] == True

In[31]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

This is a consequence of the one-to-one property of squaring for natural numbers:

```
In[32]:= SubstTest[equal, natsq[nat[s]], natsq[nat[t]],
          {s -> natadd[nat[x], nat[y]], t → natadd[nat[u], nat[v]]}]

Out[32]= equal[natadd[natmul[nat[u], nat[u]], natmul[nat[u], nat[v]],
           natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]], natadd[natmul[nat[x], nat[x]],
           natmul[nat[x], nat[y]], natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]] ==
         equal[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]

In[33]:= ((First[%] /. {u → u_, v → v_, x → x_, y → y_}) == Last[%]) /. Equal → SetDelayed
```

Cancellation lemma:

```
In[34]:= SubstTest[implies, and[equal[nat[s], nat[t]],
           equal[natadd[nat[s], nat[x]], natadd[nat[t], nat[u]]]], equal[nat[x], nat[u]],
          {s → natsq[natadd[nat[x], nat[y]]], t → natsq[natadd[nat[u], nat[v]]]}]

Out[34]= or[equal[nat[u], nat[x]], not[equal[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]]],
           not[equal[natadd[nat[u], natmul[nat[u], nat[u]],
              natmul[nat[u], nat[v]], natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
             natadd[nat[x], natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
              natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]]] == True

In[35]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary. If **f[u,v] = f[x,y]**, then **u = x**.

```
In[36]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           not[implies[p1, p3]], {p1 -> equal[f[nat[x], nat[y]], f[nat[u], nat[v]]], p2 ->
             equal[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]], p3 → equal[nat[x], nat[u]]}]]

Out[36]= or[equal[nat[u], nat[x]],
           not[equal[natadd[nat[u], natmul[nat[u], nat[u]], natmul[nat[u], nat[v]],
              natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
             natadd[nat[x], natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
              natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]]] == True

In[37]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Similarly, **v = y**.

```
In[38]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 -> equal[f[nat[x], nat[y]], f[nat[u], nat[v]]],
            p2 -> equal[natadd[nat[u], nat[v]], natadd[nat[x], nat[y]]],
            p3 → equal[nat[x], nat[u]], p4 → equal[nat[y], nat[v]]}]]

Out[38]= or[equal[nat[v], nat[y]],
           not[equal[natadd[nat[u], natmul[nat[u], nat[u]], natmul[nat[u], nat[v]],
              natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
             natadd[nat[x], natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
              natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]]] == True

In[39]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Restatement.

```
In[40]:= implies[
           equal[image[F, set[PAIR[nat[x], nat[y]]]], image[F, set[PAIR[nat[u], nat[v]]]]],
           and[equal[nat[x], nat[u]], equal[nat[y], nat[v]]] // NotNotTest

Out[40]= or[and[equal[nat[u], nat[x]], equal[nat[v], nat[y]]],
           not[equal[natadd[nat[u], natmul[nat[u], nat[u]],
              natmul[nat[u], nat[v]], natmul[nat[u], nat[v]], natmul[nat[v], nat[v]]],
             natadd[nat[x], natmul[nat[x], nat[x]], natmul[nat[x], nat[y]],
              natmul[nat[x], nat[y]], natmul[nat[y], nat[y]]]]] == True

In[41]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

eliminating the variables

First, all the **nat** wrappers are removed:

```
In[42]:= SubstTest[implies,
          and[equal[u, nat[q]], equal[v, nat[r]], equal[x, nat[s]], equal[y, nat[t]]],
          implies[equal[image[F, set[PAIR[x, y]]], image[F, set[PAIR[u, v]]]],
           and[equal[x, u], equal[y, v]]], {q → u, r → v, s → x, t → y}]

Out[42]= or[and[equal[u, x], equal[v, y]],
           not[equal[set[natadd[u, natmul[u, u], natmul[u, v], natmul[u, v], natmul[v, v]]],
             set[natadd[x, natmul[x, x], natmul[x, y], natmul[x, y], natmul[y, y]]]]],
           not[member[u, omega]], not[member[v, omega]], not[member[x, omega]],
           not[member[y, omega]]] == True

In[43]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

The next step, the removal of all variables, is expedited by clearing the **simplify** flag.

```
In[44]:= simplify = False;

In[45]:= Map[composite[id[cart[V, V]], complement[#], id[cart[V, V]]] &,
           SubstTest[class, pair[pair[u, v], pair[x, y]],
            implies[and[member[u, omega], member[v, omega], member[x, omega], member[y, omega],
              equal[image[funpart[z], set[PAIR[x, y]]], image[funpart[z], set[PAIR[u, v]]]]],
             and[equal[u, x], equal[v, y]], z → F]] // Reverse

Out[45]= composite[id[cart[omega, omega]],
           intersection[Di, composite[intersection[composite[inverse[FIRST], FIRST],
              composite[inverse[NATADD], inverse[DUP], inverse[NATMUL], SECOND]],
             inverse[NATADD], NATADD, intersection[composite[inverse[FIRST], FIRST],
              composite[inverse[SECOND], NATMUL, DUP, NATADD]]]], id[cart[omega, omega]]] == 0

In[46]:= % /. Equal → SetDelayed
```

final steps

At this point, the **simplify** flag can be set again.

```
In[47]:= simplify = True;
```

Lemma.

```
In[48]:= SubstTest[equal, 0, intersection[Di, composite[inverse[z], z]],
          z → composite[x, id[cart[omega, omega]]]]

Out[48]= equal[0, intersection[omega,
            image[fix[composite[Di, id[cart[omega, omega]], inverse[x], x]], omega]]] ==
          FUNCTION[composite[id[cart[omega, omega]], inverse[x]]]

In[49]:= equal[0, intersection[omega,
            image[fix[composite[Di, id[cart[omega, omega]], inverse[x_], x_]], omega]]] :=
          FUNCTION[composite[id[cart[omega, omega]], inverse[x]]]

In[50]:= intersection[cart[cart[omega, omega], cart[omega, omega]],
          Di, composite[inverse[z], z]]

Out[50]= composite[id[cart[omega, omega]],
          intersection[Di, composite[inverse[z], z]], id[cart[omega, omega]]]
```

Theorem. The function **F** is one-to-one.

```
In[51]:= SubstTest[equal, 0, composite[id[cart[omega, omega]],
            intersection[Di, composite[inverse[z], z]], id[cart[omega, omega]]], z → F] // Reverse

Out[51]= FUNCTION[
           composite[intersection[composite[inverse[FIRST], FIRST], composite[inverse[NATADD],
             inverse[DUP], inverse[NATMUL], SECOND]], inverse[NATADD]]] == True

In[52]:= % /. Equal → SetDelayed
```

Lemma. The function **F** is a set.

```
In[53]:= member[F, V] // AssertTest

Out[53]= member[composite[NATADD, intersection[composite[inverse[FIRST], FIRST],
             composite[inverse[SECOND], NATMUL, DUP, NATADD]]], V] == True

In[54]:= % /. Equal → SetDelayed
```

Main theorem. The cartesian square of **omega** is countably infinite.

```
In[55]:= SubstTest[implies, member[x, BIJ], equal[card[domain[x]], card[range[x]]], x → F]

Out[55]= equal[omega, card[cart[omega, omega]]] == True

In[56]:= card[cart[omega, omega]] := omega
```