the double counting rule

Johan G. F. Belinfante
2005 July 16

```
In[1]:= SetDirectory["i:"]; << goedel71.13a; << tools.m

        :Package Title: goedel71.13a          2005 July 13 at 2:45 a.m.

        It is now:  2005 Jul 16 at 14:51

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 July 13

        weightlimit = 40
```

summary

In this notebook the double counting theorem is derived: the sum of the cardinalities of two finite sets is equal to the sum of the cardinalities of their union and intersection. The derivation relies on properties of the function **plus[x] = composite[NATADD, LEFT[x]]** which adds **x** to any number:

```
In[2]:= APPLY[plus[x], y]

Out[2]= natadd[x, y]
```

replacing a rule for range[plus[x]]

In this section a replacement will be derived for the following unnecessarily complicated rewrite rule:

```
In[3]:= range[plus[x]]

Out[3]= intersection[omega, image[S, set[x]], image[V, intersection[omega, set[x]]]]
```

We remove the rewrite rule now:

```
In[4]:= range[plus[x_]] =.
```

The **GOEDEL** program automatically recognizes the equality of the old formula for **range[plus[x]]** and what will become the new one. This equation amounts to the state-

ment that the set of natural numbers greater than or equal to **x** is the same as the set of natural numbers that are not less than **x**.

```
In[5]:= equal[intersection[omega,
          image[S, set[x]], image[V, intersection[omega, set[x]]]],
         intersection[omega, complement[x], image[V, intersection[omega, set[x]]]]]

Out[5]= True
```

This observation justifies the following rewrite rule:

```
In[6]:= intersection[omega, image[S, set[x_]],
          image[V, intersection[omega, set[x_]]]] :=
         intersection[omega, complement[x], image[V, intersection[omega, set[x]]]]
```

The following special case of the above rewrite rule is useful when **nat** wrappers are present.

```
In[7]:= SubstTest[intersection, omega, image[S, set[y]],
         image[V, intersection[omega, set[y]]], y → nat[x]]

Out[7]= intersection[omega, image[S, set[nat[x]]]] ==
         intersection[omega, complement[nat[x]]]

In[8]:= intersection[omega, image[S, set[nat[x_]]]] :=
         intersection[omega, complement[nat[x]]]
```

A replacement rule for **range[plus[x]]** can be derived as follows:

```
In[9]:= SubstTest[image, NATADD, cart[V, y], y → set[x]]

Out[9]= range[plus[x]] ==
         intersection[omega, complement[x], image[V, intersection[omega, set[x]]]]

In[10]:= range[plus[x_]] :=
          intersection[omega, complement[x], image[V, intersection[omega, set[x]]]]
```

Corollary. Since any image is contained in the range, one obtains:

```
In[11]:= SubstTest[subclass, image[w, y], range[w], w → plus[x]]

Out[11]= equal[0, intersection[x, image[plus[x], y]]] == True

In[12]:= intersection[x_, image[plus[x_], y_]] := 0
```

This says that if one adds **x** to every number belonging to a set **y**, then none of the resulting numbers can be less than **x**.

natadd lemmas

This section is devoted to deriving some simple rewrite rules about **natadd**. The first lemma says that since any natural number is the set of all lesser numbers, then in particular the sum of two natural numbers is a set of natural numbers.

```
In[13]:= SubstTest[subclass, nat[z], omega, z → natadd[nat[x], nat[y]]]

Out[13]= subclass[natadd[nat[x], nat[y]], omega] == True

In[14]:= subclass[natadd[nat[x_], nat[y_]], omega] := True
```

Corollary.

```
In[15]:= equal[intersection[omega, natadd[nat[x], nat[y]]], natadd[nat[x], nat[y]]]

Out[15]= True

In[16]:= intersection[omega, natadd[nat[x_], nat[y_]]] := natadd[nat[x], nat[y]]
```

The following rewrite rule follows from the fact that **x** is less than or equal to **x + y**.

```
In[17]:= equal[union[x, natadd[x, y]], natadd[x, y]]

Out[17]= True

In[18]:= union[x_, natadd[x_, y_]] := natadd[x, y]
```

The following rewrite rule for cardinality of a sum holds because **natadd[x,y]** is either a natural number of the universal class **V**. Any natural number is its own cardinality, and in the second case, **card[V] = V**.

```
In[19]:= (implies[or[equal[V, w], member[w, omega]], equal[card[w], w]] //
            NotNotTest) /. w → natadd[x, y]

Out[19]= equal[card[natadd[x, y]], natadd[x, y]] == True

In[20]:= card[natadd[x_, y_]] := natadd[x, y]
```

intersections of images

The set **image[plus[x], y]** is a set of natural numbers:

```
In[21]:= ImageComp[id[omega], plus[x], y] // Reverse

Out[21]= intersection[omega, image[plus[x], y]] == image[plus[x], y]

In[22]:= intersection[omega, image[plus[x_], y_]] := image[plus[x], y]
```

Lemma. Since the composite of **plus[x]** and **plus[y]** is **plus[natadd[x, y]]**, one has:

```
In[23]:= ImageComp[plus[nat[x]], plus[nat[y]], V] // Reverse

Out[23]= image[plus[nat[x]], intersection[omega, complement[nat[y]]]] ==
          intersection[omega, complement[natadd[nat[x], nat[y]]]]

In[24]:= image[plus[nat[x_]], intersection[omega, complement[nat[y_]]]] :=
          intersection[omega, complement[natadd[nat[x], nat[y]]]]
```

Since the function **plus[nat[x]]** is one-to-one, its images preserve relative complements:

```
In[25]:= Map[equal[image[plus[nat[x]], nat[y]], #] &,
           Map[dif[omega, union[nat[x], #]] &,
            SubstTest[intersection, image[plus[nat[x]], w],
             complement[image[plus[nat[x]], z]], {w → omega, z → nat[y]}]]] // Reverse

Out[25]= equal[image[plus[nat[x]], nat[y]],
           intersection[complement[nat[x]], natadd[nat[x], nat[y]]]] == True

In[26]:= image[plus[nat[x_]], nat[y_]] :=
          intersection[complement[nat[x]], natadd[nat[x], nat[y]]]
```

The following corollary will be used in the next section: **natadd[nat[x],nat[y]]** is the disjoint union of **nat[x]** and **image[plus[nat[x]],nat[y]]**.

```
In[27]:= intersection[nat[x], image[plus[nat[x]], nat[y]]]

Out[27]= 0

In[28]:= union[nat[x], image[plus[nat[x]], nat[y]]]

Out[28]= natadd[nat[x], nat[y]]
```

---

cardinality of disjoint unions of finite sets

Since **plus[nat[x]]** is one-to-one, it follows that the set of numbers less than **y** is equipollent to the set obtained by adding **x** to every number less than **y**.

```
In[29]:= SubstTest[implies, member[w, BIJ], member[pair[domain[w], range[w]], Q],
          w → composite[plus[nat[x]], id[nat[y]]]]

Out[29]= member[pair[nat[y],
           intersection[complement[nat[x]], natadd[nat[x], nat[y]]]], Q] == True

In[30]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the **nat** wrappers, and specializing to cardinal numbers, one finds:

```
In[31]:= (SubstTest[implies, and[equal[u, nat[w]], equal[v, nat[z]]],
            member[pair[u, intersection[complement[v], natadd[u, v]]], Q],
            {u → w, v → z}]) /. {w → card[x], z → card[y]}

Out[31]= or[member[pair[card[x],
             intersection[complement[card[y]], natadd[card[x], card[y]]]], Q],
           not[member[x, FINITE]], not[member[y, FINITE]]] == True

In[32]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Equipollence preserves disjoint unions.

```
In[33]:= Map[implies[member[x, FINITE], #] &,
          SubstTest[implies, and[member[pair[x, u], Q], member[pair[y, v], Q],
            disjoint[x, y], disjoint[u, v]], member[pair[union[x, y], union[u, v]], Q],
           {u → card[x], v -> dif[natadd[card[x], card[y]], card[x]]}]]

Out[33]= or[member[pair[union[x, y], natadd[card[x], card[y]]], Q],
           not[equal[0, intersection[x, y]]],
           not[member[x, FINITE]], not[member[pair[y,
              intersection[complement[card[x]], natadd[card[x], card[y]]]], Q]]] == True

In[34]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[35]:= Map[implies[member[x, FINITE], #] &,
          SubstTest[implies, and[member[pair[x, w], Q], member[pair[w, y], Q]],
           member[pair[x, y], Q], w → card[x]]]

Out[35]= or[member[pair[x, y], Q], not[member[x, FINITE]],
           not[member[pair[card[x], y], Q]]] == True

In[36]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

It follows that the union of two disjoint finite sets is equipollent to the sum of their cardinalities.

```
In[37]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           implies[and[p1, p3], p4], not[implies[p1, p4]],
           {p1 → and[member[x, FINITE], member[y, FINITE], disjoint[x, y]],
            p2 → member[pair[card[y], intersection[complement[card[x]],
                natadd[card[x], card[y]]]], Q], p3 → member[pair[y,
               intersection[complement[card[x]], natadd[card[x], card[y]]]], Q],
            p4 -> member[pair[union[x, y], natadd[card[x], card[y]]], Q]}]]
```

```
Out[37]= or[member[pair[union[x, y], natadd[card[x], card[y]]], Q],
           not[equal[0, intersection[x, y]]],
           not[member[x, FINITE]], not[member[y, FINITE]]] == True
```

```
In[38]:= or[member[pair[union[x_, y_], natadd[card[x_], card[y_]]], Q],
           not[equal[0, intersection[x_, y_]]],
           not[member[x_, FINITE]], not[member[y_, FINITE]]] := True
```

Lemma.

```
In[39]:= SubstTest[implies, and[member[pair[w, z], Q], member[w, FINITE]],
          equal[card[w], card[z]], {w → union[x, y], z -> natadd[card[x], card[y]]}]
```

```
Out[39]= or[equal[card[union[x, y]], natadd[card[x], card[y]]],
           not[member[x, FINITE]], not[member[y, FINITE]],
           not[member[pair[union[x, y], natadd[card[x], card[y]]], Q]]] == True
```

```
In[40]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The cardinality of the union of two disjoint finite sets is the sum of their cardinalities.

```
In[41]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> and[member[x, FINITE], member[y, FINITE], disjoint[x, y]],
            p2 -> member[pair[union[x, y], natadd[card[x], card[y]]], Q],
            p3 -> equal[card[union[x, y]], natadd[card[x], card[y]]]}]]
```

```
Out[41]= or[equal[card[union[x, y]], natadd[card[x], card[y]]],
           not[equal[0, intersection[x, y]]],
           not[member[x, FINITE]], not[member[y, FINITE]]] == True
```

```
In[42]:= or[equal[card[union[x_, y_]], natadd[card[x_], card[y_]]],
           not[equal[0, intersection[x_, y_]]],
           not[member[x_, FINITE]], not[member[y_, FINITE]]] := True
```

double counting theorem

Lemma.

```
In[43]:= SubstTest[implies, and[member[x, FINITE], member[z, FINITE], disjoint[x, z]],
          equal[card[union[x, z]], natadd[card[x], card[z]]], z → dif[y, x]]

Out[43]= or[equal[card[union[x, y]], natadd[card[x],
             card[intersection[y, complement[x]]]]], not[member[x, FINITE]],
           not[member[intersection[y, complement[x]], FINITE]]] == True

In[44]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[45]:= SubstTest[implies, and[member[w, FINITE], member[z, FINITE], disjoint[w, z]],
          equal[card[union[w, z]], natadd[card[w], card[z]]],
          {w → intersection[x, y], z → dif[y, x]}]

Out[45]= or[equal[card[y],
            natadd[card[intersection[x, y]], card[intersection[y, complement[x]]]]],
           not[member[intersection[x, y], FINITE]],
           not[member[intersection[y, complement[x]], FINITE]]] == True

In[46]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The double counting theorem follows:

```
In[47]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[and[p1, p3], p4], implies[and[p1, p2, p3], p5],
           implies[and[p1, p4, p5], p6], not[implies[p1, p6]],
           {p1 → and[member[x, FINITE], member[y, FINITE]],
            p2 → member[intersection[x, y], FINITE],
            p3 -> member[intersection[y, complement[x]], FINITE],
            p4 -> equal[card[union[x, y]],
              natadd[card[x], card[intersection[y, complement[x]]]]],
            p5 -> equal[card[y], natadd[card[intersection[x, y]],
               card[intersection[y, complement[x]]]]],
            p6 → equal[natadd[card[x], card[y]], natadd[
               card[union[x, y]], card[intersection[x, y]]]]}]]

Out[47]= or[equal[natadd[card[x], card[y]],
            natadd[card[intersection[x, y]], card[union[x, y]]]],
           not[member[x, FINITE]], not[member[y, FINITE]]] == True

In[48]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

When either **x** or **y** is not a finite set, the equation still holds because both sides are equal to **V**.

```
In[49]:= SubstTest[implies, and[equal[u, v], equal[v, w]],
          equal[u, w], {u -> natadd[card[x], card[y]], v → V,
           w -> natadd[card[intersection[x, y]], card[union[x, y]]]}]

Out[49]= or[and[member[x, FINITE], member[y, FINITE]], equal[natadd[card[x], card[y]],
           natadd[card[intersection[x, y]], card[union[x, y]]]]] == True

In[50]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Athough this does not add anything to our knowledge about cardinalities of unions, it does yield a simple rewrite rule:

```
In[51]:= SubstTest[and, implies[p1, p2], or[p1, p2],
          {p1 -> and[member[x, FINITE], member[y, FINITE]],
           p2 -> equal[natadd[card[x], card[y]],
             natadd[card[intersection[x, y]], card[union[x, y]]]]}]

Out[51]= True == equal[natadd[card[x], card[y]],
           natadd[card[intersection[x, y]], card[union[x, y]]]]

In[52]:= natadd[card[intersection[x_, y_]], card[union[x_, y_]]] :=
          natadd[card[x], card[y]]
```

---

a variable-free version of the double-counting theorem

A variable-free version of the double-counting rule is derived in this section.

```
In[53]:= symdif[composite[NATADD, cross[CARD, CARD], cross[CAP, CUP], DUP],
           composite[NATADD, cross[CARD, CARD]]] // VSTriNormality

Out[53]= union[composite[intersection[composite[NATADD, cross[CARD, CARD]], composite[
              complement[NATADD], intersection[composite[inverse[FIRST], CARD, CAP],
               composite[inverse[SECOND], CARD, CUP]]]], id[cart[V, V]]],
            composite[intersection[composite[NATADD, intersection[composite[
                 inverse[FIRST], CARD, CAP], composite[inverse[SECOND], CARD, CUP]]],
              composite[complement[NATADD], cross[CARD, CARD]]], id[cart[V, V]]]] == 0

In[54]:= % /. Equal → SetDelayed
```

Lemma.

```
In[55]:= equal[composite[intersection[x, composite[y, CAP]], id[cart[V, V]]],
          composite[intersection[x, composite[y, CAP]]]]

Out[55]= True

In[56]:= composite[intersection[x_, composite[y_, CAP]], id[cart[V, V]]] :=
          intersection[x, composite[y, CAP]]
```

The variable-free version of the double-counting theorem for finite sets follows:

```
In[57]:= SubstTest[equal, 0, composite[symdif[u, v], id[cart[V, V]]],
          {u -> composite[NATADD, cross[CARD, CARD], cross[CAP, CUP], DUP],
           v -> composite[NATADD, cross[CARD, CARD]]}]

Out[57]= True == equal[composite[NATADD, cross[CARD, CARD]],
           composite[NATADD, intersection[composite[inverse[FIRST], CARD, CAP],
             composite[inverse[SECOND], CARD, CUP]]]]

In[58]:= composite[NATADD, intersection[composite[inverse[FIRST], CARD, CAP],
            composite[inverse[SECOND], CARD, CUP]]] :=
          composite[NATADD, cross[CARD, CARD]]
```

Comment. It is emphasized once more that this rewrite rule only concerns the case of finite sets. The finiteness restriction in this formula comes from the presence of the function **NATADD**, which only adds natural numbers. One can also see this by examining the domain of the function on the right side of this equation:

```
In[59]:= domain[composite[NATADD, cross[CARD, CARD]]]

Out[59]= cart[FINITE, FINITE]
```