EQUIDIFF and subtraction

Johan G. F. Belinfante
2003 August 17

```
In[1]:= << goedel52.s78; << tools.m

        :Package Title: goedel52.s78       2003 August 14 at 4:45 p.m.

        It is now:  2003 Aug 19 at 13:12

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 August 9

        weightlimit = 40
```

summary

Integer arithmetic can be derived from natural number arithmetic by studying quantities that are invariant under the equivalence relation **EQUIDIFF**. Some properties of this equivalence relation are derived, especially ones that connect **EQUIDIFF** to the binary function **rotate[NATADD]** for subtraction of natural numbers. For example, the function **rotate[NATADD]** is invariant under **EQUIDIFF**, which implies that this function can be factored through the canonical projection; an explicit formula for this factorization is presented.

VERTSECT[EQUIDIFF]

The inverse of the canonical projection associated with the equivalence relation **EQUIDIFF** is:

```
In[2]:= composite[inverse[E], id[Z]] // DoubleInverse

Out[2]= composite[inverse[E], id[Z]] ==
         composite[id[cart[omega, omega]], inverse[VERTSECT[EQUIDIFF]]]

In[3]:= composite[inverse[E], id[Z]] :=
         composite[id[cart[omega, omega]], inverse[VERTSECT[EQUIDIFF]]]
```

The canonical projection is **EQUIDIFF** invariant:

```
In[4]:= Map[composite[IMAGE[EQUIDIFF], inverse[#]] &, Assoc[EQUIDIFF, inverse[E], id[Z]]]

Out[4]= composite[VERTSECT[EQUIDIFF], EQUIDIFF] ==
         composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]]

In[5]:= composite[VERTSECT[EQUIDIFF], EQUIDIFF] :=
         composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]]
```

This result is easy to understand. The canonical projection assigns to each point in the plane **cart[omega,omega]** the unique integer (line with slope 1) to which it belongs. Two points are related by the equivalence relation **EQUIDIFF** if they are on the same line. The canonical projection therefore assigns the same integer to two points that are equivalent.

flip and INVERSE

This section is about the flip of the canonical projection. If two points in the plane **cart[omega,omega]** are related by reflection, then the integers to which they belong are the inverse functions of each other.

```
In[6]:= dif[composite[VERTSECT[EQUIDIFF], SWAP],
          composite[INVERSE, VERTSECT[EQUIDIFF]]] // VSNormality

Out[6]= intersection[composite[complement[INVERSE], VERTSECT[EQUIDIFF]],
          composite[VERTSECT[EQUIDIFF], SWAP]] == 0

In[7]:= intersection[composite[complement[INVERSE], VERTSECT[EQUIDIFF]],
          composite[VERTSECT[EQUIDIFF], SWAP]] := 0

In[8]:= SubstTest[equal, 0, dif[u, v], {u -> composite[VERTSECT[EQUIDIFF], SWAP],
           v -> composite[INVERSE, VERTSECT[EQUIDIFF]]}] // Reverse

Out[8]= subclass[composite[VERTSECT[EQUIDIFF], SWAP],
          composite[INVERSE, VERTSECT[EQUIDIFF]]] == True

In[9]:= subclass[composite[VERTSECT[EQUIDIFF], SWAP],
          composite[INVERSE, VERTSECT[EQUIDIFF]]] := True

In[10]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
            equal[u, composite[v, id[domain[u]]]], {u -> composite[VERTSECT[EQUIDIFF], SWAP],
           v -> composite[INVERSE, VERTSECT[EQUIDIFF]]}]

Out[10]= equal[composite[VERTSECT[EQUIDIFF], SWAP],
           composite[INVERSE, VERTSECT[EQUIDIFF], id[cart[V, V]]]] == True

In[11]:= composite[VERTSECT[EQUIDIFF], SWAP] :=
          composite[INVERSE, VERTSECT[EQUIDIFF], id[cart[V, V]]]
```

A factor of **id[Z]** on the left of **VERTSECT[EQUIDIFF]** can be replaced by a factor **id[cart[omega,omega]]** on the right. A similar result holds when a factor of **INVERSE** is present:

```
In[12]:= Assoc[INVERSE, id[Z], VERTSECT[EQUIDIFF]] // Reverse

Out[12]= composite[id[Z], INVERSE, VERTSECT[EQUIDIFF]] ==
          composite[INVERSE, VERTSECT[EQUIDIFF], id[cart[omega, omega]]]

In[13]:= composite[id[Z], INVERSE, VERTSECT[EQUIDIFF]] :=
          composite[INVERSE, VERTSECT[EQUIDIFF], id[cart[omega, omega]]]
```

rotate[NATADD]

Here are some simplification rules for the natural subtraction function **rotate[NATADD]** and its flip:

```
In[14]:= Assoc[rotate[NATADD], id[domain[rotate[NATADD]]], id[cart[omega, omega]]] // Reverse

Out[14]= composite[rotate[NATADD], id[cart[omega, omega]]] == rotate[NATADD]

In[15]:= composite[rotate[NATADD], id[cart[omega, omega]]] := rotate[NATADD]
```

```
In[16]:= Assoc[rotate[NATADD], id[cart[omega, omega]], SWAP]

Out[16]= composite[rotate[NATADD], SWAP, id[cart[omega, omega]]] ==
          composite[rotate[NATADD], SWAP]

In[17]:= composite[rotate[NATADD], SWAP, id[cart[omega, omega]]] :=
          composite[rotate[NATADD], SWAP]

In[18]:= Assoc[rotate[NATADD],
           id[restrict[inverse[S], omega, omega]], id[inverse[S]]] // Reverse

Out[18]= composite[rotate[NATADD], id[inverse[S]]] == rotate[NATADD]

In[19]:= composite[rotate[NATADD], id[inverse[S]]] := rotate[NATADD]

In[20]:= Assoc[rotate[NATADD], id[inverse[S]], SWAP]

Out[20]= composite[rotate[NATADD], SWAP, id[S]] == composite[rotate[NATADD], SWAP]

In[21]:= composite[rotate[NATADD], SWAP, id[S]] := composite[rotate[NATADD], SWAP]
```

connection between EQUIDIFF and rotate[NATADD]

A simple formula relating **EQUIDIFF** and **rotate[NATADD]** will now be derived.

```
In[22]:= symdif[rotate[NATADD], composite[inverse[RIGHT[0]], EQUIDIFF]] // VSNormality

Out[22]= union[intersection[complement[rotate[NATADD]],
            composite[inverse[RIGHT[0]], EQUIDIFF]], intersection[
            composite[inverse[RIGHT[0]], complement[EQUIDIFF]], rotate[NATADD]]] == 0

In[23]:= union[intersection[complement[rotate[NATADD]],
            composite[inverse[RIGHT[0]], EQUIDIFF]], intersection[
            composite[inverse[RIGHT[0]], complement[EQUIDIFF]], rotate[NATADD]]] := 0

In[24]:= SubstTest[equal, 0, symdif[u, v],
             {u -> rotate[NATADD], v -> composite[inverse[RIGHT[0]], EQUIDIFF]}] // Reverse

Out[24]= equal[composite[inverse[RIGHT[0]], EQUIDIFF], rotate[NATADD]] == True

In[25]:= composite[inverse[RIGHT[0]], EQUIDIFF] := rotate[NATADD]
```

Some related formulas are:

```
In[26]:= composite[EQUIDIFF, RIGHT[0]] // DoubleInverse

Out[26]= composite[EQUIDIFF, RIGHT[0]] == inverse[rotate[NATADD]]

In[27]:= composite[EQUIDIFF, RIGHT[0]] := inverse[rotate[NATADD]]

In[28]:= Assoc[inverse[RIGHT[0]], EQUIDIFF, SWAP]

Out[28]= composite[inverse[LEFT[0]], EQUIDIFF] == composite[rotate[NATADD], SWAP]

In[29]:= composite[inverse[LEFT[0]], EQUIDIFF] := composite[rotate[NATADD], SWAP]
```

```
In[30]:= composite[EQUIDIFF, LEFT[0]] // DoubleInverse

Out[30]= composite[EQUIDIFF, LEFT[0]] == composite[SWAP, inverse[rotate[NATADD]]]

In[31]:= composite[EQUIDIFF, LEFT[0]] := composite[SWAP, inverse[rotate[NATADD]]]
```

The function **rotate[NATADD]** is invariant under **EQUIDIFF**.

```
In[32]:= Assoc[inverse[RIGHT[0]], EQUIDIFF, EQUIDIFF] // Reverse

Out[32]= composite[rotate[NATADD], EQUIDIFF] == rotate[NATADD]

In[33]:= composite[rotate[NATADD], EQUIDIFF] := rotate[NATADD]
```

Here are some related formulas:

```
In[34]:= composite[EQUIDIFF, inverse[rotate[NATADD]]] // DoubleInverse

Out[34]= composite[EQUIDIFF, inverse[rotate[NATADD]]] == inverse[rotate[NATADD]]

In[35]:= composite[EQUIDIFF, inverse[rotate[NATADD]]] := inverse[rotate[NATADD]]

In[36]:= Assoc[rotate[NATADD], EQUIDIFF, SWAP]

Out[36]= composite[rotate[NATADD], SWAP, EQUIDIFF] == composite[rotate[NATADD], SWAP]

In[37]:= composite[rotate[NATADD], SWAP, EQUIDIFF] := composite[rotate[NATADD], SWAP]
```

PLUS

The function **PLUS** takes each natural number to the corresponding non−negative integer. The natural projection takes a pair of natural numbers to the integer whose graph holds this pair of natural numbers. In this section, a relation between **PLUS** and **VERTSECT[EQUIDIFF]** is derived. This follows from the observation that the graph of the positive integer **plus[x]** holds the point **pair[0,x]**. In other words, the unary function **PLUS** can be obtained from the binary function **VERTSECT[EQUIDIFF]** by setting the left−hand argument equal to **0**. To show this, it is easiest to begin by deriving an inclusion, and later strengthening it to an equation.

```
In[38]:= dif[PLUS, composite[VERTSECT[EQUIDIFF], LEFT[0]]] // VSNormality

Out[38]= intersection[PLUS, composite[complement[VERTSECT[EQUIDIFF]], LEFT[0]]] == 0

In[39]:= intersection[PLUS, composite[complement[VERTSECT[EQUIDIFF]], LEFT[0]]] := 0

In[40]:= SubstTest[equal, 0, dif[u, v],
           {u -> PLUS, v -> composite[VERTSECT[EQUIDIFF], LEFT[0]]}] // Reverse

Out[40]= subclass[PLUS, composite[VERTSECT[EQUIDIFF], LEFT[0]]] == True

In[41]:= subclass[PLUS, composite[VERTSECT[EQUIDIFF], LEFT[0]]] := True
```

A subclass of a function is a restriction of that function. This allows us to replace the above inclusion with an equation.

```
In[42]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
             equal[u, composite[v, id[domain[u]]]],
           {u -> PLUS, v -> composite[VERTSECT[EQUIDIFF], LEFT[0]]}]

Out[42]= equal[PLUS, composite[VERTSECT[EQUIDIFF], LEFT[0], id[omega]]] == True

In[43]:= composite[VERTSECT[EQUIDIFF], LEFT[0], id[omega]] := PLUS
```

A similar result can be obtained with **RIGHT[0]** replacing **LEFT[0]**.

```
In[44]:= Assoc[VERTSECT[EQUIDIFF], SWAP, composite[LEFT[0], id[omega]]]

Out[44]= composite[VERTSECT[EQUIDIFF], RIGHT[0], id[omega]] == composite[INVERSE, PLUS]

In[45]:= composite[VERTSECT[EQUIDIFF], RIGHT[0], id[omega]] := composite[INVERSE, PLUS]
```

factoring

Lemmas:

```
In[46]:= composite[inverse[PLUS], id[Z]] // DoubleInverse

Out[46]= composite[inverse[PLUS], id[Z]] == inverse[PLUS]

In[47]:= composite[inverse[PLUS], id[Z]] := inverse[PLUS]

In[48]:= Assoc[inverse[PLUS], id[Z], VERTSECT[EQUIDIFF]]

Out[48]= composite[inverse[PLUS], VERTSECT[EQUIDIFF], id[cart[omega, omega]]] ==
          composite[inverse[PLUS], VERTSECT[EQUIDIFF]]

In[49]:= composite[inverse[PLUS], VERTSECT[EQUIDIFF], id[cart[omega, omega]]] :=
          composite[inverse[PLUS], VERTSECT[EQUIDIFF]]

In[50]:= Assoc[VERTSECT[EQUIDIFF], EQUIDIFF, RIGHT[0]]

Out[50]= composite[VERTSECT[EQUIDIFF], inverse[rotate[NATADD]]] == composite[INVERSE, PLUS]

In[51]:= composite[VERTSECT[EQUIDIFF], inverse[rotate[NATADD]]] := composite[INVERSE, PLUS]

In[52]:= composite[rotate[NATADD], inverse[VERTSECT[EQUIDIFF]]] // DoubleInverse

Out[52]= composite[rotate[NATADD], inverse[VERTSECT[EQUIDIFF]]] ==
          composite[inverse[PLUS], INVERSE]

In[53]:= composite[rotate[NATADD], inverse[VERTSECT[EQUIDIFF]]] :=
          composite[inverse[PLUS], INVERSE]
```

Any function invariant under **EQUIDIFF** can be factored through the canonical projection. In the case of **rotate[NATADD]**, this can be done explicitly:

```
In[54]:= Assoc[rotate[NATADD],
           composite[inverse[VERTSECT[EQUIDIFF]], id[Z]], VERTSECT[EQUIDIFF]] // Reverse

Out[54]= composite[inverse[PLUS], INVERSE, VERTSECT[EQUIDIFF]] == rotate[NATADD]

In[55]:= composite[inverse[PLUS], INVERSE, VERTSECT[EQUIDIFF]] := rotate[NATADD]
```

```
In[56]:= Assoc[composite[inverse[PLUS], INVERSE], composite[id[Z], VERTSECT[EQUIDIFF]], SWAP]

Out[56]= composite[inverse[PLUS], VERTSECT[EQUIDIFF]] == composite[rotate[NATADD], SWAP]

In[57]:= composite[inverse[PLUS], VERTSECT[EQUIDIFF]] := composite[rotate[NATADD], SWAP]

In[58]:= composite[inverse[VERTSECT[EQUIDIFF]], PLUS] // DoubleInverse

Out[58]= composite[inverse[VERTSECT[EQUIDIFF]], PLUS] ==
          composite[SWAP, inverse[rotate[NATADD]]]

In[59]:= composite[inverse[VERTSECT[EQUIDIFF]], PLUS] :=
          composite[SWAP, inverse[rotate[NATADD]]]

In[60]:= ImageComp[inverse[VERTSECT[EQUIDIFF]], PLUS, V] // Reverse

Out[60]= image[inverse[VERTSECT[EQUIDIFF]], range[PLUS]] == composite[id[omega], S, id[omega]]

In[61]:= image[inverse[VERTSECT[EQUIDIFF]], range[PLUS]] := composite[id[omega], S, id[omega]]

In[62]:= composite[inverse[VERTSECT[EQUIDIFF]], INVERSE, PLUS] // DoubleInverse

Out[62]= composite[inverse[VERTSECT[EQUIDIFF]], INVERSE, PLUS] == inverse[rotate[NATADD]]

In[63]:= composite[inverse[VERTSECT[EQUIDIFF]], INVERSE, PLUS] := inverse[rotate[NATADD]]
```

simplifications

The following formulas are used to simplify some formulas in the next section.

```
In[64]:= Assoc[EQUIDIFF, id[cart[omega, omega]], id[x]]

Out[64]= composite[EQUIDIFF, id[composite[id[omega], x, id[omega]]]] ==
          composite[EQUIDIFF, id[x]]

In[65]:= composite[EQUIDIFF, id[composite[id[omega], x_, id[omega]]]] :=
          composite[EQUIDIFF, id[x]]

In[66]:= composite[id[composite[id[omega], x, id[omega]]], EQUIDIFF] // DoubleInverse

Out[66]= composite[id[composite[id[omega], x, id[omega]]], EQUIDIFF] ==
          composite[id[x], EQUIDIFF]

In[67]:= composite[id[composite[id[omega], x_, id[omega]]], EQUIDIFF] :=
          composite[id[x], EQUIDIFF]
```

EQUIDIFF commutes with id[S]

A pair of points **pair[u,v]** and **pair[x,y]** in the plane **cart[omega,omega]** are related by **EQUIDIFF** if and only if they belong to the same integer, that is, they lie on a line with slope 1. If one of these points lies above the main diagonal **id[omega]**, then so does the other. This implies that **EQUIDIFF** commutes with **id[S]**, a fact that will be derived in this section. For these points **v** is greater than **u** and **y** is greater than **x**, and the differences are equal: **natsub[v,u] = natsub[y,x]**. A similar result holds for pairs of points below the diagonal, which is expressed in variable−free form as follows:

```
In[68]:= Map[composite[SWAP, inverse[VERTSECT[EQUIDIFF]], #, SWAP] &,
          Assoc[PLUS, inverse[PLUS], VERTSECT[EQUIDIFF]]]

Out[68]= composite[inverse[rotate[NATADD]], rotate[NATADD]] ==
          composite[EQUIDIFF, id[inverse[S]]]

In[69]:= composite[inverse[rotate[NATADD]], rotate[NATADD]] :=
          composite[EQUIDIFF, id[inverse[S]]]

In[70]:= SubstTest[inverse, composite[inverse[x], x], x -> rotate[NATADD]]

Out[70]= composite[id[inverse[S]], EQUIDIFF] == composite[EQUIDIFF, id[inverse[S]]]

In[71]:= composite[id[inverse[S]], EQUIDIFF] := composite[EQUIDIFF, id[inverse[S]]]

In[72]:= Map[composite[SWAP, #] &, Assoc[id[inverse[S]], EQUIDIFF, SWAP]]

Out[72]= composite[id[S], EQUIDIFF] == composite[EQUIDIFF, id[S]]

In[73]:= composite[id[S], EQUIDIFF] := composite[EQUIDIFF, id[S]]
```

EQUIDIFF and rotate[NATADD]

The complete relation **EQUIDIFF** is the union of the restrictions above and below the diagonal that were discussed in the preceding section.

```
In[74]:= SubstTest[union, composite[EQUIDIFF, id[x]],
          composite[EQUIDIFF, id[inverse[x]]], x -> restrict[S, omega, omega]]

Out[74]= composite[EQUIDIFF, id[union[S, inverse[S]]]] == EQUIDIFF

In[75]:= composite[EQUIDIFF, id[union[S, inverse[S]]]] := EQUIDIFF
```

The relation **EQUIDIFF** can therefore be constructed as a union of two parts, each constructed from **rotate[NATADD]**.

```
In[76]:= union[composite[inverse[rotate[NATADD]], rotate[NATADD]],
          composite[SWAP, inverse[rotate[NATADD]], rotate[NATADD], SWAP]]

Out[76]= EQUIDIFF

In[77]:= union[composite[SWAP, inverse[rotate[NATADD]], rotate[NATADD]],
          composite[inverse[rotate[NATADD]], rotate[NATADD], SWAP]]

Out[77]= composite[SWAP, EQUIDIFF]
```

some twist results

```
In[78]:= SubstTest[twist, twist[x], x -> composite[cross[NATADD, NATADD], inverse[RIF]]]

Out[78]= intersection[EQUIDIFF, composite[inverse[FIRST], S, FIRST]] ==
          composite[cross[NATADD, NATADD], inverse[RIF]]

In[79]:= intersection[EQUIDIFF, composite[inverse[FIRST], S, FIRST]] :=
          composite[cross[NATADD, NATADD], inverse[RIF]]
```

```
In[80]:= SubstTest[twist, twist[x],
          x -> composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]]

Out[80]= intersection[EQUIDIFF, composite[inverse[FIRST], inverse[S], FIRST]] ==
          composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]

In[81]:= intersection[EQUIDIFF, composite[inverse[FIRST], inverse[S], FIRST]] :=
          composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]

In[82]:= SubstTest[twist, twist[x], x ->
            intersection[EQUIDIFF, composite[inverse[SECOND], inverse[S], SECOND]]] // Reverse

Out[82]= intersection[EQUIDIFF, composite[inverse[SECOND], inverse[S], SECOND]] ==
          composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]

In[83]:= intersection[EQUIDIFF, composite[inverse[SECOND], inverse[S], SECOND]] :=
          composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]

In[84]:= intersection[EQUIDIFF, composite[inverse[SECOND], S, SECOND]] // DoubleInverse

Out[84]= intersection[EQUIDIFF, composite[inverse[SECOND], S, SECOND]] ==
          composite[cross[NATADD, NATADD], inverse[RIF]]

In[85]:= intersection[EQUIDIFF, composite[inverse[SECOND], S, SECOND]] :=
          composite[cross[NATADD, NATADD], inverse[RIF]]

In[86]:= AssInt[EQUIDIFF, composite[inverse[SECOND], S, SECOND],
           composite[inverse[SECOND], S, SECOND]] // Reverse

Out[86]= intersection[composite[cross[NATADD, NATADD], inverse[RIF]],
           composite[inverse[SECOND], S, SECOND]] ==
          composite[cross[NATADD, NATADD], inverse[RIF]]

In[87]:= intersection[composite[cross[NATADD, NATADD], inverse[RIF]],
           composite[inverse[SECOND], S, SECOND]] :=
          composite[cross[NATADD, NATADD], inverse[RIF]]

In[88]:= AssInt[EQUIDIFF, composite[inverse[FIRST], S, FIRST],
           composite[inverse[FIRST], S, FIRST]] // Reverse

Out[88]= intersection[composite[cross[NATADD, NATADD], inverse[RIF]],
           composite[inverse[FIRST], S, FIRST]] == composite[cross[NATADD, NATADD], inverse[RIF]]

In[89]:= intersection[composite[cross[NATADD, NATADD], inverse[RIF]],
           composite[inverse[FIRST], S, FIRST]] :=
          composite[cross[NATADD, NATADD], inverse[RIF]]

In[90]:= AssInt[EQUIDIFF, composite[inverse[SECOND], S, SECOND],
          composite[inverse[FIRST], S, FIRST]]

Out[90]= intersection[EQUIDIFF, cross[S, S]] == composite[cross[NATADD, NATADD], inverse[RIF]]

In[91]:= intersection[EQUIDIFF, cross[S, S]] := composite[cross[NATADD, NATADD], inverse[RIF]]

In[92]:= intersection[EQUIDIFF, inverse[cross[S, S]]] // DoubleInverse

Out[92]= intersection[EQUIDIFF, cross[inverse[S], inverse[S]]] ==
          composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]

In[93]:= intersection[EQUIDIFF, cross[inverse[S], inverse[S]]] :=
          composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]
```

a curious twist

The following formula relates **NATADD** and **rotate[NATADD]** in a curious way. The **twist** formula amounts to the observation that natural numbers satisfying either **u − v = x − y** or **v − u = y − x** must satisfy **u + y = x + v**.

```
In[94]:= SubstTest[twist, twist[x], x -> union[
             composite[RIF, cross[composite[SWAP, inverse[rotate[NATADD]]], inverse[NATADD]]],
             composite[RIF, cross[inverse[NATADD], inverse[rotate[NATADD]]]]] // Reverse

Out[94]= union[
           composite[RIF, cross[composite[SWAP, inverse[rotate[NATADD]]], inverse[NATADD]]],
           composite[RIF, cross[inverse[NATADD], inverse[rotate[NATADD]]]]] ==
         composite[inverse[NATADD], NATADD]
```