integer addition and vector addition

Johan G. F. Belinfante
2006 September 30

```
In[1]:= SetDirectory["l:"]; << goedel85.30a; << tools.m

        :Package Title: goedel85.30a          2006 September 30 at 3:10 p.m.

        It is now:  2006 Sep 30 at 18:19

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 September 24

        weightlimit = 40
```

summary

The set **Z** of integers has been constructed in the **GOEDEL** program as the equivalence classes of a relation **EQUIDIFF**. These equivalence classes are the lines in the natural number plane **cart[omega, omega]** that are parallel to **id[omega]**. The line **id[omega]** itself represents the integer zero. The lines above **id[omega]** represent positive integers, and those below represent negative integers. Vector addition of subsets of the plane, like integer addition, is commutative and associatve but these operations are not identical. The problem is that the vector sum of two integers may be only a subset of an integer. In order to obtain an integer , one needs to use the function **HULL[Z]**, which transforms any non-empty subset of an integer to the unique integer that contains it. This completion process is only needed when applied to a positive and a negative integer. In this notebook explicit formulas are derived relating integer addition to vector addition of subsets of the natural number plane. These formulas are derived by considering separately the case of adding two positive integers and the case of adding a positive integer to a negative integer.

vector addition of number pairs

The points **PAIR[x,y]** of the plane **cart[omega,omega]** can be thought of as vectors. The binary operation of vector addition is the direct product of **NATADD** with itself. By definition, the direct product is the composite of the cross product with the function **TWIST**. The twist is needed to interchange **x2** and **y1** so that the first component of the vector sum is the sum of the first components of the individual vectors, and similarly for the second components.

```
In[2]:= APPLY[composite[cross[NATADD, NATADD], TWIST], PAIR[PAIR[x1, x2], PAIR[y1, y2]]]

Out[2]= PAIR[natadd[x1, y1], natadd[x2, y2]]
```

In addition to being commutative and associative, vector addition also has a reflection symmetry with respect to interchanging the two components of each vector. This reflection symmetry can be formulated in a variable-free fashion as follows:

```
In[3]:= Assoc[composite[SWAP, cross[NATADD, NATADD]], SWAP, composite[TWIST, cross[SWAP, SWAP]]]

Out[3]= composite[SWAP, cross[NATADD, NATADD], TWIST] ==
         composite[cross[NATADD, NATADD], TWIST, cross[SWAP, SWAP]]

In[4]:= composite[SWAP, cross[NATADD, NATADD], TWIST] :=
         composite[cross[NATADD, NATADD], TWIST, cross[SWAP, SWAP]]
```

---

vector addition of subsets of cart[omega,omega]

For any binary operation on a set, there is a corresponding binary operation on subsets. The vector sum of subsets **x** and **y** of the number plane is the set of all vector sums **u** + **v** where **u** belongs to **x** and **v** belongs to **y**. This associative binary operation of vector addition of subsets of the natural number plane **cart[omega, omega]** is the restriction of the function **composite[IMAGE[cross[NATADD, NATADD]], CROSS]** to the power set of **cart[omega, omega]**.

```
In[5]:= member[composite[IMAGE[cross[NATADD, NATADD]], CROSS,
          id[cart[P[cart[omega, omega]], P[cart[omega, omega]]]]], SEMIGPS]

Out[5]= True
```

Explicitly, if **x** and **y** are subsets of the natural number plane **cart[omega, omega]**, then their vector sum is the subset **composite[NATADD, cross[x,y], inverse[NATADD]]**.

```
In[6]:= APPLY[composite[IMAGE[cross[NATADD, NATADD]], CROSS], PAIR[setpart[x], setpart[y]]]

Out[6]= composite[NATADD, cross[setpart[x], setpart[y]], inverse[NATADD]]
```

Vector addition of subsets is commutative, and the inverse of a vector sum of subsets is the vector sum of their inverses:

```
In[7]:= inverse[composite[NATADD, cross[x, y], inverse[NATADD]]]

Out[7]= composite[NATADD, cross[inverse[x], inverse[y]], inverse[NATADD]]
```

A variable-free formulation of this symmetry can be derived. First some lemmas:

```
In[8]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], IMAGE[cross[x, y]]]

Out[8]= composite[IMAGE[SWAP], IMAGE[cross[x, y]]] == composite[INVERSE, IMAGE[cross[x, y]]]

In[9]:= composite[IMAGE[SWAP], IMAGE[cross[x_, y_]]] := composite[INVERSE, IMAGE[cross[x, y]]]
```

Lemma.

```
In[10]:= Map[VERTSECT, Assoc[cross[NATADD, NATADD], SWAP, inverse[E]]]

Out[10]= composite[IMAGE[cross[NATADD, NATADD]], IMAGE[SWAP]] ==
          composite[INVERSE, IMAGE[cross[NATADD, NATADD]]]

In[11]:= composite[IMAGE[cross[NATADD, NATADD]], IMAGE[SWAP]] :=
          composite[INVERSE, IMAGE[cross[NATADD, NATADD]]]
```

Theorem. Reflection symmetry of vector addition of subsets of the number plane.

```
In[12]:= Assoc[IMAGE[cross[NATADD, NATADD]], IMAGE[SWAP], CROSS] // Reverse

Out[12]= composite[INVERSE, IMAGE[cross[NATADD, NATADD]], CROSS] ==
          composite[IMAGE[cross[NATADD, NATADD]], CROSS, cross[IMAGE[SWAP], IMAGE[SWAP]]]

In[13]:= composite[INVERSE, IMAGE[cross[NATADD, NATADD]], CROSS] :=
          composite[IMAGE[cross[NATADD, NATADD]], CROSS, cross[IMAGE[SWAP], IMAGE[SWAP]]]
```

In deriving results about vector sums of integers **x + y**, there are in principle four cases to consider, because each of the two integers could be either positive or negative. Since negative integers are constructed as inverses of positive integers, the reflection symmetry of vector addition permits the reduction of these four cases to just two: the vector sum of two positive integers, and the vector sum of a positive and a negative integer.

vector addition for integers with the same sign

If **x** and **y** are natural numbers, then **plus[x]** and **plus[y]** are non-negative integers, and their vector sum is also a non-negative integer:

```
In[14]:= Map[composite[#, inverse[NATADD]] &,
          Assoc[composite[NATADD, cross[NATADD, NATADD]], TWIST, RIGHT[PAIR[x, y]]]]

Out[14]= composite[NATADD, cross[plus[x], plus[y]], inverse[NATADD]] == plus[natadd[x, y]]

In[15]:= composite[NATADD, cross[plus[x_], plus[y_]], inverse[NATADD]] := plus[natadd[x, y]]
```

A variable-free restatement of this fact is obtained using **reify** as follows:

```
In[16]:= Map[equal[#, composite[INTADD, cross[PLUS, PLUS]]] &,
          SubstTest[reify, x, image[z, set[x]],
           z -> composite[IMAGE[cross[NATADD, NATADD]], CROSS, cross[PLUS, PLUS]]]]

Out[16]= True == equal[composite[INTADD, cross[PLUS, PLUS]],
           composite[IMAGE[cross[NATADD, NATADD]], CROSS, cross[PLUS, PLUS]]]

In[17]:= composite[IMAGE[cross[NATADD, NATADD]], CROSS, cross[PLUS, PLUS]] :=
          composite[INTADD, cross[PLUS, PLUS]]
```

Corollary.

```
In[18]:= Assoc[composite[IMAGE[cross[NATADD, NATADD]], CROSS],
          cross[PLUS, PLUS], cross[inverse[PLUS], inverse[PLUS]]]

Out[18]= composite[IMAGE[cross[NATADD, NATADD]], CROSS, id[cart[range[PLUS], range[PLUS]]]] ==
          composite[INTADD, id[cart[range[PLUS], range[PLUS]]]]

In[19]:= % /. Equal → SetDelayed
```

A corresponding result for two non-positive integers can be obtained as follows:

```
In[20]:= Map[composite[#, inverse[cross[composite[INVERSE, PLUS], composite[INVERSE, PLUS]]]] &,
          Assoc[IMAGE[SWAP], composite[IMAGE[cross[NATADD, NATADD]], CROSS],
            cross[PLUS, PLUS]] // Reverse]

Out[20]= composite[IMAGE[cross[NATADD, NATADD]], CROSS,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]] ==
         composite[INTADD, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]

In[21]:= % /. Equal → SetDelayed
```

---

vector sums of integers with opposite signs

Lemma.

```
In[22]:= composite[inverse[NATADD], inverse[plus[x]]] // DoubleInverse

Out[22]= composite[inverse[NATADD], inverse[plus[x]]] ==
          composite[cross[Id, inverse[plus[x]]], inverse[NATADD]]

In[23]:= composite[inverse[NATADD], inverse[plus[x_]]] :=
          composite[cross[Id, inverse[plus[x]]], inverse[NATADD]]
```

Theorem. The vector sum of a positive and a negative integer is their composite. This composite need not be an integer, but is a subset of an integer.

```
In[24]:= Map[composite[#, inverse[plus[y]]] &,
           Assoc[composite[NATADD, cross[Id, plus[x]]], SWAP, inverse[NATADD]]] // Reverse

Out[24]= composite[NATADD, cross[plus[x], inverse[plus[y]]], inverse[NATADD]] ==
          composite[plus[x], inverse[plus[y]]]

In[25]:= composite[NATADD, cross[plus[x_], inverse[plus[y_]]], inverse[NATADD]] :=
          composite[plus[x], inverse[plus[y]]]
```

Observation. The vector sum of integers with opposite signs is only a subset of an integer. When **x** and **y** are natural numbers, the composite in the reverse order is an integer:

```
In[26]:= member[composite[inverse[plus[y]], plus[x]], Z]

Out[26]= and[member[x, omega], member[y, omega]]
```

Taking the hull of **composite[plus[x], inverse[plus[y]]]** yields this integer **composite[inverse[plus[y]],plus[x]]**.

```
In[27]:= hull[Z, composite[NATADD, cross[plus[x], inverse[plus[y]]], inverse[NATADD]]]

Out[27]= composite[inverse[plus[y]], plus[x]]
```

Thus to obtain integer addition, one needs to use **HULL[Z]**:

```
In[28]:= Assoc[HULL[Z], composite[COMPOSE, id[cart[Z, Z]]],
          cross[PLUS, composite[INVERSE, PLUS]]]

Out[28]= composite[HULL[Z], COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]] ==
          composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]

In[29]:= composite[HULL[Z], COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]] :=
          composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]
```

Again, one can use **reify** to obtain a variable-free restatement of the formula for vector sum of a positive and negative integer considered as subsets of **cart[omega, omega]**.

```
In[30]:= Map[composite[#, inverse[cross[PLUS, composite[INVERSE, PLUS]]]] &,
          Map[composite[HULL[Z], #] &,
           SubstTest[reify, x, image[z, set[x]], z -> composite[IMAGE[cross[NATADD, NATADD]],
               CROSS, cross[PLUS, composite[INVERSE, PLUS]]]] // Reverse]]

Out[30]= composite[HULL[Z], IMAGE[cross[NATADD, NATADD]],
           CROSS, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]] ==
          composite[INTADD, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]]

In[31]:= % /. Equal → SetDelayed
```

Lemmas.

```
In[32]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], id[range[PLUS]]]

Out[32]= composite[IMAGE[SWAP], id[range[PLUS]]] == composite[INVERSE, id[range[PLUS]]]

In[33]:= composite[IMAGE[SWAP], id[range[PLUS]]] := composite[INVERSE, id[range[PLUS]]]

In[34]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], id[image[INVERSE, range[PLUS]]]]

Out[34]= composite[IMAGE[SWAP], id[image[INVERSE, range[PLUS]]]] ==
          composite[INVERSE, id[image[INVERSE, range[PLUS]]]]

In[35]:= composite[IMAGE[SWAP], id[image[INVERSE, range[PLUS]]]] :=
          composite[INVERSE, id[image[INVERSE, range[PLUS]]]]
```

Theorem.

```
In[36]:= Map[composite[#, cross[INVERSE, INVERSE]] &,
          Assoc[HULL[Z], INVERSE, composite[IMAGE[cross[NATADD, NATADD]],
            CROSS, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]]]]

Out[36]= composite[HULL[Z], IMAGE[cross[NATADD, NATADD]],
           CROSS, id[cart[image[INVERSE, range[PLUS]], range[PLUS]]]] ==
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], range[PLUS]]]]

In[37]:= % /. Equal → SetDelayed
```

the final result

Lemma.

```
In[38]:= Assoc[HULL[Z], id[Z], INTADD]

Out[38]= composite[HULL[Z], INTADD] == INTADD

In[39]:= composite[HULL[Z], INTADD] := INTADD
```

Combining all the cases yields this formula that expresses integer addition as a completion of vector addition:

```
In[40]:= SubstTest[composite, t, union[w, x, y, z],
          {t -> composite[HULL[Z], IMAGE[cross[NATADD, NATADD]], CROSS],
           w → id[cart[range[PLUS], range[PLUS]]],
           x -> id[cart[range[PLUS], image[INVERSE, range[PLUS]]]],
           y -> id[cart[image[INVERSE, range[PLUS]], range[PLUS]]],
           z -> id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]}]

Out[40]= composite[HULL[Z], IMAGE[cross[NATADD, NATADD]], CROSS, id[cart[Z, Z]]] == INTADD

In[41]:= composite[HULL[Z], IMAGE[cross[NATADD, NATADD]], CROSS, id[cart[Z, Z]]] := INTADD
```

reintroducing variables

Reintroducing variables, one can derive an explicit formula that expresses the integer sum **intadd[x, y]** of two integers as the integer hull of their vector sum, **composite[NATADD, cross[x, y], inverse[NATADD]]**. To derive this formula, it is convenient to beging with a lemma in which all variables are wrapped with **setpart**:

```
In[42]:= Map[equal[intadd[setpart[x], setpart[y]], #] &,
           SubstTest[APPLY, composite[funpart[u], id[v]], w,
            {u -> composite[HULL[Z], IMAGE[cross[NATADD, NATADD]], CROSS],
             v -> cart[Z, Z], w → PAIR[setpart[x], setpart[y]]}]] // Reverse

Out[42]= or[equal[hull[Z, composite[NATADD, cross[setpart[x], setpart[y]], inverse[NATADD]]],
            intadd[setpart[x], setpart[y]]],
           not[member[setpart[x], Z]], not[member[setpart[y], Z]]] == True

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

Removing the **setpart** wrappers yields this explicit formula for the integer sum of two integers as the hull of their vector sum.

```
In[44]:= SubstTest[implies, and[equal[x, setpart[u]], equal[y, setpart[v]]],
          or[equal[hull[Z, composite[NATADD, cross[x, y], inverse[NATADD]]], intadd[x, y]],
           not[member[x, Z]], not[member[y, Z]]], {u → x, v → y}]

Out[44]= or[equal[hull[Z, composite[NATADD, cross[x, y], inverse[NATADD]]], intadd[x, y]],
           not[member[x, Z]], not[member[y, Z]]] == True
```

```
In[45]:= or[equal[hull[Z, composite[NATADD, cross[x_, y_], inverse[NATADD]]], intadd[x_, y_]],
           not[member[x_, Z]], not[member[y_, Z]]] := True
```

For comparison, a similar formula expresses the integer sum as the hull of their composite:

```
In[46]:= Map[equal[intadd[x, y], #] &, SubstTest[APPLY, composite[funpart[u], id[v]], w,
            {u -> composite[HULL[Z], COMPOSE], v -> cart[Z, Z], w → PAIR[x, y]}]] // Reverse

Out[46]= or[equal[hull[Z, composite[x, y]], intadd[x, y]],
           not[member[x, Z]], not[member[y, Z]]] == True

In[47]:= or[equal[hull[Z, composite[x_, y_]], intadd[x_, y_]],
           not[member[x_, Z]], not[member[y_, Z]]] := True
```

In the **GOEDEL** program, integer addition was defined using integer hulls of composites. The upshot of this notebook is that one could just as well have used vector addition instead of composition to define integer addition.