INTADD is commutative

Johan G. F. Belinfante
2003 July 26

```
In[1]:= << goedel52.s64; << tools.m

        :Package Title: goedel52.s64       2003 July 25 at 10:30 p.m.

        It is now:  2003 Aug 15 at 10:36

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 August 9

        weightlimit = 40
```

summary

It is shown that addition of integers is commutative. The derivation is done by considering all possible combinations of positive and negative integers. A key ingredient of the proof is that **plus[x]** subcommutes with **inverse[plus[y]]**.

PLUS rules

The first step is to derive some simplification rules for **INVERSE** and **PLUS**.

```
In[2]:= equal[composite[id[Z], PLUS], PLUS]

Out[2]= True

In[3]:= composite[id[Z], PLUS] := PLUS

In[4]:= equal[composite[id[Z], INVERSE, PLUS], composite[INVERSE, PLUS]]

Out[4]= True

In[5]:= composite[id[Z], INVERSE, PLUS] := composite[INVERSE, PLUS]

In[6]:= commute[INVERSE, id[Z]] // AssertTest

Out[6]= equal[composite[INVERSE, id[Z]], composite[id[Z], INVERSE]] == True
```

The following orientation for a rewrite rule is tentative:

```
In[7]:= composite[INVERSE, id[Z]] := composite[id[Z], INVERSE]

In[8]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], id[Z]]

Out[8]= composite[IMAGE[SWAP], id[Z]] == composite[id[Z], INVERSE]
```

```
In[9]:= composite[IMAGE[SWAP], id[Z]] := composite[id[Z], INVERSE]

In[10]:= Assoc[IMAGE[SWAP], id[Z], PLUS]

Out[10]= composite[IMAGE[SWAP], PLUS] == composite[INVERSE, PLUS]

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

The following is added for completeness:

```
In[12]:= Assoc[IMAGE[SWAP], IMAGE[SWAP], id[Z]] // Reverse

Out[12]= composite[IMAGE[id[cart[V, V]]], id[Z]] == id[Z]

In[13]:= composite[IMAGE[id[cart[V, V]]], id[Z]] := id[Z]
```

case of +−

The key case is the case of a positive integer and a negative integer. This is the hardest case. The idea is to use:

```
In[14]:= subcommute[plus[x], inverse[plus[y]]]

Out[14]= True
```

First some rules are added to convert expressions invvolving **COMPOSE** to formulas with **INTADD**.

```
In[15]:= Assoc[composite[id[Z], S, COMPOSE],
          id[cart[Z, Z]], cross[PLUS, composite[INVERSE, PLUS]]]

Out[15]= composite[id[Z], S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]] ==
          composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]

In[16]:= composite[id[Z], S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]] :=
          composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]
```

The following was not expected, and is probably not important:

```
In[17]:= Assoc[composite[id[Z], S, COMPOSE],
          id[cart[Z, Z]], cross[composite[INVERSE, PLUS], PLUS]]

Out[17]= composite[id[Z], S, VERTSECT[EQUIDIFF], id[cart[omega, omega]]] ==
          composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]]

In[18]:= composite[id[Z], S, VERTSECT[EQUIDIFF], id[cart[omega, omega]]] :=
          composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]]

In[19]:= Assoc[composite[id[Z], S, COMPOSE], cross[PLUS, composite[INVERSE, PLUS]], SWAP]

Out[19]= composite[id[Z], S, COMPOSE, SWAP, cross[composite[INVERSE, PLUS], PLUS]] ==
          composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]]

In[20]:= composite[id[Z], S, COMPOSE, SWAP, cross[composite[INVERSE, PLUS], PLUS]] :=
          composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]]

In[21]:= Assoc[composite[id[Z], S, COMPOSE], cross[composite[INVERSE, PLUS], PLUS], SWAP]

Out[21]= composite[id[Z], S, COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]] ==
          composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]]
```

```
In[22]:= composite[id[Z], S, COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]] :=
          composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]]
```

The main argument uses the subcommutativity of **plus[x]** and **inverse[plus[y]]**:

```
In[23]:= dif[composite[COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
            composite[S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]]] // VSTriNormality

Out[23]= composite[intersection[composite[COMPOSE, SWAP], composite[complement[S], COMPOSE]],
           cross[PLUS, composite[INVERSE, PLUS]]] == 0

In[24]:= composite[intersection[composite[COMPOSE, SWAP], composite[complement[S], COMPOSE]],
           cross[PLUS, composite[INVERSE, PLUS]]] := 0

In[25]:= SubstTest[equal, 0, dif[u, v],
           {u -> composite[COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
             v -> composite[S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]]}] // Reverse

Out[25]= subclass[composite[COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
           composite[S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]]] == True

In[26]:= subclass[composite[COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
           composite[S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]]] := True

In[27]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
             {u -> composite[COMPOSE, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
             v -> composite[S, COMPOSE, cross[PLUS, composite[INVERSE, PLUS]]],
                 w -> composite[id[Z], S]}]

Out[27]= subclass[composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
           composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]] == True

In[28]:= subclass[composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
           composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]] := True

In[29]:= SubstTest[implies, subclass[u, v], subclass[flip[u], flip[v]],
             {u -> composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]],
           v -> composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]}]

Out[29]= subclass[composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]],
           composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]]] == True

In[30]:= subclass[composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]],
           composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]]] := True
```

To transform this inclusion to an equation, one needs to use the fact that a subclass of a function must be a restriction.

```
In[31]:= SubstTest[implies, and[subclass[x, y], FUNCTION[y]],
          equal[x, composite[y, id[domain[x]]]],
             {x -> composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]],
                 y -> composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]]}]

Out[31]= equal[composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]],
           composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]]] == True

In[32]:= composite[INTADD, SWAP, cross[composite[INVERSE, PLUS], PLUS]] :=
          composite[INTADD, cross[composite[INVERSE, PLUS], PLUS]]

In[33]:= Assoc[composite[INTADD, SWAP], cross[composite[INVERSE, PLUS], PLUS],
          inverse[cross[composite[INVERSE, PLUS], PLUS]]]

Out[33]= composite[INTADD, SWAP, id[cart[image[INVERSE, range[PLUS]], range[PLUS]]]] ==
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], range[PLUS]]]]
```

```
In[34]:= composite[INTADD, SWAP, id[cart[image[INVERSE, range[PLUS]], range[PLUS]]]] :=
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], range[PLUS]]]]

In[35]:= Assoc[composite[INTADD, SWAP], cross[composite[INVERSE, PLUS], PLUS], SWAP] // Reverse

Out[35]= composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]] ==
          composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]

In[36]:= composite[INTADD, SWAP, cross[PLUS, composite[INVERSE, PLUS]]] :=
          composite[INTADD, cross[PLUS, composite[INVERSE, PLUS]]]

In[37]:= Assoc[flip[INTADD], cross[PLUS, composite[INVERSE, PLUS]],
          inverse[cross[PLUS, composite[INVERSE, PLUS]]]]

Out[37]= composite[INTADD, SWAP, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]] ==
          composite[INTADD, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]]

In[38]:= composite[INTADD, SWAP, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]] :=
          composite[INTADD, id[cart[range[PLUS], image[INVERSE, range[PLUS]]]]]
```

case of ++

```
In[39]:= Map[composite[#, inverse[cross[PLUS, PLUS]]] &, Assoc[PLUS, NATADD, SWAP]] // Reverse

Out[39]= composite[COMPOSE, SWAP, id[cart[range[PLUS], range[PLUS]]]] ==
          composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]

In[40]:= composite[COMPOSE, SWAP, id[cart[range[PLUS], range[PLUS]]]] :=
          composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]

In[41]:= Assoc[HULL[Z], composite[COMPOSE, id[cart[Z, Z]]], id[cart[range[PLUS], range[PLUS]]]]

Out[41]= composite[HULL[Z], COMPOSE, id[cart[range[PLUS], range[PLUS]]]] ==
          composite[INTADD, id[cart[range[PLUS], range[PLUS]]]]

In[42]:= composite[HULL[Z], COMPOSE, id[cart[range[PLUS], range[PLUS]]]] :=
          composite[INTADD, id[cart[range[PLUS], range[PLUS]]]]

In[43]:= Assoc[HULL[Z], composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]], SWAP] // Reverse

Out[43]= composite[INTADD, SWAP, id[cart[range[PLUS], range[PLUS]]]] ==
          composite[INTADD, id[cart[range[PLUS], range[PLUS]]]]

In[44]:= composite[INTADD, SWAP, id[cart[range[PLUS], range[PLUS]]]] :=
          composite[INTADD, id[cart[range[PLUS], range[PLUS]]]]
```

case of −−

```
In[45]:= Map[composite[#, inverse[cross[composite[INVERSE, PLUS], composite[INVERSE, PLUS]]]] &,
          Assoc[composite[INVERSE, PLUS], NATADD, SWAP]]

Out[45]= composite[COMPOSE, SWAP,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]] == composite[
           COMPOSE, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]

In[46]:= composite[COMPOSE, SWAP,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]] := composite[
           COMPOSE, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]
```

```
In[47]:= Assoc[HULL[Z], composite[COMPOSE, id[cart[Z, Z]]],
          id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]

Out[47]= composite[HULL[Z], COMPOSE,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]] ==
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]

In[48]:= composite[HULL[Z], COMPOSE,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]] :=
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]

In[49]:= Assoc[HULL[Z], composite[COMPOSE, id[
             cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]], SWAP] // Reverse

Out[49]= composite[INTADD, SWAP,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]] ==
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]

In[50]:= composite[INTADD, SWAP,
           id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]] :=
          composite[INTADD, id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]]
```

combining all the cases

```
In[51]:= SubstTest[composite, flip[INTADD], union[u, v, x, y],
             {u -> id[cart[range[PLUS], range[PLUS]]],
           v -> id[cart[range[PLUS], image[INVERSE, range[PLUS]]]],
                 x -> id[cart[image[INVERSE, range[PLUS]], range[PLUS]]],
           y -> id[cart[image[INVERSE, range[PLUS]], image[INVERSE, range[PLUS]]]]}]

Out[51]= composite[INTADD, SWAP] == INTADD
```

This is a variable−free statement of the commutative law of integer addition.

```
In[52]:= composite[INTADD, SWAP] := INTADD
```

corollary

This section is concerned with a simplification rule related to the commutative law.

```
In[53]:= Assoc[composite[HULL[Z], COMPOSE], id[cart[Z, Z]], SWAP]

Out[53]= composite[HULL[Z], COMPOSE, SWAP, id[cart[Z, Z]]] == INTADD

In[54]:= composite[HULL[Z], COMPOSE, SWAP, id[cart[Z, Z]]] := INTADD

In[55]:= Assoc[composite[id[Z], S, COMPOSE], id[cart[Z, Z]], SWAP]

Out[55]= composite[id[Z], S, COMPOSE, SWAP, id[cart[Z, Z]]] == INTADD

In[56]:= composite[id[Z], S, COMPOSE, SWAP, id[cart[Z, Z]]] := INTADD
```