associativity of integer addition

Johan G. F. Belinfante
2003 August 9

```
In[1]:= << goedel52.s70; << tools.m

        :Package Title: goedel52.s70       2003 August 8 at 9:10 p.m.

        It is now:  2003 Aug 15 at 9:23

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 August 9

        weightlimit = 40
```

summary

The associative law for addition of integers is derived in this notebook, using the fact that direct products of associative relations are associative, and that the homomorphic image of an associative operation is associative. Specifically, it is shown that **INTADD** is the homomorphic image of the direct product of **NATADD** with itself under the natural projection

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

Out[2]= composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]]
```

In practice the factor **id[cart[omega,omega]]** can be eliminated by using:

```
In[3]:= Assoc[INTADD, id[cart[Z, Z]], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]

Out[3]= composite[INTADD, cross[composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]],
          composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]]]] ==
         composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]

In[4]:= composite[INTADD, cross[composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]],
          composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]]]] :=
         composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]
```

derivation

The fact that **plus[x]** subcommutes with **inverse[plus[y]]** implies:

```
In[5]:= SubstTest[implies, subclass[u, v], subclass[composite[s, u, t], composite[s, v, t]],
           {u -> composite[plus[x], inverse[plus[y]]],
         v -> composite[inverse[plus[y]], plus[x]],
              s -> inverse[plus[w]], t -> plus[z]}]

Out[5]= subclass[composite[inverse[plus[w]], plus[x], inverse[plus[y]], plus[z]],
          composite[inverse[plus[natadd[w, y]]], plus[natadd[x, z]]]] == True
```

```
In[6]:= subclass[composite[inverse[plus[w_]], plus[x_], inverse[plus[y_]], plus[z_]],
          composite[inverse[plus[natadd[w_, y_]]], plus[natadd[x_, z_]]]] := True
```

The **simplify** and **cond** flags will be turned off briefly to speed up some hard steps.

```
In[7]:= simplify = False; cond = False;
```

The following simplification rule is needed to clean up the results of a **VSTerNormality** test to be carried out shortly.

```
In[8]:= equal[composite[intersection[
           composite[complement[INTADD], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
           composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]],
          id[cart[cart[V, V], V]]], intersection[
          composite[complement[INTADD], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
          composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]]]

Out[8]= True
```

```
In[9]:= composite[intersection[
           composite[complement[INTADD], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
           composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]],
          id[cart[cart[V, V], V]]] := intersection[
          composite[complement[INTADD], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
          composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]]
```

The fact that **INTADD** is a homomorphic image of the direct product of two copies of **NATADD** is based on the following application of **VSTerNormality**.

```
In[10]:= dif[composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST],
           composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]] // VSTerNormality

Out[10]= intersection[
           composite[complement[INTADD], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
           composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]] == 0
```

```
In[11]:= intersection[
           composite[complement[INTADD], cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
           composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]] := 0
```

This only produces an inclusion, but it will quickly be strengthened to an equation.

```
In[12]:= SubstTest[equal, 0, dif[u, v],
             {u -> composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST],
              v -> composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]}] // Reverse

Out[12]= subclass[composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST],
           composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]] == True
```

```
In[13]:= subclass[composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST],
           composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]] := True
```

Any subclass of a function is a restriction. In the present case, these functions are equal:

```
In[14]:= SubstTest[implies, and[subclass[x, y], FUNCTION[y]],
          equal[x, composite[y, id[domain[x]]]],
             {x -> composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST],
           y -> composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]}]

Out[14]= equal[composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]],
           composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST]] == True
```

This says that **INTADD** is a homomorphic image of the direct product of **NATADD** with itself.

```
In[15]:= composite[VERTSECT[EQUIDIFF], cross[NATADD, NATADD], TWIST] :=
          composite[INTADD, cross[VERTSECT[EQUIDIFF], VERTSECT[EQUIDIFF]]]
```

The latter is associative:

```
In[16]:= SubstTest[implies, and[associative[x], associative[y]],
            associative[composite[cross[x, y], TWIST]], {x -> NATADD, y -> NATADD}]

Out[16]= associative[composite[cross[NATADD, NATADD], TWIST]] == True

In[17]:= associative[composite[cross[NATADD, NATADD], TWIST]] := True
```

The associativity of the direct product of **NATADD** with itself can be restated:

```
In[18]:= AssertTest[associative[x]] /. x -> composite[cross[NATADD, NATADD], TWIST] // Reverse

Out[18]= equal[composite[cross[NATADD, NATADD],
            TWIST, cross[composite[cross[NATADD, NATADD], TWIST], Id]],
           composite[cross[NATADD, NATADD], TWIST,
            cross[Id, composite[cross[NATADD, NATADD], TWIST]], ASSOC]] == True

In[19]:= composite[cross[NATADD, NATADD], TWIST,
           cross[Id, composite[cross[NATADD, NATADD], TWIST]], ASSOC] := composite[
           cross[NATADD, NATADD], TWIST, cross[composite[cross[NATADD, NATADD], TWIST], Id]]
```

The **simplify** flag is now turned back on.

```
In[20]:= simplify = True;
```

Two simplification rules are needed:

```
In[21]:= Assoc[INTADD, id[cart[Z, Z]], cross[INTADD, Id]]

Out[21]= composite[INTADD, cross[INTADD, id[Z]]] == composite[INTADD, cross[INTADD, Id]]

In[22]:= composite[INTADD, cross[INTADD, id[Z]]] := composite[INTADD, cross[INTADD, Id]]

In[23]:= Assoc[INTADD, id[cart[Z, Z]], cross[Id, INTADD]]

Out[23]= composite[INTADD, cross[id[Z], INTADD]] == composite[INTADD, cross[Id, INTADD]]

In[24]:= composite[INTADD, cross[id[Z], INTADD]] := composite[INTADD, cross[Id, INTADD]]
```

The fact that **INTADD** is associative follows from the fact that the natural projection is onto, that is, its range is **Z**.

```
In[25]:= Map[composite[#, cross[cross[z, z], z]] &,
           Assoc[VERTSECT[EQUIDIFF], composite[cross[NATADD, NATADD], TWIST],
             composite[cross[Id, composite[cross[NATADD, NATADD], TWIST]], ASSOC]] // Reverse] /.
          z -> composite[inverse[VERTSECT[EQUIDIFF]], id[Z]]

Out[25]= composite[INTADD, cross[Id, INTADD], ASSOC] == composite[INTADD, cross[INTADD, Id]]

In[26]:= composite[INTADD, cross[Id, INTADD], ASSOC] := composite[INTADD, cross[INTADD, Id]]
```

Restatement:

```
In[27]:= (associative[x] // AssertTest) /. x -> INTADD

Out[27]= associative[INTADD] == True
```

```
In[28]:= associative[INTADD] := True
```