

id[Z] o INVERSE is an automorphism of INTADD

Johan G. F. Belinfante
2003 August 10

```
In[1]:= << goedel52.s73; << tools.m

:Package Title: goedel52.s73      2003 August 9 at 4:00 p.m.

It is now: 2003 Aug 15 at 11:27

Loading Simplification Rules

TOOLS.M                          Revised 2003 August 9

weightlimit = 40
```

summary

The restriction of **INVERSE** to **Z** is an automorphism of **INTADD**. This automorphism sends each integer to its negative.

lemmas

The orientation for the following rule is tentatively adopted to facilitate subsequent computations.

```
In[2]:= commute[INVERSE, id[Z]] // AssertTest

Out[2]= equal[composite[INVERSE, id[Z]], composite[id[Z], INVERSE]] == True

In[3]:= composite[INVERSE, id[Z]] := composite[id[Z], INVERSE]
```

Technical lemma:

```
In[4]:= equiv[or[not[member[x, Z]], not[member[domain[x], V]], not[member[range[x], V]]],
             not[member[x, Z]]]

Out[4]= True

In[5]:= or[not[member[x_, Z]], not[member[domain[x_], V]], not[member[range[x_], V]]] :=
         not[member[x, Z]]
```

Theorem: The (additive) inverse of an integer is an integer. Here **inverse** means negative.

```
In[6]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
                 {u -> singleton[x], v -> Z, w -> IMAGE[SWAP]}]

Out[6]= or[member[inverse[x], Z], not[member[x, Z]]] == True

In[7]:= or[member[inverse[x_], Z], not[member[x_, Z]]] := True
```

Some simplifications needed in the sequel.

```
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[INVERSE, id[Z]]

In[10]:= Assoc[IMAGE[SWAP], id[Z], INTADD]
Out[10]= composite[IMAGE[SWAP], INTADD] = composite[INVERSE, INTADD]

In[11]:= composite[IMAGE[SWAP], INTADD] := composite[INVERSE, INTADD]

In[12]:= Assoc[INTADD, id[cart[Z, Z]], cross[INVERSE, INVERSE]]
Out[12]= composite[INTADD, cross[composite[id[Z], INVERSE], composite[id[Z], INVERSE]]] =
  composite[INTADD, cross[INVERSE, INVERSE]]

In[13]:= composite[INTADD, cross[composite[id[Z], INVERSE], composite[id[Z], INVERSE]]] :=
  composite[INTADD, cross[INVERSE, INVERSE]]
```

Technical lemma.

```
In[14]:= AssInt[cart[cart[V, V], V], INTADD, x]
Out[14]= composite[intersection[INTADD, x], id[cart[V, V]]] = intersection[INTADD, x]

In[15]:= composite[intersection[INTADD, x_], id[cart[V, V]]] := intersection[INTADD, x]
```

main idea

The main idea is to use the membership rule for **INTADD**.

```
In[16]:= ((implies[member[pair[pair[x, y], z], INTADD],
  member[pair[pair[inverse[y], inverse[x]], inverse[z]], INTADD]] // NotNotTest) /.
  {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

The variables can be eliminated:

```
In[17]:= Map[composite[complement[#], id[cart[V, V]]] &,
  SubstTest[class, pair[pair[x, y], z], implies[member[pair[pair[x, y], z], w],
  member[pair[pair[inverse[y], inverse[x]], inverse[z]], w]],
  w -> INTADD]] // Reverse

Out[17]= intersection[INTADD, composite[
  complement[inverse[IMAGE[SWAP]]], INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]]] = 0

In[18]:= intersection[INTADD, composite[complement[inverse[IMAGE[SWAP]]],
  INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]]] := 0
```

This can be rewritten as an inclusion.

```
In[19]:= SubstTest[equal, 0, dif[u, v], {u -> INTADD, v -> composite[
  inverse[IMAGE[SWAP]], INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]}] // Reverse

Out[19]= subclass[INTADD,
  composite[inverse[IMAGE[SWAP]], INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]]] = True
```

```
In[20]:= subclass[INTADD,
  composite[inverse[IMAGE[SWAP]], INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]] := True
```

It is better to rewrite this as an inclusion involving functions:

```
In[21]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
  {u -> INTADD,
  v -> composite[inverse[IMAGE[SWAP]], INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]],
  w -> IMAGE[SWAP]}]
```

```
Out[21]= subclass[composite[INVERSE, INTADD],
  composite[INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]] == True
```

```
In[22]:= subclass[composite[INVERSE, INTADD],
  composite[INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]] := True
```

An inclusion involving functions can be rewritten as an equation. This is the main result.

```
In[23]:= SubstTest[implies, and[subclass[x, y], FUNCTION[y]],
  equal[x, composite[y, id[domain[x]]],
  {x -> composite[INVERSE, INTADD],
  y -> composite[INTADD, cross[IMAGE[SWAP], IMAGE[SWAP]]]}]
```

```
Out[23]= equal[composite[INTADD, cross[INVERSE, INVERSE]], composite[INVERSE, INTADD]] == True
```

```
In[24]:= composite[INVERSE, INTADD] := composite[INTADD, cross[INVERSE, INVERSE]]
```

comment

One of the lemmas derived earlier is rewritten by the main result. We remove this lemma:

```
In[25]:= composite[IMAGE[SWAP], INTADD] =.
```

A replacement is derived:

```
In[26]:= Assoc[IMAGE[SWAP], id[Z], INTADD]
```

```
Out[26]= composite[IMAGE[SWAP], INTADD] == composite[INTADD, cross[INVERSE, INVERSE]]
```

```
In[27]:= composite[IMAGE[SWAP], INTADD] := composite[INTADD, cross[INVERSE, INVERSE]]
```