whole numbers

Johan G. F. Belinfante
2012 August 4

```
In[1]:= SetDirectory["l:"]; << goedel.12aug03a
```

```
:Package Title: goedel.12aug03a                2012 August 3 at 11:30 a.m.

Loading takes about sixteen minutes, half that time due to builtin pauses.

It is now:  2012 Aug 4 at 5:10

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2012 Aug 4 at 5:25
```

summary

A whole number is a rational number of the form **inttimes[int[x]]**. Whole numbers are closed under composition, and forms a commutative monoid with **id[Z]** as neutral element.

whole numbers

The **GOEDEL** program contains a rewrite rule about whole numbers being rational, but the following is an improvement.

Theorem. Whole numbers are rational.

```
In[2]:= SubstTest[member, composite[inverse[inttimes[u]], inttimes[v]],
          RATS, {u → plus[set[0]], v → x}] // Reverse
```

```
Out[2]= member[inttimes[x], RATS] == member[x, Z]
```

```
In[3]:= member[inttimes[x_], RATS] := member[x, Z]
```

The old rewrite rule can be removed. The following is also an improvement of an existing rewrite rule.

Theorem. Reciprocal non-zero whole numbers are rational.

```
In[4]:= SubstTest[member, composite[inverse[inttimes[u]], inttimes[v]],
          RATS, {u → x, v → plus[set[0]]}] // Reverse
```

```
Out[4]= member[inverse[inttimes[x]], RATS] == and[member[x, Z], not[equal[x, id[omega]]]]
```

```
In[5]:= member[inverse[inttimes[x_]], RATS] := and[member[x, Z], not[equal[x, id[omega]]]]
```

The class of whole numbers is the range of **INTTIMES**.

```
In[6]:= range[INTTIMES]
```

```
Out[6]= binhom[INTADD, INTADD]
```

symmetric whole numbers

In this sectionit is shown that the only whole numbers that are also reciprocal whole numbers are **id[Z]** and **id[Z]** ∘ **INVERSE**.

Lemma.

```
In[13]:= SubstTest[image, INTDIV, set[inverse[int[x]]], x → plus[set[0]]] // Reverse
```

```
Out[13]= image[INTDIV, set[composite[inverse[SUCC], id[omega]]]] == Z
```

```
In[14]:= image[INTDIV, set[composite[inverse[SUCC], id[omega]]]] := Z
```

Lemma. Simplification rule.

```
In[16]:= equiv[and[equal[x, composite[id[omega], SUCC]], member[x, V]],
           equal[x, composite[id[omega], SUCC]]]
```

```
Out[16]= True
```

```
In[17]:= and[equal[x_, composite[id[omega], SUCC]], member[x_, V]] :=
           equal[x, composite[id[omega], SUCC]]
```

Lemma. Simplification rule.

```
In[18]:= equiv[and[equal[x, composite[inverse[SUCC], id[omega]]], member[x, V]],
           equal[x, composite[inverse[SUCC], id[omega]]]]
```

```
Out[18]= True
```

```
In[19]:= and[equal[x_, composite[inverse[SUCC], id[omega]]], member[x_, V]] :=
           equal[x, composite[inverse[SUCC], id[omega]]]
```

Theorem.

```
In[20]:= SubstTest[member, x, image[inverse[t], set[plus[set[0]]]], t → INTDIV]

Out[20]= member[pair[x, composite[id[omega], SUCC]], INTDIV] == or[
           equal[x, composite[id[omega], SUCC]], equal[x, composite[inverse[SUCC], id[omega]]]]

In[21]:= member[pair[x_, composite[id[omega], SUCC]], INTDIV] := or[
           equal[x, composite[id[omega], SUCC]], equal[x, composite[inverse[SUCC], id[omega]]]]
```

Lemma.

```
In[22]:= SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
           {u → plus[set[0]], v → Z, w → image[INTDIV, set[x]]}] // Reverse

Out[22]= or[equal[x, composite[id[omega], SUCC]], equal[x, composite[inverse[SUCC], id[omega]]],
           not[equal[Z, image[INTDIV, set[x]]]]] == True

In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[24]:= equiv[equal[Z, image[INTDIV, set[x]]], or[equal[x, composite[id[omega], SUCC]],
             equal[x, composite[inverse[SUCC], id[omega]]]]] // not // not

Out[24]= True

In[25]:= equal[Z, image[INTDIV, set[x_]]] := or[equal[x, composite[id[omega], SUCC]],
           equal[x, composite[inverse[SUCC], id[omega]]]]
```

Lemma.

```
In[27]:= SubstTest[implies, equal[u, v], equal[domain[u], domain[v]],
           {u → inttimes[int[x]], v → inverse[inttimes[int[x]]]}] // Reverse

Out[27]= or[equal[composite[id[omega], SUCC], int[x]],
           equal[composite[inverse[SUCC], id[omega]], int[x]],
           not[equal[inttimes[int[x]], inverse[inttimes[int[x]]]]]] == True

In[28]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Eliminate the **int** wrapper.

```
In[29]:= SubstTest[implies, equal[x, int[t]], or[
             equal[composite[id[omega], SUCC], x], equal[composite[inverse[SUCC], id[omega]], x],
             not[equal[inttimes[x], inverse[inttimes[x]]]]], t → x] // Reverse

Out[29]= or[equal[x, composite[id[omega], SUCC]], equal[x, composite[inverse[SUCC], id[omega]]],
           not[equal[inttimes[x], inverse[inttimes[x]]]], not[member[x, Z]]] == True

In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Eliminate the variable **x**.)

```
In[37]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[implies[member[x, u], member[x, v]]],
           {v → set[composite[id[omega], SUCC], composite[inverse[SUCC], id[omega]]],
            u -> image[inverse[INTTIMES], intersection[SYM, binhom[INTADD, INTADD]]]}]]

Out[37]= subclass[image[inverse[INTTIMES], intersection[SYM, binhom[INTADD, INTADD]]],
           set[composite[id[omega], SUCC], composite[inverse[SUCC], id[omega]]]] == True

In[38]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[40]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]], {t → INTTIMES,
            u -> image[inverse[INTTIMES], intersection[SYM, binhom[INTADD, INTADD]]], v ->
             set[composite[id[omega], SUCC], composite[inverse[SUCC], id[omega]]]}] // Reverse

Out[40]= subclass[intersection[SYM, binhom[INTADD, INTADD]],
           set[composite[id[Z], INVERSE], id[Z]]] == True

In[41]:= % /. Equal→ SetDelayed
```

Corollary. A better rewrite rule.

```
In[42]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[SYM, binhom[INTADD, INTADD]],
           v -> set[composite[id[Z], INVERSE], id[Z]]}]

Out[42]= equal[intersection[SYM, binhom[INTADD, INTADD]],
           set[composite[id[Z], INVERSE], id[Z]]] == True

In[44]:= intersection[SYM, binhom[INTADD, INTADD]] := set[composite[id[Z], INVERSE], id[Z]]
```

closure under composition

Whole numbers are closed under composition.

```
In[46]:= subclass[image[COMPOSE, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]],
          binhom[INTADD, INTADD]]

Out[46]= True
```

This rewrite rule can be improved upon.

Lemma.

```
In[50]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → COMPOSE, u -> cart[set[id[Z]], binhom[INTADD, INTADD]],
            v → cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]}] // Reverse

Out[50]= subclass[binhom[INTADD, INTADD],
           image[COMPOSE, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]] == True
```

```
In[51]:= % /. Equal→ SetDelayed
```

Theorem. Improved rewrite rule.

```
In[52]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[COMPOSE, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]],
           v -> binhom[INTADD, INTADD]}]

Out[52]= equal[binhom[INTADD, INTADD],
           image[COMPOSE, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]] == True

In[55]:= image[COMPOSE, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]] :=
          binhom[INTADD, INTADD]
```

The restriction of **COMPOSE** to **binhom[INTADD,INTADD]** is a semigroup.

```
In[56]:= member[
          composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]], SEMIGPS]

Out[56]= True
```

Lemma. A neutral element for this semigroup is **id[Z]**.

```
In[58]:= member[id[Z], ids[composite[COMPOSE,
             id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]]] // AssertTest

Out[58]= member[id[Z], ids[composite[COMPOSE,
             id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]]] == True

In[59]:= member[id[Z], ids[composite[COMPOSE,
             id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]]] := True
```

Lemma.

```
In[62]:= SubstTest[implies, and[member[v, BINOPS], member[u, ids[v]]],
           equal[e[v], u], {u → id[Z], v -> composite[COMPOSE,
              id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]}] // Reverse

Out[62]= equal[e[composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]],
           id[Z]] == True

In[64]:= e[composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]] :=
          id[Z]
```

Theorem. The only neutral element is **id[Z]**.

```
In[68]:= SubstTest[implies, member[x, BINOPS], equal[ids[x], set[e[x]]], x -> composite[COMPOSE,
             id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]] // Reverse

Out[68]= equal[
           ids[composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]],
           set[id[Z]]] == True
```

```
In[70]:= ids[composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]] :=
          set[id[Z]]
```

Corollary. The restriction of **COMPOSE** to whole numbers is a monoid.

```
In[72]:= SubstTest[and, member[x, SEMIGPS], member[e[x], V],
          x -> composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]]

Out[72]= member[composite[COMPOSE,
            id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]], MONOIDS] == True

In[73]:= member[composite[COMPOSE,
            id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]], MONOIDS] := True
```

Lemma.

```
In[86]:= SubstTest[class, pair[u, v], member[pair[u, v], w],
          w -> composite[inverse[INTTIMES], COMMUTE, INTTIMES]]

Out[86]= composite[inverse[INTTIMES], COMMUTE, INTTIMES] == cart[Z, Z]

In[87]:= composite[inverse[INTTIMES], COMMUTE, INTTIMES] := cart[Z, Z]
```

Lemma.

```
In[90]:= Map[equal[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]], #] &,
          ImageComp[cross[INTTIMES, INTTIMES], inverse[cross[INTTIMES, INTTIMES]], COMMUTE]]

Out[90]= subclass[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]], COMMUTE] == True

In[91]:= subclass[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]], COMMUTE] := True
```

Theorem.

```
In[94]:= SubstTest[subclass, domain[funpart[x]],
          fix[composite[inverse[funpart[x]], funpart[y]]],
          {x → composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]],
           y → composite[COMPOSE, SWAP]}]

Out[94]= subclass[composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]],
           composite[COMPOSE, SWAP]] == True

In[95]:= % /. Equal→ SetDelayed
```

Corollary. The restriction of **COMPOSE** to whole numbers is commutative.

```
In[98]:= SubstTest[equal, composite[funpart[y], id[domain[x]]], x,
           {x -> composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]],
            y -> composite[COMPOSE, SWAP]}] // Reverse

Out[98]= equal[composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]],
           composite[COMPOSE, SWAP,
            id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]] == True
```

In[100]:=

```
composite[COMPOSE, SWAP, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]] :=
 composite[COMPOSE, id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]]
```

Restatement.

In[101]:=

```
member[composite[COMPOSE,
  id[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]], COMMUTATIVE]
```

Out[101]=

```
True
```