additive inverses for rational numbers

Johan G. F. Belinfante
2012 October 6

```
In[1]:= SetDirectory["l:"]; << goedel.12oct05a

        :Package Title: goedel.12oct05a               2012 October 4 at 2:55 p.m.

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

        It is now:  2012 Oct 6 at 12:16

        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 Oct 6 at 12:32
```

summary

The inversion function for rational addition is multiplication by the rational number minus one, **– 1 = id[Z] ∘ INVERSE**. The rational numbers form a group under addition.

derivation

Theorem. The product of a rational number with the rational number minus one is its composite with **INVERSE** in either order.

```
In[2]:= SubstTest[equal, ratmul[rat[x], t],
          ratmul[t, rat[x]], t -> composite[id[Z], INVERSE]] // Reverse

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

In[3]:= ratmul[composite[id[Z], INVERSE], rat[x_]] := composite[rat[x], INVERSE]
```

The fraction **inttimes[int[x]] = 1 \ int[x]** whose denominator is the integer one and whose numerator is the integer **int[x]** will here be called a **whole number**. There is an isomorphism between integer addition and rational addition of whole numbers: **(1 \ x) + (1 \ y) = 1 \ (x + y)**.

Theorem. The numerator of the sum of two whole numbers is the integer sum of their numerators.

```
In[5]:= SubstTest[hull, RATS, composite[INTADD,
          intersection[composite[inverse[FIRST], rat[u]], composite[inverse[SECOND], rat[v]]]],
         {u → inttimes[int[x]], v → inttimes[int[y]]}]
```

```
Out[5]= ratadd[inttimes[int[x]], inttimes[int[y]]] == inttimes[intadd[int[x], int[y]]]
```

```
In[6]:= ratadd[inttimes[int[x_]], inttimes[int[y_]]] := inttimes[intadd[int[x], int[y]]]
```

Lemma. Rational addition is commutative.

```
In[7]:= Map[equal[#, ratadd[x, y]] &, ApComp[RATADD, SWAP, PAIR[x, y]]]
```

```
Out[7]= equal[ratadd[x, y], ratadd[y, x]] == True
```

```
In[8]:= equal[ratadd[x_, y_], ratadd[y_, x_]] := True
```

Theorem. The sum of the rational numbers plus one and minus one is the rational number zero.

```
In[9]:= SubstTest[ratadd, inttimes[int[x]], inttimes[int[y]],
          {x → plus[set[0]], y → inverse[plus[set[0]]]}] // Reverse
```

```
Out[9]= ratadd[id[Z], composite[id[Z], INVERSE]] == cart[Z, set[id[omega]]]
```

```
In[10]:= ratadd[id[Z], composite[id[Z], INVERSE]] := cart[Z, set[id[omega]]]
```

Corollary. The sum of minus one and plus one is zero.

```
In[11]:= SubstTest[equal, ratadd[x, y], ratadd[y, x],
           {x -> composite[id[Z], INVERSE], y → id[Z]}] // Reverse
```

```
Out[11]= equal[cart[Z, set[id[omega]]], ratadd[composite[id[Z], INVERSE], id[Z]]] == True
```

```
In[12]:= ratadd[composite[id[Z], INVERSE], id[Z]] := cart[Z, set[id[omega]]]
```

Theorem. An application of the distributive law.

```
In[13]:= SubstTest[ratmul, rat[x], ratadd[u, v], {u → id[Z], v -> composite[id[Z], INVERSE]}]
```

```
Out[13]= ratadd[rat[x], composite[rat[x], INVERSE]] == cart[Z, set[id[omega]]]
```

```
In[14]:= ratadd[rat[x_], composite[rat[x_], INVERSE]] := cart[Z, set[id[omega]]]
```

Corollary.

```
In[15]:= SubstTest[ratmul, rat[x], ratadd[u, v], {u → composite[id[Z], INVERSE], v → id[Z]}]
```

```
Out[15]= ratadd[composite[rat[x], INVERSE], rat[x]] == cart[Z, set[id[omega]]]
```

```
In[16]:= ratadd[composite[rat[x_], INVERSE], rat[x_]] := cart[Z, set[id[omega]]]
```

Theorem. Membership rule for **RATADD**.

```
In[17]:= (member[pair[w, z], funpart[t]] // AssertTest) /. {w → pair[x, y], t → RATADD}
Out[17]= member[pair[pair[x, y], z], RATADD] == and[equal[z, ratadd[x, y]], member[z, V]]

In[18]:= member[pair[pair[x_, y_], z_], RATADD] := and[equal[z, ratadd[x, y]], member[z, V]]
```

---

inv[RATADD]

Theorem. Rational addition is a semigroup.

```
In[19]:= SubstTest[and, associative[x], member[x, BINOPS], x → RATADD]
Out[19]= member[RATADD, SEMIGPS] == True

In[20]:= member[RATADD, SEMIGPS] := True
```

Theorem. Inversion is a function for rational addition.

```
In[21]:= SubstTest[FUNCTION, inv[semigp[t]], t → RATADD] // Reverse
Out[21]= FUNCTION[inv[RATADD]] == True

In[22]:= FUNCTION[inv[RATADD]] := True
```

Theorem.

```
In[23]:= member[pair[rat[x], composite[rat[x], INVERSE]], inv[RATADD]] // AssertTest
Out[23]= member[pair[rat[x], composite[rat[x], INVERSE]], inv[RATADD]] == True

In[24]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[25]:= SubstTest[member, pair[rat[x], composite[rat[x], INVERSE]],
           inv[RATADD], x → id[Z]] // Reverse
Out[25]= member[pair[id[Z], composite[id[Z], INVERSE]], inv[RATADD]] == True

In[26]:= member[pair[id[Z], composite[id[Z], INVERSE]], inv[RATADD]] := True
```

Corollary.

```
In[27]:= SubstTest[member, pair[composite[id[Z], INVERSE], id[Z]],
           inverse[t], t -> inv[RATADD]] // Reverse
Out[27]= member[pair[composite[id[Z], INVERSE], id[Z]], inv[RATADD]] == True

In[28]:= member[pair[composite[id[Z], INVERSE], id[Z]], inv[RATADD]] := True
```

Lemma.

```
In[29]:= member[composite[rat[x], INVERSE], V] // AssertTest

Out[29]= member[composite[rat[x], INVERSE], V] == True

In[30]:= member[composite[rat[x_], INVERSE], V] := True
```

Theorem. A temporary **APPLY** formula for **inv[RATADD]**.

```
In[31]:= (member[pair[u, v], funpart[t]] // AssertTest // Reverse) /.
          {t → inv[RATADD], u → rat[x], v -> composite[rat[x], INVERSE]}

Out[31]= equal[APPLY[inv[RATADD], rat[x]], composite[rat[x], INVERSE]] == True

In[32]:= APPLY[inv[RATADD], rat[x_]] := composite[rat[x], INVERSE]
```

Theorem. (Removing the **rat** wrapper.)

```
In[33]:= SubstTest[implies, equal[x, rat[t]],
           equal[APPLY[inv[RATADD], x], composite[x, INVERSE]], t → x] // Reverse

Out[33]= or[equal[APPLY[inv[RATADD], x], composite[x, INVERSE]], not[member[x, RATS]]] == True

In[34]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[35]:= SubstTest[member, APPLY[t, rat[x]], V, t → inv[RATADD]]

Out[35]= member[rat[x], domain[inv[RATADD]]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Eliminating the wrapped variable.)

```
In[37]:= Map[or[subclass[RATS, domain[inv[RATADD]]], equal[V, domain[#]]] &,
           SubstTest[reify, x, case[member[rat[x], t]], t -> domain[inv[RATADD]]]]

Out[37]= subclass[RATS, domain[inv[RATADD]]] == True

In[38]:= % /. Equal → SetDelayed
```

Theorem. Rational addition is a monoid.

```
In[39]:= SubstTest[and, member[x, SEMIGPS], member[e[x], V], x → RATADD]

Out[39]= member[RATADD, MONOIDS] == True

In[40]:= member[RATADD, MONOIDS] := True
```

Corollary. Rational addition is a category.

```
In[41]:= SubstTest[category, semigp[x], x → RATADD] // Reverse

Out[41]= category[RATADD] == True
```

```
In[42]:= category[RATADD] := True
```

Theorem. An application of the preceding corollary.

```
In[43]:= SubstTest[subclass, domain[inv[cat[x]]], domain[domain[cat[x]]], x → RATADD] // Reverse
Out[43]= subclass[domain[inv[RATADD]], RATS] == True

In[44]:= % /. Equal→ SetDelayed
```

Theorem. An equation for the domain of **inv[RATADD]**.

```
In[45]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> domain[inv[RATADD]], v → RATS}]
Out[45]= equal[RATS, domain[inv[RATADD]]] == True

In[46]:= domain[inv[RATADD]] := RATS
```

Corollary. The function inv[x] is an involution.

```
In[47]:= SubstTest[composite, inv[semigp[x]], inv[semigp[x]], x → RATADD] // Reverse
Out[47]= composite[inv[RATADD], inv[RATADD]] == id[RATS]

In[48]:= composite[inv[RATADD], inv[RATADD]] := id[RATS]
```

Theorem. A wrapper-free **APPLY** formula for **inv[RATADD]**.

```
In[49]:= equal[union[complement[image[V, intersection[RATS, set[x]]]], composite[x, INVERSE]],
           APPLY[inv[RATADD], x]]
Out[49]= True

In[50]:= APPLY[inv[RATADD], x_] :=
           union[complement[image[V, intersection[RATS, set[x]]]], composite[x, INVERSE]]
```

rational addition is a group

Theorem. Rational addition is a group composition law.

```
In[63]:= SubstTest[and, member[t, MONOIDS], equal[domain[inv[t]], range[t]], t → RATADD]
Out[63]= member[RATADD, GROUPS] == True

In[64]:= member[RATADD, GROUPS] := True
```

Theorem.

```
In[66]:= SubstTest[composite, gp[t], inverse[FIRST], t → RATADD] // Reverse
Out[66]= composite[RATADD, inverse[FIRST]] == cart[RATS, RATS]
```

```
In[68]:= composite[RATADD, inverse[FIRST]] := cart[RATS, RATS]
```

Theorem.

```
In[67]:= SubstTest[composite, gp[t], inverse[SECOND], t → RATADD] // Reverse
Out[67]= composite[RATADD, inverse[SECOND]] == cart[RATS, RATS]
```

```
In[69]:= composite[RATADD, inverse[SECOND]] := cart[RATS, RATS]
```

Theorem. Rational subtraction is the rotation of rational addition.

```
In[71]:= SubstTest[rotate, gp[t], t → RATADD] // Reverse
Out[71]= rotate[RATADD] == composite[RATADD, cross[Id, inv[RATADD]]]
```

```
In[72]:= rotate[RATADD] := composite[RATADD, cross[Id, inv[RATADD]]]
```

---

explicit formulas for inv[RATADD]

Theorem. The function **inv[RATADD]** is composition with **INVERSE**, restricted to rational numbers.

```
In[52]:= Map[composite[VERTSECT[#], id[RATS]] &,
           SubstTest[reify, x, APPLY[funpart[t], x], t → inv[RATADD]]] // Reverse
Out[52]= composite[IMAGE[cross[INVERSE, Id]], id[RATS]] == inv[RATADD]
```

```
In[53]:= composite[IMAGE[cross[INVERSE, Id]], id[RATS]] := inv[RATADD]
```

Theorem. The function **inv[RATADD]** is composition with **INVERSE** in the reverse order, also restricted to rational numbers.

```
In[59]:= Map[composite[VERTSECT[#], id[RATS]] &,
          SubstTest[reify, x, composite[t, rat[x]], t → INVERSE]]
Out[59]= composite[IMAGE[cross[Id, INVERSE]], id[RATS]] == inv[RATADD]
```

```
In[60]:= composite[IMAGE[cross[Id, INVERSE]], id[RATS]] := inv[RATADD]
```

Theorem. The function **inv[RATADD]** is multiplication by **–1**.

```
In[54]:= Map[composite[VERTSECT[#], id[RATS]] &,
          SubstTest[reify, x, ratmul[rat[x], t], t -> composite[id[Z], INVERSE]]]
Out[54]= rattimes[composite[id[Z], INVERSE]] == inv[RATADD]
```

```
In[55]:= rattimes[composite[id[Z], INVERSE]] := inv[RATADD]
```

Theorem. An explicit wrapper-free formula for multiplication by minus one.

```
In[61]:= SubstTest[APPLY, rattimes[t], x, t -> composite[id[Z], INVERSE]]

Out[61]= ratmul[composite[id[Z], INVERSE], x] ==
          union[complement[image[V, intersection[RATS, set[x]]]], composite[x, INVERSE]]

In[62]:= ratmul[composite[id[Z], INVERSE], x_] :=
          union[complement[image[V, intersection[RATS, set[x]]]], composite[x, INVERSE]]
```