associative law for rational addition

Johan G. F. Belinfante
2012 October 1

```
In[1]:= SetDirectory["l:"]; << goedel.12sep29a
```

```
:Package Title: goedel.12sep29a            2012 September 29 at 6:05 p.m.

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

It is now:  2012 Oct 1 at 10:10

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 Oct 1 at 10:26
```

summary

A derivation of the associative law for rational addition is presented in this notebook.

function addition

Rational numbers are functions whose graphs are maximal straight lines through the origin in the integer plane $\mathbf{Z} \times \mathbf{Z}$. The function sum $\mathbf{z} = \mathbf{x} + \mathbf{y}$ of functions $\mathbf{x}$ and $\mathbf{y}$ in the integer plane is defined by $\mathbf{z(t) = x(t) + y(t)}$. The following temporary definition for function addition will be used in this notebook.

```
In[2]:= funadd[x_, y_] := composite[INTADD,
          intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]
```

In general the function sum of two rational numbers is a linear function, but it need not be maximal. One can always add some more points, if need be, to the graph to obtain a rational number, and this rational number is unique. In other words, the rational sum of rational numbers **rat[x]** and **rat[y]** is the rational hull of their function sum.

```
In[3]:= hull[RATS, funadd[rat[x], rat[y]]]

Out[3]= ratadd[rat[x], rat[y]]
```

If a subset $\mathbf{x} \subset \mathbf{Z} \times \mathbf{Z}$ of the integer plane is a subset of a rational number $\mathbf{y}$, and if $\mathbf{x}$ is **non-trivial** in the sense that **domain[x]** is not a subset of **{id[ω]}**, then **hull[RATS, x] = y**.

a non-triviality lemma

In this section it is shown that the intersection of the domains of three rational numbers is not a subset of **{id[ω]}**.

Observation. The domain of a rational number is a nontrivial vertical section of the integer divisibility relation.

```
In[4]:= member[domain[rat[x]], image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]]
```

```
Out[4]= True
```

The same is true for the intersection of the domains of two rational numbers.

```
In[5]:= member[intersection[domain[rat[x]], domain[rat[y]]],
          image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]]
```

```
Out[5]= True
```

Theorem. The intersection of the domains of three rational numbers is a nontrivial vertical section of the integer divisibility relation.

```
In[6]:= SubstTest[implies, and[member[u, image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]],
            member[v, image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]],
           member[intersection[u, v], image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]],
           {u → intersection[domain[rat[x]], domain[rat[y]]], v → domain[rat[z]]}] // Reverse
```

```
Out[6]= member[intersection[domain[rat[x]], domain[rat[y]], domain[rat[z]]],
          image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]] == True
```

```
In[7]:= member[intersection[domain[rat[x_]], domain[rat[y_]], domain[rat[z_]]],
          image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]] := True
```

Corollary. The intersection of the domains of three rational numbers is not a subset of **{id[ω]}**.

```
In[8]:= Map[not, SubstTest[implies, member[t, image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]],
            not[subclass[t, set[id[omega]]]],
            t -> intersection[domain[rat[x]], domain[rat[y]], domain[rat[z]]]]] // Reverse
```

```
Out[8]= subclass[
          intersection[domain[rat[x]], domain[rat[y]], domain[rat[z]]], set[id[omega]]] == False
```

```
In[9]:= subclass[intersection[domain[rat[x_]], domain[rat[y_]], domain[rat[z_]]],
          set[id[omega]]] := False
```

associativity of function addition

Observation. Function addition is associative.

```
In[10]:= associative[composite[IMAGE[cross[inverse[DUP], INTADD]], CROSS]]

Out[10]= True
```

The following theorem introduces three variables wrapped with **rat** into the associative law for function addition.

Theorem. Function addition is associative.

```
In[11]:= Map[APPLY[#, PAIR[PAIR[rat[x], rat[y]], rat[z]]] &,
           SubstTest[composite, assoc[t], cross[Id, assoc[t]], ASSOC,
            t -> composite[IMAGE[cross[inverse[DUP], INTADD]], CROSS]]] // Reverse

Out[11]= composite[INTADD, intersection[composite[inverse[FIRST], rat[x]],
           composite[inverse[SECOND], INTADD, intersection[
             composite[inverse[FIRST], rat[y]], composite[inverse[SECOND], rat[z]]]]]] ==
         composite[INTADD, intersection[composite[inverse[SECOND], rat[z]],
           composite[inverse[FIRST], INTADD, intersection[
             composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]]
```

```
In[12]:= composite[INTADD, intersection[composite[inverse[FIRST], rat[x_]],
           composite[inverse[SECOND], INTADD, intersection[
             composite[inverse[FIRST], rat[y_]], composite[inverse[SECOND], rat[z_]]]]]] :=
         composite[INTADD, intersection[composite[inverse[SECOND], rat[z]],
           composite[inverse[FIRST], INTADD, intersection[
             composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]]
```

Restatement.

```
In[13]:= equal[funadd[rat[x], funadd[rat[y], rat[z]]], funadd[funadd[rat[x], rat[y]], rat[z]]]

Out[13]= True
```

The strategy now is to show that both sides of the associative law for rational addition are equal to the rational hull of this set.

---

one side of the associative law

Observation. The following statement is automatically recognized to be true by existing rewrite rules.

```
In[14]:= subclass[funadd[funadd[rat[x], rat[y]], rat[z]],
          funadd[ratadd[rat[x], rat[y]], rat[z]]]

Out[14]= True
```

Lemma.

```
In[15]:= SubstTest[subclass, funadd[rat[t], rat[z]],
           ratadd[rat[t], rat[z]], t → ratadd[rat[x], rat[y]]] // Reverse

Out[15]= subclass[
           composite[INTADD, intersection[composite[inverse[FIRST], ratadd[rat[x], rat[y]]],
             composite[inverse[SECOND], rat[z]]]],
           ratadd[ratadd[rat[x], rat[y]], rat[z]]] == True

In[16]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[17]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → funadd[funadd[rat[x], rat[y]], rat[z]],
            v -> funadd[ratadd[rat[x], rat[y]], rat[z]],
            w → ratadd[ratadd[rat[x], rat[y]], rat[z]]}] // Reverse

Out[17]= subclass[composite[INTADD, intersection[
             composite[inverse[SECOND], rat[z]], composite[inverse[FIRST], INTADD, intersection[
               composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]],
           ratadd[ratadd[rat[x], rat[y]], rat[z]]] == True

In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. An equation for one side of the associative law for rational addition.

```
In[19]:= SubstTest[implies,
           and[subclass[u, v], not[subclass[domain[u], set[id[omega]]]], member[v, RATS]],
           equal[hull[RATS, u], v], {u -> funadd[funadd[rat[x], rat[y]], rat[z]],
            v -> ratadd[ratadd[rat[x], rat[y]], rat[z]]}] // Reverse

Out[19]= equal[hull[RATS, composite[INTADD, intersection[composite[inverse[SECOND], rat[z]],
             composite[inverse[FIRST], INTADD, intersection[
               composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]]],
           ratadd[ratadd[rat[x], rat[y]], rat[z]]] == True

In[20]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

the other side of the associative law

Lemma.

```
In[21]:= SubstTest[implies, and[subclass[t, v], subclass[u, w]],
           subclass[funadd[t, u], funadd[v, w]], {t → rat[x], v → rat[x],
            u -> funadd[rat[y], rat[z]], w → ratadd[rat[y], rat[z]]}] // Reverse

Out[21]= subclass[composite[INTADD, intersection[
             composite[inverse[SECOND], rat[z]], composite[inverse[FIRST], INTADD, intersection[
               composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]],
           composite[INTADD, intersection[composite[inverse[FIRST], rat[x]],
             composite[inverse[SECOND], ratadd[rat[y], rat[z]]]]]] == True
```

```
In[22]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[23]:= SubstTest[subclass, funadd[rat[x], rat[t]],
           ratadd[rat[x], rat[t]], t → ratadd[rat[y], rat[z]]] // Reverse

Out[23]= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x]],
             composite[inverse[SECOND], ratadd[rat[y], rat[z]]]]],
           ratadd[rat[x], ratadd[rat[y], rat[z]]]] == True
```

```
In[24]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[25]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → funadd[rat[x], funadd[rat[y], rat[z]]],
            v -> funadd[rat[x], ratadd[rat[y], rat[z]]],
            w → ratadd[rat[x], ratadd[rat[y], rat[z]]]}] // Reverse

Out[25]= subclass[composite[INTADD, intersection[
             composite[inverse[SECOND], rat[z]], composite[inverse[FIRST], INTADD, intersection[
               composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]],
           ratadd[rat[x], ratadd[rat[y], rat[z]]]] == True
```

```
In[26]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. An equation for the other side of the associative law for rational addition.

```
In[27]:= SubstTest[implies,
           and[subclass[u, v], not[subclass[domain[u], set[id[omega]]]], member[v, RATS]],
           equal[hull[RATS, u], v], {u -> funadd[rat[x], funadd[rat[y], rat[z]]],
            v -> ratadd[rat[x], ratadd[rat[y], rat[z]]]}] // Reverse

Out[27]= equal[hull[RATS, composite[INTADD, intersection[composite[inverse[SECOND], rat[z]],
             composite[inverse[FIRST], INTADD, intersection[
               composite[inverse[FIRST], rat[x]], composite[inverse[SECOND], rat[y]]]]]]],
           ratadd[rat[x], ratadd[rat[y], rat[z]]]] == True
```

```
In[28]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

the associative law

The results of the preceding two sections can be combined to obtain the following version of the associative law for rational addition.

Theorem. The associative law with wrapped variables.

```
In[29]:= SubstTest[implies, and[equal[u, v], equal[v, w]],
           equal[u, w], {u -> ratadd[rat[x], ratadd[rat[y], rat[z]]],
            v → hull[RATS, funadd[funadd[rat[x], rat[y]], rat[z]]],
            w → ratadd[ratadd[rat[x], rat[y]], rat[z]]}] // Reverse

Out[29]= equal[ratadd[rat[x], ratadd[rat[y], rat[z]]],
           ratadd[ratadd[rat[x], rat[y]], rat[z]]] == True

In[30]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

A better result will ne derived that does not require the variables to be wrapped.

Theorem. (Eliminate the wrappers.)

```
In[31]:= SubstTest[implies, and[equal[x, rat[u]], equal[y, rat[v]], equal[z, rat[w]]],
           equal[ratadd[x, ratadd[y, z]], ratadd[ratadd[x, y], z]], {u → x, v → y, w → z}] // Reverse

Out[31]= or[equal[ratadd[x, ratadd[y, z]], ratadd[ratadd[x, y], z]],
           not[member[x, RATS]], not[member[y, RATS]], not[member[z, RATS]]] == True

In[32]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The above result has three redundant literals.

Theorem. (The case of non-rational numbers.)

```
In[33]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
           {u -> ratadd[x, ratadd[y, z]], v → V, w -> ratadd[ratadd[x, y], z]}] // Reverse

Out[33]= or[and[member[x, RATS], member[y, RATS], member[z, RATS]],
           equal[ratadd[x, ratadd[y, z]], ratadd[ratadd[x, y], z]]] == True

In[34]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The three redundant literals can now be eliminated.

Theorem. The associative law for rational number addition.

```
In[35]:= SubstTest[and, implies[p, q], or[p, q],
          {p -> and[member[x, RATS], member[y, RATS], member[z, RATS]],
           q -> equal[ratadd[x, ratadd[y, z]], ratadd[ratadd[x, y], z]]}]

Out[35]= equal[ratadd[x, ratadd[y, z]], ratadd[ratadd[x, y], z]] == True

In[36]:= ratadd[x_, ratadd[y_, z_]] := ratadd[ratadd[x, y], z]
```

eliminating the variables

A variable-free formulation of the associative law for rational addition can be derived using reification.

Lemma. A simplification rule.

```
In[37]:= composite[RATADD, cross[RATADD, id[RATS]]] // TripleRotate

Out[37]= composite[RATADD, cross[RATADD, id[RATS]]] == composite[RATADD, cross[RATADD, Id]]

In[38]:= composite[RATADD, cross[RATADD, id[RATS]]] := composite[RATADD, cross[RATADD, Id]]
```

Theorem. A variable-free statement of the associative law.

```
In[39]:= Map[VERTSECT, SubstTest[reify, x, APPLY[funpart[t], PAIR[first[first[x]],
             APPLY[funpart[t], PAIR[second[first[x]], second[x]]]]], t → RATADD]]

Out[39]= composite[RATADD, cross[Id, RATADD], ASSOC] == composite[RATADD, cross[RATADD, Id]]

In[40]:= composite[RATADD, cross[Id, RATADD], ASSOC] := composite[RATADD, cross[RATADD, Id]]
```

Theorem. Restatement.

```
In[41]:= associative[RATADD] // AssertTest

Out[41]= associative[RATADD] == True

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