multiplication of rational numbers

Johan G. F. Belinfante
2012 August 11

```
In[1]:= SetDirectory["l:"]; << goedel.12aug10a
```

```
:Package Title: goedel.12aug10a                2012 August 10 at 6:10 p.m.

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

It is now:  2012 Aug 11 at 8:4

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 11 at 8:20
```

summary

The product of two rational numbers is defined to be the rational hull of their composite. An explicit formula for the rational hull of the composite of two rational numbers is derived and used to show that multiplication of rational numbers is a binary operation.

definition of RATMUL

Definition of **RATMUL**.

```
In[2]:= composite[HULL[RATS], COMPOSE, id[cart[RATS, RATS]]] := RATMUL
```

Theorem.

```
In[3]:= SubstTest[FUNCTION,
          composite[HULL[RATS], COMPOSE, id[t]], t -> cart[RATS, RATS]] // Reverse
```

```
Out[3]= FUNCTION[RATMUL] == True
```

```
In[4]:= FUNCTION[RATMUL] := True
```

Theorem.

```
In[5]:= Assoc[composite[HULL[RATS], COMPOSE], id[cart[RATS, RATS]], id[cart[V, V]]] // Reverse

Out[5]= composite[RATMUL, id[cart[V, V]]] == RATMUL

In[6]:= composite[RATMUL, id[cart[V, V]]] := RATMUL
```

It will be shown below that **RATMUL** is a binary operation on the set **RATS**.

---

definition of ratmul

Definition. The binary constructor **ratmul** is defined as follows.

```
In[7]:= APPLY[RATMUL, PAIR[x_, y_]] := ratmul[x, y]
```

Theorem. Vertical section rule.

```
In[8]:= SubstTest[image, funpart[t], set[PAIR[x, y]], t → RATMUL] // Reverse

Out[8]= image[RATMUL, cart[set[x], set[y]]] == set[ratmul[x, y]]

In[9]:= image[RATMUL, cart[set[x_], set[y_]]] := set[ratmul[x, y]]
```

Lemma.

```
In[10]:= ApComp[COMPOSE, id[cart[RATS, RATS]], PAIR[x, y]] // Reverse

Out[10]= A[image[COMPOSE, cart[intersection[RATS, set[x]], intersection[RATS, set[y]]]]] ==
          union[complement[image[V, intersection[RATS, set[x]]]],
           complement[image[V, intersection[RATS, set[y]]]], composite[x, y]]

In[11]:= A[image[COMPOSE, cart[intersection[RATS, set[x_]], intersection[RATS, set[y_]]]]] :=
          union[complement[image[V, intersection[RATS, set[x]]]],
           complement[image[V, intersection[RATS, set[y]]]], composite[x, y]]
```

Lemma. Simplification rule.

```
In[12]:= ApComp[HULL[RATS], composite[COMPOSE, id[cart[RATS, RATS]]], PAIR[x, y]]

Out[12]= union[complement[image[V, intersection[RATS, set[x]]]],
           complement[image[V, intersection[RATS, set[y]]]],
           hull[RATS, composite[x, y]]] == ratmul[x, y]

In[13]:= union[complement[image[V, intersection[RATS, set[x_]]]],
           complement[image[V, intersection[RATS, set[y_]]]],
           hull[RATS, composite[x_, y_]]] := ratmul[x, y]
```

Theorem. If **x** and **y** are rational numbers, then **ratmul[x, y] = hull[RATS, x ∘ y]**.

```
In[14]:= Map[implies[and[member[x, RATS], member[y, RATS]], #] &,
           SubstTest[implies, and[equal[u, union[u, v]], equal[union[u, v], w]],
            equal[u, w], {u -> hull[RATS, composite[x, y]],
             v -> union[complement[image[V, intersection[RATS, set[x]]]], complement[image[V,
                 intersection[RATS, set[y]]]]], w → ratmul[x, y]}]] // Reverse // MapNotNot

Out[14]= or[equal[hull[RATS, composite[x, y]], ratmul[x, y]],
           not[member[x, RATS]], not[member[y, RATS]]] == True

In[15]:= or[equal[hull[RATS, composite[x_, y_]], ratmul[x_, y_]],
           not[member[x_, RATS]], not[member[y_, RATS]]] := True
```

---

domain of RATMUL

Lemma.

```
In[16]:= SubstTest[or, equal[hull[RATS, composite[x, y]], ratmul[x, y]],
           not[member[x, RATS]], not[member[y, RATS]], y → id[Z]] // Reverse

Out[16]= or[equal[hull[RATS, composite[x, id[Z]]], ratmul[x, id[Z]]],
           not[member[x, RATS]]] == True

In[17]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[18]:= SubstTest[implies, equal[u, v],
           equal[hull[RATS, u], hull[RATS, v]], {u → composite[x, id[Z]], v → x}] // Reverse

Out[18]= or[equal[hull[RATS, x], hull[RATS, composite[x, id[Z]]]],
           not[subclass[x, cart[Z, V]]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If **x** is a rational number, then **ratmul[x, id[Z]] = x**.

```
In[20]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            implies[p3, p4], implies[p1, p5], implies[p1, p6], not[implies[p1, p7]],
            {p1 → member[x, RATS], p2 → subclass[x, cart[Z, Z]], p3 -> subclass[x, cart[Z, V]],
             p4 -> equal[hull[RATS, x], hull[RATS, composite[x, id[Z]]]],
             p5 -> equal[hull[RATS, x], x],
             p6 -> equal[hull[RATS, composite[x, id[Z]]], ratmul[x, id[Z]]],
             p7 → equal[ratmul[x, id[Z]], x]}]] // Reverse

Out[20]= or[equal[x, ratmul[x, id[Z]]], not[member[x, RATS]]] == True

In[21]:= or[equal[x_, ratmul[x_, id[Z]]], not[member[x_, RATS]]] := True
```

Theorem. The domain of **RATMUL** is the cartesian square **RATS** × **RATS**.

```
In[22]:= Map[equal[#, cart[RATS, RATS]] &,
          IminComp[composite[HULL[RATS], COMPOSE], id[cart[RATS, RATS]], V]]

Out[22]= equal[cart[RATS, RATS], domain[RATMUL]] == True

In[23]:= domain[RATMUL] := cart[RATS, RATS]
```

lower bound for the range

Lemma.

```
In[24]:= SubstTest[member, APPLY[t, PAIR[x, y]], V, t → RATMUL] // Reverse

Out[24]= member[ratmul[x, y], V] == and[member[x, RATS], member[y, RATS]]

In[25]:= member[ratmul[x_, y_], V] := and[member[x, RATS], member[y, RATS]]
```

Lemma.

```
In[26]:= Map[empty[domain[complement[#]]] &,
          SubstTest[reify, x, case[implies[member[x, RATS], equal[APPLY[funpart[t], x], x]]],
           t -> composite[RATMUL, RIGHT[id[Z]]]]]

Out[26]= subclass[RATS,
           image[inverse[fix[composite[inverse[FIRST], RATMUL]]], set[id[Z]]]] == True

In[27]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[28]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → RATS, v -> image[inverse[fix[composite[inverse[FIRST], RATMUL]]], set[id[Z]]],
            w → range[RATMUL]}] // Reverse

Out[28]= subclass[RATS, range[RATMUL]] == True

In[29]:= % /. Equal→ SetDelayed
```

an explicit formula for ratmul

In this section an explicit formula for the product of two rational numbers is derived. It is suggestive to write fractions as **d \ n** instead of **n / d**. To this end, the following temporary abbreviation is introduced.

```
In[30]:= frac[x_, y_] := composite[inverse[inttimes[x]], inttimes[y]]
```

Lemma. A point on the composite of two fractions.

```
In[31]:= SubstTest[implies,
           and[member[pair[p, q], composite[Id, t]], member[pair[q, r], composite[Id, s]]],
           member[pair[p, r], composite[s, t]], {p → intmul[u, x], q → intmul[v, x],
            r → intmul[v, y], s → frac[x, y], t → frac[u, v]}] // Reverse

Out[31]= or[member[pair[intmul[u, x], intmul[v, y]],
            composite[inverse[inttimes[x]], inttimes[y], inverse[inttimes[u]], inttimes[v]]],
           not[member[u, Z]], not[member[v, Z]], not[member[x, Z]], not[member[y, Z]]] == True

In[32]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. An application of a theorem about rational hulls.

```
In[33]:= SubstTest[or, equal[s, id[omega]], equal[q, hull[RATS, p]],
           not[member[q, RATS]], not[member[pair[s, t], p]],
           not[subclass[p, q]], {p → composite[frac[u, v], frac[x, y]],
            q → frac[intmul[u, x], intmul[v, y]], s → intmul[u, x], t → intmul[v, y]}] // Reverse

Out[33]= or[equal[u, id[omega]], equal[x, id[omega]],
           equal[composite[inverse[inttimes[intmul[u, x]]], inttimes[intmul[v, y]]], hull[RATS,
             composite[inverse[inttimes[u]], inttimes[v], inverse[inttimes[x]], inttimes[y]]]],
           not[member[u, Z]], not[member[v, Z]], not[member[x, Z]], not[member[y, Z]],
           not[member[pair[intmul[u, x], intmul[v, y]], composite[inverse[inttimes[u]],
              inttimes[v], inverse[inttimes[x]], inttimes[y]]]]] == True

In[34]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. An explicit formula for the rational hull of the composite of two fractions.

```
In[35]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → and[member[pair[u, v], domain[RATIO]], member[pair[x, y], domain[RATIO]]],
            p2 -> member[pair[intmul[u, x], intmul[v, y]], composite[
               inverse[inttimes[u]], inttimes[v], inverse[inttimes[x]], inttimes[y]]],
            p3 -> equal[composite[inverse[inttimes[intmul[u, x]]], inttimes[intmul[v, y]]],
              hull[RATS, composite[inverse[inttimes[u]], inttimes[v],
                inverse[inttimes[x]], inttimes[y]]]]}]] // Reverse

Out[35]= or[equal[u, id[omega]], equal[x, id[omega]],
           equal[composite[inverse[inttimes[intmul[u, x]]], inttimes[intmul[v, y]]], hull[RATS,
             composite[inverse[inttimes[u]], inttimes[v], inverse[inttimes[x]], inttimes[y]]]],
           not[member[u, Z]], not[member[v, Z]], not[member[x, Z]], not[member[y, Z]]] == True

In[37]:= or[equal[composite[inverse[inttimes[intmul[u_, x_]]], inttimes[intmul[v_, y_]]],
            hull[RATS, composite[inverse[inttimes[u_]], inttimes[v_],
              inverse[inttimes[x_]], inttimes[y_]]]], equal[id[omega], u_],
           equal[id[omega], x_], not[member[u_, Z]], not[member[v_, Z]],
           not[member[x_, Z]], not[member[y_, Z]]] := True
```

Corollary. (Introducing **int** wrappers.)

```
In[38]:= SubstTest[or, equal[p, id[omega]], equal[r, id[omega]],
           equal[composite[inverse[inttimes[intmul[p, r]]], inttimes[intmul[q, s]]], hull[RATS,
             composite[inverse[inttimes[p]], inttimes[q], inverse[inttimes[r]], inttimes[s]]]],
           not[member[p, Z]], not[member[q, Z]], not[member[r, Z]], not[member[s, Z]],
           {p → int[u], q → int[v], r → int[x], s → int[y]}] // Reverse

Out[38]= or[equal[composite[inverse[inttimes[intmul[int[u], int[x]]]],
           inttimes[intmul[int[v], int[y]]]], hull[RATS, composite[inverse[inttimes[int[u]]],
            inttimes[int[v]], inverse[inttimes[int[x]]], inttimes[int[y]]]]],
          equal[id[omega], int[u]], equal[id[omega], int[x]]] == True

In[39]:= or[equal[composite[inverse[inttimes[intmul[int[u_], int[x_]]]],
           inttimes[intmul[int[v_], int[y_]]]],
          hull[RATS, composite[inverse[inttimes[int[u_]]], inttimes[int[v_]],
            inverse[inttimes[int[x_]]], inttimes[int[y_]]]]],
          equal[id[omega], int[u_]], equal[id[omega], int[x_]]] := True
```

range of RATMUL

From the explicit formula for the rational hull of the composite of two rationals, one derives the following corollary.

Corollary. The rational hull of the composite of two rationals is a rational.

```
In[40]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> and[member[pair[u, v], domain[RATIO]], member[pair[x, y], domain[RATIO]]],
            p2 → equal[composite[inverse[inttimes[intmul[u, x]]], inttimes[intmul[v, y]]],
              hull[RATS, composite[inverse[inttimes[u]],
                inttimes[v], inverse[inttimes[x]], inttimes[y]]]],
            p3 → member[hull[RATS, composite[inverse[inttimes[u]], inttimes[v],
                inverse[inttimes[x]], inttimes[y]]], RATS]}]] // Reverse

Out[40]= or[equal[u, id[omega]], equal[x, id[omega]],
          member[hull[RATS, composite[inverse[inttimes[u]], inttimes[v],
             inverse[inttimes[x]], inttimes[y]]], RATS], not[member[u, Z]],
          not[member[v, Z]], not[member[x, Z]], not[member[y, Z]]] == True

In[41]:= or[equal[id[omega], u_], equal[id[omega], x_],
          member[hull[RATS, composite[inverse[inttimes[u_]], inttimes[v_],
             inverse[inttimes[x_]], inttimes[y_]]], RATS], not[member[u_, Z]],
          not[member[v_, Z]], not[member[x_, Z]], not[member[y_, Z]]] := True
```

Corollary.

```
In[42]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p1, p2, p3], p4],
           not[implies[p1, p4]], {p1 -> and[member[w, domain[RATIO]], member[z, domain[RATIO]]],
            p2 → equal[APPLY[RATIO, w], frac[first[w], second[w]]],
            p3 → equal[APPLY[RATIO, z], frac[first[z], second[z]]], p4 →
             member[hull[RATS, composite[APPLY[RATIO, w], APPLY[RATIO, z]]], RATS]}]] // Reverse
```

```
Out[42]= or[equal[first[w], id[omega]], equal[first[z], id[omega]],
           member[hull[RATS, composite[APPLY[RATIO, w], APPLY[RATIO, z]]], RATS],
           not[member[first[w], Z]], not[member[first[z], Z]],
           not[member[second[w], Z]], not[member[second[z], Z]]] == True
```

```
In[43]:= (% /. {w → w_, z → z_}) /. Equal→ SetDelayed
```

Lemma

```
In[45]:= Assoc[composite[HULL[RATS], COMPOSE], id[cart[RATS, RATS]], cross[RATIO, RATIO]]
```

```
Out[45]= composite[HULL[RATS], COMPOSE, cross[RATIO, RATIO]] ==
          composite[RATMUL, cross[RATIO, RATIO]]
```

```
In[46]:= composite[HULL[RATS], COMPOSE, cross[RATIO, RATIO]] :=
          composite[RATMUL, cross[RATIO, RATIO]]
```

Lemma.

```
In[53]:= Map[implies[
            and[member[w, domain[RATIO]], member[z, domain[RATIO]]], member[pair[w, z], #]] &,
          IminComp[composite[HULL[RATS], COMPOSE], cross[RATIO, RATIO], RATS]]
```

```
Out[53]= or[equal[first[w], id[omega]], equal[first[z], id[omega]],
           member[pair[APPLY[RATIO, w], APPLY[RATIO, z]], image[inverse[RATMUL], RATS]],
           not[member[first[w], Z]], not[member[first[z], Z]],
           not[member[second[w], Z]], not[member[second[z], Z]]] == True
```

```
In[54]:= (% /. {w → w_, z → z_}) /. Equal→ SetDelayed
```

Lemma.

```
In[56]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[w, z],
           implies[and[member[w, u], member[z, u]], member[pair[w, z], v]], {u → domain[RATIO],
            v -> composite[inverse[RATIO], image[inverse[RATMUL], RATS], RATIO]}]]
```

```
Out[56]= subclass[range[RATMUL], RATS] == True
```

```
In[57]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[58]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → range[RATMUL], v → RATS}]
```

```
Out[58]= equal[RATS, range[RATMUL]] == True
```

```
In[60]:= range[RATMUL] := RATS
```

Lemma. The function **RATMUL** is a set.

```
In[62]:= SubstTest[member, domain[funpart[t]], V, t → RATMUL]

Out[62]= member[RATMUL, V] == True

In[63]:= member[RATMUL, V] := True
```

Corollary.

```
In[64]:= member[RATMUL, map[cart[RATS, RATS], RATS]] // AssertTest

Out[64]= member[RATMUL, map[cart[RATS, RATS], RATS]] == True

In[65]:= member[RATMUL, map[cart[RATS, RATS], RATS]] := True
```

Corollary.

```
In[68]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
           {u → RATMUL, v -> map[cart[RATS, RATS], RATS], w → BINOPS}] // Reverse

Out[68]= member[RATMUL, BINOPS] == True

In[69]:= member[RATMUL, BINOPS] := True
```