rational multiplication is commutative

Johan G. F. Belinfante
2012 August 16

\texttt{In[1]}:= \texttt{SetDirectory["1:"]; \texttt{\textless \textless goedel.12aug11a}}

:Package Title: goedel.12aug11a 2012 August 11 at 8:25 p.m.
Loading takes about sixteen minutes, half that time due to builtin pauses.
It is now: 2012 Aug 16 at 14:48
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 16 at 15:3

\textbf{summary}

An explicit formula for the product of two rational number is used to show that rational multiplication is commutative. It is also shown that \texttt{id[Z]} is the neutral element for rational multiplication.

\textbf{temporary abbreviations}

It is suggestive to write fractions as \texttt{d\ n} instead of \texttt{n/d}. The following temporary abbreviation is introduced for the fraction \texttt{x/y}.

\texttt{In[2]}:= \texttt{frac[x\_, y\_] := composite[inverse[inttimes[x]], inttimes[y]]}

In addition, the following temporary abbreviation saves some writing.

\texttt{In[3]}:= \texttt{rat[x\_] := frac[first[x], second[x]]}
 derivation

Multiplication of fractions involves two numerators and two denominators. Each rational number is a straight line through the origin in the integer plane \( \mathbb{Z} \times \mathbb{Z} \). To reduce the number of variables, it is convenient to introduce just one variable for each fraction, representing some point \( x \) other than the origin on the straight line for that fraction. The denominator is then written as \( \text{first}[x] \), and the numerator as \( \text{second}[x] \). The fraction itself can be written as \( \text{APPLY}[\text{RATIO}, x] \) or as \( \text{rat}[x] \). These are equal only when \( x \in \text{domain}[\text{RATIO}] \). The following two lemmas reflect the definition of rational multiplication, using each of these two ways of writing a fraction.

Lemma. (Rewriting the hull of a composite as a rational product.)

\[
\text{In}[4] := \text{SubstTest}[	ext{implies, and}[	ext{member}[u, \text{RATS}], \text{member}[v, \text{RATS}]],
\text{equal}[	ext{hull}[\text{RATS}, \text{composite}[u, v]], \text{ratmul}[u, v]],
\{u \rightarrow \text{APPLY}[\text{RATIO}, x], v \rightarrow \text{APPLY}[\text{RATIO}, y]\}] // \text{Reverse}
\]

\[
\text{Out}[4] = \text{or}[\text{equal}[	ext{first}[x], \text{id}[	ext{omega}]], \text{equal}[	ext{first}[y], \text{id}[	ext{omega}]],
\text{equal}[	ext{hull}[\text{RATS}, \text{composite}[	ext{APPLY}[\text{RATIO}, x], \text{APPLY}[\text{RATIO}, y]]],
\text{ratmul}[	ext{APPLY}[\text{RATIO}, x], \text{APPLY}[\text{RATIO}, y]],
\text{not}[	ext{member}[	ext{first}[x], \mathbb{Z}]],
\text{not}[	ext{member}[	ext{first}[y], \mathbb{Z}], \text{not}[	ext{member}[	ext{second}[x], \mathbb{Z}], \text{not}[	ext{member}[	ext{second}[y], \mathbb{Z}]]) = \text{True}
\]

\[
\text{In}[5] := (\% /. \{x \rightarrow x_, y \rightarrow y_\}) /. \text{Equal} \rightarrow \text{SetDelayed}
\]

Lemma. (Rewriting the hull of a composite as a rational product.)

\[
\text{In}[6] := \text{SubstTest}[	ext{implies, and}[	ext{member}[u, \text{RATS}], \text{member}[v, \text{RATS}]],
\text{equal}[	ext{hull}[\text{RATS}, \text{composite}[u, v]], \text{ratmul}[u, v]], \{u \rightarrow \text{rat}[x], v \rightarrow \text{rat}[y]\}] // \text{Reverse}
\]

\[
\text{Out}[6] = \text{or}[\text{equal}[	ext{first}[x], \text{id}[	ext{omega}]], \text{equal}[	ext{first}[y], \text{id}[	ext{omega}]],
\text{equal}[	ext{hull}[\text{RATS}, \text{composite}[	ext{inverse}[	ext{inttimes}[	ext{first}[x]]],
\text{inttimes}[	ext{second}[x]], \text{inttimes}[	ext{second}[y]]],
\text{ratmul}[	ext{composite}[	ext{inverse}[	ext{inttimes}[	ext{first}[y]]],
\text{inttimes}[	ext{second}[x]]],
\text{composite}[	ext{inverse}[	ext{inttimes}[	ext{first}[y]],
\text{inttimes}[	ext{second}[y]]],
\text{not}[	ext{member}[	ext{first}[x], \mathbb{Z}], \text{not}[	ext{member}[	ext{first}[y], \mathbb{Z}],
\text{not}[	ext{member}[	ext{second}[x], \mathbb{Z}], \text{not}[	ext{member}[	ext{second}[y], \mathbb{Z}]]) = \text{True}
\]

\[
\text{In}[7] := (\% /. \{x \rightarrow x_, y \rightarrow y_\}) /. \text{Equal} \rightarrow \text{SetDelayed}
\]

The following result relates rational multiplication to integer multiplication. It says that \( (a \backslash b) \cdot (c \backslash d) = (a \cdot c) \backslash (b \cdot d) \).

Theorem. An explicit formula for the product of two rational numbers.
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
not[implies[p1, p4]], {p1 -> and[member[x, domain[RATIO]], member[y, domain[RATIO]]],
p2 -> equal[hull[RATS, composite[rat[x], rat[y]]],
frac[intmul[first[x], first[y]], intmul[second[x], second[y]]]],
p3 -> equal[hull[RATS, composite[rat[x], rat[y]]], ratmul[rat[x], rat[y]]],
p4 -> equal[ratmul[rat[x], rat[y]],
frac[intmul[first[x], first[y]], intmul[second[x], second[y]]]]], // Reverse
Out[8]= or[equal[composite[inverse[inttimes[intmul[first[x], first[y]]]]],
inttimes[intmul[second[x], second[y]]]],
ratmul[composite[inverse[inttimes[first[x]]]], inttimes[second[x]]],
composite[inverse[inttimes[first[y]]], inttimes[second[y]]]],
equal[first[x], id[omega]], equal[first[y], id[omega]],
not[member[first[x], Z]], not[member[first[y], Z]],
not[member[second[x], Z]], not[member[second[y], Z]]] = True

In[9]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

The following corollary restates theorem using APPLY[RATIO, x] instead of rat[x] for each fraction. This makes it easier to later to eliminate the variables.

Corollary. Another explicit formula for the product of two rational numbers.

In[10]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
implies[p1, p4], implies[and[p2, p3, p4], p5], not[implies[p1, p5]],
{p1 -> and[member[x, domain[RATIO]], member[y, domain[RATIO]]],
p2 -> equal[ratmul[rat[x], rat[y]],
frac[intmul[first[x], first[y]], intmul[second[x], second[y]]]],
p3 -> equal[APPLY[RATIO, x], rat[x]], p4 -> equal[APPLY[RATIO, y], rat[y]],
p5 -> equal[ratmul[APPLY[RATIO, x], APPLY[RATIO, y]],
frac[intmul[first[x], first[y]], intmul[second[x], second[y]]]]], // Reverse
Out[10]= or[equal[composite[inverse[inttimes[intmul[first[x], first[y]]]]],
inttimes[intmul[second[x], second[y]]]], ratmul[APPLY[RATIO, x], APPLY[RATIO, y]],
equal[first[x], id[omega]], equal[first[y], id[omega]], not[member[first[x], Z]],
not[member[first[y], Z]], not[member[second[x], Z]], not[member[second[y], Z]]] = True

In[11]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

It is possible, but difficult, to eliminate the variables x and y in this result. This will be done in a separate notebook. For now, a corollary will be derived that does not involve the constructor intmul. It depends only on the fact that integer multiplication is commutative.

Theorem. A form of the commutative law for multiplying fractions that involves two variables.
Normality

In[12]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
   not[implies[p1, p4]], (p1 -> and[member[x, domain[RATIO]], member[y, domain[RATIO]]],
   p2 -> equal[ratmul[APPLY[RATIO, x], APPLY[RATIO, y]],
   frac[intmul[first[x], first[y]], intmul[second[x], second[y]]]],
   p3 -> equal[ratmul[APPLY[RATIO, y], APPLY[RATIO, x]],
   frac[intmul[first[x], first[y]], intmul[second[x], second[y]]]],
   p4 -> equal[ratmul[APPLY[RATIO, x], APPLY[RATIO, y]],
   ratmul[APPLY[RATIO, y], APPLY[RATIO, x]]]]] // Reverse

Out[12]= or[equal[first[x], id[omega]], equal[first[y], id[omega]],
   equal[ratmul[APPLY[RATIO, x], APPLY[RATIO, y]],
   ratmul[APPLY[RATIO, y], APPLY[RATIO, x]]], not[member[first[x], Z]],
   not[member[first[y], Z]], not[member[second[x], Z]], not[member[second[y], Z]]] = True

In[13]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

Theorem. A simplification rule. (The rational product of two classes is a set exactly when both classes are rational numbers.)

In[14]:= image[V, set[ratmul[x, y]]] // Normality

Out[14]= image[V, set[ratmul[x, y]]] = intersection[
   image[V, intersection[RATS, set[x]]], image[V, intersection[RATS, set[y]]]]

In[15]:= image[V, set[ratmul[x_, y_]]] = intersection{
   image[V, intersection[RATS, set[x]]], image[V, intersection[RATS, set[y]]]]

Theorem. A simplification rule. (The fraction APPLY[RATIO, x] is a rational number exactly when x ∈ domain[RATIO].)

In[16]:= SubstTest[case, member[t, RATS], t -> APPLY[RATIO, x]]

Out[16]= image[V, intersection[RATS, set[APPLY[RATIO, x]]]] =
   case[and[member[first[x], Z], member[second[x], Z], not[equal[first[x], id[omega]]]]]

In[17]:= image[V, intersection[RATS, set[APPLY[RATIO, x_]]]] =
   case[and[member[first[x], Z], member[second[x], Z], not[equal[first[x], id[omega]]]]]

Lemma. (Eliminating both variables at the same time. This takes a while.)

In[18]:= Map[equal[V, domain[#]] &,
   SubstTest[reify, x, case[implies[member[x, u], equal[APPLY[funpart[v],
   PAIR[APPLY[funpart[w], first[x]], APPLY[funpart[w], second[x]]]],
   APPLY[funpart[v], PAIR[APPLY[funpart[w], second[x]], APPLY[funpart[w], first[x]]]]]],
   {u -> cartsq[domain[RATIO]], v -> RATMUL, w -> RATIO}]]

   cart[intersection[Z, complement[set[id[omega]]]], Z]], composite[
   inverse[RATIO], fix[composite[inverse[RATMUL], RATMUL, SWAP]], RATIO]] = True

In[19]:= % /. Equal -> SetDelayed

Lemma. A better variable-free statement.
the neutral element for rational multiplication

It has been shown earlier that \( \text{id}[\mathbb{Z}] \) is right-neutral for rational multiplication. A variable-free statement of this will now be derived.

Lemma. Eliminating a variable.

\[
\text{Map}[\text{equal}[	ext{ratmul}[x, y], \#] \&, \text{ApComp}[\text{RATMUL}, \text{SWAP}, \text{PAIR}[x, y]]] = \text{True}
\]

\[
\text{equal}[	ext{ratmul}[x, y], \text{ratmul}[y, x]] = \text{True}
\]

\[
\text{equal}[	ext{ratmul}[x_, y_], \text{ratmul}[y_, x_]] = \text{True}
\]
Theorem. A better formulation.

In[30]:- \texttt{SubstTest[subclass, domain[funpart[t]],}
\texttt{fix[funpart[t]], t \to composite[RATMUL, \text{RIGHT[id[Z]]}]]}

In[31]:- \% \texttt{/. Equal \to SetDelayed}

A still better rule can be obtained by replacing the above inclusion by an equation.

Corollary. Right-neutrality of \texttt{id[Z]}.

In[33]:- \texttt{composite[RATMUL, \text{RIGHT[id[Z]]}] := id[RATS]}

Since \texttt{RATMUL} is commutative, \texttt{id[Z]} is also left-neutral.

Corollary.

In[34]:- \texttt{Assoc[RATMUL, SWAP, \text{RIGHT[id[Z]]}]}\]
Out[34]= composite[RATMUL, LEFT[id[Z]]] = id[RATS]

Theorem. The rational number \texttt{id[Z]} is a neutral element for rational multiplication.

In[36]:- \texttt{member[id[Z] \to ids[RATMUL]] // AssertTest}
Out[36]= member[id[Z], ids[RATMUL]] = True

In[37]:- \% \texttt{/. Equal \to SetDelayed}

Since \texttt{RATMUL} is a binary operation, \texttt{id[Z]} is the only neutral element.

Theorem.

In[38]:- \texttt{SubstTest[implies, and\{member[x, BINOPS], member[y, ids[x]]\],}
\texttt{equal[ids[x], set[y]], \{x \to RATMUL, y \to id[Z]\]] // Reverse}
Out[38]= equal[ids[RATMUL], set[id[Z]]] = True

In[39]:- \texttt{ids[RATMUL] := set[id[Z]]}

Corollary. The neutral element for rational multiplication is \texttt{id[Z]}.
\(\text{In}[40]:= \text{SubstTest}[A, \text{ids}[x], x \rightarrow \text{RATMUL}]\)

\(\text{Out}[40]= \text{e[RATMUL]} = \text{id[Z]}\)

\(\text{In}[41]:= \text{e[RATMUL]} := \text{id[Z]}\)