RATIO as a binary homomorphism

Johan G. F. Belinfante
2012 August 17

```
In[1]:= SetDirectory["l:"]; << goedel.12aug17a

        :Package Title: goedel.12aug17a              2012 August 17 at 11:40 a.m.

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

        It is now:  2012 Aug 17 at 14:0

        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 17 at 14:16
```

summary

It is shown in this notebook that the function **RATIO** is a binary homomorphism from the direct product of multiplication of nonzero integers with multiplication of all integers to multiplication of rational numbers. This result is used to show that rational multiplication is associative.

INTMUL rules

Simplification rules for integer multiplication are derived in this section.

Theorem. A simplification rule.

```
In[2]:= Assoc[id[x], id[Z], INTMUL] // Reverse

Out[2]= composite[id[intersection[x, Z]], INTMUL] == composite[id[x], INTMUL]

In[3]:= composite[id[intersection[x_, Z]], INTMUL] := composite[id[x], INTMUL]
```

Theorem. A simplification rule.

```
In[4]:= composite[INTMUL, cross[set[x], set[y]], inverse[INTMUL]] // FastReifNormality

Out[4]= composite[INTMUL, cross[set[x], set[y]], inverse[INTMUL]] ==
         cart[set[intmul[first[x], first[y]]], set[intmul[second[x], second[y]]]]
```

```
In[5]:= composite[INTMUL, cross[set[x_], set[y_]], inverse[INTMUL]] :=
         cart[set[intmul[first[x], first[y]]], set[intmul[second[x], second[y]]]]
```

Theorem. The direct product of multiplication of nonzero integers with multiplication of all integers is a semigroup multiplication law.

```
In[6]:= SubstTest[implies,
          and[member[x, SEMIGPS], member[y, SEMIGPS], member[direct[x, y], SEMIGPS],
          {x → composite[id[complement[set[id[omega]]]], INTMUL], y -> INTMUL}] // Reverse

Out[6]= member[
          composite[cross[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL], TWIST],
          SEMIGPS] == True

In[7]:= member[
          composite[cross[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL], TWIST],
          SEMIGPS] := True
```

RATIO relates rational to integer multiplication

Fractions are written here as **d \ n** instead of **n / d**. Rational multiplication is related to integer multiplication by: **(a \ b) · (c \ d) = (a · c) \ (b · d)**. In this section the four variables in this formula will be eliminated. Note that the order of the variables on the two sides of this equation differ in that the middle two variables are interchanged. This change in the order of variables is reflected in the variable-free equation by the presence of the function **TWIST**. The above explicit formula relating rational multiplication to integer multiplication was derived 2012 August 16 in the notebook **rmul-sw.nb**. The variables can be eliminated all at once by replacing them by a single one, and rewriting the equation as a pair of inclusions. This will be done now.

Theorem. An inclusion involving a single variable.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> member[x, cartsq[domain[RATIO]]],
            p2 -> equal[composite[inverse[inttimes[intmul[first[first[x]], first[second[x]]]]],
               inttimes[intmul[second[first[x]], second[second[x]]]]],
              ratmul[APPLY[RATIO, first[x]], APPLY[RATIO, second[x]]]], p3 →
             subclass[composite[inverse[inttimes[intmul[first[first[x]], first[second[x]]]]],
               inttimes[intmul[second[first[x]], second[second[x]]]]],
              ratmul[APPLY[RATIO, first[x]], APPLY[RATIO, second[x]]]]}]] // Reverse

Out[8]= or[equal[first[first[x]], id[omega]], equal[first[second[x]], id[omega]],
          not[member[first[first[x]], Z]], not[member[first[second[x]], Z]],
          not[member[second[first[x]], Z]], not[member[second[second[x]], Z]],
          subclass[composite[inverse[inttimes[intmul[first[first[x]], first[second[x]]]]],
            inttimes[intmul[second[first[x]], second[second[x]]]]],
           ratmul[APPLY[RATIO, first[x]], APPLY[RATIO, second[x]]]] == True

In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The opposite inclusion.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> member[x, cartsq[domain[RATIO]]], p2 ->
             equal[composite[inverse[inttimes[intmul[first[first[x]], first[second[x]]]]],
               inttimes[intmul[second[first[x]], second[second[x]]]]],
              ratmul[APPLY[RATIO, first[x]], APPLY[RATIO, second[x]]]], p3 →
             contains[composite[inverse[inttimes[intmul[first[first[x]], first[second[x]]]]],
               inttimes[intmul[second[first[x]], second[second[x]]]]],
              ratmul[APPLY[RATIO, first[x]], APPLY[RATIO, second[x]]]]}]] // Reverse

Out[10]= or[equal[first[first[x]], id[omega]], equal[first[second[x]], id[omega]],
           not[member[first[first[x]], Z]], not[member[first[second[x]], Z]],
           not[member[second[first[x]], Z]], not[member[second[second[x]], Z]],
           subclass[ratmul[APPLY[RATIO, first[x]], APPLY[RATIO, second[x]]],
            composite[inverse[inttimes[intmul[first[first[x]], first[second[x]]]]],
             inttimes[intmul[second[first[x]], second[second[x]]]]]] == True

In[11]:= (% /. x → x_) /. Equal→ SetDelayed
```

It takes about one whole minute to eliminate the variable in each of these two inclusions.

Theorem. Eliminating the variable in one of the two inclusions.

```
In[12]:= Map[equal[V, domain[#]] &, SubstTest[reify, x,
           case[implies[member[x, s], subclass[APPLY[funpart[t], APPLY[funpart[u], x]],
              APPLY[funpart[v], APPLY[funpart[w], x]]]],
           {s → cartsq[domain[RATIO]], t → RATMUL, u -> cross[RATIO, RATIO],
            v → composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]],
            w -> direct[INTMUL, INTMUL]}]]

Out[12]= subclass[cart[cart[intersection[Z, complement[set[id[omega]]]], Z],
            cart[intersection[Z, complement[set[id[omega]]]], Z]],
           fix[composite[cross[inverse[RATIO], inverse[RATIO]], inverse[RATMUL],
             inverse[S], COMPOSE, cross[composite[COMPOSE, SWAP,
               cross[composite[INVERSE, INTTIMES], composite[INVERSE, INTTIMES]]],
              composite[COMPOSE, cross[INTTIMES, INTTIMES]]], TWIST]]] == True

In[13]:= % /. Equal→ SetDelayed
```

Theorem. Eliminating the variable for the inclusion in the other direction.

```
In[14]:= Map[equal[V, domain[#]] &, SubstTest[reify, x,
           case[implies[member[x, s], contains[APPLY[funpart[t], APPLY[funpart[u], x]],
              APPLY[funpart[v], APPLY[funpart[w], x]]]],
           {s → cartsq[domain[RATIO]], t → RATMUL, u -> cross[RATIO, RATIO],
            v → composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]],
            w -> direct[INTMUL, INTMUL]}]]

Out[14]= subclass[cart[cart[intersection[Z, complement[set[id[omega]]]], Z],
            cart[intersection[Z, complement[set[id[omega]]]], Z]],
           fix[composite[cross[inverse[RATIO], inverse[RATIO]],
             inverse[RATMUL], S, COMPOSE, cross[composite[COMPOSE, SWAP,
               cross[composite[INVERSE, INTTIMES], composite[INVERSE, INTTIMES]]],
              composite[COMPOSE, cross[INTTIMES, INTTIMES]]], TWIST]]] == True
```

```
In[15]:= % /. Equal→ SetDelayed
```

Lemma. A temporary simplification rule.

```
In[16]:= Map[composite[inverse[RATMUL], #] &,
          Assoc[id[RATS], composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]],
           direct[INTMUL, INTMUL]]]
```

```
Out[16]= composite[inverse[RATMUL], COMPOSE, cross[composite[COMPOSE, SWAP,
           cross[composite[INVERSE, INTTIMES], composite[INVERSE, INTTIMES]]],
          composite[COMPOSE, cross[INTTIMES, INTTIMES]]], TWIST] ==
         composite[inverse[RATMUL], RATIO, cross[INTMUL, INTMUL], TWIST]
```

```
In[17]:= % /. Equal→ SetDelayed
```

Theorem. Combining two inclusions.

```
In[19]:= SubstTest[subclass, cartsq[domain[RATIO]], intersection[u, v],
           {u -> fix[composite[cross[inverse[RATIO], inverse[RATIO]],
               inverse[RATMUL], S, COMPOSE, cross[composite[COMPOSE, SWAP,
                 cross[composite[INVERSE, INTTIMES], composite[INVERSE, INTTIMES]]],
                composite[COMPOSE, cross[INTTIMES, INTTIMES]], TWIST]],
            v -> fix[composite[cross[inverse[RATIO], inverse[RATIO]], inverse[RATMUL],
               inverse[S], COMPOSE, cross[composite[COMPOSE, SWAP,
                 cross[composite[INVERSE, INTTIMES], composite[INVERSE, INTTIMES]]],
                composite[COMPOSE, cross[INTTIMES, INTTIMES]], TWIST]]}] // Reverse
```

```
Out[19]= subclass[cart[cart[intersection[Z, complement[set[id[omega]]]], Z],
           cart[intersection[Z, complement[set[id[omega]]]], Z]],
          fix[composite[cross[inverse[RATIO], inverse[RATIO]],
            inverse[RATMUL], RATIO, cross[INTMUL, INTMUL], TWIST]]] == True
```

```
In[20]:= % /. Equal→ SetDelayed
```

Lemma. A simpler inclusion.

```
In[21]:= SubstTest[subclass, domain[funpart[u]],
          fix[composite[inverse[funpart[u]], v]], {u → composite[RATMUL, cross[RATIO, RATIO]],
           v → composite[RATIO, cross[INTMUL, INTMUL], TWIST]}]
```

```
Out[21]= subclass[composite[RATMUL, cross[RATIO, RATIO]],
           composite[RATIO, cross[INTMUL, INTMUL], TWIST]] == True
```

```
In[22]:= % /. Equal→ SetDelayed
```

Theorem. An equation.

```
In[23]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]], {u -> composite[RATMUL, cross[RATIO, RATIO]],
            v -> composite[RATIO, cross[INTMUL, INTMUL], TWIST]}] // Reverse
```

```
Out[23]= equal[composite[RATMUL, cross[RATIO, RATIO]],
           composite[RATIO, cross[INTMUL, INTMUL], TWIST]] == True
```

```
In[24]:= composite[RATIO, cross[INTMUL, INTMUL], TWIST] := composite[RATMUL, cross[RATIO, RATIO]]
```

Theorem. A related equation.

```
In[25]:= Assoc[RATIO, id[domain[RATIO]], direct[INTMUL, INTMUL]]

Out[25]= composite[RATIO, cross[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL],
           TWIST] == composite[RATMUL, cross[RATIO, RATIO]]

In[26]:= composite[RATIO, cross[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL],
           TWIST] := composite[RATMUL, cross[RATIO, RATIO]]
```

Theorem. The function **RATIO** is a binary homomorphism.

```
In[27]:= member[RATIO, binhom[direct[
             composite[id[complement[set[id[omega]]]], INTMUL], INTMUL], RATMUL]] // AssertTest

Out[27]= member[RATIO, binhom[
            composite[cross[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL], TWIST],
            RATMUL]] == True

In[28]:= member[RATIO,
           binhom[composite[cross[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL],
             TWIST], RATMUL]] := True
```

associative law for rational multiplication

The fact that **RATIO** is a binary homomorphism from a semigroup to rational multiplication will now be used to show that rational multiplication is associative.

Theorem. The rational numbers form a semigroup under multiplication.

```
In[29]:= SubstTest[implies, and[member[x, SEMIGPS], member[t, binhom[x, y]]],
           member[composite[y, id[cart[range[t], range[t]]]], SEMIGPS],
           {t → RATIO, x -> direct[composite[id[complement[set[id[omega]]]], INTMUL], INTMUL],
            y → RATMUL}] // Reverse

Out[29]= member[RATMUL, SEMIGPS] == True

In[30]:= member[RATMUL, SEMIGPS] := True
```

Corollary. The rational numbers form a monoid under multiplication.

```
In[31]:= SubstTest[and, member[x, SEMIGPS], not[empty[ids[x]]], x → RATMUL]

Out[31]= member[RATMUL, MONOIDS] == True

In[32]:= member[RATMUL, MONOIDS] := True
```

Corollary.

```
In[33]:= SubstTest[implies, member[x, SEMIGPS], associative[x], x → RATMUL] // Reverse
Out[33]= associative[RATMUL] == True

In[34]:= associative[RATMUL] := True
```

Corollary.

```
In[35]:= SubstTest[composite, assoc[x], cross[Id, assoc[x]], ASSOC, x → RATMUL] // Reverse
Out[35]= composite[RATMUL, cross[Id, RATMUL], ASSOC] == composite[RATMUL, cross[RATMUL, Id]]

In[36]:= composite[RATMUL, cross[Id, RATMUL], ASSOC] := composite[RATMUL, cross[RATMUL, Id]]
```