distributive laws for rational arithmetic

Johan G. F. Belinfante
2012 September 29

```
In[1]:= SetDirectory["l:"]; << goedel.12sep25a

:Package Title: goedel.12sep25a            2012 September 25 at 4:18 p.m.

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

It is now:  2012 Sep 29 at 16:6

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 Sep 29 at 16:21
```

summary

Derivations of the left and right distributive laws for rational arithmetic are presented in this notebook.

introduction

By definition, rational numbers in the **GOEDEL** program are maximal non-vertical straight lines through the origin in the integer plane **Z** × **Z**. Intuitively, one may regard a rational number as a linear function whose slope is that rational number. For example, the rational number one is the slant line **id[Z]** with slope one, and the rational number zero is the horizontal line **Z** × **{id[ω]}** with slope zero. In general, the linear functions encountered in the sequel need not be total; both their domains and ranges are subsets of the set **Z** of all integers. A non-trivial set in the integer plane can only be contained in a single rational number, called its **rational hull**. Non-triviality here means that the domain of the set is not a subset of the singleton of **id[ω]**.

The product of rational numbers **rat[x]** and **rat[y]** is the rational hull of their composite.

```
In[2]:= hull[RATS, composite[rat[x], rat[y]]]

Out[2]= ratmul[rat[x], rat[y]]
```

The sum **h = f + g** of functions **f** and **g** in the integer plane is defined by **h(x) = f(x) + g(x)**. This translates to **h = INTADD ∘ (f ⊗ g) ∘ DUP** upon eliminating the variable **x**. The interpretation is this: first the argument **x** is duplicated, then **f** and **g** are applied in parallel to the two copies of **x**, and finally the results are added together to get **h(x)**. The following temporary definition for function addition will be used in this notebook.

```
In[3]:= funadd[x_, y_] := composite[INTADD,
          intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]
```

The sum of rational numbers **rat[x]** and **rat[y]** is the rational hull of their function sum.

```
In[4]:= hull[RATS, funadd[rat[x], rat[y]]]

Out[4]= ratadd[rat[x], rat[y]]
```

Observation. The domain of the function sum of two functions in the integer plane is the intersection of their domains.

```
In[5]:= domain[funadd[x, y]]

Out[5]= intersection[image[inverse[x], Z], image[inverse[y], Z]]
```

Theorem. The sum of two rational numbers is a rational number.

```
In[6]:= SubstTest[member, APPLY[funpart[t], w],
          range[funpart[t]], {t → RATADD, w → PAIR[x, y]}] // Reverse

Out[6]= member[ratadd[x, y], RATS] == and[member[x, RATS], member[y, RATS]]

In[7]:= member[ratadd[x_, y_], RATS] := and[member[x, RATS], member[y, RATS]]
```

Theorem. If **x** or **y** is not a rational number, then **ratadd[x, y] = V**.

```
In[8]:= SubstTest[equal, V, APPLY[funpart[t], PAIR[x, y]], t → RATADD] // Reverse

Out[8]= equal[V, ratadd[x, y]] == or[not[member[x, RATS]], not[member[y, RATS]]]

In[9]:= equal[V, ratadd[x_, y_]] := or[not[member[x, RATS]], not[member[y, RATS]]]
```

a weak distributive law

In this section it is shown that composition is distributive over function addition only in a weak sense (an inclusion, not an equation), but this will suffice.

Theorem. An inclusion.

```
In[17]:= SubstTest[implies, equal[s, id[Z]], subclass[composite[id[Z], t], composite[s, t]],
           {s → composite[inverse[inttimes[w]], inttimes[w]], t ->
             composite[INTADD, cross[inverse[inttimes[w]], inverse[inttimes[w]]]]}] // Reverse

Out[17]= or[equal[w, id[omega]], not[member[w, Z]],
           subclass[composite[INTADD, cross[inverse[inttimes[w]], inverse[inttimes[w]]]],
            composite[inverse[inttimes[w]], INTADD]]] == True

In[18]:= (% /. w → w_) /. Equal → SetDelayed
```

The following weak distributive law is a corollary.

Corollary. If **INTADD ∘ (t ⊗ t) ⊂ t ∘ INTADD**, then **funadd[t ∘ x, t ∘ y] ⊂ t ∘ funadd[x, y]**.

```
In[19]:= SubstTest[implies, subclass[u, v],
           subclass[composite[u, w], composite[v, w]], {u -> composite[INTADD, cross[t, t]],
            v → composite[t, INTADD], w → composite[cross[x, y], DUP]}] // Reverse
```

```
Out[19]= or[not[subclass[composite[INTADD, cross[t, t]], composite[t, INTADD]]],
           subclass[composite[INTADD, intersection[composite[inverse[FIRST], t, x],
              composite[inverse[SECOND], t, y]]], composite[t, INTADD,
             intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]]] == True
```

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

Restatement. (An inclusion for **funadd**.) If **INTADD ∘ (t ⊗ t) ⊂ t ∘ INTADD**, then **funadd[t ∘ x, t ∘ y] ⊂ t ∘ funadd[x, y]**.

```
In[21]:= implies[subclass[composite[INTADD, cross[t, t]], composite[t, INTADD]],
          subclass[funadd[composite[t, x], composite[t, y]], composite[t, funadd[x, y]]]]
```

```
Out[21]= True
```

Observation. The hypothesis for the above corollary holds when **t = rat[x]**.

```
In[22]:= subclass[composite[INTADD, cross[rat[x], rat[x]]], composite[rat[x], INTADD]]
```

```
Out[22]= True
```

Theorem. (A weak distributive law.) **funadd[rat[x] ∘ rat[y], rat[x] ∘ rat[z]] ⊂ rat[x] ∘ funadd[rat[y], rat[z]]**.

```
In[23]:= SubstTest[implies, subclass[composite[INTADD, cross[t, t]], composite[t, INTADD]],
           subclass[funadd[composite[t, u], composite[t, v]], composite[t, funadd[u, v]]],
           {t → rat[x], u → rat[y], v → rat[z]}] // Reverse
```

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

```
In[24]:= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x_], rat[y_]],
             composite[inverse[SECOND], rat[x_], rat[z_]]]],
           composite[rat[x_], INTADD, intersection[composite[inverse[FIRST], rat[y_]],
             composite[inverse[SECOND], rat[z_]]]]] := True
```

The basic strategy will be to show that the set **funadd[rat[x] ∘ rat[y], rat[x] ∘ rat[z]]** is a nontrivial subset of both sides of the left distributive law for rational arithmetic. (The right distributive law follows from the left one by the commutative laws for rational arithmetic.) To deal with the problem of taking the rational hull one needs to show that this functional sum of composites is not trivial. Note that the domain of this set is an intersection of the domains of composites of rational numbers.

```
In[25]:= domain[funadd[composite[rat[x], rat[y]], composite[rat[x], rat[z]]]]

Out[25]= intersection[image[inverse[rat[y]], domain[rat[x]]],
          image[inverse[rat[z]], domain[rat[x]]]]
```

The nontriviality of this set is shown in the next section.

---

domains of composites of rational numbers

The integer zero **id[ω] ∈ Z** is not to be confused with the rational number zero **Z × {id[ω]} ∈ RATS**. They are related by the function **INTTIMES**.

```
In[26]:= APPLY[INTTIMES, id[omega]]

Out[26]= cart[Z, set[id[omega]]]
```

Lemma. The product of two rational numbers cannot be the singleton of the origin of the integer plane.

```
In[27]:= Map[not, SubstTest[implies, and[equal[u, v], member[v, w]],
           member[u, w], {u -> cart[set[id[omega]], set[id[omega]]],
            v → ratmul[rat[x], rat[y]], w → RATS}]] // Reverse

Out[27]= equal[cart[set[id[omega]], set[id[omega]]], ratmul[rat[x], rat[y]]] == False

In[28]:= equal[cart[set[id[omega]], set[id[omega]]], ratmul[rat[x_], rat[y_]]] := False
```

Lemma. The composite of two rational numbers cannot be the singleton of the origin of the integer plane.

```
In[29]:= Map[not, SubstTest[implies, equal[u, v],
           equal[hull[RATS, u], hull[RATS, v]], {u -> cart[set[id[omega]], set[id[omega]]],
            v -> composite[rat[x], rat[y]]}]] // Reverse

Out[29]= equal[cart[set[id[omega]], set[id[omega]]], composite[rat[x], rat[y]]] == False

In[30]:= equal[cart[set[id[omega]], set[id[omega]]], composite[rat[x_], rat[y_]]] := False
```

Theorem. The domain of the composite of two rational numbers is not a subset of **{id[ω]}**.

```
In[31]:= SubstTest[equal, u, composite[u, id[v]],
          {u → composite[rat[y], rat[x]], v → set[id[omega]]}]

Out[31]= subclass[image[inverse[rat[x]], domain[rat[y]]], set[id[omega]]] == False

In[32]:= subclass[image[inverse[rat[x_]], domain[rat[y_]]], set[id[omega]]] := False
```

Corollary. The domain of the composite of two rational numbers can not be equal to **{id[ω]}**.

```
In[33]:= equal[image[inverse[rat[x]], domain[rat[y]]], set[id[omega]]] // AssertTest

Out[33]= equal[image[inverse[rat[x]], domain[rat[y]]], set[id[omega]]] == False
```

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

To deal with intersection of the domains of two composites, a group theoretic argument will be used. Recall that any subgroup of a group is determined by its range. The following temporary abbreviation for the set of ranges of subgroups of **x** will be used in this notebook.

```
In[35]:= rsg[x_] := image[IMAGE[SECOND], intersection[GROUPS, P[x]]]
```

For example, the range if any subgroup of **INTADD** is the set of multiples of some integer. This rewrite rule just anounts to the familiar fact that every subgroup of **INTADD** is cyclic.

```
In[36]:= rsg[INTADD]

Out[36]= image[VERTSECT[INTDIV], Z]
```

Theorem. The composite of rational numbers is a subgroup of **direct[INTADD, INTADD]**.

```
In[37]:= SubstTest[implies, and[member[u, GROUPS], member[v, GROUPS],
            member[w, GROUPS], member[s, rsg[direct[v, w]]], member[t, rsg[direct[u, v]]]],
          member[composite[s, t], rsg[direct[u, w]]],
          {s → rat[x], t → rat[y], u → INTADD, v → INTADD, w → INTADD}] // Reverse

Out[37]= member[composite[rat[x], rat[y]], image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] == True
```

```
In[38]:= member[composite[rat[x_], rat[y_]], image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := True
```

Corollary. The domain of the composite of a pair of rational numbers is a the range of a subgroup of **INTADD**.

```
In[39]:= SubstTest[implies, member[t, rsg[direct[INTADD, INTADD]]],
          member[domain[t], rsg[INTADD]], t → composite[rat[y], rat[x]]] // Reverse

Out[39]= member[image[inverse[rat[x]], domain[rat[y]]], image[VERTSECT[INTDIV], Z]] == True
```

```
In[40]:= member[image[inverse[rat[x_]], domain[rat[y_]]], image[VERTSECT[INTDIV], Z]] := True
```

The following theorem just sharpens the above corollary slightly by adding the non-trviality condition.

Theorem. The domain of the composite of two rational numbers is the range of a non-trivial cyclic subgroup of **INTADD**.

```
In[41]:= SubstTest[member, image[inverse[rat[x]], domain[rat[y]]], intersection[u, v],
          {u → image[VERTSECT[INTDIV], Z], v -> complement[set[set[id[omega]]]]}] // Reverse

Out[41]= member[image[inverse[rat[x]], domain[rat[y]]],
          image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]] == True
```

```
In[42]:= member[image[inverse[rat[x_]], domain[rat[y_]]],
          image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]] := True
```

The set of ranges of non-trivial cyclic subgroups of **INTADD** is closed under binary intersections.

Theorem. The intersection of the domains of two composites of rational numbers is the range of a non-trivial cyclic subgroup of **INTADD**.

```
In[43]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → CAP, u → set[PAIR[image[inverse[rat[w]], domain[rat[x]]],
               image[inverse[rat[y]], domain[rat[z]]]]],
            v -> cartsq[image[VERTSECT[INTDIV], dif[Z, set[id[omega]]]]]}] // Reverse

Out[43]= member[intersection[image[inverse[rat[w]], domain[rat[x]]],
            image[inverse[rat[y]], domain[rat[z]]]],
          image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]] == True

In[44]:= member[intersection[image[inverse[rat[w_]], domain[rat[x_]]],
            image[inverse[rat[y_]], domain[rat[z_]]]],
          image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]] := True
```

Theorem. The range of a non-trivial cyclic subgroup of **INTADD** is not a subset of **{id[ω]}**.

```
In[45]:= Map[implies[#, not[subclass[x, set[id[omega]]]]] &,
          SubstTest[member, x, intersection[u, v],
           {u → range[VERTSECT[INTDIV]], v -> complement[set[0, set[id[omega]]]]}]] // Reverse

Out[45]= or[not[
            member[x, image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]]],
          not[subclass[x, set[id[omega]]]]] == True

In[46]:= or[not[
            member[x_, image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]]],
          not[subclass[x_, set[id[omega]]]]] := True
```

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

```
In[47]:= Map[not, SubstTest[implies,
            member[t, image[VERTSECT[INTDIV], intersection[Z, complement[set[id[omega]]]]]],
            not[subclass[t, set[id[omega]]]],
            t -> intersection[image[inverse[rat[w]], domain[rat[x]]],
              image[inverse[rat[y]], domain[rat[z]]]]]] // Reverse

Out[47]= subclass[intersection[image[inverse[rat[w]], domain[rat[x]]],
            image[inverse[rat[y]], domain[rat[z]]]], set[id[omega]]] == False

In[48]:= subclass[intersection[image[inverse[rat[w_]], domain[rat[x_]]],
            image[inverse[rat[y_]], domain[rat[z_]]]], set[id[omega]]] := False
```

---

distributive laws for rational arithmetic

Since the sum of rational numbers is the rational hull of their function sum, the following inclusion holds.

Theorem. An inclusion: **funadd[rat[x], rat[y]] ⊂ ratadd[rat[x], rat[y]]**.

```
In[49]:= SubstTest[subclass, t, hull[RATS, t], t -> funadd[rat[x], rat[y]]] // Reverse

Out[49]= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x]],
           composite[inverse[SECOND], rat[y]]]], ratadd[rat[x], rat[y]]] == True

In[50]:= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x_]],
           composite[inverse[SECOND], rat[y_]]]], ratadd[rat[x_], rat[y_]]] := True
```

Theorem. A consequence of the weak distributive law.

```
In[51]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → funadd[composite[rat[x], rat[y]], composite[rat[x], rat[z]]],
           v → composite[rat[x], funadd[rat[y], rat[z]]],
           w → composite[rat[x], ratadd[rat[y], rat[z]]]}] // Reverse

Out[51]= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
           composite[inverse[SECOND], rat[x], rat[z]]]],
          composite[rat[x], ratadd[rat[y], rat[z]]]] == True

In[52]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Corollary. **funadd[rat[x] ∘ rat[y], rat[x] ∘ rat[z]] ⊂ ratmul[rat[x], ratadd[rat[y], rat[z]]]**.

```
In[53]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → funadd[composite[rat[x], rat[y]], composite[rat[x], rat[z]]],
           v → composite[rat[x], ratadd[rat[y], rat[z]]],
           w → ratmul[rat[x], ratadd[rat[y], rat[z]]]}] // Reverse

Out[53]= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
           composite[inverse[SECOND], rat[x], rat[z]]]],
          ratmul[rat[x], ratadd[rat[y], rat[z]]]] == True

In[54]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

In the above inclusion, the left side is non-trivial, and the right side is a rational number. Hence, the rational hull of the left side is equal to the right side.

Theorem. An equation for one side of the left distributive law.

```
In[55]:= SubstTest[implies, and[not[subclass[domain[t], set[id[omega]]]],
           subclass[t, r], member[r, RATS]], equal[hull[RATS, t], r],
          {t -> composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
              composite[inverse[SECOND], rat[x], rat[z]]]],
           r -> ratmul[rat[x], ratadd[rat[y], rat[z]]]}] // Reverse

Out[55]= equal[
          hull[RATS, composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
             composite[inverse[SECOND], rat[x], rat[z]]]]],
          ratmul[rat[x], ratadd[rat[y], rat[z]]]] == True

In[56]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

A similar equation will now be derived for the other side of the distributive law, **ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]]**.

Lemma.

```
In[57]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> cross[inverse[DUP], INTADD], u → cross[w, x], v → cross[y, z]}] // Reverse

Out[57]= or[and[not[equal[0, domain[w]]], not[equal[0, domain[x]]],
            not[subclass[composite[Id, w], y]]], and[not[equal[0, domain[w]]],
            not[equal[0, domain[x]]], not[subclass[composite[Id, x], z]]],
           subclass[composite[INTADD, intersection[composite[inverse[FIRST], w],
              composite[inverse[SECOND], x]]], composite[INTADD,
             intersection[composite[inverse[FIRST], y], composite[inverse[SECOND], z]]]]] == True

In[58]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. Joint monotonicity of **funadd**.

```
In[59]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 -> and[subclass[w, y], subclass[x, z]], p2 -> subclass[cross[w, x], cross[y, z]],
             p3 -> subclass[funadd[w, x], funadd[y, z]]}]] // Reverse

Out[59]= or[not[subclass[w, y]], not[subclass[x, z]],
           subclass[composite[INTADD, intersection[composite[inverse[FIRST], w],
              composite[inverse[SECOND], x]]], composite[INTADD,
             intersection[composite[inverse[FIRST], y], composite[inverse[SECOND], z]]]]] == True

In[60]:= or[not[subclass[w_, y_]], not[subclass[x_, z_]],
           subclass[composite[INTADD, intersection[composite[inverse[FIRST], w_],
              composite[inverse[SECOND], x_]]], composite[INTADD, intersection[
              composite[inverse[FIRST], y_], composite[inverse[SECOND], z_]]]]] := True
```

Restatement.

```
In[61]:= implies[and[subclass[t, v], subclass[u, w]], subclass[funadd[t, u], funadd[v, w]]]

Out[61]= True
```

Lemma.

```
In[62]:= SubstTest[subclass, funadd[rat[u], rat[v]], ratadd[rat[u], rat[v]],
           {u -> ratmul[rat[x], rat[y]], v -> ratmul[rat[x], rat[z]]}] // Reverse

Out[62]= subclass[
           composite[INTADD, intersection[composite[inverse[FIRST], ratmul[rat[x], rat[y]]],
             composite[inverse[SECOND], ratmul[rat[x], rat[z]]]]],
           ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]]] == True

In[63]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[64]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> funadd[composite[rat[x], rat[y]], composite[rat[x], rat[z]]],
            v -> funadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]],
            w → ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]]}] // Reverse

Out[64]= subclass[composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
             composite[inverse[SECOND], rat[x], rat[z]]]],
           ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]]] == True

In[65]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Here again, the left side is non-trivial, and the right side is a rational number, so the rational hull of the left side is equal to the right side.

Theorem. An equation for the other side of the left distributive law.

```
In[66]:= SubstTest[implies, and[not[subclass[domain[t], set[id[omega]]]],
            subclass[t, r], member[r, RATS]], equal[hull[RATS, t], r],
           {t -> composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
               composite[inverse[SECOND], rat[x], rat[z]]]],
            r -> ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]]}] // Reverse

Out[66]= equal[
           hull[RATS, composite[INTADD, intersection[composite[inverse[FIRST], rat[x], rat[y]],
              composite[inverse[SECOND], rat[x], rat[z]]]]],
           ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]]] == True

In[67]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Combining the two equations, one obtains the left distributive law for rational arithmetic.

Corollary. The left distributive law with **rat** wrappers.

```
In[68]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
           {u -> ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]],
            v -> hull[RATS, composite[INTADD, intersection[composite[inverse[FIRST],
                 rat[x], rat[y]], composite[inverse[SECOND], rat[x], rat[z]]]]],
            w -> ratmul[rat[x], ratadd[rat[y], rat[z]]]}] // Reverse

Out[68]= equal[ratadd[ratmul[rat[x], rat[y]], ratmul[rat[x], rat[z]]],
           ratmul[rat[x], ratadd[rat[y], rat[z]]]] == True

In[69]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

In fact, the **rat** wrappers are not needed, as will now be shown.

Theorem.

```
In[70]:= SubstTest[implies, and[equal[x, rat[u]], equal[y, rat[v]], equal[z, rat[w]]],
           equal[ratadd[ratmul[x, y], ratmul[x, z]], ratmul[x, ratadd[y, z]]],
           {u → x, v → y, w → z}] // Reverse

Out[70]= or[equal[ratadd[ratmul[x, y], ratmul[x, z]], ratmul[x, ratadd[y, z]]],
           not[member[x, RATS]], not[member[y, RATS]], not[member[z, RATS]]] == True

In[71]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[72]:= SubstTest[implies, and[equal[u, v], equal[v, w]],
           equal[u, w], {u -> ratadd[ratmul[x, y], ratmul[x, z]],
            v → V, w -> ratmul[x, ratadd[y, z]]}] // Reverse

Out[72]= or[and[member[x, RATS], member[y, RATS], member[z, RATS]],
           equal[ratadd[ratmul[x, y], ratmul[x, z]], ratmul[x, ratadd[y, z]]]] == True

In[73]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. A wrapper-free statement of the left distributive law.

```
In[74]:= SubstTest[and, or[p, q], implies[p, q],
          {p -> and[member[x, RATS], member[y, RATS], member[z, RATS]],
           q -> equal[ratadd[ratmul[x, y], ratmul[x, z]], ratmul[x, ratadd[y, z]]]}]

Out[74]= equal[ratadd[ratmul[x, y], ratmul[x, z]], ratmul[x, ratadd[y, z]]] == True

In[75]:= ratmul[x_, ratadd[y_, z_]] := ratadd[ratmul[x, y], ratmul[x, z]]
```

The right distributive law will now be derived from left distributive law.

Lemma.

```
In[76]:= SubstTest[equal, ratmul[t, z], ratmul[z, t], t → ratadd[x, y]] // Reverse

Out[76]= equal[ratadd[ratmul[z, x], ratmul[z, y]], ratmul[ratadd[x, y], z]] == True

In[77]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[78]:= SubstTest[implies, and[equal[t, v], equal[u, w]], equal[ratadd[t, u], ratadd[v, w]],
           {t -> ratmul[z, x], u -> ratmul[z, y], v → ratmul[x, z], w → ratmul[y, z]}] // Reverse

Out[78]= equal[ratadd[ratmul[x, z], ratmul[y, z]], ratadd[ratmul[z, x], ratmul[z, y]]] == True

In[79]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Corollary. The right distributive law.

```
In[80]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
           {u -> ratmul[ratadd[x, y], z], v -> ratadd[ratmul[z, x], ratmul[z, y]],
            w -> ratadd[ratmul[x, z], ratmul[y, z]]}] // Reverse

Out[80]= equal[ratadd[ratmul[x, z], ratmul[y, z]], ratmul[ratadd[x, y], z]] == True

In[81]:= ratmul[ratadd[x_, y_], z_] := ratadd[ratmul[x, z], ratmul[y, z]]
```

rattimes[rat[x]]

Some additional forms of the distributive law are considered in this final section.

Theorem. A simplification law.

```
In[82]:= Assoc[rattimes[rat[x]], id[RATS], id[y]]

Out[82]= composite[rattimes[rat[x]], id[intersection[RATS, y]]] ==
          composite[rattimes[rat[x]], id[y]]

In[83]:= composite[rattimes[rat[x_]], id[intersection[RATS, y_]]] :=
          composite[rattimes[rat[x]], id[y]]
```

Theorem. An upper bound.

```
In[84]:= SubstTest[subclass, composite[Id, t], cart[y, z], t → rattimes[rat[x]]] // Reverse

Out[84]= subclass[rattimes[rat[x]], cart[y, z]] ==
          and[subclass[RATS, y], subclass[range[rattimes[rat[x]]], z]]

In[85]:= subclass[rattimes[rat[x_]], cart[y_, z_]] :=
          and[subclass[RATS, y], subclass[range[rattimes[rat[x]]], z]]
```

Theorem. An inclusion.

```
In[86]:= Map[subclass[#, RATS] &, ImageComp[RATMUL, RIGHT[x], V]]

Out[86]= subclass[range[rattimes[x]], RATS] == True

In[87]:= subclass[range[rattimes[x_]], RATS] := True
```

Corollary.

```
In[88]:= SubstTest[subclass, composite[Id, t], cart[RATS, V], t → rattimes[x]] // Reverse

Out[88]= subclass[rattimes[x], cart[RATS, V]] == True

In[89]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. A simplification rule.

```
In[90]:= equal[composite[rattimes[x], id[RATS]], rattimes[x]]

Out[90]= True

In[91]:= composite[rattimes[x_], id[RATS]] := rattimes[x]
```

Theorem. An intertwine equation.

```
In[92]:= Map[VERTSECT, SubstTest[reify, y,
           ratmul[x, APPLY[funpart[t], PAIR[first[y], second[y]]]], t → RATADD]]

Out[92]= composite[rattimes[x], RATADD] == composite[RATADD, cross[rattimes[x], rattimes[x]]]

In[93]:= composite[rattimes[x_], RATADD] := composite[RATADD, cross[rattimes[x], rattimes[x]]]
```

Corollary. The function **rattimes[x]** is a binary homomorphism.

```
In[94]:= member[rattimes[x], binhom[RATADD, RATADD]] // AssertTest

Out[94]= member[rattimes[x], binhom[RATADD, RATADD]] == member[x, RATS]

In[95]:= member[rattimes[x_], binhom[RATADD, RATADD]] := member[x, RATS]
```

A variable-free version of a distributive law can be derived using **reify**.

Theorem. A variable-free formulation of a distributive law.

```
In[96]:= Map[rotate[inverse[#]] &,
           SubstTest[reify, x, composite[t, RIGHT[x], RATADD], t → RATMUL]] // Reverse

Out[96]= composite[RATADD, cross[RATMUL, RATMUL], TWIST, cross[DUP, Id]] ==
          composite[RATMUL, cross[Id, RATADD]]

In[97]:= composite[RATADD, cross[RATMUL, RATMUL], TWIST, cross[DUP, Id]] :=
          composite[RATMUL, cross[Id, RATADD]]
```

Corollary. A variable-free formulation of the other distributive law.

```
In[98]:= Assoc[composite[RATADD, cross[RATMUL, RATMUL], TWIST, cross[Id, DUP]], SWAP, SWAP]

Out[98]= composite[RATADD, cross[RATMUL, RATMUL], TWIST, cross[Id, DUP]] ==
          composite[RATMUL, cross[RATADD, Id]]

In[99]:= composite[RATADD, cross[RATMUL, RATMUL], TWIST, cross[Id, DUP]] :=
          composite[RATMUL, cross[RATADD, Id]]
```