fractional representations of rationals

Johan G. F. Belinfante
2012 August 29

In[1]:= **SetDirectory["l:"]; << goedel.12aug28a**

```
:Package Title: goedel.12aug28a                2012 August 28 at 3:25 p.m.

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

It is now:  2012 Aug 29 at 14:3

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 29 at 14:18
```

---

summary

A rational number can be written in infinitely many ways as a fraction. In the **GOEDEL** program, rational numbers are straight lines through the origin in the integer plane **Z** × **Z**. The rational number represents the slope of the corresponding line. Each point **(u, v)** on the graph of a rational number other than the origin determines a representation of the rational number as a fraction with **u** as denominator and **v** as numerator.

---

fractions

The following temporary definition for the fraction **u \ v** with denominator **u** and numerator **v** is used in this notebook.

In[2]:= **frac[u_, v_] := composite[inverse[inttimes[u]], inttimes[v]]**

Each pair of integers determines a fraction. The point **(u, v)** is a member of the fraction **u \ v**.

In[3]:= **member[pair[u, v], frac[u, v]]**

Out[3]= and[member[u, Z], member[v, Z]]

This fraction is not necessarily a rational number, however, unless its numerator is an integer and the denominator is a non-zero integer. The integer zero is **id[ω]**.

```
In[4]:= member[frac[u, v], RATS]

Out[4]= and[member[u, Z], member[v, Z], not[equal[u, id[omega]]]]
```

introduction

It is often convenient to use the **rat** wrapper for rational numbers to avoid literals of the form **x ∈ RATS**. Most of the important results in this notebook will be stated twice, with and without the use of the wrapper.

Every rational number is a function. If **(u, v) ∈ rat[x]**, then **v** is the value of **rat[x]** at **u**.

```
In[5]:= implies[member[pair[u, v], rat[x]], equal[v, APPLY[rat[x], u]]]

Out[5]= True
```

Theorem. (Restatement without the **rat** wrapper.)

```
In[7]:= SubstTest[implies, equal[x, rat[t]],
          or[equal[v, APPLY[x, u]], not[member[pair[u, v], x]]], t → x] // Reverse

Out[7]= or[equal[v, APPLY[x, u]], not[member[x, RATS]], not[member[pair[u, v], x]]] == True

In[8]:= or[equal[v_, APPLY[x_, u_]],
          not[member[x_, RATS]], not[member[pair[u_, v_], x_]]] := True
```

It is sometimes convenient to denote an ordered pair **(u,v)** by a single variable **w**, with **u = first[w]** and **v = second[w]**. The above statement can then be rewritten as follows.

Theorem. For any point on the graph of a rational number regarded as a straight line function in the integer place, the second coordinate of the point is the value of the rational number at the first coordinate.

```
In[9]:= SubstTest[implies, member[w, funpart[t]],
          equal[APPLY[funpart[t], first[w]], second[w]], t → rat[x]] // Reverse

Out[9]= or[equal[APPLY[rat[x], first[w]], second[w]], not[member[w, rat[x]]]] == True

In[10]:= or[equal[APPLY[rat[x_], first[w_]], second[w_]], not[member[w_, rat[x_]]]] := True
```

Corollary. (Restatement without the **rat** wrapper.)

```
In[11]:= SubstTest[implies, equal[x, rat[t]],
           or[equal[second[w], APPLY[x, first[w]]], not[member[w, x]]], t → x] // Reverse

Out[11]= or[equal[APPLY[x, first[w]], second[w]], not[member[w, x]], not[member[x, RATS]]] == True

In[12]:= or[equal[APPLY[x_, first[w_]], second[w_]],
           not[member[w_, x_]], not[member[x_, RATS]]] := True
```

fractional representations of rationals

The function **RATIO** connects rational numbers with integer fractions. The domain of the function **RATIO** is **(Z – {id[ω]}) × Z**.

```
In[14]:= domain[RATIO]

Out[14]= cart[intersection[Z, complement[set[id[omega]]]], Z]
```

The function **RATIO** takes each point of the integer plane other than the vertical axis to the corresponding fraction.

```
In[15]:= implies[member[w, domain[RATIO]], equal[APPLY[RATIO, w], frac[first[w], second[w]]]]

Out[15]= True
```

The basic connection between fractions and rational numbers is currently available in the following form.

```
In[16]:= implies[and[member[w, rat[x]], not[equal[first[w], id[omega]]]],
          equal[APPLY[RATIO, w], rat[x]]]

Out[16]= True
```

In this section, several equivalent statements are derived that do not explicitly involve the function **RATIO**. The **APPLY** rule for **RATIO** involves **case**.

```
In[17]:= APPLY[RATIO, pair[u, v]]

Out[17]= union[case[or[equal[u, id[omega]], not[member[u, Z]], not[member[v, Z]]]],
          composite[inverse[inttimes[u]], inttimes[v]]]
```

A theorem will be needed to deal with the **case** construct.

Lemma. Temporary simplification rule.

```
In[18]:= equiv[or[and[p, equal[V, x]], and[equal[V, x], equal[V, y]], and[equal[x, y], not[p]]],
            or[and[p, equal[V, x]], and[equal[x, y], not[p]]]] // not // not

Out[18]= True

In[19]:= or[and[p_, equal[V, x_]], and[equal[V, x_], equal[V, y_]],
           and[equal[x_, y_], not[p_]]] := or[and[p, equal[V, x]], and[equal[x, y], not[p]]]
```

Theorem. An equation involving **case**.

```
In[20]:= equal[x, union[y, case[p]]] // AssertTest

Out[20]= equal[x, union[y, case[p]]] == or[and[p, equal[V, x]], and[equal[x, y], not[p]]]

In[21]:= equal[x_, union[y_, case[p_]]] := or[and[p, equal[V, x]], and[equal[x, y], not[p]]]
```

The following theorem connects rational numbers and fractions directy, eliminating the function **RATIO** altogether.

Theorem. If **(u, v) ∈ rat[x]**, and **u** is not the integer zero, then **rat[x] = u \ v**.

```
In[22]:= SubstTest[implies, and[member[w, rat[x]], not[equal[first[w], id[omega]]]],
           equal[APPLY[RATIO, w], rat[x]], w → pair[u, v]] // Reverse // MapNotNot

Out[22]= or[equal[u, id[omega]], equal[composite[inverse[inttimes[u]], inttimes[v]], rat[x]],
          not[member[pair[u, v], rat[x]]]] == True

In[23]:= or[equal[u_, id[omega]],
          equal[composite[inverse[inttimes[u_]], inttimes[v_]], rat[x_]],
          not[member[pair[u_, v_], rat[x_]]]] := True
```

Corollary. (Eliminate the **rat** wrapper.)

```
In[24]:= SubstTest[implies, equal[x, rat[t]],
          or[equal[u, id[omega]], equal[composite[inverse[inttimes[u]], inttimes[v]], x],
           not[member[pair[u, v], x]]], t → x] // Reverse

Out[24]= or[equal[u, id[omega]], equal[x, composite[inverse[inttimes[u]], inttimes[v]]],
          not[member[x, RATS]], not[member[pair[u, v], x]]] == True

In[25]:= or[equal[composite[inverse[inttimes[u_]], inttimes[v_]], x_],
          equal[id[omega], u_], not[member[pair[u_, v_], x_]], not[member[x_, RATS]]] := True
```

The result can also be restated using a single variable for the ordered pair.

Lemma. Simplification rule.

```
In[26]:= SubstTest[member, w, composite[Id, t], t → rat[x]]

Out[26]= and[member[w, rat[x]], member[first[w], V]] == member[w, rat[x]]

In[27]:= and[member[w_, rat[x_]], member[first[w_], V]] := member[w, rat[x]]
```

Theorem. If **w ∈ rat[x]** and if **first[w]** is not zero, then **rat[x] = first[w] \ second[w]**.

```
In[28]:= SubstTest[implies, and[member[w, rat[x]], not[equal[first[w], id[omega]]]],
           equal[APPLY[RATIO, w], rat[x]], w → PAIR[first[w], second[w]]] // Reverse // MapNotNot

Out[28]= or[equal[composite[inverse[inttimes[first[w]]], inttimes[second[w]]], rat[x]],
          equal[first[w], id[omega]], not[member[w, rat[x]]]] == True

In[29]:= or[equal[composite[inverse[inttimes[first[w_]]], inttimes[second[w_]]], rat[x_]],
          equal[first[w_], id[omega]], not[member[w_, rat[x_]]]] := True
```

Corollary. (Eliminating the **rat** wrapper.)

```
In[30]:= SubstTest[implies, equal[x, rat[t]],
           or[equal[composite[inverse[inttimes[first[w]]], inttimes[second[w]]], x],
            equal[first[w], id[omega]], not[member[w, x]]], t → x] // Reverse

Out[30]= or[equal[x, composite[inverse[inttimes[first[w]]], inttimes[second[w]]]],
           equal[first[w], id[omega]], not[member[w, x]], not[member[x, RATS]]] == True

In[31]:= or[equal[composite[inverse[inttimes[first[w_]]], inttimes[second[w_]]], x_],
           equal[first[w_], id[omega]], not[member[w_, x_]], not[member[x_, RATS]]] := True
```

Given a member of the domain of a function, one can obtain a point on the function itself using **APPLY**.

Theorem. Points on a rational number **rat[x]** are determined by their first coordinates.

```
In[32]:= SubstTest[member, pair[u, APPLY[funpart[t], u]], funpart[t], t → rat[x]] // Reverse

Out[32]= member[pair[u, APPLY[rat[x], u]], rat[x]] == member[u, domain[rat[x]]]

In[33]:= member[pair[u_, APPLY[rat[x_], u_]], rat[x_]] := member[u, domain[rat[x]]]
```

Theorem. If **u** is a non-zero member of **domain[rat[x]]**, then **rat[x] = u \ APPLY[rat[x], u]**.

```
In[34]:= SubstTest[implies, and[member[pair[u, v], rat[x]], not[equal[u, id[omega]]]],
           equal[rat[x], frac[u, v]], v → APPLY[rat[x], u]] // Reverse

Out[34]= or[equal[u, id[omega]],
           equal[composite[inverse[inttimes[u]], inttimes[APPLY[rat[x], u]]], rat[x]],
           not[member[u, domain[rat[x]]]]] == True

In[35]:= or[equal[u_, id[omega]],
           equal[composite[inverse[inttimes[u_]], inttimes[APPLY[rat[x_], u_]]], rat[x_]],
           not[member[u_, domain[rat[x_]]]]] := True
```

Corollary. (Restatement without the **rat** wrapper.)

```
In[36]:= SubstTest[implies, equal[x, rat[t]], or[equal[u, id[omega]],
            equal[composite[inverse[inttimes[u]], inttimes[APPLY[x, u]]], x],
            not[member[u, domain[x]]]], t → x] // Reverse

Out[36]= or[equal[u, id[omega]],
           equal[x, composite[inverse[inttimes[u]], inttimes[APPLY[x, u]]]],
           not[member[u, domain[x]]], not[member[x, RATS]]] == True

In[38]:= or[equal[composite[inverse[inttimes[u_]], inttimes[APPLY[x_, u_]]], x_],
           equal[id[omega], u_], not[member[u_, domain[x_]]], not[member[x_, RATS]]] := True
```