the rational number zero

Johan G. F. Belinfante
2012 September 10

In[1]:= **SetDirectory["l:"]; << goedel.12sep08a**

```
:Package Title: goedel.12sep08a             2012 September 8 at 3:15 p.m.

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

It is now:  2012 Sep 10 at 5:27

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 10 at 5:43
```

---

summary

Rational numbers are straight lines through the origin in the integer plane **Z** × **Z**. Consequently, the integer zero **id[ω]** belongs to the domain and range of every rational number. The only rational number whose range has no other members is zero. It is shown below that the only rational number with range **{id[ω]}** is the rational number zero.

---

derivation

The rational number wrapper **rat[x]** and fractions are both useful for deriving results about rational numbers. The following abbreviation for the fraction **x \ y** with denominator **x** and numerator **y** is convenient.

In[2]:= **frac[x_, y_] := composite[inverse[inttimes[x]], inttimes[y]]**

A fraction is a rational number only if the numerator and denominator are both integers, and the denominator is not the integer zero.

In[4]:= **member[frac[x, y], RATS]**

Out[4]= and[member[x, Z], member[y, Z], not[equal[x, id[omega]]]]

A fraction whose denominator is the integer one will be called a **whole number**. The function **inttimes[int[x]]** is the whole number corresponding to the integer **int[x]**.

```
In[3]:= frac[plus[set[0]], int[x]]

Out[3]= inttimes[int[x]]
```

One should not confuse the integer zero **id[ω]** with the rational number zero **Z × {id[ω]}**.

```
In[5]:= inttimes[plus[0]]

Out[5]= cart[Z, set[id[omega]]]
```

Theorem. If **rat[x] = frac[u, v]**, then **u** and **v** are integers, and **u** is not zero.

```
In[6]:= SubstTest[implies, and[equal[r, s], member[s, t]],
          member[r, t], {r → frac[u, v], s → rat[x], t → RATS}] // Reverse

Out[6]= or[and[member[u, Z], member[v, Z], not[equal[u, id[omega]]]],
          not[equal[composite[inverse[inttimes[u]], inttimes[v]], rat[x]]]] == True

In[7]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

The rational number zero can be written as a fraction in many ways. The numerator is always zero, but the denominator can be any nonzero integer. No matter how one writes zero as a fraction, the range is always the same. The range of the fraction **int[x] \ id[ω]** is given by the following expression.

```
In[9]:= range[frac[int[x], id[omega]]]

Out[9]= image[inverse[inttimes[int[x]]], set[id[omega]]]
```

The improper fraction **id[ω] \ id[ω]** is not a rational number. It is not even a function; it is the whole integer plane.

```
In[10]:= frac[id[omega], id[omega]]

Out[10]= cart[Z, Z]
```

Theorem. The range of **int[x] \ id[ω]** is **{id[ω]}** unless **int[x]** is zero. If the numerator and denominator are both zero, the range is **Z**.

```
In[11]:= Map[intersection[Z, domain[reify[t, #]]] &,
          SubstTest[case, member[t, u], u -> image[inverse[inttimes[int[x]]], set[id[omega]]]]]

Out[11]= image[inverse[inttimes[int[x]]], set[id[omega]]] == union[
           intersection[Z, complement[image[V, intersection[omega, complement[fix[int[x]]]]]]],
           set[id[omega]]]

In[12]:= image[inverse[inttimes[int[x_]]], set[id[omega]]] := union[
           intersection[Z, complement[image[V, intersection[omega, complement[fix[int[x]]]]]]],
           set[id[omega]]]
```

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

```
In[13]:= SubstTest[implies, equal[x, int[t]], implies[not[equal[id[omega], x]],
            equal[frac[x, id[omega]], frac[plus[set[0]], id[omega]]]], t → x] // Reverse

Out[13]= or[equal[x, id[omega]],
          equal[image[inverse[inttimes[x]], set[id[omega]]], set[id[omega]]],
          not[member[x, Z]]] == True

In[14]:= or[equal[x_, id[omega]], equal[image[inverse[inttimes[x_]], set[id[omega]]],
            set[id[omega]]], not[member[x_, Z]]] := True
```

Lemma. (A converse result.)

```
In[15]:= SubstTest[implies, equal[t, set[id[omega]]], not[empty[t]],
          t -> image[inverse[inttimes[x]], set[id[omega]]]] // Reverse

Out[15]= or[member[x, Z],
          not[equal[image[inverse[inttimes[x]], set[id[omega]]], set[id[omega]]]]] == True

In[16]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. A rewrite rule conerning the range of the fraction **x \ id[ω]**.

```
In[17]:= equiv[equal[image[inverse[inttimes[x]], set[id[omega]]], set[id[omega]]],
            and[member[x, Z], not[equal[x, id[omega]]]]] // not // not

Out[17]= True

In[18]:= equal[image[inverse[inttimes[x_]], set[id[omega]]], set[id[omega]]] :=
          and[member[x, Z], not[equal[x, id[omega]]]]
```

Any rational number **rat[x]** can be written as a fraction **u \ v**, with denominator **u** any point of its domain other than zero, and **v** the value of **rat[x]** at **u**.

Lemma. The only rational number **rat[x]** that can be written as a fraction with numerator zero is the rational number zero.

```
In[19]:= (Map[not, SubstTest[and, implies[and[p0, p1], p2],
              implies[and[p0, p1, p3], p5], implies[and[p2, p5], p6],
              not[implies[and[p0, p1, p3], p6]], {p0 → equal[v, APPLY[rat[x], u]],
               p1 -> and[member[u, domain[rat[x]]], not[equal[u, id[omega]]]],
               p2 -> equal[rat[x], frac[u, v]], p3 → equal[v, id[omega]],
               p5 → equal[frac[u, v], e[RATADD]],
               p6 → equal[rat[x], e[RATADD]]}]] /. v → id[omega]) // Reverse

Out[19]= or[equal[u, id[omega]], equal[cart[Z, set[id[omega]]], rat[x]],
          not[equal[APPLY[rat[x], u], id[omega]]], not[member[u, domain[rat[x]]]]] == True

In[20]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Lemma. If the range of a rational number is **{id[ω]}** then its value at any point of its domain is **id[ω]**.

```
In[21]:= SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
           {u → APPLY[rat[x], u], v → range[rat[x]], w → set[id[omega]]}] // Reverse

Out[21]= or[equal[APPLY[rat[x], u], id[omega]],
           not[equal[range[rat[x]], set[id[omega]]]], not[member[u, domain[rat[x]]]]] == True

In[22]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Lemma. (Combining the preceding two lemmas.)

```
In[23]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 -> and[member[u, domain[rat[x]]], not[equal[u, id[omega]]]],
             p2 → equal[range[rat[x]], set[id[omega]]], p3 -> equal[APPLY[rat[x], u], id[omega]],
             p4 -> equal[cart[Z, set[id[omega]]], rat[x]]}]] // Reverse

Out[23]= or[equal[u, id[omega]], equal[cart[Z, set[id[omega]]], rat[x]],
           not[equal[range[rat[x]], set[id[omega]]]], not[member[u, domain[rat[x]]]]] == True

In[24]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

The variable **u** can be eliminated using reification.

Lemma. If the range of a rational number is **{id[ω]}**, then its domain holds every integer.

```
In[25]:= Map[equal[V, domain[#]] &, SubstTest[reify, u, case[
            or[equal[u, id[omega]], equal[t, rat[x]], not[equal[range[rat[x]], set[id[omega]]]],
             not[member[u, domain[rat[x]]]]]], t -> cart[Z, set[id[omega]]]]]]

Out[25]= or[not[equal[range[rat[x]], set[id[omega]]]],
           subclass[Z, image[inverse[rat[x]], set[id[omega]]]]] == True

In[26]:= (% /. {x → x_}) /. Equal → SetDelayed
```

Main Theorem. If **range[rat[x]] = {id[ω]}**, then **rat[x] = Z × {id[ω]}**.

```
In[27]:= Map[implies[equal[range[rat[x]], set[id[omega]]], #] &,
           equal[rat[x], cart[Z, set[id[omega]]]] // AssertTest] // MapNotNot

Out[27]= or[equal[cart[Z, set[id[omega]]], rat[x]],
           not[equal[range[rat[x]], set[id[omega]]]]] == True

In[28]:= or[equal[cart[Z, set[id[omega]]], rat[x_]],
           not[equal[range[rat[x_]], set[id[omega]]]]] := True
```

Corollary. (Remove the **rat** wrapper.) If the integer zero is the only member of the range of a rational number, then the rational number is the rational number zero.

```
In[29]:= SubstTest[implies, equal[x, rat[t]], or[equal[cart[Z, set[id[omega]]], x],
            not[equal[range[x], set[id[omega]]]]], t → x] // Reverse

Out[29]= or[equal[x, cart[Z, set[id[omega]]]],
           not[equal[range[x], set[id[omega]]]], not[member[x, RATS]]] == True
```

```
In[30]:= or[equal[x_, cart[Z, set[id[omega]]]],
           not[equal[range[x_], set[id[omega]]]], not[member[x_, RATS]]] := True
```

Lemma. A variable-free statement.

```
In[32]:= Map[equal[V, #] &,
          complement[intersection[RATS, image[inverse[IMAGE[SECOND]], set[set[id[omega]]]]],
             complement[set[cart[Z, set[id[omega]]]]]]] // Normality]

Out[32]= subclass[intersection[RATS, image[inverse[IMAGE[SECOND]], set[set[id[omega]]]]],
           set[cart[Z, set[id[omega]]]]] == True

In[33]:= % /. Equal → SetDelayed
```

Theorem. An equation.

```
In[34]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[RATS, image[inverse[IMAGE[SECOND]], set[set[id[omega]]]]],
           v -> set[cart[Z, set[id[omega]]]]}]

Out[34]= equal[intersection[RATS, image[inverse[IMAGE[SECOND]], set[set[id[omega]]]]],
           set[cart[Z, set[id[omega]]]]] == True

In[36]:= intersection[RATS, image[inverse[IMAGE[SECOND]], set[set[id[omega]]]]] :=
          set[cart[Z, set[id[omega]]]]
```