x ∈ y ∈ RATS

Johan G. F. Belinfante
2012 July 20

```
In[1]:= SetDirectory["l:"]; << goedel.12jul19a

        :Package Title: goedel.12jul19a                2012 July 19 at 11:40 a.m.

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

        It is now:  2012 Jul 20 at 11:43

        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 Jul 20 at 11:59
```

summary

In the **GOEDEL** program, rational numbers are straight lines through the origin in the $\mathbf{Z} \times \mathbf{Z}$ plane. Each rational number is identified with the slope of the corresponding line. If **x** is a point on the rational number **y** other than the origin, then **y = APPLY[RATIO, x]**.

derivation

Theorem. If **x ∈ y ∈ RATS** and **x ∈ domain[RATIO]**, then **y = APPLY[RATIO, x]**.

```
In[2]:= Map[implies[#, equal[y, APPLY[RATIO, x]]] &,
         SubstTest[member, pair[y, x], composite[inverse[E],
           IMAGE[id[cart[intersection[Z, complement[set[id[omega]]]], Z]]], id[t]], t → RATS]]

Out[2]= or[equal[y, APPLY[RATIO, x]], equal[first[x], id[omega]], not[member[x, y]],
          not[member[y, RATS]], not[member[first[x], Z]], not[member[second[x], Z]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The above result has two redundant literals.

Lemma. Simplification rule.

```
In[4]:= equiv[or[equal[x, id[omega]], member[x, Z]], member[x, Z]]

Out[4]= True

In[5]:= or[equal[x_, id[omega]], member[x_, Z]] := member[x, Z]
```

Lemma. If **x ∈ y ∈ RATS**, then **first[x] ∈ Z**.

```
In[6]:= Map[or[member[first[x], Z], #] &, SubstTest[implies,
          and[member[x, y], member[y, z]], member[x, U[z]], z → RATS]] // Reverse // MapNotNot

Out[6]= or[member[first[x], Z], not[member[x, y]], not[member[y, RATS]]] == True

In[7]:= or[member[first[x_], Z], not[member[x_, y_]], not[member[y_, RATS]]] := True
```

Theorem. If **x ∈ y ∈ RATS**, then **second[x] ∈ Z**.

```
In[8]:= Map[or[member[second[x], Z], #] &, SubstTest[implies,
          and[member[x, y], member[y, z]], member[x, U[z]], z → RATS]] // Reverse // MapNotNot

Out[8]= or[member[second[x], Z], not[member[x, y]], not[member[y, RATS]]] == True

In[9]:= or[member[second[x_], Z], not[member[x_, y_]], not[member[y_, RATS]]] := True
```

Main Theorem. If **x** is a point on the rational number **y** other than the origin, then **y = APPLY[RATIO, x]**.

```
In[10]:= Map[not,
           SubstTest[and, implies[p1, p2], implies[p1, p3], (*implies[and[p0,p1,p2,p3],p4],*)
            not[implies[and[p0, p1], p4]], {p0 → not[equal[first[x], id[omega]]],
             p1 -> and[member[x, y], member[y, RATS]], p2 -> member[first[x], Z],
             p3 -> member[second[x], Z], p4 -> equal[y, APPLY[RATIO, x]]}]] // Reverse

Out[10]= or[equal[y, APPLY[RATIO, x]], equal[first[x], id[omega]],
           not[member[x, y]], not[member[y, RATS]]] == True

In[11]:= or[equal[y_, APPLY[RATIO, x_]], equal[first[x_], id[omega]],
           not[member[x_, y_]], not[member[y_, RATS]]] := True
```

eliminating variables

In this section, some corollaries are obtained in which one or both variables are eliminated.

Theorem. If **x** is a rational number, then the restriction of **RATIO** to **x** is constant, with value **x**.

```
In[12]:= Map[equal[V, domain[#]] &, SubstTest[reify, y,
           case[or[equal[x, APPLY[funpart[t], y]], not[member[y, domain[funpart[t]]]],
             not[member[x, RATS]], not[member[y, x]]]], t → RATIO]]

Out[12]= or[not[member[x, RATS]], subclass[image[RATIO, x], set[x]]] == True

In[13]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[14]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → RATIO, u → set[x], v → APPLY[RATIO, x]}] // Reverse

Out[14]= or[equal[first[x], id[omega]], member[APPLY[RATIO, x], image[RATIO, APPLY[RATIO, x]]],
           not[member[first[x], Z]], not[member[second[x], Z]]] == True

In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. An inclusion.

```
In[16]:= Map[equal[V, domain[#]] &,
           SubstTest[reify, x, case[implies[member[x, domain[funpart[t]]], member[
               APPLY[funpart[t], x], image[t, APPLY[funpart[t], x]]]]], t → RATIO]] // InvertFix

Out[16]= subclass[RATS, fix[composite[inverse[E], IMAGE[RATIO]]]] == True

In[17]:= % /. Equal → SetDelayed
```

Lemma. Reverse inclusion.

```
In[18]:= SubstTest[subclass, fix[t], range[t],
           t -> composite[inverse[E], IMAGE[RATIO]]] // Reverse

Out[18]= subclass[fix[composite[inverse[E], IMAGE[RATIO]]], RATS] == True

In[19]:= % /. Equal → SetDelayed
```

Theorem. An equation.

```
In[20]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> fix[composite[inverse[E], IMAGE[RATIO]]], v -> RATS}]

Out[20]= equal[RATS, fix[composite[inverse[E], IMAGE[RATIO]]]] == True

In[21]:= fix[composite[inverse[E], IMAGE[RATIO]]] := RATS
```

The next theorem reintroduces a variable.

Theorem.

```
In[22]:= (member[x, fix[t]] // AssertTest) /. t -> composite[inverse[E], IMAGE[RATIO]] // Reverse

Out[22]= member[x, image[RATIO, x]] == member[x, RATS]

In[23]:= member[x_, image[RATIO, x_]] := member[x, RATS]
```

Corollary.

```
In[24]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p → member[x, RATS], u → set[x], v → image[RATIO, x]}]

Out[24]= or[equal[image[RATIO, x], set[x]], not[member[x, RATS]]] == True
```

```
In[25]:= or[equal[image[RATIO, x_], set[x_]], not[member[x_, RATS]]] := True
```

Corollary.

```
In[26]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[and[p1, p2], p3], not[implies[p1, p3]], {p1 → member[x, RATS],
            p2 → equal[image[RATIO, x], set[x]], p3 → equal[U[image[RATIO, x]], x]}]] // Reverse

Out[26]= or[equal[x, U[image[RATIO, x]]], not[member[x, RATS]]] == True

In[27]:= or[equal[x_, U[image[RATIO, x_]]], not[member[x_, RATS]]] := True
```

Another variable-free statement will now be derived.

Lemma.

```
In[28]:= Map[complement[domain[#]] &, SubstTest[reify, x,
           case[or[equal[image[RATIO, x], set[x]], not[member[x, t]]]], t → RATS]]

Out[28]= intersection[RATS,
           fix[composite[inverse[SINGLETON], Di, id[complement[set[0]]], IMAGE[RATIO]]]] == 0

In[29]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[30]:= SubstTest[empty, fix[composite[inverse[SINGLETON], Di, funpart[t]]],
          t → composite[id[complement[set[0]]], IMAGE[RATIO], id[RATS]]]

Out[30]= subclass[composite[id[complement[set[0]]],
            IMAGE[RATIO], id[RATS], inverse[SINGLETON]], Id] == True

In[31]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[32]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
           {u → composite[id[complement[set[0]]], IMAGE[RATIO], id[RATS], inverse[SINGLETON]],
            v -> Id, w → SINGLETON}] // Reverse

Out[32]= subclass[composite[id[complement[set[0]]], IMAGE[RATIO], id[RATS]], SINGLETON] == True

In[33]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[34]:= AssInt[RATS, P[U[RATS]],
           P[complement[cart[intersection[Z, complement[set[id[omega]]]], Z]]]] // Reverse

Out[34]= intersection[RATS,
           P[complement[cart[intersection[Z, complement[set[id[omega]]]], Z]]]] == 0

In[35]:= % /. Equal→ SetDelayed
```

Lemma. An inclusion.

```
In[36]:= Map[subclass[#, SINGLETON] &, SubstTest[composite, union[u, v], IMAGE[RATIO],
           id[RATS], {u → id[set[0]], v → id[complement[set[0]]]}]] // Reverse
```

```
Out[36]= subclass[composite[IMAGE[RATIO], id[RATS]], SINGLETON] == True
```

```
In[37]:= % /. Equal→ SetDelayed
```

Theorem. A variable-free equation.

```
In[38]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u -> composite[IMAGE[RATIO], id[RATS]], v -> SINGLETON}] // Reverse
```

```
Out[38]= equal[composite[SINGLETON, id[RATS]], composite[IMAGE[RATIO], id[RATS]]] == True
```

```
In[39]:= composite[IMAGE[RATIO], id[RATS]] := composite[SINGLETON, id[RATS]]
```