inclusions for rationals

Johan G. F. Belinfante
2012 July 24

```
In[1]:= SetDirectory["l:"]; << goedel.12jul23a

        :Package Title: goedel.12jul23a                 2012 July 23 at 8:55 a.m.

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

        It is now:  2012 Jul 24 at 15:18

        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 24 at 15:34
```

summary

If **x** and **y** are rational numbers, and **x ⊂ y**, then **x = y**.

derivation

Lemma. Simplification rule.

```
In[2]:= SubstTest[composite, t, id[domain[t]], t → RATIO] // Reverse

Out[2]= composite[RATIO, id[cart[complement[set[id[omega]]], V]]] == RATIO

In[3]:= composite[RATIO, id[cart[complement[set[id[omega]]], V]]] := RATIO
```

Lemma.

```
In[4]:= SubstTest[implies, and[member[x, t], not[equal[first[x], id[omega]]], member[t, RATS]],
          equal[t, APPLY[RATIO, x]], t → APPLY[RATIO, y]] // Reverse

Out[4]= or[equal[APPLY[RATIO, x], APPLY[RATIO, y]], equal[first[x], id[omega]],
          equal[first[y], id[omega]], not[member[x, APPLY[RATIO, y]]],
          not[member[first[y], Z]], not[member[second[y], Z]]] == True

In[5]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= SubstTest[implies, and[member[r, s], subclass[s, t]],
          member[r, t], {r → x, s → APPLY[RATIO, x], t -> APPLY[RATIO, y]}] // Reverse

Out[6]= or[member[x, APPLY[RATIO, y]], not[member[x, V]],
          not[subclass[APPLY[RATIO, x], APPLY[RATIO, y]]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[8]:= Map[not, SubstTest[and, implies[and[p0, p1], p2],
           implies[and[p2, p3, p4], p5], not[implies[and[p1, p3, p4], p5]],
           {p0 → member[x, V], p1 -> subclass[APPLY[RATIO, x], APPLY[RATIO, y]],
            p2 -> member[x, APPLY[RATIO, y]], p3 → member[x, domain[RATIO]], p4 →
             member[y, domain[RATIO]], p5 → equal[APPLY[RATIO, x], APPLY[RATIO, y]]}]] // Reverse

Out[8]= or[equal[APPLY[RATIO, x], APPLY[RATIO, y]], equal[first[x], id[omega]],
          equal[first[y], id[omega]], not[member[first[x], Z]],
          not[member[first[y], Z]], not[member[second[x], Z]], not[member[second[y], Z]],
          not[subclass[APPLY[RATIO, x], APPLY[RATIO, y]]]] == True

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (Eliminating both variables.)

```
In[10]:= Map[composite[Id, complement[#]] &,
          SubstTest[class, pair[x, y], implies[and[member[x, domain[r]],
             member[y, domain[r]], subclass[APPLY[funpart[r], x], APPLY[funpart[r], y]]],
            equal[APPLY[funpart[r], x], APPLY[funpart[r], y]]], r → RATIO]]

Out[10]= composite[id[cart[intersection[Z, complement[set[id[omega]]]], Z]],
           intersection[complement[inverse[RATIO]], composite[inverse[RATIO], S]], RATIO] == 0

In[11]:= composite[id[cart[intersection[Z, complement[set[id[omega]]]], Z]],
           intersection[complement[inverse[RATIO]], composite[inverse[RATIO], S]], RATIO] := 0
```

Theorem.

```
In[12]:= Map[empty[composite[#, inverse[RATIO]]] &,
           Assoc[RATIO, id[cart[intersection[Z, complement[set[id[omega]]]], Z]],
            composite[intersection[complement[inverse[RATIO]],
              composite[inverse[RATIO], S]], RATIO]]] // Reverse

Out[12]= equal[0, intersection[RATS, image[PS, RATS]]] == True

In[13]:= intersection[RATS, image[PS, RATS]] := 0
```

Corollary.

```
In[14]:= equal[composite[id[RATS], S, id[RATS]], id[RATS]]

Out[14]= True

In[15]:= composite[id[RATS], S, id[RATS]] := id[RATS]
```

Corollary. Reintroducing variables.

```
In[16]:= Map[implies[#, equal[x, y]] &,
          SubstTest[member, pair[x, y], composite[id[t], S, id[t]], t → RATS]]

Out[16]= or[equal[x, y], not[member[x, RATS]], not[member[y, RATS]], not[subclass[x, y]]] == True

In[17]:= or[equal[x_, y_], not[member[x_, RATS]],
            not[member[y_, RATS]], not[subclass[x_, y_]]] := True
```