rat[x] wrapper

Johan G. F. Belinfante
2012 August 24

```
In[1]:= SetDirectory["l:"]; << goedel.12aug22a
```

```
:Package Title: goedel.12aug22a                2012 August 22 at 7:00 p.m.

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

It is now:  2012 Aug 24 at 10:11

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 24 at 10:26
```

summary

A wrapper for rational numbers is defined, and a few basic facts about it are derived.

case rules

Some general rewrite rules for **case[p]** needed later are derived in this section.

Theorem.

```
In[2]:= case[not[member[x, y]]] // DoubleComplement
```

```
Out[2]= case[not[member[x, y]]] == complement[image[V, intersection[y, set[x]]]]
```

```
In[3]:= case[not[member[x_, y_]]] := complement[image[V, intersection[y, set[x]]]]
```

Lemma. A basic membership rule.

```
In[4]:= Map[equal[V, #] &, Map[image[V, intersection[z, #]] &,
          set[union[intersection[x, image[V, w]], intersection[y, complement[image[V, w]]]]] //
           Renormality]]
```

```
Out[4]= member[union[intersection[x, image[V, w]], intersection[y, complement[image[V, w]]]],
          z] == or[and[equal[0, w], member[y, z]], and[member[x, z], not[equal[0, w]]]]
```

```
In[5]:= member[union[intersection[x_, image[V, w_]],
           intersection[y_, complement[image[V, w_]]]], z_] :=
         or[and[equal[0, w], member[y, z]], and[member[x, z], not[equal[0, w]]]]
```

Theorem. A simplification rule.

```
In[6]:= image[x, case[p]] // Normality

Out[6]= image[x, case[p]] == intersection[case[p], range[x]]
```

```
In[7]:= image[x_, case[p_]] := intersection[case[p], range[x]]
```

Corollary. A general membership rule for case-based definitions.

```
In[8]:= SubstTest[member, union[intersection[x, image[V, w]],
           intersection[y, complement[image[V, w]]]], z, w → case[p]] // Reverse

Out[8]= member[union[intersection[x, case[p]], intersection[y, case[not[p]]]], z] ==
         or[and[p, member[x, z]], and[member[y, z], not[p]]]
```

```
In[9]:= member[union[intersection[x_, case[p_]], intersection[y_, case[not[p_]]]], z_] :=
         or[and[p, member[x, z]], and[member[y, z], not[p]]]
```

Observation. The expression **(x ⋂ case[p]) ⋃ (y ⋂ case[not[p]])** is equal to **x** if **p** is true, and equal to **y** if **p** is not true.

---

definition of the rat[x] wrapper

The rational number wrapper **rat[x]** is defined as follows.

```
In[10]:= union[id[intersection[Z, complement[image[V, intersection[RATS, set[x_]]]]]],
           intersection[x_, image[V, intersection[RATS, set[x_]]]]] := rat[x]
```

Lemma. A simplification rule.

```
In[11]:= SubstTest[image, x, case[p], x → inverse[DUP]] // Reverse

Out[11]= fix[case[p]] == case[p]
```

```
In[12]:= fix[case[p_]] := case[p]
```

Observation. The definition of **rat[x]** can be written as a case-based definition.

```
In[13]:= union[intersection[x, case[member[x, RATS]]],
           intersection[id[Z], case[not[member[x, RATS]]]]]

Out[13]= rat[x]
```

This says that **rat[x]** is equal to **x** if **x** is rational, and is equal to **id[Z]** if **x** is not rational. Since **id[Z]** is a rational number, it follows that **rat[x]** is a rational number whether or not **x** is.

wrapper introduction and removal rules

The most basic properties for any wrapper are its introduction and removal rules. The following introduction rule allows one to convert statements containing membership literals of the form **x ∈ RATS** with shorter statements that involve wrapped variables of the form **rat[x]**. Some examples will be given in the last section.

Theorem. (Wrapper introduction rule.)

```
In[14]:= SubstTest[member,
           union[intersection[x, image[V, w]], intersection[y, complement[image[V, w]]]],
           RATS, {y → id[Z], w → intersection[RATS, set[x]]}] // Reverse
Out[14]= member[rat[x], RATS] == True

In[15]:= member[rat[x_], RATS] := True
```

Corollary. Rational numbers are sets.

```
In[16]:= SubstTest[implies, member[u, v], member[u, V], {u → rat[x], v → RATS}] // Reverse
Out[16]= member[rat[x], V] == True

In[17]:= member[rat[x_], V] := True
```

The following simplification rule will be used to derive the wrapper removal rule.

Lemma. Simplification rule.

```
In[18]:= equiv[or[equal[x, id[Z]], member[x, RATS]], member[x, RATS]]
Out[18]= True

In[19]:= or[equal[x_, id[Z]], member[x_, RATS]] := member[x, RATS]
```

The following wrapper removal rule is useful for replacing statements involving **rat[x]** with equivalent statements with literals **x ∈ RATS**.

Theorem. Wrapper removal rule.

```
In[20]:= SubstTest[member,
           union[intersection[x, image[V, w]], intersection[y, complement[image[V, w]]]],
           set[x], {y → id[Z], w → intersection[RATS, set[x]]}] // Reverse
Out[20]= equal[x, rat[x]] == member[x, RATS]

In[21]:= equal[x_, rat[x_]] := member[x, RATS]
```

Theorem. Automatic removal rule.

```
In[22]:= implies[member[x, RATS], equal[rat[x], x]]

Out[22]= True

In[23]:= rat[x_] := x /; member[x, RATS]
```

reify rule for rat[x]

Theorem. (Reify rule.)

```
In[24]:= SubstTest[reify, x, union[intersection[f[x], image[V, g[x]]],
            intersection[w, complement[image[V, g[x]]]]],
           {g[x] → intersection[RATS, set[f[x]]], w → id[Z]}] // Reverse

Out[24]= reify[x, rat[f[x]]] ==
          union[cart[complement[image[inverse[VERTSECT[reify[x, f[x]]]], RATS]], id[Z]],
           composite[reify[x, f[x]], id[image[inverse[VERTSECT[reify[x, f[x]]]], RATS]]]]

In[25]:= reify[x_, rat[y_]] :=
          union[cart[complement[image[inverse[VERTSECT[reify[x, y]]], RATS]], id[Z]],
           composite[reify[x, y], id[image[inverse[VERTSECT[reify[x, y]]], RATS]]]]
```

The following example is the function **λ x. rat[x]** which takes any set **x** to the corresponding rational number **rat[x]**. It is the union of the identity function on the set **RATS** of rational numbers, and the constant function with value **id[Z]** on the complement of **RATS**.

```
In[26]:= VERTSECT[reify[x, rat[x]]]

Out[26]= union[cart[complement[RATS], set[id[Z]]], id[RATS]]
```

Theorem. The class **(complement[RATS] × {id[Z]}) ⋃ id[RATS]** is a function.

```
In[27]:= SubstTest[subclass, composite[t, inverse[t]], Id,
           t -> union[cart[complement[RATS], set[id[Z]]], id[RATS]]]

Out[27]= FUNCTION[union[cart[complement[RATS], set[id[Z]]], id[RATS]]] == True

In[28]:= FUNCTION[union[cart[complement[RATS], set[id[Z]]], id[RATS]]] := True
```

properties of rat[x]

The main value of the wrapper **rat[x]** is the ability to make concise statements about rational numbers without having to introduce membership literals of the form **x ∈ RATS**. A few examples are considered in this section.

Theorem. Rational numbers are not empty.

```
In[29]:= Map[not, SubstTest[implies, member[t, RATS], not[empty[t]], t → rat[x]]] // Reverse
Out[29]= equal[0, rat[x]] == False

In[30]:= equal[0, rat[x_]] := False
```

Theorem. Rational numbers are functions.

```
In[31]:= SubstTest[implies, member[t, RATS], FUNCTION[t], t → rat[x]] // Reverse
Out[31]= FUNCTION[rat[x]] == True

In[32]:= FUNCTION[rat[x_]] := True
```

Theorem. The domain of any rational number is countably infinite.

```
In[33]:= SubstTest[implies, member[t, RATS], equal[card[domain[t]], omega], t → rat[x]] // Reverse
Out[33]= equal[omega, card[domain[rat[x]]]] == True

In[34]:= card[domain[rat[x_]]] := omega
```

Theorem. Rational numbers are countably infinite sets.

```
In[35]:= SubstTest[card, domain[funpart[t]], t → rat[x]]
Out[35]= card[rat[x]] == omega

In[36]:= card[rat[x_]] := omega
```

The following corollary is obtained by using the wrapper removal rule.

Corollary.

```
In[38]:= SubstTest[implies, equal[x, rat[t]], equal[card[x], omega], t → x] // Reverse
Out[38]= or[equal[omega, card[x]], not[member[x, RATS]]] == True

In[39]:= or[equal[omega, card[x_]], not[member[x_, RATS]]] := True
```