ratplus[x]

Johan G. F. Belinfante
2012 November 21

```
In[1]:= SetDirectory["l:"]; << goedel.12nov17a
        :Package Title: goedel.12nov17a              2012 November 17 at 9:40 a.m.
        Loading takes about sixteen minutes, half that time due to builtin pauses.
        It is now:  2012 Nov 21 at 4:40
        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 Nov 21 at 4:56
```

summary

The function **ratplus[x]** is defined and some of its properties are derived.

derivation

Definition.

```
In[2]:= composite[RATADD, LEFT[x_]] := ratplus[x]
```

Theorem. Simplification rule.

```
In[3]:= Assoc[RATADD, SWAP, LEFT[x]]
Out[3]= composite[RATADD, RIGHT[x]] == ratplus[x]
```

```
In[4]:= composite[RATADD, RIGHT[x_]] := ratplus[x]
```

Theorem. The class **ratplus[x]** is a function.

```
In[5]:= SubstTest[FUNCTION, composite[binop[t], LEFT[x]], t → RATADD] // Reverse
Out[5]= FUNCTION[ratplus[x]] == True
```

```
In[6]:= FUNCTION[ratplus[x_]] := True
```

Theorem. The domain of **ratplus[x]**.

```
In[7]:= IminComp[RATADD, LEFT[x], V]
Out[7]= domain[ratplus[x]] == intersection[RATS, image[V, intersection[RATS, set[x]]]]
In[8]:= domain[ratplus[x_]] := intersection[RATS, image[V, intersection[RATS, set[x]]]]
```

Corollary.

```
In[9]:= Assoc[ratplus[x], id[domain[ratplus[x]]], id[RATS]] // Reverse
Out[9]= composite[ratplus[x], id[RATS]] == ratplus[x]
In[10]:= composite[ratplus[x_], id[RATS]] := ratplus[x]
```

Theorem.

```
In[11]:= SubstTest[empty, composite[t, LEFT[x]], t → RATADD] // Reverse
Out[11]= equal[0, ratplus[x]] == not[member[x, RATS]]
In[12]:= equal[0, ratplus[x_]] := not[member[x, RATS]]
```

Theorem. Function application rule.

```
In[13]:= SubstTest[APPLY, composite[binop[t], LEFT[x]], y, t → RATADD] // Reverse
Out[13]= APPLY[ratplus[x], y] == ratadd[x, y]
In[14]:= APPLY[ratplus[x_], y_] := ratadd[x, y]
```

Theorem. Vertical section rule.

```
In[15]:= SubstTest[image, funpart[t], set[y], t → ratplus[x]] // Reverse
Out[15]= image[ratplus[x], set[y]] == set[ratadd[x, y]]
In[16]:= image[ratplus[x_], set[y_]] := set[ratadd[x, y]]
```

Theorem.

```
In[17]:= Assoc[composite[RATADD, cross[Id, RATADD]], ASSOC, RIGHT[x]] // Reverse
Out[17]= composite[ratplus[x], RATADD] == composite[RATADD, cross[Id, ratplus[x]]]
In[18]:= composite[ratplus[x_], RATADD] := composite[RATADD, cross[Id, ratplus[x]]]
```

Theorem. Rule for composites.

```
In[19]:= Assoc[ratplus[x], RATADD, RIGHT[y]]
Out[19]= composite[ratplus[x], ratplus[y]] == ratplus[ratadd[x, y]]
In[20]:= composite[ratplus[x_], ratplus[y_]] := ratplus[ratadd[x, y]]
```

The following rule is a replacement for an existing rewrite rule that will be removed.

Theorem. Replacement rule.

```
In[21]:= SubstTest[composite, RATADD, LEFT[t], t → e[RATADD]]
Out[21]= ratplus[cart[Z, set[id[omega]]]] == id[RATS]

In[22]:= ratplus[cart[Z, set[id[omega]]]] := id[RATS]
```

Lemma. Simplication rule.

```
In[23]:= composite[inverse[LEFT[x]], inverse[RATADD]] // DoubleInverse
Out[23]= composite[inverse[LEFT[x]], inverse[RATADD]] == inverse[ratplus[x]]

In[24]:= composite[inverse[LEFT[x_]], inverse[RATADD]] := inverse[ratplus[x]]
```

Lemma. Simplication rule.

```
In[25]:= composite[inverse[RIGHT[x]], inverse[RATADD]] // DoubleInverse
Out[25]= composite[inverse[RIGHT[x]], inverse[RATADD]] == inverse[ratplus[x]]

In[26]:= composite[inverse[RIGHT[x_]], inverse[RATADD]] := inverse[ratplus[x]]
```

Theorem.

```
In[27]:= Map[FUNCTION, SubstTest[composite, flip[rotate[t]], LEFT[x], t → RATADD]]
Out[27]= FUNCTION[inverse[ratplus[x]]] == True

In[28]:= FUNCTION[inverse[ratplus[x_]]] := True
```