# domain[RATIO]

Johan G. F. Belinfante
2012 June 3

```
In[1]:= SetDirectory["l:"]; << goedel.12may27a

        :Package Title: goedel.12may27a                    2012 May 27 at 4:45 a.m.

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

        It is now:  2012 Jun 3 at 5:23

        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 Jun 3 at 5:40
```

## summary

A function **RATIO** is defined, which can be regarded as mapping each ordered pair **(d, n)** of integers, with **d** non-zero, to the corresponding fraction **n/d**. The domain of **RATIO** is shown to be **(Z – {id[ω]}) × Z**.

## definition of RATIO

The class **RATIO** is defined by the following equation.

```
In[2]:= composite[id[FUNS], COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]] := RATIO
```

## properties of RATIO

Theorem. The class **RATIO** is a function.

```
In[3]:= SubstTest[FUNCTION, composite[id[t], COMPOSE,
           cross[composite[INVERSE, INTTIMES], INTTIMES]], t → FUNS] // Reverse

Out[3]= FUNCTION[RATIO] == True

In[4]:= FUNCTION[RATIO] := True
```

Lemma. A sethood result.

```
In[5]:= member[composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]], V] //
         AssertTest

Out[5]= member[composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]], V] == True

In[6]:= member[composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]], V] := True
```

Theorem. The class **RATIO** is a set.

```
In[7]:= SubstTest[member, composite[id[FUNS], setpart[t]], V,
          t -> composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]]] // Reverse

Out[7]= member[RATIO, V] == True

In[8]:= member[RATIO, V] := True
```

## domain

The following is a key observation that forms the basis for computing the domain of **RATIO**.

```
In[9]:= FUNCTION[composite[inverse[inttimes[x]], inttimes[y]]]

Out[9]= or[not[equal[x, id[omega]]], not[member[y, Z]]]
```

A second key idea used to compute the domain of **RATIO** is the following:

```
In[10]:= member[APPLY[RATIO, x], V]

Out[10]= member[x, domain[RATIO]]
```

A formula for **APPLY[RATIO, x]** will be derived that involves **case**, and then following lemmas will be needed.

Lemma. A special **case** formula.

```
In[11]:= SubstTest[case, member[composite[inverse[inttimes[x]], inttimes[y]], t], t → FUNS]

Out[11]= image[V, intersection[FUNS, set[composite[inverse[inttimes[x]], inttimes[y]]]]] ==
          case[or[not[equal[x, id[omega]]], not[member[y, Z]]]]

In[12]:= image[V, intersection[FUNS, set[composite[inverse[inttimes[x_]], inttimes[y_]]]]] :=
          case[or[not[equal[x, id[omega]]], not[member[y, Z]]]]
```

Theorem. A sethood simplification rule for **case**.

```
In[13]:= SubstTest[and, member[x, V], or[equal[0, x], equal[V, x]], x → case[p]] // Reverse

Out[13]= member[case[p], V] == not[p]

In[14]:= member[case[p_], V] := not[p]
```

Lemma.

```
In[15]:= Map[member[#, V] &, ApComp[composite[id[FUNS], COMPOSE],
           cross[composite[INVERSE, INTTIMES], INTTIMES], x]] // Reverse

Out[15]= member[x, domain[RATIO]] ==
          and[member[first[x], Z], member[second[x], Z], not[equal[first[x], id[omega]]]]

In[16]:= member[x_, domain[RATIO]] :=
          and[member[first[x], Z], member[second[x], Z], not[equal[first[x], id[omega]]]]
```

Theorem. The domain of the function **RATIO** is the cartesian product of the set of non-zero integers with the set of integers.

```
In[17]:= domain[RATIO] // Normality

Out[17]= domain[RATIO] == cart[intersection[Z, complement[set[id[omega]]]], Z]

In[18]:= domain[RATIO] := cart[intersection[Z, complement[set[id[omega]]]], Z]
```

## simplification rules

Theorem. A simplification rule for **RATIO**.

```
In[19]:= Assoc[RATIO, id[domain[RATIO]], id[cart[Z, Z]]] // Reverse

Out[19]= composite[RATIO, id[cart[Z, Z]]] == RATIO

In[20]:= composite[RATIO, id[cart[Z, Z]]] := RATIO
```

Corollary.

```
In[21]:= Assoc[RATIO, id[cart[Z, Z]], id[cart[V, V]]] // Reverse

Out[21]= composite[RATIO, id[cart[V, V]]] == RATIO

In[22]:= composite[RATIO, id[cart[V, V]]] := RATIO
```

Theorem. A simplification rule for **RATIO**.

```
In[23]:= Assoc[id[FUNS], id[FUNS],
          composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]]

Out[23]= composite[id[FUNS], RATIO] == RATIO

In[24]:= composite[id[FUNS], RATIO] := RATIO
```

Corollary.

```
In[27]:= SubstTest[subclass, composite[id[FUNS], t], cart[V, FUNS], t → RATIO] // Reverse

Out[27]= subclass[RATIO, cart[V, FUNS]] == True
```

```
In[28]:= % /. Equal → SetDelayed
```

## another formula for RATIO

In this section, another formula for **RATIO** is derived.

Lemma.

```
In[29]:= Map[subclass[RATIO, #] &, Assoc[id[FUNS], composite[COMPOSE,
           cross[composite[INVERSE, INTTIMES], INTTIMES]], id[domain[RATIO]]]]

Out[29]= subclass[RATIO, composite[COMPOSE, cross[composite[INVERSE, INTTIMES,
             id[intersection[Z, complement[set[id[omega]]]]]], INTTIMES]]] == True

In[30]:= % /. Equal → SetDelayed
```

Theorem. Another formula for **RATIO**.

```
In[31]:= SubstTest[implies, and[subclass[x, y], FUNCTION[y]],
           equal[x, composite[y, id[domain[x]]]],
           {x → RATIO, y -> composite[COMPOSE, cross[composite[INVERSE, INTTIMES,
                id[intersection[Z, complement[set[id[omega]]]]]], INTTIMES]]}] // Reverse

Out[31]= equal[RATIO, composite[COMPOSE, cross[composite[INVERSE, INTTIMES,
             id[intersection[Z, complement[set[id[omega]]]]]], INTTIMES]]] == True

In[32]:= composite[COMPOSE, cross[composite[INVERSE, INTTIMES,
             id[intersection[Z, complement[set[id[omega]]]]]], INTTIMES]] := RATIO
```

## an APPLY formula

Theorem. An **APPLY** formula for **RATIO** using **int** wrappers.

```
In[35]:= Map[implies[not[equal[int[x], id[omega]]],
             equal[#, composite[inverse[inttimes[int[x]]], inttimes[int[y]]]]] &,
           ApComp[composite[COMPOSE, cross[composite[INVERSE, INTTIMES], INTTIMES]],
            id[domain[RATIO]], PAIR[int[x], int[y]]] // Reverse

Out[35]= or[equal[APPLY[RATIO, PAIR[int[x], int[y]]],
             composite[inverse[inttimes[int[x]]], inttimes[int[y]]]],
           equal[id[omega], int[x]]] == True

In[36]:= or[equal[APPLY[RATIO, PAIR[int[x_], int[y_]]],
             composite[inverse[inttimes[int[x_]]], inttimes[int[y_]]]],
           equal[id[omega], int[x_]]] := True
```

The following corollary restates this without the **int** wrappers.

Corollary.

```
In[37]:= SubstTest[implies, and[equal[x, int[u]], equal[y, int[v]]],
           or[equal[APPLY[RATIO, PAIR[x, y]], composite[inverse[inttimes[x]], inttimes[y]]],
            equal[id[omega], x]], {u → x, v → y}] // Reverse

Out[37]= or[equal[x, id[omega]],
           equal[APPLY[RATIO, PAIR[x, y]], composite[inverse[inttimes[x]], inttimes[y]]],
           not[member[x, Z]], not[member[y, Z]]] == True

In[38]:= or[equal[APPLY[RATIO, PAIR[x_, y_]], composite[inverse[inttimes[x_]], inttimes[y_]]],
           equal[id[omega], x_], not[member[x_, Z]], not[member[y_, Z]]] := True
```