natdiv[x, y], part 1: basics

Johan G. F. Belinfante
2007 February 4

```
In[1]:= SetDirectory["l:"]; << goedel90.03a; << tools.m

        :Package Title: goedel90.03a        2007 February 3 at 12:05 noon

        It is now:  2007 Feb 9 at 18:2

        Loading Simplification Rules

        TOOLS.M                      Revised 2007 January 7

        weightlimit = 40

In[2]:= Begin["Goedel`Private`"];
```

summary

The following new membership rule for **natdiv[x,y]** has been added to the **GOEDEL** program:

```
In[3]:= ?? natdiv

        natdiv[x,y] is the quotient when a natural number x is divided by y when x is divisible by y

In[4]:= FirstMatch[class[t_, member[w_, HoldPattern[natdiv[x_, y_]]]]]

Out[4]= class[u_, member[v_, natdiv[x_, y_]]] := Module[{w = Unique[]}, class[u, and[member[
          v, V], forall[w, implies[member[pair[pair[y, w], x], NATMUL], member[v, w]]]]]]
```

Comment: It is not necessary to wrap **NATMUL** in this definition with **HoldPattern** because there is currently no membership rule at all in the **GOEDEL** program for **NATMUL**, but just a rewrite rule for **image[V, intersection[NATMUL, set[x]]]**, which serves as a substitute. To save writing, the class **natdiv[x,y]** will often informally be denoted by **x/y** in this notebook.

natdiv replacement rules

Currently the **GOEDEL** program contains a few rewrite rules for **APPLY[inverse[times[x]], y]**, which must all be replaced prior to normalizing the class **natdiv[x,y]** to avoid having to rederive all these rules. The following temporary rewrite rule will be used for this task.

```
In[5]:= equal[union[intersection[APPLY[inverse[times[x]], y], image[V, x]],
          intersection[APPLY[inverse[times[x]], y], image[V, y]]], APPLY[inverse[times[x]], y]]

Out[5]= True
```

```
In[6]:= union[intersection[APPLY[inverse[times[x_]], y_], image[V, x_]], intersection[
          APPLY[inverse[times[x_]], y_], image[V, y_]]] := APPLY[inverse[times[x]], y]
```

The class **natdiv[x, y]** is equal to **V** when **x** is not divisible by **y**. One can think of the class **V** here informally as a substitute for "meaningless."

```
In[7]:= Map[equal[V, #] &, natdiv[x, y] // Normality]
Out[7]= equal[V, natdiv[x, y]] == not[member[pair[y, x], DIV]]

In[8]:= equal[V, natdiv[x_, y_]] := not[member[pair[y, x], DIV]]
```

If **x** is divisible by **y**, then **natdiv[x, y]** is a natural number.

```
In[9]:= Map[member[#, omega] &, natdiv[x, y] // Normality]
Out[9]= member[natdiv[x, y], omega] == member[pair[y, x], DIV]

In[10]:= member[natdiv[x_, y_], omega] := member[pair[y, x], DIV]
```

Corollary.

```
In[11]:= Map[member[#, V] &, natdiv[x, y] // Normality]
Out[11]= member[natdiv[x, y], V] == member[pair[y, x], DIV]

In[12]:= member[natdiv[x_, y_], V] := member[pair[y, x], DIV]
```

special equations

Theorem.

```
In[13]:= Map[equal[0, #] &, natdiv[x, y] // Normality]
Out[13]= equal[0, natdiv[x, y]] == and[equal[0, x], member[y, omega]]

In[14]:= equal[0, natdiv[x_, y_]] := and[equal[0, x], member[y, omega]]
```

Theorem.

```
In[15]:= Map[equal[set[0], #] &, natdiv[x, y] // Normality]
Out[15]= equal[natdiv[x, y], set[0]] == and[equal[x, y], member[y, omega], not[equal[0, y]]]

In[16]:= equal[natdiv[x_, y_], set[0]] := and[equal[x, y], member[y, omega], not[equal[0, y]]]
```

Theorem.

```
In[17]:= Map[equal[x, natmul[#, y]] &, natdiv[x, y] // Normality]
Out[17]= equal[x, natmul[y, natdiv[x, y]]] == or[equal[V, x], member[pair[y, x], DIV]]
```

```
In[18]:= equal[x_, natmul[y_, natdiv[x_, y_]]] := or[equal[V, x], member[pair[y, x], DIV]]
```

Lemma.

```
In[19]:= Map[implies[#, equal[x, natdiv[x, set[0]]]] &,
           SubstTest[equal, x, natmul[y, natdiv[x, y]], y → set[0]]] // MapNotNot

Out[19]= or[equal[x, natdiv[x, set[0]]], not[member[x, omega]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[21]:= equiv[equal[x, natdiv[x, set[0]]], or[equal[V, x], member[x, omega]]] // not // not

Out[21]= True

In[22]:= equal[x_, natdiv[x_, set[0]]] := or[equal[V, x], member[x, omega]]
```

special inequalities

```
In[23]:= Map[member[0, #] &, natdiv[x, y] // Normality]

Out[23]= member[0, natdiv[x, y]] == or[not[equal[0, x]], not[member[pair[y, x], DIV]]]

In[24]:= member[0, natdiv[x_, y_]] := or[not[equal[0, x]], not[member[pair[y, x], DIV]]]

In[25]:= Map[member[set[0], #] &, natdiv[x, y] // Normality]

Out[25]= member[set[0], natdiv[x, y]] == or[member[y, x], not[member[pair[y, x], DIV]]]

In[26]:= member[set[0], natdiv[x_, y_]] := or[member[y, x], not[member[pair[y, x], DIV]]]
```

normalization

At this point **natdiv[x, y]** can be normalized:

```
In[27]:= natdiv[y, x] // Normality // Reverse

Out[27]= APPLY[inverse[times[x]], y] == natdiv[y, x]

In[28]:= APPLY[inverse[times[x_]], y_] := natdiv[y, x]
```

A temporary rewrite rule introduced earlier can now be replaced with the following:

```
In[29]:= equal[union[intersection[image[V, x], natdiv[x, y]],
           intersection[image[V, y], natdiv[x, y]]], natdiv[x, y]]

Out[29]= True
```

```
In[30]:= union[intersection[image[V, x_], natdiv[x_, y_]],
            intersection[image[V, y_], natdiv[x_, y_]]] := natdiv[x, y]
```

replacing a vertical section rule

First, the old rule is removed:

```
In[31]:= image[inverse[times[x_]], set[y_]] =.
```

Lemma.

```
In[32]:= SubstTest[image, funpart[w], set[y], w → inverse[times[x]]] // Reverse

Out[32]= intersection[image[V, x], image[inverse[times[x]], set[y]]] ==
          intersection[image[V, x], set[natdiv[y, x]]]

In[33]:= intersection[image[V, x_], image[inverse[times[x_]], set[y_]]] :=
          intersection[image[V, x], set[natdiv[y, x]]]
```

Lemma.

```
In[34]:= equal[intersection[complement[image[V, x]], image[inverse[times[x]], set[y]]],
          intersection[omega, complement[image[V, x]], complement[image[V, y]]]]

Out[34]= True

In[35]:= intersection[complement[image[V, x_]], image[inverse[times[x_]], set[y_]]] :=
          intersection[omega, complement[image[V, x]], complement[image[V, y]]]
```

Theorem. (Replacement rule.)

```
In[36]:= Map[equal[#, union[set[natdiv[y, x]],
             intersection[omega, complement[image[V, x]], complement[image[V, y]]]]] &,
           SubstTest[union, intersection[u, v], intersection[u, complement[v]],
            {u -> image[inverse[times[x]], set[y]], v → image[V, x]}]]

Out[36]= equal[image[inverse[times[x]], set[y]],
           union[intersection[omega, complement[image[V, x]], complement[image[V, y]]],
            set[natdiv[y, x]]]] == True

In[37]:= image[inverse[times[x_]], set[y_]] := union[intersection[omega,
             complement[image[V, x]], complement[image[V, y]]], set[natdiv[y, x]]]
```

special values

Theorem. One cannot ordinarily divide by **0**, but this is allowed in one very special case: **0/0 = 0**.

```
In[38]:= SubstTest[APPLY, inverse[times[w]], x, w → 0]

Out[38]= natdiv[x, 0] == image[V, x]

In[39]:= natdiv[x_, 0] := image[V, x]
```

Theorem. The quotient **x/1** is **x** when **x** is a natural number, and meaningless otherwise, provided that "meaningless" is interpreted as before.

```
In[40]:= SubstTest[APPLY, inverse[times[w]], x, w → set[0]]

Out[40]= natdiv[x, set[0]] == union[x, complement[image[V, intersection[omega, set[x]]]]]

In[41]:= natdiv[x_, set[0]] := union[x, complement[image[V, intersection[omega, set[x]]]]]
```

Theorem. The quotient **0/x** is **0** if **x** is a natural number, and meaningless otherwise. Note that the case **x = 0** need not be excluded here.

```
In[42]:= natdiv[0, x] // Normality

Out[42]= natdiv[0, x] == complement[image[V, intersection[omega, set[x]]]]

In[43]:= natdiv[0, x_] := complement[image[V, intersection[omega, set[x]]]]
```

Theorem. The quotient **1/x** is **1** if **x = 1**, and meaningless otherwise.

```
In[44]:= equal[V, natdiv[set[0], x]]

Out[44]= not[equal[x, set[0]]]

In[45]:= natdiv[set[0], x] // Normality

Out[45]= natdiv[set[0], x] == union[complement[image[V, intersection[x, set[0]]]],
           image[V, intersection[x, complement[set[0]]], set[0]]

In[46]:= natdiv[set[0], x_] := union[complement[image[V, intersection[x, set[0]]]],
           image[V, intersection[x, complement[set[0]]], set[0]]
```

Comment. With this rewrite rule in place, to obtain the same result for the equation **equal[V, natdiv[set[0], x]]** as before, one needs the following new rewrite rule:

```
In[47]:= or[not[member[x, y]], not[subclass[y, set[x]]]] // NotNotTest

Out[47]= or[not[member[x, y]], not[subclass[y, set[x]]]] ==
          or[not[equal[y, set[x]]], not[member[x, V]]]

In[48]:= or[not[member[x_, y_]], not[subclass[y_, set[x_]]]] :=
          or[not[equal[y, set[x]]], not[member[x, V]]]
```

divisibility rules

Lemma. If **x** is a divisor of **y**, then so is **y/x**.

```
In[49]:= SubstTest[implies, and[member[x, omega], member[v, omega], equal[natmul[x, v], w]],
           member[pair[v, w], DIV],
           {v → natdiv[y, x], w → y}] // Reverse
```

```
Out[49]= or[member[pair[natdiv[y, x], y], DIV], not[member[pair[x, y], DIV]]] == True
```

```
In[50]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The converse is also true:

```
In[51]:= SubstTest[implies, member[pair[t, y], DIV],
           member[t, omega], t → natdiv[y, x]] // Reverse
```

```
Out[51]= or[member[pair[x, y], DIV], not[member[pair[natdiv[y, x], y], DIV]]] == True
```

```
In[52]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. Combining the preceding two lemmas yields this rewrite rule.

```
In[53]:= equiv[member[pair[natdiv[x, y], x], DIV], member[pair[y, x], DIV]]
```

```
Out[53]= True
```

```
In[54]:= member[pair[natdiv[x_, y_], x_], DIV] := member[pair[y, x], DIV]
```

a cancellation law

In the expression **y/(y/x)**, one can cancel **y**, provided **x** divides **y** and **y** is not **0**.

```
In[55]:= Map[or[not[member[pair[natdiv[y, x], y], DIV]], #] &,
          SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
            {u -> natmul[natdiv[y, x], x], v → y,
             w -> natmul[natdiv[y, x], natdiv[y, natdiv[y, x]]]}] // MapNotNot // Reverse]
```

```
Out[55]= or[equal[0, y], equal[x, natdiv[y, natdiv[y, x]]], not[member[pair[x, y], DIV]]] == True
```

```
In[56]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[57]:= equal[natdiv[x, V], V] // assert
```

```
Out[57]= True
```

```
In[58]:= natdiv[x_, V] := V
```

Theorem.

```
In[59]:= equiv[equal[x, natdiv[y, natdiv[y, x]]], or[equal[V, x],
           and[member[pair[x, y], DIV], implies[equal[0, y], equal[0, x]]]]] // not // not

Out[59]= True

In[60]:= equal[x_, natdiv[y_, natdiv[y_, x_]]] := or[and[equal[0, x], equal[0, y]],
           and[member[pair[x, y], DIV], not[equal[0, y]]], equal[V, x]]
```

x/x

```
In[61]:= equal[natdiv[x, x], union[intersection[set[0], image[V, x]],
           complement[image[V, intersection[omega, set[x]]]]]]

Out[61]= True

In[62]:= natdiv[x_, x_] := union[
           complement[image[V, intersection[omega, set[x]]]], intersection[image[V, x], set[0]]]
```

the equation (x y)/y = x

Lemma.

```
In[63]:= member[pair[x, V], DIV] // AssertTest

Out[63]= member[pair[x, V], DIV] == False

In[64]:= member[pair[x_, V], DIV] := False
```

Lemma.

```
In[65]:= SubstTest[equal, t, natmul[y, natdiv[t, y]], t → natmul[x, y]] // Reverse

Out[65]= or[equal[0, y], equal[x, natdiv[natmul[x, y], y]],
           not[member[x, omega]], not[member[y, omega]]] == True

In[66]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Temporary rewrite rule.

```
In[67]:= equiv[equal[natdiv[natmul[x, y], y], x], or[and[equal[0, x], member[y, omega]],
           and[member[x, omega], member[y, omega], not[equal[0, y]]], equal[V, x]]] // not // not

Out[67]= True

In[68]:= equal[x_, natdiv[natmul[x_, y_], y_]] := or[and[equal[0, x], member[y, omega]],
           and[member[x, omega], member[y, omega], not[equal[0, y]]], equal[V, x]]
```

Corollary. A temporary simplification rule which uses **nat** wrappers.

```
In[69]:= equal[natdiv[natmul[nat[x], nat[y]], nat[y]], intersection[nat[x], image[V, nat[y]]]]

Out[69]= True

In[70]:= natdiv[natmul[nat[x_], nat[y_]], nat[y_]] := intersection[image[V, nat[y]], nat[x]]
```

Theorem. (The special rule suggests the following general rule, which the **GOEDEL** program recognizes as being true.)

```
In[71]:= equal[natdiv[natmul[x, y], y],
          union[intersection[x, image[V, y]], complement[image[V, intersection[omega, set[x]]]],
           complement[image[V, intersection[omega, set[y]]]]]]

Out[71]= True

In[72]:= natdiv[natmul[x_, y_], y_] := union[complement[image[V, intersection[omega, set[x]]]],
           complement[image[V, intersection[omega, set[y]]]], intersection[x, image[V, y]]]
```