divisibility and natdiv

Johan G. F. Belinfante
2007 April 3

```
In[1]:= SetDirectory["l:"]; << goedel92.03a; << tools.m

        :Package Title: goedel92.03a        2007 April 3 at 5:20 p.m.

        It is now:  2007 Apr 3 at 23:58

        Loading Simplification Rules

        TOOLS.M                       Revised 2007 March 25

        weightlimit = 40
```

summary

In this notebook two rewrite rules are derived that serve to automatically convert divisibility statements involving **natdiv** into equivalent statements involving **natmul**. These rewrite rules are both consequences of a cancellation law for statements of the form **x z | y z**.

```
In[2]:= member[pair[natmul[x, z], natmul[y, z]], DIV]

Out[2]= or[and[member[z, omega], member[pair[x, y], DIV]],
          and[equal[0, z], member[x, omega], member[y, omega]]]
```

simplification rules for divisibility

Theorem.

```
In[3]:= member[pair[x, image[V, y]], DIV] // AssertTest

Out[3]= member[pair[x, image[V, y]], DIV] == and[equal[0, y], member[x, omega]]

In[4]:= member[pair[x_, image[V, y_]], DIV] := and[equal[0, y], member[x, omega]]
```

Theorem.

```
In[5]:= member[pair[image[V, x], y], DIV] // AssertTest

Out[5]= member[pair[image[V, x], y], DIV] == and[equal[0, x], equal[0, y]]

In[6]:= member[pair[image[V, x_], y_], DIV] := and[equal[0, x], equal[0, y]]
```

Corollary.

```
In[8]:= SubstTest[member, pair[x, image[V, t]], DIV, t → complement[image[V, y]]] // Reverse

Out[8]= member[pair[x, complement[image[V, y]]], DIV] == and[member[x, omega], not[equal[0, y]]]

In[9]:= member[pair[x_, complement[image[V, y_]]], DIV] :=
         and[member[x, omega], not[equal[0, y]]]
```

Lemma.

```
In[11]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]], equal[z, nat[w]]],
            implies[and[member[pair[natmul[x, y], z], DIV], equal[0, y]], equal[0, z]],
            {u → x, v → y, w → z}] // MapNotNot // Reverse

Out[11]= or[equal[0, z], not[equal[0, y]], not[member[pair[natmul[x, y], z], DIV]]] == True

In[12]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[13]:= equiv[and[equal[0, x], member[pair[natmul[x, y], z], DIV]],
            and[equal[0, x], equal[0, z], member[y, omega]]] // not // not

Out[13]= True

In[14]:= and[equal[0, x_], member[pair[natmul[x_, y_], z_], DIV]] :=
          and[equal[0, x], equal[0, z], member[y, omega]]
```

Corollary.

```
In[15]:= or[not[equal[0, x]], not[member[pair[natmul[x, y], z], DIV]]] // NotNotTest

Out[15]= or[not[equal[0, x]], not[member[pair[natmul[x, y], z], DIV]]] ==
          or[not[equal[0, x]], not[equal[0, z]], not[member[y, omega]]]

In[16]:= or[not[equal[0, x_]], not[member[pair[natmul[x_, y_], z_], DIV]]] :=
          or[not[equal[0, x]], not[equal[0, z]], not[member[y, omega]]]
```

---

temporary rules involving natdiv

Lemma.

```
In[17]:= SubstTest[implies, member[pair[z, t], DIV],
            member[t, omega], t → natdiv[y, x]] // Reverse

Out[17]= or[member[pair[x, y], DIV], not[member[pair[z, natdiv[y, x]], DIV]]] == True

In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2],
           not[implies[p1, p3]], {p1 -> member[pair[x, natdiv[y, z]], DIV],
            p2 -> member[pair[z, y], DIV], p3 → member[z, omega]}]] // Reverse

Out[19]= or[member[z, omega], not[member[pair[x, natdiv[y, z]], DIV]]] == True

In[20]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[21]:= SubstTest[implies, member[pair[t, z], DIV],
          member[t, omega], t → natdiv[y, x]] // Reverse

Out[21]= or[member[pair[x, y], DIV], not[member[pair[natdiv[y, x], z], DIV]]] == True

In[22]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Corollary.

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2],
           not[implies[p1, p3]], {p1 -> member[pair[natdiv[x, y], z], DIV],
            p2 -> member[pair[y, x], DIV], p3 → member[y, omega]}]] // Reverse

Out[23]= or[member[y, omega], not[member[pair[natdiv[x, y], z], DIV]]] == True

In[24]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

first transformation rule

From the cancellation law, one finds:

```
In[25]:= Map[implies[#, member[pair[x, natdiv[y, z]], DIV]] &, SubstTest[member,
            pair[natmul[x, z], natmul[t, z]], DIV, t → natdiv[y, z]]] // MapNotNot // Reverse

Out[25]= or[member[pair[x, natdiv[y, z]], DIV], not[member[pair[z, y], DIV]],
          not[member[pair[natmul[x, z], intersection[y, image[V, z]]], DIV]]] == True

In[26]:= (% /. {x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

A lemma is needed to eliminate **image[V, z]**:

```
In[27]:= SubstTest[implies, and[equal[y, t], member[pair[natmul[x, z], y], DIV]], member[
            pair[natmul[x, z], t], DIV], t → intersection[y, image[V, z]]] // Reverse // MapNotNot

Out[27]= or[equal[0, z], member[pair[natmul[x, z], intersection[y, image[V, z]]], DIV],
          not[member[pair[natmul[x, z], y], DIV]]] == True

In[28]:= (% /. {x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

The following result contains a redundant literal.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, or[p3, p4]],
            implies[and[p2, p3], p5], not[implies[p1, or[p4, p5]]],
            {p1 -> member[pair[natmul[x, z], y], DIV], p2 -> member[pair[z, y], DIV],
             p3 -> member[pair[natmul[x, z], intersection[y, image[V, z]]], DIV],
             p4 → equal[0, z], p5 -> member[pair[x, natdiv[y, z]], DIV]}]] // Reverse

Out[29]= or[equal[0, z], member[pair[x, natdiv[y, z]], DIV],
           not[member[pair[natmul[x, z], y], DIV]]] == True

In[30]:= (% /. {x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

The redundant literal can be eliminated as follows.

```
In[31]:= Map[not, SubstTest[and, implies[p1, or[p2, p3]], implies[and[p1, p3], p4],
            implies[p1, p5], implies[and[p3, p4, p5], p2], not[implies[p1, p2]],
            {p1 -> member[pair[natmul[x, z], y], DIV], p2 -> member[pair[x, natdiv[y, z]], DIV],
             p3 -> equal[0, z], p4 → equal[0, y], p5 → member[x, omega]}]] // Reverse

Out[31]= or[member[pair[x, natdiv[y, z]], DIV], not[member[pair[natmul[x, z], y], DIV]]] == True

In[32]:= (% /. {x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

Lemma.

```
In[33]:= Map[implies[or[and[equal[0, y], equal[0, z], member[x, omega]],
              member[pair[x, natdiv[y, z]], DIV]], #] &, SubstTest[member,
            pair[natmul[x, z], natmul[t, z]], DIV, t → natdiv[y, z]]] // Reverse // MapNotNot

Out[33]= or[member[pair[natmul[x, z], intersection[y, image[V, z]]], DIV],
           not[member[pair[x, natdiv[y, z]], DIV]]] == True

In[34]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[35]:= SubstTest[implies, and[equal[y, t], member[pair[natmul[x, z], t], DIV]],
            member[pair[natmul[x, z], y], DIV],
            t -> intersection[y, image[V, z]]] // Reverse // MapNotNot

Out[35]= or[equal[0, z], member[pair[natmul[x, z], y], DIV],
           not[member[pair[natmul[x, z], intersection[y, image[V, z]]], DIV]]] == True

In[36]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[37]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, or[p3, p4]],
           implies[and[p1, p3], p5], implies[p1, p6], implies[and[p3, p5, p6], p4],
           not[implies[p1, p4]], {p1 -> member[pair[x, natdiv[y, z]], DIV],
            p2 -> member[pair[natmul[x, z], intersection[y, image[V, z]]], DIV],
            p3 -> equal[0, z], p4 -> member[pair[natmul[x, z], y], DIV],
            p5 → equal[0, y], p6 → member[x, omega]}]] // Reverse

Out[37]= or[member[pair[natmul[x, z], y], DIV], not[member[pair[x, natdiv[y, z]], DIV]]] == True

In[38]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

First Transformation Theorem.

```
In[39]:= equiv[member[pair[x, natdiv[y, z]], DIV], member[pair[natmul[x, z], y], DIV]]

Out[39]= True

In[40]:= member[pair[x_, natdiv[y_, z_]], DIV] := member[pair[natmul[x, z], y], DIV]
```

a second transformation law

Lemma.

```
In[41]:= SubstTest[member, pair[natmul[t, z], natmul[y, z]], DIV, t → natdiv[x, z]] // MapNotNot //
          Reverse

Out[41]= member[pair[union[complement[image[V, intersection[image[DIV, set[z]], set[x]]]],
             intersection[x, image[V, z]]], natmul[y, z]], DIV] ==
          or[and[equal[0, x], equal[0, z], member[y, omega]], member[pair[natdiv[x, z], y], DIV]]

In[42]:= member[pair[union[complement[image[V, intersection[image[DIV, set[z_]], set[x_]]]],
             intersection[x_, image[V, z_]]], natmul[y_, z_]], DIV] :=
          or[and[equal[0, x], equal[0, z], member[y, omega]], member[pair[natdiv[x, z], y], DIV]]
```

This takes quite a while, but it does eventually terminate.

```
In[43]:= Map[or[#, member[pair[x, natmul[y, z]], DIV]] &,
          ((SubstTest[implies, and[equal[x, t], member[pair[t, w], DIV]], member[pair[x, w],
                DIV], t → union[complement[image[V, intersection[image[DIV, set[z]], set[x]]]],
                 intersection[x, image[V, z]]]] // Reverse) /. w → natmul[y, z]) // MapNotNot]

Out[43]= or[member[pair[x, natmul[y, z]], DIV], not[member[pair[natdiv[x, z], y], DIV]]] == True

In[44]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Converse theorem. (Interchange the roles of **x** and **t**.)

```
In[45]:= (SubstTest[implies, and[equal[x, t], member[pair[x, w], DIV]], member[pair[t, w], DIV],
            t → union[complement[image[V, intersection[image[DIV, set[z]], set[x]]]],
              intersection[x, image[V, z]]]] // Reverse) /. w → natmul[y, z] // MapNotNot

Out[45]= or[equal[0, z], member[pair[natdiv[x, z], y], DIV],
           not[member[pair[x, natmul[y, z]], DIV]], not[member[pair[z, x], DIV]]] == True

In[46]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[47]:= implies[member[pair[natdiv[x, y], z], DIV],
           or[and[member[pair[x, natmul[y, z]], DIV], member[pair[y, x], DIV], not[equal[0, y]]],
            and[equal[0, x], equal[0, z], member[y, omega]]]] // NotNotTest

Out[47]= or[and[equal[0, x], equal[0, z], member[y, omega]],
           and[member[pair[x, natmul[y, z]], DIV], member[pair[y, x], DIV], not[equal[0, y]]],
           not[member[pair[natdiv[x, y], z], DIV]]] == True

In[48]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Second Transformation Theorem.

```
In[49]:= equiv[member[pair[natdiv[x, y], z], DIV], or[
             and[member[pair[x, natmul[y, z]], DIV], member[pair[y, x], DIV], not[equal[0, y]]],
             and[equal[0, x], equal[0, z], member[y, omega]]]] // not // not

Out[49]= True

In[50]:= member[pair[natdiv[x_, y_], z_], DIV] :=
          or[and[equal[0, x], equal[0, z], member[y, omega]],
           and[member[pair[x, natmul[y, z]], DIV], member[pair[y, x], DIV], not[equal[0, y]]]]
```

serendipity: other simplification rules

Simplification rule:

```
In[51]:= equiv[and[equal[0, y], member[pair[x, y], DIV]], and[equal[0, y], member[x, omega]]]

Out[51]= True

In[52]:= and[equal[0, y_], member[pair[x_, y_], DIV]] := and[equal[0, y], member[x, omega]]
```

Simplification rule:

```
In[53]:= equiv[or[member[pair[x, y], DIV], member[pair[natmul[x, z], y], DIV]],
           member[pair[x, y], DIV]]

Out[53]= True
```

```
In[54]:= or[member[pair[x_, y_], DIV], member[pair[natmul[x_, z_], y_], DIV]] :=
          member[pair[x, y], DIV]
```