n/d rules

Johan G. F. Belinfante
2007 February 1

```
In[1]:= SetDirectory["l:"]; << goedel89.30a; << tools.m

        :Package Title: goedel89.30a        2007 January 30 at 10:35 p.m.

        It is now:  2007 Feb 1 at 21:43

        Loading Simplification Rules

        TOOLS.M                      Revised 2007 January 7

        weightlimit = 40
```

summary

The class **APPLY[inverse[times[d]], n]** will be informally called **n/d** in this notebook. Some rewrite rules are derived in this notebook for various conditions of interest: **n/d = 0**, **n/d = 1**, **n/d > 0**, **n/d > 1**. A key step in several cases is the removal of redundant literals, essential for obtaining converses. Simpler rewrite rules can be formulated if one derives "if and only if" theorems.

temporary abbreviation

To save writing, the following temporary abbreviation will be used.

```
In[2]:= natdiv[x_, y_] := APPLY[inverse[times[y]], x]
```

trichotomy rules

Lemma.

```
In[3]:= SubstTest[implies,
          and[equal[x, nat[t]], member[set[0], x]], member[0, x], t → x] // Reverse

Out[3]= or[member[0, x], not[member[x, omega]], not[member[set[0], x]]] == True

In[4]:= or[member[0, x_], not[member[x_, omega]], not[member[set[0], x_]]] := True
```

Lemma.

```
In[5]:= SubstTest[implies, equal[x, nat[t]],
          or[equal[0, x], equal[x, set[0]], member[set[0], x]], t → x] // Reverse

Out[5]= or[equal[0, x], equal[x, set[0]], member[set[0], x], not[member[x, omega]]] == True

In[6]:= or[equal[0, x_], equal[x_, set[0]], member[set[0], x_], not[member[x_, omega]]] := True
```

---

a simplification rule about 0/0

Lemma.

```
In[7]:= equiv[and[equal[0, x], member[pair[x, y], DIV]], and[equal[0, x], equal[0, y]]]

Out[7]= True

In[8]:= and[equal[0, x_], member[pair[x_, y_], DIV]] := and[equal[0, x], equal[0, y]]
```

Contrapositively:

```
In[9]:= or[not[equal[0, x]], not[member[pair[x, y], DIV]]] // NotNotTest

Out[9]= or[not[equal[0, x]], not[member[pair[x, y], DIV]]] ==
         or[not[equal[0, x]], not[equal[0, y]]]

In[10]:= or[not[equal[0, x_]], not[member[pair[x_, y_], DIV]]] :=
          or[not[equal[0, x]], not[equal[0, y]]]
```

---

the condition y/x = 0

Lemma.

```
In[11]:= SubstTest[implies, equal[0, t], not[equal[V, t]], t → natdiv[y, x]] // Reverse

Out[11]= or[member[pair[x, y], DIV], not[equal[0, APPLY[inverse[times[x]], y]]]] == True

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

Lemma.

```
In[13]:= SubstTest[implies, and[FUNCTION[w], member[pair[u, v], w]], equal[APPLY[w, u], v],
           {u → nat[y], v → 0, w → inverse[times[x]]}] // Reverse

Out[13]= or[equal[0, x], equal[0, APPLY[inverse[times[x]], nat[y]]],
           not[equal[0, nat[y]]], not[member[x, omega]]] == True

In[14]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The **nat** wrapper can be removed as follows:

```
In[15]:= SubstTest[implies, equal[y, nat[t]],
           or[equal[0, x], equal[0, APPLY[inverse[times[x]], y]],
            not[equal[0, y]], not[member[x, omega]]], t → y] // Reverse // MapNotNot

Out[15]= or[equal[0, x], equal[0, APPLY[inverse[times[x]], y]],
           not[equal[0, y]], not[member[x, omega]]] == True

In[16]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The condition that **x** be nonzero is not needed, so the above can be simplified:

```
In[17]:= SubstTest[and, implies[p, q], or[p, q], {p -> equal[0, x],
           q -> or[equal[0, natdiv[y, x]], not[equal[0, y]], not[member[x, omega]]]}]

Out[17]= or[equal[0, APPLY[inverse[times[x]], y]],
           not[equal[0, y]], not[member[x, omega]]] == True

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

Lemma. (This can also be cleaned up.)

```
In[19]:= SubstTest[implies, and[equal[0, z], member[x, omega], equal[y, natmul[x, z]]],
           equal[0, y], z → natdiv[y, x]] // Reverse

Out[19]= or[equal[0, y], not[equal[0, APPLY[inverse[times[x]], y]]],
           not[member[pair[x, y], DIV]]] == True

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

Theorem.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 -> equal[0, natdiv[y, x]], p2 -> member[pair[x, y], DIV], p3 → equal[0, y]}]] //
          Reverse

Out[21]= or[equal[0, y], not[equal[0, APPLY[inverse[times[x]], y]]]] == True

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

Conversely:

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 -> equal[0, natdiv[y, x]], p2 -> member[pair[x, y], DIV], p3 → member[x, omega]}]] /
          Reverse

Out[23]= or[member[x, omega], not[equal[0, APPLY[inverse[times[x]], y]]]] == True

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

Combining the theorem and its converse yields an equivalence that can be made into a simple rewrite rule:

```
In[25]:= equiv[equal[0, APPLY[inverse[times[x]], y]],
            and[member[x, omega], equal[0, y]]] // not // not

Out[25]= True

In[26]:= equal[0, APPLY[inverse[times[x_]], y_]] := and[equal[0, y], member[x, omega]]
```

the condition y/x = 1

Lemma.

```
In[27]:= SubstTest[implies, equal[set[0], t], not[equal[V, t]], t → natdiv[y, x]] // Reverse

Out[27]= or[member[pair[x, y], DIV], not[equal[APPLY[inverse[times[x]], y], set[0]]]] == True

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

Lemma.

```
In[29]:= SubstTest[implies, and[FUNCTION[w], member[pair[u, v], w]], equal[APPLY[w, u], v],
           {u → y, v → set[0], w → inverse[times[x]]}] // Reverse

Out[29]= or[equal[0, x], equal[APPLY[inverse[times[x]], y], set[0]],
           not[equal[x, y]], not[member[x, omega]]] == True

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

Lemma. (This can be cleaned up further.)

```
In[31]:= SubstTest[implies, and[equal[set[0], z], member[x, omega], equal[y, natmul[x, z]]],
           equal[x, y], z → natdiv[y, x]] // Reverse

Out[31]= or[equal[x, y], not[equal[APPLY[inverse[times[x]], y], set[0]]],
           not[member[pair[x, y], DIV]]] == True

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

Theorem.

```
In[33]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 -> equal[set[0], natdiv[y, x]], p2 -> member[pair[x, y], DIV], p3 → equal[x, y]}]] /
          Reverse

Out[33]= or[equal[x, y], not[equal[APPLY[inverse[times[x]], y], set[0]]]] == True

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

Conversely:

```
In[35]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
           {p1 -> equal[set[0], natdiv[y, x]],
            p2 -> member[pair[x, y], DIV], p3 → member[x, omega]}]] // Reverse

Out[35]= or[member[x, omega], not[equal[APPLY[inverse[times[x]], y], set[0]]]] == True

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

Combining the two directions yields:

```
In[37]:= equiv[equal[APPLY[inverse[times[x]], y], set[0]],
           and[member[x, omega], equal[x, y], not[equal[0, x]]]] // not // not

Out[37]= True

In[38]:= equal[APPLY[inverse[times[x_]], y_], set[0]] :=
          and[equal[x, y], member[x, omega], not[equal[0, x]]]
```

the condition 0 < y/x

Lemma.

```
In[39]:= SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
           {u → 0, v → V, w → natdiv[y, x]}] // Reverse

Out[39]= or[member[0, APPLY[inverse[times[x]], y]], member[pair[x, y], DIV]] == True

In[40]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[41]:= SubstTest[implies, and[member[t, omega], not[equal[0, t]]],
           member[0, t], t → natdiv[y, x]] // Reverse

Out[41]= or[equal[0, y], member[0, APPLY[inverse[times[x]], y]],
           not[member[pair[x, y], DIV]]] == True

In[42]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (This has a redundant literal.)

```
In[43]:= SubstTest[implies, member[0, t], not[equal[0, t]], t → natdiv[y, x]] // Reverse

Out[43]= or[not[equal[0, y]],
           not[member[0, APPLY[inverse[times[x]], y]]], not[member[x, omega]]] == True

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

The redundant literal is removed as follows.

```
In[45]:= SubstTest[and, implies[p, q], or[p, q], {p -> member[pair[x, y], DIV],
           q -> or[equal[0, y], member[0, APPLY[inverse[times[x]], y]]]}]

Out[45]= or[equal[0, y], member[0, APPLY[inverse[times[x]], y]]] == True

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

Temporary rewrite rule.

```
In[47]:= member[0, APPLY[inverse[times[x]], 0]] // AssertTest

Out[47]= member[0, APPLY[inverse[times[x]], 0]] == not[member[x, omega]]

In[48]:= member[0, APPLY[inverse[times[x_]], 0]] := not[member[x, omega]]
```

Main theorem.

```
In[49]:= equiv[member[0, APPLY[inverse[times[x]], y]],
           or[not[equal[0, y]], not[member[pair[x, y], DIV]]]] // not // not

Out[49]= True

In[50]:= member[0, APPLY[inverse[times[x_]], y_]] :=
          or[not[equal[0, y]], not[member[pair[x, y], DIV]]]
```

---

the condition 1 < y/x

Lemma.

```
In[51]:= SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
           {u → set[0], v → V, w -> APPLY[inverse[times[x]], y]}] // Reverse

Out[51]= or[member[pair[x, y], DIV], member[set[0], APPLY[inverse[times[x]], y]]] == True

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

If **x** divides **y**, then **w = y/x** is a natural number. If **1 < w**, then the product of **w** and a nonzero number **x** yields a number greater than **x**. By introducing and then eliminating **nat** wrappers, one can deduce:

```
In[53]:= (SubstTest[implies, and[equal[w, nat[u]], equal[x, nat[v]],
              equal[y, natmul[w, x]], member[set[0], w], not[equal[0, x]]],
            member[x, y], {u → w, v → x}] /. w → natdiv[y, x]) // Reverse // MapNotNot

Out[53]= or[equal[0, x], member[x, y], not[member[pair[x, y], DIV]],
           not[member[set[0], APPLY[inverse[times[x]], y]]]] == True

In[54]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing a redundant literal:

```
In[55]:= SubstTest[and, implies[p, q], or[p, q],
          {p → equal[0, x], q ->
            implies[and[member[pair[x, y], DIV], member[set[0], natdiv[y, x]]], member[x, y]]}]

Out[55]= or[member[x, y], not[member[pair[x, y], DIV]],
           not[member[set[0], APPLY[inverse[times[x]], y]]]] == True

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

Lemma.

```
In[57]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]], member[y, natmul[x, y]]],
           member[set[0], x], {u → x, v → y}] // Reverse

Out[57]= or[member[set[0], x], not[member[x, omega]],
           not[member[y, omega]], not[member[y, natmul[x, y]]]] == True

In[58]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[59]:= SubstTest[implies, and[member[x, omega], member[z, omega], not[equal[0, x]],
            member[x, y], equal[y, natmul[x, z]]], member[set[0], z], z → natdiv[y, x]] // Reverse

Out[59]= or[equal[0, x], member[set[0], APPLY[inverse[times[x]], y]],
           not[member[x, y]], not[member[pair[x, y], DIV]]] == True

In[60]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing a redundant literal.

```
In[61]:= SubstTest[and, implies[p, q], or[p, q], {p → equal[0, x], q ->
            or[member[set[0], natdiv[y, x]], not[member[x, y]], not[member[pair[x, y], DIV]]]}]

Out[61]= or[member[set[0], APPLY[inverse[times[x]], y]],
           not[member[x, y]], not[member[pair[x, y], DIV]]] == True

In[62]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing yet another redundant literal.

```
In[63]:= SubstTest[and, implies[p, q], or[p, q], {p → member[pair[x, y], DIV],
           q -> or[member[set[0], APPLY[inverse[times[x]], y]], not[member[x, y]]]}]

Out[63]= or[member[set[0], APPLY[inverse[times[x]], y]], not[member[x, y]]] == True

In[64]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[65]:= equiv[member[set[0], APPLY[inverse[times[x]], y]],
            or[member[x, y], not[member[pair[x, y], DIV]]]] // not // not

Out[65]= True
```

```
In[66]:= member[set[0], APPLY[inverse[times[x_]], y_]] :=
          or[member[x, y], not[member[pair[x, y], DIV]]]
```