an inequality for natural multiplication

Johan G. F. Belinfante
2007 March 10

```
In[1]:= SetDirectory["l:"]; << goedel91.07a; << tools.m

        :Package Title: goedel91.07a        2007 March 7 at 8:35 a.m.

        It is now:  2007 Mar 10 at 13:25

        Loading Simplification Rules

        TOOLS.M                       Revised 2007 March 3

        weightlimit = 40
```

summary

A wrapper-free generalization of the following rewrite rule for natural multiplication is derived in this notebook.

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

Out[2]= and[member[nat[y], nat[z]], not[equal[0, nat[x]]]]
```

derivation

The **nat** wrappers on the existing rule can be replaced with numberhood literals:

```
In[3]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]], equal[z, nat[w]]],
           or[equal[0, x], member[y, z], not[member[natmul[x, y], natmul[x, z]]]],
           {u → x, v → y, w → z}] // Reverse // MapNotNot

Out[3]= or[equal[0, x], member[y, z], not[member[x, omega]], not[member[y, omega]],
          not[member[z, omega]], not[member[natmul[x, y], natmul[x, z]]]] == True

In[4]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Three of the six literals here are redundant, and can be removed as follows:

```
In[5]:= Map[not,
          SubstTest[and, implies[and[p1, p2, p3, p4, p5], p6], implies[and[p1, p2, not[p3]], p6],
           implies[p2, p4], implies[p2, p5], not[implies[and[p1, p2], p6]],
           {p1 → member[z, omega], p2 -> member[natmul[x, y], natmul[x, z]], p3 → not[equal[0, x]],
            p4 → member[x, omega], p5 → member[y, omega], p6 → member[y, z]}]] // Reverse

Out[5]= or[member[y, z], not[member[z, omega]], not[member[natmul[x, y], natmul[x, z]]]] == True
```

```
In[6]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[7]:= SubstTest[implies, and[member[t, u], equal[u, v]], member[t, v],
           {t → natmul[x, y], u → V, v → natmul[x, z]}] // Reverse // MapNotNot

Out[7]= or[member[z, omega], member[natmul[x, y], natmul[x, z]],
          not[member[x, omega]], not[member[y, omega]]] == True

In[8]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[9]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]], equal[z, nat[w]]],
           or[equal[0, x], not[member[y, z]], member[natmul[x, y], natmul[x, z]]],
           {u → x, v → y, w → z}] // Reverse // MapNotNot

Out[9]= or[equal[0, x], member[natmul[x, y], natmul[x, z]],
          not[member[x, omega]], not[member[y, z]], not[member[z, omega]]] == True

In[10]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Removing a redundant literal.)

```
In[11]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[z, omega], q -> or[equal[0, x], member[natmul[x, y], natmul[x, z]],
             not[member[x, omega]], not[member[y, omega]], not[member[y, z]]]}]

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

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

Main theorem.

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

Out[13]= True

In[14]:= member[natmul[x_, y_], natmul[x_, z_]] :=
          or[and[member[x, omega], member[y, omega], not[member[z, omega]]],
           and[member[x, omega], member[y, omega], member[y, z], not[equal[0, x]]]]
```