reducing x – y·z modulo z

Johan G. F. Belinfante
2007 March 16

```
In[1]:= SetDirectory["l:"]; << goedel91.15a; << tools.m

        :Package Title: goedel91.15a        2007 March 15 at 5:25 p.m.

        It is now:  2007 Mar 16 at 19:19

        Loading Simplification Rules

        TOOLS.M                      Revised 2007 March 3

        weightlimit = 40
```

summary

A formula for reducing **x – y · z** modulo **z** is derived, together with a corollary that enables the **GOEDEL** program to recognize the truth of the Corollary to Quaife's Theorem **(Q6)** and Quaife's Theorem **(Q8)**.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical
           Theories, Appendix 3. Theorems Proved in Peano's Arithmetic,
           Kluwer Academic Publishers, Dordrecht, 1992. Cf. pp. 195-6.";
```

reducing x – y · z modulo z

Lemma. (Temporary rewrite rule.)

```
In[3]:= equal[union[complement[image[V, intersection[omega, set[y]]]],
          natmod[natsub[x, natmul[y, z]], z]], natmod[natsub[x, natmul[y, z]], z]]

Out[3]= True

In[4]:= union[complement[image[V, intersection[omega, set[y_]]]],
          natmod[natsub[x_, natmul[y_, z_]], z_]] := natmod[natsub[x, natmul[y, z]], z]
```

Theorem.

```
In[5]:= SubstTest[natmod, natadd[t, natmul[y, z]], z, t → natsub[x, natmul[y, z]]]

Out[5]= natmod[natsub[x, natmul[y, z]], z] ==
         union[complement[image[V, intersection[omega, set[y]]]],
          image[V, intersection[complement[x], natmul[y, z]]], natmod[x, z]]
```

```
In[6]:= natmod[natsub[x_, natmul[y_, z_]], z_] :=
          union[complement[image[V, intersection[omega, set[y]]]],
           image[V, intersection[complement[x], natmul[y, z]]], natmod[x, z]]
```

main theorem

Using **nat** wrappers yields a result which can be used to derived the Corollary to Quaife's theorem **(Q6)**.

```
In[7]:= Map[not, SubstTest[member, t, natmod[t, nat[z]],
            t → natsub[nat[x], natmul[nat[y], nat[z]]]] // Reverse]

Out[7]= or[member[nat[x], natmul[nat[y], nat[z]]],
          not[member[nat[x], natadd[natmod[nat[x], nat[z]], natmul[nat[y], nat[z]]]]]] == True

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

Removing the **nat** wrappers yields a wrapper-free counterpart, but with redundant literals.

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

Out[9]= or[member[x, natmul[y, z]], not[member[x, omega]],
          not[member[x, natadd[natmod[x, z], natmul[y, z]]]],
          not[member[y, omega]], not[member[z, omega]]] == True

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

The following lemma suffices to eliminate the redundant literals:

```
In[11]:= SubstTest[implies, and[equal[u, V], equal[v, V]], implies[member[x, u], member[x, v]],
            {u -> natadd[natmod[x, z], natmul[y, z]], v → natmul[y, z]}] // Reverse

Out[11]= or[and[member[y, omega], member[z, omega]], member[x, natmul[y, z]],
            not[member[x, natadd[natmod[x, z], natmul[y, z]]]]] == True

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

Removing the redundant literals yields:

```
In[13]:= SubstTest[and, implies[p, q], or[p, q], {p → and[member[y, omega], member[z, omega]],
             q -> or[not[member[x, omega]], member[x, natmul[y, z]],
               not[member[x, natadd[natmod[x, z], natmul[y, z]]]]]}] // MapNotNot

Out[13]= or[member[x, natmul[y, z]], not[member[x, omega]],
            not[member[x, natadd[natmod[x, z], natmul[y, z]]]]] == True

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

One final lemma is needed.

```
In[15]:= Map[or[member[x, omega], #] &, SubstTest[implies, and[member[x, V], equal[V, v]],
           member[x, v], v -> natadd[natmod[x, z], natmul[y, z]]]] // Reverse // MapNotNot

Out[15]= or[member[x, omega],
          member[x, natadd[natmod[x, z], natmul[y, z]]], not[member[x, V]]] == True

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

Main theorem.

```
In[17]:= equiv[member[x, natadd[natmod[x, z], natmul[y, z]]],
          or[and[member[x, V], not[member[x, omega]]], member[x, natmul[y, z]]]] // not // not

Out[17]= True

In[18]:= member[x_, natadd[natmod[x_, z_], natmul[y_, z_]]] :=
         or[and[member[x, V], not[member[x, omega]]], member[x, natmul[y, z]]]
```