Quaife's theorems (Q11) and (Q12).

Johan G. F. Belinfante
2007 March 21

```
In[1]:= SetDirectory["l:"]; << goedel91.18a; << tools.m

        :Package Title: goedel91.18a        2007 March 18 at 8:00 p.m.

        It is now:  2007 Mar 21 at 17:4

        Loading Simplification Rules

        TOOLS.M                       Revised 2007 March 3

        weightlimit = 40
```

summary

Quaife's theorems **(Q11)** and **(Q12)** are derived in this notebook.

```
In[2]:= "Art Quaife, Automated Development of Fundamental
           Mathematical Theories, Appendix 3. Theorems Proved in Peano's
           Arithmetic, Kluwer Academic Publishers, Dordrecht, 1992. Cf. p. 196";
```

derivation of (Q11)

Theorem **(Q11)**.

```
In[3]:= Map[
          implies[member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], nat[z]], #] &,
          SubstTest[equal, u, natmod[u, v],
           {u → natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], v → nat[z]}]] // Reverse

Out[3]= or[equal[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]],
           natmod[natadd[nat[x], nat[y]], nat[z]]],
          not[member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], nat[z]]]] == True

In[4]:= or[equal[natadd[natmod[nat[x_], nat[z_]], natmod[nat[y_], nat[z_]]],
           natmod[natadd[nat[x_], nat[y_]], nat[z_]]], not[
           member[natadd[natmod[nat[x_], nat[z_]], natmod[nat[y_], nat[z_]]], nat[z_]]]] := True
```

derivation of (Q12)

Lemma.

```
In[5]:= Map[or[member[natadd[nat[w], nat[x]], natadd[nat[y], nat[z]]], #] &,
          SubstTest[implies, and[subclass[nat[s], nat[y]], subclass[nat[t], nat[z]]],
           subclass[natadd[nat[s], nat[t]], natadd[nat[y], nat[z]]],
           {s → succ[nat[w]], t → succ[nat[x]]}]] // Reverse

Out[5]= or[member[natadd[nat[w], nat[x]], natadd[nat[y], nat[z]]],
          not[member[nat[w], nat[y]]], not[member[nat[x], nat[z]]]] == True

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

Theorem.

```
In[7]:= SubstTest[or, member[natadd[nat[u], nat[v]], natadd[nat[z], nat[z]]],
          not[member[nat[u], nat[z]]], not[member[nat[v], nat[z]]],
          {u → natmod[nat[x], nat[z]], v -> natmod[nat[y], nat[z]]}] // Reverse

Out[7]= or[equal[0, nat[z]], member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]],
           natadd[nat[z], nat[z]]]] == True

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

Lemma.

```
In[9]:= SubstTest[implies, and[member[pair[u, v], DIV], member[pair[u, w], DIV]],
          member[pair[u, natadd[v, w]], DIV],
          {u → nat[z], v → natsub[nat[x], natmod[nat[x], nat[z]]],
           w → natsub[nat[y], natmod[nat[y], nat[z]]]}] // Reverse

Out[9]= member[pair[nat[z], natsub[natadd[nat[x], nat[y]],
            natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]]], DIV] == True

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

Lemma.

```
In[11]:= Map[not, SubstTest[implies, member[pair[u, v], DIV], member[v, omega],
            {u -> nat[z], v -> natsub[natadd[nat[x], nat[y]],
               natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]]}]] // Reverse

Out[11]= member[natadd[nat[x], nat[y]],
           natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]] == False

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

Lemma.

```
In[13]:= SubstTest[implies, and[member[pair[u, v], DIV], member[pair[u, w], DIV]],
           member[pair[u, natadd[v, w]], DIV],
           {u → nat[z], v → natsub[natadd[nat[x], nat[y]], natadd[natmod[nat[x], nat[z]],
               natmod[nat[y], nat[z]]]], w → nat[z]}] // MapNotNot // Reverse

Out[13]= member[pair[nat[z], natsub[natadd[nat[x], nat[y], nat[z]],
            natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]], DIV] == True

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

Theorem.

```
In[15]:= SubstTest[implies,
           and[member[pair[u, natsub[v, w]], DIV], member[w, u]], equal[w, natmod[v, u]],
           {u → nat[z], v -> natadd[nat[x], nat[y]], w -> natsub[natadd[natmod[nat[x], nat[z]],
               natmod[nat[y], nat[z]]], nat[z]]}] // Reverse // MapNotNot

Out[15]= or[equal[natadd[nat[z], natmod[natadd[nat[x], nat[y]], nat[z]]],
           natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]],
          member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], nat[z]],
          not[member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]],
            natadd[nat[z], nat[z]]]] == True

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

Simplified version of Quaife's Theorem **(Q12)**.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], implies[not[p1], p4], implies[and[p2, p3], p4],
           not[implies[p3, p4]], {p1 → not[equal[0, nat[z]]], p2 -> member[
              natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], natadd[nat[z], nat[z]]],
            p3 → not[member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], nat[z]]],
            p4 -> equal[natadd[nat[z], natmod[natadd[nat[x], nat[y]], nat[z]]],
              natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]]}]] // Reverse

Out[17]= or[equal[natadd[nat[z], natmod[natadd[nat[x], nat[y]], nat[z]]],
           natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]]],
          member[natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], nat[z]]] == True

In[18]:= or[equal[natadd[nat[z_], natmod[natadd[nat[x_], nat[y_]], nat[z_]]],
           natadd[natmod[nat[x_], nat[z_]], natmod[nat[y_], nat[z_]]]],
          member[natadd[natmod[nat[x_], nat[z_]], natmod[nat[y_], nat[z_]]], nat[z_]]] := True
```

Corollary. (Original statement of **(Q12)**, using floored subtraction.)

In[19]:=
```
Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
   {p1 → not[member[natadd[natmod[nat[x], nat[y]], natmod[nat[z], nat[y]]], nat[y]]],
    p2 -> equal[natadd[nat[y], natmod[natadd[nat[x], nat[z]], nat[y]]],
      natadd[natmod[nat[x], nat[y]], natmod[nat[z], nat[y]]]],
    p3 -> equal[nat[natsub[natadd[natmod[nat[x], nat[y]], natmod[nat[z], nat[y]]],
        nat[y]]], natmod[natadd[nat[x], nat[z]], nat[y]]]}] // Reverse
```

Out[19]=
```
or[equal[nat[natsub[natadd[natmod[nat[x], nat[y]], natmod[nat[z], nat[y]]], nat[y]]],
   natmod[natadd[nat[x], nat[z]], nat[y]]],
  member[natadd[natmod[nat[x], nat[y]], natmod[nat[z], nat[y]]], nat[y]] == True
```

In[20]:=
```
or[equal[
   nat[natsub[natadd[natmod[nat[x_], nat[y_]], natmod[nat[z_], nat[y_]]], nat[y_]]],
   natmod[natadd[nat[x_], nat[z_]], nat[y_]]],
  member[natadd[natmod[nat[x_], nat[y_]], natmod[nat[z_], nat[y_]]], nat[y_]] := True
```