Quaife's (O26) clauses

Johan G. F. Belinfante and Claudia D. Huang
Revised 2005 July 9

```
In[1]:= SetDirectory["i:"]; << goedel71.09a; << tools.m

        :Package Title: goedel71.09a          2005 July 9 at 10:40 a.m.

        It is now:  2005 Jul 9 at 11:33

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 June 17

        weightlimit = 40
```

summary

A cancellation law is derived which implies both clauses of Quaife's theorem **(O26)**.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical Theories,
           Kluwer Academic Publishers, Dordrecht, the Netherlands, 1992.";
```

a positivity result

If natural numbers satisfy $\mathbf{x < y}$, then $\mathbf{0 < y}$. The following corollary follows from this:

```
In[3]:= SubstTest[implies, member[nat[z], nat[w]],
         member[0, nat[w]], w → natsub[nat[y], nat[x]]]

Out[3]= or[member[nat[x], nat[y]],
          not[member[nat[z], nat[natsub[nat[y], nat[x]]]]]] == True

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

removing a complicated rule

A rule derived earlier made transposition automatic, but it was found to yield unnecessarily complicated expressions. This rule is removed here, and will be replaced with a less automatic rule in the next section.

```
In[5]:= member[monus[nat[x], nat[y]], nat[z]]

Out[5]= or[and[member[nat[x], nat[y]], not[equal[0, nat[z]]]],
         and[member[nat[x], natadd[nat[y], nat[z]]], not[member[nat[x], nat[y]]]]]

In[6]:= member[monus[nat[x_], nat[y_]], nat[z_]] =.
```

The removal of this rule has no effect on Quaife's **DF** group, but it does affect the **(O26)** clauses.

---

a replacement transposition rule for monus

Lemma. (Removing a **nat** wrapper.)

```
In[7]:= SubstTest[implies, and[equal[u, v], member[v, w]], member[u, w],
          {u → natsub[nat[x], nat[y]], v -> nat[natsub[nat[x], nat[y]]], w -> nat[z]}]

Out[7]= or[member[nat[x], nat[y]], member[natsub[nat[x], nat[y]], nat[z]],
           not[member[nat[natsub[nat[x], nat[y]]], nat[z]]]] == True

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

The following rewrite rule for transposition involving **monus** is a replacement for the one removed in the preceding section.

```
In[9]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[p3, p4], not[implies[and[p1, p2], p4]],
            {p1 → member[nat[natsub[nat[x], nat[y]]], nat[z]], p2 →
              not[member[nat[x], nat[y]]], p3 → member[natsub[nat[x], nat[y]], nat[z]],
             p4 → member[nat[x], natadd[nat[y], nat[z]]]}]]

Out[9]= or[member[nat[x], nat[y]], member[nat[x], natadd[nat[y], nat[z]]],
           not[member[nat[natsub[nat[x], nat[y]]], nat[z]]]] == True

In[10]:= or[member[nat[x_], nat[y_]], member[nat[x_], natadd[nat[y_], nat[z_]]],
            not[member[nat[natsub[nat[x_], nat[y_]]], nat[z_]]]] := True
```

---

a cancellation law

A temporary rewrite rule

```
In[11]:= SubstTest[natadd, nat[x], natsub[nat[z], nat[w]],
           w → intersection[nat[y], nat[z]]] // Reverse

Out[11]= natsub[natadd[nat[x], nat[z]], intersection[nat[y], nat[z]]] ==
          natadd[nat[x], nat[natsub[nat[z], nat[y]]]]

In[12]:= natsub[natadd[nat[x_], nat[z_]], intersection[nat[y_], nat[z_]]] :=
          natadd[nat[x], nat[natsub[nat[z], nat[y]]]]
```

Transposition lemma.

```
In[13]:= SubstTest[implies,
          and[not[member[nat[u], nat[v]]], member[nat[z], natsub[nat[u], nat[v]]]],
          member[natadd[nat[z], nat[v]], nat[u]],
          {u → natadd[nat[x], nat[z]], v → intersection[nat[y], nat[z]]}]

Out[13]= or[member[nat[y], nat[x]], member[nat[z], nat[x]],
           not[member[nat[z], natadd[nat[x], nat[natsub[nat[z], nat[y]]]]]]] == True

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

Transposition lemma.

```
In[15]:= SubstTest[implies, member[natadd[nat[z], nat[v]], nat[u]],
          member[nat[z], natsub[nat[u], nat[v]]],
          {u → natadd[nat[x], nat[z]], v → intersection[nat[y], nat[z]]}]

Out[15]= or[and[not[member[nat[y], nat[x]]], not[member[nat[z], nat[x]]]],
           member[nat[z], natadd[nat[x], nat[natsub[nat[z], nat[y]]]]]] == True

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

The following cancellation law admittedly does not cancel any variables, but it does remove arithmetic operations.

```
In[17]:= equiv[member[nat[z], natadd[nat[x], nat[natsub[nat[z], nat[y]]]]],
          or[member[nat[y], nat[x]], member[nat[z], nat[x]]]]

Out[17]= True

In[18]:= member[nat[z_], natadd[nat[x_], nat[natsub[nat[z_], nat[y_]]]]] :=
          or[member[nat[y], nat[x]], member[nat[z], nat[x]]]
```

a cancellation law related to Quaife's (O26)

```
In[19]:= SubstTest[implies,
          and[not[member[nat[x], nat[y]]], member[monus[nat[x], nat[y]], nat[w]]],
          member[nat[x], natadd[nat[y], nat[w]]], w → natsub[nat[x], nat[z]]]

Out[19]= or[member[nat[x], nat[y]], member[nat[z], nat[y]], not[
            member[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[x], nat[z]]]]] == True

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

The first literal in the above is redundant.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           implies[p1, or[p4, p5]], implies[and[p3, p4], p5], not[implies[p1, p5]],
           {p1 -> member[monus[nat[x], nat[y]], monus[nat[x], nat[z]]],
            p2 -> member[nat[z], nat[x]], p3 → not[member[nat[x], nat[z]]],
            p4 -> member[nat[x], nat[y]], p5 -> member[nat[z], nat[y]]}]]

Out[21]= or[member[nat[z], nat[y]], not[
            member[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[x], nat[z]]]]] == True

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

To obtain an implication in the reverse direction, the following temporary lemma will be used:

```
In[23]:= SubstTest[member, union[nat[x], nat[y]], nat[w], w -> natadd[nat[x], nat[z]]]

Out[23]= member[union[nat[x], nat[y]], natadd[nat[x], nat[z]]] ==
          and[member[nat[y], natadd[nat[x], nat[z]]], not[equal[0, nat[z]]]]

In[24]:= member[union[nat[x_], nat[y_]], natadd[nat[x_], nat[z_]]] :=
          and[member[nat[y], natadd[nat[x], nat[z]]], not[equal[0, nat[z]]]]
```

The result derived in the preceding section now implies:

```
In[25]:= Map[implies[and[member[nat[z], nat[x]], member[nat[z], nat[y]]], #] &,
           SubstTest[member, natadd[nat[w], nat[u]],
            natadd[nat[w], nat[v]], {u → natsub[nat[x], nat[y]],
             v → natsub[nat[x], nat[z]], w → nat[y]}]] // Reverse

Out[25]= or[member[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[x], nat[z]]]],
           not[member[nat[z], nat[x]]], not[member[nat[z], nat[y]]]] == True

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

Combining various results obtained above yields a logical equivalence, which is made into a rewrite rule that subsumes both of Quaife's **(O26)** clauses.

```
In[27]:= equiv[member[monus[nat[x], nat[y]], monus[nat[x], nat[z]]],
            and[member[nat[z], nat[x]], member[nat[z], nat[y]]]] // not // not

Out[27]= True

In[28]:= member[nat[natsub[nat[x_], nat[y_]]], nat[natsub[nat[x_], nat[z_]]]] :=
          and[member[nat[z], nat[x]], member[nat[z], nat[y]]]
```