Quaife's Theorem (O25)

Johan G. F. Belinfante and Claudia D. Huang
2005 July 10

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

        :Package Title: goedel71.09b          2005 July 9 at 11:50 a.m.

        It is now:  2005 Jul 10 at 14:30

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 June 17

        weightlimit = 40
```

---

summary

This notebook contains the derivation of rewrite rules that imply the two clauses of Quaife's Theorem **(O25)** and its corollaries.

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

Quaife's Theorem (O25)

A cancellation law for **natadd** yields the following rewrite rule:

```
In[3]:= SubstTest[member, natadd[nat[z], nat[u]], natadd[nat[z], nat[v]],
          {u → natsub[nat[x], nat[z]], v → natsub[nat[y], nat[z]]} // Reverse

Out[3]= member[nat[natsub[nat[x], nat[z]]], nat[natsub[nat[y], nat[z]]]] ==
         and[member[nat[x], nat[y]], member[nat[z], nat[y]]]

In[4]:= member[nat[natsub[nat[x_], nat[z_]]], nat[natsub[nat[y_], nat[z_]]]] :=
         and[member[nat[x], nat[y]], member[nat[z], nat[y]]]
```

This rewrite rule implies both clauses of Quaife's Theorem **(O25)**:

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

Out[5]= True
```

```
In[6]:= implies[member[nat[x], nat[y]],
          or[member[monus[nat[x], nat[z]], monus[nat[y], nat[z]]],
           member[nat[x], nat[z]]]]

Out[6]= True
```

U[nat[x]]

The predecessor of a nonzero natural number is **U[nat[x]]**. A formula connecting this with **monus** is easily derived.

```
In[7]:= SubstTest[equal, succ[nat[u]], succ[nat[v]],
          {u → U[nat[x]], v -> monus[nat[x], set[0]]}]

Out[7]= True == equal[nat[natsub[nat[x], set[0]]], U[nat[x]]]

In[8]:= nat[natsub[nat[x_], set[0]]] := U[nat[x]]
```

---

corollaries of (O25)

The corollaries for Quaife's Theorem **(O25)** are special cases of the following rewrite rule.

```
In[9]:= equiv[member[succ[x], nat[y]], member[x, U[nat[y]]]]

Out[9]= True

In[10]:= member[succ[x_], nat[y_]] := member[x, U[nat[y]]]
```

The following two corollaries were given by Quaife:

```
In[11]:= implies[member[succ[set[0]], nat[x]], member[set[0], monus[nat[x], set[0]]]]

Out[11]= True

In[12]:= implies[member[succ[succ[set[0]]], nat[x]],
           member[succ[set[0]], monus[nat[x], set[0]]]]

Out[12]= True
```

a transposition law for monus

Instead of deriving **(O25)** from the cancellation law for **natadd**, as was done above, another approach would be to use transposition. The following transposition lemma for **monus** holds:

```
In[13]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → member[nat[z], nat[natsub[nat[x], nat[y]]]], p2 →
             not[member[nat[x], nat[y]]], p3 → member[nat[z], natsub[nat[x], nat[y]]],
            p4 → member[natadd[nat[y], nat[z]], nat[x]]}]]

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

In[14]:= or[member[nat[x_], nat[y_]], member[natadd[nat[y_], nat[z_]], nat[x_]],
           not[member[nat[z_], nat[natsub[nat[x_], nat[y_]]]]]] := True
```

Note that this is almost identical to a corresponding rule for **natsub**:

```
In[15]:= or[member[nat[x], nat[y]], member[natadd[nat[y], nat[z]], nat[x]],
          not[member[nat[z], natsub[nat[x], nat[y]]]]]

Out[15]= True
```

serendipity

This section contains a result discovered in the course of deriving the corollary to **(O25)**. A lemma is needed:

```
In[16]:= SubstTest[subclass, nat[x], nat[y], y → set[0]]

Out[16]= subclass[nat[x], set[0]] == not[member[set[0], nat[x]]]

In[17]:= subclass[nat[x_], set[0]] := not[member[set[0], nat[x]]]
```

Theorem.

```
In[18]:= SubstTest[implies, member[y, z], not[empty[z]], z → U[nat[x]]]

Out[18]= or[member[set[0], nat[x]], not[member[y, U[nat[x]]]]] == True

In[19]:= or[member[set[0], nat[x_]], not[member[y_, U[nat[x_]]]]] := True
```