Quaife's Theorems (MM10), (MM11) and (MM12)

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

```
In[1]:= SetDirectory["i:"]; << goedel71.04a; << tools.m

        :Package Title: goedel71.04a          2005 July 4 at 5:45 p.m.

        It is now:  2005 Jul 9 at 10:15

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 June 17

        weightlimit = 40
```

---

summary

Rewrite rules are derived to enable the **GOEDEL** program to recognize the validity of the clauses in Theorems **(MM11)** and **(MM12)** of Quaife's **MM** group. In addition, some corollaries of his Schematic Theorem **(MM10)** are derived.

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

---

corollaries of (MM10)

Quaife's schematic theorems **(MM10)** can be replaced by ordinary theorems in the **GOEDEL** program because any statement about a natural number can be rewritten as the assertion that the number in question belongs to some suitably chosen class **z**. These clauses are already in the **GOEDEL** program. Replacing the class **z** with its complement yields clauses resembling those of **(MM10)**, but with each literal replaced with its negation.

```
In[3]:= SubstTest[implies, and[member[nat[x], w], member[nat[y], w]],
         member[intersection[nat[x], nat[y]], w], w → complement[z]]

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

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

In[5]:= SubstTest[implies, and[member[nat[x], w], member[nat[y], w]],
         member[union[nat[x], nat[y]], w], w → complement[z]]

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

In[6]:= or[member[nat[x_], z_], member[nat[y_], z_],
          not[member[union[nat[x_], nat[y_]], z_]]] := True
```

Two special corollaries can be derived for the special case when **z** is a natural number. The first of these is:

```
In[7]:= Map[not, SubstTest[subclass, nat[z],
           nat[w], w → intersection[nat[x], nat[y]]]] // Reverse

Out[7]= member[intersection[nat[x], nat[y]], nat[z]] ==
         or[member[nat[x], nat[z]], member[nat[y], nat[z]]]

In[8]:= member[intersection[nat[x_], nat[y_]], nat[z_]] :=
         or[member[nat[x], nat[z]], member[nat[y], nat[z]]]
```

The following logical equivalence justifies a similar rewrite rule with **union** in place of **intersection**.

```
In[9]:= equiv[member[union[nat[x], nat[y]], nat[z]],
           and[member[nat[x], nat[z]], member[nat[y], nat[z]]]] // not // not

Out[9]= True

In[10]:= member[union[nat[x_], nat[y_]], nat[z_]] :=
          and[member[nat[x], nat[z]], member[nat[y], nat[z]]]
```

---

the four clauses of (MM11)

For Quaife's first clause, one can get a simple rule without **nat** wrappers.

```
In[11]:= equal[intersection[x, y], intersection[x, succ[y]]] // AssertTest

Out[11]= equal[intersection[x, y], intersection[x, succ[y]]] ==
          or[member[y, y], not[member[y, x]]]

In[12]:= equal[intersection[x_, y_], intersection[x_, succ[y_]]] :=
          or[member[y, y], not[member[y, x]]]
```

For the second clause, one only gets a simple result when **nat** wrappers are retained.

```
In[13]:= (equal[succ[union[nat[x], nat[z]]], union[succ[nat[x]], nat[z]]] //
           AssertTest) /. z → nat[y]

Out[13]= equal[succ[union[nat[x], nat[y]]], union[nat[y], succ[nat[x]]]] ==
          not[member[nat[x], nat[y]]]

In[14]:= equal[succ[union[nat[x_], nat[y_]]], union[nat[y_], succ[nat[x_]]]] :=
          not[member[nat[x], nat[y]]]
```

Likewise, for Quaife's third clause, one only finds a simple result when **nat** wraps the variables. The following implies Quaife's third clause.

```
In[15]:= (equal[intersection[nat[z], succ[nat[y]]],
             succ[intersection[nat[y], nat[z]]]] // AssertTest) /. z → nat[x]

Out[15]= equal[intersection[nat[x], succ[nat[y]]],
           succ[intersection[nat[x], nat[y]]]] == member[nat[y], nat[x]]

In[16]:= equal[intersection[nat[x_], succ[nat[y_]]],
           succ[intersection[nat[x_], nat[y_]]]] := member[nat[y], nat[x]]
```

For Quaife's fourth clause, one does not need **nat** wrappers.

```
In[17]:= equal[union[x, y], union[x, succ[y]]] // AssertTest

Out[17]= equal[union[x, y], union[x, succ[y]]] ==
          or[member[y, x], member[y, y], not[member[y, V]]]

In[18]:= equal[union[x_, y_], union[x_, succ[y_]]] :=
          or[member[y, x], member[y, y], not[member[y, V]]]
```

---

a cancellation lemma

The following cancellation law for **monus** resembles a corresponding one for **natsub**.

```
In[19]:= SubstTest[natadd, natsub[u, v], natsub[v, w],
          {u → union[nat[x], nat[y]], v → nat[y], w → intersection[nat[y], nat[z]]}]

Out[19]= natadd[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[y], nat[z]]]] ==
          natsub[union[nat[x], nat[y]], intersection[nat[y], nat[z]]]

In[20]:= natadd[nat[natsub[nat[x_], nat[y_]]], nat[natsub[nat[y_], nat[z_]]]] :=
          natsub[union[nat[x], nat[y]], intersection[nat[y], nat[z]]]
```

---

the two clauses of (MM12)

Lemma.

```
In[21]:= SubstTest[implies, equal[0, u], equal[natadd[u, nat[v]], union[u, nat[v]]],
          {u -> nat[natsub[nat[x], nat[y]]], v -> nat[natsub[nat[y], nat[x]]]}]

Out[21]= or[equal[natsub[union[nat[x], nat[y]], intersection[nat[x], nat[y]]],
            union[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[y], nat[x]]]]],
           member[nat[y], nat[x]]] == True

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

In[23]:= SubstTest[and, implies[p1, p3], implies[p2, p3], or[p1, p2],
            {p1 → not[member[nat[x], nat[y]]], p2 → not[member[nat[y], nat[x]]],
             p3 -> equal[natsub[union[nat[x], nat[y]], intersection[nat[x], nat[y]]],
               union[nat[natsub[nat[x], nat[y]]],
                nat[natsub[nat[y], nat[x]]]]]}] // MapNotNot // Reverse

Out[23]= equal[natsub[union[nat[x], nat[y]], intersection[nat[x], nat[y]]],
           union[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[y], nat[x]]]]] == True
```

The following rewrite rule implies the first clause of Quaife's **(MM12)**.

```
In[24]:= union[nat[natsub[nat[x_], nat[y_]]], nat[natsub[nat[y_], nat[x_]]]] :=
          natsub[union[nat[x], nat[y]], intersection[nat[x], nat[y]]]
```

Lemma.

```
In[25]:= SubstTest[implies, equal[0, u], equal[intersection[u, v], 0],
          {u -> nat[natsub[nat[x], nat[y]]], v -> nat[natsub[nat[y], nat[x]]]}]

Out[25]= or[equal[0, intersection[nat[natsub[nat[x], nat[y]]],
             nat[natsub[nat[y], nat[x]]]], member[nat[y], nat[x]]] == True

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

Second clause of **(MM12)**.

```
In[27]:= SubstTest[and, implies[p1, p3], implies[p2, p3], or[p1, p2],
           {p1 → not[member[nat[x], nat[y]]], p2 → not[member[nat[y], nat[x]]],
            p3 -> equal[0, intersection[nat[natsub[nat[x], nat[y]]],
               nat[natsub[nat[y], nat[x]]]]]}] // MapNotNot

Out[27]= True == equal[0,
           intersection[nat[natsub[nat[x], nat[y]]], nat[natsub[nat[y], nat[x]]]]]
```

```
In[28]:= intersection[nat[natsub[nat[x_], nat[y_]]],
           nat[natsub[nat[y_], nat[x_]]]] := 0
```

---

comment

The following was derived previously in **SUCC-NAT.NB**.

```
In[29]:= or[member[x, nat[y]], not[member[succ[x], nat[y]]]] // AssertTest

Out[29]= or[member[x, nat[y]], not[member[succ[x], nat[y]]]] == True

In[30]:= or[member[x_, nat[y_]], not[member[succ[x_], nat[y_]]]] := True
```