derivations of some basic theorems

Johan G. F. Belinfante
2004 February 8

```
In[1]:= << goedel54.07c; << tools.m

        :Package Title: goedel54.07c      2004 February 7 at 8:20 p.m.

        It is now:  2004 Feb 9 at 12:26

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 January 3

        weightlimit = 40
```

summary

In this notebook some gaps are filled, providing derivations of various basic theorems that were proved long ago using **Otter**, but which are still not recognized by the **GOEDEL** program.

Quaife's theorem (UP2)

The second clause in Quaife's Theorem **(UP2)** is the following:

```
In[2]:= implies[not[member[y, V]], equal[pairset[x, y], singleton[x]]]

Out[2]= or[equal[pairset[x, y], singleton[x]], member[y, V]]
```

This theorem can be proved using **AssertTest**.

```
In[3]:= or[equal[pairset[x, y], singleton[x]], member[y, V]] // AssertTest

Out[3]= or[equal[pairset[x, y], singleton[x]], member[y, V]] == True
```

In this case, a more useful result is the following:

```
In[4]:= equal[pairset[x, y], singleton[y]] // AssertTest

Out[4]= equal[pairset[x, y], singleton[y]] == or[equal[x, y], not[member[x, V]]]

In[5]:= equal[pairset[x_, y_], singleton[y_]] := or[equal[x, y], not[member[x, V]]]
```

Theorem UP−SS−2

A proof of length 10 on level 8 for the following Theorem **UP−SS−2** was found 1994 September 22 by **Otter**.

```
In[6]:= implies[equal[singleton[x], singleton[y]], equal[pairset[x, z], pairset[y, z]]]

Out[6]= or[equal[pairset[x, z], pairset[y, z]], not[equal[singleton[x], singleton[y]]]]
```

This theorem can be established using **AssertTest**, but that takes 24 seconds. Turning off the **cond** flag has little effect, but turning off the **simplify** flag does reduce the execution time to 2 seconds. A much faster technique, which does not require turning off either flag, is the following:

```
In[7]:= SubstTest[implies, equal[u, v], equal[union[u, w], union[v, w]],
          {u -> singleton[x], v -> singleton[y], w -> singleton[z]}]

Out[7]= or[equal[pairset[x, z], pairset[y, z]], not[equal[singleton[x], singleton[y]]]] == True

In[8]:= or[equal[pairset[x_, z_], pairset[y_, z_]],
           not[equal[singleton[x_], singleton[y_]]]] := True
```

Corollary SS−12−C

The following easy Corollary **SS−12−C** of Quaife's Theorem **(SS12)** was proved 1998 May 10 using **Otter**. It is readily established using **AssertTest**.

```
In[9]:= implies[subclass[x, singleton[y]], or[equal[0, x], member[y, x]]] // AssertTest

Out[9]= or[equal[0, x], member[y, x], not[subclass[x, singleton[y]]]] == True

In[10]:= or[equal[0, x_], member[y_, x_], not[subclass[x_, singleton[y_]]]] := True
```

Corollary SS−4A

The following Corollary **SS−4A** of Quaife's theorem **(SS4)** was proved 2000 April 1 using **Otter**.

```
In[11]:= implies[subclass[y, union[z, singleton[x]]], or[member[x, y], subclass[y, z]]]

Out[11]= or[member[x, y], not[subclass[y, union[z, singleton[x]]]], subclass[y, z]]
```

Specializing Corollary **SS−12−C** to the case of **dif[y,z]** almost yields it:

```
In[12]:= SubstTest[implies, subclass[w, singleton[x]],
           or[equal[0, w], member[x, w]], w -> dif[y, z]]

Out[12]= or[and[member[x, y], not[member[x, z]]],
            not[subclass[y, union[z, singleton[x]]]], subclass[y, z]] == True
```

The only extra step needed here is to extract a part out of the compound conclusion **and[member[x,y], not[member[x,z]]]**.

```
In[13]:= Map[or[member[x, y], #] &, SubstTest[implies,
            subclass[w, singleton[x]], or[equal[0, w], member[x, w]], w -> dif[y, z]]]

Out[13]= or[member[x, y], not[subclass[y, union[z, singleton[x]]]], subclass[y, z]] == True

In[14]:= or[member[x_, y_],
            not[subclass[y_, union[z_, singleton[x_]]]], subclass[y_, z_]] := True
```

Quaife's Theorem (C5)

Quaife's theorem **(C5)** is recognized to be true by the **GOEDEL** program when one writes it like this:

```
In[15]:= implies[and[disjoint[x, y], equal[union[x, y], V]], equal[x, complement[y]]]

Out[15]= True
```

The clause form of this theorem is not currently recognized to be true, but it can be derived by an application of double negation:

```
In[16]:= or[not[equal[union[x, y], V]],
            not[equal[intersection[x, y], 0]], equal[x, complement[y]]] // NotNotTest

Out[16]= or[equal[x, complement[y]],
            not[equal[0, intersection[x, y]]], not[equal[V, union[x, y]]]] == True

In[17]:= or[equal[x_, complement[y_]],
            not[equal[0, intersection[x_, y_]]], not[equal[V, union[x_, y_]]]] := True
```

a special theorem added to Quaife's D group

The following special theorem was proved 1996 December 28 using **Otter**. Although this theorem was not among those listed by Quaife in his **D** group of theorems, it is closely related to the distributive law and it therefore seems logical to add it at the end of this group.

```
In[18]:= implies[and[disjoint[x, y], subclass[x, union[y, z]]], subclass[x, z]]

Out[18]= True
```

Although the **GOEDEL** program recognizes this theorem to be true when it is written in this fashion, its truth is not recognized when the same theorem is presented in clause form. The remedy here again is to apply double negation.

```
In[19]:= or[not[equal[0, intersection[x, y]]],
            not[subclass[x, union[y, z]]], subclass[x, z]] // NotNotTest

Out[19]= or[not[equal[0, intersection[x, y]]],
            not[subclass[x, union[y, z]]], subclass[x, z]] == True

In[20]:= or[not[equal[0, intersection[x_, y_]]],
            not[subclass[x_, union[y_, z_]]], subclass[x_, z_]] := True
```

Theorem DJ−MEM

Theorem **DJ−MEM** proved 1999 July 22 using **Otter** can be written in clause form as follows:

```
In[21]:= implies[member[z, intersection[x, y]], not[disjoint[x, y]]]

Out[21]= or[not[equal[0, intersection[x, y]]], not[member[z, x]], not[member[z, y]]]
```

This theorem does not yield to double negation, nor to **AssertTest**, but it can be derived using **SubstTest**:

```
In[22]:= SubstTest[implies, member[z, w], not[equal[0, w]], w -> intersection[x, y]]

Out[22]= or[not[equal[0, intersection[x, y]]], not[member[z, x]], not[member[z, y]]] == True

In[23]:= or[not[equal[0, intersection[x_, y_]]],
           not[member[z_, x_]], not[member[z_, y_]]] := True
```