APPLY[GLB[DIV], union[x,set[0]]]

Johan G. F. Belinfante
2007 April 24

```
In[1]:= SetDirectory["l:"]; << goedel92.23a; << tools.m

        :Package Title: goedel92.23a        2007 April 23 at 4:20 a.m.

        It is now:  2007 Apr 24 at 7:10

        Loading Simplification Rules

        TOOLS.M                      Revised 2007 March 25

        weightlimit = 40
```

summary

Adjoining the number **0** to a set does not change its greatest common divisor.

derivation

Lemma. (The case that **x** is not a set of numbers.)

```
In[2]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
          {u -> APPLY[GLB[DIV], x], v → V, w -> APPLY[GLB[DIV], union[x, set[0]]]}] // Reverse

Out[2]= or[equal[APPLY[GLB[DIV], x], APPLY[GLB[DIV], union[x, set[0]]]], subclass[x, omega]] ==
         True

In[3]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Divisibility in one direction.)

```
In[4]:= SubstTest[implies, and[subclass[x, t], subclass[t, omega]], member[
           pair[APPLY[GLB[DIV], t], APPLY[GLB[DIV], x]], DIV], t → union[x, set[0]]] // Reverse

Out[4]= or[member[pair[APPLY[GLB[DIV], union[x, set[0]]], APPLY[GLB[DIV], x]], DIV],
          not[subclass[x, omega]]] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= SubstTest[implies, and[subclass[y, t], subclass[t, omega]],
          member[pair[APPLY[GLB[DIV], t], APPLY[GLB[DIV], y]], DIV],
          t → image[DIV, set[nat[x]]]] // Reverse

Out[6]= or[member[pair[nat[x], APPLY[GLB[DIV], y]], DIV],
          not[subclass[y, image[DIV, set[nat[x]]]]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the **nat** wrapper yields:

```
In[8]:= SubstTest[implies, equal[x, nat[t]], or[member[pair[x, APPLY[GLB[DIV], y]], DIV],
           not[subclass[y, image[DIV, set[x]]]]], t → x] // Reverse

Out[8]= or[member[pair[x, APPLY[GLB[DIV], y]], DIV],
          not[member[x, omega]], not[subclass[y, image[DIV, set[x]]]]] == True

In[9]:= or[member[pair[x_, APPLY[GLB[DIV], y_]], DIV],
          not[member[x_, omega]], not[subclass[y_, image[DIV, set[x_]]]]] := True
```

Application:

```
In[10]:= SubstTest[implies, and[member[u, omega], subclass[v, image[DIV, set[u]]]],
           member[pair[u, APPLY[GLB[DIV], v]], DIV],
           {u -> APPLY[GLB[DIV], x], v → union[x, set[0]]}] // Reverse

Out[10]= or[member[pair[APPLY[GLB[DIV], x], APPLY[GLB[DIV], union[x, set[0]]]], DIV],
           not[subclass[x, omega]]] == True

In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (The case that **x** is a set of numbers.)

```
In[12]:= SubstTest[and, implies[p1, p2], implies[p1, p3], {p1 → subclass[x, omega],
           p2 -> member[pair[APPLY[GLB[DIV], x], APPLY[GLB[DIV], union[x, set[0]]]], DIV],
           p3 -> member[pair[APPLY[GLB[DIV], union[x, set[0]]], APPLY[GLB[DIV], x]], DIV]}]

Out[12]= or[equal[APPLY[GLB[DIV], x], APPLY[GLB[DIV], union[x, set[0]]]],
           not[subclass[x, omega]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining the case that **x** is a set of numbers with the case that **x** is not a set of numbers yields an unconditional equation that can be made into a rewrite rule.

```
In[14]:= SubstTest[and, implies[p, q], or[p, q], {p -> subclass[x, omega],
           q -> equal[APPLY[GLB[DIV], x], APPLY[GLB[DIV], union[x, set[0]]]]}]

Out[14]= equal[APPLY[GLB[DIV], x], APPLY[GLB[DIV], union[x, set[0]]]] == True

In[15]:= APPLY[GLB[DIV], union[x_, set[0]]] := APPLY[GLB[DIV], x]
```

A variable-free formulation can be obtained by using **reify**.

```
In[16]:= Map[VERTSECT, SubstTest[reify, x, APPLY[t, union[x, set[0]]], t -> GLB[DIV]]]
Out[16]= composite[GLB[DIV], ADJOIN[set[0]]] == GLB[DIV]

In[17]:= composite[GLB[DIV], ADJOIN[set[0]]] := GLB[DIV]
```

Corollary. (The case of a singleton.)

```
In[18]:= SubstTest[APPLY, GLB[DIV], union[t, set[0]], t → set[x]] // Reverse
Out[18]= APPLY[GLB[DIV], set[0, x]] == APPLY[GLB[DIV], set[x]]
```

One can have more than one listed element. For the case of a pairset, one has:

```
In[19]:= SubstTest[APPLY, GLB[DIV], union[t, set[0]], t → set[x, y]] // Reverse
Out[19]= APPLY[GLB[DIV], set[0, x, y]] == APPLY[GLB[DIV], set[x, y]]
```

The general case is covered by the following rewrite rule that uses *Mathematica*'s **BlankSequence**.

```
In[20]:= APPLY[GLB[DIV], set[0, x__]] := APPLY[GLB[DIV], set[x]]
```

---

a special case

Lemma.

```
In[21]:= SubstTest[implies, equal[x, nat[t]], equal[x, APPLY[GLB[DIV], set[x]]], t → x] // Reverse
Out[21]= or[equal[x, APPLY[GLB[DIV], set[x]]], not[member[x, omega]]] == True

In[22]:= or[equal[x_, APPLY[GLB[DIV], set[x_]]], not[member[x_, omega]]] := True
```

This can also be made into a conditional rewrite rule:

```
In[23]:= implies[member[x, omega], equal[APPLY[GLB[DIV], set[x]], x]]
Out[23]= True

In[24]:= APPLY[GLB[DIV], set[x_]] := x /; member[x, omega]

In[28]:= SubstTest[APPLY, GLB[DIV], union[x, set[0]], x → set[set[0]]] // Reverse
Out[28]= APPLY[GLB[DIV], succ[set[0]]] == set[0]

In[29]:= APPLY[GLB[DIV], succ[set[0]]] := set[0]
```