associative law for gcd

Johan G. F. Belinfante
2009 March 31

```
In[1]:= SetDirectory["l:"]; << goedel.09mar29a; << tools.m
```

```
:Package Title: goedel.09mar29a            2009 March 29 at 4:35 p.m.

It is now:  2009 Mar 31 at 7:33

Loading Simplification Rules

TOOLS.M                                    Revised 2009 February 18

weightlimit = 40
```

summary

Art Quaife's Theorems **(GCD18)**, **(GCD19)**, and **(GCD20)** and two of three corollaries of the latter are derived in this notebook.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical
         Theories, Appendix 3. Theorems Proved in Peano's Arithmetic,
         Kluwer Academic Publishers, Dordrecht, 1992. Cf. pp. 202-206";
```

The associative law for gcd's is derived in this notebook from a more general law that says that the gcd of a union of sets **x** and **y** of natural numbers can be computed in two stages. First one computes the gcd for each of the individual sets. Then one computes the gcd of **union[x,y]** as the gcd of the pair of these two gcd's.

temporary definitions

To facilitate input and to increase readability, two temporary definitions are convenient:

```
In[3]:= gcd[x_] := APPLY[GLB[DIV], x]
```

```
General::spell1 : Possible spelling error: new symbol name "gcd" is similar to existing symbol "GCD". More…
```

```
In[4]:= divides[x_, y_] := member[pair[x, y], DIV]
```

Quaife's Theorem (GCD18)

Theorem. Quaife's Theorem **(GCD18)**. The product of relatively prime divisors is a divisor.

```
In[5]:= SubstTest[implies, equal[z, natmul[u, x]],
          or[not[equal[APPLY[GLB[DIV], set[x, y]], set[0]]],
           not[member[pair[x, z], DIV]], not[member[pair[y, z], DIV]],
           member[pair[natmul[x, y], z], DIV]], u → natdiv[z, x]] // Reverse

Out[5]= or[member[pair[natmul[x, y], z], DIV], not[equal[APPLY[GLB[DIV], set[x, y]], set[0]]],
          not[member[pair[x, z], DIV]], not[member[pair[y, z], DIV]]] == True

In[6]:= or[member[pair[natmul[x_, y_], z_], DIV],
          not[equal[APPLY[GLB[DIV], set[x_, y_]], set[0]]],
          not[member[pair[x_, z_], DIV]], not[member[pair[y_, z_], DIV]]] := True
```

---

common divisors

A number **t** is a common divisor of a set **z** if divides every element of **z**. This is equivalent to the statement that **z** is a subclass of the set **image[DIV, set[t]]** of all multiples of **t**. If a number **t** is a common divisor of **z**, then **t** divides **gcd[z]**.

```
In[7]:= implies[and[member[t, omega], subclass[z, image[DIV, set[t]]]], divides[t, gcd[z]]]

Out[7]= True
```

Theorem. The gcd of a union is a common divisor of the individual gcd's.

```
In[8]:= SubstTest[implies, and[subclass[x, u], subclass[u, omega]],
          member[pair[gcd[u], gcd[x]], DIV], u → union[x, y]] // Reverse

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

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Converse.

```
In[10]:= SubstTest[implies, divides[u, v],
           member[u, omega], {u → gcd[union[x, y]], v -> gcd[x]}] // Reverse

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

In[11]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The above theorem and its converse can be combined into a single rewrite rule:

```
In[12]:= equiv[member[pair[APPLY[GLB[DIV], union[x, y]], APPLY[GLB[DIV], x]], DIV],
          and[subclass[x, omega], subclass[y, omega]]]

Out[12]= True

In[13]:= member[pair[APPLY[GLB[DIV], union[x_, y_]], APPLY[GLB[DIV], x_]], DIV] :=
          and[subclass[x, omega], subclass[y, omega]]
```

two stage computation of the gcd of a union

To show that **gcd[union[x,y]]** is equal to **gcd[set[gcd[x], gcd[y]]]** when **x** and **y** are sets of natural numbers, it suffices to show that each divides the other.

Lemma.

```
In[14]:= (SubstTest[implies, and[member[u, v], subclass[v, omega]], divides[gcd[v], u],
            {u → gcd[t], v -> set[gcd[t], gcd[y]]}] // Reverse) /. t → intersection[omega, x]

Out[14]= member[pair[
           APPLY[GLB[DIV], set[APPLY[GLB[DIV], y], APPLY[GLB[DIV], intersection[omega, x]]]],
           APPLY[GLB[DIV], intersection[omega, x]]], DIV] == True

In[15]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

One can regard **intersection[omega, x]** as a wrapper for a set of natural numbers. This wrapper can be eliminated as follows.

Theorem. If **x** is a set of natural numbers, then **gcd[set[gcd[x], gcd[y]]]** divides **gcd[x]**.

```
In[16]:= SubstTest[implies, equal[x, intersection[t, omega]],
           member[pair[APPLY[GLB[DIV], set[APPLY[GLB[DIV], y], APPLY[GLB[DIV], x]]],
             APPLY[GLB[DIV], x]], DIV], t → x] // Reverse

Out[16]= or[member[pair[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
             APPLY[GLB[DIV], x]], DIV], not[subclass[x, omega]]] == True

In[17]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary. If both **x** and **y** are sets of natural numbers, then **gcd[union[x, y]]** divides **gcd[set[gcd[x], gcd[y]]]**.

```
In[18]:= SubstTest[implies, and[divides[t, gcd[x]], divides[t, gcd[y]]],
           divides[t, gcd[set[gcd[x], gcd[y]]]], t -> gcd[union[x, y]]] // Reverse

Out[18]= or[member[pair[APPLY[GLB[DIV], union[x, y]],
             APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]]], DIV],
           not[subclass[x, omega]], not[subclass[y, omega]]] == True

In[19]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

the opposite direction

It remains to show that if **x** and **y** are sets of natural numbers, then also **gcd[set[gcd[x], gcd[y]]]** divides **gcd[union[x, y]]**. The idea is to use the fact that a number **t** is a common divisor of a set **union[x, y]** if it is a common divisor for both **x** and for **y**.

Lemma.

```
In[20]:= (SubstTest[implies, and[subclass[x, u], subclass[u, v]], subclass[x, v],
             {u → image[DIV, set[w]], v → image[DIV, set[t]]}] /.
           {t → gcd[set[gcd[x], gcd[y]]], w → gcd[x]}) // Reverse

Out[20]= or[not[member[pair[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
            APPLY[GLB[DIV], x]], DIV]], not[subclass[x, omega]], subclass[x, image[DIV,
           set[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]]]]]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[22]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> subclass[x, omega], p2 → divides[gcd[set[gcd[x], gcd[y]]], gcd[x]],
            p3 -> subclass[x, image[DIV, set[gcd[set[gcd[x], gcd[y]]]]]]}]] // Reverse

Out[22]= or[not[subclass[x, omega]], subclass[x, image[DIV,
            set[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]]]]] == True

In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[24]:= subclass[set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]], omega] // AssertTest

Out[24]= subclass[set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]], omega] == True

In[25]:= subclass[set[APPLY[GLB[DIV], x_], APPLY[GLB[DIV], y_]], omega] := True
```

Lemma.

```
In[26]:= SubstTest[implies, and[member[t, omega], subclass[z, image[DIV, set[t]]]],
           divides[t, gcd[z]], {t → gcd[set[gcd[x], gcd[y]]], z → union[x, y]}] // Reverse

Out[26]= or[member[pair[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
            APPLY[GLB[DIV], union[x, y]]], DIV], not[subclass[x,
            image[DIV, set[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]]]]],
          not[subclass[y, image[DIV,
             set[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]]]]]] == True

In[27]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. If **x** and **y** are both subsets of **omega**, then **gcd[set[gcd[x],gcd[y]]]** divides **gcd[union[x,y]]**.

```
In[28]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[p1, p4]], {p1 → subclass[union[x, y], omega],
            p2 -> subclass[x, image[DIV, set[gcd[set[gcd[x], gcd[y]]]]]],
            p3 -> subclass[y, image[DIV, set[gcd[set[gcd[x], gcd[y]]]]]],
            p4 → divides[gcd[set[gcd[x], gcd[y]]], gcd[union[x, y]]]}]] // Reverse

Out[28]= or[member[pair[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
            APPLY[GLB[DIV], union[x, y]]], DIV],
          not[subclass[x, omega]], not[subclass[y, omega]]] == True

In[29]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. If **x** and **y** are sets of natural numbers, then **gcd[set[gcd[x],gcd[y]]] = gcd[union[x,y]]**.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[p1, p4]], {p1 -> subclass[union[x, y], omega],
            p2 -> divides[gcd[set[gcd[x], gcd[y]]], gcd[union[x, y]]],
            p3 -> divides[gcd[union[x, y]], gcd[set[gcd[x], gcd[y]]]],
            p4 → equal[gcd[union[x, y]], gcd[set[gcd[x], gcd[y]]]]}]] // Reverse

Out[30]= or[equal[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
            APPLY[GLB[DIV], union[x, y]]],
          not[subclass[x, omega]], not[subclass[y, omega]]] == True

In[31]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The converse also holds.

Lemma.

```
In[32]:= Map[not, SubstTest[and, equal[u, v], equal[v, V],
           {u -> APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
            v -> APPLY[GLB[DIV], union[x, y]]}]] // Reverse

Out[32]= or[and[subclass[x, omega], subclass[y, omega]],
          not[equal[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
            APPLY[GLB[DIV], union[x, y]]]]] == True

In[33]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The equation **gcd[set[gcd[x],gcd[y]]] = gcd[union[x,y]]** holds if and only if **x** and **y** are both subsets of **omega**.

```
In[34]:= equiv[equal[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x], APPLY[GLB[DIV], y]]],
           APPLY[GLB[DIV], union[x, y]]], and[subclass[x, omega], subclass[y, omega]]]

Out[34]= True

In[35]:= equal[APPLY[GLB[DIV], set[APPLY[GLB[DIV], x_], APPLY[GLB[DIV], y_]]],
           APPLY[GLB[DIV], union[x_, y_]]] := and[subclass[x, omega], subclass[y, omega]]
```

the associative law for gcd

In this section a rewrite rule is derived that implies the associative law for gcd's. The associative law itself can not be made into a rewrite rule.

Theorem. Given three numbers, the gcd of the gcd of the first two with the third is the gcd of the set of all three.

```
In[36]:= SubstTest[equal, APPLY[GLB[DIV], set[APPLY[GLB[DIV], u], APPLY[GLB[DIV], v]]],
           APPLY[GLB[DIV], union[u, v]], {u → set[nat[x], nat[y]], v → set[nat[z]]}] // Reverse

Out[36]= equal[APPLY[GLB[DIV], set[APPLY[GLB[DIV], set[nat[x], nat[y]]], nat[z]]],
           APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]]] == True

In[37]:= APPLY[GLB[DIV], set[APPLY[GLB[DIV], set[nat[x_], nat[y_]]], nat[z_]]] :=
          APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]]
```

The associative law for gcd's of pairsets, Quaife's Theorem **(GCD19)**, follows as a corollary. No further rewrite rules are needed.

```
In[38]:= equal[gcd[set[gcd[set[nat[x], nat[y]]], nat[z]]],
          gcd[set[nat[x], gcd[set[nat[y], nat[z]]]]]]

Out[38]= True
```

Quaife's Theorem (GCD20)

Theorem. The gcd of a set of three numbers divides each of the three numbers.

```
In[39]:= SubstTest[implies, and[member[u, v], subclass[v, omega]],
           divides[gcd[v], u], {u → nat[x], v -> set[nat[x], nat[y], nat[z]]}] // Reverse

Out[39]= member[pair[APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]], nat[x]], DIV] == True

In[40]:= member[pair[APPLY[GLB[DIV], set[nat[x_], nat[y_], nat[z_]]], nat[x_]], DIV] := True
```

Theorem. The gcd of a set of three numbers divides the gcd of any pair of these numbers.

```
In[41]:= SubstTest[divides, gcd[union[u, v]], gcd[u],
           {u → set[nat[x], nat[y]], v → set[nat[z]]}] // Reverse

Out[41]= member[pair[APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]],
            APPLY[GLB[DIV], set[nat[x], nat[y]]]], DIV] == True

In[42]:= member[pair[APPLY[GLB[DIV], set[nat[x_], nat[y_], nat[z_]]],
           APPLY[GLB[DIV], set[nat[x_], nat[y_]]]], DIV] := True
```

Quaife considers only certain gcd's of pairsets of two gcd's of pairsets, but one can be perfectly general about this.

Theorem. The gcd of a pairset of two gcd's of pairsets is the gcd of a set of all four numbers.

```
In[43]:= SubstTest[equal, gcd[set[gcd[s], gcd[t]]], gcd[union[s, t]],
           {s → set[nat[u], nat[v]], t → set[nat[x], nat[y]]}] // Reverse

Out[43]= equal[APPLY[GLB[DIV],
            set[APPLY[GLB[DIV], set[nat[u], nat[v]]], APPLY[GLB[DIV], set[nat[x], nat[y]]]]],
           APPLY[GLB[DIV], set[nat[u], nat[v], nat[x], nat[y]]]] == True

In[44]:= APPLY[GLB[DIV], set[APPLY[GLB[DIV], set[nat[u_], nat[v_]]],
            APPLY[GLB[DIV], set[nat[x_], nat[y_]]]]] :=
          APPLY[GLB[DIV], set[nat[u], nat[v], nat[x], nat[y]]]
```

Theorem. Corollary 1 of Quaife's Theorem **(GCD20)**.

```
In[45]:= (SubstTest[implies, and[equal[v, set[0]], divides[u, v]],
            equal[u, set[0]], {u → gcd[union[s, t]], v → gcd[s]}] // Reverse) /.
          {s → set[nat[x], nat[y]], t → set[nat[z]]}

Out[45]= or[equal[APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]], set[0]],
           not[equal[APPLY[GLB[DIV], set[nat[x], nat[y]]], set[0]]]] == True

In[46]:= or[equal[APPLY[GLB[DIV], set[nat[x_], nat[y_], nat[z_]]], set[0]],
           not[equal[APPLY[GLB[DIV], set[nat[x_], nat[y_]]], set[0]]]] := True
```

Lemma.

```
In[47]:= SubstTest[implies,
           and[equal[set[0], gcd[set[nat[u], nat[v]]]], divides[nat[u], nat[z]],
            divides[nat[v], nat[z]]], divides[natmul[nat[u], nat[v]], nat[z]],
           {u → gcd[set[nat[x], nat[z]]], v → gcd[set[nat[y], nat[z]]]}] // Reverse

Out[47]= or[member[pair[natmul[APPLY[GLB[DIV], set[nat[x], nat[z]]],
              APPLY[GLB[DIV], set[nat[y], nat[z]]]], nat[z]], DIV],
           not[equal[APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]], set[0]]]] == True

In[48]:= (% /. {x → x_, y → y_, z → z_}) /. Equal→ SetDelayed
```

Theorem. Corollary 3 of Quaife's Theorem **(GCD20)**.

```
In[49]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            not[implies[p1, p3]], {p1 -> equal[APPLY[GLB[DIV], set[nat[x], nat[y]]], set[0]],
             p2 -> equal[APPLY[GLB[DIV], set[nat[x], nat[y], nat[z]]], set[0]],
             p3 -> member[pair[natmul[APPLY[GLB[DIV], set[nat[x], nat[z]]],
                 APPLY[GLB[DIV], set[nat[y], nat[z]]]], nat[z]], DIV]}]] // Reverse

Out[49]= or[member[pair[natmul[APPLY[GLB[DIV], set[nat[x], nat[z]]],
              APPLY[GLB[DIV], set[nat[y], nat[z]]]], nat[z]], DIV],
           not[equal[APPLY[GLB[DIV], set[nat[x], nat[y]]], set[0]]]] == True

In[50]:= or[member[pair[natmul[APPLY[GLB[DIV], set[nat[x_], nat[z_]]],
              APPLY[GLB[DIV], set[nat[y_], nat[z_]]]], nat[z_]], DIV],
           not[equal[APPLY[GLB[DIV], set[nat[x_], nat[y_]]], set[0]]]] := True
```