Quaife's Theorem (GCD21)

Johan G. F. Belinfante
2009 March 31

```
In[1]:= SetDirectory["l:"]; << goedel.09mar31a; << tools.m

        :Package Title: goedel.09mar31a            2009 March 31 at 8:05 a.m.

        It is now:  2009 Mar 31 at 14:47

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 February 18

        weightlimit = 40
```

summary

Art Quaife's Theorem **(GCD21)** is derived in this notebook. This theorem states that gcd distributes over products of relatively prime numbers. Among the key ingredients are his Corollary 3 of Theorem **(GCD20)** and his Theorem **(LD13)** which states that linear differences of linear differences are linear differences.

```
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";
```

temporary abbreviations

```
In[3]:= divides[x_, y_] := member[pair[x, y], DIV]

In[4]:= gcd[x_] := APPLY[GLB[DIV], x]
```

Quaife's first lemma

Lemma.

```
In[5]:= SubstTest[implies, and[divides[s, u], divides[t, v]],
          divides[natmul[s, t], natmul[u, v]], {s -> gcd[set[nat[x], nat[y]]],
           t -> gcd[set[nat[x], nat[z]]], u → nat[y], v → nat[z]}] // Reverse

Out[5]= member[pair[natmul[APPLY[GLB[DIV], set[nat[x], nat[y]]],
           APPLY[GLB[DIV], set[nat[x], nat[z]]]], natmul[nat[y], nat[z]]], DIV] == True

In[6]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. First lemma for Quaife's Theorem **(GCD21)**.

```
In[7]:= Map[not, SubstTest[and, implies[and[p0, p1], p2],
            implies[p0, p3], implies[and[p2, p3], p4], not[implies[and[p0, p1], p4]],
            {p0 → equal[t, natmul[gcd[set[nat[x], nat[y]]], gcd[set[nat[x], nat[z]]]]],
             p1 → equal[set[0], gcd[set[nat[y], nat[z]]]],
             p2 → divides[t, nat[x]], p3 -> divides[t, natmul[nat[y], nat[z]]],
             p4 → divides[t, gcd[set[nat[x], natmul[nat[y], nat[z]]]]]}] /.
          t -> natmul[gcd[set[nat[x], nat[y]]], gcd[set[nat[x], nat[z]]]] // Reverse

Out[7]= or[member[pair[
            natmul[APPLY[GLB[DIV], set[nat[x], nat[y]]], APPLY[GLB[DIV], set[nat[x], nat[z]]]],
            APPLY[GLB[DIV], set[nat[x], natmul[nat[y], nat[z]]]], DIV],
          not[equal[APPLY[GLB[DIV], set[nat[y], nat[z]]], set[0]]]] == True

In[8]:= or[member[pair[natmul[APPLY[GLB[DIV], set[nat[x_], nat[y_]]],
             APPLY[GLB[DIV], set[nat[x_], nat[z_]]]],
            APPLY[GLB[DIV], set[nat[x_], natmul[nat[y_], nat[z_]]]], DIV],
          not[equal[APPLY[GLB[DIV], set[nat[y_], nat[z_]]], set[0]]]] := True
```

Quaife's second lemma

The case that **nat[x] = 0** needs to be done separately. The following lemma quickly disposes of this case.

Lemma.

```
In[9]:= SubstTest[gcd, set[nat[t]], t -> natmul[nat[x], nat[y]]] // Reverse

Out[9]= APPLY[GLB[DIV], set[natmul[nat[x], nat[y]]]] == natmul[nat[x], nat[y]]

In[10]:= APPLY[GLB[DIV], set[natmul[nat[x_], nat[y_]]]] := natmul[nat[x], nat[y]]
```

Strategy for the case that **nat[x]** is nonzero. Imagine writing each of **gcd[set[nat[x],nat[y]]]** and **gcd[set[nat[x],nat[z]]]** as linear differences and multiply these equations:

gcd[set[nat[x], nat[y]]] = a · nat[x] – b · nat[y]
gcd[set[nat[x], nat[z]]] = c · nat[x] – d · nat[z].

The product is then a linear difference of **nat[x]** and **nat[y] · nat[z]**. To complete the proof we just need the fact that any such linear difference is a multiple of **gcd[set[nat[x], natmul[nat[y], nat[z]]]]**.

Lemma.

```
In[11]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], not[implies[p1, p3]], {p1 → member[nat[u], ld[x, y]],
             p2 → member[natmul[nat[u], nat[v]], ld[natmul[nat[v], x], natmul[nat[v], y]]],
             p3 → member[natmul[nat[u], nat[v]], ld[x, natmul[nat[v], y]]]}]] // Reverse

Out[11]= or[member[natmul[nat[u], nat[v]], ld[x, natmul[y, nat[v]]]],
           not[member[nat[u], ld[x, y]]] == True

In[12]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. A corollary of Quaife's Theorem **(LD13)**.

```
In[13]:= SubstTest[implies, and[member[u, ld[x, z]], member[y, ld[x, z]], member[w, ld[u, y]]],
           member[w, ld[x, z]], {u → x}] // Reverse

Out[13]= or[member[w, ld[x, z]], not[member[w, ld[x, y]]], not[member[y, ld[x, z]]]] == True

In[14]:= or[member[w_, ld[x_, z_]],
           not[member[w_, ld[x_, y_]]], not[member[y_, ld[x_, z_]]]] := True
```

Lemma.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
            implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
            {p1 → member[nat[u], ld[x, nat[y]]], p2 -> member[nat[v], ld[x, z]],
             p3 -> member[natmul[nat[u], nat[v]], ld[x, natmul[nat[y], nat[v]]]],
             p4 -> member[natmul[nat[y], nat[v]], ld[x, natmul[nat[y], z]]],
             p5 → member[natmul[nat[u], nat[v]], ld[x, natmul[nat[y], z]]]}]] // Reverse

Out[15]= or[member[natmul[nat[u], nat[v]], ld[x, natmul[z, nat[y]]]],
           not[member[nat[u], ld[x, nat[y]]]], not[member[nat[v], ld[x, z]]]] == True

In[16]:= (% /. {u → u_, v → v_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[17]:= SubstTest[implies,
           and[member[nat[u], ld[nat[x], nat[y]]], member[nat[v], ld[nat[x], nat[z]]]],
           member[natmul[nat[u], nat[v]], ld[nat[x], natmul[nat[z], nat[y]]]],
           {u → gcd[set[nat[x], nat[y]]], v → gcd[set[nat[x], nat[z]]]}] // Reverse

Out[17]= or[member[natmul[APPLY[GLB[DIV], set[nat[x], nat[y]]],
             APPLY[GLB[DIV], set[nat[x], nat[z]]]], ld[nat[x], natmul[nat[y], nat[z]]]],
           not[member[APPLY[GLB[DIV], set[nat[x], nat[y]]], ld[nat[x], nat[y]]]],
           not[member[APPLY[GLB[DIV], set[nat[x], nat[z]]], ld[nat[x], nat[z]]]]] == True

In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. Quaife's second lemma for the special case that **nat[x]** is nonzero.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           implies[p4, p5], not[implies[p1, p5]], {p1 → not[empty[nat[x]]],
            p2 -> member[APPLY[GLB[DIV], set[nat[x], nat[y]]], ld[nat[x], nat[y]]],
            p3 -> member[APPLY[GLB[DIV], set[nat[x], nat[z]]], ld[nat[x], nat[z]]],
            p4 -> member[natmul[APPLY[GLB[DIV], set[nat[x], nat[y]]],
               APPLY[GLB[DIV], set[nat[x], nat[z]]]], ld[nat[x], natmul[nat[y], nat[z]]]],
            p5 → divides[gcd[set[nat[x], natmul[nat[y], nat[z]]]], natmul[APPLY[GLB[DIV],
                set[nat[x], nat[y]]], APPLY[GLB[DIV], set[nat[x], nat[z]]]]]}]] // Reverse

Out[19]= or[equal[0, nat[x]], member[
           pair[APPLY[GLB[DIV], set[nat[x], natmul[nat[y], nat[z]]]], natmul[APPLY[GLB[DIV],
              set[nat[x], nat[y]]], APPLY[GLB[DIV], set[nat[x], nat[z]]]], DIV]] == True

In[20]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The final step is to combine this case with the case that **nat[x]** is **0**.

Lemma. Second lemma for Quaife's Theorem **(GCD21)**.

```
In[21]:= SubstTest[and, implies[p, q], or[p, q], {p → empty[nat[x]],
           q -> member[pair[APPLY[GLB[DIV], set[nat[x], natmul[nat[y], nat[z]]]], natmul[APPLY[
                GLB[DIV], set[nat[x], nat[y]]], APPLY[GLB[DIV], set[nat[x], nat[z]]]], DIV]}]

Out[21]= member[pair[APPLY[GLB[DIV], set[nat[x], natmul[nat[y], nat[z]]]],
           natmul[APPLY[GLB[DIV], set[nat[x], nat[y]]],
            APPLY[GLB[DIV], set[nat[x], nat[z]]]], DIV] == True

In[22]:= member[pair[APPLY[GLB[DIV], set[nat[x_], natmul[nat[y_], nat[z_]]]],
           natmul[APPLY[GLB[DIV], set[nat[x_], nat[y_]]],
            APPLY[GLB[DIV], set[nat[x_], nat[z_]]]], DIV] := True
```

Theorem (GCD21)

In this section Quaife's two lemmas are combined to obtain his Theorem **(GCD21)**.

Theorem. Quaife's Theorem **(GCD21)**. Distributivity of gcd over the product of relatively prime numbers.

```
In[23]:= SubstTest[and, implies[p, divides[u, v]], divides[v, u],
          {p -> equal[APPLY[GLB[DIV], set[nat[y], nat[z]]], set[0]],
           u -> natmul[gcd[set[nat[x], nat[y]]], gcd[set[nat[x], nat[z]]]],
           v -> gcd[set[nat[x], natmul[nat[y], nat[z]]]]}]

Out[23]= or[equal[APPLY[GLB[DIV], set[nat[x], natmul[nat[y], nat[z]]]],
            natmul[APPLY[GLB[DIV], set[nat[x], nat[y]]], APPLY[GLB[DIV], set[nat[x], nat[z]]]],
           not[equal[APPLY[GLB[DIV], set[nat[y], nat[z]]], set[0]]]] == True

In[24]:= or[equal[APPLY[GLB[DIV], set[nat[x_], natmul[nat[y_], nat[z_]]]], natmul[
             APPLY[GLB[DIV], set[nat[x_], nat[y_]]], APPLY[GLB[DIV], set[nat[x_], nat[z_]]]],
           not[equal[APPLY[GLB[DIV], set[nat[y_], nat[z_]]], set[0]]]] := True
```