linear differences, part 2. almost symmetry

Johan G. F. Belinfante
2007 March 2; corrected 2007 April 4

```
In[1]:= SetDirectory["l:"]; << goedel91.02a; << tools.m

        :Package Title: goedel91.02a      corrected 2007 April 4 at 8:05 p.m.

        It is now:  2007 Apr 4 at 20:9

        Loading Simplification Rules

        TOOLS.M                       Revised 2007 March 2

        weightlimit = 40
```

summary

In this notebook it is shown that if **x** and **y** are natural numbers, and if **y** is not zero, then any linear difference of **x** and **y** is also a linear difference of **y** and **x**. The idea for this theorem, as well as the general outline of the proof follows that given by Art Quaife fifteen years ago, using William McCune's automated reasoning program **Otter**.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical
           Theories, Appendix 3. Thoerems Proved in Peano's Arithmetic, Kluwer
           Academic Publishers, Dordrecht, 1992. Cf. page 201, Theorem (LD8).";
```

The main difference is that the present derivation is done in the general setting of Gödel's class theory, whereas Quaife developed arithmetic based on a set of axioms for arithmetic. A corollary of this symmetry is that the set of linear differences of a pair of natural numbers is the set of multiples of some natural number.

the basic strategy

The basic idea is to use the following equation for natural numbers, which holds provided it makes sense. This is the case when the product **u x** is not less than the product **v y**, the number **w** is sufficiently large, and **y** is not zero:

$$\mathbf{u\,x - v\,y = (w\,x - v)\,y - (w\,y - u)\,x}.$$

The condition that **w** be sufficiently large is met, for example, if one takes **w = max[u, v]**.

lemma

This consequence of the transitive law for **subclass** is needed below.

```
In[3]:= Map[or[empty[nat[y]], #] &,
          SubstTest[implies, and[subclass[nat[v], nat[w]], subclass[nat[w], nat[t]]],
           subclass[nat[v], nat[t]], t → natmul[nat[w], nat[y]]]] // Reverse

Out[3]= or[equal[0, nat[y]], member[nat[w], nat[v]],
          not[member[natmul[nat[w], nat[y]], nat[v]]]] == True

In[4]:= or[equal[0, nat[y_]], member[nat[w_], nat[v_]],
          not[member[natmul[nat[w_], nat[y_]], nat[v_]]]] := True
```

lemma

This monotonicity law for **natmul** will be needed later.

```
In[5]:= SubstTest[member, natmul[nat[x], nat[u]],
          natmul[nat[x], nat[t]], t → natmul[nat[w], nat[y]]] // Reverse

Out[5]= member[natmul[nat[u], nat[x]], natmul[nat[w], nat[x], nat[y]]] ==
         and[member[nat[u], natmul[nat[w], nat[y]]], not[equal[0, nat[x]]]]

In[6]:= member[natmul[nat[u_], nat[x_]], natmul[nat[w_], nat[x_], nat[y_]]] :=
         and[member[nat[u], natmul[nat[w], nat[y]]], not[equal[0, nat[x]]]]
```

lemma

The following lemma verifies that the choice made for **w** suffices to meet the desired conditions.

```
In[7]:= SubstTest[implies, and[subclass[r, s], equal[s, t], subclass[t, u]], subclass[r, u],
          {r → nat[v], s → union[nat[u], nat[v]], t → nat[w], u -> natmul[nat[w], nat[x]]}] //
         Reverse

Out[7]= or[equal[0, nat[x]], not[equal[nat[w], union[nat[u], nat[v]]]],
          not[member[natmul[nat[w], nat[x]], nat[v]]]] == True

In[8]:= or[equal[0, nat[x_]], not[equal[nat[w_], union[nat[u_], nat[v_]]]],
          not[member[natmul[nat[w_], nat[x_]], nat[v_]]]] := True
```

tactical considerations

One needs several equations to define three intermediate variables **p**, **q** and **w** about which statements are proved. Although these variables are eventually eliminated, the abundance of possible equality substitutions arising from all these equations threatens to cause combinatorial explosion. In addition, the addition of extra literals produces a combinatorial explosion in the Boolean algebra needed to verify the correctness of the derivation. To keep down the number of literals, the steps where the extra variables are essential are identified and that part of the derivation is separated into one hard lemma, thereby confining the extra literals to a small part of the overall proof. Equality substitution in the **GOEDEL** program is not symmetric with respect to negation. This asymmetry is exploited so that equality is used to derive negative facts, but these are not to be turned into positive facts via negation until after the temporary variables have been elminated. By delaying negation until after the variables **p** and **q** are eliminated, the execution time for the hard lemma is reduced from 72 seconds to about 4 seconds.

hard lemma

The literals **p6** and **p7** are stumbling blocks, and are the main reason that the abbreviations **p** and **q** are needed at all. The following lemma would take 72 seconds if the variables **p** and **q** were eliminated after the **Map[not, ...].** By delaying the negation until after the variables **p** and **q** are eliminated, the execution time is reduced to about 4 seconds.

```
In[9]:= Map[not, (SubstTest[and, or[p6, not[p3b], not[p4b], not[p5]],
              or[p7, not[p1f], not[p1j], not[p3a], not[p4a]], or[p8, not[p6], not[p7]],
              not[implies[and[p1f, p1j, p3a, p4a, p3b, p4b, p5], p8]],
              {p1f → not[member[natmul[nat[w], nat[x]], nat[v]]],
               p1j → not[member[natmul[nat[w], nat[y]], nat[u]]],
               p3a → equal[p, natsub[natmul[nat[w], nat[x]], nat[v]]],
               p3b → member[p, omega], p4a → equal[q, natsub[natmul[nat[w], nat[y]], nat[u]]],
               p4b → member[q, omega], p5 → subclass[natmul[q, nat[x]], natmul[p, nat[y]]],
               p6 → member[pair[nat[x], natsub[natmul[p, nat[y]], natmul[q, nat[x]]]],
                 composite[image[inverse[NATADD], image[DIV, set[nat[y]]]], DIV]],
               p7 → equal[natsub[natmul[p, nat[y]], natmul[q, nat[x]]],
                 natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]]],
               p8 → member[natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]],
                 ld[nat[y], nat[x]]]} ]) /. {p -> natsub[natmul[nat[w], nat[x]], nat[v]],
            q -> natsub[natmul[nat[w], nat[y]], nat[u]]} // Reverse

Out[9]= or[member[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]],
          member[natmul[nat[w], nat[x]], nat[v]],
          member[natmul[nat[w], nat[y]], nat[u]], member[
           natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], ld[nat[y], nat[x]]]] == True

In[10]:= (% /. {u → u_, v → v_, w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

main theorem

For the main theorem, again the execution time is reduced by delaying the **Map[not, ..]** until after the variables **p** and **q** are eliminated, this time from 22 seconds to 15 seconds. The variable **w** needs to be eliminated after **p** and **q** are gone.

```
In[11]:= Map[not, (SubstTest[and, implies[and[p0a, p1a], p1f], implies[and[p0b, p1a], p1j],
              or[p5, not[p0c], not[p1j], not[p3a], not[p4a]], implies[and[p0c, p1f, p1j], p8],
              not[implies[and[p0a, p0b, p0c, p1a, p3a, p4a], p8]],
              {p0a → not[equal[0, nat[x]]], p0b -> not[equal[0, nat[y]]],
               p0c -> not[member[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]]],
               p1a → equal[nat[w], union[nat[u], nat[v]]],
               p1f → not[member[natmul[nat[w], nat[x]], nat[v]]],
               p1j → not[member[natmul[nat[w], nat[y]], nat[u]]],
               p3a → equal[p, natsub[natmul[nat[w], nat[x]], nat[v]]],
               p4a → equal[q, natsub[natmul[nat[w], nat[y]], nat[u]]],
               p5 → subclass[natmul[q, nat[x]], natmul[p, nat[y]]],
               p8 → member[natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]],
                 ld[nat[y], nat[x]]]} ] /. {p -> natsub[natmul[nat[w], nat[x]], nat[v]],
              q -> natsub[natmul[nat[w], nat[y]], nat[u]]}) /.
            w -> union[nat[u], nat[v]]] // Reverse

Out[11]= or[equal[0, nat[x]], equal[0, nat[y]],
           member[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], member[
            natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], ld[nat[y], nat[x]]]] == True

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

eliminating a redundant literal

The condition that **nat[x]** be nonzero is not really needed.

```
In[13]:= SubstTest[and, implies[p, q], or[p, q], {p → equal[0, nat[x]], q ->
            or[equal[0, nat[y]], member[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], member[
              natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], ld[nat[y], nat[x]]]]}]

Out[13]= or[equal[0, nat[y]], member[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], member[
            natsub[natmul[nat[u], nat[x]], natmul[nat[v], nat[y]]], ld[nat[y], nat[x]]]] == True

In[14]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

eliminating u and v

Lemma.

```
In[15]:= member[pair[u, v], composite[inverse[times[x]], z, times[y]]] // AssertTest

Out[15]= member[pair[u, v], composite[inverse[times[x]], z, times[y]]] ==
          and[member[u, omega], member[v, omega], member[x, omega],
           member[y, omega], member[pair[natmul[u, y], natmul[v, x]], z]]

In[16]:= member[pair[u_, v_], composite[inverse[times[x_]], z_, times[y_]]] :=
          and[member[u, omega], member[v, omega], member[x, omega],
           member[y, omega], member[pair[natmul[u, y], natmul[v, x]], z]]
```

First, the **nat** wrappers on **u** and **v** are removed:

```
In[17]:= SubstTest[implies, and[equal[u, nat[s]], equal[v, nat[t]]],
           implies[and[not[equal[0, nat[y]]], subclass[natmul[v, nat[y]], natmul[u, nat[x]]]],
            member[natsub[natmul[u, nat[x]], natmul[v, nat[y]]], ld[nat[y], nat[x]]],
           {s → u, t → v}] // Reverse

Out[17]= or[equal[0, nat[y]],
           member[natsub[natmul[u, nat[x]], natmul[v, nat[y]]], ld[nat[y], nat[x]]],
           not[member[u, omega]], not[member[v, omega]],
           not[subclass[natmul[v, nat[y]], natmul[u, nat[x]]]]] == True

In[18]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Now the variables **u** and **v** are eliminated.

```
In[19]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[u, v], implies[
            and[not[empty[t]], member[pair[u, v], w]], member[pair[u, v], z]], {t → nat[y],
            w -> domain[composite[rotate[NATADD], cross[times[nat[x]], times[nat[y]]]]],
            z -> composite[inverse[times[nat[y]]],
              image[inverse[rotate[NATADD]], ld[nat[y], nat[x]]], times[nat[x]]]}]]

Out[19]= or[equal[0, nat[y]], subclass[
            composite[id[image[DIV, set[nat[y]]]], inverse[S], id[image[DIV, set[nat[x]]]]],
            composite[FIRST, id[cart[V, ld[nat[y], nat[x]]]], inverse[NATADD]]]] == True

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

Lemma.

```
In[21]:= ImageComp[rotate[NATADD], inverse[rotate[NATADD]], z] // Reverse

Out[21]= range[fix[composite[inverse[FIRST], FIRST, id[cart[V, z]], inverse[NATADD], NATADD]]] ==
          intersection[omega, z]

In[22]:= range[fix[composite[inverse[FIRST], FIRST,
             id[cart[V, z_]], inverse[NATADD], NATADD]]] := intersection[omega, z]
```

The variable-free result obtained above can be reformulated in a more understandable form: the set **ld[nat[x],nat[y]]** is (almost) symmetric in **x** and **y**.

```
In[23]:= SubstTest[implies, or[p, subclass[s, t]], or[p, subclass[image[r, s], image[r, t]]],
           {p → equal[0, nat[y]], r → rotate[NATADD], s ->
             composite[id[image[DIV, set[nat[y]]]], inverse[S], id[image[DIV, set[nat[x]]]]],
            t -> composite[FIRST, id[cart[V, image[image[inverse[NATADD], image[DIV,
                   set[nat[y]]]], image[DIV, set[nat[x]]]]]], inverse[NATADD]]}] // Reverse

Out[23]= or[equal[0, nat[y]], subclass[ld[nat[x], nat[y]], ld[nat[y], nat[x]]]] == True

In[24]:= or[equal[0, nat[y_]], subclass[ld[nat[x_], nat[y_]], ld[nat[y_], nat[x_]]]] := True
```

closure under addition

Linear differences are closed under addition.

```
In[25]:= SubstTest[subclass, image[NATADD,
            cart[image[image[inverse[NATADD], t], u], image[image[inverse[NATADD], v], w]]],
           image[image[inverse[NATADD], image[NATADD, cart[t, v]]], image[NATADD, cart[u, w]]],
           {t → image[DIV, set[x]], u → image[DIV, set[y]],
            v → image[DIV, set[x]], w → image[DIV, set[y]]}] // Reverse

Out[25]= subclass[image[NATADD, cart[ld[x, y], ld[x, y]]], ld[x, y]] == True

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

Restatement.

```
In[27]:= member[ld[x, y], binclosed[NATADD]]

Out[27]= True
```

(almost) closure under subtraction

The set of linear differences satisfies the following property under subtraction, in which both **ld[x,y]** and **ld[y,x]** occur.

```
In[28]:= SubstTest[subclass, image[image[inverse[NATADD], image[image[inverse[NATADD], t], u]],
            image[image[inverse[NATADD], v], w]],
           image[image[inverse[NATADD], image[NATADD, cart[t, w]]], image[NATADD, cart[u, v]]],
           {t → image[DIV, set[x]], u → image[DIV, set[y]],
            v -> image[DIV, set[y]], w → image[DIV, set[x]]}] // Reverse

Out[28]= subclass[image[image[inverse[NATADD], ld[x, y]], ld[y, x]], ld[x, y]] == True

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

If a set of natural numbers is closed under both addition and subtraction, then it is the set of multiples of some natural number.

```
In[30]:= Map[implies[#, member[x, range[VERTSECT[DIV]]]] &,
          SubstTest[member, x, intersection[u, v, w],
           {u → binclosed[NATADD], v -> binclosed[rotate[NATADD]], w -> P[omega]}]]

Out[30]= or[member[x, range[VERTSECT[DIV]]],
           not[subclass[x, omega]], not[subclass[image[NATADD, cart[x, x]], x]],
           not[subclass[image[image[inverse[NATADD], x], x], x]]] == True

In[31]:= or[member[x_, range[VERTSECT[DIV]]],
           not[subclass[x_, omega]], not[subclass[image[NATADD, cart[x_, x_]], x_]],
           not[subclass[image[image[inverse[NATADD], x_], x_], x_]]] := True
```

Corollary.

```
In[32]:= SubstTest[implies, and[equal[s, t], subclass[image[NATADD, cart[t, t]], t],
            subclass[image[rotate[NATADD], cart[t, s]], t], subclass[t, omega]],
           member[t, range[VERTSECT[DIV]]], {s → ld[y, x], t → ld[x, y]}] // Reverse

Out[32]= or[member[ld[x, y], range[VERTSECT[DIV]]], not[equal[ld[x, y], ld[y, x]]]] == True

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

Even though the hypothesis does not hold when $\mathbf{y} = \mathbf{0}$, this case is not an exception.

```
In[34]:= SubstTest[and, implies[p1, subclass[u, v]], implies[p2, subclass[v, u]],
           {p1 → not[equal[0, nat[y]]], p2 → not[equal[0, nat[x]]],
            u → ld[nat[x], nat[y]], v → ld[nat[y], nat[x]]}] // MapNotNot

Out[34]= or[equal[0, nat[x]], equal[0, nat[y]],
           equal[ld[nat[x], nat[y]], ld[nat[y], nat[x]]]] == True

In[35]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The set of linear differences of any two natural numbers is the set of multiples of some number.

```
In[36]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
             implies[p3, p4], implies[not[p1], p4], implies[not[p2], p4], not[p4],
             {p1 → not[equal[0, nat[x]]], p2 -> not[equal[0, nat[y]]],
              p3 → equal[ld[nat[x], nat[y]], ld[nat[y], nat[x]]],
              p4 → member[ld[nat[x], nat[y]], range[VERTSECT[DIV]]]}]] // Reverse // MapNotNot

Out[36]= member[ld[nat[x], nat[y]], range[VERTSECT[DIV]]] == True

In[37]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

removing the nat wrappers

Removing the **nat** wrappers yields:

```
In[38]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]]],
           member[ld[x, y], range[VERTSECT[DIV]]], {u → x, v → y}] // Reverse

Out[38]= or[member[ld[x, y], range[VERTSECT[DIV]]],
           not[member[x, omega]], not[member[y, omega]]] == True

In[39]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The numberhood literal for **y** is redundant.

```
In[40]:= (Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
             {p1 → and[not[member[y, omega]], equal[t, ld[x, y]]], p2 → equal[0, t],
              p3 → member[t, range[VERTSECT[DIV]]]}]] // Reverse) /. t → ld[x, y]

Out[40]= or[member[y, omega], member[ld[x, y], range[VERTSECT[DIV]]]] == True

In[41]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The numberhood literal for **x** is also redundant.

```
In[42]:= SubstTest[implies, equal[0, z],
           equal[0, image[rotate[NATADD], cart[z, image[DIV, set[y]]]]],
           z → image[DIV, set[x]]] // Reverse

Out[42]= or[member[x, omega], not[member[y,
             image[inverse[DIV], domain[image[inverse[NATADD], image[DIV, set[x]]]]]]] == True

In[43]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[44]:= SubstTest[implies, equal[0, t],
             member[t, range[VERTSECT[DIV]]], t → ld[x, y]] // Reverse // MapNotNot

Out[44]= or[member[x, omega], member[ld[x, y], range[VERTSECT[DIV]]]] == True

In[45]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The redundant numberhood literals can be removed.

```
In[46]:= SubstTest[and, implies[p, q], or[p, q], {p → and[member[x, omega], member[y, omega]],
             q → member[ld[x, y], range[VERTSECT[DIV]]]}] // MapNotNot

Out[46]= member[ld[x, y], range[VERTSECT[DIV]]] == True

In[47]:= member[ld[x_, y_], range[VERTSECT[DIV]]] := True
```

This statement says that **ld[x, y]** is either empty, or is the set of multiples of some natural number.