Quaife's (LD) group

Johan G. F. Belinfante and Ming Li
2007 April 17

```
In[1]:= SetDirectory["l:"]; << goedel92.12a; << tools.m

        :Package Title: goedel92.12a        2007 April 12 at 8:30 p.m.

        It is now:  2007 Apr 17 at 13:40

        Loading Simplification Rules

        TOOLS.M                      Revised 2007 March 25

        weightlimit = 40
```

summary

In this notebook various clauses of Quaife's **(LD)** group are derived, especially those about closure properties of 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. 200–202";
```

simplification rules

Lemma.

```
In[3]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
           {p1 → member[natmul[x, y], z], p2 → member[x, omega], p3 → member[x, V]}]] // Reverse

Out[3]= or[member[x, V], not[member[natmul[x, y], z]]] == True

In[4]:= or[member[x_, V], not[member[natmul[x_, y_], z_]]] := True
```

Lemma.

```
In[5]:= equiv[and[member[x, V], member[natmul[x, y], z]], member[natmul[x, y], z]]

Out[5]= True

In[6]:= and[member[x_, V], member[natmul[x_, y_], z_]] := member[natmul[x, y], z]
```

Simplification rule:

```
In[7]:= equal[and[member[w, ld[x, y]], member[x, omega]], member[w, ld[x, y]]]
Out[7]= equal[and[member[w, ld[x, y]], member[x, omega]], member[w, ld[x, y]]]

In[8]:= and[member[w_, ld[x_, y_]], member[x_, omega]] := member[w, ld[x, y]]
```

special cases

Theorem.

```
In[9]:= ld[x, V] // Normality
Out[9]= ld[x, V] == 0

In[10]:= ld[x_, V] := 0
```

Theorem.

```
In[11]:= ld[V, x] // Normality
Out[11]= ld[V, x] == 0

In[12]:= ld[V, x_] := 0
```

Quaife's theorem (LD9)

Quaife's theorem **(LD9)** asserts that the set **ld[x, y]** is closed under addition.

```
In[13]:= SubstTest[implies, subclass[r, s], subclass[image[t, r], image[t, s]],
           {t -> NATADD, r → set[PAIR[u, v]], s → cartsq[ld[x, y]]}] // Reverse
Out[13]= or[member[natadd[u, v], ld[x, y]],
           not[member[u, ld[x, y]]], not[member[v, ld[x, y]]]] == True

In[14]:= or[member[natadd[u_, v_], ld[x_, y_]],
           not[member[u_, ld[x_, y_]]], not[member[v_, ld[x_, y_]]]] := True
```

Quaife's clause (LD10a)

The first clause of Quaife's theorem **(LD10)** is a corollary of a distributive law for **monus**. To derive it, a lemma is needed:

```
In[15]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]]],
           equal[natmul[nat[w], monus[x, y]], monus[natmul[nat[w], x], natmul[nat[w], y]]],
           {u → x, v → y}] // Reverse // MapNotNot

Out[15]= or[equal[0, nat[w]],
          equal[natadd[natmul[y, nat[w]], natmul[nat[w], nat[natsub[x, y]]]],
           natmul[x, nat[w]]], not[member[x, omega]], not[member[y, x]]] == True

In[16]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. This rewrite rule suffices to recognize Quaife's clause **(LD10a)**:

```
In[17]:= equal[natmul[nat[x], nat[natsub[y, z]]],
           nat[natsub[natmul[y, nat[x]], natmul[z, nat[x]]]]]

Out[17]= True

In[18]:= natmul[nat[x_], nat[natsub[y_, z_]]] :=
           nat[natsub[natmul[y, nat[x]], natmul[z, nat[x]]]]
```

(LD10b) and (LD10c)

Lemma.

```
In[19]:= SubstTest[implies, and[member[w, u], subclass[u, v]],
           member[w, v], {u -> image[DIV, set[z]], v -> image[DIV, t]}] // Reverse

Out[19]= or[member[w, image[DIV, t]], not[member[pair[z, w], DIV]],
           not[subclass[image[DIV, set[z]], image[DIV, t]]]] == True

In[20]:= (% /. {t → t_, w → w_, z → z_}) /. Equal → SetDelayed
```

Theorem. Quaife's clauses **(LD10b)** and **(LD10c)** say that the set **ld[x, y]** is closed under multiplication.

```
In[21]:= ((Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4], not[
                 implies[and[p1, p2], p4]], {p1 → member[w, s], p2 -> member[pair[w, t], DIV], p3 →
                  subclass[image[DIV, set[w]], image[DIV, s]], p4 → member[t, image[DIV, s]]}]] //
             Reverse) /. {s → ld[x, y], t → natmul[nat[z], w]})

Out[21]= or[member[natmul[w, nat[z]], ld[x, y]], not[member[w, ld[x, y]]]] == True

In[22]:= or[member[natmul[w_, nat[z_]], ld[x_, y_]], not[member[w_, ld[x_, y_]]]] := True
```

Quaife's theorem (LD12)

Quaife's theorem **(LD12)** states that the set **ld[x,y]** of linear differences is closed under floored subtraction. There are two cases to consider: **monus[x,y]** is either equal to **0** or to **natsub[x,y]**. When either **x** or **y** fails to be a natural number, the set **ld[x,y]** of linear differences is **0**. A simplification rule for this case helps to avoid redundant literals.

Lemma. (The case that **monus** is **0**.)

```
In[23]:= Map[implies[and[member[u, ld[x, y]], member[v, ld[x, y]]], #] &,
           SubstTest[or, member[nat[natsub[u, v]], z], not[member[0, z]],
            not[subclass[u, v]], z → ld[x, y]]] // Reverse // MapNotNot

Out[23]= or[member[nat[natsub[u, v]], ld[x, y]], not[member[u, ld[x, y]]],
           not[member[v, ld[x, y]]], not[subclass[u, v]]] == True

In[24]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

A variable-free formulation of closure under subtraction is already available. Introducing a variable yields

```
In[25]:= SubstTest[implies, subclass[r, s], subclass[image[t, r], image[t, s]],
           {t → rotate[NATADD], r → set[PAIR[u, v]], s → cartsq[ld[x, y]]}] // Reverse

Out[25]= or[member[natsub[u, v], ld[x, y]], not[member[u, ld[x, y]]],
           not[member[v, ld[x, y]]], not[subclass[v, u]]] == True

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

Corollary. (The case that **monus** is **natsub**.)

```
In[27]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 → and[member[u, ld[x, y]], member[v, ld[x, y]], subclass[v, u]],
             p2 -> member[natsub[u, v], ld[x, y]], p3 → member[monus[u, v], ld[x, y]]}]] // Reverse

Out[27]= or[member[nat[natsub[u, v]], ld[x, y]], not[member[u, ld[x, y]]],
           not[member[v, ld[x, y]]], not[subclass[v, u]]] == True

In[28]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Combining these lemmas yields Quaife's Theorem **(LD12)**.

```
In[29]:= Map[not, SubstTest[and, implies[and[p2, p3], or[p4, p5]], not[implies[p1, p6]],
             {p1 → and[member[u, ld[x, y]], member[v, ld[x, y]]], p2 → member[u, omega],
              p3 → member[v, omega], p4 → subclass[v, u], p5 → subclass[u, v],
              p6 → member[monus[u, v], ld[x, y]]}]] // Reverse // MapNotNot

Out[29]= or[member[nat[natsub[u, v]], ld[x, y]],
           not[member[u, ld[x, y]]], not[member[v, ld[x, y]]]] == True

In[30]:= or[member[nat[natsub[u_, v_]], ld[x_, y_]],
           not[member[u_, ld[x_, y_]]], not[member[v_, ld[x_, y_]]]] := True
```

(LD13)

Theorem **(LD13)** says that a linear difference of a linear difference is a linear difference.

```
In[31]:= Map[or[not[member[u, ld[x, y]]], not[member[v, ld[x, y]]], #] &,
           SubstTest[implies, subclass[s, t], subclass[image[r, s], image[r, t]],
            {r → composite[rotate[NATADD], cross[DIV, DIV]],
             s → set[PAIR[u, v]], t → cartsq[ld[x, y]]}]] // Reverse // MapNotNot

Out[31]= or[not[member[u, ld[x, y]]],
           not[member[v, ld[x, y]]], subclass[ld[u, v], ld[x, y]]] == True

In[32]:= or[not[member[u_, ld[x_, y_]]],
           not[member[v_, ld[x_, y_]]], subclass[ld[u_, v_], ld[x_, y_]]] := True
```

Introducing an extra variable yields Quaife's Theorem **(LD13)**.

```
In[33]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 → and[member[w, ld[u, v]], member[u, ld[x, y]], member[v, ld[x, y]]],
             p2 -> subclass[ld[u, v], ld[x, y]], p3 → member[w, ld[x, y]]}]] // Reverse

Out[33]= or[member[w, ld[x, y]], not[member[u, ld[x, y]]],
           not[member[v, ld[x, y]]], not[member[w, ld[u, v]]]] == True

In[34]:= or[member[w_, ld[x_, y_]], not[member[u_, ld[x_, y_]]],
           not[member[v_, ld[x_, y_]]], not[member[w_, ld[u_, v_]]]] := True
```

Corollary.

```
In[35]:= SubstTest[implies, and[member[r, ld[u, v]], member[u, ld[x, y]], member[v, ld[x, y]]],
           member[r, ld[x, y]], r → monus[natmul[nat[s], u], natmul[nat[t], v]]] // Reverse

Out[35]= or[member[nat[natsub[natmul[u, nat[s]], natmul[v, nat[t]]]], ld[x, y]],
           not[member[u, ld[x, y]]], not[member[v, ld[x, y]]]] == True

In[36]:= or[member[nat[natsub[natmul[u_, nat[s_]], natmul[v_, nat[t_]]]], ld[x_, y_]],
           not[member[u_, ld[x_, y_]]], not[member[v_, ld[x_, y_]]]] := True
```

(LD15)

Lemma.

```
In[37]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
            {u → nat[y], v → intersection[image[V, nat[x]], ld[nat[y], nat[x]]],
             w → ld[nat[x], nat[y]]}] // Reverse // MapNotNot

Out[37]= or[equal[0, nat[x]], member[nat[y], ld[nat[x], nat[y]]]] == True

In[38]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[39]:= equiv[member[nat[y], ld[nat[x], nat[y]]],
            or[equal[0, nat[y]], not[equal[0, nat[x]]]]] // not // not

Out[39]= True

In[40]:= member[nat[y_], ld[nat[x_], nat[y_]]] := or[equal[0, nat[y]], not[equal[0, nat[x]]]]
```

Lemma. This has a redundant literal.

```
In[41]:= SubstTest[implies, and[member[u, ld[nat[x], nat[y]]], member[v, ld[nat[x], nat[y]]]],
            member[monus[natmul[nat[s], u], natmul[nat[t], v]], ld[nat[x], nat[y]]], {s → set[0],
             t → natquot[nat[x], nat[y]], u → nat[x], v → nat[y]}] // Reverse // MapNotNot

Out[41]= or[equal[0, nat[x]], member[natmod[nat[x], nat[y]], ld[nat[x], nat[y]]]] == True

In[42]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. Temporary version with **nat** wrappers.

```
In[43]:= SubstTest[and, implies[p, q], or[p, q],
          {p -> equal[0, nat[x]], q -> member[natmod[nat[x], nat[y]], ld[nat[x], nat[y]]]}]

Out[43]= member[natmod[nat[x], nat[y]], ld[nat[x], nat[y]]] == True

In[44]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the wrappers:

```
In[45]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]]],
            member[natmod[x, y], ld[x, y]], {u → x, v → y}] // Reverse

Out[45]= or[member[natmod[x, y], ld[x, y]], not[member[x, omega]], not[member[y, omega]]] == True

In[46]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. Quaife's Theorem **(LD15)** says that **natmod[x,y]** belongs to **ld[x,y]** for any pair of natural numbers.

```
In[47]:= equiv[member[natmod[x, y], ld[x, y]], and[member[x, omega], member[y, omega]]] // not //
          not

Out[47]= True

In[48]:= member[natmod[x_, y_], ld[x_, y_]] := and[member[x, omega], member[y, omega]]
```

(LD17)

Lemma.

```
In[49]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → rotate[NATADD], u → cart[image[DIV, set[nat[x]]], image[DIV, set[nat[y]]]],
            v → cartsq[image[DIV, set[w]]]}] // Reverse // MapNotNot

Out[49]= or[not[member[pair[w, nat[x]], DIV]], not[member[pair[w, nat[y]], DIV]],
           subclass[ld[nat[x], nat[y]], image[DIV, set[w]]]] == True

In[50]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the **nat** wrappers yields:

```
In[51]:= SubstTest[implies, and[equal[x, nat[u]], equal[y, nat[v]]],
           or[not[member[pair[w, x], DIV]], not[member[pair[w, y], DIV]],
            subclass[ld[x, y], image[DIV, set[w]]]], {u → x, v → y}] // Reverse // MapNotNot

Out[51]= or[not[member[pair[w, x], DIV]],
           not[member[pair[w, y], DIV]], subclass[ld[x, y], image[DIV, set[w]]]] == True

In[52]:= or[not[member[pair[w_, x_], DIV]], not[member[pair[w_, y_], DIV]],
           subclass[ld[x_, y_], image[DIV, set[w_]]]] := True
```

Quaife's Theorem **(LD17)**.

```
In[53]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2, p3], p5]],
           {p1 -> member[pair[w, x], DIV], p2 -> member[pair[w, y], DIV],
            p3 → member[z, ld[x, y]], p4 -> subclass[ld[x, y], image[DIV, set[w]]],
            p5 → member[pair[w, z], DIV]}]] // Reverse

Out[53]= or[member[pair[w, z], DIV], not[member[z, ld[x, y]]],
           not[member[pair[w, x], DIV]], not[member[pair[w, y], DIV]]] == True

In[54]:= or[member[pair[w_, z_], DIV], not[member[pair[w_, x_], DIV]],
           not[member[pair[w_, y_], DIV]], not[member[z_, ld[x_, y_]]]] := True
```

(LD18)

Theorem **(LD18a)**.

```
In[55]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → times[nat[z]], u → set[w], v → ld[x, y]}] // Reverse

Out[55]= or[member[natmul[w, nat[z]], ld[natmul[x, nat[z]], natmul[y, nat[z]]]],
           not[member[w, ld[x, y]]]] == True

In[56]:= or[member[natmul[w_, nat[z_]], ld[natmul[x_, nat[z_]], natmul[y_, nat[z_]]]],
           not[member[w_, ld[x_, y_]]]] := True
```

The following corollary of **(LD18a)** will be used in the derivation of **(LD19)**.

```
In[57]:= SubstTest[implies, member[t, ld[x, y]], member[natmul[t, nat[w]],
           ld[natmul[x, nat[w]], natmul[y, nat[w]]]], t → set[0]] // Reverse

Out[57]= or[member[nat[w], ld[natmul[x, nat[w]], natmul[y, nat[w]]]],
          not[member[set[0], ld[x, y]]]] == True

In[58]:= or[member[nat[w_], ld[natmul[x_, nat[w_]], natmul[y_, nat[w_]]]],
          not[member[set[0], ld[x_, y_]]]] := True
```

The key observation for the derivation of Theorem **(LD18b)** is this rewrite rule:

```
In[59]:= oopart[times[x]]

Out[59]= composite[times[x], id[image[V, x]]]
```

Theorem **(LD18b)**.

```
In[60]:= Map[or[equal[0, u], member[v, #],
            not[member[natmul[u, v], ld[natmul[u, x], natmul[u, y]]]]] &, SubstTest[image,
           inverse[oopart[t]], image[oopart[t], ld[x, y]], t → times[u]]] // MapNotNot

Out[60]= or[equal[0, u], member[v, ld[x, y]],
          not[member[natmul[u, v], ld[natmul[u, x], natmul[u, y]]]]] == True

In[61]:= or[equal[0, u_], member[v_, ld[x_, y_]],
          not[member[natmul[u_, v_], ld[natmul[u_, x_], natmul[u_, y_]]]]] := True
```

LD19

Lemma.

```
In[62]:= SubstTest[subclass, ld[natmul[x, nat[w]], natmul[t, nat[v]]],
          ld[x, t], {t -> natmul[z, nat[w]], v → set[0]}] // Reverse

Out[62]= subclass[ld[natmul[x, nat[w]], natmul[z, nat[w]]], ld[x, natmul[z, nat[w]]]] == True

In[63]:= subclass[ld[natmul[x_, nat[w_]], natmul[z_, nat[w_]]],
          ld[x_, natmul[z_, nat[w_]]]] := True
```

Lemma.

```
In[64]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → member[set[0], ld[x, y]],
            p2 -> member[nat[z], ld[natmul[x, nat[z]], natmul[y, nat[z]]]],
            p3 → member[nat[z], ld[x, natmul[y, nat[z]]]]}]] // Reverse

Out[64]= or[member[nat[z], ld[x, natmul[y, nat[z]]]], not[member[set[0], ld[x, y]]]] == True

In[65]:= or[member[nat[z_], ld[x_, natmul[y_, nat[z_]]]],
          not[member[set[0], ld[x_, y_]]]] := True
```

Lemma. using **(LD13)**

```
In[66]:= SubstTest[implies, and[member[r, ld[nat[x], t]],
           member[s, ld[nat[x], t]], member[u, ld[r, s]]], member[u, ld[nat[x], t]],
          {t → natmul[nat[y], nat[z]], r → nat[x], s → nat[z]}] // Reverse

Out[66]= or[member[u, ld[nat[x], natmul[nat[y], nat[z]]]], not[member[u, ld[nat[x], nat[z]]]],
          not[member[nat[z], ld[nat[x], natmul[nat[y], nat[z]]]]] == True

In[67]:= or[member[u_, ld[nat[x_], natmul[nat[y_], nat[z_]]]],
          not[member[u_, ld[nat[x_], nat[z_]]]],
          not[member[nat[z_], ld[nat[x_], natmul[nat[y_], nat[z_]]]]] := True
```

Some assembly is required to obtain this **nat**-wrapped version Theorem **(LD19)**:

```
In[68]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 -> member[set[0], ld[nat[x], nat[y]]], p2 -> member[u, ld[nat[x], nat[z]]],
            p3 -> member[nat[z], ld[nat[x], natmul[nat[y], nat[z]]]],
            p4 -> member[u, ld[nat[x], natmul[nat[y], nat[z]]]]}]] // Reverse

Out[68]= or[member[u, ld[nat[x], natmul[nat[y], nat[z]]]],
          not[member[u, ld[nat[x], nat[z]]]], not[member[set[0], ld[nat[x], nat[y]]]] == True

In[69]:= (% /. {u → u_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

a corollary and a wrapper-free version of LD19

Variable elimination:

```
In[70]:= Map[equal[V, #] &,
          SubstTest[class, u, or[member[u, v], not[member[u, w]], not[member[set[0], z]]],
           {v -> ld[nat[x], natmul[nat[y], nat[z]]],
            w -> ld[nat[x], nat[z]], z -> ld[nat[x], nat[y]]}]]

Out[70]= or[not[member[set[0], ld[nat[x], nat[y]]]],
          subclass[ld[nat[x], nat[z]], ld[nat[x], natmul[nat[y], nat[z]]]] == True

In[71]:= or[not[member[set[0], ld[nat[x_], nat[y_]]]],
          subclass[ld[nat[x_], nat[z_]], ld[nat[x_], natmul[nat[y_], nat[z_]]]] := True
```

Lemma.

```
In[72]:= SubstTest[subclass, ld[natmul[u, nat[x]], natmul[v, nat[y]]],
          ld[nat[x], nat[y]], v → set[0]] // Reverse

Out[72]= subclass[ld[natmul[u, nat[x]], nat[y]], ld[nat[x], nat[y]]] == True

In[73]:= subclass[ld[natmul[u_, nat[x_]], nat[y_]], ld[nat[x_], nat[y_]]] := True
```

Similarly:

```
In[74]:= SubstTest[subclass, ld[natmul[u, nat[x]], natmul[v, nat[y]]],
           ld[nat[x], nat[y]], u → set[0]] // Reverse

Out[74]= subclass[ld[nat[x], natmul[v, nat[y]]], ld[nat[x], nat[y]]] == True

In[75]:= subclass[ld[nat[x_], natmul[v_, nat[y_]]], ld[nat[x_], nat[y_]]] := True
```

Corollary.

```
In[76]:= SubstTest[and, implies[p, subclass[u, v]],
           subclass[v, u], {p -> member[set[0], ld[nat[x], nat[y]]],
            u -> ld[nat[x], nat[z]], v -> ld[nat[x], natmul[nat[y], nat[z]]]}] // MapNotNot

Out[76]= or[equal[ld[nat[x], nat[z]], ld[nat[x], natmul[nat[y], nat[z]]]],
           not[member[set[0], ld[nat[x], nat[y]]]]] == True

In[77]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[78]:= Map[or[#, member[z, omega]] &, SubstTest[implies, and[equal[v, V], equal[t, 0]],
             equal[ld[x, v], t], {t → ld[x, z], v → natmul[y, z]}]] // Reverse // MapNotNot

Out[78]= or[equal[ld[x, z], ld[x, natmul[y, z]]], member[z, omega]] == True

In[79]:= or[equal[ld[x_, z_], ld[x_, natmul[y_, z_]]], member[z_, omega]] := True
```

Remove wraps.

```
In[80]:= SubstTest[implies,
            and[equal[x, nat[u]], equal[y, nat[v]], equal[z, nat[w]], member[set[0], ld[x, y]]],
            equal[ld[x, z], ld[x, natmul[y, z]]], {u → x, v → y, w → z}] // MapNotNot // Reverse

Out[80]= or[equal[ld[x, z], ld[x, natmul[y, z]]],
           not[member[z, omega]], not[member[set[0], ld[x, y]]]] == True

In[81]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Simplified corollary of Quaife's Theorem **(LD19)**.

```
In[82]:= SubstTest[and, or[p, q], implies[p, q], {p -> member[z, omega],
           q -> or[equal[ld[x, z], ld[x, natmul[y, z]]], not[member[set[0], ld[x, y]]]]}]

Out[82]= or[equal[ld[x, z], ld[x, natmul[y, z]]], not[member[set[0], ld[x, y]]]] == True

In[83]:= or[equal[ld[x_, z_], ld[x_, natmul[y_, z_]]], not[member[set[0], ld[x_, y_]]]] := True
```

Wrapper-free statement of Quaife's **(LD19)**.

```
In[84]:= Map[not, SubstTest[and, implies[p1, p3],
            not[implies[and[p1, p2], p4]], {p1 -> member[set[0], ld[x, y]],
             p2 -> member[u, ld[x, z]], p3 → equal[ld[x, z], ld[x, natmul[y, z]]],
             p4 -> member[u, ld[x, natmul[y, z]]]}]] // Reverse

Out[84]= or[member[u, ld[x, natmul[y, z]]],
           not[member[u, ld[x, z]]], not[member[set[0], ld[x, y]]]] == True

In[85]:= or[member[u_, ld[x_, natmul[y_, z_]]],
           not[member[u_, ld[x_, z_]]], not[member[set[0], ld[x_, y_]]]] := True
```