Quaife's Theorem (Q10)

Johan G. F. Belinfante and Claudia D. Huang
2005 July 20

```
In[1]:= SetDirectory["i:"]; << goedel71.20a; << tools.m

        :Package Title: goedel71.20a          2005 July 20 at 9:10 a.m.

        It is now:  2005 Jul 20 at 15:3

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 July 18

        weightlimit = 40
```

---

summary

Quaife's Theorem **(Q10)** and its corollary are derived in this notebook. The derivations are simplified by using **nat** wrappers, but most of the final results are wrapper-free. Removing some variables yield functional statements for both results.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical Theories,
           Kluwer Academic Publishers, Dordrecht, the Netherlands, 1992.";
```

---

derivation of (Q10)

Lemma.

```
In[3]:= SubstTest[implies, and[member[pair[w, u], DIV], member[pair[w, v], DIV]],
         member[pair[w, natadd[u, v]], DIV],
         {u → natsub[nat[x], natmod[nat[x], nat[y]]],
          v → natsub[nat[z], natmod[nat[z], nat[y]]], w → nat[y]}]

Out[3]= member[pair[nat[y], natsub[natadd[nat[x], nat[z]],
            natadd[natmod[nat[x], nat[y]], natmod[nat[z], nat[y]]]]], DIV] == True

In[4]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

```
In[5]:= SubstTest[implies, member[pair[w, natsub[u, v]], DIV],
         equal[natmod[u, w], natmod[v, w]], {u → natadd[nat[x], nat[y]],
          v -> natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], w -> nat[z]}]

Out[5]= equal[natmod[natadd[nat[x], nat[y]], nat[z]], natmod[
          natadd[natmod[nat[x], nat[z]], natmod[nat[y], nat[z]]], nat[z]]] == True

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

The **nat** wrappers can be replaced with numberhood literals.

```
In[7]:= SubstTest[implies, and[equal[u, nat[x]], equal[v, nat[y]], equal[w, nat[z]]],
         equal[natmod[natadd[u, v], w], natmod[natadd[natmod[u, w], natmod[v, w]], w]],
         {u → x, v → y, w → z}]

Out[7]= or[
          equal[natmod[natadd[x, y], z], natmod[natadd[natmod[x, z], natmod[y, z]], z]],
          not[member[x, omega]], not[member[y, omega]], not[member[z, omega]]] == True

In[8]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

When any variable is not a number, the same equation remains valid because both sides are then equal to **V**.

```
In[9]:= SubstTest[implies, and[equal[u, v], equal[v, w]],
         equal[u, w], {u -> natmod[natadd[x, y], z], v → V,
          w -> natmod[natadd[natmod[x, z], natmod[y, z]], z]}]

Out[9]= or[and[member[x, omega], member[y, omega], member[z, omega]],
          equal[natmod[natadd[x, y], z],
           natmod[natadd[natmod[x, z], natmod[y, z]], z]]] == True

In[10]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Combining these results yields a wrapper-free equation that can be made into a temporary rewrite rule. This is Quaife's Theorem **(Q10)**.

```
In[11]:= SubstTest[and, implies[p1, p2], or[p1, p2],
          {p1 -> and[member[x, omega], member[y, omega], member[z, omega]], p2 -> equal[
             natmod[natadd[x, y], z], natmod[natadd[natmod[x, z], natmod[y, z]], z]]}]

Out[11]= True ==
          equal[natmod[natadd[x, y], z], natmod[natadd[natmod[x, z], natmod[y, z]], z]]

In[12]:= natmod[natadd[natmod[x_, z_], natmod[y_, z_]], z_] := natmod[natadd[x, y], z]
```

The following corollary subsumes this result, and will be made a permanent rewrite rule.

```
In[13]:= SubstTest[natmod,
           natadd[natmod[x, z], natmod[w, z]], z, w → natmod[y, z]] // Reverse

Out[13]= natmod[natadd[x, natmod[y, z]], z] == natmod[natadd[x, y], z]

In[14]:= natmod[natadd[x_, natmod[y_, z_]], z_] := natmod[natadd[x, y], z]
```

The Orderless attribute of **natadd** makes it unnecessary to add a similar rule with **x** and **y** interchanged.

```
In[15]:= natmod[natadd[natmod[x, z], y], z]

Out[15]= natmod[natadd[x, y], z]
```

---

functional version

Two variables can be removed from the statement of **(Q10)** using **reify**.

```
In[16]:= Map[VERTSECT, SubstTest[reify, w, natmod[
             f[natmod[first[w], x], natmod[second[w], x]], x], f → natadd]] // Reverse

Out[16]= composite[modulo[x], NATADD, cross[modulo[x], modulo[x]]] ==
          composite[modulo[x], NATADD]

In[17]:= composite[modulo[x_], NATADD, cross[modulo[x_], modulo[x_]]] :=
          composite[modulo[x], NATADD]
```

Corollaries:

```
In[18]:= Assoc[composite[modulo[x], NATADD],
           cross[modulo[x], modulo[x]], cross[modulo[x], Id]] // Reverse

Out[18]= composite[modulo[x], NATADD, cross[modulo[x], Id]] ==
          composite[modulo[x], NATADD]

In[19]:= composite[modulo[x_], NATADD, cross[modulo[x_], Id]] :=
          composite[modulo[x], NATADD]

In[20]:= Assoc[composite[modulo[x], NATADD],
           cross[modulo[x], modulo[x]], cross[Id, modulo[x]]] // Reverse

Out[20]= composite[modulo[x], NATADD, cross[Id, modulo[x]]] ==
          composite[modulo[x], NATADD]

In[21]:= composite[modulo[x_], NATADD, cross[Id, modulo[x_]]] :=
          composite[modulo[x], NATADD]
```

Reintroducing a variable yields:

```
In[22]:= Assoc[composite[modulo[x], NATADD], cross[modulo[x], Id], RIGHT[y]]

Out[22]= composite[modulo[x], plus[y], modulo[x]] == composite[modulo[x], plus[y]]

In[23]:= composite[modulo[x_], plus[y_], modulo[x_]] := composite[modulo[x], plus[y]]
```

---

removing variables from Quaife's corollary

Quaife's corollary of his theorem **(Q10)** is already availabe in the **GOEDEL** program. Only one variable needs a **nat** wrapper.

```
In[24]:= natmod[natadd[x, natmul[nat[y], z]], z]

Out[24]= natmod[x, z]
```

Lemma.

```
In[25]:= Assoc[id[x], id[image[V, intersection[omega, set[y]]]], modulo[y]] // Reverse

Out[25]= composite[id[intersection[x, image[V, intersection[omega, set[y]]]]],
           modulo[y]] == composite[id[x], modulo[y]]

In[26]:= composite[id[intersection[x_, image[V, intersection[omega, set[y_]]]]],
           modulo[y_]] := composite[id[x], modulo[y]]
```

A functional version of this result can be derived using **reify**.

```
In[27]:= Map[composite[VERTSECT[#], id[omega]] &, SubstTest[reify,
           w, natmod[natadd[x, f[nat[w], y]], y], f → natmul]] // Reverse

Out[27]= composite[modulo[y], plus[x], times[y]] == cart[omega, set[natmod[x, y]]]

In[28]:= composite[modulo[y_], plus[x_], times[y_]] := cart[omega, set[natmod[x, y]]]
```

Another variable can be removed by applying **reify** a second time.

```
In[29]:= Map[rotate[inverse[#]] &, SubstTest[reify, y,
           composite[modulo[x], f[y], times[x]], f → plus]] // Reverse

Out[29]= composite[modulo[x], NATADD, cross[Id, times[x]]] ==
          composite[modulo[x], FIRST, id[cart[V, omega]]]

In[30]:= composite[modulo[x_], NATADD, cross[Id, times[x_]]] :=
          composite[modulo[x], FIRST, id[cart[V, omega]]]
```

A special case:

```
In[31]:= SubstTest[composite, modulo[nat[x]], plus[y], times[nat[x]], y → 0]

Out[31]= composite[modulo[nat[x]], times[nat[x]]] == cart[omega, set[0]]

In[32]:= composite[modulo[nat[x_]], times[nat[x_]]] := cart[omega, set[0]]
```