a transitivity property for integer equi-ratios

Johan G. F. Belinfante
2012 June 26

Equality of fractions satisfies a transitive property: if \( \frac{a}{b} = \frac{c}{d} \) and \( \frac{c}{d} = \frac{e}{f} \), then \( \frac{a}{b} = \frac{e}{f} \). The two equations in the hypothesis imply that \( ad = bc \) and \( cf = de \), from which one can deduce \( adf = bcf = bde \). If \( d \) is nonzero, one can cancel out the middle factor \( d \) to obtain \( af = be \). In this notebook a wrapper-free statement involving six integer variables is derived that amounts to this transitivity statement about equal integer ratios.

wrapped version

At first blush it would appear that the complete statement that needs to be derived must have an integer-hood hypothesis for each of the six variables in addition to the three equations about products and the condition that the middle variable be nonzero, for a total of ten literals. The GOEDEL program automatically scans any statement made to see whether any of the many available rewrite rules can be used to simplify it, and this would take a long time for a statement with ten literals. The number of literals can be reduced to four if each of the variables is wrapped with \texttt{int}. It is easy to derive a transitivity statement involving six variables if all of them are wrapped with \texttt{int}. This is done in the present section. One lemma is needed.

Lemma. A cancellation law for products of three integers.
removing four int wrappers

Although it is difficult to remove all the int wrappers, four can be removed. This will be done in two steps.

Theorem. Removing the first pair does not take long.

In[6]:= SubstTest[implies, and[equal[u, int[s]], equal[v, int[t]]],
   or[equal[id[omega], int[x]], equal[intmul[u, int[x]], intmul[v, int[w]]]],
   not[equal[intmul[int[u], int[x]], intmul[int[v], int[w]]]],
   not[member[u, Z]], not[member[v, Z]]] = True


In[7]:= (% /. {u → u_, v → v_, w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed

Theorem. Removing a second pair takes a little longer.
eliminating four literals

Before attempting to remove the remaining two `int` wrappers, it is desirable to reduce the number of literals. The following lemma shows that at least two integer-hood literals are redundant. The lemma implies that if `u` is an integer, then so is `v`, and vice versa. The same goes for `y` and `z`.

Lemma.

A further reduction of integer-hood literals is possible.

Lemma. Either `u` or `y` must be an integer.

Theorem. The literal `member[z, Z]` is redundant.

```
In[8]:= SubstTest[implies, and[equal[y, int[s]], equal[z, int[t]]], or[equal[id[omega], int[x]],
   equal[intmul[u, z], intmul[v, y]], not[equal[intmul[u, int[x]], intmul[v, int[w]]]],
   not[equal[intmul[int[w], z], intmul[int[x], y]]],
   not[member[u, Z]], not[member[v, Z]], {s -> y, t -> z}] // Reverse
Out[8]= or[equal[id[omega], int[x]], equal[intmul[u, z], intmul[v, y]],
   not[equal[intmul[u, int[x]], intmul[v, int[w]]]],
   not[equal[intmul[y, int[x]], intmul[z, int[w]]]], not[member[u, Z]],
   not[member[v, Z]], not[member[y, Z]], not[member[z, Z]]] = True

In[9]:= (% /. {u -> u_, v -> v_, w -> w_, x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed

Out[9]= SubstTest[implies, and[equal[x, s], member[s, Z]],
   member[r, Z], {r -> intmul[u, int[x]], s -> intmul[v, int[w]]}] // Reverse
Out[10]= or[member[u, Z],
   not[equal[intmul[u, int[x]], intmul[v, int[w]]]], not[member[v, Z]]] = True

In[11]:= or[member[u_, Z],
   not[equal[intmul[int[w_], v_], intmul[int[x_], u_]]], not[member[v_, Z]]] := True
```

```
By symmetry, the same goes for the other integer-hood literals. These redundant literals can be removed all at once.

Theorem. Elimination of four integer-hood literals.

By symmetry, the same goes for the other integer-hood literals. These redundant literals can be removed all at once.

The final two int wrappers can now be removed, introducing two new integer-hood literals.

Theorem. A transitivity statement with six literals, and six variables, but no int wrappers.