EQUIDIFF and RIF

Johan G. F. Belinfante
2002 December 19

 improt goedel52.q69; improt tools.m

: Package Title: goedel52.q69 2002 December 18 at 1:30 p.m

It is now: 2002 Dec 19 at 6:56

Loading Simplification Rules

TOOLS.M Revised 2002 November 30

weightlimit = 40

summary

This notebook contains a formula that is expected to be useful in connection with addition of integers. All integers can be written as composites of negative and positive integers:

member[ composite[ inverse[ plus[ x ]], plus[ y ]], Z ]
and[ member[ x, omega], member[ y, omega] ]

These factors do not commute, but if the order is reversed, one obtains a subset of the same integer. The integer itself can be recovered from this reversed product by taking the image with EQUIDIFF. This completion process whereby a subset of an integer is completed to the whole integer will be needed to define addition of integers: the sum of two integers is the completion of their composite.

derivation details

The first step is to show that reversing the product yields a subset of the original integer.

dif[ composite[ plus[ x ], inverse[ plus[ y ]]],
   composite[ inverse[ plus[ y ]], plus[ x ]]] // VSNormality
intersection[ composite[ NATADD, RIGHT[ x ], inverse[ RIGHT[ y ]], inverse[ NATADD]],
   composite[ inverse[ RIGHT[ y ]], complement[ inverse[ NATADD]], NATADD, RIGHT[ x ]]] == 0

intersection[ composite[ NATADD, RIGHT[ x ], inverse[ RIGHT[ y ]], inverse[ NATADD]],
   composite[ inverse[ RIGHT[ y ]], complement[ inverse[ NATADD]], NATADD, RIGHT[ x ]]] := 0

SubstTest[ equal, 0, dif[ u, v], { u -> composite[ plus[ x ], inverse[ plus[ y ]]],
   v -> composite[ inverse[ plus[ y ]], plus[ x ]]]] // Reverse

subclass[ composite[ NATADD, RIGHT[ x ], inverse[ RIGHT[ y ]], inverse[ NATADD]],
   composite[ inverse[ RIGHT[ y ]], inverse[ NATADD], NATADD, RIGHT[ x ]]] == True

subclass[ composite[ NATADD, RIGHT[ x_ ], inverse[ RIGHT[ y_ ]], inverse[ NATADD]],
   composite[ inverse[ RIGHT[ y_ ]], inverse[ NATADD], NATADD, RIGHT[ x_ ]]] := True
Any integer is its own completion:

```
ImageComp[EQUIDIFF, EQUIDIFF, singleton[PAIR[x, y]]] // Reverse
```

Thus:
```
image[EQUIDIFF, composite[inverse[plus[x]], plus[y]]] ==
composite[inverse[plus[x]], plus[y]]
```

True

To show that the completion of a sub–integer is the integer, we need to establish inclusions in two directions. In one direction the inclusion used is the one derived in the preceding section:

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
{u -> composite[plus[x], inverse[plus[y]]],
v -> composite[inverse[plus[y]], plus[x]], w -> EQUIDIFF}]
```

That is, the completion of the reversed product is contained in the original product:
```
subclass[image[EQUIDIFF, composite[plus[x], inverse[plus[y]]]],
composite[inverse[plus[y]], plus[x]]]
```

True

In the other direction one can use the fact that the point \texttt{PAIR[x,y]} lies on the sub–integer \texttt{composite[plus[y],inverse[plus[x]]]}. That is true because substracting \texttt{x} from itself and then adding \texttt{y} yields \texttt{y}.

```
SubstTest[member, y, image[z, singleton[x]],
  z -> composite[plus[y], inverse[plus[x]]]] // Reverse
```

```
member[pair[x, y], composite[NATADD, RIGHT[y], inverse[RIGHT[x]], inverse[NATADD]]] ==
and[member[x, omega], member[y, omega]]
```

```
member[pair[x_, y_],
  composite[NATADD, RIGHT[y_], inverse[RIGHT[x_]], inverse[NATADD]]] :=
and[member[x, omega], member[y, omega]]
```
This fact yields the desired inclusion:

\[
\begin{equation}
\text{SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],}
\{u \rightarrow \text{singleton[PAIR[y, x]]},
\ v \rightarrow \text{composite[plus[x], inverse[plus[y]]]}, w \rightarrow \text{EQUIDIFF}\}
\end{equation}
\]

or[and[member[x, V], member[y, V], not[member[x, omega]]],
 and[member[x, V], member[y, V], not[member[y, omega]]],
 subclass[composite[inverse[RIGHT[y]], inverse[NATADD], NATADD, RIGHT[x]],
 composite[SECOND, intersection[composite[inverse[NATADD], NATADD],
 composite[inverse[FIRST], NATADD, RIGHT[y], inverse[RIGHT[x]],
 inverse[NATADD], FIRST]], inverse[SECOND]]] == True

The above formula can be cleaned up a bit:

\[
\begin{equation}
\text{Map[or[not[member[x, omega]], not[member[y, omega]]], \#] \&, \%}
\end{equation}
\]

or[not[member[x, omega]], not[member[y, omega]],
 subclass[composite[inverse[RIGHT[y]], inverse[NATADD], NATADD, RIGHT[x]],
 composite[SECOND, intersection[composite[inverse[NATADD], NATADD],
 composite[inverse[FIRST], NATADD, RIGHT[y], inverse[RIGHT[x]],
 inverse[NATADD], FIRST]], inverse[SECOND]]] == True

or[not[member[x_, omega]], not[member[y_, omega]],
 subclass[composite[inverse[RIGHT[y_]], inverse[NATADD], NATADD, RIGHT[x_]],
 composite[SECOND, intersection[composite[inverse[NATADD], NATADD],
 composite[inverse[FIRST], NATADD, RIGHT[y_], inverse[RIGHT[x_]],
 inverse[NATADD], FIRST]], inverse[SECOND]]] := True

The two inclusions can now be combined to obtain a conditional equation:

\[
\begin{equation}
\text{SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],}
\{u \rightarrow \text{composite[inverse[plus[y]], plus[x]]},
\ v \rightarrow \text{image[EQUIDIFF, composite[plus[x], inverse[plus[y]]]]}\} // \text{Reverse}
\end{equation}
\]

or[equal[composite[SECOND],
 intersection[composite[inverse[NATADD], NATADD], composite[inverse[FIRST], NATADD,
 RIGHT[y], inverse[RIGHT[x]], inverse[NATADD], FIRST]], inverse[SECOND]],
 composite[inverse[RIGHT[y]], inverse[NATADD], NATADD, RIGHT[x]]],
 not[member[x, omega]], not[member[y, omega]]] == True

or[equal[composite[SECOND],
 intersection[composite[inverse[NATADD], NATADD], composite[inverse[FIRST], NATADD,
 RIGHT[y_], inverse[RIGHT[x_]], inverse[NATADD], FIRST]], inverse[SECOND]],
 composite[inverse[RIGHT[y_]], inverse[NATADD], NATADD, RIGHT[x_]]],
 not[member[x_, omega]], not[member[y_, omega]]] := True

That is:

\[
\begin{equation}
\text{implies[and[member[x, omega], member[y, omega]],}
\text{equal[\text{image[EQUIDIFF, composite[plus[x], inverse[plus[y]]]]},
\ \text{composite[inverse[plus[y]], plus[x]]}\}]
\end{equation}
\]

True

The next step is to remove the variables from this equation, and also to eliminate the explicit condition that \(x\) and \(y\) be natural numbers.
eliminating the variables

\[ \text{simplify} = \text{False}; \]

The following two functions are not explicitly needed, but they inspired the method that is used to eliminate the variables from the equation in the preceding section.

\[
\lambda \text{pair}[x, y], \text{composite}[\text{inverse}[\text{plus}[x]], \text{plus}[y]]
\]

\[
\text{composite}[\text{VERTSECT}[\text{EQUIDIFF}], \text{id}[	ext{cart}[V, V]]]
\]

\[
\lambda \text{pair}[x, y], \text{composite}[\text{plus}[y], \text{inverse}[\text{plus}[x]]]
\]

\[
\text{composite}[\text{VERTSECT}[\text{composite}[\text{cross}[\text{NATADD}, \text{NATADD}], \text{inverse}[\text{RIF}]]], \text{id}[	ext{cart}[V, V]]]
\]

These formulas suggested writing the conditional equation at the end of the last section as follows:

\[
\text{implies} [\text{and}[\text{member}[x, \text{omega}], \text{member}[y, \text{omega}]], \\
\text{equal}[	ext{image}[	ext{composite}[\text{EQUIDIFF}, \text{cross}[\text{NATADD}, \text{NATADD}], \text{inverse}[\text{RIF}]], \\
\text{singleton}[\text{PAIR}[x, y]]], \text{image}[	ext{EQUIDIFF}, \text{singleton}[\text{PAIR}[x, y]]]]
\]

\[ \text{True} \]

The actual elimination of variables is now done as follows:

\[
\text{SubstTest}[\text{class}, \text{pair}[x, y], \text{or}[\text{not}[	ext{member}[x, \text{omega}]], \text{not}[	ext{member}[y, \text{omega}]]], \\
\text{equal}[	ext{image}[	ext{u}, \text{singleton}[\text{PAIR}[x, y]]], \text{image}[	ext{v}, \text{singleton}[\text{PAIR}[x, y]]]], \\
\{u \to \text{composite}[\text{EQUIDIFF}, \text{cross}[\text{NATADD}, \text{NATADD}], \text{inverse}[\text{RIF}]], \\
v \to \text{EQUIDIFF}]\]
\]

\[
\text{cart}[V, V] = \text{union}[	ext{cart}[V, \text{complement}[\text{omega}]], \text{cart}[\text{complement}[\text{omega}], V], \\
\text{composite}[\text{id}, \text{intersection}][\text{complement}][\text{fix}][\text{composite}[\text{EQUIDIFF}, \\
\text{complement}[\text{composite}[\text{EQUIDIFF}, \text{cross}[\text{NATADD}, \text{NATADD}], \text{inverse}[\text{RIF}]]]]], \\
\text{complement}[	ext{fix}][\text{composite}[\text{RIF}, \text{cross}[\text{inverse}[\text{NATADD}], \text{inverse}[\text{NATADD}]], \\
\text{complement}[\text{EQUIDIFF}]]]]]
\]

This formula can be cleaned up. The first step is to take the complement:

\[
\text{Map}[	ext{composite}[\text{id}, \text{complement}][\#] &\& \%, \%]
\]

\[
0 = \text{union}[	ext{composite}[	ext{id}][\text{omega}], \text{fix}[	ext{composite}[\text{EQUIDIFF}, \\
\text{complement}][\text{composite}[\text{EQUIDIFF}, \text{cross}[\text{NATADD}, \text{NATADD}], \text{inverse}[\text{RIF}]]]]], \text{id}[	ext{omega}]], \\
\text{composite}[	ext{id}][\text{omega}], \text{fix}[	ext{composite}[\text{RIF}, \text{cross}[\text{inverse}[\text{NATADD}], \text{inverse}[\text{NATADD}]], \\
\text{complement}[\text{EQUIDIFF}]]], \text{id}[	ext{omega}]]]
\]

more cleanup activities

The immediate aim is to eliminate all explicit mention of the set \text{omega} of natural numbers.

\[ \text{simplify} = \text{True}; \]

Two formulas will be needed. The easier of the two is this one:
SubstTest[fix, composite[id[cart[w, w]], EQUIDIFF, x]], w -> omega] // Reverse

composite[id[omega], fix[composite[EQUIDIFF, x]], id[omega]] ==
fix[composite[EQUIDIFF, x]]

composite[id[omega], fix[composite[EQUIDIFF, x]], id[omega]] :=
fix[composite[EQUIDIFF, x]]

A similar result holds for composite[RIF,cross[inverse[NATADD],inverse[NATADD]]].

reverse[composite[RIF, cross[inverse[NATADD], inverse[NATADD]]]] // range
cart[omega, omega]

To exploit this observation, an extra step is required. The GOEDEL program contains the following general formula:

composite[id[x], RIF]

composite[RIF, id[composite[SWAP, cross[Id, x]]]]

This is correct, but it produces a mess in the following special case:

composite[id[cart[x, y]], RIF]

composite[RIF, id[composite[id[cart[y, V]], inverse[SECOND], FIRST, id[cart[V, x]]]]]

This is correct, but can be simplified as follows:

Assoc[RIF, id[composite[inverse[SECOND], FIRST]], id[cart[x, y]]]

composite[RIF, id[composite[id[y], inverse[SECOND], FIRST, id[x]]]] ==
composite[RIF, id[cart[x, y]]]

composite[RIF, id[composite[id[y_], inverse[SECOND], FIRST, id[x_]]]] :=
composite[RIF, id[cart[x, y]]]

One now proceeds as before:

SubstTest[fix, composite[id[cart[w, w]], y],
{w -> omega, y -> composite[RIF, cross[inverse[NATADD], inverse[NATADD]], x]}] // Reverse

composite[id[omega], fix[composite[RIF, cross[inverse[NATADD], inverse[NATADD]], x]],
id[omega]] == fix[composite[RIF, cross[inverse[NATADD], inverse[NATADD]], x]]

composite[id[omega], fix[composite[RIF, cross[inverse[NATADD], inverse[NATADD]], x]],
id[omega]] := fix[composite[RIF, cross[inverse[NATADD], inverse[NATADD]], x]]

■ continuation

The result obtained in the penultimate section now simplifies:
This is actually two equations. In general:

```plaintext
equal[0, union[x, y]]
and[equal[0, x], equal[0, y]]
```

The two equations are made into separate rules:

```plaintext
fix[composite[EQUIDIFF,
    complement[composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]]]], id[omega]] := 0
fix[composite[RIF, cross[inverse[NATADD], inverse[NATADD]], complement[EQUIDIFF]]] := 0
```

One little extra step is still needed:

```plaintext
domain[symdif[EQUIDIFF, composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]]]] //
InvertFixTest
fix[composite[complement[EQUIDIFF], cross[NATADD, NATADD], inverse[RIF]]] == 0
fix[composite[complement[EQUIDIFF], cross[NATADD, NATADD], inverse[RIF]]] := 0
```

### Final steps

From the empty domain we derive that the relation itself is empty:

```plaintext
SubstTest[composite, w, id[domain[w]],
    w -> symdif[EQUIDIFF, composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]]]]
0 == union[intersection[EQUIDIFF, complement[composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]]]], intersection[
    complement[EQUIDIFF], composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]]]]
```

This can be rewritten as an equation.

```plaintext
Map[equal[0, #] &, %]
True == equal[EQUIDIFF, composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]]]
```

This is the final result:

```plaintext
composite[EQUIDIFF, cross[NATADD, NATADD], inverse[RIF]] := EQUIDIFF
```