successor invariance of integers

Johan G. F. Belinfante
2002 December 19

```
<< goedel52.q71; << tools.m

:Package Title: goedel52.q71          2002 December 19 at 10:55 a.m.

It is now:  2002 Dec 19 at 16:39

Loading Simplification Rules

TOOLS.M                       Revised 2002 November 30

weightlimit = 40
```

■ summary

Every integer is a one−to−onefunction, whose graph can be visualized as a line with slope 1 in the **cart[omega,omega]** plane. The set **Z** of all the integers is thus a set of parallel lines, one line going through each point of the plane. The integer zero is the graph of the identity function **id[omega]**, namely the line **y = x**. The positive integers are those lines that intercept the **y** axis, and the negative ones are those that intercept the **x** axis. The negative of any integer is the inverse function, whose graph is the line obtained by reflection with respect to the line **y = x**.

Note that if **pair[x,y]** is some point on one of these lines, one gets another point on the same line by incrementing both **x** and **y** by one. In this notebook a simple formula is derived which expresses this successor−invarianceproperty of each integer.

■ derivation

To get started, the **Assoc** test is used to derive the successor−invarianceof a positive integer **plus[x]**.

```
Assoc[plus[x], SUCC, inverse[SUCC]] // Reverse

composite[SUCC, NATADD, RIGHT[x], inverse[SUCC]] ==
 composite[NATADD, RIGHT[x], id[range[SUCC]]]

composite[SUCC, NATADD, RIGHT[x_], inverse[SUCC]] :=
 composite[NATADD, RIGHT[x], id[range[SUCC]]]
```

This fact can be rewritten as follows:

```
member[plus[x], invar[cross[SUCC, SUCC]]]

True
```

The variable **x** can now be eliminated:

```
Map[equal[V, #] &, SubstTest[class, y, member[composite[n, RIGHT[y]], z],
   {n -> NATADD, z -> invar[cross[SUCC, SUCC]]}]] // Reverse
```

```
subclass[omega, image[inverse[PLUS], invar[cross[SUCC, SUCC]]]] == True
```

```
subclass[omega, image[inverse[PLUS], invar[cross[SUCC, SUCC]]]] := True
```

From this one derives:

```
ImageComp[PLUS, inverse[PLUS], invar[cross[SUCC, SUCC]]]
```

```
intersection[invar[cross[SUCC, SUCC]], range[PLUS]] == range[PLUS]
```

How this came about is actually a bit mysterious. A special conditional rewrite rule kicked in:

```
image[PLUS, image[inverse[PLUS], invar[cross[SUCC, SUCC]]]] // Trace // NoFail
```

```
{{image[inverse[PLUS], invar[cross[SUCC, SUCC]]],
  {}, image[inverse[PLUS], invar[cross[SUCC, SUCC]]]},
 image[PLUS, image[inverse[PLUS], invar[cross[SUCC, SUCC]]]], {{{domain[PLUS], omega},
   subclass[omega, image[inverse[PLUS], invar[cross[SUCC, SUCC]]]], True},
  RuleCondition[$ConditionHold[$ConditionHold[range[PLUS]]], True],
  $ConditionHold[$ConditionHold[range[PLUS]]]}, range[PLUS]}
```

```
intersection[invar[cross[SUCC, SUCC]], range[PLUS]] := range[PLUS]
```

This equation can also be restated as an inclusion:

```
SubstTest[equal, intersection[u, v], v,
  {u -> invar[cross[SUCC, SUCC]], v -> range[PLUS]}] // Reverse
```

```
subclass[range[PLUS], invar[cross[SUCC, SUCC]]] == True
```

```
subclass[range[PLUS], invar[cross[SUCC, SUCC]]] := True
```

■ including the negative integers

First a general result will be derived.

```
image[INVERSE, invar[cross[x, y]]] // Normality
```

```
image[INVERSE, invar[cross[x, y]]] == intersection[invar[cross[y, x]],
  P[union[cart[V, intersection[complement[domain[x]], domain[VERTSECT[x]]]],
    cart[complement[domain[y]], complement[domain[x]]],
    cart[domain[VERTSECT[y]], domain[VERTSECT[x]]],
    cart[intersection[complement[domain[y]], domain[VERTSECT[y]]], V]]]]
```

This complicated formula is not needed, but only a simple corollary of it:

```
Map[subclass[#, invar[cross[y, x]]] &, %]
```

```
subclass[image[INVERSE, invar[cross[x, y]]], invar[cross[y, x]]] == True
```

In the appenndix, a stronger result is derived.

```
subclass[image[INVERSE, invar[cross[x_, y_]]], invar[cross[y_, x_]]] := True
```

The negative of an integer **z** is **inverse[z]**. So one can derive the successor−invarianceof the negative integers from that of the positive ones by taking inverses.

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
    {u -> range[PLUS], v -> invar[cross[SUCC, SUCC]], w -> INVERSE}]
```

```
subclass[image[INVERSE, range[PLUS]], image[INVERSE, invar[cross[SUCC, SUCC]]]] == True
```

```
subclass[image[INVERSE, range[PLUS]], image[INVERSE, invar[cross[SUCC, SUCC]]]] := True
```

The transitive property of inclusion yields:

```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
    {u -> image[INVERSE, range[PLUS]], v -> image[INVERSE, invar[cross[SUCC, SUCC]]],
        w -> invar[cross[SUCC, SUCC]]}]
```

```
subclass[image[INVERSE, range[PLUS]], invar[cross[SUCC, SUCC]]] == True
```

```
subclass[image[INVERSE, range[PLUS]], invar[cross[SUCC, SUCC]]] := True
```

The final step is to use the fact that the set **Z** of all integers is the union of the set **range[PLUS]** of non−negativeintegers and the set of non−positiveintegers.

```
SubstTest[subclass, union[u, v], w,
    {u -> range[PLUS], v -> image[INVERSE, range[PLUS]], w -> invar[cross[SUCC, SUCC]]}]
```

```
subclass[Z, invar[cross[SUCC, SUCC]]] == True
```

This is the main result.

```
subclass[Z, invar[cross[SUCC, SUCC]]] := True
```

■ appendix

In this appendix, the general inclusion involving **invar[cross[x,y]]** is strengthened to an equation.

```
SubstTest[subclass, image[w, x], range[w], w -> INVERSE]
```

```
subclass[U[image[INVERSE, x]], cart[V, V]] == True
```

```
subclass[U[image[INVERSE, x_]], cart[V, V]] := True
```

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
    {u -> image[INVERSE, invar[cross[y, x]]], v -> invar[cross[x, y]], w -> INVERSE}]
```

```
subclass[intersection[invar[cross[y, x]], P[cart[V, V]]],
  image[INVERSE, invar[cross[x, y]]]] == True
```

```
subclass[intersection[invar[cross[y_, x_]], P[cart[V, V]]],
  image[INVERSE, invar[cross[x_, y_]]]] := True
```

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> intersection[invar[cross[y, x]], P[cart[V, V]]],
   v -> image[INVERSE, invar[cross[x, y]]]}] // Reverse
```

```
equal[image[INVERSE, invar[cross[x, y]]],
  intersection[invar[cross[y, x]], P[cart[V, V]]]] == True
```

```
image[INVERSE, invar[cross[x_, y_]]] := intersection[invar[cross[y, x]], P[cart[V, V]]]
```