left subtraction

Johan G. F. Belinfante
2002 September 12

```
<< goedel52.p45; << tools.m
```

```
:Package Title: goedel52.p45           2002 September 11 at 6:26 p.m.

It is now:  2002 Sep 12 at 21:15

Loading Simplification Rules

TOOLS.M                    Revised 2002 August 30

weightlimit = 40
```

■ subtraction

Subtraction is obtained from addition by rotation. For example, the fact **3 −1 = 2** is obtained from **1 + 2 = 3** by rotating the three numbers:

```
image[rotate[NATADD],
  cart[singleton[succ[succ[singleton[0]]]], singleton[singleton[0]]]] // Normality
```

```
image[image[inverse[NATADD], singleton[succ[succ[singleton[0]]]]],
  singleton[singleton[0]]] == singleton[succ[singleton[0]]]
```

Since addition is commutative, adding on the left is the same as adding on the right:

```
composite[NATADD, LEFT[x]]
```

```
composite[NATADD, RIGHT[x]]
```

On the other hand, subtraction is not commutative, and so the process of subtracting a fixed number **x** differs from the process of subtracting from a fixed number **x**. Subtracting **x** is just the inverse of adding **x**.

```
composite[rotate[NATADD], RIGHT[x]]
```

```
composite[inverse[RIGHT[x]], inverse[NATADD]]
```

The process of subtracting from **x** is an involution, that is, this process is its own inverse:

```
composite[rotate[NATADD], LEFT[x]]
```

```
image[inverse[NATADD], singleton[x]]
```

```
inverse[image[inverse[NATADD], singleton[x]]]
```

```
image[inverse[NATADD], singleton[x]]
```

Both of these processes are one−to−one functions:

```
ONEONE[composite[rotate[NATADD], RIGHT[x]]]
```

```
True
```

```
ONEONE[composite[rotate[NATADD], LEFT[x]]]
```

```
True
```

The goal in this notebook is to deive a formula for the domain and range of left−subtraction,**composite[rotate[NATADD],-LEFT[x]]**.

■ a formula for the successor of a natural number

We will need as a prerequisite a formula that says that the successor of a natural number **x** is precisely the set of all natural numbers contained in **x**. To derive this result, we begin with this observation:

```
equal[0, composite[id[omega],
  intersection[composite[complement[inverse[E]], SUCC], inverse[S]], id[omega]]]
```

```
True
```

While the **GOEDEL** program recognizes the truth of this assertion, it lacks the corresponding rewrite rule, which we now add on a temporary basis:

```
composite[id[omega],
  intersection[composite[complement[inverse[E]], SUCC], inverse[S]], id[omega]] := 0
```

When the **ImageComp** test is performed using this fact, one encounters the following expression which we simplify using **Renormality**:

```
fix[composite[complement[inverse[E]], SUCC,
   id[intersection[omega, singleton[x]]], S]] // Renormality
```

```
fix[
  composite[complement[inverse[E]], SUCC, id[intersection[omega, singleton[x]]], S]] ==
 intersection[complement[x], complement[singleton[x]],
  image[V, intersection[omega, singleton[x]]], P[x]]
```

```
fix[composite[complement[inverse[E]], SUCC,
   id[intersection[omega, singleton[x_]]], S]] := intersection[complement[x],
  complement[singleton[x]], image[V, intersection[omega, singleton[x]]], P[x]]
```

Now the **ImageComp** test is performed, yielding almost what we want:

```
Map[equal[0, #] &, ImageComp[id[omega],
   composite[intersection[composite[complement[inverse[E]], SUCC], inverse[S]],
    id[omega]], singleton[x]]] // Reverse
```

```
or[not[member[x, omega]], subclass[intersection[omega, P[x]], succ[x]]] == True
```

```
or[not[member[x_, omega]], subclass[intersection[omega, P[x_]], succ[x_]]] := True
```

We can derive a stronger result, replacing the inclusion with equality. This is obtained by combining **AssertTest** with double negation.

```
Map[not, Map[not[implies[member[x, omega], #]] &,
  equal[intersection[omega, P[x]], succ[x]] // AssertTest]]
```

```
or[equal[intersection[omega, P[x]], succ[x]], not[member[x, omega]]] == True
```

```
or[equal[intersection[omega, P[x_]], succ[x_]], not[member[x_, omega]]] := True
```

The following corollary is noted:

```
equal[intersection[omega, image[V, intersection[omega, singleton[x]]], P[x]],
 intersection[image[V, intersection[omega, singleton[x]]], succ[x]]]
```

```
True
```

This fact justifies adding the following new rewrite rule:

```
intersection[omega, image[V, intersection[omega, singleton[x_]]], P[x_]] :=
 intersection[image[V, intersection[omega, singleton[x]]], succ[x]]
```

■ application to left−subtraction

The rewrite rule derived in the preceding section is used here to get a simple formula for the range of left−subtraction function:

```
SubstTest[image, rotate[w], cart[V, singleton[x]], w -> rotate[NATADD]]
```

```
range[image[inverse[NATADD], singleton[x]]] ==
 intersection[image[V, intersection[omega, singleton[x]]], succ[x]]
```

```
range[image[inverse[NATADD], singleton[x_]]] :=
 intersection[image[V, intersection[omega, singleton[x]]], succ[x]]
```

The domain is the same:

```
SubstTest[image, inverse[w], V, w -> image[inverse[NATADD], singleton[x]]] // Reverse
```

```
domain[image[inverse[NATADD], singleton[x]]] ==
 intersection[image[V, intersection[omega, singleton[x]]], succ[x]]
```

```
domain[image[inverse[NATADD], singleton[x_]]] :=
 intersection[image[V, intersection[omega, singleton[x]]], succ[x]]
```

More generally, one has the following, but it is unclear how to orient this more general formula.

```
SubstTest[image, inverse[w], V, w -> image[inverse[NATADD], x]] // Reverse
```

```
domain[image[inverse[NATADD], x]] == range[image[inverse[NATADD], x]]
```

The following involution property does not require adding any new rules:

```
composite[image[inverse[NATADD], singleton[x]], image[inverse[NATADD], singleton[x]]]
```

```
id[intersection[image[V, intersection[omega, singleton[x]]], succ[x]]]
```

■ serendipity: variable−freeformulation of the involution property

The following formula was discovered accidentally:

```
Assoc[DUP, inverse[DUP], union[composite[DUP, id[omega]],
   composite[inverse[E], IMAGE[DUP], id[omega]]]] // Reverse
```

```
union[composite[DUP, id[omega]], composite[inverse[E], IMAGE[DUP], id[omega]]] ==
 composite[DUP, id[omega], inverse[S], id[omega]]
```

```
union[composite[DUP, id[omega]], composite[inverse[E], IMAGE[DUP], id[omega]]] :=
 composite[DUP, id[omega], inverse[S], id[omega]]
```

The following variable−freeformulation of the involution property follows as a corollary of this discovery:

```
Map[inverse, composite[RIF, cross[inverse[NATADD], inverse[NATADD]], DUP] // VSNormality]
```

```
composite[intersection[composite[NATADD, FIRST], composite[NATADD, SECOND]],
  inverse[RIF]] == composite[id[omega], S, id[omega], inverse[DUP]]
```

```
composite[intersection[composite[NATADD, FIRST], composite[NATADD, SECOND]],
  inverse[RIF]] := composite[id[omega], S, id[omega], inverse[DUP]]
```