# order and addition for natural numbers

*Johan G. F. Belinfante*
*2002 June 16*

```
<< goedel52.o56; << tools.m
```

```
:Package Title: goedel52.o56          2002 June 15 at 6:30 p.m.
```

```
It is now:  2002 Jun 15 at 23:37
```

```
Loading Simplification Rules
```

```
TOOLS.M                   Revised 2002 June 12
```

```
weightlimit = 40
```

## ■ Introduction

Natural numbers satisfy **a ≤ b** if and only if there is a number **c** such that **a + c = b**. In this notebook we derive a formula that expresses this idea without variables. The restriction of the subset relation **S** to **omega** is the less−than−or−equal relation. The addition statement is represented as the relation

```
class[pair[x, y], exists[z, member[pair[pair[x, z], y], n]]] /. n -> NATADD
```

```
composite[NATADD, inverse[FIRST]]
```

## ■ The derivation

We introduce the abbreviation:

```
one := singleton[0]
```

This simplification rule is needed:

```
Assoc[NATADD, id[cart[omega, omega]], id[cart[V, omega]]] // Reverse
```

```
composite[NATADD, id[cart[V, omega]]] == NATADD
```

```
composite[NATADD, id[cart[V, omega]]] := NATADD
```

The following rule relates transitive closure to iteration

```
SubstTest[image, image[power[x], y], z, y -> complement[one]]
```

```
image[trv[x], z] == image[iterate[x, z], complement[singleton[0]]]
```

```
image[trv[x_], z_] := image[iterate[x, z], complement[singleton[0]]]
```

The following fact about natural numbers is needed.

```
equal[intersection[omega, complement[P[complement[singleton[0]]]]],
  intersection[omega, complement[singleton[0]]]] // AssertTest

equal[intersection[omega, complement[P[complement[singleton[0]]]]],
  intersection[omega, complement[singleton[0]]]] == True

intersection[omega, complement[P[complement[singleton[0]]]]] :=
 intersection[omega, complement[singleton[0]]]
```

The uniqueness theorem for iteration is our main tool.

```
SubstTest[implies, and[equal[composite[u, w], composite[w, SUCC]],
        equal[image[w, singleton[0]], v]],
    equal[composite[w, id[omega]], iterate[u, v]],
    {u -> SUCC, v -> dif[omega, one], w -> composite[id[omega], E]}]

equal[composite[id[omega], E],
  composite[id[omega], iterate[SUCC, complement[singleton[0]]]]] == True

composite[id[omega], iterate[SUCC, complement[singleton[0]]]] := composite[id[omega], E]
```

The following equation is a consequence of commutativity of addition.

```
ImageComp[NATADD, SWAP, cart[x, complement[one]]] // Reverse

intersection[omega, image[iterate[SUCC, x], complement[singleton[0]]]] ==
 intersection[omega, image[iterate[SUCC, complement[singleton[0]]], x]]
```

The following rewrite rule is potentially dangerous; it might loop, but we only use it briefly.

```
intersection[omega, image[iterate[SUCC, x_], complement[singleton[0]]]] :=
 intersection[omega, image[iterate[SUCC, complement[singleton[0]]], x]]
```

The following is an **image** counterpart of a **composite** equation:

```
ImageComp[id[omega], iterate[SUCC, complement[one]], x] // Reverse

intersection[omega, image[iterate[SUCC, complement[singleton[0]]], x]] ==
 intersection[omega, complement[P[complement[x]]]]

intersection[omega, image[iterate[SUCC, complement[singleton[0]]], x_]] :=
 intersection[omega, complement[P[complement[x]]]]
```

Now we go back to composites via **VSNormality**.

```
composite[id[omega], trv[SUCC]] // VSNormality

composite[id[omega], trv[SUCC]] == composite[id[omega], E]

composite[id[omega], trv[SUCC]] := composite[id[omega], E]
```

The promised equation now emerges.

```
ImageComp[id[cart[V, omega]], power[SUCC], omega]

composite[NATADD, inverse[FIRST]] == composite[id[omega], S, id[omega]]
```