# recursion relation for natural multiplication

*Johan G. F. Belinfante*
*2002 Ausgust 23*

```
<< goedel52.p09; << tools.m

:Package Title: goedel52.p09         2002 August 21 at 10:30 p.m.

It is now:  2002 Aug 23 at 17:42

Loading Simplification Rules

TOOLS.M                   Revised 2002 August 22

weightlimit = 40
```

## ■ Summary

This notebook contains a derivation of a basic recursion relation for natural multiplication.

## ■ iterate uniqueness argument

The starting point uses the uniqueness theorem for **iterate**:

```
SubstTest[implies, and[equal[composite[w, SUCC], composite[z, w]],
        equal[image[w, singleton[0]], y]],
    equal[composite[w, id[omega]], iterate[z, y]],
    {z -> SUCC, y -> intersection[omega, singleton[x]],
        w -> composite[
    id[image[V, intersection[omega, singleton[x]]]], iterate[SUCC, singleton[x]]]}]

equal[composite[NATADD, RIGHT[x]], composite[
   id[image[V, intersection[omega, singleton[x]]]], iterate[SUCC, singleton[x]]] == True

composite[id[image[V, intersection[omega, singleton[x_]]]],
  iterate[SUCC, singleton[x_]]] := composite[NATADD, RIGHT[x]]
```

## ■ Simplification rule

The following simplification rule is needed in the final step below:

```
SubstTest[union,
 complement[image[V, intersection[omega, singleton[z]]]], natmul[x, z], z → succ[y]]

union[complement[image[V, intersection[omega, singleton[y]]]], natmul[x, succ[y]]] ==
 natmul[x, succ[y]]

union[complement[image[V, intersection[omega, singleton[y_]]]], natmul[x_, succ[y_]]] :=
 natmul[x, succ[y]]
```

## ■ ImageComp arguments

The **ImageComp** tool is used twice. This is the first application:

```
ImageComp[NATMUL, composite[LEFT[x], SUCC], singleton[y]]

image[NATADD,
  cart[image[iterate[iterate[SUCC, singleton[x]], singleton[0]], singleton[y]],
   singleton[x]]] == singleton[natmul[x, succ[y]]]
```

This is added as a temporary rule:

```
image[NATADD,
  cart[image[iterate[iterate[SUCC, singleton[x_]], singleton[0]], singleton[y_]],
   singleton[x_]]] := singleton[natmul[x, succ[y]]]
```

The second application of **ImageComp** yields the recursion relation:

```
Map[A, ImageComp[composite[NATADD, RIGHT[x], NATMUL], LEFT[x], singleton[y]]]

natmul[x, succ[y]] == natadd[x, natmul[x, y]]
```

This can be added as a permanent rewrite rule:

```
natmul[x_, succ[y_]] := natadd[x, natmul[x, y]]
```

This remarkable result does not require any hypothesis about **x** and **y** being natural numbers. If either is not, then the equation reduces to **V = V**.

## ■ Corollary

The result obtained above can be reformulated without variables, but it appears doubtful that such a reformulation is actually useful.

```
symdif[composite[NATMUL, cross[Id, SUCC]],
  composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]]] // VSNormality

union[intersection[composite[NATMUL, cross[Id, SUCC]],
   composite[complement[NATADD], cross[Id, NATMUL], ASSOC, cross[DUP, Id]]],
  intersection[composite[complement[NATMUL], cross[Id, SUCC]],
   composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]]]] == 0

union[intersection[composite[NATMUL, cross[Id, SUCC]],
   composite[complement[NATADD], cross[Id, NATMUL], ASSOC, cross[DUP, Id]]],
  intersection[composite[complement[NATMUL], cross[Id, SUCC]],
   composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]]]] := 0

SubstTest[equal, 0, symdif[u, v],
    {u -> composite[NATMUL, cross[Id, SUCC]],
  v -> composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]]}]

True == equal[composite[NATMUL, cross[Id, SUCC]],
  composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]]]
```

This can also be rewritten another way which appears to be even less attractive:

```
composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]] // TripleRotate
```

```
composite[NATADD, cross[Id, NATMUL], ASSOC, cross[DUP, Id]] == composite[NATADD,
  cross[NATMUL, Id], inverse[ASSOC], id[inverse[SECOND]], cross[Id, inverse[FIRST]]]
```