# domain[NATMUL]

*Johan G. F. Belinfante*
*2002 July 1*

```
<< goedel52.o86; << tools.m
```

```
:Package Title: goedel52.o86          2002 July 1 at 12:45 noon

It is now:  2002 Jul 1 at 13:56

Loading Simplification Rules

TOOLS.M                   Revised 2002 June 12

weightlimit = 40
```

## ■ Introduction

This notebook establishes the fact that **domain[NATMUL]** is **cart[omega,omega]**. The main ingredient is a generalization of a theorem about the domain of **iterate[x,y]** that was proved earlier today in the notebook **DOIT−IVR.NB** Actually, two theorems were proved in that notebook, the more memorable one being the statement that if the starting generation is a nonempty invariant subclass **y** of the domain of **x**, then the iteration process never ends. That particular theorem does not yield anything useful in the present context. It is the other less memorable theorem that is needed here:

```
implies[subclass[range[iterate[u, v]], domain[u]],
    or[equal[0, v], equal[omega, domain[iterate[u, v]]]]]
```

```
True
```

The other ingredient of our proof is this formula:

```
composite[NATMUL, LEFT[x]]
```

```
composite[id[image[V, intersection[omega, singleton[x]]]],
 iterate[iterate[SUCC, singleton[x]], singleton[0]]]
```

Our strategy will be to prove that the following is true, and then to use it to deduce the domain of **NATMUL**.

```
implies[member[x, omega], equal[omega, domain[composite[NATMUL, LEFT[x]]]]]
```

```
or[equal[omega, domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]]],
 not[member[x, omega]]]
```

The following partial information about **domain[NATMUL]** is known:

```
composite[NATMUL, id[cart[omega, omega]]]
```

```
NATMUL
```

From this one deduces:

```
SubstTest[equal, composite[x, id[y]], x, {x -> NATMUL, y -> cart[omega, omega]}] // Reverse
```

```
and[subclass[domain[domain[NATMUL]], omega],
  subclass[range[domain[NATMUL]], omega]] == True
```

```
and[subclass[domain[domain[NATMUL]], omega],
  subclass[range[domain[NATMUL]], omega]] := True
```

This yields an upper bound on the domain of **NATMUL**.

```
subclass[domain[NATMUL], cart[omega, omega]]
```

```
True
```

## ■ Application

Before the new theorem can be applied successfully, a small preliminary matter must be disposed of; namely the possibility that the domain might be entirely empty. This is not the case because the starting generation is not empty:

```
Map[not, SubstTest[implies, and[member[0, u], equal[u, v]], member[0, v],
    {u -> domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]], v -> 0}]]
```

```
equal[0, domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]]] == False
```

```
equal[0, domain[iterate[iterate[SUCC, singleton[x_]], singleton[0]]]] := False
```

The new theorem can now be applied to the case of interest:

```
SubstTest[implies, subclass[range[iterate[u, v]], domain[u]],
    or[equal[0, v], equal[omega, domain[iterate[u, v]]]],
    {u -> iterate[SUCC, singleton[x]], v -> singleton[0]}]
```

```
or[equal[omega, domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]]],
  not[member[x, V]], not[
   subclass[range[iterate[iterate[SUCC, singleton[x]], singleton[0]]], omega]]] == True
```

This is added as a temporary rewrite rule:

```
or[equal[omega, domain[iterate[iterate[SUCC, singleton[x_]], singleton[0]]]],
  not[member[x_, V]], not[
   subclass[range[iterate[iterate[SUCC, singleton[x_]], singleton[0]]], omega]]] := True
```

From this we deduce:

```
Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
       not[implies[p1, p4]], {p1 -> member[x, omega], p2 -> member[x, V],
       p3 ->
    subclass[range[iterate[iterate[SUCC, singleton[x]], singleton[0]]], omega],
       p4 -> equal[omega, domain[iterate[
      iterate[SUCC, singleton[x]], singleton[0]]]]}]]
```

```
or[equal[omega, domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]]],
  not[member[x, omega]]] == True
```

```
or[equal[omega, domain[iterate[iterate[SUCC, singleton[x_]], singleton[0]]]],
  not[member[x_, omega]]] := True
```

This accomplishes our first main goal:

```
implies[member[x, omega], equal[omega, domain[composite[NATMUL, LEFT[x]]]]]

True
```

## ■ The rest of the story.

The rest of the notebook is devoted to eliminating the variable **x** in the above result. This turned out to be trickier than anticipated. The first step is to get **iterate** expressions to disappear:

```
SubstTest[domain, composite[n, LEFT[x]], n -> NATMUL]

intersection[domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]],
  image[V, intersection[omega, singleton[x]]]] == image[domain[NATMUL], singleton[x]]

intersection[domain[iterate[iterate[SUCC, singleton[x_]], singleton[0]]],
  image[V, intersection[omega, singleton[x_]]]] := image[domain[NATMUL], singleton[x]]
```

The theorem about domain of **composite[NATMUL,LEFT[x]]** can now be rewritten in a simpler fashion in which **iterate** does not appear:

```
Map[implies[member[x, omega], #] &, SubstTest[equal, omega,
  intersection[domain[iterate[iterate[SUCC, singleton[x]], singleton[0]]], image[V, y]],
  y -> intersection[omega, singleton[x]]]]

or[equal[omega, image[domain[NATMUL], singleton[x]]], not[member[x, omega]]] == True

or[equal[omega, image[domain[NATMUL], singleton[x_]]], not[member[x_, omega]]] := True
```

The next step is to eliminate the variable **x**, which we do as follows:

```
Map[equal[V, #] &, SubstTest[class, x, implies[member[x, o],
            equal[o, image[domain[n], singleton[x]]]], {o -> omega, n -> NATMUL}]]

True == and[subclass[cart[omega, omega], domain[NATMUL]],
  subclass[cart[omega, complement[omega]], complement[domain[NATMUL]]]]
```

This is actually two statements, of which we need only the simpler one. This is extracted as follows:

```
Map[implies[#, subclass[cart[omega, omega], domain[NATMUL]]] &, %]

subclass[cart[omega, omega], domain[NATMUL]] == True

subclass[cart[omega, omega], domain[NATMUL]] := True
```

The final step is to combine this lower bound with the upper bound found earlier.

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> domain[NATMUL], v -> cart[omega, omega]}] // Reverse

equal[cart[omega, omega], domain[NATMUL]] == True
```

This yields:

```
domain[NATMUL] := cart[omega, omega]
```

## ■ Final comments.

Now that we have the domain formula, many temporary formulas about the domain of **NATMUL** are no longer needed, and can be safely removed. For example, one no longer needs an explicit rewrite rule for **composite[-NATMUL,id[cart[omega,omega]]]**.