how to add natural numbers

Johan G. F. Belinfante
2002 July 28

```
<< goedel52.o98; << tools.m

:Package Title: goedel52.o98          2002 July 28 at 7:15 P.m.

It is now:  2002 Jul 28 at 23:25

Loading Simplification Rules

TOOLS.M                  Revised 2002 June 12

weightlimit = 40
```

■ Introduction

This notebook is concerned with developing practical rules for adding any two natural numbers. The first few natural numbers are:

```
NestList[succ, 0, 5]

{0, singleton[0], succ[singleton[0]], succ[succ[singleton[0]]],
 succ[succ[succ[singleton[0]]]], succ[succ[succ[succ[singleton[0]]]]]}
```

A particular number such as **3 = succ[succ[singleton[0]]]** can be conveniently written as

```
Nest[succ, 0, 3]

succ[succ[singleton[0]]]
```

The following rule in the **GOEDEL** program precedes the setting of attributes for **natadd**.

```
natadd[x]

x
```

Following this rule, the following attributes are set:

```
Attributes[natadd]

{Flat, OneIdentity, Orderless}
```

Because the commutative and associative laws are consequently known to the **GOEDEL** program, all one needs in practice to add any two numbers are rules for adding **0**, **singleton[0]**, and a rule for adding successors. These rules are derived in this notebook.

■ rules for adding 0 and 1 = singleton[0]

The rule for adding **0** is this:

```
natadd[x, 0] // Normality
```

```
natadd[0, x] == union[x, complement[image[V, intersection[omega, singleton[x]]]]]
```

Note that if **x** is a natural number, this says **0 + x = x**, while if **x** is not a natural number, this says **0 + x = V**. Both cases are taken care of with a single formula.

```
natadd[0, x_] := union[x, complement[image[V, intersection[omega, singleton[x]]]]]
```

The rule for adding **1 = singleton[0]** is similar:

```
natadd[x, singleton[0]] // Normality
```

```
natadd[x, singleton[0]] ==
 union[complement[image[V, intersection[omega, singleton[x]]]], succ[x]]
```

```
natadd[x_, singleton[0]] :=
 union[complement[image[V, intersection[omega, singleton[x]]]], succ[x]]
```

■ temporary simplification rules

Lemma 1.

```
natadd[x, union[y, complement[image[V, z]]]] // Normality
```

```
natadd[x, union[y, complement[image[V, z]]]] ==
 union[complement[image[V, z]], natadd[x, y]]
```

```
natadd[x_, union[y_, complement[image[V, z_]]]] :=
 union[complement[image[V, z]], natadd[x, y]]
```

Lemma 2

```
equal[
 union[complement[image[V, intersection[omega, singleton[w]]]], natadd[y, succ[w]]],
    natadd[y, succ[w]]]
```

```
True
```

```
union[complement[image[V, intersection[omega, singleton[w_]]]], natadd[y_, succ[w_]]] :=
 natadd[y, succ[w]]
```

Lemma 3

```
equal[
 union[complement[image[V, intersection[omega, singleton[x]]]], succ[natadd[x, y]]],
    succ[natadd[x, y]]]
```

```
True
```

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

■ successor rule

The following temporary abbreviation is useful:

```
plus[x_] := composite[NATADD, RIGHT[x]]
```

```
Map[A[image[#, singleton[0]]] &,
  SubstTest[composite, plus[x], plus[w], w -> succ[y]]] // Reverse
```

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

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

It should be noted that **x** and **y** need not be natural numbers here. This rule holds for arbitrary classes **x** and **y**. If either one of these classes fails to be a natural number, this equation reduces to **V = V**.

■ confluence

The successor rule and the rule for adding 1 are not confluent. To remedy this, we add a new successor rule:

```
succ[union[x, complement[image[V, y]]]] // Normality
```

```
succ[union[x, complement[image[V, y]]]] == union[complement[image[V, y]], succ[x]]
```

```
succ[union[x_, complement[image[V, y_]]]] := union[complement[image[V, y]], succ[x]]
```

This fixes the confluence problem:

```
SubstTest[natadd, x, succ[y], y -> 0]
```

```
True
```

■ An example

The following example shows that **2 + 3 = 5**.

```
natadd[Nest[succ, 0, 2], Nest[succ, 0, 3]] == Nest[succ, 0, 5]
```

```
True
```