Addition of natural numbers: rewrite rules and attributes for natadd.

Johan G. F. Belinfante
2002 July 28

```
<< goedel52.o97; << tools.m
```

```
:Package Title: goedel52.o97          2002 July 28 at 8:05 a.m.

It is now:  2002 Jul 28 at 17:29

Loading Simplification Rules

TOOLS.M                  Revised 2002 June 12

weightlimit = 40
```

■ Introduction

A wrapped membership rule for **natadd[x,y]** has been added to the **GOEDEL** program, from which the relation between the function **NATADD** and **natadd[x,y]** can be deduced:

```
lambda[pair[x, y], natadd[x, y]]
```

```
NATADD
```

In this notebook, basic rewrite rules involving **natadd[x,y]** are derived. Old rules related to the associative law of addition and the additive law of exponents are replaced. Justification is given for setting **Flat** and **Orderless** attributes for **natadd**.

■ Normality for natadd[x,y]

The **Normality** rule is needed for just about everything else, so it is a natural place to start.

```
natadd[x, y] // Normality // Reverse
```

```
A[image[NATADD, cart[singleton[x], singleton[y]]]] == natadd[x, y]
```

```
A[image[NATADD, cart[singleton[x_], singleton[y_]]]] := natadd[x, y]
```

From this it follows that **natadd[x,y]** is equal to **V** unless **x** and **y** are both natural numbers:

```
SubstTest[equal, V, A[image[NATADD, cart[u, v]]], {u -> singleton[x], v -> singleton[y]}]
```

```
equal[V, natadd[x, y]] == or[not[member[x, omega]], not[member[y, omega]]]
```

This is added as a rewrite rule, used below to derive the commutative property of **natadd**.

```
equal[V, natadd[x_, y_]] := or[not[member[x, omega]], not[member[y, omega]]]
```

■ Commutativity and the Orderless attribute

The commutativity of **NATADD** is used to derive the corresponding property of **natadd**.

```
SubstTest[implies, equal[u, v], equal[A[u], A[v]],
    {u -> image[NATADD, cart[singleton[x], singleton[y]]],
        v -> image[NATADD, cart[singleton[y], singleton[x]]]}]
```

```
equal[natadd[x, y], natadd[y, x]] == True
```

This commutative property justifies adding the attribute **Orderless** to **natadd**.

```
SetAttributes[natadd, Orderless]
```

Now *Mathematica* recognizes that **natadd** is commutative without needing to add any rewrite rules.

```
equal[natadd[x, y], natadd[y, x]]
```

```
True
```

■ Vertical section rule for NATADD

A vertical section rule for the binary function **NATADD** is deduced by using rules for **PAIR**:

```
SubstTest[implies, FUNCTION[z],
 equal[image[z, singleton[w]], singleton[A[image[z, singleton[w]]]]],
    {z -> NATADD, w -> PAIR[x, y]}]
```

```
equal[image[NATADD, cart[singleton[x], singleton[y]]], singleton[natadd[x, y]]] == True
```

This justifies adding the following rewrite rule:

```
image[NATADD, cart[singleton[x_], singleton[y_]]] := singleton[natadd[x, y]]
```

■ Sethood rule

From the vertical section rule one easily deduces a sethood rule:

```
Map[not, SubstTest[equal, 0, image[NATADD, singleton[z]], z -> PAIR[x, y]]]
```

```
member[natadd[x, y], V] == and[member[x, omega], member[y, omega]]
```

```
member[natadd[x_, y_], V] := and[member[x, omega], member[y, omega]]
```

The following corollary is worth noting.

```
image[V, singleton[natadd[x, y]]] // Normality

image[V, singleton[natadd[x, y]]] ==
 intersection[image[V, intersection[omega, singleton[x]]],
  image[V, intersection[omega, singleton[y]]]]

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

This rewrite rule causes problems unless it is supplemented an additional rule. The **GOEDEL** program recognizes this truth:

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

True
```

On account of this, one is justified in adding a corresponding rewrite rule:

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

■ A consequence of the associative law

There is an old rewrite rule that follows from the associative law:

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

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

This rule can be generalized. First, the old rule is removed:

```
composite[NATADD, RIGHT[x_], NATADD, RIGHT[y_]] =.
```

Instead of restoring it, we deduce a more general rewrite rule:

```
Assoc[composite[NATADD, cross[Id, NATADD]], ASSOC, RIGHT[x]] // Reverse

composite[NATADD, RIGHT[x], NATADD] ==
 composite[NATADD, cross[Id, composite[NATADD, RIGHT[x]]]]

composite[NATADD, RIGHT[x_], NATADD] :=
 composite[NATADD, cross[Id, composite[NATADD, RIGHT[x]]]]
```

The old rule follows from this new rule as a special case:

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

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

■ New additive law of exponents

The **GOEDEL** program currently contains the following additive law of exponents which involves **iterate** and **SUCC**:

```
composite[image[power[x], u], image[power[x], v]]
```

```
image[power[x], image[iterate[SUCC, v], u]]
```

We remove this old rule preparatory to deriving a better replacement rule.

```
composite[image[power[x_], u_], image[power[x_], v_]] =.
```

The **RIF** version of the additive law of exponents is still available:

```
composite[RIF, cross[power[x], power[x]]]
```

```
composite[SWAP, power[x], NATADD]
```

From it we can deduce a replacement for the removed additive law of exponents:

```
ImageComp[composite[SWAP, RIF], cross[power[x], power[x]], cart[u, v]] // Reverse
```

```
composite[image[power[x], u], image[power[x], v]] ==
 image[power[x], image[NATADD, cart[u, v]]]
```

```
composite[image[power[x_], u_], image[power[x_], v_]] :=
 image[power[x], image[NATADD, cart[u, v]]]
```

The special case of greatest interest is the following, which involves **natadd**.

```
composite[image[power[x], singleton[u]], image[power[x], singleton[v]]]
```

```
image[power[x], singleton[natadd[u, v]]]
```

■ The sum of natural numbers is a natural number

We start with this result:

```
ImageComp[id[omega], NATADD, cart[singleton[x], singleton[y]]] // Reverse
```

```
intersection[omega, singleton[natadd[x, y]]] == singleton[natadd[x, y]]
```

```
intersection[omega, singleton[natadd[x_, y_]]] := singleton[natadd[x, y]]
```

From it we deduce a temporary rule:

```
SubstTest[equal, intersection[omega, singleton[w]],
  singleton[w], w -> natadd[x, y]] // Reverse
```

```
or[member[natadd[x, y], omega], not[member[x, omega]], not[member[y, omega]]] == True
```

```
or[member[natadd[x_, y_], omega], not[member[x_, omega]], not[member[y_, omega]]] := True
```

The converse also holds:

```
or[and[member[x, omega], member[y, omega]],
  not[member[natadd[x, y], omega]]] // AssertTest
```

```
or[and[member[x, omega], member[y, omega]], not[member[natadd[x, y], omega]]] == True
```

```
or[and[member[x_, omega], member[y_, omega]], not[member[natadd[x_, y_], omega]]] := True
```

These two results can be combined into a single rule:

```
equiv[member[natadd[x, y], omega], and[member[x, omega], member[y, omega]]]
```

```
True
```

```
member[natadd[x_, y_], omega] := and[member[x, omega], member[y, omega]]
```

Various related facts are automatically recognized as consequences, and do not require additional rules. For example:

```
or[equal[natadd[x, y], V], member[natadd[x, y], omega]]
```

```
True
```

■ Associativity and the Flat attribute

The **GOEDEL** program recognizes this truth:

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

```
True
```

Consequently one is justified in adding a corresponding rewrite rule:

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

With this rule in place, one deduces that **natadd** is associative.

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

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

This associative property justifies adding the attribute **Flat** to **natadd**.

```
SetAttributes[natadd, Flat]
```

Now *Mathematica* recognizes that **natadd** is associative without needing to add any rewrite rules.

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

```
True
```