FUNCTION[NATADD]

Johan G. F. Belinfante
2002 June 13

```
<< goedel52.o51; << tools.m
```

```
:Package Title: goedel52.o51          2002 June 13 at 4:30 p.m.

It is now:  2002 Jun 13 at 20:11

Loading Simplification Rules

TOOLS.M                   Revised 2002 June 12

weightlimit = 40
```

■ Introduction

This notebook contains a proof that **NATADD** is a function. The idea is to show that a ternary relation is a function if all vertical sections parallel to one of the coordinate axes is a function.

■ A rule for binary functions

```
FUNCTION[composite[x, id[cart[V, V]]]] // AssertTest // Reverse
```

```
equal[0, domain[fix[composite[inverse[x], Di, x]]]] ==
 FUNCTION[composite[x, id[cart[V, V]]]]
```

```
equal[0, domain[fix[composite[inverse[x_], Di, x_]]]] :=
 FUNCTION[composite[x, id[cart[V, V]]]]
```

A double application of **assert** is needed to prove the theorem alluded to:

```
assert[assert[forall[y, FUNCTION[composite[x, LEFT[y]]]]]]
```

```
FUNCTION[composite[x, id[cart[V, V]]]]
```

There is also a version with **RIGHT** in place of **LEFT**.

```
assert[assert[forall[y, FUNCTION[composite[x, RIGHT[y]]]]]]
```

```
FUNCTION[composite[x, id[cart[V, V]]]]
```

For **NATADD** either will do because the commutative law of addition says these are the same:

```
Assoc[NATADD, SWAP, RIGHT[x]]
```

```
composite[NATADD, LEFT[x]] == composite[NATADD, RIGHT[x]]
```

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

■ first steps

There is little in the **GOEDEL** program that is helpful for proving that relations constructed by iterations are functions. About all that is available at this point is this:

```
FUNCTION[iterate[funpart[x], singleton[y]]]
```

```
True
```

We apply this to the case of the restriction of **SUCC** to **omega**.

```
SubstTest[FUNCTION, iterate[funpart[f], singleton[x]], f -> composite[id[omega], SUCC]]
```

```
FUNCTION[iterate[composite[id[omega], SUCC], singleton[x]]] == True
```

```
FUNCTION[iterate[composite[id[omega], SUCC], singleton[x_]]] := True
```

```
Map[FUNCTION,
 SubstTest[composite, SECOND, id[cart[singleton[x], V]], power[funpart[y]],
    y -> composite[id[omega], SUCC]]]
```

```
FUNCTION[union[cart[singleton[0], singleton[x]], composite[NATADD, RIGHT[x]]]] == True
```

```
FUNCTION[union[cart[singleton[0], singleton[x]], composite[NATADD, RIGHT[x_]]]] := True
```

```
SubstTest[implies, and[subclass[u, v], FUNCTION[v]], FUNCTION[u],
        {u -> composite[NATADD, RIGHT[x]],
  v -> union[cart[singleton[0], singleton[x]], composite[NATADD, RIGHT[x]]]}]
```

```
FUNCTION[composite[NATADD, RIGHT[x]]] == True
```

```
FUNCTION[composite[NATADD, RIGHT[x_]]] := True
```

```
Map[complement,
  SubstTest[class, x, FUNCTION[composite[y, RIGHT[x]]], y -> NATADD]] // Reverse
```

```
fix[composite[SECOND, intersection[composite[inverse[FIRST], FIRST],
    composite[inverse[NATADD], Di, NATADD]], inverse[SECOND]]] == 0
```

```
fix[composite[SECOND, intersection[composite[inverse[FIRST], FIRST],
    composite[inverse[NATADD], Di, NATADD]], inverse[SECOND]]] := 0
```

```
SubstTest[assert,
  equal[0, fix[composite[SECOND, intersection[composite[inverse[FIRST], FIRST],
      composite[inverse[x], Di, x]], inverse[SECOND]]]], x -> NATADD] // Reverse
```

```
FUNCTION[NATADD] == True
```

```
FUNCTION[NATADD] := True
```