divisors of 1

Johan G. F. Belinfante
2002 September 3

```
<< goedel52.p31; << tools.m
```

```
:Package Title: goedel52.p31        2002 September 2 at 8:48 p.m.
```

```
It is now:  2002 Sep 3 at 3:26
```

```
Loading Simplification Rules
```

```
TOOLS.M                  Revised 2002 August 30
```

```
weightlimit = 40
```

■ summary

The goal in this notebook is to derive the elementary fact that **1 = singleton[0]** is the only divisor of **1**, and the related fact that any factorization of **1** must be trivial. The latter result is stated first without variables, and then with variables.

■ divisors of 1

The key to determining the set of divisors of **1** is to make use of the connection between the divisivility relation **DIV** and the subclass relation **S**.

```
SubstTest[implies, subclass[u, v],
    subclass[image[inverse[u], w], image[inverse[v], w]],
    {u -> DIV, v -> union[S, cart[omega, singleton[0]]], w -> singleton[singleton[0]]}]
```

```
subclass[image[inverse[DIV], singleton[singleton[0]]], succ[singleton[0]]] == True
```

```
subclass[image[inverse[DIV], singleton[singleton[0]]], succ[singleton[0]]] := True
```

Note that the class **succ[singleton[0]]** only has two elements:

```
pairset[0, singleton[0]]
```

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

The only thing left to do therefore is to rule out the possibility that **0** is a divisor of **1**. This follows from the known fact that the only multiple of **0** is **0**.

```
SubstTest[member, singleton[0], image[DIV, singleton[x]], x -> 0] // Reverse
```

```
member[pair[0, singleton[0]], DIV] == False
```

```
member[pair[0, singleton[0]], DIV] := False
```

Thus the set of divisors of **1** is contained in the singleton of **1**.

```
SubstTest[subclass, image[inverse[DIV], singleton[singleton[0]]],
    intersection[u, v],
    {u -> succ[singleton[0]], v -> complement[singleton[0]]}]
```

```
subclass[image[inverse[DIV], singleton[singleton[0]]], singleton[singleton[0]]] == True
```

```
subclass[image[inverse[DIV], singleton[singleton[0]]], singleton[singleton[0]]] := True
```

This containment can be strengthened to an equation as follows.

```
SubstTest[member, singleton[0], image[DIV, singleton[x]], x -> singleton[0]] // Reverse
```

```
member[pair[singleton[0], singleton[0]], DIV] == True
```

```
member[pair[singleton[0], singleton[0]], DIV] := True
```

We use the fact that two classes are equal if each is contained in the other:

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> singleton[singleton[0]], v -> image[inverse[DIV], singleton[singleton[0]]]}]
```

```
True == equal[image[inverse[DIV], singleton[singleton[0]]], singleton[singleton[0]]]
```

This is the final result.

```
image[inverse[DIV], singleton[singleton[0]]] := singleton[singleton[0]]
```

■ all factorizations of 1 are trivial

From the theorem about divisors of 1 one can derive that fact that factorizations of 1 are trivial. A key step used in establishing this corollary turned out to be something one might not have expected:

```
IminComp[NATMUL, id[cart[V, V]], x] // Reverse
```

```
composite[Id, image[inverse[NATMUL], x]] == image[inverse[NATMUL], x]
```

```
composite[Id, image[inverse[NATMUL], x_]] := image[inverse[NATMUL], x]
```

The reason this fact is so important is that the **GOEDEL** program contains some conditional rewrite rules that apply only to relations. In these rewrite rules the condition **composite[Id,x] == x** is used to determine whether a given class **x** is a relation. The entire derivation cannow be done in one step:

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> id[singleton[singleton[0]]],
  v -> image[inverse[NATMUL], singleton[singleton[0]]]}]
```

```
True == equal[cart[singleton[singleton[0]], singleton[singleton[0]]],
  image[inverse[NATMUL], singleton[singleton[0]]]]
```

This establishes a variable−freeversion of the corollary:

```
image[inverse[NATMUL], singleton[singleton[0]]] :=
 cart[singleton[singleton[0]], singleton[singleton[0]]]
```

■ reformulating the corollary

We now derive a restatement of the theorem about factoring **1** using two variables. First we have to derive the fact that any factorization of **1** must use natural numbers as factors:

```
SubstTest[implies, and[equal[u, v], member[v, V]], member[u, V],
    {u -> natmul[x, y], v -> singleton[0]}]
```

```
or[and[member[x, omega], member[y, omega]],
  not[equal[natmul[x, y], singleton[0]]]] == True
```

```
or[and[member[x_, omega], member[y_, omega]],
  not[equal[natmul[x_, y_], singleton[0]]]] := True
```

The following observation ...

```
equiv[and[equal[natmul[x, y], singleton[0]], member[x, omega], member[y, omega]],
    equal[natmul[x, y], singleton[0]]]
```

```
True
```

... justifies adding this temporary simplification rule:

```
and[equal[natmul[x_, y_], singleton[0]], member[x_, omega], member[y_, omega]] :=
 equal[natmul[x, y], singleton[0]]
```

The first version of the corollary now is used to deduce an alternate version of it:

```
SubstTest[member, pair[x, y], image[inverse[NATMUL], w],
    w -> singleton[singleton[0]]] // Reverse
```

```
equal[natmul[x, y], singleton[0]] == and[equal[x, singleton[0]], equal[y, singleton[0]]]
```

Thus **1** is the product of **x** and **y** if and only if **x** = **y** = **1**:

```
equal[natmul[x_, y_], singleton[0]] :=
 and[equal[x, singleton[0]], equal[y, singleton[0]]]
```