ON−7−Aa lemma needed for a variant of induction

Johan G. F. Belinfante
2002 April 20

```
<< goedel52.n32; << tests.m

:Package Title: GOEDEL52.N32     2002 April 19 at 8:55 p.m.

It is now:  2002 Apr 20 at 22:7

Loading Simplification Rules

TESTS.M                  Revised 2002 April 16

weightlimit = 40


Context switch to ‘Goedel‘Private is needed for ReplaceTest

Just ignore the error message about Unterminated use of BeginPackage

Get::bebal : Unterminated uses of BeginPackage or Begin in << tests.m.
```

■ Mathematical Induction

The basic rule of induction is this:

```
implies[and[member[0, x], subclass[image[SUCC, x], x]], subclass[omega, x]]

True
```

Various other versions can be derived from this one. One of these variants relies on Lemma **ON−7−A** which says that the empty set belongs to every nonempty natural number. This theorem had been proved using **Otter** as a corollary of a more general theorem about ordinal numbers. Here we give a different proof of this lemma, using a short induction argument. The way induction is used in this proof struck me as quite novel. In particular the proof uses **inverse[SUCC]** rather than **SUCC**.

Before starting we point out that the class of sets holding the empty set is **image[E,singleton[0]]**. The **GOEDEL** program rewrites this using power classes:

```
member[x, image[E, singleton[0]]]

and[member[0, x], member[x, V]]

image[E, singleton[0]]

complement[P[complement[singleton[0]]]]
```

■ The Proof

The novel step is the first one:

```
SubstTest[equal, V, union[complement[v], image[inverse[SUCC], v]],
v -> union[singleton[0], image[E, singleton[0]]]] // Reverse
```

```
subclass[intersection[image[SUCC, complement[P[complement[singleton[0]]]]],
   P[complement[singleton[0]]]], singleton[0]] == True
```

```
subclass[intersection[image[SUCC, complement[P[complement[singleton[0]]]]],
   P[complement[singleton[0]]]], singleton[0]] := True
```

Next we use the basic version of induction:

```
SubstTest[implies, and[member[0, z], subclass[image[SUCC, z], z]], subclass[omega, z],
z -> union[singleton[0], image[E, singleton[0]]]]
```

```
subclass[intersection[omega, P[complement[singleton[0]]]], singleton[0]] == True
```

```
subclass[intersection[omega, P[complement[singleton[0]]]], singleton[0]] := True
```

We are really done at this point. The next step is optional; it just helps to understand what it is that we have proved.

```
SubstTest[implies, and[member[x, y], subclass[y, z]], member[x, z],
{y -> intersection[omega, P[complement[singleton[0]]]], z -> singleton[0]}]
```

```
or[equal[0, x], member[0, x], not[member[x, omega]]] == True
```

```
or[equal[0, x_], member[0, x_], not[member[x_, omega]]] := True
```

■ A slightly better result

The inclusion proved above is easily replaced by an equality; that amounts to adding the (trivial) converse theorem. This is well worth doing because equations make better simplification rules.

```
SubstTest[and, subclass[u, v], subclass[v, u],
 {u -> singleton[0], v -> intersection[omega, P[complement[singleton[0]]]]}]
```

```
True == equal[intersection[omega, P[complement[singleton[0]]]], singleton[0]]
```

This justifies the following rule:

```
intersection[omega, P[complement[singleton[0]]]] := singleton[0]
```