A[dif[y,x]] formula

Johan G. F. Belinfante
2002 May 31

```
<< goedel52.o16; << tools.m
```

```
:Package Title: goedel52.o16                        2002 May 31 at 1:00 p.m.

It is now:  2002 May 31 at 20:48

Loading Simplification Rules

TOOLS.M                    Revised 2002 May 22

weightlimit = 40
```

■ Introduction

In this notebook it is shown that the least member of the set theoretic difference between two distinct ordinals is the lesser of the two.

■ lemma: a corollary of ON−5B

This lemma just says **OMEGA** is full.

```
SubstTest[implies, and[member[x, y], member[y, z]], member[x, U[z]], z -> OMEGA]

or[member[x, OMEGA], not[member[x, y]], not[member[y, OMEGA]]] == True

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

■ Theorem ON−A−5

An important ingredient in our derivaion is Theorem **ON−A−5**:

```
implies[member[x, OMEGA], equal[x, A[dif[OMEGA, x]]]]

True
```

■ another lemma

The functor **A** is antitone. We need a certain variant of this fact.

```
SubstTest[implies, subclass[u, v], subclass[A[v], A[u]],
    {u -> dif[y, x], v -> dif[z, x]}]
```

```
or[not[subclass[y, union[x, z]]], subclass[
   A[intersection[z, complement[x]]], A[intersection[y, complement[x]]]]] == True
```

```
or[not[subclass[y_, union[x_, z_]]], subclass[
   A[intersection[z_, complement[x_]]], A[intersection[y_, complement[x_]]]]] := True
```

```
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
        {p1 -> subclass[y, z], p2 -> subclass[y, union[x, z]],
            p3 -> subclass[A[dif[z, x]], A[dif[y, x]]]}]]
```

```
or[not[subclass[y, z]], subclass[
   A[intersection[z, complement[x]]], A[intersection[y, complement[x]]]]] == True
```

```
or[not[subclass[y_, z_]], subclass[
   A[intersection[z_, complement[x_]]], A[intersection[y_, complement[x_]]]]] := True
```

We will need this special case: **z = OMEGA**.

```
implies[subclass[y, OMEGA], subclass[A[dif[OMEGA, x]], A[dif[y, x]]]]
```

```
True
```

Similarly, we need the following related result:

```
SubstTest[implies, member[x, z], subclass[A[z], x], z -> dif[y, x]]
```

```
or[member[x, x], not[member[x, y]],
  subclass[A[intersection[y, complement[x]]], x]] == True
```

```
or[member[x_, x_], not[member[x_, y_]],
  subclass[A[intersection[y_, complement[x_]]], x_]] := True
```

■ Put it all together...

I tried first to derive the desired formula all in one step, but got tired of waiting. It is much faster in two steps. The first is:

```
Map[not, SubstTest[and, implies[and[p1, p2], p3],
  implies[p3, p4], implies[p2, p5], implies[p5, p6],
    implies[and[p4, p6], p7],     not[implies[and[p1, p2], p7]],
    {p1 -> member[x, y], p2 -> member[y, OMEGA], p3 -> member[x, OMEGA],
        p4 -> equal[x, A[dif[OMEGA, x]]], p5 -> subclass[y, OMEGA],
        p6 -> subclass[A[dif[OMEGA, x]], A[dif[y, x]]],
   p7 -> subclass[x, A[dif[y, x]]]}]]
```

```
or[not[member[x, y]], not[member[y, OMEGA]],
  subclass[x, A[intersection[y, complement[x]]]]] == True
```

```
or[not[member[x_, y_]], not[member[y_, OMEGA]],
  subclass[x_, A[intersection[y_, complement[x]]]]] := True
```

The second step yields what we wanted:

```
Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p2], p7],
    implies[p3, p8], implies[and[p1, p8], p9], implies[and[p7, p9], p10],
    not[implies[and[p1, p2], p10]],
    {p1 -> member[x, y], p2 -> member[y, OMEGA], p3 -> member[x, OMEGA],
        p4 -> equal[x, A[dif[OMEGA, x]]], p5 -> subclass[y, OMEGA],
        p6 -> subclass[A[dif[OMEGA, x]], A[dif[y, x]]],
   p7 -> subclass[x, A[dif[y, x]]],
        p8 -> not[member[x, x]], p9 -> subclass[A[dif[y, x]], x],
   p10 -> equal[A[dif[y, x]], x]}]]
```

```
or[equal[x, A[intersection[y, complement[x]]]],
  not[member[x, y]], not[member[y, OMEGA]]] == True
```

```
or[equal[x_, A[intersection[y_, complement[x_]]]],
  not[member[x_, y_]], not[member[y_, OMEGA]]] := True
```

■ A special case of interest: y = omega.

Since the variables **x** and **y** refer to sets, it is possible to eliminate them. In general, a rather complicated result is obtained when this is done, but I did succeed in getting a variable−freeformula for a special case of interest, namely when **y** is the set **omega** of all natural numbers.

```
SubstTest[implies, and[member[x, y], member[y, OMEGA]],
 equal[x, A[dif[y, x]]], y -> omega]
```

```
or[equal[x, A[intersection[omega, complement[x]]]], not[member[x, omega]]] == True
```

```
or[equal[x_, A[intersection[omega, complement[x_]]]], not[member[x_, omega]]] := True
```

```
or[and[equal[x, A[intersection[omega, complement[x]]]], subclass[x, omega]],
  not[member[x, omega]]] // NotNotTest
```

```
or[and[equal[x, A[intersection[omega, complement[x]]]], subclass[x, omega]],
  not[member[x, omega]]] == True
```

```
or[and[equal[x_, A[intersection[omega, complement[x_]]]], subclass[x_, omega]],
  not[member[x_, omega]]] := True
```

```
Map[equal[V, #] &,
 union[complement[omega], fix[composite[BIGCAP, RC[omega]]]] // Renormality]
```

```
subclass[omega, fix[composite[BIGCAP, RC[omega]]]] == True
```

```
subclass[omega, fix[composite[BIGCAP, RC[omega]]]] := True
```

```
SubstTest[subclass, fix[w], range[w], w -> composite[BIGCAP, RC[omega]]]
```

```
subclass[fix[composite[BIGCAP, RC[omega]]], omega] == True
```

```
subclass[fix[composite[BIGCAP, RC[omega]]], omega] := True
```

```
SubstTest[and, subclass[x, y], subclass[y, x],
    {x -> omega, y -> fix[composite[BIGCAP, RC[omega]]]}]
```

```
True == equal[omega, fix[composite[BIGCAP, RC[omega]]]]
```

Thus we have derived the following interesting rewrite rule:

```
fix[composite[BIGCAP, RC[omega]]] := omega
```