A[dif[y,x]] formula

Johan G. F. Belinfante
2002 May 31

\[
\text{SubstTest}[\text{implies}, \text{and}[[\text{member}[x, y], \text{member}[y, z]], \text{member}[x, U[z]], z \rightarrow \text{OMEGA}]]
\]

or[[\text{member}[x, \text{OMEGA}], \text{not}[[\text{member}[x, y]], \text{not}[[\text{member}[y, \text{OMEGA}]]] == \text{True}]

or[[\text{member}[x_\_], \text{OMEGA}], \text{not}[[\text{member}[x_\_, y_\_]], \text{not}[[\text{member}[y_\_, \text{OMEGA}]]] == \text{True}

\textbf{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.

\textbf{lemma: a corollary of ON–5B}

This lemma just says \text{OMEGA} is full.

\textbf{Theorem ON–A–5}

An important ingredient in our derivation is Theorem \text{ON–A–5}:

\text{implies}[\text{member}[x, \text{OMEGA}], \text{equal}[x, \text{A[dif[OMEGA, x]]}]]

\text{True}

\textbf{another lemma}

The functor \text{A} is antitone. We need a certain variant of this fact.
The second step yields what we wanted:

We will need this special case: \( z = \text{OMEGA} \).

Similarly, we need the following related result:


\[
\text{SubstTest}[\text{implies}, \text{member}[x, z], \text{subclass}[A[x], x], z \rightarrow \text{dif}[y, x]]
\]

\[
\text{or}[\text{member}[x, x], \text{not} [\text{member}[x, y]], \text{subclass}[A[\text{intersection}[y, \text{complement}[x]]], x]] = \text{True}
\]

\[
\text{or}[\text{member}[x, _], \text{not} [\text{member}[x, _]], \text{subclass}[A[\text{intersection}[y, \text{complement}[x]]], x]] = \text{True}
\]

**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:

\[
\text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[p1, p2], \text{implies}[p2, p3], \text{not} [\text{implies}[p1, p2]], \\
\{p1 \rightarrow \text{subclass}[y, z], p2 \rightarrow \text{subclass}[y, \text{union}[x, z]], \text{p3} \rightarrow \text{subclass}[A[\text{dif}[z, x]], A[\text{dif}[y, x]]]]]]
\]

\[
\text{or}[\text{subclass}[y, \text{OMEGA}], \text{subclass}[A[\text{dif[OMEGA, x]]], A[\text{dif}[y, x]]]]
\]

\[
\text{True}
\]

The second step yields what we wanted:
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–free formula for a special case of interest, namely when \( y \) is the set \( \omega \) of all natural numbers.

```
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[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
```

```
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
```