the limit ordinals form a proper class

Johan G. F. Belinfante
2003 June 6

In[1]:=
```
<< goedel52.r97; << tools.m
```

```
:Package Title: goedel52.r97       2003 June 4 at 12:05 noon

It is now:  2003 Jun 6 at 11:15

Loading Simplification Rules

TOOLS.M                      Revised 2003 May 31

weightlimit = 40
```

■ summary

In Gödel's class theory, there are two kinds of classes, sets and proper classes. A class is a set if it is a member of some class. Proper classes are not members of any class. The class of all ordinals **OMEGA** is an example of a proper class; that is, this class does not belong to any class. Every ordinal **x** is full, that is, its sum class **U[x]** is contained in **x**. An ordinal **x** is called a limit ordinal if equality holds: **U[x] = x**. The class of limit ordinals is

In[2]:=
```
class[x, and[member[x, OMEGA], equal[U[x], x]]]
```

Out[2]=
```
intersection[OMEGA, fix[BIGCUP]]
```

A successor ordinal is an ordinal that is the successor of some other ordinal. An ordinal is a successor ordinal if and only if it is not a limit ordinal:

In[3]:=
```
class[x, exists[y, and[equal[x, succ[y]], member[y, OMEGA]]]]
```

Out[3]=
```
intersection[OMEGA, complement[fix[BIGCUP]]]
```

It was shown in the notebook **ON-SC-4B.NB** that the successor ordinals form a proper class.

In[4]:=
```
member[intersection[OMEGA, complement[fix[BIGCUP]]], V]
```

Out[4]=
```
False
```

In this notebook the uniqueness theorem for **iterate** is used to derive the fact that the class of limit ordinals is also a proper class.   The basic strategy is to show that any ordinal belongs to a limit ordinal. If **x** is an ordinal number, then the set **range[iterate[SUCC,singleton[x]]]** is the successor−invariantset of ordinals **{x, x+1, x+2, ... }**. It will be shown in this notebook that the sum class of this set is a limit ordinal to which **x** belongs. Knowing that any ordinal belongs to a limit ordinal allows one to deduce that the sum class of the the class of limit ordinals is the class of all ordinals. Since the latter is a proper class, it follows immediately that the class of limit ordinals is also a proper class. This is because a class is a proper class if and only if its sum class is a proper class.

■ lemma

The first task is to show that the class of limit ordinals is identical with the class of unions of successor−invariant sets of ordinals. We begin with a lemma based on Theorem **ON−SUC−1**which says that for ordinals **x** < **y** implies **x + 1 < y + 1**. The implication **implies[p4,p5]** in the following reasoning can be obtained from Theorem **ON−SUC−1** by eliminating the variable **x**.

```
In[5]:= Map[not, SubstTest[and, implies[and[p1, p2], p4], implies[p4, p5],
          implies[and[p1, p3], p6], implies[p6, p7], implies[and[p5, p7], p8],
                not[implies[and[p1, p2, p3], p8]],
                {p1 -> member[z, x], p2 -> subclass[x, OMEGA],
           p3 -> invariant[SUCC, x], p4 -> member[z, OMEGA],
                    p5 -> subclass[image[SUCC, z], succ[z]],
           p6 -> member[succ[z], x], p7 -> subclass[succ[z], U[x]],
                p8 -> subclass[image[SUCC, z], U[x]]}]]

Out[5]= or[not[member[z, x]], not[subclass[x, OMEGA]],
          not[subclass[image[SUCC, x], x]], subclass[image[SUCC, z], U[x]]] == True
```

The variable **z** can be eliminated:

```
In[6]:= Map[assert[forall[z, #]] &, %]

Out[6]= or[not[subclass[x, OMEGA]],
          not[subclass[image[SUCC, x], x]], subclass[image[SUCC, U[x]], U[x]]] == True

In[7]:= or[not[subclass[x_, OMEGA]], not[subclass[image[SUCC, x_], x_]],
          subclass[image[SUCC, U[x_]], U[x_]]] := True
```

Restatement: if **x** is a successor−invariant set of ordinals, the **U[x]** is also successor−invariant. Eliminating the variable **x** yields:.

```
In[8]:= Map[equal[0, #] &, dif[intersection[P[OMEGA], invar[SUCC]],
           image[inverse[BIGCUP], invar[SUCC]]] // Renormality]

Out[8]= subclass[image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]], invar[SUCC]] == True

In[9]:= subclass[image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]], invar[SUCC]] := True
```

The sum class of any subset of the class of ordinal numbers is an ordinal number: The following corollary of this will be needed:

```
In[10]:= SubstTest[equal, 0, dif[u, v],
             {u -> intersection[x, P[OMEGA]], v -> image[inverse[BIGCUP], OMEGA]}] // Reverse

Out[10]= subclass[image[BIGCUP, intersection[x, P[OMEGA]]], OMEGA] == True

In[11]:= subclass[image[BIGCUP, intersection[x_, P[OMEGA]]], OMEGA] := True
```

From the above facts one deduces:

```
In[12]:= SubstTest[subclass, u, intersection[v, w],
             {u -> image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]],
          v -> OMEGA, w -> invar[SUCC]}]

Out[12]= subclass[image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]], fix[BIGCUP]] == True
```

```
In[13]:= subclass[image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]], fix[BIGCUP]] := True
```

In the next section, the reverse inclusion is derived.

■ inclusion in the other direction

We begin with a lemma:

```
In[14]:= SubstTest[implies, and[member[y, x], subclass[x, z]],
           member[y, z], z -> fix[BIGCUP]] // MapNotNot

Out[14]= or[equal[y, U[y]], not[member[y, x]], not[subclass[x, fix[BIGCUP]]]] == True

In[15]:= or[equal[y_, U[y_]], not[member[y_, x_]], not[subclass[x_, fix[BIGCUP]]]] := True
```

From this one obtains:

```
In[16]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[p1, p4], implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
             {p1 -> member[y, x], p2 -> subclass[x, fix[BIGCUP]], p3 -> equal[U[y], y],
                 p4 -> member[U[y], image[BIGCUP, x]], p5 -> member[y, image[BIGCUP, x]]}]]

Out[16]= or[member[y, image[BIGCUP, x]],
           not[member[y, x]], not[subclass[x, fix[BIGCUP]]]] == True
```

The variable **y** can be eliminated:

```
In[17]:= Map[assert[forall[y, #]] &, %]

Out[17]= or[not[subclass[x, fix[BIGCUP]]], subclass[x, image[BIGCUP, x]]] == True
```

This says that any class of fixed points of **BIGCUP** is subvariant under **BIGCUP**.   Remark: This is by no means limited to **BIGCUP**.

```
In[18]:= or[not[subclass[x_, fix[BIGCUP]]], subclass[x_, image[BIGCUP, x_]]] := True
```

To apply this to the case at hand requires two inclusions. This is the first:

```
In[19]:= SubstTest[implies, subclass[x, fix[BIGCUP]], subclass[x, image[BIGCUP, x]],
             x -> intersection[OMEGA, fix[BIGCUP]]]

Out[19]= subclass[intersection[OMEGA, fix[BIGCUP]],
           image[BIGCUP, intersection[OMEGA, fix[BIGCUP]]]] == True

In[20]:= subclass[intersection[OMEGA, fix[BIGCUP]],
           image[BIGCUP, intersection[OMEGA, fix[BIGCUP]]]] := True
```

We need a lemma to derive the second inclusion.

```
In[21]:= SubstTest[subclass, intersection[x, y], intersection[x, P[U[y]]], y -> OMEGA]

Out[21]= subclass[U[intersection[OMEGA, x]], OMEGA] == True

In[22]:= subclass[U[intersection[OMEGA, x_]], OMEGA] := True
```

This is the second inclusion:

```
In[23]:= SubstTest[implies, subclass[u, v], subclass[image[BIGCUP, u], image[BIGCUP, v]],
           {u -> intersection[OMEGA, fix[BIGCUP]], v -> intersection[P[OMEGA], invar[SUCC]]}]

Out[23]= subclass[image[BIGCUP, intersection[OMEGA, fix[BIGCUP]]],
          image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]]] == True

In[24]:= subclass[image[BIGCUP, intersection[OMEGA, fix[BIGCUP]]],
          image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]]] := True
```

Applying transitivity of inclusion, one obtains:

```
In[25]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> intersection[OMEGA, fix[BIGCUP]],
             v -> image[BIGCUP, intersection[OMEGA, fix[BIGCUP]]],
             w -> image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]]}]

Out[25]= subclass[intersection[OMEGA, fix[BIGCUP]],
          image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]]] == True

In[26]:= subclass[intersection[OMEGA, fix[BIGCUP]],
          image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]]] := True
```

It only remains to combine the two inclusions into an equation:

```
In[27]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> intersection[OMEGA, fix[BIGCUP]],
          v -> image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]]}]

Out[27]= True == equal[image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]],
          intersection[OMEGA, fix[BIGCUP]]]

In[28]:= image[BIGCUP, intersection[invar[SUCC], P[OMEGA]]] := intersection[OMEGA, fix[BIGCUP]]
```

■ iterate uniqueness

Apply the uniqueness theorem for **iterate**:

```
In[29]:= SubstTest[implies, and[equal[image[w, singleton[0]], v],
               equal[composite[u, w], composite[w, SUCC]]],
           equal[iterate[u, v], composite[w, id[omega]]],
           {u -> SUCC, v -> singleton[x],
          w -> composite[id[OMEGA], iterate[SUCC, singleton[x]]]}]

Out[29]= or[and[member[x, V], not[member[x, OMEGA]]],
          subclass[range[iterate[SUCC, singleton[x]]], OMEGA]] == True
```

This can be cleaned up:

```
In[30]:= Map[implies[member[x, OMEGA], #] &, %]

Out[30]= or[not[member[x, OMEGA]], subclass[range[iterate[SUCC, singleton[x]]], OMEGA]] == True

In[31]:= or[not[member[x_, OMEGA]],
          subclass[range[iterate[SUCC, singleton[x_]]], OMEGA]] := True
```

This just says that if **x** is an ordinal, then **{x, x+1, x+2, ... }** is a set of ordinals.

■ lemma

This section is concerned with a general lemma, not specifically about ordinals. This lemma is needed to show that **x** belongs to the sum class of the set of ordinals **{x, x+1, ... }**.

```
In[32]:= SubstTest[subclass, image[u, v], range[u],
              {u -> iterate[x, y], v -> singleton[singleton[0]]}]

Out[32]= subclass[image[x, y], range[iterate[x, y]]] == True

In[33]:= subclass[image[x_, y_], range[iterate[x_, y_]]] := True

In[34]:= SubstTest[subclass, image[u, v], range[iterate[u, v]], {u -> SUCC, v -> singleton[x]}]

Out[34]= or[member[succ[x], range[iterate[SUCC, singleton[x]]]], not[member[x, V]]] == True

In[35]:= or[member[succ[x_], range[iterate[SUCC, singleton[x_]]]], not[member[x_, V]]] := True

In[36]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
                 {p1 -> member[x, y], p2 -> member[x, V],
            p3 -> member[succ[x], range[iterate[SUCC, singleton[x]]]],
                     p4 -> member[x, U[range[iterate[SUCC, singleton[x]]]]]}]]

Out[36]= or[member[x, U[range[iterate[SUCC, singleton[x]]]]], not[member[x, y]]] == True

In[37]:= or[member[x_, U[range[iterate[SUCC, singleton[x_]]]]], not[member[x_, y_]]] := True
```

For the next section we rewrite this a bit. Note that

```
In[38]:= class[y, member[x, U[y]]]

Out[38]= complement[P[P[complement[singleton[x]]]]]
```

The result of this section can therefore be rewritten as follows:

```
In[39]:= implies[member[x, y], member[range[iterate[SUCC, singleton[x]]],
           complement[P[P[complement[singleton[x]]]]]]]

Out[39]= True
```

■ eliminating the details of the construction

The next step is to hide the details of the **iterate** construction. All one really needs to know is that there exists a limit ordinal to which any given ordinal belongs; how it is constructed is not important. We will shortly need this:

```
In[40]:= SubstTest[U, image[BIGCUP, x], x -> intersection[invar[SUCC], P[OMEGA]]] // Reverse

Out[40]= U[U[intersection[invar[SUCC], P[OMEGA]]]] == U[intersection[OMEGA, fix[BIGCUP]]]

In[41]:= U[U[intersection[invar[SUCC], P[OMEGA]]]] := U[intersection[OMEGA, fix[BIGCUP]]]
```

and this..

```
In[42]:= implies[member[x, OMEGA], member[range[iterate[SUCC, singleton[x]]],
               intersection[P[OMEGA], invar[SUCC],
           complement[P[P[complement[singleton[x]]]]]]]] // NotNotTest

Out[42]= or[and[member[x, U[range[iterate[SUCC, singleton[x]]]]],
            subclass[range[iterate[SUCC, singleton[x]]], OMEGA]], not[member[x, OMEGA]]] == True

In[43]:= or[and[member[x_, U[range[iterate[SUCC, singleton[x_]]]]], subclass[
           range[iterate[SUCC, singleton[x_]]], OMEGA]], not[member[x_, OMEGA]]] := True
```

These facts are combined as follows:

```
In[44]:= SubstTest[implies, implies[p, member[r, s]], implies[p, not[equal[0, s]]],
           {p -> member[x, OMEGA], r -> range[iterate[SUCC, singleton[x]]], s ->
          intersection[P[OMEGA], invar[SUCC], complement[P[P[complement[singleton[x]]]]]]}]

Out[44]= or[member[x, U[intersection[OMEGA, fix[BIGCUP]]]], not[member[x, OMEGA]]] == True
```

The variable **x** is eliminated:

```
In[45]:= Map[assert[forall[x, #]] &, %]

Out[45]= subclass[OMEGA, U[intersection[OMEGA, fix[BIGCUP]]]] == True

In[46]:= subclass[OMEGA, U[intersection[OMEGA, fix[BIGCUP]]]] := True
```

This can be strengthened to an equation:

```
In[47]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> OMEGA, v -> U[intersection[OMEGA, fix[BIGCUP]]]}]

Out[47]= True == equal[OMEGA, U[intersection[OMEGA, fix[BIGCUP]]]]
```

The sum class of the class of limit ordinals is the class of all ordinals:

```
In[48]:= U[intersection[OMEGA, fix[BIGCUP]]] := OMEGA
```

It now follows immediately that the class of limit ordinals is a proper class:

```
In[49]:= SubstTest[member, U[x], V, x -> intersection[OMEGA, fix[BIGCUP]]] // Reverse

Out[49]= member[intersection[OMEGA, fix[BIGCUP]], V] == False

In[50]:= member[intersection[OMEGA, fix[BIGCUP]], V] := False
```

A more general rewrite rule can be derived:

```
In[51]:= member[intersection[OMEGA, fix[BIGCUP]], x] // AssertTest

Out[51]= member[intersection[OMEGA, fix[BIGCUP]], x] == False

In[52]:= member[intersection[OMEGA, fix[BIGCUP]], x_] := False
```

To summarize: the class of all ordinals is the union of two disjoint subclasses, the class of successor ordinals and the class of limit ordinals, both of which are proper classes.