the successor ordinals form a proper class

Johan G. F. Belinfante
2003 June 3

```
In[1]:= << goedel52.r93; << tools.m

        :Package Title: goedel52.r93       2003 June 1 at 11:11 a.m.

        It is now:  2003 Jun 3 at 9:57

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 May 31

        weightlimit = 40
```

■ introduction

This notebook contains an elementary derivation of the fact that the successor ordinals form a proper class. An automated proof of this theorem was obtained in 1999 using McCune's program **Otter**. It is Corollary **ON−SC−4B** in my paper on computer proofs in ordinal number theory.

> **Johan G. F. Belinfante, On Computer − Assisted Proofs in Ordinal Number Theory,
> Journal of Automated Reasoning, volume 22, pages 341 − 378 (1999).**

Some of the same ideas in this proof can be used to derive the fact that the limit ordinals also form a proper class. The latter theorem is slightly more involved in that it requires the use of iteration, and for this reason that derivation will be posted in a separate notebook. A key fact that is used in both derivations is the sum class axiom: if **x** is a set, then so is the sum class **U[x]**.

```
In[2]:= class[y, exists[z, and[member[y, z], member[z, x]]]]

Out[2]= U[x]
```

The power class axiom implies that if **x** is a set, then so is the power class **P[x]**. The sum class and the power class are related to each other:

```
In[3]:= U[P[x]]

Out[3]= x

In[4]:= subclass[x, P[U[x]]]

Out[4]= True
```

The converse of the sum class axiom therefore also holds: **U[x]** is a set if and only if **x** is a set. The **GOEDEL** program contains a rewrite rule based on this fact:

```
In[5]:= member[U[x], V]

Out[5]= member[x, V]
```

The successor function **SUCC** takes each set **x** to its successor **succ[x] = union[x, singleton[x]]**. The class of successor ordinals is the intersection of the class **OMEGA** of all ordinals and the class **range[SUCC]** of all successors. This intersection can be rewritten as the class of all non−limit ordinals:

```
In[6]:= ImageComp[SUCC, inverse[SUCC], OMEGA]

Out[6]= intersection[OMEGA, range[SUCC]] == intersection[OMEGA, complement[fix[BIGCUP]]]

In[7]:= intersection[OMEGA, range[SUCC]] := intersection[OMEGA, complement[fix[BIGCUP]]]
```

Adding this rewrite rule helps to increase analogies with similar rules for the set **omega** of natural numbers:

```
In[8]:= intersection[omega, range[SUCC]]

Out[8]= intersection[omega, complement[singleton[0]]]
```

■ the successor ordinals form a proper class

It is convenient to introduce a temporary abbreviation for the restriction of a relation **x** to the class **OMEGA** of ordinals:

```
In[9]:= r[x_] := restrict[x, OMEGA, OMEGA]
```

The composite of the restriction of the inverse of the membership relation **E** and the restriction of the successor function **SUCC** is the restriction of the inverse of the subclass relation **S**:

```
In[10]:= composite[r[inverse[E]], r[SUCC]] == r[inverse[S]]

Out[10]= True
```

From this, one readily deduces:

```
In[11]:= ImageComp[r[inverse[E]], r[SUCC], OMEGA] // Reverse

Out[11]= U[intersection[OMEGA, complement[fix[BIGCUP]]]] == OMEGA
```

This fact can be made into a rewrite rule:

```
In[12]:= U[intersection[OMEGA, complement[fix[BIGCUP]]]] := OMEGA
```

The fact that **OMEGA** is a proper class now implies that the class of successor ordinals is a proper class:

```
In[13]:= SubstTest[member, U[x], V, x -> intersection[OMEGA, complement[fix[BIGCUP]]]] // Reverse

Out[13]= member[intersection[OMEGA, complement[fix[BIGCUP]]], V] == False
```

This could be made into a rewrite rule, too. We do so as a temporary measure:

```
In[14]:= member[intersection[OMEGA, complement[fix[BIGCUP]]], V] := False
```

There is however a more general rewrite rule which follows immediately from the above rule, namely:

```
In[15]:= member[intersection[OMEGA, complement[fix[BIGCUP]]], x] // AssertTest

Out[15]= member[intersection[OMEGA, complement[fix[BIGCUP]]], x] == False
```

This more general rule will be made permanent, and the temporary rule discarded.

```
In[16]:= member[intersection[OMEGA, complement[fix[BIGCUP]]], x_] := False
```