Zermelo numbers are ordinals

Johan G. F. Belinfante
2010 June 11

```
In[1]:= SetDirectory["l:"]; << goedel.10jun10a; << tools.m

        :Package Title: goedel.10jun10a              2010 June 10 at 5:05 p.m.

        It is now:  2010 Jun 11 at 14:35

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 February 26

        weightlimit = 40
```

summary

In an unpublished notebook, Ernst Zermelo in 1915 attempted to characterize ordinals as sets satisfying the conditions **Uclosure[x] ⊂ succ[x]** and **image[SUCC, x]⊂ succ[x]**. The temporary abbreviation **ZERMELO** will be used for the class of these **Zermelo numbers**.

```
In[2]:= ZERMELO = intersection[fix[composite[inverse[SUCC], S, UCLOSURE]],
            fix[composite[inverse[SUCC], S, IMAGE[SUCC]]]];
```

In this notebook it is shown that **ZERMELO = OMEGA**; that is, Ernst Zermelo's characterization of ordinals is equivalent to John Isbell's, which is the definition used in the **GOEDEL** program, as well as the author's earlier work using William McCune's automated reasoning program **Otter**. This equivalence does not require assuming the axiom of regularity.

```
In[3]:= "Johan G. F. Belinfante, On Computer-Assisted Proofs in Ordinal Number
          Theory, Journal of Automated Reasoning, vol. 22(1999), pp. 341-378.";
```

The following facts which were proved in the notebooks **ZER-ON-1.NB** and **ZER-ON-2.NB** will be used.

```
In[4]:= subclass[OMEGA, ZERMELO]

Out[4]= True

In[5]:= subclass[ZERMELO, REGULAR]

Out[5]= True

In[6]:= subclass[ZERMELO, FULL]

Out[6]= True
```

core[Ω, x]

Rewrite rules about **core[Ω, x]** are derived in this section.

Theorem. A general condition for membership in **core[x, y]**.

```
In[7]:= SubstTest[implies, and[member[u, v], member[v, t]],
          member[u, U[t]], t → intersection[x, P[y]]] // Reverse

Out[7]= or[member[u, core[x, y]], not[member[u, v]],
          not[member[v, x]], not[subclass[v, y]]] == True

In[8]:= or[member[u_, core[x_, y_]], not[member[u_, v_]],
          not[member[v_, x_]], not[subclass[v_, y_]]] := True
```

Theorem. The class **core[Ω, x]** does not belong to itself.

```
In[9]:= Map[not,
         ((implies[or[equal[t, OMEGA], member[t, OMEGA]], not[member[t, t]]] // NotNotTest) /.
           t → core[OMEGA, x])]

Out[9]= member[core[OMEGA, x], core[OMEGA, x]] == False

In[10]:= member[core[OMEGA, x_], core[OMEGA, x_]] := False
```

Corollary.

```
In[11]:= SubstTest[implies, and[member[u, v], member[v, t], subclass[v, x]],
           member[u, core[t, x]], {t → OMEGA, u → core[OMEGA, x], v → x}] // Reverse

Out[11]= or[not[member[x, OMEGA]], not[member[core[OMEGA, x], x]]] == True

In[12]:= or[not[member[x_, OMEGA]], not[member[core[OMEGA, x_], x_]]] := True
```

Corollary.

```
In[13]:= Map[not, SubstTest[implies,
             and[member[u, v], member[v, t], subclass[v, w]], member[u, core[t, w]],
             {t → OMEGA, u → core[OMEGA, w], v → succ[core[OMEGA, w]]}] // Reverse] /. w → setpart[x]

Out[13]= member[core[OMEGA, setpart[x]], setpart[x]] == False

In[14]:= member[core[OMEGA, setpart[x_]], setpart[x_]] := False
```

The **setpart** wrapper can be removed.

Theorem.

```
In[15]:= Map[implies[member[x, y], #] &, SubstTest[implies,
           equal[x, setpart[t]], not[member[core[OMEGA, x], x]], t → x]] // Reverse

Out[15]= or[not[member[x, y]], not[member[core[OMEGA, x], x]]] == True

In[16]:= or[not[member[x_, y_]], not[member[core[OMEGA, x_], x_]]] := True
```

---

a core lemma for almost Uclosed sets

In this section a key lemma is derived that is needed to allow **core[Ω, x]** to be used to derive properties of sets satisfying the almost Uclosed condition **Uclosure[x] ⊂ succ[x]**.

Lemma.

```
In[17]:= SubstTest[implies, member[t, P[w]], member[U[t], Uclosure[w]],
           {t → core[y, setpart[x]], w → setpart[x]}] // Reverse

Out[17]= equal[core[setpart[x], U[core[y, setpart[x]]]], U[core[y, setpart[x]]]] == True

In[18]:= core[setpart[x_], U[core[y_, setpart[x_]]]] := U[core[y, setpart[x]]]
```

Lemma.

```
In[19]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u → U[core[y, setpart[x]]],
            v → Uclosure[setpart[x]], w → succ[setpart[x]]}] // Reverse

Out[19]= or[equal[setpart[x], U[core[y, setpart[x]]]],
           member[U[core[y, setpart[x]]], setpart[x]],
           not[subclass[Uclosure[setpart[x]], succ[setpart[x]]]]] == True

In[20]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The following theorem is obtained by eliminating the **setpart** wrapper from the lemma.

Theorem.

```
In[21]:= Map[implies[member[x, z], #] &,
           SubstTest[implies, equal[x, setpart[t]], or[equal[x, U[core[y, x]]],
             member[U[core[y, x]], x], not[subclass[Uclosure[x], succ[x]]]], t → x]] // Reverse

Out[21]= or[equal[x, U[core[y, x]]], member[U[core[y, x]], x],
           not[member[x, z]], not[subclass[Uclosure[x], succ[x]]]] == True

In[22]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

clearing the equality flag

Although equality substitution is generally convenient, it does sometimes slow down derivations when many steps are involved. For this reason the **GOEDEL** program has only a restricted form of equality substitution. One can easily get around these restrictions by introducing a temporary variable **t = core[Ω, x]** to identify how the substitutions should be carried out, but the price for doing so is that this definition must then be added as an extra literal, complicating the reasoning. The **equality** flag will be cleared in this notebook.

```
In[23]:= equality = False;
```

To compensate for this, three lemmas are derived in this section that would not otherwise be needed.

Temporary lemma. (Needed only because **equality** flag is cleared.)

```
In[24]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → equal[x, U[y]], p2 → member[y, OMEGA],
            p3 → member[U[y], OMEGA], p4 → member[x, OMEGA]}]] // Reverse
Out[24]= or[member[x, OMEGA], not[equal[x, U[y]]], not[member[y, OMEGA]]] == True

In[25]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Temporary lemma. (Needed only because **equality** flag is cleared.)

```
In[26]:= SubstTest[implies, and[equal[u, v], member[v, w]],
           member[u, w], {u → x, v → succ[U[y]], w → OMEGA}] // Reverse // MapNotNot
Out[26]= or[member[x, OMEGA], not[equal[x, succ[U[y]]]], not[member[U[y], OMEGA]]] == True

In[27]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Temporary lemma. (Needed only because **equality** flag is cleared.)

```
In[28]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 -> equal[x, succ[U[y]]], p2 -> member[y, OMEGA],
            p3 → member[U[y], OMEGA], p4 → member[x, OMEGA]}]] // Reverse
Out[28]= or[member[x, OMEGA], not[equal[x, succ[U[y]]]], not[member[y, OMEGA]]] == True

In[29]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Zermelo numbers are ordinals

The proof that Zermelo numbers are ordinals is presented in this section. Because the proof involves ten literals, and several clauses that involve three literals, the author was not able to reduce the overall execution time below one minute without breaking up the reasoning. Several lemmas are introduced, each of which eliminate one literal at a time.

Lemma 1. Eliminate **p3**.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p2, p3], q],
           implies[p1, or[p3, p4]], not[implies[p1, or[q, p4]]],
           {q → member[x, OMEGA], p1 → member[x, ZERMELO], p2 → member[core[OMEGA, x], OMEGA],
            p3 → equal[x, U[core[OMEGA, x]]], p4 → member[U[core[OMEGA, x]], x]}]] // Reverse

Out[30]= or[member[x, OMEGA], member[U[core[OMEGA, x]], x], not[member[x, V]],
           not[subclass[image[SUCC, x], succ[x]]], not[subclass[Uclosure[x], succ[x]]]] == True

In[31]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma 2. Eliminate **p7**.

```
In[32]:= Map[not,
           SubstTest[and, implies[p1, p2], implies[p1, or[q, p4]], implies[and[p2, p7], q],
            implies[and[p1, p4], or[p7, p8]], not[implies[p1, or[q, p8]]],
            {q → member[x, OMEGA], p1 → member[x, ZERMELO], p2 → member[core[OMEGA, x], OMEGA],
             p4 → member[U[core[OMEGA, x]], x], p7 → equal[x, succ[U[core[OMEGA, x]]]],
             p8 → member[succ[U[core[OMEGA, x]]], x]}]] // Reverse

Out[32]= or[member[x, OMEGA], member[succ[U[core[OMEGA, x]]], x], not[member[x, V]],
           not[subclass[image[SUCC, x], succ[x]]], not[subclass[Uclosure[x], succ[x]]]] == True

In[33]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. Eliminate **p2**.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, or[p5, p6]],
            not[implies[p1, or[p5, p6]]], {q → member[x, OMEGA], p1 → member[x, ZERMELO],
             p2 → member[core[OMEGA, x], OMEGA], p5 → equal[core[OMEGA, x], U[core[OMEGA, x]]],
             p6 → equal[core[OMEGA, x], succ[U[core[OMEGA, x]]]]}]] // Reverse

Out[34]= or[equal[core[OMEGA, x], succ[U[core[OMEGA, x]]]],
           equal[core[OMEGA, x], U[core[OMEGA, x]]], not[member[x, V]],
           not[subclass[image[SUCC, x], succ[x]]], not[subclass[Uclosure[x], succ[x]]]] == True

In[35]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. Eliminate **p8**.

```
In[36]:= Map[not, SubstTest[and, implies[p1, p8], implies[and[p6, p8], p9],
            not[implies[and[p1, p6], p9]], {p1 → and[member[x, ZERMELO], not[member[x, OMEGA]]],
             p6 → equal[core[OMEGA, x], succ[U[core[OMEGA, x]]]],
             p8 → member[succ[U[core[OMEGA, x]]], x], p9 → member[core[OMEGA, x], x]}]] // Reverse

Out[36]= or[member[x, OMEGA], member[core[OMEGA, x], x],
           not[equal[core[OMEGA, x], succ[U[core[OMEGA, x]]]]], not[member[x, V]],
           not[subclass[image[SUCC, x], succ[x]]], not[subclass[Uclosure[x], succ[x]]]] == True

In[37]:= (% /. x → x_) /. Equal→ SetDelayed
```

Main Theorem. Zermelo numbers are ordinals.

```
In[38]:= Map[implies[member[x, y], not[#]] &, SubstTest[and, implies[p1, p4],
           implies[p1, or[p5, p6]], implies[and[p4, p5], p9], implies[and[p1, p6], p9],
           implies[p1, not[p9]], p1, {p1 → and[member[x, ZERMELO], not[member[x, OMEGA]]],
            p4 → member[U[core[OMEGA, x]], x], p5 → equal[core[OMEGA, x], U[core[OMEGA, x]]],
            p6 → equal[core[OMEGA, x], succ[U[core[OMEGA, x]]]],
            p9 → member[core[OMEGA, x], x]}]] // Reverse

Out[38]= or[member[x, OMEGA], not[member[x, y]],
           not[subclass[image[SUCC, x], succ[x]]], not[subclass[Uclosure[x], succ[x]]]] == True

In[39]:= or[member[x_, OMEGA], not[member[x_, y_]], not[subclass[image[SUCC, x_], succ[x_]]],
           not[subclass[Uclosure[x_], succ[x_]]]] := True
```

A variable-free restatement will next be derived.

Lemma.

```
In[40]:= Map[equal[V, #] &,
          SubstTest[class, x, implies[member[x, u], member[x, v]], {u → ZERMELO, v → OMEGA}]]

Out[40]= subclass[intersection[fix[composite[inverse[SUCC], S, UCLOSURE]],
             fix[composite[inverse[SUCC], S, IMAGE[SUCC]]]], OMEGA] == True

In[41]:= % /. Equal → SetDelayed
```

Theorem. Zermelo's characterization of ordinals.

```
In[42]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → ZERMELO, v → OMEGA}]

Out[42]= equal[OMEGA, intersection[fix[composite[inverse[SUCC], S, UCLOSURE]],
             fix[composite[inverse[SUCC], S, IMAGE[SUCC]]]]] == True

In[43]:= intersection[fix[composite[inverse[SUCC], S, UCLOSURE]],
           fix[composite[inverse[SUCC], S, IMAGE[SUCC]]]] := OMEGA
```

Mycielski's characterization

Jan Mycielski in 2006 proposed (on page 208) to define the class **OMEGA** of ordinal numbers as the smallest class **x** that is Uclosed and invariant under the successor function **SUCC**.

```
In[44]:= "Jan Mycielski, A System of Axioms of Set Theory
           for the Rationalists, Notices of the American Mathematical
           Society, Volume 53, Number 2, pages 206–213, February 2006.";
```

In the posted notebook **UCL-ON.NB** dated 2007 September 6, the author derived these facts:

```
In[45]:= equal[Uclosure[OMEGA], OMEGA]

Out[45]= True
```

```
In[46]:= invariant[SUCC, OMEGA]

Out[46]= True

In[47]:= implies[and[equal[Uclosure[x], x], invariant[SUCC, x]], subclass[OMEGA, x]]

Out[47]= True
```

This characterization of **OMEGA** cannot however be made into a definition in the context of the **GOEDEL** program.