the next aleph

Johan G. F. Belinfante
2010 November 19

```
In[1]:= SetDirectory["l:"]; << goedel.10nov19b
        :Package Title: goedel.10nov19b               2010 November 19 at 1:35 p.m.
        It is now:  2010 Nov 19 at 15:20
        Loading Simplification Rules
        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3
        weightlimit = 40
```

summary

Every natural number is a cardinal, and thus for finite cardinals, then next larger one is simply its successor. For infinite cardinals the Hartogs number plays the role of the successor. Recall that the Hartogs number **hartogs[x]** of any class **x** is defined as the class of all ordinals that are equipollent to some subset of **x**.

```
In[2]:= hartogs[x]
Out[2]= intersection[OMEGA, image[Q, P[x]]]
```

The **ℵ = ALEPH** function enumerates the infinite cardinals in increasing order. In this notebook it is shown that the Hartogs number of any aleph is the next aleph. A variable-free expression of this is: **HARTOGS ∘ ℵ = ℵ ∘ SUCC**.

a hull rule

A general theorem about hulls is derived in this section.

Lemma.

```
In[3]:= member[succ[ord[x]], image[inverse[S], ord[x]]] // AssertTest
Out[3]= member[succ[ord[x]], image[inverse[S], ord[x]]] == False
In[4]:= member[succ[ord[x_]], image[inverse[S], ord[x_]]] := False
```

Lemma.

```
In[5]:= SubstTest[and, subclass[u, v], subclass[v, u],
         {u -> intersection[OMEGA, complement[P[complement[set[ord[x]]]]]],
          v -> intersection[OMEGA, image[S, set[succ[ord[x]]]]]}]

Out[5]= equal[intersection[OMEGA, complement[P[complement[set[ord[x]]]]]],
          intersection[OMEGA, image[S, set[succ[ord[x]]]]]] == True

In[6]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem.

```
In[7]:= SubstTest[implies, equal[u, v], equal[A[intersection[x, u]], A[intersection[x, v]]],
          {u -> intersection[OMEGA, complement[P[complement[set[ord[y]]]]]],
           v -> intersection[OMEGA, image[S, set[succ[ord[y]]]]]}] // Reverse

Out[7]= equal[hull[intersection[OMEGA, x], set[ord[y]]],
          hull[intersection[OMEGA, x], succ[ord[y]]]] == True

In[8]:= equal[hull[intersection[OMEGA, x_], set[ord[y_]]],
          hull[intersection[OMEGA, x_], succ[ord[y_]]]] := True
```

---

an inclusion in one direction

Lemma. A corollary of the inclusion **image[ℵ, ord[x]] ⊂ APPLY[ℵ, ord[x]]**.

```
In[9]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u → image[ALEPH, ord[x]],
           v → APPLY[ALEPH, ord[x]], w -> succ[APPLY[ALEPH, ord[x]]]}] // Reverse

Out[9]= subclass[image[ALEPH, ord[x]], succ[APPLY[ALEPH, ord[x]]]] == True

In[10]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. (Successors are unions, and images preserve unions.)

```
In[11]:= Map[subclass[image[ALEPH, succ[ord[x]]], #] &,
          SubstTest[image, ALEPH, union[u, v], {u -> ord[x], v → set[ord[x]]}]]

Out[11]= subclass[image[ALEPH, succ[ord[x]]],
           union[image[ALEPH, ord[x]], set[APPLY[ALEPH, ord[x]]]]] == True

In[12]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem.

```
In[13]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → image[ALEPH, succ[ord[x]]],
            v → union[image[ALEPH, ord[x]], set[APPLY[ALEPH, ord[x]]]],
            w -> succ[APPLY[ALEPH, ord[x]]]}] // Reverse

Out[13]= subclass[image[ALEPH, succ[ord[x]]], succ[APPLY[ALEPH, ord[x]]]] == True
```

```
In[14]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. Simplification rule.

```
In[15]:= SubstTest[tc, APPLY[enum[t], x], t → dif[fix[CARD], omega]] // Reverse

Out[15]= tc[APPLY[ALEPH, x]] == APPLY[ALEPH, x]

In[16]:= tc[APPLY[ALEPH, x_]] := APPLY[ALEPH, x]
```

Lemma.

```
In[17]:= SubstTest[subclass, tc[t], hull[OMEGA, t], t -> set[APPLY[ALEPH, ord[x]]]] // Reverse

Out[17]= subclass[APPLY[ALEPH, ord[x]], hull[OMEGA, set[APPLY[ALEPH, ord[x]]]]] == True

In[18]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (An inclusion in one direction.)

```
In[19]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → image[ALEPH, succ[ord[x]]], v → tc[set[APPLY[ALEPH, ord[x]]]],
            w → hull[OMEGA, set[APPLY[ALEPH, ord[x]]]]}] // Reverse

Out[19]= subclass[image[ALEPH, succ[ord[x]]], hull[OMEGA, set[APPLY[ALEPH, ord[x]]]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

---

inclusion in the opposite direction

Lemma.

```
In[21]:= member[image[ALEPH, succ[ord[x]]], image[inverse[S],
            intersection[OMEGA, P[complement[set[APPLY[ALEPH, ord[x]]]]]]]] // AssertTest

Out[21]= member[image[ALEPH, succ[ord[x]]], image[inverse[S],
            intersection[OMEGA, P[complement[set[APPLY[ALEPH, ord[x]]]]]]]] == False

In[22]:= (% /. x → x_) /. Equal → SetDelayed
```

The following temporary abbreviation was introduced by Anthony P. Morse in his book, ***A Theory of Sets***, page 74. It is often convenient in dealing with hulls since **hull[x, y] = A[x ⋂ sp[y]]**.

```
In[23]:= sp[x_] := image[S, set[x]]
```

Lemma.

```
In[24]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> intersection[OMEGA, sp[image[ALEPH, succ[ord[x]]]]],
            v -> intersection[OMEGA, sp[image[ALEPH, set[ord[x]]]]]}] // MapNotNot

Out[24]= equal[intersection[OMEGA, complement[P[complement[set[APPLY[ALEPH, ord[x]]]]]]],
           intersection[OMEGA, image[S, set[image[ALEPH, succ[ord[x]]]]]]] == True

In[25]:= (% /. x_ → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[26]:= (SubstTest[implies, subclass[ord[y], domain[enum[t]]],
            equal[hull[intersection[OMEGA, t], image[enum[t], ord[y]]],
             APPLY[enum[t], ord[y]]], y → succ[ord[x]]]) /.
           t -> intersection[complement[omega], fix[CARD]] // Reverse

Out[26]= equal[APPLY[ALEPH, succ[ord[x]]],
           hull[intersection[complement[omega], fix[CARD]], image[ALEPH, succ[ord[x]]]]] == True

In[29]:= Equal[APPLY[ALEPH, succ[ord[x]]], hull[
            intersection[complement[omega], fix[CARD]], image[ALEPH, succ[ord[x]]]]] // Reverse

Out[29]= hull[intersection[complement[omega], fix[CARD]], image[ALEPH, succ[ord[x]]]] ==
          APPLY[ALEPH, succ[ord[x]]]

In[30]:= hull[intersection[complement[omega], fix[CARD]], image[ALEPH, succ[ord[x_]]]] :=
          APPLY[ALEPH, succ[ord[x]]]
```

Lemma.

```
In[31]:= SubstTest[implies, equal[u, v], equal[A[intersection[t, u]], A[intersection[t, v]]],
            {u -> intersection[OMEGA, complement[P[complement[set[APPLY[ALEPH, ord[x]]]]]]],
             v -> intersection[OMEGA, image[S, set[image[ALEPH, succ[ord[x]]]]]]}] /.
           t → dif[fix[CARD], omega] // Reverse

Out[31]= equal[APPLY[ALEPH, succ[ord[x]]],
           hull[intersection[complement[omega], fix[CARD]], set[APPLY[ALEPH, ord[x]]]]] == True

In[32]:= hull[intersection[complement[omega], fix[CARD]], set[APPLY[ALEPH, ord[x_]]]] :=
          APPLY[ALEPH, succ[ord[x]]]
```

Theorem.

```
In[33]:= SubstTest[implies, subclass[ord[x], domain[enum[t]]],
           equal[hull[intersection[OMEGA, t], image[enum[t], ord[x]]], APPLY[enum[t], ord[x]]],
           t -> intersection[complement[omega], fix[CARD]]] // Reverse

Out[33]= equal[APPLY[ALEPH, ord[x]],
           hull[intersection[complement[omega], fix[CARD]], image[ALEPH, ord[x]]]] == True

In[34]:= hull[intersection[complement[omega], fix[CARD]], image[ALEPH, ord[x_]]] :=
          APPLY[ALEPH, ord[x]]
```

simplifying the hull formulas

One can simplify the **hull** expressions encountered above, eliminating **complement[omega]**.

Lemma.

```
In[35]:= SubstTest[or, member[APPLY[ALEPH, t], APPLY[ALEPH, ord[x]]],
           not[member[t, ord[x]]], t → 0] // Reverse

Out[35]= or[equal[0, ord[x]], member[omega, APPLY[ALEPH, ord[x]]]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[37]:= equiv[member[omega, APPLY[ALEPH, ord[x]]], not[equal[0, ord[x]]]]

Out[37]= True

In[38]:= member[omega, APPLY[ALEPH, ord[x_]]] := not[equal[0, ord[x]]]
```

Lemma.

```
In[39]:= SubstTest[member, hartogs[ord[t]], OMEGA, t → APPLY[ALEPH, ord[x]]] // Reverse

Out[39]= member[intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]], OMEGA] ==
          True

In[40]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[41]:= SubstTest[image, Q, P[ord[t]], t → APPLY[ALEPH, ord[x]]] // Reverse

Out[41]= image[Q, P[APPLY[ALEPH, ord[x]]]] == image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]

In[42]:= image[Q, P[APPLY[ALEPH, ord[x_]]]] := image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]
```

Theorem.

```
In[43]:= SubstTest[implies,
           and[subclass[ord[u], ord[v]], member[ord[v], ord[w]]], member[ord[u], ord[w]],
           {u → omega, v → APPLY[ALEPH, ord[x]], w → hartogs[APPLY[ALEPH, ord[x]]]}] // Reverse

Out[43]= subclass[omega, APPLY[ALEPH, ord[x]]] == True

In[44]:= subclass[omega, APPLY[ALEPH, ord[x_]]] := True
```

Lemma.

```
In[45]:= SubstTest[member, hartogs[setpart[t]], V, t → APPLY[ALEPH, ord[x]]] // Reverse

Out[45]= member[intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]], V] == True

In[46]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[47]:= SubstTest[card, hartogs[ord[t]], t -> APPLY[ALEPH, ord[x]]] // Reverse

Out[47]= card[intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]]] ==
          intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]]

In[48]:= card[intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x_]]]]]] :=
          intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]]
```

Lemma.

```
In[49]:= SubstTest[and, member[ord[u], ord[v]], member[ord[v], ord[u]],
           {u → omega, v → hartogs[APPLY[ALEPH, ord[x]]]}] // Reverse

Out[49]= member[intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]], omega] ==
          False

In[50]:= member[intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x_]]]], omega] :=
          False
```

main theorem

Main Theorem. The Hartogs number of an aleph is the next aleph.

```
In[51]:= SubstTest[hull, intersection[u, v], hull[u, w], {u → fix[CARD],
             v → complement[omega], w → set[ord[t]]}] /. t → APPLY[ALEPH, ord[x]] // Reverse

Out[51]= intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x]]]]] ==
          APPLY[ALEPH, succ[ord[x]]]

In[52]:= intersection[OMEGA, image[inverse[CARD], P[APPLY[ALEPH, ord[x_]]]]] :=
          APPLY[ALEPH, succ[ord[x]]]
```

Restatement:

```
In[53]:= hartogs[APPLY[ALEPH, ord[x]]]

Out[53]= APPLY[ALEPH, succ[ord[x]]]
```

A variable-free statement will now be derived.

Lemma.

```
In[54]:= SubstTest[implies, equal[x, ord[t]], equal[image[composite[HARTOGS, ALEPH], set[x]],
           image[composite[ALEPH, SUCC], set[x]]], t → x] // Reverse

Out[54]= or[equal[set[APPLY[ALEPH, succ[x]]], set[
            intersection[OMEGA, image[Q, P[APPLY[ALEPH, x]]]]]], not[member[x, OMEGA]]] == True

In[55]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[56]:= SubstTest[reify, x, image[t, set[ord[x]]],
          t -> symdif[composite[HARTOGS, ALEPH], composite[ALEPH, SUCC]]]

Out[56]= union[composite[
           intersection[composite[ALEPH, SUCC], composite[complement[HARTOGS], ALEPH]],
           id[OMEGA]], composite[intersection[composite[HARTOGS, ALEPH],
            composite[complement[ALEPH], SUCC]], id[OMEGA]]] == 0

In[57]:= % /. Equal → SetDelayed
```

Main Theorem. The $\aleph$ function intertwines the successor function with the Hartogs function.

```
In[58]:= SubstTest[empty, composite[symdif[u, v], id[w]],
          {u -> composite[HARTOGS, ALEPH], v -> composite[ALEPH, SUCC], w → OMEGA}]

Out[58]= equal[composite[ALEPH, SUCC], composite[HARTOGS, ALEPH]] == True

In[59]:= composite[HARTOGS, ALEPH] := composite[ALEPH, SUCC]
```

Restatement.

```
In[60]:= intertwine[ALEPH, SUCC, HARTOGS]

Out[60]= True
```

---

serendipity

The following fact discovered in the course of this work was not needed here, but may be useful on another occasion.

Theorem. A simplification rule.

```
In[61]:= Map[equal[hull[OMEGA, tc[x]], #] &,
          SubstTest[hull, intersection[u, v], hull[v, x], {u → OMEGA, v → FULL}]]

Out[61]= equal[hull[OMEGA, x], hull[OMEGA, tc[x]]] == True

In[62]:= hull[OMEGA, tc[x_]] := hull[OMEGA, x]
```