Veblen's theorem illustrated

Johan G. F. Belinfante
2011 July 31

```
In[1]:= SetDirectory["l:"]; << goedel.11jul31a
```

```
:Package Title: goedel.11jul31a               2011 July 31 at 10:30 p.m.

Loading takes about eleven minutes, half that time due to builtin pauses.

It is now:  2011 Jul 31 at 22:30

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Jul 31 at 22:41
```

summary

A special case of a theorem due to Veblen is derived using the compound wrapper **pco[Uclosed[x]]** for a Uclosed proper class of ordinals.

```
In[2]:= "Oswald Veblen, Continuous increasing functions of finite and transfinite ordinals,
         Transactions of the American Mathematical Society, volume 9 (no. 3), pp. 280–292.";
```

As an example, it is shown that there is no ordinal can be an upper bound for the fixed points of the function **enum[fix[-CARD]]** that enumerates all cardinals in strictly increasing order.

temporary abbreviations

The following temporary abbreviations will help to save some writing. When there is no chance of confusion, the variables **x** and **y** will be suppressed in the informal comments accompanying the derivations. Thus, for example, **f[x]** will be further abbreviated to **f**.

```
In[8]:= f[x_] := enum[pco[Uclosure[x]]]
```

For any given ordinal **ord[y]**, the list **it[x, y] = iterate[f[x], {ord[y]}]** of iterates obtained by applying **f[x]** repeatedly is a monotone function.

```
In[9]:= it[x_, y_] := iterate[enum[pco[Uclosure[x]]], set[ord[y]]]
```

The range of **it[x, y]** is a hull expression that will be abbreviated as **h[x, y]**. This is a subclass of **Ω**.

```
In[10]:= h[x_, y_] := hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]
```

The goal will be to show that **U[h[x, y]] ∈ Ω** is a fixed point of **f[x]**.

The idea is to use *reductio ad absurdum*. The hypothetical statement that there are no fixed points of **f[x]** at or above **ord[y]** will be abbreviated as **hyp[x, y]**.

```
In[11]:= hyp[x_, y_] := not[member[ord[y], image[inverse[S], fix[enum[pco[Uclosure[x]]]]]]]
```

The strategy of the derivation to be presented below can be briefly summarized as follows. If **x ∈ U[h] = U[range[it]]** then there exists an ordinal **y ∈ h = range[it]** with **x ∈ y**. That is, there exists a natural number **z** such that **y = it(z)**. Then **f(x) ∈ f(y) = f(it(z)) = it(succ[z]) ⊂ U[h]**, which implies **f(x) ∈ U[h]**. Since **x ∈ U[h]** implies **f(x) ∈ U[h]**, it follows that **U[h]** is invariant under **f**. That is: **image[f, U[h]] ⊂ U[h]**. The sum classes therefore satisfy **f(U[h]) = U[image[f, U[h]]] ⊂ U[U[h]] =U[h]**. Finally, since it is known that **U[h] ⊂ f(U[h])**, one can conclude that **f(U[h]) = U[h]**.

derivation

Theorem. The iteration goes on forever. That is, the list of iterates is infinitely long.

```
In[12]:= SubstTest[implies,
           and[member[v, domain[funpart[u]]], invariant[funpart[u], domain[funpart[u]]]],
           equal[omega, domain[iterate[funpart[u], set[v]]]],
           {u → enum[pco[x]], v → ord[y]}] // Reverse
```

```
Out[12]= equal[omega, domain[iterate[enum[pco[x]], set[ord[y]]]]] == True
```

```
In[13]:= domain[iterate[enum[pco[x_]], set[ord[y_]]]] := omega
```

Lemma. The ordinal **it(z)** appearing on the list of **it** at any point **z ∈ ω** belongs to **h**.

```
In[17]:= Map[implies[equal[omega, domain[iterate[enum[pco[x]], set[ord[y]]]]], #] &,
           SubstTest[member, APPLY[funpart[t], nat[z]], range[funpart[t]],
            t → iterate[enum[pco[x]], set[ord[y]]]] // Reverse
```

```
Out[17]= member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
           hull[invar[enum[pco[x]]], set[ord[y]]]] == True
```

```
In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Corollary. **it(z) ⊂ U[h]**.

```
In[19]:= SubstTest[implies, member[u, v], subclass[u, U[v]],
           {u -> APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
            v -> hull[invar[enum[pco[x]]], set[ord[y]]]}] // Reverse
```

```
Out[19]= subclass[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
           U[hull[invar[enum[pco[x]]], set[ord[y]]]]] == True
```

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

Theorem. **y ⊂ U[h]**.

```
In[21]:= SubstTest[subclass, APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
           U[hull[invar[enum[pco[x]]], set[ord[y]]]], z → 0] // Reverse
```

```
Out[21]= subclass[ord[y], U[hull[invar[enum[pco[x]]], set[ord[y]]]]] == True
```

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

Corollary.

```
In[23]:= equal[intersection[ord[y], complement[U[hull[invar[enum[pco[x]]], set[ord[y]]]]]], 0]
```

```
Out[23]= True
```

```
In[24]:= intersection[complement[U[hull[invar[enum[pco[x_]]], set[ord[y_]]]]], ord[y_]] := 0
```

Theorem. **it(z) ∈ Ω**

```
In[25]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u -> APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
            v -> hull[invar[enum[pco[x]]], set[ord[y]]], w -> OMEGA}] // Reverse
```

```
Out[25]= member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], OMEGA] == True
```

```
In[26]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. Monotone property of **it**.

```
In[28]:= SubstTest[implies,
           and[member[u, domain[t]], subclass[P[t], monotone[S, S]], subclass[u, v]],
           subclass[APPLY[t, u], APPLY[t, v]], t → iterate[enum[pco[x]], set[ord[y]]]] // Reverse
```

```
Out[28]= or[not[member[u, omega]], not[subclass[u, v]],
           subclass[APPLY[iterate[enum[pco[x]], set[ord[y]]], u],
            APPLY[iterate[enum[pco[x]], set[ord[y]]], v]]] == True
```

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

Theorem.

```
In[32]:= Map[not, SubstTest[subclass, ord[y], ord[t],
           t -> APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]]
```

```
Out[32]= member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], ord[y]] == False
```

```
In[33]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[35]:= SubstTest[or, member[ord[u], image[inverse[S], fix[enum[pco[x]]]]],
           member[ord[v], ord[u]], not[member[ord[v], fix[enum[pco[x]]]]],
           {u → y, v → APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]}] // Reverse

Out[35]= or[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]], not[member[
           APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], fix[enum[pco[x]]]]]] == True

In[36]:= (% /. {x → x_, y → y_, z → z_}) /. Equal→ SetDelayed
```

Lemma.

```
In[37]:= SubstTest[implies, and[equal[APPLY[funpart[t], u], u], member[u, domain[funpart[t]]]],
           member[u, fix[funpart[t]]],
           {u -> APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], t → enum[pco[x]]}] // Reverse

Out[37]= or[member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], fix[enum[pco[x]]]],
           not[equal[APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]],
             APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]]] == True

In[38]:= (% /. {x → x_, y → y_, z → z_}) /. Equal→ SetDelayed
```

Theorem. **it(z)** is not a fixed point of **f**.

```
In[39]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 → not[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]]], p2 → not[
              member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], fix[enum[pco[x]]]]],
             p3 → not[equal[APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]],
                  nat[z]]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]]}] // Reverse

Out[39]= or[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]],
           not[equal[APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]],
             APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]]] == True

In[40]:= (% /. {x → x_, y → y_, z → z_}) /. Equal→ SetDelayed
```

Theorem. **f(it(z)) ⊂ U[h]**.

```
In[41]:= SubstTest[subclass, APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[t]],
           U[hull[invar[enum[pco[x]]], set[ord[y]]]], t → succ[nat[z]]] // Reverse

Out[41]= subclass[APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]],
           U[hull[invar[enum[pco[x]]], set[ord[y]]]]] == True

In[42]:= (% /. {x → x_, y → y_, z → z_}) /. Equal→ SetDelayed
```

Corollary.

```
In[43]:= Map[not, SubstTest[subclass, ord[u], ord[v],
           {u -> APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]],
            v -> U[hull[invar[enum[pco[x]]], set[ord[y]]]]}]]

Out[43]= member[U[hull[invar[enum[pco[x]]], set[ord[y]]]],
           APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]] == False
```

```
In[44]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem Strict monotonicity. **it(z) ∈ f(it(z))**.

```
In[45]:= Map[implies[not[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]]], #] &,
           SubstTest[and, subclass[ord[u], ord[v]], not[equal[ord[u], ord[v]]],
            {u → APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
             v → APPLY[iterate[enum[pco[x]], set[ord[y]]], succ[nat[z]]]}]]
```

```
Out[45]= or[member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
             APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]],
            member[ord[y], image[inverse[S], fix[enum[pco[x]]]]] == True
```

```
In[46]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. **it(z) ∈ U[h]**.

```
In[47]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
              {p1 -> not[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]]],
               p2 -> member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
                 APPLY[enum[pco[x]], APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]],
               p3 -> member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
                 U[hull[invar[enum[pco[x]]], set[ord[y]]]]]}]] // Reverse
```

```
Out[47]= or[member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
             U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
            member[ord[y], image[inverse[S], fix[enum[pco[x]]]]] == True
```

```
In[48]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Corollary. **y ∈ U[h]**.

```
In[49]:= SubstTest[or, member[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
              U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
            member[ord[y], image[inverse[S], fix[enum[pco[x]]]]], z → 0] // Reverse
```

```
Out[49]= or[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]],
            member[ord[y], U[hull[invar[enum[pco[x]]], set[ord[y]]]]] == True
```

```
In[50]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Eliminating the variable **z** yields **h ⊂ U[h]**.

Theorem. **hyp ⟹ h ⊂ U[h]**.

```
In[51]:= Map[empty, SubstTest[reify, z, intersection[
              complement[image[V, intersection[fix[enum[pco[x]]], image[S, set[ord[y]]]]]],
              dif[set[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]], t]],
            t -> U[hull[invar[enum[pco[x]]], set[ord[y]]]]]]
```

```
Out[51]= or[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]],
            subclass[hull[invar[enum[pco[x]]], set[ord[y]]],
             U[hull[invar[enum[pco[x]]], set[ord[y]]]]] == True
```

```
In[52]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[53]:= SubstTest[implies, and[subclass[t, OMEGA], subclass[t, U[t]]],
           equal[U[U[t]], U[t]], t -> hull[invar[enum[pco[x]]], set[ord[y]]]] // Reverse

Out[53]= or[equal[U[hull[invar[enum[pco[x]]], set[ord[y]]]],
            U[U[hull[invar[enum[pco[x]]], set[ord[y]]]]]],
           not[subclass[hull[invar[enum[pco[x]]], set[ord[y]]],
             U[hull[invar[enum[pco[x]]], set[ord[y]]]]]] == True
```

```
In[54]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. **hyp** $\Longrightarrow$ **UU[h] = U[h]**. (That is, **U[h]** is a limit ordinal.)

```
In[55]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → not[member[ord[y], image[inverse[S], fix[enum[pco[x]]]]]],
            p2 -> subclass[hull[invar[enum[pco[x]]], set[ord[y]]],
              U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
            p3 -> equal[U[hull[invar[enum[pco[x]]], set[ord[y]]]],
              U[U[hull[invar[enum[pco[x]]], set[ord[y]]]]]]}]] // Reverse

Out[55]= or[equal[U[hull[invar[enum[pco[x]]], set[ord[y]]]],
            U[U[hull[invar[enum[pco[x]]], set[ord[y]]]]]],
           member[ord[y], image[inverse[S], fix[enum[pco[x]]]]] == True
```

```
In[56]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The Uclosedness hypothesis is needed here:

Theorem. A continuity property of the enumerator of **pco[Uclosure[x]]**.

```
In[58]:= SubstTest[implies, subclass[ord[y], domain[enum[Uclosure[t]]]],
           equal[APPLY[enum[Uclosure[t]], U[ord[y]]], U[image[enum[Uclosure[t]], ord[y]]]],
           t → pco[Uclosure[x]]] // Reverse

Out[58]= equal[APPLY[enum[pco[Uclosure[x]]], U[ord[y]]],
           U[image[enum[pco[Uclosure[x]]], ord[y]]]] == True
```

```
In[59]:= U[image[enum[pco[Uclosure[x_]]], ord[y_]]] := APPLY[enum[pco[Uclosure[x]]], U[ord[y]]]
```

Theorem.

```
In[60]:= SubstTest[U, image[enum[pco[Uclosure[x]]], ord[z]],
           z -> U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]] // Reverse

Out[60]= U[image[enum[pco[Uclosure[x]]],
             U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]] ==
           APPLY[enum[pco[Uclosure[x]]], U[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]]]
```

```
In[61]:= U[image[enum[pco[Uclosure[x_]]],
           U[hull[invar[enum[pco[Uclosure[x_]]]], set[ord[y_]]]]]] :=
         APPLY[enum[pco[Uclosure[x]]], U[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]]]
```

executing the strategy

Lemma.

```
In[62]:= SubstTest[implies, and[member[u, v], member[v, OMEGA]], member[u, OMEGA],
           v → APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]] // Reverse

Out[62]= or[member[u, OMEGA],
           not[member[u, APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]]] == True

In[63]:= (% /. {u → u_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[64]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p1, p2], p3], implies[p3, p4], not[implies[p1, p4]],
            {p1 → member[t, APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]],
             p2 → member[t, OMEGA], p3 → member[APPLY[enum[pco[x]], t], APPLY[enum[pco[x]],
                APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]], p4 → member[
               APPLY[enum[pco[x]], t], U[hull[invar[enum[pco[x]]], set[ord[y]]]]]}]] // Reverse

Out[64]= or[member[APPLY[enum[pco[x]], t], U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
           not[member[t, APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]]]]] == True

In[65]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Eliminate **t**.)

```
In[66]:= Map[equal[V, #] &,
          SubstTest[class, t, implies[member[t, u], member[APPLY[funpart[w], t], v]],
           {u → APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
            v → U[hull[invar[enum[pco[x]]], set[ord[y]]]], w → enum[pco[x]]}]]

Out[66]= subclass[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]],
           image[inverse[enum[pco[x]]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]]] == True

In[67]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Eliminate **z**.)

```
In[68]:= Map[implies[empty[range[complement[#]]],
           invariant[enum[pco[x]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]]] &, SubstTest[
          reify, z, complement[dif[APPLY[iterate[enum[pco[x]], set[ord[y]]], nat[z]], t]],
          t → image[inverse[enum[pco[x]]],
            U[hull[invar[enum[pco[x]]], set[ord[y]]]]]] // Reverse

Out[68]= subclass[image[enum[pco[x]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
          U[hull[invar[enum[pco[x]]], set[ord[y]]]]] == True

In[69]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[70]:= Map[implies[subclass[APPLY[enum[pco[x]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
            U[hull[invar[enum[pco[x]]], set[ord[y]]]], #] &, SubstTest[and, subclass[u, v],
          subclass[v, u], {u → APPLY[enum[pco[x]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
           v → U[hull[invar[enum[pco[x]]], set[ord[y]]]]}]]

Out[70]= or[equal[APPLY[enum[pco[x]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
            U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
          not[subclass[APPLY[enum[pco[x]], U[hull[invar[enum[pco[x]]], set[ord[y]]]]],
            U[hull[invar[enum[pco[x]]], set[ord[y]]]]]] == True

In[71]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[73]:= SubstTest[implies, subclass[u, v], subclass[U[u], U[v]],
          {u → image[f[x], U[h[x, y]]], v → U[h[x, y]]}] // Reverse

Out[73]= subclass[APPLY[enum[pco[Uclosure[x]]],
            U[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]]],
          U[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]]] == True

In[74]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. **hyp $\Longrightarrow$ f(U[h]) = U[h]**.

```
In[75]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
            not[implies[p1, p4]], {p1 → hyp[x, y], p2 → equal[U[U[h[x, y]]], U[h[x, y]]],
             p3 → subclass[APPLY[f[x], U[h[x, y]]], U[h[x, y]]],
             p4 → equal[APPLY[f[x], U[h[x, y]]], U[h[x, y]]]}]] // Reverse

Out[75]= or[equal[
            APPLY[enum[pco[Uclosure[x]]], U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]],
            U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]],
          member[ord[y], image[inverse[S], fix[enum[pco[Uclosure[x]]]]]]] == True

In[76]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary.

```
In[77]:= SubstTest[implies, and[equal[APPLY[funpart[u], v], v], member[v, w]],
           member[v, fix[funpart[u]]], {u → f[x], v → U[h[x, y]], w → V}] // Reverse

Out[77]= or[member[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]],
           fix[enum[pco[Uclosure[x]]]]], not[equal[APPLY[enum[pco[Uclosure[x]]],
             U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]],
            U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]]]] == True

In[78]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[79]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → hyp[x, y], p2 → equal[APPLY[f[x], U[h[x, y]]], U[h[x, y]]],
            p3 → member[U[h[x, y]], fix[f[x]]]}]] // Reverse

Out[79]= or[member[ord[y], image[inverse[S], fix[enum[pco[Uclosure[x]]]]]],
           member[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]],
            fix[enum[pco[Uclosure[x]]]]] == True

In[80]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[81]:= SubstTest[implies, and[subclass[u, v], member[v, w]],
           member[u, image[inverse[S], w]], {u → ord[y], v → U[h[x, y]], w → fix[f[x]]}] // Reverse

Out[81]= or[member[ord[y], image[inverse[S], fix[enum[pco[Uclosure[x]]]]]],
           not[member[U[hull[invar[enum[pco[Uclosure[x]]]], set[ord[y]]]],
             fix[enum[pco[Uclosure[x]]]]]] == True

In[82]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[83]:= SubstTest[and, implies[p, q], or[p, q],
           {p → member[U[h[x, y]], fix[f[x]]], q → not[hyp[x, y]]}]

Out[83]= member[ord[y], image[inverse[S], fix[enum[pco[Uclosure[x]]]]]] == True

In[84]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. Eliminate **y**. (This inclusion can be improved upon.)

```
In[85]:= Map[or[subclass[OMEGA, image[inverse[S], fix[f[x]]]], empty[#]] &,
           SubstTest[reify, y, dif[set[ord[y]], t], t → image[inverse[S], fix[f[x]]]]]

Out[85]= subclass[OMEGA, image[inverse[S], fix[enum[pco[Uclosure[x]]]]]] == True

In[86]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[87]:= SubstTest[or, equal[OMEGA, U[t]], not[subclass[OMEGA, image[inverse[S], t]]],
           not[subclass[t, OMEGA]], t → fix[f[x]]] // Reverse

Out[87]= equal[OMEGA, U[fix[enum[pco[Uclosure[x]]]]]] == True

In[88]:= U[fix[enum[pco[Uclosure[x_]]]]] := OMEGA
```

Corollary. (Improvement of the lemma above.)

```
In[89]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → image[inverse[S], fix[enum[pco[Uclosure[x]]]]], v → P[OMEGA]}]

Out[89]= equal[image[inverse[S], fix[enum[pco[Uclosure[x]]]]], P[OMEGA]] == True

In[90]:= image[inverse[S], fix[enum[pco[Uclosure[x_]]]]] := P[OMEGA]
```

The final step is to eliminate the double wrapper **pco[Uclosure[x]]**.

Veblen's Theorem. If **x** is a proper Uclosed class of ordinals, then **U[fix[enum[x]]] = Ω**.

```
In[91]:= SubstTest[implies, equal[x, pco[Uclosure[t]]],
           equal[OMEGA, U[fix[enum[x]]]], t → x] // Reverse

Out[91]= or[equal[OMEGA, U[fix[enum[x]]]], member[x, V],
           not[equal[x, Uclosure[x]]], not[subclass[x, OMEGA]]] == True

In[92]:= or[equal[OMEGA, U[fix[enum[x_]]]], member[x_, V],
           not[equal[x_, Uclosure[x_]]], not[subclass[x_, OMEGA]]] := True
```

The following provides an example.

Corollary. Every ordinal is less than some fixed point of **enum[fix[CARD]]**.

```
In[93]:= SubstTest[or, equal[OMEGA, U[fix[enum[x]]]], member[x, V],
           not[equal[x, Uclosure[x]]], not[subclass[x, OMEGA]], x → fix[CARD]] // Reverse

Out[93]= equal[OMEGA, U[fix[enum[fix[CARD]]]]] == True

In[94]:= U[fix[enum[fix[CARD]]]] := OMEGA
```