enumerating all ordinals in a given class: part 4. recursion relation for enum[x]

Johan G. F. Belinfante
2010 October 7

```
In[1]:= SetDirectory["l:"]; << goedel.10oct06a

        :Package Title: goedel.10oct06a            2010 October 6 at 5:00 a.m.

        It is now:  2010 Oct 7 at 11:11

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

In this fourth notebook in a series about enumerating ordinals it is shown that the function **enum[x]** satisfies the same recursion inclusion that the partial enumerations satisfy. An equation expressing **partenum[x]** in terms of **enum[x]** is derived.

relation between total and partial enumerations

In this section some basic results are derived to facilitate deriving properties of the total enumeration function **enum[x]** from those of the partial enumerations.

Theorem. Every partial enumeration is a subset of the total enumeration.

```
In[2]:= SubstTest[implies, member[w, t], subclass[w, U[t]], t → partenum[x]] // Reverse

Out[2]= or[not[member[w, partenum[x]]], subclass[w, enum[x]]] == True

In[3]:= or[not[member[w_, partenum[x_]]], subclass[w_, enum[x_]]] := True
```

Corollary. Every partial enumeration is a restriction of the total enumeration.

```
In[4]:= SubstTest[implies, and[member[w, t], FUNCTION[U[t]]],
          equal[w, composite[U[t], id[domain[w]]]], t → partenum[x]] // Reverse

Out[4]= or[equal[w, composite[enum[x], id[domain[w]]]], not[member[w, partenum[x]]]] == True

In[5]:= or[equal[w_, composite[enum[x_], id[domain[w_]]]],
          not[member[w_, partenum[x_]]]] := True
```

recursion relation for enum[x]

In this section it is shown that **enum[x]** satisfies the same recursion relation as the partial enumerations do.

Lemma.

```
In[6]:= Map[implies[#, equal[w, composite[HULL[x], IMAGE[w], id[domain[w]]]]] &, SubstTest[
          equal, w, composite[funpart[t], id[domain[w]]], t → composite[HULL[x], IMAGE[w]]]]

Out[6]= or[equal[w, composite[HULL[x], IMAGE[w], id[domain[w]]]],
          not[subclass[w, composite[HULL[x], IMAGE[w]]]] == True

In[7]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Theorem. An equational form of the recursion inclusion for partial enumerations.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → member[w, partenum[x]],
            p2 -> subclass[w, composite[HULL[intersection[OMEGA, x]], IMAGE[w]]], p3 → equal[w,
              composite[HULL[intersection[OMEGA, x]], IMAGE[w], id[domain[w]]]]}]] // Reverse

Out[8]= or[equal[w, composite[HULL[intersection[OMEGA, x]], IMAGE[w], id[domain[w]]]],
          not[member[w, partenum[x]]] == True

In[9]:= or[equal[w_, composite[HULL[intersection[OMEGA, x_]], IMAGE[w_], id[domain[w_]]]],
          not[member[w_, partenum[x_]]]] := True
```

Lemma.

```
In[10]:= SubstTest[implies, equal[w, t], equal[IMAGE[w], IMAGE[t]],
           t -> composite[enum[x], id[domain[w]]]] // Reverse

Out[10]= or[equal[composite[IMAGE[enum[x]], IMAGE[id[domain[w]]]], IMAGE[w]],
           not[equal[w, composite[enum[x], id[domain[w]]]]] == True

In[11]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Corollary.

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 → member[w, partenum[x]], p2 → equal[w, composite[enum[x], id[domain[w]]]],
             p3 → equal[composite[IMAGE[enum[x]], IMAGE[id[domain[w]]]], IMAGE[w]]}]] // Reverse

Out[12]= or[equal[composite[IMAGE[enum[x]], IMAGE[id[domain[w]]]], IMAGE[w]],
           not[member[w, partenum[x]]]] == True

In[13]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma. The domain of any partial enumeration is full.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → member[w, partenum[x]],
            p2 → member[domain[w], OMEGA], p3 → full[domain[w]]}]] // Reverse

Out[14]= or[not[member[w, partenum[x]]], subclass[U[domain[w]], domain[w]]] == True

In[15]:= or[not[member[w_, partenum[x_]]], subclass[U[domain[w_]], domain[w_]]] := True
```

Lemma. An equation that allows one to eliminate a factor **IMAGE[id[x]]** when **x** is full.

```
In[16]:= SubstTest[implies, equal[u, v], equal[composite[t, u], composite[t, v]],
           {u -> composite[IMAGE[id[x]], id[x]], v -> id[x]}] // Reverse

Out[16]= or[equal[composite[t, id[x]], composite[t, IMAGE[id[x]], id[x]]],
           not[subclass[U[x], x]]] == True

In[17]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma. Every partial enumeration is a subclass of **HULL[Ω ⋂ x] ∘ IMAGE[enum[x]]**.

```
In[18]:= Map[not, SubstTest[and, implies[p1, p4], implies[and[p2, p3, p5], p6],
           not[implies[p1, p6]], {p1 → member[w, partenum[x]],
            p2 → equal[w, composite[HULL[intersection[OMEGA, x]], IMAGE[w], id[domain[w]]]],
            p3 → equal[composite[IMAGE[enum[x]], IMAGE[id[domain[w]]]], IMAGE[w]],
            p4 → full[domain[w]], p5 -> equal[composite[HULL[intersection[OMEGA, x]],
               IMAGE[enum[x]], IMAGE[id[domain[w]]], id[domain[w]]],
              composite[HULL[intersection[OMEGA, x]], IMAGE[enum[x]], id[domain[w]]]], p6 →
             subclass[w, composite[HULL[intersection[OMEGA, x]], IMAGE[enum[x]]]]}]] // Reverse

Out[18]= or[not[member[w, partenum[x]]],
           subclass[w, composite[HULL[intersection[OMEGA, x]], IMAGE[enum[x]]]]] == True

In[19]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Eliminating the variable **w** yields a recursion inclusion for **enum[x]**.

Theorem. Recursion inclusion for **enum[x]**.

```
In[20]:= Map[equal[V, #] &, SubstTest[class, w, implies[member[w, y], subclass[w, z]],
           {y → partenum[x], z -> composite[HULL[intersection[OMEGA, x]], IMAGE[enum[x]]]}]]

Out[20]= subclass[enum[x], composite[HULL[intersection[OMEGA, x]], IMAGE[enum[x]]]] == True

In[21]:= subclass[enum[x_], composite[HULL[intersection[OMEGA, x_]], IMAGE[enum[x_]]]] := True
```

The recursion inclusion can be formulated as an equation for **APPLY[enum[x], t]**.

Theorem. Recursion equation for **enum[x]**.

```
In[22]:= SubstTest[implies, and[member[t, domain[u]], subclass[u, funpart[v]]],
           equal[APPLY[u, t], APPLY[funpart[v], t]],
           {u → enum[x], v -> composite[HULL[intersection[OMEGA, x]], IMAGE[enum[x]]]}] // Reverse

Out[22]= or[equal[APPLY[enum[x], t], hull[intersection[OMEGA, x], image[enum[x], t]]],
           not[member[t, domain[enum[x]]]]] == True

In[23]:= or[equal[APPLY[enum[x_], t_], hull[intersection[OMEGA, x_], image[enum[x_], t_]]],
           not[member[t_, domain[enum[x_]]]]] := True
```

---

an equation for partenum[x]

In this section an equation is derived that expresses the class **partenum[x]** in terms of **enum[x]**.

Lemma. The intersection of the domain of **enum[x]** with an ordinal is an ordinal.

```
In[24]:= ((implies[and[subclass[t, OMEGA], full[t]], member[intersection[t, ord[y]], OMEGA]] //
             NotNotTest) /. t → domain[enum[x]]) // MapNotNot

Out[24]= member[intersection[domain[enum[x]], ord[y]], OMEGA] == True

In[25]:= member[intersection[domain[enum[x_]], ord[y_]], OMEGA] := True
```

Theorem. The restriction of **enum[x]** to any ordinal is a partial enumeration.

```
In[26]:= (Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
             not[implies[p1, p4]], {p1 → equal[w, composite[enum[x], id[ord[y]]]],
              p2 -> subclass[w, composite[HULL[intersection[OMEGA, x]], IMAGE[w]]],
              p3 → member[domain[w], OMEGA], p4 → member[w, partenum[x]]}]] /.
            w -> composite[enum[x], id[ord[y]]]) // Reverse

Out[26]= member[composite[enum[x], id[ord[y]]], partenum[x]] == True

In[27]:= member[composite[enum[x_], id[ord[y_]]], partenum[x_]] := True
```

Corollary. Restatement without the **ord** wrapper.

```
In[28]:= SubstTest[implies, equal[y, ord[t]],
           member[composite[enum[x], id[y]], partenum[x]], t → y] // Reverse

Out[28]= or[member[composite[enum[x], id[y]], partenum[x]], not[member[y, OMEGA]]] == True

In[29]:= or[member[composite[enum[x_], id[y_]], partenum[x_]], not[member[y_, OMEGA]]] := True
```

Eliminating the variable **y** yields the following inclusion.

Theorem. The class of partial enumerations contains all restrictions of **enum[x]** to ordinals.

```
In[30]:= Map[equal[V, #] &, SubstTest[class, y, implies[member[y, u], member[y, v]], {u → OMEGA,
           v -> image[inverse[IMAGE[composite[id[enum[x]], inverse[FIRST]]]], partenum[x]]}]]

Out[30]= subclass[image[IMAGE[composite[id[enum[x]], inverse[FIRST]]], OMEGA], partenum[x]] ==
          True

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

An inclusion in the opposite direction will now be derived.

Lemma.

```
In[32]:= SubstTest[implies, member[t, partenum[x]],
           member[domain[t], OMEGA], t → composite[enum[x], id[y]]] // Reverse

Out[32]= or[member[intersection[y, domain[enum[x]]], OMEGA],
           not[member[composite[enum[x], id[y]], partenum[x]]]] == True

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

Lemma.

```
In[34]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u],
            member[t, v]],
           {u -> image[inverse[IMAGE[composite[id[enum[x]], inverse[FIRST]]]], partenum[x]],
            v -> image[inverse[IMAGE[id[domain[enum[x]]]]], OMEGA]}]]

Out[34]= subclass[image[IMAGE[id[domain[enum[x]]]],
            image[inverse[IMAGE[composite[id[enum[x]], inverse[FIRST]]]],
             partenum[x]]], OMEGA] == True

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

Lemma. Every partial enumeration is a restriction of **enum[x]** to an ordinal.

```
In[36]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> IMAGE[composite[id[enum[x]], inverse[FIRST]]],
            u -> image[IMAGE[id[domain[enum[x]]]], image[inverse[IMAGE[
                 composite[id[enum[x]], inverse[FIRST]]]], partenum[x]]], v -> OMEGA}] // Reverse

Out[36]= subclass[partenum[x],
           image[IMAGE[composite[id[enum[x]], inverse[FIRST]]], OMEGA]] == True

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

Theorem. A formula expressing **partenum[x]** in terms of **enum[x]**.

```
In[38]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[IMAGE[composite[id[enum[x]], inverse[FIRST]]], OMEGA], v → partenum[x]}]

Out[38]= equal[image[IMAGE[composite[id[enum[x]], inverse[FIRST]]], OMEGA], partenum[x]] == True

In[39]:= image[IMAGE[composite[id[enum[x_]], inverse[FIRST]]], OMEGA] := partenum[x]
```

Each of the classes **partenum[x]** and **enum[x]** determines the other. The function **enum[x]** is the sum class of the class **partenum[x]** of partial enumerations, and the class **partenum[x]** is the class of all restrictions of **enum[x]** to ordinals.

Corollary. A simplification rule.

```
In[40]:= ImageComp[composite[id[enum[x]], inverse[FIRST]], inverse[E], OMEGA] // Reverse

Out[40]= composite[enum[x], id[OMEGA]] == enum[x]

In[41]:= composite[enum[x_], id[OMEGA]] := enum[x]
```

the class of domains of partial enumerations

It was shown in the second of this series of notebooks on enumerating ordinals that the class **image[IMAGE[FIRST], partenum[x]]** of the domains of partial enumerations is either an ordinal or the class of all ordinals. In this section it will be shown that this class is equal to the successor of the domain of **enum[x]**. To derive this result, a formula for **image[IMAGE[id[ord[x]]], Ω]** will be needed. This is the class of all intersections of ordinals with the particular ordinal **ord[x]**.

Lemma.

```
In[42]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IMAGE[id[ord[x]]], u → ord[x], v → OMEGA}] // Reverse

Out[42]= subclass[ord[x], image[IMAGE[id[ord[x]]], OMEGA]] == True

In[43]:= subclass[ord[x_], image[IMAGE[id[ord[x_]]], OMEGA]] := True
```

Lemma.

```
In[44]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IMAGE[id[ord[x]]], u → set[ord[x]], v → OMEGA}] // Reverse

Out[44]= member[ord[x], image[IMAGE[id[ord[x]]], OMEGA]] == True

In[45]:= member[ord[x_], image[IMAGE[id[ord[x_]]], OMEGA]] := True
```

Lemma.

```
In[46]:= SubstTest[implies, and[equal[t, intersection[ord[x], ord[y]]], member[t, ord[x]]],
           member[intersection[ord[x], ord[y]], ord[x]], t → ord[y]] // Reverse

Out[46]= or[member[intersection[ord[x], ord[y]], ord[x]], not[member[ord[y], ord[x]]]] == True

In[47]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. Result of eliminating the variable **y**.

```
In[48]:= Map[equal[V, #] &, SubstTest[class, t, member[ord[t], w],
           w -> image[inverse[IMAGE[id[ord[x]]]], succ[ord[x]]]]]
```

```
Out[48]= subclass[image[IMAGE[id[ord[x]]], OMEGA], succ[ord[x]]] == True
```

```
In[49]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. The class of all intersections of ordinals with **ord[x]** is its successor.

```
In[50]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[IMAGE[id[ord[x]]], OMEGA], v -> succ[ord[x]]}]
```

```
Out[50]= equal[image[IMAGE[id[ord[x]]], OMEGA], succ[ord[x]]] == True
```

```
In[51]:= image[IMAGE[id[ord[x_]]], OMEGA] := succ[ord[x]]
```

Eliminating the **ord** wrapper yields the following.

Theorem.

```
In[52]:= SubstTest[implies, equal[x, ord[t]],
           equal[image[IMAGE[id[x]], OMEGA], succ[x]], t → x] // Reverse
```

```
Out[52]= or[equal[image[IMAGE[id[x]], OMEGA], succ[x]], not[member[x, OMEGA]]] == True
```

```
In[53]:= or[equal[image[IMAGE[id[x_]], OMEGA], succ[x_]], not[member[x_, OMEGA]]] := True
```

In the following lemma, this result is applied to the domain of **enum[x]**.

Lemma.

```
In[54]:= Map[not, SubstTest[and, or[p1, p2], implies[p1, p3], implies[p2, p3], not[p3],
            {p1 → member[domain[enum[x]], OMEGA], p2 → equal[domain[enum[x]], OMEGA], p3 → equal[
              image[IMAGE[id[domain[enum[x]]]], OMEGA], succ[domain[enum[x]]]]}]] // Reverse
```

```
Out[54]= equal[image[IMAGE[id[domain[enum[x]]]], OMEGA], succ[domain[enum[x]]]] == True
```

```
In[55]:= image[IMAGE[id[domain[enum[x_]]]], OMEGA] := succ[domain[enum[x]]]
```

Theorem. The class of domains of all partial enumerations of the ordinals in a class **x** is the successor of the domain of **enum[x]**.

```
In[57]:= ImageComp[IMAGE[FIRST],
           IMAGE[composite[id[enum[x]], inverse[FIRST]]], OMEGA] // Reverse
```

```
Out[57]= image[IMAGE[FIRST], partenum[x]] == succ[domain[enum[x]]]
```

```
In[58]:= image[IMAGE[FIRST], partenum[x_]] := succ[domain[enum[x]]]
```

Comment. If **domain[enum[x]]** is the class of all ordinals, then so is **image[IMAGE[FIRST], partenum[x]]**, because **Ω** is its own successor.

In[59]:= **succ[OMEGA]**

Out[59]= OMEGA