enum[Uclosure[x]]

Johan G. F. Belinfante
2010 November 28

```
In[1]:= SetDirectory["l:"]; << goedel.10nov27a

        :Package Title: goedel.10nov25a              2010 November 27 at 1:40 p.m.

        It is now:  2010 Nov 28 at 6:57

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

If **x** is closed under unions of arbitrary subsets, and if **y ⊂ domain[enum[x]]** is an ordinal, then the value of **enum[x]** at **U[y]** is the union of the values of **enum[x]** at all ordinals less than **y**. This result is applied to **enum[fix[CARD]]**.

general results

For the results derived in this section no restriction is placed on the classes **x** and **y**.

Lemma.

```
In[2]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> image[enum[x], y], v → APPLY[enum[x], y], w → P[APPLY[enum[x], y]]}] // Reverse

Out[2]= subclass[U[image[enum[x], y]], APPLY[enum[x], y]] == True

In[3]:= subclass[U[image[enum[x_], y_]], APPLY[enum[x_], y_]] := True
```

Theorem.

```
In[4]:= Map[not,
         ((implies[and[subclass[u, v], full[v], subclass[v, OMEGA]], not[member[v, u]]] //
             NotNotTest) /. {u → U[image[enum[x], y]], v → APPLY[enum[x], y]})]

Out[4]= member[APPLY[enum[x], y], U[image[enum[x], y]]] == False

In[5]:= member[APPLY[enum[x_], y_], U[image[enum[x_], y_]]] := False
```

enum[x] at successor ordinals

If **y ⊂ domain[enum[x]]** is a successor ordinal, then **APPLY[enum[x], U[y]] = U[image[enum[x], y]]**. In the following corollary, an **ord** wrapper is introduced.

Theorem.

```
In[6]:= SubstTest[implies,
          and[member[t, OMEGA], subclass[t, domain[enum[x]]], not[equal[U[t], t]]],
          equal[APPLY[enum[x], U[t]], U[image[enum[x], t]]], t → ord[y]] // Reverse

Out[6]= or[equal[APPLY[enum[x], U[ord[y]]], U[image[enum[x], ord[y]]]],
          equal[ord[y], U[ord[y]]], not[subclass[ord[y], domain[enum[x]]]]] == True

In[7]:= or[equal[APPLY[enum[x_], U[ord[y_]]], U[image[enum[x_], ord[y_]]]],
          equal[ord[y_], U[ord[y_]]], not[subclass[ord[y_], domain[enum[x_]]]]] := True
```

inverse[enum[x]]

The function **enum[x]** is one-to-one, so its inverse is also a function. Three results involving the inverse of **enum[x]** are derived in this section.

Theorem. If **y ∈ range[enum[x]]**, then **APPLY[inverse[enum[x]], y] ∈ domain[enum[x]]**.

```
In[8]:= Map[implies[#, member[APPLY[inverse[enum[x]], y], domain[enum[x]]]] &,
         SubstTest[member, APPLY[funpart[t], y], range[funpart[t]], t → inverse[enum[x]]]]

Out[8]= or[member[APPLY[inverse[enum[x]], y], domain[enum[x]]],
          not[member[y, OMEGA]], not[member[y, x]]] == True

In[9]:= or[member[APPLY[inverse[enum[x_]], y_], domain[enum[x_]]],
          not[member[y_, OMEGA]], not[member[y_, x_]]] := True
```

Theorem. If **y ∈ range[enum[x]]**, then **y = APPLY[enum[x], APPLY[inverse[enum[x]], y]]**.

```
In[10]:= Map[implies[member[y, range[enum[x]]], equal[y, #]] &,
          ApComp[enum[x], inverse[enum[x]], y]]

Out[10]= or[equal[y, APPLY[enum[x], APPLY[inverse[enum[x]], y]]],
           not[member[y, OMEGA]], not[member[y, x]]] == True

In[11]:= or[equal[y_, APPLY[enum[x_], APPLY[inverse[enum[x_]], y_]]],
           not[member[y_, OMEGA]], not[member[y_, x_]]] := True
```

Theorem. If **ord[y] = APPLY[inverse[enum[x]], z]**, then **z = APPLY[enum[x], ord[y]]**.

```
In[13]:= SubstTest[implies, and[equal[w, APPLY[inverse[oopart[t]], z]], member[w, V]],
           equal[z, APPLY[oopart[t], w]], {w → ord[y], t → enum[x]}] // Reverse

Out[13]= or[equal[z, APPLY[enum[x], ord[y]]],
           not[equal[APPLY[inverse[enum[x]], z], ord[y]]]] == True

In[14]:= or[equal[z_, APPLY[enum[x_], ord[y_]]],
           not[equal[APPLY[inverse[enum[x_]], z_], ord[y_]]]] := True
```

---

enumerating a Uclosed class

If a class is its own **Uclosure**, then it holds the union of any subset. In the following lemma, this observation is applied to the subset **image[enum[x], ord[y]** $\subset$ $\Omega \bigcap$ **x**.

Lemma.

```
In[15]:= SubstTest[implies, and[member[u, P[v]], equal[Uclosure[v], v]], member[U[u], v],
            {u → image[enum[x], ord[y]], v → intersection[OMEGA, x]}] // Reverse // MapNotNot

Out[15]= or[and[member[U[image[enum[x], ord[y]]], OMEGA], member[U[image[enum[x], ord[y]]], x]],
           not[equal[intersection[OMEGA, x], Uclosure[intersection[OMEGA, x]]]]] == True

In[16]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The temporary abbreviation **t = APPLY[inverse[enum[Uclosure[x]]],U[image[enum[Uclosure[x]],ord[y]]]]** is used in several results below.

Theorem. **t** $\in$ **domain[enum[Uclosure[x]]]**.

```
In[17]:= ((Map[not, SubstTest[and, implies[p1, p2],
                implies[and[p0, p2], p3], not[implies[and[p0, p1], p3]],
                {p0 → equal[t, APPLY[inverse[enum[u]], U[image[enum[u], ord[y]]]]],
                 p1 -> equal[intersection[OMEGA, u], Uclosure[intersection[OMEGA, u]]],
                 p2 -> member[U[image[enum[u], ord[y]]], range[enum[u]]],
                 p3 → member[t, domain[enum[u]]]}]] // Reverse) /.
             t -> APPLY[inverse[enum[u]], U[image[enum[u], ord[y]]]]) /. u → Uclosure[
             x]

Out[17]= member[APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]],
           domain[enum[Uclosure[x]]]] == True

In[18]:= member[APPLY[inverse[enum[Uclosure[x_]]], U[image[enum[Uclosure[x_]], ord[y_]]]],
           domain[enum[Uclosure[x_]]]] := True
```

Theorem. Simplification rule for **APPLY[enum[Uclosure[x]], t]**.

```
In[19]:= ((Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p0, p2], p3], not[implies[and[p0, p1], p3]],
            {p0 → equal[t, APPLY[inverse[enum[u]], U[image[enum[u], ord[y]]]]],
             p1 -> equal[intersection[OMEGA, u], Uclosure[intersection[OMEGA, u]]],
             p2 -> member[U[image[enum[u], ord[y]]], range[enum[u]]],
             p3 → equal[U[image[enum[u], ord[y]]], APPLY[enum[u], t]]}]] // Reverse) /.
          t -> APPLY[inverse[enum[u]], U[image[enum[u], ord[y]]]]) /. u → Uclosure[
          x]

Out[19]= equal[APPLY[enum[Uclosure[x]],
           APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]],
         U[image[enum[Uclosure[x]], ord[y]]]] == True

In[20]:= APPLY[enum[Uclosure[x_]],
           APPLY[inverse[enum[Uclosure[x_]]], U[image[enum[Uclosure[x_]], ord[y_]]]]] :=
         U[image[enum[Uclosure[x]], ord[y]]]
```

Theorem. A reformulation of the trichotomy principle for ordinals.

```
In[21]:= SubstTest[implies, equal[x, ord[t]],
           or[equal[x, ord[y]], member[x, ord[y]], member[ord[y], x]], t → x] // Reverse

Out[21]= or[equal[x, ord[y]], member[x, ord[y]], member[ord[y], x], not[member[x, OMEGA]]] == True

In[22]:= or[equal[ord[y_], x_], member[ord[y_], x_],
           member[x_, ord[y_]], not[member[x_, OMEGA]]] := True
```

To show that **t = ord[y]**, it therefore suffices to show that **t ∉ ord[y]** and **ord[y] ∉ t**. The next two results do this.

Theorem. If **U[ord[y]] = ord[y] ⊂ domain[enum[Uclosure[x]]]**, then **t ∉ ord[y]**.

```
In[23]:= Map[not, SubstTest[and, implies[p0, p1], implies[p0, p2], implies[p3, p4],
             implies[and[p2, p3, p4], p5], not[implies[and[p0, p3], p5]], {p0 → equal[t,
               APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]],
              p1 → member[t, domain[enum[Uclosure[x]]]],
              p2 → equal[APPLY[enum[Uclosure[x]], t], U[image[enum[Uclosure[x]], ord[y]]]],
              p3 → and[equal[ord[y], U[ord[y]]], subclass[ord[y], domain[enum[Uclosure[x]]]]],
              p4 → not[member[U[image[enum[Uclosure[x]], ord[y]]],
                 image[enum[Uclosure[x]], ord[y]]]], p5 → not[member[t, ord[y]]]}] // Reverse] /.
          t -> APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]

Out[23]= or[not[equal[ord[y], U[ord[y]]]],
           not[member[APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]],
             ord[y]]], not[subclass[ord[y], domain[enum[Uclosure[x]]]]]] == True

In[24]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. **ord[y] ∉ t**.

```
In[25]:= (SubstTest[and, implies[p0, p1], implies[p0, p2], implies[p2, p6],
            implies[and[p1, p6], p7], not[implies[p0, p7]], {p0 → equal[t,
              APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]],
             p1 → member[t, domain[enum[Uclosure[x]]]],
             p2 → equal[APPLY[enum[Uclosure[x]], t], U[image[enum[Uclosure[x]], ord[y]]]],
             p6 → not[member[APPLY[enum[Uclosure[x]], ord[y]], APPLY[enum[Uclosure[x]], t]]],
             p7 → not[member[ord[y], t]]}] // Reverse) /.
          t -> APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]

Out[25]= member[ord[y],
           APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]] == False

In[26]:= member[ord[y_],
           APPLY[inverse[enum[Uclosure[x_]]], U[image[enum[Uclosure[x_]], ord[y_]]]]] := False
```

The above two results are now combined to show that **t = ord[y]**. (This takes a while.)

Lemma. If **U[ord[y]] = ord[y] ⊂ domain[enum[Uclosure[x]]]**, then **t = ord[y]**.

```
In[27]:= Map[not, SubstTest[and, implies[p0, p1],
             implies[and[p0, p3], p5], implies[p0, p7], implies[p1, p8],
             implies[and[p5, p7, p8], p9], not[implies[and[p0, p3], p9]], {p0 → equal[t,
               APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]],
              p1 → member[t, domain[enum[Uclosure[x]]]],
              p3 → and[equal[ord[y], U[ord[y]]], subclass[ord[y], domain[enum[Uclosure[x]]]]],
              p5 → not[member[t, ord[y]]], p7 → not[member[ord[y], t]],
              p8 → member[t, OMEGA], p9 → equal[t, ord[y]]}] // Reverse] /.
          t -> APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]]

Out[27]= or[equal[APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]],
            ord[y]], not[equal[ord[y], U[ord[y]]]],
           not[subclass[ord[y], domain[enum[Uclosure[x]]]]]] == True

In[28]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If **ord[y] ⊂ domain[enum[Uclosure[x]]]**, then

APPLY[enum[Uclosure[x]], U[ord[y]]] = U[image[enum[Uclosure[x]], ord[y]]].

```
In[29]:= Map[not, SubstTest[and, implies[and[p1, not[p2]], p4],
            implies[and[p1, p2], p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 → subclass[ord[y], domain[enum[Uclosure[x]]]], p2 → equal[ord[y], U[ord[y]]],
             p3 → equal[APPLY[inverse[enum[Uclosure[x]]], U[image[enum[Uclosure[x]], ord[y]]]],
               ord[y]], p4 → equal[APPLY[enum[Uclosure[x]], U[ord[y]]],
               U[image[enum[Uclosure[x]], ord[y]]]]}] // Reverse

Out[29]= or[equal[APPLY[enum[Uclosure[x]], U[ord[y]]], U[image[enum[Uclosure[x]], ord[y]]]],
           not[subclass[ord[y], domain[enum[Uclosure[x]]]]]] == True

In[30]:= or[
           equal[APPLY[enum[Uclosure[x_]], U[ord[y_]]], U[image[enum[Uclosure[x_]], ord[y_]]]],
           not[subclass[ord[y_], domain[enum[Uclosure[x_]]]]]] := True
```

Corollary. (Eliminating the **ord** wrapper.)

```
In[33]:= SubstTest[implies, equal[y, ord[t]],
           or[equal[APPLY[enum[Uclosure[x]], U[y]], U[image[enum[Uclosure[x]], y]]],
            not[subclass[y, domain[enum[Uclosure[x]]]]]], t → y] // Reverse

Out[33]= or[equal[APPLY[enum[Uclosure[x]], U[y]], U[image[enum[Uclosure[x]], y]]],
           not[member[y, OMEGA]], not[subclass[y, domain[enum[Uclosure[x]]]]]] == True

In[36]:= or[equal[APPLY[enum[Uclosure[x_]], U[y_]], U[image[enum[Uclosure[x_]], y_]]],
           not[member[y_, OMEGA]], not[subclass[y_, domain[enum[Uclosure[x_]]]]]] := True
```

an application

Lemma. A simplification rule.

```
In[41]:= equal[domain[enum[fix[CARD]]], OMEGA]

Out[41]= True

In[42]:= domain[enum[fix[CARD]]] := OMEGA
```

Theorem. An application of the main theorem.

```
In[43]:= SubstTest[implies, subclass[ord[x], domain[enum[Uclosure[t]]]],
           equal[U[image[enum[Uclosure[t]], ord[x]]], APPLY[enum[Uclosure[t]], U[ord[x]]]],
           t → fix[CARD]] // Reverse

Out[43]= equal[APPLY[enum[fix[CARD]], U[ord[x]]], U[image[enum[fix[CARD]], ord[x]]]] == True

In[44]:= U[image[enum[fix[CARD]], ord[x_]]] := APPLY[enum[fix[CARD]], U[ord[x]]]
```

Lemma. A simplification rule.

```
In[45]:= equal[intersection[OMEGA, complement[image[inverse[BIGCUP], OMEGA]]], 0]

Out[45]= True

In[46]:= intersection[OMEGA, complement[image[inverse[BIGCUP], OMEGA]]] := 0
```

The variable **x** can be eliminated using **reify**.

Theorem. Variable-free formulation of a theorem about enumerating cardinals.

```
In[47]:= Map[composite[VERTSECT[#], id[OMEGA]] &,
           SubstTest[reify, x, U[image[t, ord[x]]], t -> enum[fix[CARD]]]]

Out[47]= composite[BIGCUP, IMAGE[enum[fix[CARD]]], id[OMEGA]] ==
           composite[enum[fix[CARD]], BIGCUP, id[OMEGA]]
```

```
In[48]:= composite[BIGCUP, IMAGE[enum[fix[CARD]]], id[OMEGA]] :=
          composite[enum[fix[CARD]], BIGCUP, id[OMEGA]]
```