enum[x] at a limit ordinal

Johan G. F. Belinfante
2010 November 27

```
In[1]:= SetDirectory["l:"]; << goedel.10nov25a

        :Package Title: goedel.10nov25a              2010 November 25 at 3:20 p.m.

        It is now:  2010 Nov 27 at 11:50

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

If **y ⊂ domain[enum[x]]** is a limit ordinal, then **image[enum[x], y] ⊂ U[image[enum[x], y]]**.

a general result

Every subclass **t ⊂ Ω** satisfies either **U[t] ∈ t** or **t ⊂ U[t]**. In particular, this is true for **t = image[enum[x], y]**. For this one does not need to know anything about the class **y**.

Theorem.

```
In[2]:= SubstTest[implies, subclass[t, OMEGA],
          or[member[U[t], t], subclass[t, U[t]]], t → image[enum[x], y]] // Reverse

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

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

Theorem.

```
In[4]:= SubstTest[implies, subclass[t, OMEGA],
          not[and[member[U[t], t], subclass[t, U[t]]]], t → image[enum[x], y]] // Reverse

Out[4]= or[not[member[U[image[enum[x], y]], image[enum[x], y]]],
          not[subclass[image[enum[x], y], U[image[enum[x], y]]]]] == True

In[5]:= or[not[member[U[image[enum[x_], y_]], image[enum[x_], y_]]],
          not[subclass[image[enum[x_], y_], U[image[enum[x_], y_]]]]] := True
```

sketch of the proof

In this notebook it will be shown that if **y ⊂ domain[enum[x]]** is a limit ordinal, then **t = image[enum[x], y]** satisfies the condition **t ⊂ U[t]**. The strategy will be to show that every member of **t** is also a member of **U[t]**. A typical member of **t** has the form **APPLY[enum[x], w]** where **w ∈ y**. If **y** is a limit ordinal, then **w ∈ succ[w] ∈ t**. Since **f = enum[x]** is a strictly monotone function, then **f(w) ∈ f([succ[w]) ∈ t**, and so **f[w] ∈ U[t]**. Eliminating the variable **w** yields the desired conclusion **t ⊂ U[t]**.

the case of a limit ordinal

Theorem. Limit ordinals are successor-invariant.

```
In[6]:= SubstTest[implies, member[t, OMEGA],
          or[member[succ[x], t], not[equal[t, U[t]]], not[member[x, t]]], t → ord[y]] // Reverse

Out[6]= or[member[succ[x], ord[y]],
          not[equal[ord[y], U[ord[y]]]], not[member[x, ord[y]]]] == True

In[7]:= or[member[succ[x_], ord[y_]],
          not[equal[ord[y_], U[ord[y_]]]], not[member[x_, ord[y_]]]] := True
```

explore

Lemma.

```
In[15]:= Map[implies[#, member[APPLY[enum[x], y], image[enum[x], z]]] &, SubstTest[member,
           APPLY[funpart[t], y], range[funpart[t]], t → composite[enum[x], id[z]]]]

Out[15]= or[member[APPLY[enum[x], y], image[enum[x], z]],
           not[member[y, z]], not[member[y, domain[enum[x]]]]] == True

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

Theorem.

```
In[20]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            not[implies[and[p1, p2], p4]], {p1 → member[y, z],
             p2 → subclass[z, domain[enum[x]]], p3 -> member[y, domain[enum[x]]],
             p4 -> member[APPLY[enum[x], y], image[enum[x], z]]}]] // Reverse

Out[20]= or[member[APPLY[enum[x], y], image[enum[x], z]],
           not[member[y, z]], not[subclass[z, domain[enum[x]]]]] == True

In[22]:= or[member[APPLY[enum[x_], y_], image[enum[x_], z_]],
           not[member[y_, z_]], not[subclass[z_, domain[enum[x_]]]]] := True
```

Lemma.

```
In[28]:= SubstTest[implies, and[member[t, v], member[v, domain[enum[x]]]],
           member[APPLY[enum[x], t], APPLY[enum[x], v]], v → succ[t]] // Reverse

Out[28]= or[member[APPLY[enum[x], t], APPLY[enum[x], succ[t]]],
           not[member[succ[t], domain[enum[x]]]]] == True

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

To speed up execution, the following proof step was left out in the following derivation: **implies[and[p6, p7], p8]**.

Theorem.

```
In[36]:= Map[not, SubstTest[and, implies[and[p2, p3], p4], implies[and[p1, p4], p5],
            implies[p5, p6], implies[and[p1, p4], p7], not[implies[and[p1, p2, p3], p8]],
            {p1 → subclass[ord[y], domain[enum[x]]], p2 → equal[U[ord[y]], ord[y]],
             p3 → member[t, ord[y]], p4 → member[succ[t], ord[y]],
             p5 → member[succ[t], domain[enum[x]]],
             p6 → member[APPLY[enum[x], t], APPLY[enum[x], succ[t]]],
             p7 → member[APPLY[enum[x], succ[t]], image[enum[x], ord[y]]],
             p8 -> member[APPLY[enum[x], t], U[image[enum[x], ord[y]]]]}]] // Reverse

Out[36]= or[member[APPLY[enum[x], t], U[image[enum[x], ord[y]]]], not[equal[ord[y], U[ord[y]]]],
           not[member[t, ord[y]]], not[subclass[ord[y], domain[enum[x]]]]] == True

In[37]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

The next step is to eliminate the variable **t**.

Lemma.

```
In[39]:= Map[equal[V, #] &, SubstTest[class, t,
           implies[and[subclass[r, s], equal[U[r], r], member[t, r]], member[t, u]], {r → ord[y],
            s → domain[enum[x]], u → image[inverse[enum[x]], U[image[enum[x], ord[y]]]]}]]

Out[39]= or[not[equal[ord[y], U[ord[y]]]], not[subclass[ord[y], domain[enum[x]]]],
           subclass[ord[y], image[inverse[enum[x]], U[image[enum[x], ord[y]]]]]] == True

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

Main Theorem. If **ord[y]** is a limit ordinal contained in the domain of enum[x], then **t = image[enum[x], ord[y]]** satisfies the inclusion **t ⊂ U[t]**.

```
In[42]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 → and[equal[ord[y], U[ord[y]]], subclass[ord[y], domain[enum[x]]]],
             p2 -> subclass[ord[y], image[inverse[enum[x]], U[image[enum[x], ord[y]]]]],
             p3 -> subclass[image[enum[x], ord[y]], U[image[enum[x], ord[y]]]]}]] // Reverse

Out[42]= or[not[equal[ord[y], U[ord[y]]]], not[subclass[ord[y], domain[enum[x]]]],
           subclass[image[enum[x], ord[y]], U[image[enum[x], ord[y]]]]] == True
```

```
In[44]:= or[not[equal[ord[y_], U[ord[y_]]]], not[subclass[ord[y_], domain[enum[x_]]]],
           subclass[image[enum[x_], ord[y_]], U[image[enum[x_], ord[y_]]]] := True
```

Corollary.

```
In[45]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 → and[equal[ord[y], U[ord[y]]], subclass[ord[y], domain[enum[x]]]],
             p2 -> subclass[image[enum[x], ord[y]], U[image[enum[x], ord[y]]]],
             p3 → not[member[U[image[enum[x], ord[y]]], image[enum[x], ord[y]]]]}]] // Reverse
```

```
Out[45]= or[not[equal[ord[y], U[ord[y]]]],
           not[member[U[image[enum[x], ord[y]]], image[enum[x], ord[y]]]],
           not[subclass[ord[y], domain[enum[x]]]]] == True
```

```
In[47]:= or[not[equal[ord[y_], U[ord[y_]]]],
           not[member[U[image[enum[x_], ord[y_]]], image[enum[x_], ord[y_]]]],
           not[subclass[ord[y_], domain[enum[x_]]]]] := True
```