APPLY[enum[x], succ[y]]

Johan G. F. Belinfante
2010 November 25

```
In[1]:= SetDirectory["l:"]; << goedel.10nov24a

        :Package Title: goedel.10nov24a              2010 November 24 at 7:25 p.m.

        It is now:  2010 Nov 25 at 14:24

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

The ordinal enumeration function **enum[x]** satisfies the following inclusion for any ordinal **ord[y]**.

```
In[2]:= subclass[U[image[enum[x], ord[y]]], APPLY[enum[x], U[ord[y]]]]

Out[2]= True
```

In this notebook it is shown that this inclusion can be sharpened to an equation when **ord[y]** is a successor ordinal contained in the domain of **enum[x]**.

derivation

Lemma. Specialization of the inclusion derived previously to the case of a successor ordinal.

```
In[3]:= SubstTest[subclass, U[image[enum[x], ord[t]]],
          APPLY[enum[x], U[ord[t]]], t → succ[ord[y]]] // Reverse

Out[3]= subclass[U[image[enum[x], succ[ord[y]]]], APPLY[enum[x], ord[y]]] == True

In[4]:= subclass[U[image[enum[x_], succ[ord[y_]]]], APPLY[enum[x_], ord[y_]]] := True
```

Lemma. A temporary rule.

```
In[5]:= subclass[x, union[setpart[x], y]] // AssertTest

Out[5]= subclass[x, union[y, setpart[x]]] == or[member[x, V], subclass[x, y]]

In[6]:= subclass[x_, union[y_, setpart[x_]]] := or[member[x, V], subclass[x, y]]
```

Lemma. An inclusion in the opposite direction.

```
In[7]:= Map[implies[member[ord[y], domain[enum[x]]], subclass[APPLY[enum[x], ord[y]], #]] &,
          SubstTest[image, t, union[u, v],
           {t → composite[inverse[E], enum[x]], u → ord[y], v → set[ord[y]]}]] // Reverse

Out[7]= or[not[member[ord[y], domain[enum[x]]]],
          subclass[APPLY[enum[x], ord[y]], U[image[enum[x], succ[ord[y]]]]]] == True

In[8]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. The two inclusions are here combined into an equation.

```
In[9]:= SubstTest[and, implies[p, subclass[u, v]],
         subclass[v, u], {p → member[ord[x], domain[enum[y]]],
          u -> APPLY[enum[y], ord[x]], v -> U[image[enum[y], succ[ord[x]]]]}]

Out[9]= or[equal[APPLY[enum[y], ord[x]], U[image[enum[y], succ[ord[x]]]]],
          not[member[ord[x], domain[enum[y]]]]] == True

In[10]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the **ord** wrapper yields one formulation of the main result.

Main theorem. If **y ∈ domain[enum[x]]**, then **APPLY[enum[x], y] = U[image[enum[x], succ[y]]**.

```
In[11]:= SubstTest[implies, equal[y, ord[t]],
           or[equal[APPLY[enum[x], y], U[image[enum[x], succ[y]]]],
            not[member[y, domain[enum[x]]]], t → y] // Reverse

Out[11]= or[equal[APPLY[enum[x], y], U[image[enum[x], succ[y]]]],
           not[member[y, domain[enum[x]]]]] == True

In[12]:= or[equal[APPLY[enum[x_], y_], U[image[enum[x_], succ[y_]]]],
           not[member[y_, domain[enum[x_]]]]] := True
```

An ordinal is a successor ordinal if it is not zero or a limit ordinal.

Corollary. A reformulation of the main theorem.

```
In[13]:= SubstTest[implies, equal[u, succ[ord[v]]],
           or[equal[APPLY[enum[x], ord[v]], U[image[enum[x], u]]],
            not[subclass[u, domain[enum[x]]]]], {u → ord[y], v → U[ord[y]]}] // Reverse

Out[13]= 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[14]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Eliminating the **ord** wrapper yields the following restatement of the main theorem.

Theorem. If **y ⊂ domain[enum[x]]** is a successor ordinal, then **APPLY[enum[x], U[y]] = U[image[enum[x], y]]**.

```
In[15]:= SubstTest[implies, equal[y, ord[t]],
           or[equal[APPLY[enum[x], U[y]], U[image[enum[x], y]]],
            equal[y, U[y]], not[subclass[y, domain[enum[x]]]]], t → y] // Reverse

Out[15]= or[equal[y, U[y]], equal[APPLY[enum[x], U[y]], U[image[enum[x], y]]],
           not[member[y, OMEGA]], not[subclass[y, domain[enum[x]]]]] == True

In[16]:= or[equal[APPLY[enum[x_], U[y_]], U[image[enum[x_], y_]]], equal[y_, U[y_]],
           not[member[y_, OMEGA]], not[subclass[y_, domain[enum[x_]]]]] := True
```

counterexample

The following two counterexamples show that the equation need not hold for non-successor ordinals.

```
In[17]:= implies[subclass[ord[y], domain[enum[x]]],
           equal[U[image[enum[x], ord[y]]], APPLY[enum[x], U[ord[y]]]]] /. {x → PRIMES, y → 0}

Out[17]= False

In[18]:= implies[subclass[ord[y], domain[enum[x]]],
           equal[U[image[enum[x], ord[y]]], APPLY[enum[x], U[ord[y]]]]] /. {x → PRIMES, y → omega}

Out[18]= False
```