U[range[ordlist[x]]]

Johan G. F. Belinfante
2007 May 15

```
In[1]:= SetDirectory["l:"]; << goedel93.13a; << tools.m

        :Package Title: goedel93.13a      2007 May 13 at 9:40 a.m.

        It is now:  2007 May 15 at 15:48

        Loading Simplification Rules

        TOOLS.M                                      Revised 2007 May 6

        weightlimit = 40
```

summary

The function **ordlist[x]** begins listing ordinals in a class **x** in increasing order, starting with the smallest ordinal, if any. The listing comes to a halt when there are no more ordinals left at some point, or else the list keeps going on forever. The set **range[ordlist[x]]** of ordinals that are placed on this list is either finite, possibly empty, or else is a countably infinite subset of **x**.

Just as ordinals can be classified as successor ordinals or limit ordinals, there is an analogous dichotomy for classes of ordinals. A class of ordinals either has a largest member, or it is unbounded. This can be restated in terms of the sum class. The sum class of any class of ordinals is either its greatest member, or else it contains the whole class. In this notebook necessary and sufficient conditions on the class **x** are derived for each of these two possibilities for the particular case of classes of ordinals of the form **range[ordlist[x]]**.

finite lists

Lemma. If there are only a finite number of ordinals in **x**, then all are listed.

```
In[2]:= SubstTest[implies, member[t, FINITE], equal[intersection[OMEGA, t], range[ordlist[t]]],
          t → intersection[OMEGA, x]] // Reverse

Out[2]= or[equal[intersection[OMEGA, x], range[ordlist[x]]],
          not[member[intersection[OMEGA, x], FINITE]]] == True

In[3]:= or[equal[intersection[OMEGA, x_], range[ordlist[x_]]],
          not[member[intersection[OMEGA, x_], FINITE]]] := True
```

Lemma. Since the length **domain[ordlist[x]]** of the list **ordlist[x]** in general is either a natural number or is the set **omega** of all natural numbers, it can not be a member of itself.

```
In[4]:= SubstTest[member, ord[t], ord[t], t → domain[ordlist[x]]] // Reverse

Out[4]= member[domain[ordlist[x]], domain[ordlist[x]]] == False

In[5]:= member[domain[ordlist[x_]], domain[ordlist[x_]]] := False
```

Lemma. The same is true for the set **succ[U[domain[ordlist[x]]]]**, which is either a natural number or the ordinal **succ[omega]**.

```
In[6]:= SubstTest[member, succ[U[ord[t]]], ord[t], t → domain[ordlist[x]]] // Reverse

Out[6]= member[succ[U[domain[ordlist[x]]]], domain[ordlist[x]]] == False

In[7]:= member[succ[U[domain[ordlist[x_]]]], domain[ordlist[x_]]] := False
```

Theorem. Either **U[domain[ordlist[x]]]** is not in the domain of the list **ordlist[x]**, or else it is the point where the list ends. In either case, this implies the following:

```
In[8]:= Map[not, SubstTest[implies, member[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]],
           member[succ[y], domain[ordlist[x]]], y → U[domain[ordlist[x]]]]] // Reverse

Out[8]= member[APPLY[ordlist[x], U[domain[ordlist[x]]]], U[intersection[OMEGA, x]]] == False

In[9]:= member[APPLY[ordlist[x_], U[domain[ordlist[x_]]]], U[intersection[OMEGA, x_]]] := False
```

Lemma. If there are only finitely many ordinals in **x**, then either the list is empty, or it stops at the point **U[domain[ordlist[x]]]**.

```
In[10]:= SubstTest[implies, member[t, omega],
           or[equal[0, t], member[U[t], t]], t → domain[ordlist[x]]] // Reverse

Out[10]= or[equal[0, intersection[OMEGA, x]],
           member[U[domain[ordlist[x]]], domain[ordlist[x]]],
           not[member[intersection[OMEGA, x], FINITE]]] == True

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

Lemma. (The case that the list is empty.)

```
In[12]:= SubstTest[implies, equal[0, t], not[member[U[t], t]], t → domain[ordlist[x]]] // Reverse

Out[12]= or[not[equal[0, intersection[OMEGA, x]]],
           not[member[U[domain[ordlist[x]]], domain[ordlist[x]]]]] == True

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

Lemma. (The case that the list does not stop.)

```
In[14]:= SubstTest[implies, equal[omega, t],
           not[member[U[t], t]], t → domain[ordlist[x]]] // Reverse

Out[14]= or[member[intersection[OMEGA, x], FINITE],
           not[member[U[domain[ordlist[x]]], domain[ordlist[x]]]]] == True
```

```
In[15]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. (The various cases considered in the above lemmas can conveniently be combined into a single rewrite rule.)

```
In[16]:= equiv[member[U[domain[ordlist[x]]], domain[ordlist[x]]],
            and[member[intersection[OMEGA, x], FINITE],
             not[equal[0, intersection[OMEGA, x]]]]] // not // not

Out[16]= True

In[17]:= member[U[domain[ordlist[x_]]], domain[ordlist[x_]]] :=
          and[member[intersection[OMEGA, x], FINITE], not[equal[0, intersection[OMEGA, x]]]]
```

Corollary. (The list goes on forever if and only if it does not stop.)

```
In[18]:= Map[not, SubstTest[member, U[ord[t]], ord[t], t → domain[ordlist[x]]]]

Out[18]= equal[domain[ordlist[x]], U[domain[ordlist[x]]]] ==
          or[equal[0, intersection[OMEGA, x]], not[member[intersection[OMEGA, x], FINITE]]]

In[19]:= equal[domain[ordlist[x_]], U[domain[ordlist[x_]]]] :=
          or[equal[0, intersection[OMEGA, x]], not[member[intersection[OMEGA, x], FINITE]]]
```

Theorem. If the number of ordinals in **x** is finite and not zero, then the last ordinal listed is the greatest ordinal in **x**.

```
In[20]:= (SubstTest[implies,
             and[member[u, v], subclass[v, OMEGA]], or[member[u, U[v]], equal[u, U[v]]],
             {u → APPLY[ordlist[x], y], v → intersection[OMEGA, x]}] // Reverse) /.
           y → U[domain[ordlist[x]]]

Out[20]= or[equal[0, intersection[OMEGA, x]],
           equal[APPLY[ordlist[x], U[domain[ordlist[x]]]], U[intersection[OMEGA, x]]],
           not[member[intersection[OMEGA, x], FINITE]]] == True

In[21]:= or[equal[0, intersection[OMEGA, x_]],
           equal[APPLY[ordlist[x_], U[domain[ordlist[x]]]], U[intersection[OMEGA, x_]]],
           not[member[intersection[OMEGA, x_], FINITE]]] := True
```

U[range[ordlist[x]]]

In this section the study of **U[range[ordlist[x]]]** begins, with the goal of finding conditions on **x** for this class to be either a member of the class of ordinals **range[ordlist[x]]** or to contain that class.

Lemma. If the class **x** is finite and not empty, then the greatest ordinal in **x** is on the list **ordlist[x]**.

```
In[22]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → and[member[intersection[OMEGA, x], FINITE], not[disjoint[OMEGA, x]]],
            p2 -> equal[APPLY[ordlist[x], U[domain[ordlist[x]]]], U[intersection[OMEGA, x]]],
            p3 → member[U[intersection[OMEGA, x]], range[ordlist[x]]]}]] // Reverse

Out[22]= or[equal[0, intersection[OMEGA, x]],
          member[U[intersection[OMEGA, x]], range[ordlist[x]]],
          not[member[intersection[OMEGA, x], FINITE]]] == True

In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

Technical lemma.

```
In[24]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
           {u -> composite[HULL[intersection[OMEGA, x]], SUCC], v -> E, w → ordlist[x]}] // Reverse

Out[24]= subclass[composite[HULL[intersection[OMEGA, x]], SUCC, ordlist[x]],
          composite[inverse[IMAGE[inverse[ordlist[x]]], E]] == True

In[25]:= subclass[composite[HULL[intersection[OMEGA, x_]], SUCC, ordlist[x_]],
          composite[inverse[IMAGE[inverse[ordlist[x_]]], E]] := True
```

Theorem. (A simple consequence of the recursion relation satisfied by **ordlist[x]**.)

```
In[26]:= (SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> composite[oopart[t], FIRST, id[cart[V, range[oopart[t]]]]],
            u → composite[oopart[t], SUCC],
            v → composite[E, oopart[t]]}] // Reverse) /. t → ordlist[x]

Out[26]= subclass[image[ordlist[x], U[domain[ordlist[x]]]], U[range[ordlist[x]]]] == True

In[27]:= subclass[image[ordlist[x_], U[domain[ordlist[x_]]]], U[range[ordlist[x_]]]] := True
```

Corollary for the case that the list goes on forever.

```
In[28]:= SubstTest[implies,
          and[equal[domain[t], omega], subclass[image[t, U[domain[t]]], U[range[t]]],
          subclass[image[t, omega], U[range[t]]], t → ordlist[x]] // Reverse

Out[28]= or[member[intersection[OMEGA, x], FINITE],
          subclass[range[ordlist[x]], U[range[ordlist[x]]]]] == True

In[29]:= (% /. x → x_) /. Equal → SetDelayed
```

condition for the list to end

Theorem. If the number of ordinals in **x** is finite and not zero, then there is a largest listed ordinal.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], not[implies[p1, p4]],
           {p1 → and[member[intersection[OMEGA, x], FINITE], not[disjoint[OMEGA, x]]],
            p2 → member[U[intersection[OMEGA, x]], range[ordlist[x]]],
            p3 → equal[intersection[OMEGA, x], range[ordlist[x]]],
            p4 → member[U[range[ordlist[x]]], range[ordlist[x]]]}]] // Reverse

Out[30]= or[equal[0, intersection[OMEGA, x]], member[U[range[ordlist[x]]], range[ordlist[x]]],
           not[member[intersection[OMEGA, x], FINITE]]] == True

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

Lemma. (Basic dichotomy for classes of ordinals, specialized to the case of **range[ordlist[x]]**.)

```
In[32]:= SubstTest[implies, subclass[t, OMEGA],
           implies[subclass[t, U[t]], not[member[U[t], t]]], t → range[ordlist[x]]] // Reverse

Out[32]= or[not[member[U[range[ordlist[x]]], range[ordlist[x]]]],
           not[subclass[range[ordlist[x]], U[range[ordlist[x]]]]]] == True

In[33]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> and[member[intersection[OMEGA, x], FINITE], not[disjoint[OMEGA, x]]],
            p2 -> member[U[range[ordlist[x]]], range[ordlist[x]]],
            p3 -> not[subclass[range[ordlist[x]], U[range[ordlist[x]]]]]}]] // Reverse

Out[34]= or[equal[0, intersection[OMEGA, x]], not[member[intersection[OMEGA, x], FINITE]],
           not[subclass[range[ordlist[x]], U[range[ordlist[x]]]]]] == True

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

Lemma. (The trivial case that **x** has no ordinal members.)

```
In[36]:= SubstTest[implies, empty[t], subclass[range[ordlist[t]], U[range[ordlist[t]]]],
           t → intersection[OMEGA, x]] // Reverse

Out[36]= or[not[equal[0, intersection[OMEGA, x]]],
           subclass[range[ordlist[x]], U[range[ordlist[x]]]]] == True

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

Theorem. A necessary and sufficient condition for the class **range[ordlist[x]]** to be unbounded.

```
In[38]:= equiv[subclass[range[ordlist[x]], U[range[ordlist[x]]]],
           or[equal[0, intersection[OMEGA, x]],
            not[member[intersection[OMEGA, x], FINITE]]]] // not // not

Out[38]= True

In[39]:= subclass[range[ordlist[x_]], U[range[ordlist[x_]]]] :=
          or[equal[0, intersection[OMEGA, x]], not[member[intersection[OMEGA, x], FINITE]]]
```

condition for the list to continue forever

Lemma. (The trivial case.)

```
In[40]:= SubstTest[implies, empty[t], not[member[U[range[ordlist[t]]], range[ordlist[t]]]],
           t → intersection[OMEGA, x]] // Reverse

Out[40]= or[not[equal[0, intersection[OMEGA, x]]],
           not[member[U[range[ordlist[x]]], range[ordlist[x]]]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (The dichotomy rule.)

```
In[42]:= SubstTest[implies, subclass[t, OMEGA], implies[subclass[t, U[t]],
             not[member[U[t], t]]], t → range[ordlist[x]]] // Reverse // MapNotNot

Out[42]= or[member[intersection[OMEGA, x], FINITE],
           not[member[U[range[ordlist[x]]], range[ordlist[x]]]]] == True

In[43]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. A necessary and sufficient condition for the class **range[ordlist[x]]** to have a largest member.

```
In[44]:= equiv[member[U[range[ordlist[x]]], range[ordlist[x]]],
            and[member[intersection[OMEGA, x], FINITE],
             not[equal[0, intersection[OMEGA, x]]]]] // not // not

Out[44]= True

In[45]:= member[U[range[ordlist[x_]]], range[ordlist[x_]]] :=
          and[member[intersection[OMEGA, x], FINITE], not[equal[0, intersection[OMEGA, x]]]]
```

Corollary. (A simple example: listing the primes.)

```
In[46]:= SubstTest[member, U[range[ordlist[x]]], range[ordlist[x]], x → PRIMES] // Reverse

Out[46]= member[omega, PRIMES] == False

In[47]:= member[omega, PRIMES] := False
```