domain[ordlist[x]]

Johan G. F. Belinfante
2006 March 18

```
In[1]:= SetDirectory["l:"]; << goedel79.17a; << tools.m

        :Package Title: goedel79.17a           2006 March 17 at 6:50 p.m.

        It is now:  2006 Mar 18 at 18:28

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 March 7

        weightlimit = 40
```

summary

The class **ordlist[x]** is finite if and only if **x** holds finitely many ordinals, and is countably infinite otherwise.

intersection[OMEGA, x] as a wrapper

The class **intersection[OMEGA, x]** behaves effectively as a wrapper for classes of ordinals. Many results are automatically recognized as true when this wrapper is used. One can removing this wrapper to obtain equivalent statements with **subclass[x, OMEGA]** as an explicit hypothesis. The following instance will be used in the sequel.

```
In[2]:= SubstTest[implies, equal[x, intersection[OMEGA, z]],
          subclass[image[HULL[x], y], x], z → x]

Out[2]= or[not[subclass[x, OMEGA]], subclass[image[HULL[x], y], x]] == True

In[3]:= or[not[subclass[x_, OMEGA]], subclass[image[HULL[x_], y_], x_]] := True
```

gap-free property

In this section a theorem is derived that amounts to the statement that there are no gaps between one member **y** of a class **x** of ordinals and the next **hull[x, succ[y]]**. Using variables, this amounts to the following statement:

```
In[4]:= Map[not,
         SubstTest[and, implies[and[p1, r1], r3], implies[and[q1, r1, r2, r3], or[r4, r5]],
          implies[and[r3, r5], r6], implies[and[r1, r6], r7], not[and[r2, r3, r7]],
          implies[and[q1, r3, r4], r8], not[implies[and[p1, q1, r1, r2], r4]],
          {p1 -> subclass[x, OMEGA], q1 → member[y, OMEGA],
           r1 → member[v, x], r2 -> member[v, hull[x, succ[y]]], r3 → member[v, OMEGA],
           r4 → subclass[v, y], r5 → member[y, v], r6 → subclass[succ[y], v],
           r7 → subclass[hull[x, succ[y]], v], r8 → member[v, succ[y]]}]]

Out[4]= or[not[member[v, x]], not[member[v, hull[x, succ[y]]]],
          not[member[y, OMEGA]], not[subclass[x, OMEGA]], subclass[v, y]] == True

In[5]:= (% /. {x → x_, y → y_, v → v_}) /. Equal → SetDelayed
```

Eliminating the set variable **v** yields:

```
In[6]:= Map[equal[V, #] &,
          SubstTest[class, v, or[not[member[v, x]], not[member[v, u]], not[member[y, w]],
            not[subclass[x, w]], subclass[v, y]], {u → hull[x, succ[y]], w → OMEGA}]] // Reverse

Out[6]= or[not[member[y, OMEGA]], not[subclass[x, OMEGA]],
          subclass[U[intersection[x, hull[x, succ[y]]]], y]] == True

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

The following lemma will be used to eliminate the sum class functor **U** in the above result.

```
In[8]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
         {u -> intersection[x, hull[x, succ[ord[y]]]], v → P[ord[y]], w → id[OMEGA]}]

Out[8]= or[not[subclass[U[intersection[x, hull[x, succ[ord[y]]]]], ord[y]]],
          subclass[intersection[OMEGA, x, hull[x, succ[ord[y]]]], succ[ord[y]]]] == True

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the **ord** wrapper and the ordinal class wrapper, one finds:

```
In[10]:= SubstTest[implies, and[equal[y, ord[z]], equal[w, intersection[OMEGA, x]]],
          or[not[subclass[U[intersection[x, hull[x, succ[y]]]], y]],
           subclass[intersection[w, hull[x, succ[y]]], succ[y]]], {w → x, z → y}]

Out[10]= or[not[member[y, OMEGA]], not[subclass[x, OMEGA]],
           not[subclass[U[intersection[x, hull[x, succ[y]]]], y]],
           subclass[intersection[x, hull[x, succ[y]]], succ[y]]] == True

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

Combining these results, one finds:

```
In[12]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → subclass[x, OMEGA],
            p2 → member[y, OMEGA], p3 -> subclass[U[intersection[x, hull[x, succ[y]]]], y],
            p4 -> subclass[intersection[x, hull[x, succ[y]]], succ[y]]}]]

Out[12]= or[not[member[y, OMEGA]], not[subclass[x, OMEGA]],
           subclass[intersection[x, hull[x, succ[y]]], succ[y]]] == True

In[13]:= or[not[member[y_, OMEGA]], not[subclass[x_, OMEGA]],
           subclass[intersection[x_, hull[x_, succ[y_]]], succ[y_]]] := True
```

a finiteness invariance

In the process of listing the ordinals in a class in increasing order, each time one adds only one more member, and so at any stage of the iteration, only finitely many have been listed. A formal statement that this finiteness property is an invariance of the iteration process is derived in this section. The basic idea is this:

```
In[14]:= SubstTest[implies, and[subclass[u, v], member[v, FINITE]], member[u, FINITE],
          {u → intersection[x, hull[x, succ[y]]], v → intersection[x, succ[y]]}]

Out[14]= or[member[intersection[x, hull[x, succ[y]]], FINITE],
           not[member[intersection[x, y], FINITE]],
           not[subclass[intersection[x, hull[x, succ[y]]], succ[y]]] == True

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

Applying the result of the preceding section yields:

```
In[16]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
           implies[and[p1, p4], p5], implies[and[p2, p3], p6], implies[and[p5, p6], p7],
           not[implies[and[p1, p2, p3], p7]], {p1 → subclass[x, OMEGA], p2 → member[y, x],
            p3 → member[intersection[x, y], FINITE], p4 → member[y, OMEGA],
            p5 -> subclass[intersection[x, hull[x, succ[y]]], succ[y]],
            p6 -> member[intersection[x, succ[y]], FINITE],
            p7 -> member[intersection[x, hull[x, succ[y]]], FINITE]}]]

Out[16]= or[member[intersection[x, hull[x, succ[y]]], FINITE], not[member[y, x]],
           not[member[intersection[x, y], FINITE]], not[subclass[x, OMEGA]]] == True

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

The variable **y** is eliminated now to obtain:

```
In[18]:= Map[implies[not[member[x, FINITE]], equal[V, #]] &,
           SubstTest[class, y, implies[and[subclass[x, u], member[y, v]],
             member[intersection[x, hull[x, succ[y]]], w]], {u → OMEGA,
             v -> intersection[x, image[inverse[IMAGE[id[x]]], FINITE]], w → FINITE}]] // Reverse

Out[18]= or[member[x, FINITE], not[subclass[x, OMEGA]],
           subclass[image[SUCC, intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]],
            image[inverse[HULL[x]], image[inverse[IMAGE[id[x]]], FINITE]]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

The inverse images can be eliminated using:

```
In[20]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → y, v → image[inverse[HULL[x]], image[inverse[IMAGE[id[x]]], z]],
           w → composite[IMAGE[id[x]], HULL[x]]}]

Out[20]= or[not[subclass[y, image[inverse[HULL[x]], image[inverse[IMAGE[id[x]]], z]]]],
           subclass[image[IMAGE[id[x]], image[HULL[x], y]], z]] == True

In[21]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[22]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → and[subclass[x, OMEGA], not[member[x, FINITE]]],
            p2 -> subclass[image[SUCC, intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]],
              image[inverse[HULL[x]], image[inverse[IMAGE[id[x]]], FINITE]]],
            p3 → subclass[image[IMAGE[id[x]], image[HULL[x], image[SUCC,
                 intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]]], FINITE]}]]

Out[22]= or[member[x, FINITE], not[subclass[x, OMEGA]],
           subclass[image[IMAGE[id[x]], image[HULL[x], image[SUCC,
               intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]]], FINITE]] == True

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

---

using the invariance

If **v** is a subclass of a class **w** that is invariant under **u**, then **range[iterate[u,v]]** is a subclass of **w**. For the special case that **u** is the function that produces the next ordinal in a class **x**, this says:

```
In[24]:= Map[or[not[member[A[intersection[OMEGA, x]], w]], #] &, SubstTest[implies,
           and[subclass[v, w], invariant[u, w]], subclass[range[iterate[u, v]], w],
           {u → composite[HULL[intersection[OMEGA, x]], SUCC],
            v → set[A[intersection[OMEGA, x]]]}]]

Out[24]= or[not[member[A[intersection[OMEGA, x]], w]],
           not[subclass[image[HULL[intersection[OMEGA, x]], image[SUCC, w]], w]],
           subclass[range[ordlist[x]], w]] == True

In[25]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Removing the ordinal-class wrapper yields:

```
In[26]:= SubstTest[implies, and[equal[y, intersection[OMEGA, x]], member[A[y], w],
           invariant[composite[HULL[y], SUCC], w]], subclass[range[ordlist[x]], w], y → x]

Out[26]= or[not[member[A[x], w]], not[subclass[x, OMEGA]],
           not[subclass[image[HULL[x], image[SUCC, w]], w]],
           subclass[range[ordlist[x]], w]] == True

In[27]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

The class that was shown to be invariant in the preceding section is **w = intersection[x,image[inverse[IMAGE[id[x]]], FINITE]]**. One obtains in this case:

```
In[28]:= SubstTest[or, not[member[A[x], w]], not[subclass[x, OMEGA]],
          not[subclass[image[HULL[x], image[SUCC, w]], w]], subclass[range[ordlist[x]], w],
          w -> intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]

Out[28]= or[equal[0, x], not[member[A[x], x]], not[member[intersection[x, A[x]], FINITE]],
           not[subclass[x, OMEGA]], not[subclass[image[HULL[x],
              image[SUCC, intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]], x]],
           not[subclass[image[IMAGE[id[x]], image[HULL[x],
               image[SUCC, intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]]], FINITE]],
           subclass[image[IMAGE[id[x]], range[ordlist[x]]], FINITE]] == True

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

This simplifies greatly when **x** is infinite. If **x** is an infinite class of ordinals, then **intersection[x,y]** is finite for every **y** listed by **ordlist[x]**.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           implies[p1, p4], implies[p1, p5], implies[p1, p6], implies[p6, p7],
           implies[p1, p8], implies[and[p2, p3, p4, p5, p7, p8], p9],
           not[implies[p1, p9]], {p1 → and[subclass[x, OMEGA], not[member[x, FINITE]]],
            p2 → not[equal[0, x]], p3 → member[A[x], x],
            p4 -> subclass[image[HULL[x], image[SUCC,
                intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]], x],
            p5 -> subclass[image[IMAGE[id[x]], image[HULL[x],
                image[SUCC, intersection[x, image[inverse[IMAGE[id[x]]], FINITE]]]]], FINITE],
            p6 → disjoint[x, A[x]], p7 -> member[intersection[x, A[x]], FINITE],
            p8 → subclass[x, OMEGA],
            p9 -> subclass[image[IMAGE[id[x]], range[ordlist[x]]], FINITE]}]]

Out[30]= or[member[x, FINITE], not[subclass[x, OMEGA]],
           subclass[image[IMAGE[id[x]], range[ordlist[x]]], FINITE]] == True

In[31]:= or[member[x_, FINITE], not[subclass[x_, OMEGA]],
           subclass[image[IMAGE[id[x_]], range[ordlist[x_]]], FINITE]] := True
```

an important corollary

The condition derived in the preceding section implies that if **x** is an infinite class of ordinals, then the range of **ordlist[x]** is contained in **U[x]**.

```
In[32]:= Map[not,
           SubstTest[and, implies[p0, p3], implies[p1, p2], implies[and[p0, p1, p2, p3], p4],
            not[implies[and[p0, p1], p4]], {p0 → equal[y, range[ordlist[x]]],
             p1 → and[subclass[x, OMEGA], not[member[x, FINITE]]],
             p2 -> subclass[image[IMAGE[id[x]], range[ordlist[x]]], FINITE],
             p3 → subclass[y, x], p4 -> subclass[y, U[x]]}]] /. y → range[ordlist[x]]

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

In[33]:= or[member[x_, FINITE], not[subclass[x_, OMEGA]],
           subclass[range[ordlist[x_]], U[x_]]] := True
```

Reintroducing the ordinal class wrapper yields:

```
In[34]:= SubstTest[implies, and[subclass[y, OMEGA], not[member[y, FINITE]]],
          subclass[range[ordlist[y]], U[y]], y → intersection[OMEGA, x]]

Out[34]= or[member[intersection[OMEGA, x], FINITE],
           subclass[range[ordlist[x]], U[intersection[OMEGA, x]]]] == True

In[35]:= or[member[intersection[OMEGA, x_], FINITE],
           subclass[range[ordlist[x_]], U[intersection[OMEGA, x_]]]] := True
```

applying induction

The recursion relation for **ordlist[x]** is used to derive a temporary rewrite rule:

```
In[36]:= IminComp[ordlist[x], SUCC, V]

Out[36]= image[inverse[ordlist[x]], U[intersection[OMEGA, x]]] ==
          image[inverse[SUCC], domain[ordlist[x]]]

In[37]:= image[inverse[ordlist[x_]], U[intersection[OMEGA, x_]]] :=
          image[inverse[SUCC], domain[ordlist[x]]]
```

From this one derives the fact that if the set of ordinals listed by **ordlist** is contained in **U[intersection[OMEGA,x]]**, then the list goes on forever.

```
In[38]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> range[ordlist[x]], v -> U[intersection[OMEGA, x]], w → inverse[ordlist[x]]}]

Out[38]= or[not[subclass[range[ordlist[x]], U[intersection[OMEGA, x]]]],
           subclass[image[SUCC, domain[ordlist[x]]], domain[ordlist[x]]]] == True
```

```
In[39]:= (% /. x → x_) /. Equal → SetDelayed
```

It was shown in the preceding section that this is the case when **x** holds an infinite number of ordinals.

```
In[40]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → not[member[intersection[OMEGA, x], FINITE]],
            p2 -> subclass[range[ordlist[x]], U[intersection[OMEGA, x]]],
            p3 → invariant[SUCC, domain[ordlist[x]]]}]]
```

```
Out[40]= or[member[intersection[OMEGA, x], FINITE],
           subclass[image[SUCC, domain[ordlist[x]]], domain[ordlist[x]]]] == True
```

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

An application of induction yields:

```
In[42]:= SubstTest[implies, INDUCTIVE[w], subclass[omega, w], w → domain[ordlist[x]]]
```

```
Out[42]= or[equal[0, intersection[OMEGA, x]],
           not[subclass[image[SUCC, domain[ordlist[x]]], domain[ordlist[x]]]],
           subclass[omega, domain[ordlist[x]]]] == True
```

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

A more precise statement is that if **x** holds infinitely many ordinals then the domain of **ordlist[x]** is equal to **omega**.

```
In[44]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[and[p2, p3], p4], implies[p4, p5], implies[p5, p6],
           not[implies[p1, p6]], {p1 -> not[member[intersection[OMEGA, x], FINITE]],
            p2 → not[empty[intersection[OMEGA, x]]],
            p3 -> invariant[SUCC, domain[ordlist[x]]], p4 -> subclass[omega, domain[ordlist[x]]],
            p5 → not[member[domain[ordlist[x]], omega]], p6 → equal[domain[ordlist[x]], omega]}]]
```

```
Out[44]= or[equal[omega, domain[ordlist[x]]], member[intersection[OMEGA, x], FINITE]] == True
```

```
In[45]:= (% /. x → x_) /. Equal → SetDelayed
```

two rewrite rules for domain[ordlist[x]]

If **x** only holds finitely many ordinals, then the domain of **ordlist[x]** is a natural number.

```
In[46]:= SubstTest[implies, member[y, FINITE],
          member[domain[ordlist[y]], omega], y → intersection[OMEGA, x]]
```

```
Out[46]= or[member[domain[ordlist[x]], omega],
           not[member[intersection[OMEGA, x], FINITE]]] == True
```

```
In[47]:= (% /. x → x_) /. Equal → SetDelayed
```

The converse also holds.

```
In[48]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[domain[ordlist[x]], omega], p2 → not[equal[domain[ordlist[x]], omega]],
            p3 -> member[intersection[OMEGA, x], FINITE]}]]

Out[48]= or[member[intersection[OMEGA, x], FINITE],
           not[member[domain[ordlist[x]], omega]]] == True

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

Since the implications go in both directions, one can introduce a rewrite rule.

```
In[50]:= equiv[member[domain[ordlist[x]], omega], member[intersection[OMEGA, x], FINITE]]

Out[50]= True

In[51]:= member[domain[ordlist[x_]], omega] := member[intersection[OMEGA, x], FINITE]
```

It obviously follows that in this case the domain of **ordlist[x]** is not the set **omega**.

```
In[52]:= SubstTest[implies, equal[omega, z], not[member[z, omega]], z → domain[ordlist[x]]]

Out[52]= or[not[equal[omega, domain[ordlist[x]]]],
           not[member[intersection[OMEGA, x], FINITE]]] == True

In[53]:= (% /. x → x_) /. Equal → SetDelayed
```

This yields a rewrite for the case that the **domain** of **ordlist[x]** is **omega**.

```
In[54]:= equiv[equal[domain[ordlist[x]], omega], not[member[intersection[OMEGA, x], FINITE]]]

Out[54]= True

In[55]:= equal[omega, domain[ordlist[x_]]] := not[member[intersection[OMEGA, x], FINITE]]
```