domain[ordlist[x]] as an ordinal

Johan G. F. Belinfante
2007 May 19

```
In[1]:= SetDirectory["l:"]; << goedel93.18a; << tools.m

        :Package Title: goedel93.18a      2007 May 18 at 6:55 p.m.

        It is now:  2007 May 19 at 16:9

        Loading Simplification Rules

        TOOLS.M                                      Revised 2007 May 6

        weightlimit = 40
```

summary

Several new rewrite rules for **domain[ordlist[x]]** are derived. An improvement of an existing rewrite rule is derived for the special case that the number of ordinals belonging to the class **x** is finite and nonzero. A recurrent theme in this notebook is that because **domain[ordlist[x]]** is either a natural number or the set **omega** of all natural numbers, it is an ordinal number, and therefore one can apply various rewrite rules for ordinals.

rules for domain[ordlist[x]]

Corollary of the fact that **domain[ordlist[x]]** is not a member of itself.

```
In[2]:= equal[APPLY[ordlist[x], domain[ordlist[x]]], V]

Out[2]= True

In[3]:= APPLY[ordlist[x_], domain[ordlist[x_]]] := V
```

Theorem.

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

Out[4]= member[U[domain[ordlist[x]]], domain[ordlist[x]]] ==
         and[member[intersection[OMEGA, x], FINITE], not[equal[0, intersection[OMEGA, x]]]]

In[5]:= member[U[domain[ordlist[x_]]], domain[ordlist[x_]]] :=
         and[member[intersection[OMEGA, x], FINITE], not[equal[0, intersection[OMEGA, x]]]]
```

Corollary. (For ordinals, **subclass** literals are equivalent to negated reversed **member** literals.)

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

Out[6]= subclass[domain[ordlist[x]], U[domain[ordlist[x]]]] ==
         or[equal[0, intersection[OMEGA, x]], not[member[intersection[OMEGA, x], FINITE]]]

In[7]:= subclass[domain[ordlist[x_]], U[domain[ordlist[x_]]]] :=
         or[equal[0, intersection[OMEGA, x]], not[member[intersection[OMEGA, x], FINITE]]]
```

Theorem.

```
In[8]:= SubstTest[member, ord[t], U[ord[t]], t → domain[ordlist[x]]] // Reverse

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

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

Theorem.

```
In[10]:= SubstTest[member, U[ord[t]], U[ord[t]], t → domain[ordlist[x]]] // Reverse

Out[10]= member[U[domain[ordlist[x]]], U[domain[ordlist[x]]]] == False

In[11]:= member[U[domain[ordlist[x_]]], U[domain[ordlist[x_]]]] := False
```

improving an existing rule

In this section a rewrite rule is derived which says that if the largest ordinal in **x** is on the list **ordlist[x]**, then it is the last item on this list.

Lemma.

```
In[12]:= ((member[y, image[inverse[funpart[t]], z]] // AssertTest) /. t → ordlist[x]) /.
           z -> U[intersection[x, OMEGA]] // Reverse

Out[12]= member[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]] ==
          member[y, U[domain[ordlist[x]]]]

In[13]:= member[APPLY[ordlist[x_], y_], U[intersection[OMEGA, x_]]] :=
          member[y, U[domain[ordlist[x]]]]
```

Theorem. (An application of a nonselfmembership rule derived in the previous section.)

```
In[14]:= SubstTest[implies, and[equal[u, v], member[u, w]], member[v, w],
           {u -> APPLY[ordlist[x], y],
            v -> U[intersection[OMEGA, x]], w -> U[intersection[OMEGA, x]]}] // Reverse

Out[14]= or[not[equal[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]]],
           not[member[y, U[domain[ordlist[x]]]]]] == True

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

Lemma.

```
In[16]:= SubstTest[and, equal[u, V], subclass[u, v],
           {u -> U[intersection[x, OMEGA]], v → OMEGA}] // Reverse

Out[16]= equal[V, U[intersection[OMEGA, x]]] == False

In[17]:= equal[V, U[intersection[OMEGA, x_]]] := False
```

Theorem. (Since the **APPLY** expression can not be **V**, it must occur at a point of the domain of **ordlist[x]**.)

```
In[18]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
           {v → APPLY[ordlist[x], y], u -> U[intersection[OMEGA, x]], w → V}] // Reverse

Out[18]= or[member[y, domain[ordlist[x]]],
           not[equal[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]]]] == True

In[19]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[20]:= SubstTest[implies, and[member[y, t], not[member[y, U[t]]]],
           not[equal[t, U[t]]], t → domain[ordlist[x]]] // Reverse

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

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

Theorem. (The largest ordinal only gets listed when the number of ordinals in **x** is finite.)

```
In[22]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 -> equal[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]],
             p2 → not[member[y, U[domain[ordlist[x]]]]], p3 -> member[y, domain[ordlist[x]]],
             p4 -> and[member[intersection[OMEGA, x], FINITE],
               not[equal[0, intersection[OMEGA, x]]]]}]] // Reverse

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

In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. A general result about ordinal numbers.

```
In[24]:= SubstTest[implies, and[member[x, v], subclass[v, w]],
           member[x, w], {v -> ord[y], w -> succ[U[ord[y]]]}] // Reverse

Out[24]= or[equal[x, U[ord[y]]], member[succ[x], ord[y]], not[member[x, ord[y]]]] == True

In[25]:= or[equal[x_, U[ord[y_]]], member[succ[x_], ord[y_]], not[member[x_, ord[y_]]]] := True
```

Lemma. (Specialization of the lemma to the case of **domain[ordlist[x]]**, which is an ordinal.)

```
In[26]:= SubstTest[implies, and[member[y, ord[t]], not[member[y, U[ord[t]]]]],
           equal[y, U[ord[t]]], t → domain[ordlist[x]]] // Reverse

Out[26]= or[equal[y, U[domain[ordlist[x]]]],
           member[succ[y], domain[ordlist[x]]], not[member[y, domain[ordlist[x]]]]] == True

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

Theorem. If the largest ordinal in **x** is listed, then it is listed last.

```
In[28]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[p3, p4], implies[and[p2, p4], p5], not[implies[p1, p5]],
            {p1 -> equal[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]],
             p2 -> member[y, domain[ordlist[x]]],
             p3 → not[member[y, U[domain[ordlist[x]]]]],
             p4 → not[member[succ[y], domain[ordlist[x]]]],
             p5 -> equal[y, U[domain[ordlist[x]]]]}]] // Reverse

Out[28]= or[equal[y, U[domain[ordlist[x]]]],
           not[equal[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]]]] == True

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

Combining the results proved above with the existing rewrite rule yields a logical equivalence:

```
In[30]:= equiv[equal[APPLY[ordlist[x], y], U[intersection[OMEGA, x]]],
            and[equal[y, U[domain[ordlist[x]]]], member[intersection[OMEGA, x], FINITE],
             not[equal[0, intersection[OMEGA, x]]]]] // not // not

Out[30]= True

In[31]:= equal[APPLY[ordlist[x_], y_], U[intersection[OMEGA, x_]]] :=
          and[equal[y, U[domain[ordlist[x]]]],
           member[intersection[OMEGA, x], FINITE], not[equal[0, intersection[OMEGA, x]]]]
```

---

eliminating the variable y

Lemma.

```
In[32]:= SubstTest[implies,
           and[member[z, V], equal[z, APPLY[ordlist[x], y]]], member[z, range[ordlist[x]]],
           {y -> U[domain[ordlist[x]]], z → U[intersection[x, OMEGA]]}] // Reverse

Out[32]= or[equal[0, intersection[OMEGA, x]],
           member[U[intersection[OMEGA, x]], range[ordlist[x]]],
           not[member[intersection[OMEGA, x], FINITE]]] == True

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

Theorem. (The largest ordinal in **x** is listed if and only if the number of ordinals in **x** is finite and not zero.)

```
In[34]:= Map[not[empty[#]] &, SubstTest[class, y, equal[APPLY[funpart[t], y], u],
           {t → ordlist[x], u -> U[intersection[OMEGA, x]]}]]

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

In[35]:= member[U[intersection[OMEGA, x_]], range[ordlist[x_]]] :=
          and[member[intersection[OMEGA, x], FINITE], not[equal[0, intersection[OMEGA, x]]]]
```

---

serendipity

A general simplification rule for ordinals:

```
In[36]:= equiv[and[member[x, OMEGA], member[x, y], member[y, OMEGA]],
          and[member[x, y], member[y, OMEGA]]]

Out[36]= True

In[37]:= and[member[x_, OMEGA], member[x_, y_], member[y_, OMEGA]] :=
          and[member[x, y], member[y, OMEGA]]
```

Theorem. (A variable-free version of a theorem about ordinals derived in this notebook.)

```
In[38]:= Map[empty[composite[Id, complement[#]]] &,
          dif[composite[SUCC, BIGCUP, id[OMEGA]], S] // complement // RelnNormality]

Out[38]= subclass[composite[SUCC, BIGCUP, id[OMEGA]], S] == True

In[39]:= subclass[composite[SUCC, BIGCUP, id[OMEGA]], S] := True
```