rewrite rules for the ordinal wrapper ord[x]

Johan G. F. Belinfante and Ming Li
2006 March 14

```
In[1]:= SetDirectory["l:"]; << goedel79.12b; << tools.m
        :Package Title: goedel79.12b           2006 March 12 at 7:25 p.m.
        It is now:  2006 Mar 14 at 14:2
        Loading Simplification Rules
        TOOLS.M                      Revised 2006 March 7
        weightlimit = 40
```

summary

Theorems about ordinals can often be formulated compactly by using the **ord** wrapper. In this notebook some facts about ordinals are reformulated as rewrite rules involving this wrapper.

A rule

Lemma.

```
In[2]:= SubstTest[implies, member[y, OMEGA], or[equal[0, A[y]], equal[0, y]], y → ord[x]]
Out[2]= or[equal[0, A[ord[x]]], equal[0, ord[x]]] == True
In[3]:= (% /. x → x_) /. Equal → SetDelayed
```

The least member of any nonzero ordinal is **0**.

```
In[4]:= equal[A[ord[x]], complement[image[V, ord[x]]]]
Out[4]= True
In[5]:= A[ord[x_]] := complement[image[V, ord[x]]]
```

fix-HULL

Any ordinal is closed under arbitrary intersections.

```
In[6]:= SubstTest[fix, HULL[intersection[y, OMEGA]], y -> ord[x]]
Out[6]= fix[HULL[ord[x]]] == ord[x]

In[7]:= fix[HULL[ord[x_]]] := ord[x]
```

image under inverse[SUCC]

Theorem.

```
In[8]:= equal[image[inverse[SUCC], ord[x]], U[ord[x]]] // AssertTest
Out[8]= equal[image[inverse[SUCC], ord[x]], U[ord[x]]] == True

In[9]:= image[inverse[SUCC], ord[x_]] := U[ord[x]]
```

As a corollary, one obtains the following formula for the restriction of the successor function to an ordinal, generalizing an existing rule for the special case of **omega**.

```
In[10]:= SubstTest[composite, id[y], funpart[z], {y → ord[x], z → SUCC}] // Reverse
Out[10]= composite[SUCC, id[U[ord[x]]]] == composite[id[ord[x]], SUCC]

In[11]:= composite[SUCC, id[U[ord[x_]]]] := composite[id[ord[x]], SUCC]
```

ordinals less than or equal to a given ordinal

The set of ordinals less than or equal to a given ordinal is its successor.

```
In[12]:= AssInt[OMEGA, FULL, P[ord[x]]] // Reverse
Out[12]= intersection[OMEGA, P[ord[x]]] == succ[ord[x]]

In[13]:= intersection[OMEGA, P[ord[x_]]] := succ[ord[x]]
```

other intersection rules

Lemma.

```
In[14]:= SubstTest[member, ord[x], intersection[u, v],
           {u -> OMEGA, v -> fix[composite[inverse[E], IMAGE[inverse[S]]]]}] // Reverse
Out[14]= member[ord[x], fix[composite[inverse[E], IMAGE[inverse[S]]]]] == False

In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[16]:= (member[y, fix[composite[inverse[E], IMAGE[inverse[S]]]]] // AssertTest // Reverse) /.
          y → ord[x]

Out[16]= member[ord[x], image[inverse[S], ord[x]]] == False

In[17]:= member[ord[x_], image[inverse[S], ord[x_]]] := False
```

Theorem.

```
In[18]:= Map[intersection[FULL, #] &,
          ImageComp[id[P[ord[x]]], composite[inverse[S], id[P[ord[x]]]], ord[x]]]

Out[18]= intersection[FULL, image[inverse[S], ord[x]]] == ord[x]

In[19]:= intersection[FULL, image[inverse[S], ord[x_]]] := ord[x]
```

Corollary.

```
In[20]:= AssInt[image[inverse[S], ord[x]], FULL, OMEGA]

Out[20]= intersection[OMEGA, image[inverse[S], ord[x]]] == ord[x]

In[21]:= intersection[OMEGA, image[inverse[S], ord[x_]]] := ord[x]
```

fullness

Every ordinal is full:

```
In[22]:= SubstTest[implies, and[member[x, z], member[z, OMEGA]], subclass[x, z], z → ord[y]]

Out[22]= or[not[member[x, ord[y]]], subclass[x, ord[y]]] == True

In[23]:= or[not[member[x_, ord[y_]]], subclass[x_, ord[y_]]] := True
```

successors

If **x** is less than an ordinal **y**, then the successor of **x** is less than or equal to **y**. Comment: The set **x** is also an ordinal, but one need not say so explicitly.

```
In[24]:= or[not[member[y, OMEGA]], not[member[x, y]], subclass[succ[x], y]] // AssertTest

Out[24]= or[not[member[x, y]], not[member[y, OMEGA]], subclass[succ[x], y]] == True

In[25]:= or[not[member[x_, y_]], not[member[y_, OMEGA]], subclass[succ[x_], y_]] := True
```

Reformulation using the **ord** wrapper.

```
In[26]:= SubstTest[implies, and[member[x, z], member[z, OMEGA]],
          subclass[succ[x], z], z → ord[y]]

Out[26]= or[not[member[x, ord[y]]], subclass[succ[x], ord[y]]] == True

In[27]:= or[not[member[x_, ord[y_]]], subclass[succ[x_], ord[y_]]] := True
```

an ordinal with no largest member is a limit ordinal

The direction of this rule is chosen to encourage the use of equality substitution rules, which helps to reduce the total number of rules needed for reasoning about ordinals.

```
In[28]:= equiv[member[U[ord[x]], ord[x]], not[equal[U[ord[x]], ord[x]]]]

Out[28]= True

In[29]:= member[U[ord[x_]], ord[x_]] := not[equal[ord[x], U[ord[x]]]]
```

members of predecessors

```
In[30]:= SubstTest[implies, and[member[v, OMEGA], member[u, U[v]]],
          member[succ[u], v], {u → x, v → ord[y]}]

Out[30]= or[member[succ[x], ord[y]], not[member[x, U[ord[y]]]]] == True

In[31]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The converse also holds.

```
In[32]:= equiv[member[x, U[ord[y]]], member[succ[x], ord[y]]]

Out[32]= True
```

The direction of this rule is chosen to agree with a similar rule for the **nat** wrapper. Comment. The set **x** is also an ordinal, but no expicit **ord** wrapper is needed here.

```
In[33]:= member[x_, U[ord[y_]]] := member[succ[x], ord[y]]
```

The dichotomy rule involves predecessors:

```
In[34]:= Map[implies[#, equal[ord[x], U[ord[y]]]] &,
          SubstTest[and, subclass[u, v], subclass[v, u], {u → ord[x], v → U[ord[y]]}]]

Out[34]= or[equal[ord[x], U[ord[y]]],
           member[succ[ord[x]], ord[y]], member[U[ord[y]], ord[x]]] == True

In[35]:= or[equal[ord[x_], U[ord[y_]]],
           member[succ[ord[x_]], ord[y_]], member[U[ord[y_]], ord[x_]]] := True
```

subclass rules involving omega

Because of dichotomy, any sStatement that one ordinal is contained in another can be rewritten in terms of membership.

```
In[36]:= SubstTest[subclass, ord[x], ord[y], y → omega]
Out[36]= subclass[ord[x], omega] == not[member[omega, ord[x]]]
```

```
In[37]:= subclass[ord[x_], omega] := not[member[omega, ord[x]]]
```

A similar result.

```
In[38]:= SubstTest[or, equal[ord[y], ord[x]], member[ord[x], ord[y]], y → omega]
Out[38]= or[equal[omega, ord[x]], member[ord[x], omega]] == not[member[omega, ord[x]]]
```

```
In[39]:= or[equal[omega, ord[x_]], member[ord[x_], omega]] := not[member[omega, ord[x]]]
```

RUSSELL

The Russell class frequently enters into reasoning about ordinals.

```
In[40]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u → ord[x], v → OMEGA, w → RUSSELL}]
Out[40]= subclass[ord[x], RUSSELL] == True
```

```
In[41]:= subclass[ord[x_], RUSSELL] := True
```

Corollary.

```
In[42]:= equal[intersection[RUSSELL, ord[x]], ord[x]]
Out[42]= True
```

```
In[43]:= intersection[RUSSELL, ord[x_]] := ord[x]
```

Uclosure rule

Successor ordinals are closed under arbitrary unions.

```
In[44]:= SubstTest[Uclosure, ord[y], y → succ[ord[x]]]
Out[44]= Uclosure[succ[ord[x]]] == succ[ord[x]]
```

```
In[45]:= Uclosure[succ[ord[x_]]] := succ[ord[x]]
```