ZN restricted to OMEGA

Johan G. F. Belinfante
2006 March 14

```
In[1]:= SetDirectory["l:"]; << goedel79.14a; << tools.m

        :Package Title: goedel79.14a           2006 March 14 at 2:20 p.m.

        It is now:  2006 Mar 14 at 15:38

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 March 7

        weightlimit = 40
```

introduction

Statements about the Zermelo-von Neumann cumulative hierarchy are formulated in the **GOEDEL** program in terms of the relation **ZN** which is the largest relation that commutes with the inverse membership relation.

```
In[2]:= commute[ZN, inverse[E]]

Out[2]= True

In[3]:= implies[subcommute[x, E], subclass[inverse[x], ZN]]

Out[3]= True
```

When the relation **ZN** is restricted to the class **OMEGA** of ordinal numbers, it is rewritten in terms of the inverse of the rank function **RANK**.

```
In[4]:= composite[ZN, id[OMEGA]]

Out[4]= composite[inverse[RANK], inverse[S], id[OMEGA]]
```

In this notebook some basic rewrite rules are derived that help in translating results about **ZN** involving ordinals to statements involving rank.

thinpart[ZN]

Some rewrite rules yield expressions involving **thinpart[ZN]**. When one restricts to the class **REGULAR**, the relation **thinpart[ZN]** is the same as **ZN**.

```
In[5]:= Assoc[ZN, id[domain[VERTSECT[ZN]]], id[REGULAR]] // Reverse

Out[5]= composite[thinpart[ZN], id[REGULAR]] == composite[ZN, id[REGULAR]]

In[6]:= composite[thinpart[ZN], id[REGULAR]] := composite[ZN, id[REGULAR]]
```

In particular, one can remove the **thinpart** wrapper for restrictions to **OMEGA**.

```
In[7]:= Assoc[thinpart[ZN], id[REGULAR], id[OMEGA]]

Out[7]= composite[thinpart[ZN], id[OMEGA]] == composite[inverse[RANK], inverse[S], id[OMEGA]]

In[8]:= composite[thinpart[ZN], id[OMEGA]] := composite[inverse[RANK], inverse[S], id[OMEGA]]
```

image rules

Lemma.

```
In[9]:= IminComp[id[OMEGA], RANK, image[inverse[S], ord[x]]]

Out[9]= image[inverse[RANK], image[inverse[S], ord[x]]] == image[inverse[RANK], ord[x]]

In[10]:= image[inverse[RANK], image[inverse[S], ord[x_]]] := image[inverse[RANK], ord[x]]
```

Theorem.

```
In[11]:= ImageComp[ZN, id[OMEGA], ord[x]] // Reverse

Out[11]= image[ZN, ord[x]] == image[inverse[RANK], ord[x]]

In[12]:= image[ZN, ord[x_]] := image[inverse[RANK], ord[x]]
```

Corollary.

```
In[13]:= ImageComp[ZN, id[OMEGA], set[ord[x]]]

Out[13]= image[inverse[RANK], P[ord[x]]] == P[image[inverse[RANK], ord[x]]]

In[14]:= image[inverse[RANK], P[ord[x_]]] := P[image[inverse[RANK], ord[x]]]
```

reify results

The results of the preceding section can be formulated without variables. Using **reify**, one finds:

```
In[45]:= Map[composite[VERTSECT[#], id[OMEGA]] &,
           SubstTest[reify, x, image[inverse[RANK], image[inverse[S], f[x]]], f → ord]] // Reverse

Out[45]= composite[IMAGE[inverse[RANK]], IMAGE[inverse[S]], id[OMEGA]] ==
          composite[IMAGE[inverse[RANK]], id[OMEGA]]
```

```
In[46]:= composite[IMAGE[inverse[RANK]], IMAGE[inverse[S]], id[OMEGA]] :=
          composite[IMAGE[inverse[RANK]], id[OMEGA]]
```

Similarly,

```
In[15]:= Map[composite[VERTSECT[#], id[OMEGA]] &,
           SubstTest[reify, x, image[ZN, f[x]], f → ord]] // Reverse

Out[15]= composite[IMAGE[ZN], id[OMEGA]] == composite[IMAGE[inverse[RANK]], id[OMEGA]]

In[16]:= composite[IMAGE[ZN], id[OMEGA]] := composite[IMAGE[inverse[RANK]], id[OMEGA]]
```

Corollary.

```
In[17]:= Assoc[POWER, IMAGE[ZN], id[OMEGA]] // Reverse

Out[17]= composite[VERTSECT[ZN], id[OMEGA]] == composite[POWER, IMAGE[inverse[RANK]], id[OMEGA]]

In[18]:= composite[VERTSECT[ZN], id[OMEGA]] := composite[POWER, IMAGE[inverse[RANK]], id[OMEGA]]
```

---

membership rule

Lemma.

```
In[19]:= member[pair[u, v], composite[inverse[RANK], w]] // AssertTest

Out[19]= member[pair[u, v], composite[inverse[RANK], w]] ==
          and[member[u, V], member[v, REGULAR], member[pair[u, rank[v]], w]]

In[20]:= member[pair[u_, v_], composite[inverse[RANK], w_]] :=
          and[member[u, V], member[v, REGULAR], member[pair[u, rank[v]], w]]
```

Lemma.

```
In[21]:= SubstTest[implies, and[subclass[u, v], member[v, V]],
          member[u, V], {u → rank[x], v → ord[y]}]

Out[21]= or[member[x, REGULAR], not[subclass[rank[x], ord[y]]]] == True

In[22]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[23]:= equiv[and[member[x, REGULAR], subclass[rank[x], ord[y]]], subclass[rank[x], ord[y]]]

Out[23]= True

In[24]:= and[member[x_, REGULAR], subclass[rank[x_], ord[y_]]] := subclass[rank[x], ord[y]]
```

Corollary.

```
In[25]:= SubstTest[and, member[x, V], and[member[x, REGULAR], subclass[rank[x], z]], z → ord[y]]

Out[25]= and[member[x, V], subclass[rank[x], ord[y]]] == subclass[rank[x], ord[y]]
```

```
In[26]:= and[member[x_, V], subclass[rank[x_], ord[y_]]] := subclass[rank[x], ord[y]]
```

The following rewrite rule is the main result of this section.

```
In[27]:= SubstTest[member, pair[u, y], composite[z, id[OMEGA]], {u → ord[x], z → ZN}] // Reverse

Out[27]= member[pair[ord[x], y], ZN] == subclass[rank[y], ord[x]]

In[28]:= member[pair[ord[x_], y_], ZN] := subclass[rank[y], ord[x]]
```

A similar result holds for **thinpart[ZN]**.

```
In[29]:= SubstTest[member, pair[ord[x], y], composite[z, id[OMEGA]], z → thinpart[ZN]] // Reverse

Out[29]= member[pair[ord[x], y], thinpart[ZN]] == subclass[rank[y], ord[x]]

In[30]:= member[pair[ord[x_], y_], thinpart[ZN]] := subclass[rank[y], ord[x]]
```

---

subclass rule

Lemma.

```
In[32]:= SubstTest[implies, and[subclass[x, z], member[z, V]],
          member[x, V], z → image[ZN, ord[y]]]

Out[32]= or[member[x, V], not[subclass[x, image[inverse[RANK], ord[y]]]]] == True

In[33]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[37]:= equiv[and[member[x, V], subclass[x, image[inverse[RANK], ord[y]]]],
          subclass[x, image[inverse[RANK], ord[y]]]]

Out[37]= True

In[39]:= and[member[x_, V], subclass[x_, image[inverse[RANK], ord[y_]]]] :=
          subclass[x, image[inverse[RANK], ord[y]]]
```

Corollary.

```
In[40]:= SubstTest[member, pair[ord[y], x], composite[inverse[E], w], w → VERTSECT[ZN]] // Reverse

Out[40]= subclass[x, image[inverse[RANK], ord[y]]] == subclass[rank[x], ord[y]]

In[41]:= subclass[x_, image[inverse[RANK], ord[y_]]] := subclass[rank[x], ord[y]]
```