ordinal addition, part 1.

Johan G. F. Belinfante
2010 December 15

```
In[1]:= SetDirectory["l:"]; << goedel.10dec13a

        :Package Title: goedel.10dec13a              2010 December 13 at 7:40 p.m.

        It is now:  2010 Dec 15 at 6:11

        Loading Simplification Rules

        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

        weightlimit = 40
```

summary

In this first of a series of notebooks about ordinal addition the function **ORDADD** for ordinal addition is defined, and a few of its properties are derived. The main tool used is the connection of its left-multiplication function **ordplus[x] = ORDADD ∘ LEFT[x]** with enumeration.

image[V, -] rules for enum[x]

In this section some simplification rules are derived that will be used later.

Theorem. Eliminating intersections in favor of composites.

```
In[3]:= SubstTest[intersection, funpart[t], image[V, y], t → enum[x]] // Reverse

Out[3]= intersection[enum[x], image[V, y]] == composite[enum[x], id[image[V, y]]]

In[4]:= intersection[enum[x_], image[V, y_]] := composite[enum[x], id[image[V, y]]]
```

Lemma. A technical simplification rule.

```
In[5]:= (subclass[composite[t, E], composite[inverse[IMAGE[inverse[t]]], id[image[V, y]], E]] //
           AssertTest) /. t → enum[x]

Out[5]= subclass[composite[enum[x], E],
          composite[inverse[IMAGE[inverse[enum[x]]]], id[image[V, y]], E]] ==
         or[not[equal[0, y]], subclass[domain[enum[x]], set[0]]]

In[6]:= subclass[composite[enum[x_], E],
          composite[inverse[IMAGE[inverse[enum[x_]]]], id[image[V, y_]], E]] :=
         or[not[equal[0, y]], subclass[domain[enum[x]], set[0]]]
```

Theorem. Moving **image[V, y]** out of **enum**.

```
In[7]:= SubstTest[equal, t, enum[range[t]],
          t → intersection[composite[enum[x], id[image[V, y]]]]] // Reverse

Out[7]= equal[composite[enum[x], id[image[V, y]]], enum[intersection[x, image[V, y]]]] == True

In[8]:= enum[intersection[x_, image[V, y_]]] := composite[enum[x], id[image[V, y]]]
```

definition

Definition. One can define **ORDADD** directly in terms of the class **ENUMS** of enumerations.

```
In[9]:= rotate[composite[SECOND, id[cart[OMEGA, OMEGA]], inverse[DIF],
           IMAGE[SECOND], id[ENUMS], inverse[IMAGE[id[cart[OMEGA, V]]]], E]] := ORDADD
```

Theorem. Simplification rule.

```
In[10]:= SubstTest[composite, rotate[t], id[cart[V, V]],
           t -> composite[SECOND, id[cart[OMEGA, OMEGA]], inverse[DIF], IMAGE[SECOND],
             id[ENUMS], inverse[IMAGE[id[cart[OMEGA, V]]]], E]] // Reverse

Out[10]= composite[ORDADD, id[cart[V, V]]] == ORDADD

In[11]:= composite[ORDADD, id[cart[V, V]]] := ORDADD
```

ordplus[x]

The main tool for deriving the properties of **ORDADD** is a formula for its left-multiplications.

Definition.

```
In[12]:= composite[ORDADD, LEFT[x_]] := ordplus[x]
```

Lemma. A technical simplification rule.

```
In[13]:= intersection[complement[FUNS], P[composite[enum[complement[x]], id[
              intersection[OMEGA, image[V, intersection[OMEGA, set[x]]]]]]]] // DoubleComplement

Out[13]= intersection[complement[FUNS], P[composite[enum[complement[x]],
             id[intersection[OMEGA, image[V, intersection[OMEGA, set[x]]]]]]] == 0

In[14]:= intersection[complement[FUNS], P[composite[enum[complement[x_]],
             id[intersection[OMEGA, image[V, intersection[OMEGA, set[x_]]]]]]]] := 0
```

Lemma. A technical simplification rule.

```
In[15]:= SubstTest[image, DIF, cart[OMEGA, set[t]],
           t -> union[x, complement[image[V, intersection[OMEGA, set[x]]]]] // Reverse

Out[15]= image[DIF, cart[OMEGA, intersection[OMEGA, set[x]]]] ==
          intersection[image[V, intersection[OMEGA, set[x]]], image[IMAGE[
            id[intersection[complement[x], image[V, intersection[OMEGA, set[x]]]]], OMEGA]]

In[16]:= image[DIF, cart[OMEGA, intersection[OMEGA, set[x_]]]] :=
          intersection[image[V, intersection[OMEGA, set[x]]], image[IMAGE[
            id[intersection[complement[x], image[V, intersection[OMEGA, set[x]]]]], OMEGA]]
```

Theorem.

```
In[17]:= SubstTest[composite, rotate[t], LEFT[x],
          t -> composite[SECOND, id[cart[OMEGA, OMEGA]], inverse[DIF],
            IMAGE[SECOND], id[ENUMS], inverse[IMAGE[id[cart[OMEGA, V]]], E]]

Out[17]= composite[enum[complement[x]], id[image[V, intersection[OMEGA, set[x]]]]] == ordplus[x]

In[18]:= composite[enum[complement[x_]],
           id[image[V, intersection[OMEGA, set[x_]]]]] := ordplus[x]
```

Corollary.

```
In[21]:= Assoc[enum[complement[x]], id[image[V, intersection[OMEGA, set[x]]]],
           id[image[V, intersection[OMEGA, set[x]]]]] // Reverse

Out[21]= composite[ordplus[x], id[image[V, intersection[OMEGA, set[x]]]]] == ordplus[x]

In[22]:= composite[ordplus[x_], id[image[V, intersection[OMEGA, set[x_]]]]] := ordplus[x]
```

ORDADD is a function

Theorem.

```
In[23]:= SubstTest[FUNCTION, composite[enum[complement[x]], id[t]],
           t → image[V, intersection[OMEGA, set[x]]]] // Reverse

Out[23]= FUNCTION[ordplus[x]] == True

In[24]:= FUNCTION[ordplus[x_]] := True
```

Corollary. Ordinal addition is a function.

```
In[25]:= Map[equal[V, #] &, SubstTest[class, x, FUNCTION[composite[t, LEFT[x]]], t → ORDADD]]

Out[25]= FUNCTION[ORDADD] == True

In[26]:= FUNCTION[ORDADD] := True
```

ordinal subtraction

Theorem.

```
In[27]:= Map[FUNCTION, composite[id[image[V, intersection[OMEGA, set[x]]]],
             inverse[enum[complement[x]]]] // DoubleInverse] // Reverse
```

```
Out[27]= FUNCTION[inverse[ordplus[x]]] == True
```

```
In[28]:= FUNCTION[inverse[ordplus[x_]]] := True
```

Theorem.

```
In[29]:= composite[inverse[LEFT[x]], inverse[ORDADD]] // DoubleInverse
```

```
Out[29]= composite[inverse[LEFT[x]], inverse[ORDADD]] == inverse[ordplus[x]]
```

```
In[30]:= composite[inverse[LEFT[x_]], inverse[ORDADD]] := inverse[ordplus[x]]
```

Lemma. Simplification rule.

```
In[31]:= fix[composite[FIRST, intersection[x, composite[inverse[SECOND], Di, SECOND]],
            inverse[FIRST]]] // Normality
```

```
Out[31]= fix[composite[FIRST, intersection[x, composite[inverse[SECOND], Di, SECOND]],
            inverse[FIRST]]] == domain[fix[composite[x, cross[Id, Di]]]]
```

```
In[32]:= fix[composite[FIRST, intersection[x_, composite[inverse[SECOND], Di, SECOND]],
            inverse[FIRST]]] := domain[fix[composite[x, cross[Id, Di]]]]
```

Theorem. Ordinal substraction is a function.

```
In[33]:= Map[equal[V, #] &,
          SubstTest[class, x, FUNCTION[composite[inverse[LEFT[x]], inverse[t]]], t → ORDADD]]
```

```
Out[33]= FUNCTION[rotate[ORDADD]] == True
```

```
In[34]:= FUNCTION[rotate[ORDADD]] := True
```

ordplus[ord[x]]

The most common situation is that **ordplus[x]** has its variable **x** wrapped with **ord**.

Theorem.

```
In[35]:= SubstTest[composite, enum[complement[t]],
            id[image[V, intersection[OMEGA, set[t]]]], t → ord[x]] // Reverse
```

```
Out[35]= enum[complement[ord[x]]] == ordplus[ord[x]]
```

```
In[36]:= enum[complement[ord[x_]]] := ordplus[ord[x]]
```

Corollary. Removing the **ord** wrapper.

```
In[37]:= SubstTest[implies, equal[x, ord[t]],
           equal[enum[complement[x]], ordplus[x]], t → x] // Reverse
```

```
Out[37]= or[equal[enum[complement[x]], ordplus[x]], not[member[x, OMEGA]]] == True
```

```
In[38]:= or[equal[enum[complement[x_]], ordplus[x_]], not[member[x_, OMEGA]]] := True
```

Theorem. A conditional rewrite rule.

```
In[39]:= implies[member[x, OMEGA], equal[enum[complement[x]], ordplus[x]]]
```

```
Out[39]= True
```

```
In[40]:= enum[complement[x_]] := ordplus[x] /; member[x, OMEGA]
```

---

simplification rules for ordplus

Further simplification rules for **ordplus[x]** are derived in this section.

Lemma.

```
In[42]:= Assoc[enum[complement[x]], id[image[V, intersection[OMEGA, set[x]]]], id[y]]
```

```
Out[42]= composite[enum[complement[x]],
           id[intersection[y, image[V, intersection[OMEGA, set[x]]]]]] ==
         composite[ordplus[x], id[y]]
```

```
In[43]:= composite[enum[complement[x_]],
           id[intersection[y_, image[V, intersection[OMEGA, set[x_]]]]]] :=
         composite[ordplus[x], id[y]]
```

Theorem.

```
In[44]:= Assoc[enum[complement[x]], id[OMEGA], id[image[V, intersection[OMEGA, set[x]]]]]
```

```
Out[44]= composite[ordplus[x], id[OMEGA]] == ordplus[x]
```

```
In[45]:= composite[ordplus[x_], id[OMEGA]] := ordplus[x]
```

Theorem.

```
In[48]:= Assoc[ordplus[x], id[image[V, intersection[OMEGA, set[x]]]], id[y]]
```

```
Out[48]= composite[ordplus[x], id[intersection[y, image[V, intersection[OMEGA, set[x]]]]]] ==
         composite[ordplus[x], id[y]]
```

```
In[49]:= composite[ordplus[x_], id[intersection[y_, image[V, intersection[OMEGA, set[x_]]]]]] :=
         composite[ordplus[x], id[y]]
```

Theorem.

```
In[51]:= SubstTest[intersection, funpart[t], image[V, y], t → ordplus[x]] // Reverse
Out[51]= intersection[image[V, y], ordplus[x]] == composite[ordplus[x], id[image[V, y]]]

In[52]:= intersection[image[V, y_], ordplus[x_]] := composite[ordplus[x], id[image[V, y]]]
```

---

serendipity

The simplification rules in this section were encountered in the course of discovering the formula for **ORDADD**.

Lemma.

```
In[53]:= equal[intersection[BIJ, ENUMS], ENUMS]
Out[53]= True

In[54]:= intersection[BIJ, ENUMS] := ENUMS
```

Theorem.

```
In[55]:= Assoc[IMAGE[id[cart[V, V]]], id[BIJ], id[ENUMS]]
Out[55]= composite[IMAGE[id[cart[V, V]]], id[ENUMS]] == id[ENUMS]

In[56]:= composite[IMAGE[id[cart[V, V]]], id[ENUMS]] := id[ENUMS]
```