ordinal addition, part 2.

Johan G. F. Belinfante
2010 December 15

```
In[1]:= SetDirectory["l:"]; << goedel.10dec15a
        :Package Title: goedel.10dec15a              2010 December 15 at 8:10 a.m.
        It is now:  2010 Dec 15 at 17:11
        Loading Simplification Rules
        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3
        weightlimit = 40
```

summary

In this second notebook about ordinal addition, formulas for the domain and range of the functions **ORDADD** and **ordplus[x]** are derived.

empty enumerations

If no ordinal belongs to a class **x** then its enumeration function **enum[x]** is empty.

Theorem.

```
In[2]:= SubstTest[empty, domain[funpart[t]], t → inverse[enum[x]]]
Out[2]= equal[0, domain[enum[x]]] == equal[0, intersection[OMEGA, x]]
In[3]:= equal[0, domain[enum[x_]]] := equal[0, intersection[OMEGA, x]]
```

Corollary.

```
In[4]:= SubstTest[empty, domain[funpart[t]], t → enum[x]]
Out[4]= equal[0, enum[x]] == equal[0, intersection[OMEGA, x]]
In[5]:= equal[0, enum[x_]] := equal[0, intersection[OMEGA, x]]
```

ordinal addition for non-ordinals

The function **ordplus[x]** is defined for any class. If **x** is not an ordinal, it is empty.

Lemma. The class **Ω** can not be a subclass of any set.

```
In[7]:= SubstTest[implies, and[subclass[t, x], member[x, y]], member[t, V], t → OMEGA] // Reverse
Out[7]= or[not[member[x, y]], not[subclass[OMEGA, x]]] == True

In[8]:= or[not[member[x_, y_]], not[subclass[OMEGA, x_]]] := True
```

Lemma. A simplification rule.

```
In[9]:= equiv[or[not[member[x, y]], subclass[OMEGA, x]], not[member[x, y]]]
Out[9]= True

In[10]:= or[not[member[x_, y_]], subclass[OMEGA, x_]] := not[member[x, y]]
```

Theorem. The function **ordplus[x]** is empty if and only if **x** is not an ordinal.

```
In[11]:= SubstTest[empty, enum[t],
           t -> intersection[complement[x], image[V, intersection[OMEGA, set[x]]]]] // Reverse
Out[11]= equal[0, ordplus[x]] == not[member[x, OMEGA]]

In[12]:= equal[0, ordplus[x_]] := not[member[x, OMEGA]]
```

Corollary. A temporary rewrite rule subsumed by a more general result derived in the next section.

```
In[13]:= SubstTest[empty, domain[funpart[t]], t → ordplus[x]] // Reverse
Out[13]= equal[0, domain[ordplus[x]]] == not[member[x, OMEGA]]

In[14]:= equal[0, domain[ordplus[x_]]] := not[member[x, OMEGA]]
```

domain

If **x** is an ordinal, then the domain of **ordplus[x]** is the class **Ω** of all ordinals.

Lemma. A temporary rewrite rule with an **ord** wrapper.

```
In[15]:= SubstTest[equal, domain[enum[complement[t]]], OMEGA, t→ ord[x]] // Reverse
Out[15]= equal[OMEGA, domain[ordplus[ord[x]]]] == True

In[16]:= domain[ordplus[ord[x_]]] := OMEGA
```

Lemma. (Removing the **ord** wrapper.)

```
In[17]:= SubstTest[implies, equal[x, ord[t]], equal[OMEGA, domain[ordplus[x]]], t → x] // Reverse

Out[17]= or[equal[OMEGA, domain[ordplus[x]]], not[member[x, OMEGA]]] == True

In[18]:= (% /. x → x_) /. Equal → SetDelayed
```

All previous special rules for the domain of ordplus[x] are subsumed by the following theorem.

Main Theorem. A general formula for the domain of **ordplus[x]**.

```
In[19]:= equal[domain[ordplus[x]], intersection[OMEGA, image[V, intersection[OMEGA, set[x]]]]]

Out[19]= True

In[20]:= domain[ordplus[x_]] := intersection[OMEGA, image[V, intersection[OMEGA, set[x]]]]
```

One can use **reify** to derive a formula for the domain of **ORDADD**.

Lemma. A temporary simplification rule.

```
In[21]:= IminComp[ORDADD, id[cart[V, V]], V] // Reverse

Out[21]= composite[Id, domain[ORDADD]] == domain[ORDADD]

In[22]:= composite[Id, domain[ORDADD]] := domain[ORDADD]
```

Theorem. Formula for the domain of **ORDADD**.

```
In[23]:= Map[reify[x, #] &, SubstTest[domain, composite[t, LEFT[x]], t → ORDADD]]

Out[23]= domain[ORDADD] == cart[OMEGA, OMEGA]

In[24]:= domain[ORDADD] := cart[OMEGA, OMEGA]
```

---

range

In this section a formula for the range of **ORDADD** is derived.

Theorem. A formula for the range of **ordplus[x]**.

```
In[25]:= ImageComp[enum[complement[x]], id[image[V, intersection[OMEGA, set[x]]]], V]

Out[25]= range[ordplus[x]] ==
          intersection[OMEGA, complement[x], image[V, intersection[OMEGA, set[x]]]]

In[26]:= range[ordplus[x_]] :=
          intersection[OMEGA, complement[x], image[V, intersection[OMEGA, set[x]]]]
```

As before, one can use **reify** to derive a variable-free result.

Theorem.

```
In[27]:= Map[reify[x, #] &, SubstTest[range, composite[t, LEFT[x]], t → ORDADD]]
Out[27]= composite[ORDADD, inverse[FIRST]] == composite[id[OMEGA], S, id[OMEGA]]
In[28]:= composite[ORDADD, inverse[FIRST]] := composite[id[OMEGA], S, id[OMEGA]]
```

Corollary.

```
In[29]:= ImageComp[ORDADD, inverse[FIRST], V] // Reverse
Out[29]= range[ORDADD] == OMEGA
In[30]:= range[ORDADD] := OMEGA
```

ORDADD is a proper class

Theorem. The function **ORDADD** is a proper class.

```
In[32]:= Map[not, SubstTest[implies, member[t, x], member[range[t], V], t → ORDADD]] // Reverse
Out[32]= member[ORDADD, x] == False
In[33]:= member[ORDADD, x_] := False
```

Theorem.

```
In[36]:= SubstTest[member, domain[funpart[t]], V, t → ordplus[x]]
Out[36]= member[ordplus[x], V] == not[member[x, OMEGA]]
In[37]:= member[ordplus[x_], V] := not[member[x, OMEGA]]
```