UBD = lambda[x, domain[UB[x]]]

Johan G. F. Belinfante
2008 July 5

```
In[1]:= SetDirectory["l:"]; << goedel.08jul04a; << tools.m
        :Package Title: goedel.08jul04a                 2008 July 4 at 10:25 p.m.
        It is now:  2008 Jul 5 at 22:38
        Loading Simplification Rules
        TOOLS.M                                    Revised 2008 May 17
        weightlimit = 40
```

summary

In this notebook basic properties of the function that takes **x** to **domain[UB[x]]** are derived.

introduction

There is no function that takes **x** to **UB[x]** because this class is never a set.

```
In[2]:= member[UB[x], V]
Out[2]= False
```

On the other hand, if **x** is a set, then so is **domain[UB[x]]**.

```
In[3]:= implies[member[x, V], member[domain[UB[x]], V]]
Out[3]= True
```

definition

For convenience the abbreviation **UBD** is introduced for the function that takes **x** to **domain[UB[x]]**. Thus

UBD = lambda[x, domain[UB[x]]].

This function is defined by the following membership rule:

```
In[4]:= member[x_, UBD] := and[equal[domain[UB[first[x]]], second[x]], member[first[x], V]]
```

Lemma.

```
In[5]:= UB[union[x, complement[image[V, y]]]] // RelnNormality

Out[5]= UB[union[x, complement[image[V, y]]]] == union[cart[V, complement[image[V, y]]], UB[x]]

In[6]:= UB[union[x_, complement[image[V, y_]]]] :=
          union[cart[V, complement[image[V, y]]], UB[x]]
```

Observation. With the above rewrite rule in place, no special membership rule is needed for ordered pairs.

```
In[7]:= member[pair[x, y], UBD]

Out[7]= and[equal[y, domain[UB[x]]], member[x, V], member[y, V]]
```

vertical section rule

Theorem. Vertical section rule.

```
In[8]:= image[UBD, set[x]] // Normality

Out[8]= image[UBD, set[x]] == intersection[image[V, set[x]], set[domain[UB[x]]]]

In[9]:= image[UBD, set[x_]] := intersection[image[V, set[x]], set[domain[UB[x]]]]
```

normalization

Lemma.

```
In[10]:= Map[empty, dif[UBD, cart[V, V]] // Normality]

Out[10]= subclass[UBD, cart[V, V]] == True

In[11]:= % /. Equal → SetDelayed
```

Theorem. (A normalization rule.)

```
In[12]:= Map[equal[#, UBD] &, SubstTest[reify, x, image[t, set[x]], t → UBD] // Reverse]

Out[12]= equal[UBD, VERTSECT[composite[FIRST,
             complement[composite[cross[E, Id], complement[inverse[E]]]]]]] == True

In[13]:= VERTSECT[
           composite[FIRST, complement[composite[cross[E, Id], complement[inverse[E]]]]]] := UBD
```

Corollary. The class **UBD** is a function.

```
In[14]:= SubstTest[FUNCTION, VERTSECT[t], t -> composite[FIRST,
             complement[composite[cross[E, Id], complement[inverse[E]]]]]] // Reverse

Out[14]= FUNCTION[UBD] == True

In[15]:= FUNCTION[UBD] := True
```

lambda[x, domain[LB[x]]]

A corresponding function for lower bounds is related to **UBD** as follows:

```
In[16]:= composite[UBD, IMAGE[SWAP]] // ReifNormality // Reverse

Out[16]= VERTSECT[
           composite[SECOND, complement[composite[cross[Id, E], complement[inverse[E]]]]] ==
          composite[UBD, IMAGE[SWAP]]

In[17]:= VERTSECT[
           composite[SECOND, complement[composite[cross[Id, E], complement[inverse[E]]]]] :=
          composite[UBD, IMAGE[SWAP]]
```

domain

Theorem.

```
In[18]:= Map[complement, SubstTest[class, x, member[image[t, set[x]], z], {t → UBD, z → set[0]}]]

Out[18]= domain[UBD] == V

In[19]:= domain[UBD] := V
```

Corollary.

```
In[20]:= Map[not[implies[member[UBD, x], #]] &,
           SubstTest[member, domain[funpart[t]], V, t → UBD]] // Reverse

Out[20]= member[UBD, x] == False

In[21]:= member[UBD, x_] := False
```

APPLY rule

Lemma. (Simplification rule.)

```
In[22]:= equal[union[complement[image[V, set[x]]], complement[image[V, set[domain[UB[x]]]]]],
          complement[image[V, set[x]]]]

Out[22]= True

In[23]:= union[complement[image[V, set[x_]]], complement[image[V, set[domain[UB[x_]]]]]] :=
          complement[image[V, set[x]]]
```

Theorem. (**APPLY** rule.)

```
In[24]:= Map[A, SubstTest[image, funpart[t], set[x], t → UBD]]

Out[24]= APPLY[UBD, x] == union[complement[image[V, set[x]]], domain[UB[x]]]

In[25]:= APPLY[UBD, x_] := union[complement[image[V, set[x]]], domain[UB[x]]]
```

inverse images

Theorem. Inverse image rule.

```
In[26]:= member[x, image[inverse[UBD], y]] // AssertTest

Out[26]= member[x, image[inverse[UBD], y]] == and[member[x, V], member[domain[UB[x]], y]]

In[27]:= member[x_, image[inverse[UBD], y_]] := and[member[x, V], member[domain[UB[x]], y]]
```

Theorem.

```
In[28]:= Map[not, SubstTest[implies, member[0, t], not[equal[0, t]], t → domain[UB[x]]] //
          Reverse

Out[28]= equal[0, domain[UB[x]]] == False

In[29]:= equal[0, domain[UB[x_]]] := False
```

Corollary. (Similar result for lower bounds.)

```
In[30]:= SubstTest[empty, domain[UB[t]], t → inverse[x]] // Reverse

Out[30]= equal[0, domain[LB[x]]] == False

In[31]:= equal[0, domain[LB[x_]]] := False
```

Corollary. (This is a substitute for **Normality**.)

```
In[32]:= SubstTest[class, x, member[x, t], t -> image[inverse[UBD], set[0]]]

Out[32]= image[inverse[UBD], set[0]] == 0

In[33]:= image[inverse[UBD], set[0]] := 0
```