some rewrite rules for VERTSECT[composite[x,y]]]

Johan G. F. Belinfante
2003 June 15

```
In[1]:= << goedel52.s04; << tools.m

        :Package Title: goedel52.s04       2003 June 11 at 6:50 p.m.

        It is now:  2003 Jun 17 at 9:36

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 June 13

        weightlimit = 40
```

■ introduction

This notebook documents the derivation of some rewrite rules of the form

```
In[2]:= "VERTSECT[composite[g,x_]]:=composite[IMAGE[g],VERTSECT[x]]";
```

In such a rule, the variable **x** is arbitrary, and can refer to any class, not just to sets. For brevity, let us just say that a class **g** is ***good*** if the the following equation holds for every class **x**:

```
In[3]:= question[g_, x_] := equal[VERTSECT[composite[g, x]], composite[IMAGE[g], VERTSECT[x]]]
```

It is worth pointing out that this question cannot be settled by using the built−inquantifier **assert[forall[x, ...]]**. The point is that the quantifier **forall** in the **GOEDEL** program is limited to sets. For sets the question is not interesting because the answer is always **True**. One can readily verify that this is the case by using one of the generic sets **a, b, c, d**. that are defined in the **TOOLS.M** file.

```
In[4]:= Map[member[#, V] &, {a, b, c, d, e, f, g}]

Out[4]= {True, True, True, True, member[e, V], member[f, V], member[g, V]}
```

When **question** is applied to the generic set **a** one finds:

```
In[5]:= question[x, a]

Out[5]= True
```

To decide whether a class is good, one must therefore consider composites with proper classes.

■ some examples

To show that a class is not good, one just needs to find one counterexample. For example, the following simple observation shows that sets are never good:

```
In[6]:= question[a, V]

Out[6]= False
```

The function **BIGCAP** is not good:

```
In[7]:= question[BIGCAP, S]

Out[7]= False
```

A relation cannot be good if the complement of its domain is a proper class. Here are some examples:

```
In[8]:= Map[question[#, cart[V, complement[domain[#]]]] &,
          {ASSOC, CAP, CART, COMPOSE, CROSS, CUP, DIF, FIRST, IMG,
           INVERSE, KURA, MAP, PAIRSET, RIF, ROT, SECOND, SWAP, SYMDIF, TWIST}]

Out[8]= {False, False, False, False, False, False, False, False, False,
         False, False, False, False, False, False, False, False, False, False}
```

On the other hand, the **GOEDEL** program already knows that certain proper classes are good:

```
In[9]:= question[V, x]

Out[9]= True

In[10]:= Map[question[#, x] &, {Id, Di}]

Out[10]= {True, True}

In[11]:= Map[question[#, x] &,
           {E, complement[E], complement[inverse[E]], S, complement[inverse[S]]}]

Out[11]= {True, True, True, True, True}
```

Many of these examples are relations whose vertical sections are all proper classes. Suppose **r** has this property:

```
In[12]:= assert[forall[y, not[member[image[r, singleton[y]], V]]]]

Out[12]= equal[0, domain[VERTSECT[r]]]

In[13]:= domain[VERTSECT[r]] := 0
```

It follows that **VERTSECT[r]** itself is empty:

```
In[14]:= SubstTest[composite, x, id[domain[x]], x -> VERTSECT[r]] // Reverse

Out[14]= VERTSECT[r] == 0

In[15]:= VERTSECT[r] := 0
```

One also has:

```
In[16]:= IMAGE[r] // VSNormality

Out[16]= IMAGE[r] == cart[singleton[0], singleton[0]]

In[17]:= IMAGE[r] := cart[singleton[0], singleton[0]]
```

It follows that **r** must be good:

```
In[18]:= question[r, x]

Out[18]= True
```

To summarize: if all vertical sections of a relation are proper classes, then the relation is good. The case of **Id** shows that the converse statement is not true; the identity relation is good even though it is thin; all its vertical sections are sets. There are many other examples of good thin relations as will be seen shortly.

■ a case of interest: inverse[E] is good.

One can sometimes show that a relation is good in two steps. The first step uses **VSNormality**.

```
In[19]:= stepone[x_, y_] :=
          VSNormality[symdif[composite[IMAGE[x], VERTSECT[y]], VERTSECT[composite[x, y]]]]
```

For example:

```
In[20]:= stepone[inverse[E], x_]

Out[20]= union[composite[VERTSECT[composite[inverse[e], x_]],
            id[complement[domain[VERTSECT[x_]]]]],
           intersection[complement[VERTSECT[composite[inverse[e], x_]]],
            composite[BIGCUP, VERTSECT[x_]]], intersection[
            composite[Di, BIGCUP, VERTSECT[x_]], VERTSECT[composite[inverse[e], x_]]]]] == 0
```

Assuming that one finds **0** on the right side of this output, one makes what is found into a rewrite rule:

```
In[21]:= % /. Equal -> SetDelayed
```

The second step in such a case uses **SubstTest**:

```
In[22]:= steptwo[x_, y_] := Module[{u = Unique[], v = Unique[]}, SubstTest[equal, 0, symdif[u, v],
                    {u -> VERTSECT[composite[x, y]], v -> composite[IMAGE[x], VERTSECT[y]]}]] //
           Reverse
```

For the present example:

```
In[23]:= steptwo[inverse[E], x]

Out[23]= equal[composite[BIGCUP, VERTSECT[x]], VERTSECT[composite[inverse[e], x]]] == True
```

This equation can be made into a rewrite rule; in other words, the relation **inverse[E]** is good.

```
In[24]:= VERTSECT[composite[inverse[E], x_]] := composite[BIGCUP, VERTSECT[x]]
```

For any good relation **g** one can derive a rewrite rule for **IMAGE[composite[g,x]]**.

```
In[25]:= stepthree[x_, y_] := Module[{z = Unique[]},
           SubstTest[VERTSECT, composite[x, z], z -> composite[y, inverse[E]]]]
```

In the case of **inverse[E]**:

```
In[26]:= stepthree[inverse[E], x]

Out[26]= IMAGE[composite[inverse[e], x]] == composite[BIGCUP, IMAGE[x]]

In[27]:= IMAGE[composite[inverse[E], x_]] := composite[BIGCUP, IMAGE[x]]
```

■ the relations inverse[S] and complement[S] are good

The same three steps work for **inverse[S]**.

```
In[28]:= stepone[inverse[S], x_]

Out[28]= union[composite[VERTSECT[composite[inverse[S], x_]],
            id[complement[domain[VERTSECT[x_]]]]],
          intersection[complement[VERTSECT[composite[inverse[S], x_]]],
            composite[IMAGE[inverse[S]], VERTSECT[x_]]],
          intersection[composite[complement[IMAGE[inverse[S]]], VERTSECT[x_]],
            VERTSECT[composite[inverse[S], x_]]]] == 0

In[29]:= % /. Equal -> SetDelayed

In[30]:= steptwo[inverse[S], x]

Out[30]= equal[composite[IMAGE[inverse[S]], VERTSECT[x]],
          VERTSECT[composite[inverse[S], x]]] == True
```

Thus **inverse[S]** is good, and one can add this rewrite rule:

```
In[31]:= VERTSECT[composite[inverse[S], x_]] := composite[IMAGE[inverse[S]], VERTSECT[x]]
```

There is also a companion rule with **VERTSECT** replaced by **IMAGE**.

```
In[32]:= stepthree[inverse[S], x]

Out[32]= IMAGE[composite[inverse[S], x]] == composite[IMAGE[inverse[S]], IMAGE[x]]

In[33]:= IMAGE[composite[inverse[S], x_]] := composite[IMAGE[inverse[S]], IMAGE[x]]
```

The case of **complement[S]** is similar:

```
In[34]:= stepone[complement[S], x_]

Out[34]= union[cart[intersection[image[inverse[x_], complement[singleton[0]]],
             image[inverse[VERTSECT[x_]], succ[singleton[0]]]], singleton[0]],
          composite[VERTSECT[composite[complement[S], x_]],
            id[union[complement[image[inverse[VERTSECT[x_]], succ[singleton[0]]]],
              image[inverse[x_], complement[singleton[0]]]]]]] == 0

In[35]:= % /. Equal -> SetDelayed

In[36]:= steptwo[complement[S], x]

Out[36]= equal[cart[image[inverse[VERTSECT[x]], succ[singleton[0]]], singleton[0]],
          VERTSECT[composite[complement[S], x]]] == True

In[37]:= VERTSECT[composite[complement[S], x_]] :=
          cart[image[inverse[VERTSECT[x]], succ[singleton[0]]], singleton[0]]
```

The companion rule is:

```
In[38]:= stepthree[complement[S], x]

Out[38]= IMAGE[composite[complement[S], x]] ==
          cart[image[inverse[IMAGE[x]], succ[singleton[0]]], singleton[0]]

In[39]:= IMAGE[composite[complement[S], x_]] :=
          cart[image[inverse[IMAGE[x]], succ[singleton[0]]], singleton[0]]
```

To summarize: the membership relation **E**, subset relation **S** and all relations obtained from them using **complement** and **inverse** are good.

```
In[40]:= Map[question[#, x] &, {E, inverse[E], complement[E], complement[inverse[E]]}]

Out[40]= {True, True, True, True}

In[41]:= Map[question[#, x] &, {S, inverse[S], complement[S], complement[inverse[S]]}]

Out[41]= {True, True, True, True}
```

■ some good functions

Some functions are shown to be good in this section. The first is **BIGCUP**.

```
In[42]:= stepone[BIGCUP, x_]

Out[42]= union[
           composite[VERTSECT[composite[BIGCUP, x_]], id[complement[domain[VERTSECT[x_]]]]],
           intersection[complement[VERTSECT[composite[BIGCUP, x_]]],
            composite[IMAGE[BIGCUP], VERTSECT[x_]]],
           intersection[composite[complement[IMAGE[BIGCUP]], VERTSECT[x_]],
            VERTSECT[composite[BIGCUP, x_]]]] == 0

In[43]:= % /. Equal -> SetDelayed

In[44]:= steptwo[BIGCUP, x]

Out[44]= equal[composite[IMAGE[BIGCUP], VERTSECT[x]], VERTSECT[composite[BIGCUP, x]]] == True

In[45]:= VERTSECT[composite[BIGCUP, x_]] := composite[IMAGE[BIGCUP], VERTSECT[x]]

In[46]:= stepthree[BIGCUP, x]

Out[46]= IMAGE[composite[BIGCUP, x]] == composite[IMAGE[BIGCUP], IMAGE[x]]

In[47]:= IMAGE[composite[BIGCUP, x_]] := composite[IMAGE[BIGCUP], IMAGE[x]]
```

Next: the function **POWER**.

```
In[48]:= stepone[POWER, x_]

Out[48]= union[composite[VERTSECT[composite[POWER, x_]], id[complement[domain[VERTSECT[x_]]]]],
           intersection[complement[VERTSECT[composite[POWER, x_]]],
            composite[IMAGE[POWER], VERTSECT[x_]]],
           intersection[composite[complement[IMAGE[POWER]], VERTSECT[x_]],
            VERTSECT[composite[POWER, x_]]]] == 0
```

```
In[49]:= % /. Equal -> SetDelayed

In[50]:= steptwo[POWER, x]

Out[50]= equal[composite[IMAGE[POWER], VERTSECT[x]], VERTSECT[composite[POWER, x]]] == True

In[51]:= VERTSECT[composite[POWER, x_]] := composite[IMAGE[POWER], VERTSECT[x]]

In[52]:= stepthree[POWER, x]

Out[52]= IMAGE[composite[POWER, x]] == composite[IMAGE[POWER], IMAGE[x]]

In[53]:= IMAGE[composite[POWER, x_]] := composite[IMAGE[POWER], IMAGE[x]]
```

The function **SINGLETON = VERTSECT[Id]** is good:

```
In[54]:= stepone[SINGLETON, x_]

Out[54]= union[
           composite[VERTSECT[composite[SINGLETON, x_]], id[complement[domain[VERTSECT[x_]]]]],
           intersection[complement[VERTSECT[composite[SINGLETON, x_]]],
            composite[IMAGE[SINGLETON], VERTSECT[x_]]],
           intersection[composite[complement[IMAGE[SINGLETON]], VERTSECT[x_]],
            VERTSECT[composite[SINGLETON, x_]]]] == 0

In[55]:= % /. Equal -> SetDelayed

In[56]:= steptwo[SINGLETON, x]

Out[56]= equal[composite[IMAGE[SINGLETON], VERTSECT[x]],
           VERTSECT[composite[SINGLETON, x]]] == True

In[57]:= VERTSECT[composite[SINGLETON, x_]] := composite[IMAGE[SINGLETON], VERTSECT[x]]

In[58]:= stepthree[SINGLETON, x]

Out[58]= IMAGE[composite[SINGLETON, x]] == composite[IMAGE[SINGLETON], IMAGE[x]]

In[59]:= IMAGE[composite[SINGLETON, x_]] := composite[IMAGE[SINGLETON], IMAGE[x]]
```

■ composites of good relations are good

If **g** and **h** are good, then so is **composite[g,h]**. This can be established by examining the generic case. Suppose:

```
In[60]:= VERTSECT[composite[g, x_]] := composite[IMAGE[g], VERTSECT[x]]

In[61]:= VERTSECT[composite[h, x_]] := composite[IMAGE[h], VERTSECT[x]]
```

One can derive:

```
In[62]:= stepthree[g, x]

Out[62]= IMAGE[composite[g, x]] == composite[IMAGE[g], IMAGE[x]]

In[63]:= IMAGE[composite[g, x_]] := composite[IMAGE[g], IMAGE[x]]
```

The desired conclusion follows:

```
In[64]:= question[composite[g, h], x]

Out[64]= True
```

As a corollary, it follows that **IMAGE[SWAP]**, for example, cannot be good. Suppose it were. Since **inverse[E]** and **SINGLETON** are good, it would follow that **SWAP** would be good:

```
In[65]:= composite[inverse[E], IMAGE[SWAP], SINGLETON]

Out[65]= SWAP
```

As was noted earlier, this is absurd because the complement of the domain of **SWAP** is a proper class.

■ inverse[SINGLETON] is not good

The global identity function **Id** is good. The restricted identity **id[x]** is not good, except possibly when **x** is the complement of a set.

```
In[66]:= question[id[x], cart[V, complement[x]]]

Out[66]= member[complement[x], V]
```

This can be used to show, for example, that **inverse[SINGLETON]**] cannot be good. If it were, then the identity restricted to **range[SINGLETON]** would be good:

```
In[67]:= composite[SINGLETON, inverse[SINGLETON]]

Out[67]= id[range[SINGLETON]]
```

This would imply that **complement[range[SINGLETON]]** is a set. This is absurd because the class **OMEGA** of ordinal numbers is a proper class, and only one ordinal is a singleton:

```
In[68]:= Map[not, SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
             {u -> OMEGA, v -> union[singleton[singleton[0]], complement[range[SINGLETON]]]}]]

Out[68]= member[complement[range[SINGLETON]], V] == False
```