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

Johan G. F. Belinfante
2003 June 15

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

\[ \text{In[2]} := \text{"VERTSECT[composite[g,x]]:=composite[IMAGE[g],VERTSECT[x]]";} \]

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 following equation holds for every class \( x \):

\[ \text{In[3]} := \text{question[g, x] := equal[VERTSECT[composite[g, x]], composite[IMAGE[g], VERTSECT[x]]]} \]

It is worth pointing out that this question cannot be settled by using the built-in quantifier \text{assert[forall[x, ...]]}. The point is that the quantifier \text{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.

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

When \text{question} is applied to the generic set \( a \) one finds:

\[ \text{In[5]} := \text{question[x, a]} \]

\[ \text{Out[5]} = \text{True} \]

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

\section*{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:
The function `BIGCAP` is not good:

```plaintext
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:

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

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

```plaintext
In[9]:= question[V, x]
Out[9]= True

In[10]:= Map[# & , Id, Di]
Out[10]= {True, True}

In[11]:= Map[# & , {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:

```plaintext
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:

```plaintext
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:

```plaintext
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:

\[ \text{In[18]} := \text{question}[r, x] \]

\[ \text{Out[18]} = \text{True} \]

To summarize: if all vertical sections of a relation are proper classes, then the relation is good. The case of \( \text{Id} \) 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: \( \text{inverse[E]} \) is good.

One can sometimes show that a relation is good in two steps. The first step uses \( \text{VSNormality} \).

\[ \text{In[19]} := \text{stepone}[x_, y_] := \text{VSNormality}[	ext{symdif}[	ext{composite}[	ext{IMAGE}[x]], \text{VERTSECT}[y]], \text{VERTSECT}[	ext{composite}[x, y]]] \]

For example:

\[ \text{In[20]} := \text{stepone}[\text{inverse[E]}, x_] \]

\[ \text{Out[20]} = \text{union}[	ext{composite}[	ext{VERTSECT}[	ext{composite}[\text{inverse}[\text{e}], x_]], \text{id}[	ext{complement}[	ext{domain}[	ext{VERTSECT}[x_]]]]], \text{intersection}[	ext{complement}[	ext{VERTSECT}[	ext{composite}[	ext{inverse}[\text{e}], x_]]], \text{complement}[	ext{BIGCUP}, \text{VERTSECT}[x_]]], \text{intersection}[	ext{composite}[	ext{Di}, \text{BIGCUP}, \text{VERTSECT}[x_]], \text{VERTSECT}[$\text{composite}[$\text{inverse}[\text{e}], x_]]]] == 0 \]

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

\[ \text{In[21]} := \% /. \text{Equal} \to \text{SetDelayed} \]

The second step in such a case uses \( \text{SubstTest} \):

\[ \text{In[22]} := \text{steptwo}[x_, y_] := \text{Module}[\{u = \text{Unique[]}\}, \text{SubstTest}[$\text{equal}, 0, \text{symdif}[u, v], \{u \to \text{VERTSECT}[$\text{composite}[x, y]]], v \to \text{composite}[$\text{IMAGE}[x], \text{VERTSECT}[y]]\}] \]

For the present example:

\[ \text{In[23]} := \text{steptwo}[\text{inverse[E]}, x] \]

\[ \text{Out[23]} = \text{equal}[$\text{composite}[$\text{BIGCUP}, \text{VERTSECT}[x]]], \text{VERTSECT}[$\text{composite}[\text{inverse}[\text{e}], x]]] == \text{True} \]

This equation can be made into a rewrite rule; in other words, the relation \( \text{inverse[E]} \) is good.

\[ \text{In[24]} := \text{VERTSECT}[$\text{composite}[$\text{inverse}[\text{E}], x_]] := \text{composite}[$\text{BIGCUP}, \text{VERTSECT}[x]] \]

For any good relation \( g \) one can derive a rewrite rule for \( \text{IMAGE}[$\text{composite}[g, x]] \).

\[ \text{In[25]} := \text{stepthree}[x_, y_] := \text{Module}[\{z = \text{Unique[]}\}, \text{SubstTest}[$\text{VERTSECT}, \text{composite}[x, z], z \to \text{composite}[y, \text{inverse}[\text{E}]]\}] \]

In the case of \( \text{inverse[E]} \):
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[composite[inverse[x_], complement[singleton[0]]]],
  IMAGE[逆[VERTSECT[x_]]], succ[singleton[0]]]],
  composite[VERTSECT[composite[complement[IMAGE[ Inv[S]]], x_]],
  id[union[complement[IMAGE[composite[VERTSECT[x_]], succ[singleton[0]]]],
  IMAGE[逆[x_], complement[singleton[0]]]]]]]] == 0

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

In[36]:= steptwo[complement[S], x]
Out[36]= equal[cart[IMAGE[composite[VERTSECT[x]]], succ[singleton[0]]],
  VERTSECT[composite[complement[S], x]]] == True

In[37]:= VERTSECT[composite[complement[S], x_]] :=
  cart[IMAGE[composite[VERTSECT[x]]], succ[singleton[0]]], singleton[0]]
The companion rule is:

\[
\text{In[38]} := \text{stepthree[complement[S], x]}
\]

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

\[
\text{In[39]} := \text{IMAGE[composite[complement[S], x_] := cart[image[inverse[IMAGE[x]], succ[singleton[0]]], singleton[0]]}
\]

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

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

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

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

\[
\text{Out[41]} = \{\text{True, True, True, True}\}
\]

**Some good functions**

Some functions are shown to be good in this section. The first is \text{BIGCUP}.

\[
\text{In[42]} := \text{stepone[BIGCUP, x_]}
\]

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

\[
\text{In[43]} := \% /. \text{Equal} \rightarrow \text{SetDelayed}
\]

\[
\text{In[44]} := \text{steptwo[BIGCUP, x]}
\]

\[
\text{Out[44]} = \text{equal[composite[IMAGE[BIGCUP], VERTSECT[x]], VERTSECT[composite[BIGCUP, x]]]} = \text{True}
\]

\[
\text{In[45]} := \text{VERTSECT[composite[BIGCUP, x_]] := composite[IMAGE[BIGCUP], VERTSECT[x]]}
\]

\[
\text{In[46]} := \text{stepthree[BIGCUP, x]}
\]

\[
\text{Out[46]} = \text{IMAGE[composite[BIGCUP, x]]} = \text{composite[IMAGE[BIGCUP], IMAGE[x]]}
\]

\[
\text{In[47]} := \text{IMAGE[composite[BIGCUP, x_]]} = \text{composite[IMAGE[BIGCUP], IMAGE[x]]}
\]

Next: the function \text{POWER}.

\[
\text{In[48]} := \text{stepone[POWER, x_]}
\]

\[
\text{Out[48]} = \text{union[composite[VERTSECT[composite[POWER, x_]], id[complement[domain[VERTSECT[x_]]]]], intersection[complement[VERTSECT[composite[POWER, x_]]], composite[IMAGE[POWER], VERTSECT[x_]]], intersection[composite[complement[IMAGE[POWER]], VERTSECT[x_]], VERTSECT[composite[POWER, x_]]]] = 0}
\]
The function \texttt{SINGLETON = VERTSECT[Id]} is good:

\begin{verbatim}
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
\end{verbatim}

\begin{verbatim}
In[55]:= % /. Equal -> SetDelayed
\end{verbatim}

\begin{verbatim}
In[56]:= steptwo[SINGLETON, x]
Out[56]= equal[composite[IMAGE[SINGLETON], VERTSECT[x]],
  VERTSECT[composite[SINGLETON, x]]] == True
\end{verbatim}

\begin{verbatim}
In[57]:= VERTSECT[composite[SINGLETON, x_]] := composite[IMAGE[SINGLETON], VERTSECT[x]]
\end{verbatim}

\begin{verbatim}
In[58]:= stepthree[SINGLETON, x]
Out[58]= IMAGE[composite[SINGLETON, x]] == composite[IMAGE[SINGLETON], IMAGE[x]]
\end{verbatim}

\begin{verbatim}
In[59]:= IMAGE[composite[SINGLETON, x_]] := composite[IMAGE[SINGLETON], IMAGE[x]]
\end{verbatim}

\textbf{Composites of good relations are good}

If \( g \) and \( h \) are good, then so is \texttt{composite}[g,h]. This can be established by examining the generic case. Suppose:

\begin{verbatim}
In[60]:= VERTSECT[composite[g, x_]] := composite[IMAGE[g], VERTSECT[x]]
In[61]:= VERTSECT[composite[h, x_]] := composite[IMAGE[h], VERTSECT[x]]
\end{verbatim}

One can derive:

\begin{verbatim}
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]]
\end{verbatim}
The desired conclusion follows:

\[ \text{In}[64] := \text{question[composite}[g, h], x] \]
\[ \text{Out}[64] = \text{True} \]

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

\[ \text{In}[65] := \text{composite[inverse[E], IMAGE}[\text{SWAP}], \text{SINGLETON]} \]
\[ \text{Out}[65] = \text{SWAP} \]

As was noted earlier, this is absurd because the complement of the domain of \text{SWAP} is a proper class.

- \text{inverse[SINGLETON]} is not good

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

\[ \text{In}[66] := \text{question[id}[x], \text{cart}[V, \text{complement}[x]]] \]
\[ \text{Out}[66] = \text{member[complement}[x], V] \]

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

\[ \text{In}[67] := \text{composite[SINGLETON, inverse[SINGLETON]]} \]
\[ \text{Out}[67] = \text{id[range[SINGLETON]]} \]

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

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