the class RS[x] of small restrictions, part 3

Johan G. F. Belinfante
revised 2004 August 10

summary

In this third notebook dealing with the theory of RS[x], a study is made of the function 

\[ \text{VERTSECT(RESTRIC T)} = \lambda x, \text{RS}[x]. \]

This function allows one to formulate variable-free equations that capture the facts about RS[x] for the case that \( x \) is a set. An additional bonus of this endeavor is the discovery of a cornucopia of new facts about the relation RESTRIC T which had up to now only been studied cursorily.

the connection between \( \text{RESTRICT} \) and RS[x]

The relation \( \text{RESTRICT} \) is the class of all ordered pairs \( \text{pair}[x, y] \) such that \( y \) is a restriction of \( x \).

\[ \text{class[pair[x, y], exists[z, and subclass[z, Id], equal[y, composite[x, z]]]]} \]

The relation \( \text{RESTRICT} \) was actually defined by the following membership rule, which will shortly be replaced.

\[ \text{member[pair[x, y], RESTRICT]} \]

\[ \text{and[member[x, V], member[y, image[COMPOSE, cart[singleton[x], P[Id]]]]]} \]
The key result needed to connect this relation \texttt{RESTRICT} with \texttt{RS[x]} is this formula:

\begin{verbatim}
In[4]:= ImageComp[COMPOSE, LEFT[x], P[Id]] // Reverse
Out[4]= image[COMPOSE, cart[singleton[x], P[Id]]] =
    intersection[image[V, singleton[x]], RS[x]]
In[5]:= image[COMPOSE, cart[singleton[x_], P[Id]]] :=
    intersection[image[V, singleton[x]], RS[x]]
\end{verbatim}

new membership rule for \texttt{RESTRICT}

The key result found in the preceding section causes the membership rule that defined the relation \texttt{RESTRICT} to be transformed:

\begin{verbatim}
In[6]:= member[x, RESTRICT]
Out[6]= and[equal[composite[first[x], id[domain[second[x]]]], second[x]],
    member[first[x], V]]
\end{verbatim}

For ordered pairs, this formula becomes rather ugly, so we temporarily remove the old rule, derive a separate rule for membership of pairs, and then restore the above rule.

\begin{verbatim}
In[7]:= member[x_, RESTRICT] =.
\end{verbatim}

Lemma.

\begin{verbatim}
In[8]:= SubstTest[member, pair[x, y], composite[Id, z], z \to RESTRICT] // Reverse
Out[8]= and[member[x, V], member[y, V], member[pair[x, y], RESTRICT]] =
    member[pair[x, y], RESTRICT]
In[9]:= and[member[x_, V], member[y_, V], member[pair[x_, y_], RESTRICT]] :=
    member[pair[x, y], RESTRICT]
\end{verbatim}

This is the new membership rule for pairs:

\begin{verbatim}
In[10]:= SubstTest[member, y, image[z, singleton[x]], z \to RESTRICT] // Reverse
Out[10]= member[pair[x, y], RESTRICT] =
    and[equal[y, composite[x, id[domain[y]]]], member[x, V], member[y, V]]
In[11]:= member[pair[x_, y_], RESTRICT] :=
    and[equal[y, composite[x, id[domain[y]]]], member[x, V], member[y, V]]
\end{verbatim}
The following tricky maneuver restores the replacement for the general membership rule:

```
In[12]:= (member[x, composite[rotate[rotate[w]], id[cart[V, P[Id]]], inverse[ FIRST]]]) // AssertTest) /. w -> rotate[ COMPOSE]
```

```
Out[12]= member[x, RESTRICT] :=
and[equal[composite[first[x], id[domain[second[x]]]], second[x]],
member[first[x], V]]
```

```
In[13]:= member[x_, RESTRICT] :=
and[equal[composite[first[x], id[domain[second[x]]]], second[x]],
member[first[x], V]]
```

normalization rule

Subsequent derivations are simplified if one normalizes **RESTRICT**.

```
In[14]:= RESTRICT // VSNormality // Reverse
```

```
Out[14]= composite[intersection[inverse[S]],
UB[union[E, composite[inverse[IMAGE[FIRST]], complement[E], FIRST]]],
IMAGE[id[cart[V, V]]]] := RESTRICT
```

```
In[15]:= composite[intersection[inverse[S]],
UB[union[E, composite[inverse[IMAGE[FIRST]], complement[E], FIRST]]],
IMAGE[id[cart[V, V]]]] := RESTRICT
```

a partial order related to **RESTRICT**

The relation **RESTRICT** is idempotent, antisymmetric, and transitive. It fails to be a partial order because it lacks the reflexive property.

```
In[16]:= {REFLEXIVE[x], ANTISYMMETRIC[x], TRANSITIVE[x], idempotent[x]} /. x -> RESTRICT
```

```
Out[16]= {False, True, True, True}
```

The breakdown of the reflexive law stems from the fact that its domain is larger than its fixed point set.

```
In[17]:= {domain[RESTRICT], range[RESTRICT], fix[RESTRICT]}
```

```
Out[17]= {V, P[cart[V, V]], P[cart[V, V]]}
```
It will now be shown using AssertTest that the restriction of RESTRICT to relations is a partial order. Sequestering RESTRICT from the action of AssertTest speeds this up.

```
In[18]:= (REFLEXIVE[composite[x, id[P[cart[V, V]]]]]) // AssertTest /. x \[Rule] RESTRICT
Out[18]= REFLEXIVE[composite[RESTRICT, id[P[cart[V, V]]]]] := True

In[19]:= REFLEXIVE[composite[RESTRICT, id[P[cart[V, V]]]]] := True

In[20]:= (TRANSITIVE[composite[x, id[P[cart[V, V]]]]]) // AssertTest /. x \[Rule] RESTRICT

In[21]:= TRANSITIVE[composite[RESTRICT, id[P[cart[V, V]]]]] := True
```

Combining these lemmas yields the stated result.

```
In[22]:= SubstTest[and, REFLEXIVE[x], ANTISYMMETRIC[x], TRANSITIVE[x],
                x \[Rule] composite[RESTRICT, id[P[cart[V, V]]]]] // Reverse
Out[22]= PARTIALORDER[composite[RESTRICT, id[P[cart[V, V]]]]] := True

In[23]:= PARTIALORDER[composite[RESTRICT, id[P[cart[V, V]]]]] := True
```

a formula for APPLY[VERTSECT[RESTRICT], x]

The relation RESTRICT is known to be thin:

```
In[24]:= thin[RESTRICT]
Out[24]= True
```

On account of this, the image under RESTRICT of any set is a set:

```
In[25]:= Map[implies[member[x, y], #] &, SubstTest[implies,
           and[thin[z], member[x, V]], member[image[z, x], V], z \[Rule] RESTRICT]]
Out[25]= or[member[image[COMPOSE, cart[x, P[Id]]], V], not[member[x, y]]] := True

In[26]:= or[member[image[COMPOSE, cart[x_, P[Id]]], V], not[member[x_, y_]]] := True
```

In particular, all its vertical sections are sets:

```
In[27]:= member[image[RESTRICT, singleton[x]], V]
Out[27]= True
```
The following consequence of this will be used to simplify a formula for the application of the function \texttt{VERTSECT[RESTRICT]}.

\begin{verbatim}
In[28]:= equal[union[complement[image[V, singleton[x]]],
            complement[image[V, singleton[RS[x]]]],
            complement[image[V, singleton[x]]]]
Out[28]= True

In[29]:= union[complement[image[V, singleton[x_]]],
            complement[image[V, singleton[RS[x_]]]]] :=
            complement[image[V, singleton[x]]]
\end{verbatim}

One now obtains the following clean formula for the application of the function \texttt{VERTSECT[RESTRICT]} to any argument.

\begin{verbatim}
In[30]:= SubstTest[A, image[z, singleton[x]], z \rightarrow VERTSECT[RESTRICT]] // Reverse
Out[30]= APPLY[VERTSECT[RESTRICT], x] =
          union[complement[image[V, singleton[x]]], RS[x]]

In[31]:= APPLY[VERTSECT[RESTRICT], x_] :=
          union[complement[image[V, singleton[x]]], RS[x]]
\end{verbatim}

Note that for sets, this formula simplifies further:

\begin{verbatim}
In[32]:= APPLY[VERTSECT[RESTRICT], setpart[x]]
Out[32]= RS[setpart[x]]
\end{verbatim}

In other words, one can regard \texttt{VERTSECT[RESTRICT]} as the function which takes any set \(x\) to \(RS[x]\). This can also be established as follows:

\begin{verbatim}
In[33]:= lambda[x, RS[x]]
Out[33]= VERTSECT[RESTRICT]
\end{verbatim}

Since \(RS[x]\) is a set whenever \(x\) is a set, the function \texttt{lambda[x, RS[x]] = VERTSECT[RESTRICT]} is total:

\begin{verbatim}
In[34]:= domain[VERTSECT[RESTRICT]]
Out[34]= V
\end{verbatim}

This is equivalent to the statement that the relation \texttt{RESTRICT} is thin.
variable-free formulation of a characterization of functions

One can use the function VERTEX[RESTRICT] to derive variable-free equations that capture that part of the theory of the class RS[x] that applies to the special case that x is a set. For example, the formula for restrictions of a function can be written as follows:

```
In[35]:= composite[VERTEX[RESTRICT], FUNPART]  // VSNormality
Out[35]= composite[VERTEX[RESTRICT], FUNPART] = composite[POWER, FUNPART]
In[36]:= composite[VERTEX[RESTRICT], FUNPART] := composite[POWER, FUNPART]
```

The theorem that RS[x] = P[x] characterizes functions can be written without variables as follows:

```
In[37]:= fix[composite[inverse[POWER], VERTEX[RESTRICT]]] // Normality
Out[37]= fix[composite[inverse[POWER], VERTEX[RESTRICT]]] = FUNS
In[38]:= fix[composite[inverse[POWER], VERTEX[RESTRICT]]] := FUNS
```

Corollary 1.

```
In[39]:= SubstTest[implies, andsubclass[u, v], FUNCTION[v]],
    equal[u, composite[v, id[domain[u]]]],
    {u -> intersection[POWER, VERTEX[RESTRICT]], v -> POWER}]
Out[39]= equal[composite[POWER, id[FUNS]],
       intersection[POWER, VERTEX[RESTRICT]]] = True
In[40]:= intersection[POWER, VERTEX[RESTRICT]] := composite[POWER, id[FUNS]]
```

Corollary 2.

```
In[41]:= SubstTest[rang, intersection[u, v],
    {u -> POWER, v -> VERTEX[RESTRICT]]] // Reverse
Out[41]= fix[composite[POWER, inverse[VERTEX[RESTRICT]]]] = image[POWER, FUNS]
In[42]:= fix[composite[POWER, inverse[VERTEX[RESTRICT]]]] := image[POWER, FUNS]
```
variable-free reformulations of some other facts about RS[x]

The variable-free counterpart of the equation \( U[RS[x]] = \text{thinpart}[x] \) is already known:

\[
\text{In}[43] := \text{composite}[\text{BIGCUP}, \text{VERTSECT}[\text{RESTRICT}]]
\]

\[
\text{Out}[43] = \text{IMAGE}[\text{id}[\text{cart}[V, V]]]
\]

The variable-free version of the formula \( RS[\text{composite}[\text{Id}, x]] = RS[x] \) can be derived as follows:

\[
\text{In}[44] := \text{Map}[\text{composite}[\text{VERTSECT}[\#], \text{SINGLETON}] \&, \\
                 \text{Assoc}[\text{RESTRICT}, \text{IMAGE}[\text{id}[\text{cart}[V, V]]], \text{inverse}[E]]]
\]

\[
\text{Out}[44] = \text{composite}[\text{VERTSECT}[\text{RESTRICT}], \text{IMAGE}[\text{id}[\text{cart}[V, V]]]] = \text{VERTSECT}[\text{RESTRICT}]
\]

A less clever derivation of this result is possible, using a standard technique using \text{VSNormality} and \text{symdif}. This requires two steps:

\[
\text{In}[45] := \text{symdif}[\text{composite}[\text{VERTSECT}[\text{RESTRICT}], \text{IMAGE}[\text{id}[\text{cart}[V, V]]]], \\
                 \text{VERTSECT}[\text{RESTRICT}]] \text{ // VSNormality}
\]

\[
\text{Out}[45] = \text{union}[\text{intersection}[\text{complement}[\text{VERTSECT}[\text{RESTRICT}]]], \\
                 \text{composite}[\text{VERTSECT}[\text{RESTRICT}], \text{IMAGE}[\text{id}[\text{cart}[V, V]]]], \\
                 \text{intersection}[\text{composite}[\text{complement}[\text{VERTSECT}[\text{RESTRICT}]]], \\
                 \text{IMAGE}[\text{id}[\text{cart}[V, V]]], \text{VERTSECT}[\text{RESTRICT}]]] = 0
\]

\[
\text{In}[46] := \% /. \text{Equal} \rightarrow \text{SetDelayed}
\]

\[
\text{In}[47] := \text{SubstTest}[\text{equal}, 0, \text{symdif}[u, v], \\
                 \{u \rightarrow \text{composite}[\text{VERTSECT}[\text{RESTRICT}], \text{IMAGE}[\text{id}[\text{cart}[V, V]]]], \\
                 v \rightarrow \text{VERTSECT}[\text{RESTRICT}]]
\]

\[
\text{Out}[47] = \text{True} = \text{equal}[\text{composite}[\text{VERTSECT}[\text{RESTRICT}], \text{IMAGE}[\text{id}[\text{cart}[V, V]]]], \text{VERTSECT}[\text{RESTRICT}]]
\]

\[
\text{In}[48] := \text{composite}[\text{VERTSECT}[\text{RESTRICT}], \text{IMAGE}[\text{id}[\text{cart}[V, V]]]] := \text{VERTSECT}[\text{RESTRICT}]
\]

reification rule for RS[x]

Reification provides a powerful tool for deriving variable-free counterparts of rewrite rules with variables. A useful reification rule for \text{RS}[x] is obtained as follows:
variable-free results obtained using reification

In this section some examples are presented to illustrate the use of reification to recast results about \texttt{RS[x]} as equations without variables. As a first example, reification is used to eliminate the variable \( x \) from the rewrite rule for \texttt{U[RS[x]]}.

\begin{verbatim}
In[52]:= SubstTest[reify, x, U[f[x]], f \rightarrow RS] // Reverse
\end{verbatim}

This particular equation can also be independently derived from existing facts about \texttt{RESTR} as follows:

\begin{verbatim}
In[53]:= Assoc[inverse[E], inverse[S], RESTR] // Reverse

In[54]:= composite[inverse[E], RESTR] :=
    composite[inverse[E], IMAGE[id[cart[V, V]]]]
\end{verbatim}

The same technique can be used to derive other new properties of \texttt{RESTR}, for instance,

\begin{verbatim}
In[55]:= Assoc[complement[inverse[E]], S, RESTR] // Reverse
\end{verbatim}

This result corresponds to the fact that \( A[RS[x]] = 0 \), as can be seen from the following alternate derivation, using reification:
Aclosure and Uclosure

The facts that RS[x] is closed under arbitrary intersections and under arbitrary unions have simple variable-free formulations:

```
In[58]:= Map[VERTSECT, SubstTest[reify, x, Aclosure[f[x]], f→RS]] // Reverse
Out[58]= composite[ACLOSURE, VERTSECT[RESTRICT]] = VERTSECT[RESTRICT]
In[59]:= composite[ACLOSURE, VERTSECT[RESTRICT]] := VERTSECT[RESTRICT]
In[60]:= Map[VERTSECT, SubstTest[reify, x, Uclosure[f[x]], f→RS]] // Reverse
Out[60]= composite[UCLOSURE, VERTSECT[RESTRICT]] = VERTSECT[RESTRICT]
In[61]:= composite[UCLOSURE, VERTSECT[RESTRICT]] := VERTSECT[RESTRICT]
```

ONEONE property of a restriction of VERTSECT[RESTRICT]

One can see that the function VERTSECT[RESTRICT] fails to be one-to-one because RS[x] for example is equal to RS[composite[Id,x]]. It will be shown that the restriction of the function VERTSECT[RESTRICT] to the class P[cart[V, V]] of all relations is one-to-one. The one-to-one property will be derived by analogy with the following:

```
In[62]:= SubstTest[implies, equal[u, v], equal[U[u], U[v]], (u→RS[x], v→RS[y])]
Out[62]= or[equal[thinpart[x], thinpart[y]], not[equal[RS[x], RS[y]]]] = True
In[63]:= or[equal[thinpart[x_], thinpart[y_]], not[equal[RS[x_], RS[y_]]]] := True
```

Lemma.

```
In[64]:= composite[inverse[VERTSECT[RESTRICT]], inverse[BIGCUP]] // DoubleInverse
Out[64]= composite[inverse[VERTSECT[RESTRICT]], inverse[BIGCUP]] :=
    inverse[IMAGE[id[cart[V, V]]]]
```
In[65]:= composite[inverse[VERTSECT[RESTRICT]], inverse[BIGCUP]] := 
inverse[IMAGE[id[cart[V, V]]]]

In[66]:= SubstTest[implies, subclass[u, v], subclass[IMAGE[w, u], IMAGE[w, v]], 
{u -> Id, v -> composite[inverse[BIGCUP], BIGCUP], 
w -> inverse[cross[VERTSECT[RESTRICT], VERTSECT[RESTRICT]]]}]

Out[66]= subclass[composite[inverse[VERTSECT[RESTRICT]], IMAGE[cart[V, V]]], 
composite[inverse[IMAGE[id[cart[V, V]]]], IMAGE[id[cart[V, V]]]]] := True

In[67]:= % /. Equal -> SetDelayed

For the reverse inclusion one needs the following lemma:

In[68]:= composite[inverse[IMAGE[cart[V, V]]], 
inverse[VERTSECT[RESTRICT]]] // DoubleInverse

Out[68]= composite[inverse[IMAGE[cart[V, V]]], inverse[VERTSECT[RESTRICT]]] := 
inverse[VERTSECT[RESTRICT]]

In[69]:= composite[inverse[IMAGE[cart[V, V]]], inverse[VERTSECT[RESTRICT]]] := 
inverse[VERTSECT[RESTRICT]]

In[70]:= SubstTest[implies, subclass[u, v], subclass[IMAGE[w, u], IMAGE[w, v]], 
{u -> Id, v -> composite[inverse[VERTSECT[RESTRICT]], VERTSECT[RESTRICT]], 
w -> inverse[cross[IMAGE[id[cart[V, V]]], IMAGE[id[cart[V, V]]]]]}]

Out[70]= subclass[composite[inverse[IMAGE[cart[V, V]]], IMAGE[cart[V, V]]], 
composite[inverse[VERTSECT[RESTRICT]], VERTSECT[RESTRICT]]] := True

In[71]:= % /. Equal -> SetDelayed

Putting these facts together yields:

In[72]:= SubstTest[and, subclass[u, v], subclass[v, u], 
{u -> composite[inverse[IMAGE[cart[V, V]]], IMAGE[cart[V, V]]], 
v -> composite[inverse[VERTSECT[RESTRICT]], VERTSECT[RESTRICT]]}]

composite[inverse[VERTSECT[RESTRICT]], VERTSECT[RESTRICT]]]

In[73]:= composite[inverse[VERTSECT[RESTRICT]], VERTSECT[RESTRICT]] := 
composite[inverse[IMAGE[cart[V, V]]], IMAGE[cart[V, V]]]

Corollary.
a formula for image(inverse[S], RS[x])

Lemma.

If \( y \) is a subset of thinpart[x], then \( y \) is contained in the restriction of thinpart[x] to domain[y]. This is the idea behind the following.

The reverse inclusion also holds, so one can sharpen this to an equation:
some new formulas for RESTRICT

The variable-free equation corresponding to this is:

```
In[84]:= Map[composite[VERTSECT[#], SINGLETON] &, Assoc[inverse[S], RESTRICT, inverse[E]]]
```

```
Out[84]= composite[IMAGE[inverse[S]], VERTSECT[RESTRICT]] ==
      composite[POWER, IMAGE[id[cart[V, V]]]]
```

```
In[85]:= composite[IMAGE[inverse[S]], VERTSECT[RESTRICT]] :=
      composite[POWER, IMAGE[id[cart[V, V]]]]
```

The following old formula for RESTRICT is useful for deriving various properties of this relation

```
In[86]:= composite[COMPOSE, id[cart[V, P[Id]]], inverse[FIRST]]
```

```
Out[86]= RESTRICT
```

For example, one can use it to derive the following property of RESTRICT:

```
In[87]:= Assoc[IMAGE[SECOND], COMPOSE, composite[id[cart[V, P[Id]]], inverse[FIRST]]]
```

```
Out[87]= composite[IMAGE[SECOND], RESTRICT] == composite[IMG, inverse[FIRST]]
```

This equation can also be derived a different way using this famous connection between the functions IMG and CART:

```
In[88]:= composite[IMAGE[rotate[Id]]], CART]
```

```
Out[88]= IMG
```

The following alternate derivation is based on this idea:

```
In[89]:= Assoc[IMAGE[rotate[Id]], CART, inverse[FIRST]]
```

```
Out[89]= composite[IMAGE[SECOND], RESTRICT] == composite[IMG, inverse[FIRST]]
```

```
In[90]:= composite[IMAGE[SECOND], RESTRICT] := composite[IMG, inverse[FIRST]]
```
A lemma is needed to get a corresponding result with \texttt{IMAGE[FIRST]} in place of \texttt{IMAGE[SECOND]}.

\begin{verbatim}
In[91]:= composite[IMG, id[cart[P[Id], V]], inverse[SECOND]] // VSNormality
Out[91]= composite[IMG, id[cart[P[Id], V]], inverse[SECOND]] = inverse[S]

In[92]:= composite[IMG, id[cart[P[Id], V]], inverse[SECOND]] := inverse[S]

In[93]:= Assoc[IMAGE[FIRST], COMPOSE, composite[id[cart[V, P[Id]]], inverse[FIRST]]]
Out[93]= composite[IMAGE[FIRST], RESTRICT] = composite[inverse[S], IMAGE[FIRST]]

In[94]:= composite[IMAGE[FIRST], RESTRICT] := composite[inverse[S], IMAGE[FIRST]]
\end{verbatim}

interpreting the formulas

In addition to finding variable-free equations corresponding to properties of \texttt{RS[x]}, one can also try to do the reverse, finding properties of \texttt{RS[x]} that correspond to identities involving \texttt{RESTRICT}. This is less automatic because the variable-free equations only yield results valid when \texttt{x} is a set. One needs to go further to obtain completely general results. Nonetheless, the variable-free equations are useful for suggesting what one must do. For example, the interpretation of the \texttt{IMAGE[SECOND]} formula found in the preceding section is that the class of all ranges of restrictions of \texttt{x} is the range of the function \texttt{IMAGE[x]}. To get a general result of this sort, one needs to use a method that does not require \texttt{x} to be a set. In this case, the following works:

\begin{verbatim}
In[95]:= ImageComp[IMAGE[SECOND], IMAGE[cross[Id, x]], P[Id]] // Reverse
Out[95]= image[IMAGE[SECOND], RS[x]] = range[IMAGE[x]]

In[96]:= image[IMAGE[SECOND], RS[x_]] := range[IMAGE[x]]
\end{verbatim}

The interpretation of the \texttt{IMAGE[FIRST]} formula is that the class of all domains of restrictions of \texttt{x} is the power class of the domain of the thinpart of \texttt{x}. This fact is established as follows:

\begin{verbatim}
In[97]:= Map[equal[P[domain[thinpart[x]]], image[#, P[Id]]] &,
        composite[IMAGE[FIRST], IMAGE[cross[Id, x]]] // VSNormality]
Out[97]= equal[image[IMAGE[FIRST], RS[x]],
        P[intersection[domain[x], domain[VERTSECT[x]]]]] = True
\end{verbatim}
Recall that the domain of the thin part of \( x \) is

\[
\text{domain[thinpart}[x]\text{]}.
\]

When \( x \) is a function, this formula reduces to

\[
\text{image[IMAGE[FIRST], P[funpart}[x]\text{]}.\]

A similar result holds with the \text{funpart} wrapper replaced with \text{thinpart} or \text{setpart}, but it is currently not known whether the wrapper can be removed altogether. The following inclusion holds, but it is currently not known whether the reverse inclusion holds for arbitrary classes:

\[
\text{subclass[image[IMAGE[FIRST], P[x]}, P[domain[x]]\text{]}.\]

\[
\text{True}\]