the class RS[x] of small restrictions, part 2

Johan G. F. Belinfante  
revised 2004 August 9

summary

In this notebook the study of the class RS[x] of small restrictions of x is continued. A number of results are derived, culminating with the theorem that a class x is a function if and only if every subset is a restriction. This result has been the source of several examples in the test suite used to develop the GOEDEL program for many years, but the tools to prove it were not available until now.

a (problematic) rule for domain[thinpart[x]]

To simplify expressions involving thinpart[x], one often needs to know that the domain of thinpart[x] is contained in the domains of x and VERTSECT[x], and one sometimes needs a membership rule for the domain of thinpart[x]. Instead of adding rules for all these facts, the following single rule is added instead. The orientation of this rule is dictated by the fact that turning it around leads to looping.

\[
\text{In[2]}:= \text{domain[thinpart[x]]} = \text{intersection[domain[x], domain[VERTSECT[x]]]}
\]

\[
\text{Out[2]= domain[thinpart[x]]} = \text{intersection[domain[x], domain[VERTSECT[x]]]}
\]

\[
\text{In[3]}:= \text{domain[thinpart[x_]]} = \text{intersection[domain[x], domain[VERTSECT[x]]]}
\]
a conditional simplification rule for RS[x]

After adding all the rewrite rules derived in part 1 to the GOEDEL program, one simple characterization of RS[x] mysteriously ceased working, possibly because the order in which rewrite rules are applied could change when the rules are presented in alphabetical order rather than chronological order. The situation is easily fixed by adding a conditional simplification rule, justified by this observation:

\[
\text{In [4]}: \text{implies}\left[\text{subclass}\left[\text{thinpart}[x], y\right], \text{equal}\left[\text{intersection}\left[\text{RS}[x], P[y]\right], \text{RS}[x]\right]\right]
\]

\[
\text{Out [4]}: \text{True}
\]

This is the new rule:

\[
\text{In [5]}: \text{intersection}\left[\text{RS}[x_\_], P[y_]\right] := \text{RS}[x] ; \text{subclass}\left[\text{thinpart}[x], y\right]
\]

This rule restores the following characterization of RS[x] back to its former glory.

\[
\text{In [6]}: \text{class}\left[\text{w}, \text{equal}\left[\text{w}, \text{composite}\left[\text{x, id}\left[\text{domain}[\text{w}]\right]\right]\right]\right]
\]

\[
\text{Out [6]}: \text{RS}[x]
\]

While we are at it, here is another result that should have been added in part 1.

\[
\text{In [7]}: \text{SubstTest}\left[\text{subclass}, \text{intersection}[u, v], v, \left\{u \rightarrow \text{P}[\text{thinpart}[x]], v \rightarrow \text{invar}\left[\text{composite}\left[\text{id}[x], \text{inverse}[\text{FIRST}], \text{FIRST}\right]\right]\right\}\right]
\]

\[
\text{Out [7]}: \text{subclass}\left[\text{RS}[x], \text{invar}\left[\text{composite}\left[\text{id}[x], \text{inverse}[\text{FIRST}], \text{FIRST}\right]\right]\right] = \text{True}
\]

\[
\text{In [8]}: \text{subclass}\left[\text{RS}[x_\_], \text{invar}\left[\text{composite}\left[\text{id}[x_\_], \text{inverse}[\text{FIRST}], \text{FIRST}\right]\right]\right] := \text{True}
\]

function rule

The rule for the class of small restrictions of functions can be derived from the fact that composite[id[x], inverse[FIRST]] is one-to-one when x is a function, and a conditional rewrite rule for range[IMAGE[x]] that holds when x is one-to-one. Using these facts, one finds:

\[
\text{In [9]}: \text{SubstTest]\left[\text{range, IMAGE[oopart[w]]}, \right.
\text{w} \rightarrow \text{composite}\left[\text{id[funpart[x]}], \text{inverse}[\text{FIRST}]\right]\left.\right]
\]

\[
\text{Out [9]}: \text{RS[funpart[x]]} = \text{P[funpart[x]]}
\]
The same result can be obtained just as quickly, and with less insight being required, using **Normality**.

```
In[10]:= RS[funpart[x]] // Normality
Out[10]= RS[funpart[x]] = P[funpart[x]]
In[11]:= RS[funpart[x_]] := P[funpart[x]]
```

This rewrite rule says that every subset of a function is a small restriction. This rule in particular applies to constant functions, yielding one of several special rules for restrictions of cartesian products.

```
In[12]:= SubstTest[RS, funpart[w], w -> cart[x, singleton[y]]]
Out[12]= RS[cart[x, singleton[y]]] = P[cart[x, singleton[y]]]
In[13]:= RS[cart[x_, singleton[y_]]] := P[cart[x, singleton[y]]]
```

It is unnecessary to add rules for particular cases, such as this one, because one can simply add a conditional rule that applies to all cases. We proceed to do this.

```
In[14]:= SubstTest[implies, and[equal[x, y], equal[RS[y], P[y]]],
   equal[RS[x], P[x]], y -> funpart[x]]
Out[14]= or[equal[P[x], RS[x]], not[FUNCTION[x]]] = True
In[15]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Based on this observation, the following useful conditional rewrite rule for functions is justified:

```
In[16]:= RS[x_] := P[x] ; FUNCTION[x]
```

A converse statement will be derived in a later section of this notebook.

---

restrictions of restrictions

A formula for the small restrictions of a given restriction can be derived:

```
In[17]:= Map[range, composite[IMAGE[cross[Id, x]], IMAGE[id[id[y]]]]] // VSNormality
Out[17]= image[IMAGE[cross[Id, x]], P[id[y]]] = RS[composite[x, id[y]]]
In[18]:= image[IMAGE[cross[Id, x_]], P[id[y_]]] := RS[composite[x, id[y]]]
```
In general, \texttt{subvar} is monotone and \texttt{invar} is antitone, and their intersection is neither. So one does not expect \texttt{RS[x]} to be monotone in general, but for restrictions, there is such a result:

\begin{verbatim}
In[19]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]], {u \rightarrow P[id[y]], v \rightarrow P[Id], w \rightarrow IMAGE[cross[Id, x]]}]
Out[19]= subclass[RS[composite[x, id[y]]], RS[x]] \Rightarrow True

In[20]:= subclass[RS[composite[x, id[y]]], RS[x]] := True
\end{verbatim}

\texttt{RS[thinpart[x]] = RS[x]}

Since \texttt{thinpart[x]} is a restriction of \texttt{x}, the result derived in the preceding section holds for it:

\begin{verbatim}
In[21]:= SubstTest[subclass, RS[composite[x, id[y]]], RS[x], y \rightarrow domain[VERTSECT[x]]]
Out[21]= subclass[RS[thinpart[x]], RS[x]] = True

In[22]:= (% /\ . x \rightarrow x_\_). Equal \rightarrow SetDelayed
\end{verbatim}

In fact the small restrictions of \texttt{x} are exactly the same as those for \texttt{thinpart[x]} as will now be shown.

\begin{verbatim}
In[23]:= SubstTest[subclass, core[u, v], v, {u \rightarrow invar[composite[id[x], inverse[FIRST], FIRST]], v \rightarrow complement[cart[complement[domain[VERTSECT[x]]], V]]}]
Out[23]= subclass[domain[core[invar[composite[id[x], inverse[FIRST], FIRST]], complement[cart[complement[domain[VERTSECT[x]]], V]]], domain[VERTSECT[x]]] = True

In[24]:= (% /\ . x \rightarrow x_\_). Equal \rightarrow SetDelayed
\end{verbatim}

The inclusion in the other direction will be derived from the fact that \texttt{invar} is antitone.

\begin{verbatim}
In[25]:= SubstTest[implies, subclass[u, v], subclass[invar[v], invar[u]], {u \rightarrow composite[id[thinpart[x]], inverse[FIRST], FIRST], v \rightarrow composite[id[x], inverse[FIRST], FIRST]}
Out[25]= subclass[invar[composite[id[x], inverse[FIRST], FIRST]], invar[composite[id[thinpart[x]], inverse[FIRST], FIRST]] = True

In[26]:= (% /\ . x \rightarrow x_\_). Equal \rightarrow SetDelayed
\end{verbatim}
Note that the power class in the following rule for `RS[thinpart[x]]` is the same that occurs in the corresponding rule for `RS[x]` itself.

```math
In[27]:= SubstTest[intersection, invar[composite[id[y], inverse[FIRST], FIRST]],
P[thinpart[y]], y \rightarrow thinpart[x]]
```

```math
Out[27]= intersection[invar[composite[id[thinpart[x]], inverse[FIRST], FIRST]],
P[thinpart[x]]] = RS[thinpart[x]]
```

```math
In[28]:= intersection[invar[composite[id[thinpart[x_]], inverse[FIRST], FIRST]],
P[thinpart[x_]]] := RS[thinpart[x]]
```

The desired inclusion follows:

```math
In[29]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
{u \rightarrow invar[composite[id[x], inverse[FIRST], FIRST]],
v \rightarrow invar[composite[id[thinpart[x]], inverse[FIRST], FIRST]],
w \rightarrow id[P[thinpart[x]]]}]
```

```math
Out[29]= subclass[RS[x], RS[thinpart[x]]] = True
```

```math
In[30]:= (% /. x \rightarrow x__) /. Equal \rightarrow SetDelayed
```

Combing these two inclusions yields an equation:

```math
In[31]:= SubstTest[and, subclass[u, v], subclass[v, u], {u \rightarrow RS[x], v \rightarrow RS[thinpart[x]]}]
```

```math
Out[31]= True = equal[RS[x], RS[thinpart[x]]]
```

```math
In[32]:= RS[thinpart[x_]] := RS[x]
```

restrictions of composites

The result of the preceding section allows one to derive a simple rule for restrictions of composites.

```math
In[33]:= Map[image[#, P[Id]] &,
composite[IMAGE[cross[Id, x]], IMAGE[cross[Id, thinpart[y]]]] // VSNormality]
```

```math
Out[33]= image[IMAGE[cross[Id, x]], RS[y]] = RS[composite[x, thinpart[y]]]
```

```math
In[34]:= image[IMAGE[cross[Id, x_]], RS[y_]] := RS[composite[x, thinpart[y]]]
```

In particular, for functions one obtains:
another formula for restrictions of restrictions

In general the composite of \text{IMAGE[cross[x,Id]]} and \text{IMAGE[cross[Id,y]]} need not be equal to \text{IMAGE[cross[x,y]]}.

Even in the special case that \( x \) is an identity function, equality does not hold in general:

For the special case that \( y \) is thin, however, one does have a simple rule:

For the special case that \( x \) is replaced by an identity function, one obtains:

Lemma:
FUNCTION characterization

A proof that functions are characterized by $P[x] = RS[x]$ will now be presented. The
starting point is this special formula for cartesian products whose domains are
singletons:

$$
In[47]:= \text{SubstTest}[image, IMAGE[cross[Id, z]],
   P[Id], z \rightarrow \text{cart[singleton}[x], y]] // Reverse
Out[47]= \text{RS}[\text{cart}[\text{singleton}[x], y]] = \text{pairset}[0, \text{cart}[\text{singleton}[x], y]]
$$

Here is a companion formula for restrictions of restrictions:

$$
In[49]:= \text{ImageComp}[\text{IMAGE}[\text{cross}[Id, x]], \text{IMAGE}[\text{id}[\text{cart}[y, V]]], P[Id]]
Out[49]= \text{image}[\text{IMAGE}[\text{cross}[\text{id}[y], x]], P[Id]] = \text{RS}[\text{composite}[x, id[y]]]
$$

A cartesian product is a singleton if and only if it is the cartesian product of two
singletons:

$$
In[51]:= \text{member}[\text{cart}[x, y], \text{range}[\text{SINGLETONE}]] // \text{AssertTest}
Out[51]= \text{member}[\text{cart}[x, y], \text{range}[\text{SINGLETONE}]]
and[\text{member}[x, \text{range}[\text{SINGLETONE}]], \text{member}[y, \text{range}[\text{SINGLETONE}]]]
$$

This result is applied as follows:
The next step is to eliminate the variable \( y \) in the standard way.

One also needs to know that if every subclass of a class is a restriction, then that class must be a relaton:

It just remains to tie up all the loose ends:

The converse also holds, so one can add the following rewrite rule: