RESTRICT

Johan G. F. Belinfante
2003 June 3

```
In[1]:= << goedel52.r87; << tools.m

        :Package Title: goedel52.r87       2003 May 30 at 9:50 a.m.

        It is now:  2003 Jun 3 at 10:52

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 May 21

        weightlimit = 40
```

■ introduction

The relation **RESTRICT** introduced in this notebook consists of all ordered pairs whose second coordinate is a restriction of the first. It will be shown, among other things, that this relation is thin and transitive. The following description was discovered 2001 May 22.

```
In[2]:= class[pair[x, y], exists[z, and[subclass[z, Id], equal[y, composite[x, z]]]]]

Out[2]= composite[COMPOSE, id[cart[V, P[Id]]], inverse[FIRST]]
```

■ membership rule

The definition of the relation **RESTRICT** is here formally specified by the following membership rule:

```
In[3]:= member[x_, RESTRICT] :=
          member[second[x], image[COMPOSE, cart[singleton[first[x]], P[Id]]]]
```

No special membership rule is needed for ordered pairs:

```
In[4]:= member[pair[x, y], RESTRICT]

Out[4]= and[member[x, V], member[y, image[COMPOSE, cart[singleton[x], P[Id]]]]]
```

■ the class RESTRICT is a relation

To derive the fact that **RESTRICT** is a relation, the following generally useful lemma is employed:

```
In[5]:= or[member[first[x], V], not[member[second[x], y]]] // AssertTest

Out[5]= or[member[first[x], V], not[member[second[x], y]]] == True
```

```
In[6]:= or[member[first[x_], V], not[member[second[x_], y_]]] := True
```

This lemma makes it easy to derive the fact that **RESTRICT** is a relation.

```
In[7]:= Map[equal[V, #] &, SubstTest[class, x,
           implies[member[x, y], member[first[x], V]], y -> RESTRICT]] // Reverse
Out[7]= subclass[RESTRICT, cart[V, V]] == True

In[8]:= subclass[RESTRICT, cart[V, V]] := True
```

The following corollary is a useful simplification rule:

```
In[9]:= equal[composite[Id, RESTRICT], RESTRICT]
Out[9]= True

In[10]:= composite[Id, RESTRICT] := RESTRICT
```

■ an explicit equation for RESTRICT

The formula for **RESTRICT** discovered in 2001 is verified as follows:

```
In[11]:= Map[equal[0, #] &, symdif[RESTRICT,
            composite[COMPOSE, id[cart[V, P[Id]]], inverse[FIRST]]] // VSNormality]
Out[11]= equal[RESTRICT, composite[COMPOSE, id[cart[V, P[Id]]], inverse[FIRST]]] == True

In[12]:= composite[COMPOSE, id[cart[V, P[Id]]], inverse[FIRST]] := RESTRICT
```

The calculation done in the introduction now yields:

```
In[13]:= class[pair[x, y], exists[z, and[subclass[z, Id], equal[y, composite[x, z]]]]]
Out[13]= RESTRICT
```

The inverse rewrite rule is also added:

```
In[14]:= composite[FIRST, id[cart[V, P[Id]]], inverse[COMPOSE]] // DoubleInverse
Out[14]= composite[FIRST, id[cart[V, P[Id]]], inverse[COMPOSE]] == inverse[RESTRICT]

In[15]:= composite[FIRST, id[cart[V, P[Id]]], inverse[COMPOSE]] := inverse[RESTRICT]
```

These equations are useful for deriving facts about the relation **RESTRICT**. For example:

```
In[16]:= Assoc[COMPOSE, cross[IMAGE[id[cart[V, V]]], Id],
          composite[id[cart[V, P[Id]]], inverse[FIRST]]]
Out[16]= composite[RESTRICT, IMAGE[id[cart[V, V]]]] == RESTRICT

In[17]:= composite[RESTRICT, IMAGE[id[cart[V, V]]]] := RESTRICT
```

The following generalizations are easily obtained:

```
In[18]:= Assoc[composite[COMPOSE, id[cart[V, P[Id]]]], inverse[FIRST], id[x]]

Out[18]= composite[COMPOSE, id[cart[x, P[Id]]], inverse[FIRST]] == composite[RESTRICT, id[x]]

In[19]:= composite[COMPOSE, id[cart[x_, P[Id]]], inverse[FIRST]] := composite[RESTRICT, id[x]]

In[20]:= Assoc[id[x], FIRST, composite[id[cart[V, P[Id]]], inverse[COMPOSE]]] // Reverse

Out[20]= composite[FIRST, id[cart[x, P[Id]]], inverse[COMPOSE]] ==
          composite[id[x], inverse[RESTRICT]]

In[21]:= composite[FIRST, id[cart[x_, P[Id]]], inverse[COMPOSE]] :=
          composite[id[x], inverse[RESTRICT]]
```

■ co−restriction

The relation of co−restriction is related to restriction via inversion. The only difference is that the identity is on the left rather than the right:

```
In[22]:= class[pair[x, y], exists[z, and[subclass[z, Id], equal[y, composite[z, x]]]]]

Out[22]= composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND]]
```

The following step is similar to what was done above for **RESTRICT**.

```
In[23]:= Assoc[COMPOSE, cross[Id, IMAGE[id[cart[V, V]]]],
          composite[id[cart[P[Id], V]], inverse[SECOND]]]

Out[23]= composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND], IMAGE[id[cart[V, V]]]] ==
          composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND]]

In[24]:= composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND], IMAGE[id[cart[V, V]]]] :=
          composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND]]
```

This yields a formula relating co−restriction to restriction:

```
In[25]:= Assoc[IMAGE[SWAP], COMPOSE,
           composite[id[cart[V, P[Id]]], inverse[FIRST], IMAGE[SWAP]]] // Reverse

Out[25]= composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND]] ==
          composite[IMAGE[SWAP], RESTRICT, IMAGE[SWAP]]

In[26]:= composite[COMPOSE, id[cart[P[Id], V]], inverse[SECOND]] :=
          composite[IMAGE[SWAP], RESTRICT, IMAGE[SWAP]]
```

■ domain

The domain is easily derived:

```
In[27]:= IminComp[COMPOSE, composite[id[cart[V, P[Id]]], inverse[FIRST]], V]

Out[27]= domain[RESTRICT] == V

In[28]:= domain[RESTRICT] := V
```

■ relation to S and the thin−ness property

The orientation of the following rewrite rule is somewhat tentative. It is unclear whether it should be best turned around, or left as is.

```
In[29]:= ImageComp[COMPOSE, composite[id[cart[V, P[Id]]], inverse[FIRST]], x]

Out[29]= image[RESTRICT, x] == image[COMPOSE, cart[x, P[Id]]]

In[30]:= image[RESTRICT, x_] := image[COMPOSE, cart[x, P[Id]]]
```

This formula is needed to show that the restriction is related to the subclass relation:

```
In[31]:= Map[equal[0, #] &, dif[RESTRICT, inverse[S]] // VSNormality]

Out[31]= subclass[RESTRICT, inverse[S]] == True

In[32]:= subclass[RESTRICT, inverse[S]] := True
```

It follows that **RESTRICT** is a thin relation; that is, its vertical sections are sets:

```
In[33]:= SubstTest[implies, and[subclass[x, y], thin[y]],
           thin[x], {x -> RESTRICT, y -> inverse[S]}]

Out[33]= equal[V, domain[VERTSECT[RESTRICT]]] == True

In[34]:= domain[VERTSECT[RESTRICT]] := V
```

■ range

The following obervation may help to understand some formulas in this section.

```
In[35]:= equal[composite[Id, x], composite[x, id[domain[x]]]]

Out[35]= True
```

This is a relational version of the same observation:

```
In[36]:= composite[COMPOSE, id[composite[IMAGE[DUP], IMAGE[FIRST]]], inverse[FIRST]]

Out[36]= IMAGE[id[cart[V, V]]]
```

The first step in deriving a formula for the range of **RESTRICT** uses a similar construction:

```
In[37]:= ImageComp[composite[COMPOSE, cross[Id, IMAGE[DUP]]],
           composite[cross[Id, IMAGE[FIRST]], DUP], V] // Reverse

Out[37]= image[COMPOSE, composite[IMAGE[DUP], IMAGE[FIRST]]] == P[cart[V, V]]

In[38]:= image[COMPOSE, composite[IMAGE[DUP], IMAGE[FIRST]]] := P[cart[V, V]]
```

This result is used to derive a formula for the range:

```
In[39]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
             {u -> composite[IMAGE[DUP], IMAGE[FIRST]], v -> cart[V, P[Id]], w -> COMPOSE}]

Out[39]= subclass[P[cart[V, V]], image[COMPOSE, cart[V, P[Id]]]] == True

In[40]:= subclass[P[cart[V, V]], image[COMPOSE, cart[V, P[Id]]]] := True

In[41]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> P[cart[V, V]], v -> image[COMPOSE, cart[V, P[Id]]]}]

Out[41]= True == equal[image[COMPOSE, cart[V, P[Id]]], P[cart[V, V]]]

In[42]:= image[COMPOSE, cart[V, P[Id]]] := P[cart[V, V]]

In[43]:= ImageComp[COMPOSE, composite[id[cart[V, P[Id]]], inverse[FIRST]], V]

Out[43]= range[RESTRICT] == P[cart[V, V]]

In[44]:= range[RESTRICT] := P[cart[V, V]]
```

■ a formula for fix[RESTRICT]

A general lemma is useful here:

```
In[45]:= SubstTest[implies, subclass[w, z],
           subclass[composite[x, w, y], composite[x, z, y]], {w -> id[u], z -> id[v]}]

Out[45]= or[not[subclass[u, v]], subclass[composite[x, id[u], y], composite[x, id[v], y]]] == True
```

The following general rewrite rule has triple blanks to cover the cases that **x** or **y** might be missing.

```
In[47]:= or[not[subclass[u_, v_]],
            subclass[composite[x___, id[u_], y___], composite[x___, id[v_], y___]]] := True
```

This lemma is used to derive a lower bound for the relalation **RESTRICT**.

```
In[48]:= SubstTest[implies, subclass[u, v],
           subclass[composite[x, id[u], y], composite[x, id[v], y]],
             {u -> composite[IMAGE[DUP], IMAGE[FIRST]],
           v -> cart[V, P[Id]], x -> COMPOSE, y -> inverse[FIRST]}]

Out[48]= subclass[IMAGE[id[cart[V, V]]], RESTRICT] == True

In[49]:= subclass[IMAGE[id[cart[V, V]]], RESTRICT] := True
```

As a byproduct, one gets a lower limit on **fix[RESTRICT]**.

```
In[50]:= SubstTest[implies, subclass[u, v],
           subclass[fix[u], fix[v]], {u -> IMAGE[id[cart[V, V]]], v -> RESTRICT}]

Out[50]= subclass[P[cart[V, V]], fix[RESTRICT]] == True

In[51]:= subclass[P[cart[V, V]], fix[RESTRICT]] := True
```

The range provides an upper bound.

```
In[52]:= SubstTest[subclass, fix[x], range[x], x -> RESTRICT]

Out[52]= subclass[U[fix[RESTRICT]], cart[V, V]] == True

In[53]:= subclass[U[fix[RESTRICT]], cart[V, V]] := True
```

Combining these bounds yields an equation:

```
In[54]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> P[cart[V, V]], v -> fix[RESTRICT]}]

Out[54]= True == equal[fix[RESTRICT], P[cart[V, V]]]

In[55]:= fix[RESTRICT] := P[cart[V, V]]
```

■ an ASSOC formula

Deriving the fact that **RESTRICT** is transitive requires some ingenuity. The starting point is this property of the function **ASSOC.**

```
In[56]:= Map[composite[#, ASSOC] &,
          composite[ASSOC, cross[cross[x, y], z], inverse[ASSOC]] // VSTriNormality]

Out[56]= composite[ASSOC, cross[cross[x, y], z]] == composite[cross[x, cross[y, z]], ASSOC]
```

The following orientation for this rule is tentative:

```
In[57]:= composite[ASSOC, cross[cross[x_, y_], z_]] := composite[cross[x, cross[y, z]], ASSOC]
```

Since the function **ASSOC** is one−to−one,it is unclear how this should be oriented. The usual rule of thumb for orienting such equations is that functions should move to the right and inverse functions to the left, but for one−to−one functions, these rules of thumb are contradictory.

■ transitivity from a formula for repeated restriction

This section is devoted to obtaining the transitive property of **RESTRICT** as a consequence of a variable−free version of the following formula:

```
In[58]:= composite[x, id[y], id[z]]

Out[58]= composite[x, id[intersection[y, z]]]
```

The abbreviation **IDP = IMAGE[DUP]** is used here:

```
In[59]:= symdif[composite[COMPOSE, cross[COMPOSE, Id], cross[cross[Id, IDP], IDP]],
           composite[COMPOSE, cross[Id, IDP], cross[Id, CAP], ASSOC]] // VSTriNormality

Out[59]= union[composite[intersection[
           composite[COMPOSE, cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]],
           composite[complement[COMPOSE],
            cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]], ASSOC]],
          id[cart[V, V]]], composite[intersection[composite[complement[COMPOSE],
            cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]],
           composite[COMPOSE, cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]],
            ASSOC]], id[cart[V, V]]]] == 0
```

This is added as a temporary rewrite rule:

```
In[60]:= union[composite[intersection[
           composite[COMPOSE, cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]],
           composite[complement[COMPOSE],
            cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]], ASSOC]],
          id[cart[V, V]]], composite[intersection[composite[complement[COMPOSE],
            cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]],
           composite[COMPOSE, cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]],
            ASSOC]], id[cart[V, V]]]] := 0
```

Since their symmetric difference is empty, these are equal:

```
In[61]:= SubstTest[equal, 0, composite[symdif[u, v], id[cart[V, V]]],
           {u -> composite[COMPOSE, cross[COMPOSE, Id], cross[cross[Id, IDP], IDP]],
           v -> composite[COMPOSE, cross[Id, IDP], cross[Id, CAP], ASSOC]}]

Out[61]= True == equal[
           composite[COMPOSE, cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]],
           composite[COMPOSE, cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]], ASSOC]]
```

This establishes the equation:

```
In[62]:= Equal[
          composite[COMPOSE, cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]],
          composite[COMPOSE, cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]], ASSOC]]

Out[62]= composite[COMPOSE, cross[composite[COMPOSE, cross[Id, IMAGE[DUP]]], IMAGE[DUP]]] ==
          composite[COMPOSE, cross[Id, composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]], ASSOC]
```

Transitivity follows as a corollary of this equation:

```
In[63]:= Map[composite[#, cross[cross[Id, y], y], cross[inverse[FIRST], Id], inverse[FIRST]] &,
           %] /. y -> IMAGE[FIRST]

Out[63]= composite[RESTRICT, RESTRICT] == RESTRICT


In[64]:= composite[RESTRICT, RESTRICT] := RESTRICT
```