# vertical sections of equivalence relations

*Johan G. F. Belinfante*
*2004 December 16*

```
In[1]:= SetDirectory["i:"]; << goedel64.15a; << tools.m
```

```
:Package Title: goedel64.15a              2004 December 15 at 4:10 p.m.

It is now:  2004 Dec 16 at 21:53

Loading Simplification Rules

TOOLS.M                       Revised 2004 December 16

weightlimit = 40
```

---

## summary

Any thin equivalence relation can be written as the composite of the inverse of the associated canonical projection with itself. For small equivalence relations, a variable-free equational formulation of this fact is derived, and from it a variable-free equation is obtained for the corollary that says that any small equivalence relation is the composite of the inverse of a function with itself.

---

## derivation

### Lemma

```
In[2]:= equal[composite[thinpart[eqv[x]], id[fix[eqv[x]]]], thinpart[eqv[x]]]
```

```
Out[2]= True
```

```
In[3]:= composite[thinpart[eqv[x_]], id[fix[eqv[x_]]]] := thinpart[eqv[x]]
```

### Basic result:

```
In[4]:= (composite[id[fix[y]], inverse[VERTSECT[y]], VERTSECT[y], id[fix[y]]] //
           RelnNormality) /. y → eqv[x]
```

```
Out[4]= composite[id[fix[eqv[x]]], inverse[VERTSECT[eqv[x]]],
          VERTSECT[eqv[x]], id[fix[eqv[x]]]] == thinpart[eqv[x]]
```

```
In[5]:= composite[id[fix[eqv[x_]]], inverse[VERTSECT[eqv[x_]]],
          VERTSECT[eqv[x_]], id[fix[eqv[x_]]]] := thinpart[eqv[x]]
```

The rest of this notebook is concerned with removing the variable **x** for the special case that **x** is a set.

---

## a simplification rule

The presence of **VS** allows one to replace **IMAGE[SWAP]** with **INVERSE**.

```
In[6]:= Assoc[composite[id[IMAGE[SWAP]], inverse[FIRST]], id[P[cart[V, V]]], VS]

Out[6]= composite[id[IMAGE[SWAP]], inverse[FIRST], VS] ==
         composite[id[INVERSE], inverse[FIRST], VS]

In[7]:= composite[id[IMAGE[SWAP]], inverse[FIRST], VS] :=
         composite[id[INVERSE], inverse[FIRST], VS]
```

Similarly:

```
In[8]:= Assoc[SWAP, id[IMAGE[SWAP]], composite[inverse[FIRST], VS]] // Reverse

Out[8]= composite[id[inverse[IMAGE[SWAP]]], inverse[SECOND], VS] ==
         composite[id[INVERSE], inverse[SECOND], VS]

In[9]:= composite[id[inverse[IMAGE[SWAP]]], inverse[SECOND], VS] :=
         composite[id[INVERSE], inverse[SECOND], VS]
```

---

## eliminating the variable x

The elimination of the variable **x** is accomplished as follows:

```
In[10]:= SubstTest[class, x,
           equal[image[u, singleton[setpart[x]]], image[v, singleton[setpart[x]]]],
           {u -> composite[IMAGE[composite[FIRST, id[cart[V, Id]]]],
              CROSS, DUP, VS, EQUIV], v → EQUIV}] // Reverse

Out[10]= image[inverse[EQUIV],
           fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]]] == V

In[11]:= % /. Equal → SetDelayed
```

Eliminating the wrapper-related function **EQUIV** yields an inclusion:

```
In[12]:= Map[subclass[EQV, #] &, ImageComp[EQUIV, inverse[EQUIV],
           fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]]]]

Out[12]= subclass[EQV,
           fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]]] == True

In[13]:= % /. Equal → SetDelayed
```

The reverse inequality will be derived shortly.

---

## corollary

In this section a variable-free statement is derived that says that any equivalence relation is the composite of the inverse of a function with itself. The derivation is based on this inclusion:

```
In[14]:= subclass[VS, cart[V, FUNS]]

Out[14]= True
```

The following function is needed.

```
In[15]:= abstract[x, fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], x]]]

Out[15]= composite[FIRST, id[composite[SECOND, id[INVERSE], inverse[COMPOSE]]]]
```

The main step uses the above two facts:

```
In[16]:= SubstTest[implies, subclass[u, v],
          subclass[image[w, u], image[w, v]], {u → VS, v → cart[V, FUNS], w ->
            composite[FIRST, id[composite[SECOND, id[INVERSE], inverse[COMPOSE]]]]}]

Out[16]= subclass[fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]],
           image[COMPOSE, composite[id[FUNS], INVERSE]]] == True

In[17]:= % /. Equal → SetDelayed
```

Th corollary follows immediately:

```
In[18]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → EQV, v -> fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]],
           w -> image[COMPOSE, composite[id[FUNS], INVERSE]]}]

Out[18]= subclass[EQV, image[COMPOSE, composite[id[FUNS], INVERSE]]] == True

In[19]:= % /. Equal → SetDelayed
```

The reverse inclusion also holds, so one obtains an equation:

```
In[20]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> EQV, v -> image[COMPOSE, composite[id[FUNS], INVERSE]]}]

Out[20]= True == equal[EQV, image[COMPOSE, composite[id[FUNS], INVERSE]]]

In[21]:= image[COMPOSE, composite[id[FUNS], INVERSE]] := EQV
```

---

## another equation

The final task is to use this corollary to replace an inclusion derived in an earlier section with an equation. The following step just repeats an earlier one, but now can take advantage of the rewrite rule derived in the preceding section.

```
In[22]:= SubstTest[implies, subclass[u, v],
          subclass[image[w, u], image[w, v]], {u → VS, v → cart[V, FUNS], w ->
            composite[FIRST, id[composite[SECOND, id[INVERSE], inverse[COMPOSE]]]]}]

Out[22]= subclass[fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]], EQV] ==
          True

In[23]:= % /. Equal → SetDelayed
```

The following equation states that (small) equivalence relations can be characterized by the fact that they are equal to the composite of the inverse of the associated canonical projection with itself.

```
In[24]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> EQV, v -> fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]]}]

Out[24]= True == equal[EQV, fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]]]

In[25]:= fix[composite[COMPOSE, id[INVERSE], inverse[SECOND], VS]] := EQV
```