# domain and range of projections

Johan G. F. Belinfante
2006 June 10

```
In[1]:= SetDirectory["l:"]; << goedel82.10a; << tools.m

        :Package Title: goedel82.10a         2006 June 10 at 7:45 a.m.

        It is now:  2006 Jun 10 at 12:44

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 June 6

        weightlimit = 40
```

## summary

In this notebook formulas for **image[DORA, PROJ]** and **image[DORA, UNOPS]** are derived. An inclusion in one direction has already been established:

```
In[2]:= SubstTest[subclass, w, intersection[x, y],
          {w -> image[DORA, PROJ], x → range[DORA], y -> inverse[S]}]

Out[2]= subclass[image[DORA, PROJ],
          union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]] == True

In[3]:= % /. Equal → SetDelayed
```

To show that the reverse inclusion also holds, a lower bound on **PROJ** will be used. The idea is to consider functions which are the union of an identity function and a constant function. If the domains of the two parts are disjoint, and if the value of the constant part is in the range of the identity, such a function is a projection. The empty function is not included in this class of functions and needs to be considered separately, but this poses no problem:

```
In[4]:= SubstTest[implies, subclass[x, y],
          subclass[image[w, x], image[w, y]], {w → DORA, x → set[0], y → PROJ}]

Out[4]= member[pair[0, 0], image[DORA, PROJ]] == True

In[5]:= % /. Equal → SetDelayed
```

## a family of projections

The union of functions with disjoint domains is a function:

```
In[6]:= SubstTest[implies, and[FUNCTION[u], FUNCTION[v], disjoint[domain[u], domain[v]]],
         FUNCTION[union[u, v]], {u -> id[y], v -> cart[dif[x, y], set[z]]}]

Out[6]= FUNCTION[union[cart[intersection[x, complement[y]], set[z]], id[y]]] == True

In[7]:= FUNCTION[union[cart[intersection[x_, complement[y_]], set[z_]], id[y_]]] := True
```

For idempotence, a lemma is needed:

```
In[8]:= implies[member[z, y],
          idempotent[union[cart[intersection[x, complement[y]], set[z]], id[y]]]] // AssertTest

Out[8]= or[equal[union[cart[intersection[x, complement[y]], set[z]], id[y]],
           union[cart[intersection[x, complement[y]], union[intersection[y, set[z]],
              intersection[x, complement[image[V, intersection[y, set[z]]]], set[z]]]],
            id[y]]], not[member[z, y]]] == True

In[9]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

## application of reify

Observation:

```
In[11]:= implies[and[member[z, y], member[x, V], member[y, V],
           equal[w, union[id[y], cart[dif[x, y], set[z]]]]], member[w, PROJ]]

Out[11]= True
```

The condition in this observation is represented by the following relation:

```
In[12]:= class[pair[pair[x, y], z], member[z, y]]

Out[12]= composite[inverse[E], SECOND]
```

If the rest of the expression is treated with **reify**, one obtains:

```
In[16]:= reify[x, union[id[y], cart[dif[x, y], set[z]]] /.
           {x → first[first[x]], y → second[first[x]], z → second[x]}]

Out[16]= union[cart[complement[cart[cart[V, V], V]], Id],
          composite[inverse[E], IMAGE[DUP], SECOND, FIRST],
          cross[composite[inverse[E], DIF], Id]]
```

Applying **VERTSECT**, and cleaning up the resulting expression leads one to the following function, for which a temporary name will be given:

```
In[23]:= temp =
           composite[CUP, intersection[composite[inverse[SECOND], CART, cross[DIF, SINGLETON]],
             composite[inverse[FIRST], IMAGE[DUP], SECOND, FIRST]]];
```

```
In[60]:= range[composite[temp, id[composite[inverse[E], SECOND]]]]

Out[60]= image[CUP, composite[CART, intersection[composite[inverse[FIRST], DISJOINT],
           composite[inverse[SECOND], inverse[E], IMAGE[SINGLETON]]], inverse[IMAGE[DUP]]]]
```

In the next section it is shown that this special class of functions is indeed contained in **PROJ**.

## inclusion in PROJ

Lemma.

```
In[30]:= composite[intersection[composite[complement[inverse[E]], CUP],
           composite[inverse[E], SECOND]], id[composite[Id, complement[S]]]] // TriNormality

Out[30]= composite[intersection[composite[complement[inverse[E]], CUP],
           composite[inverse[E], SECOND]], id[composite[Id, complement[S]]]] == 0

In[31]:= % /. Equal → SetDelayed

In[33]:= dif[composite[inverse[E], SECOND], image[inverse[temp], PROJ]] // VSTriNormality

Out[33]= intersection[composite[inverse[E], SECOND], composite[inverse[SINGLETON],
           complement[composite[SECOND, intersection[composite[inverse[FIRST], DIF],
             composite[inverse[CART], image[inverse[CUP], PROJ], IMAGE[DUP], SECOND]]]]]] == 0

In[34]:= % /. Equal → SetDelayed

In[35]:= dif[composite[inverse[E], SECOND], image[inverse[temp], PROJ]]

Out[35]= 0

In[36]:= SubstTest[equal, 0, dif[u, v],
           {u → composite[inverse[E], SECOND], v -> image[inverse[temp], PROJ]}] // Reverse

Out[36]= subclass[composite[inverse[E], SECOND],
           composite[inverse[SINGLETON], SECOND, intersection[composite[inverse[FIRST], DIF],
             composite[inverse[CART], image[inverse[CUP], PROJ], IMAGE[DUP], SECOND]]] == True

In[37]:= % /. Equal → SetDelayed
```

Lemma.

```
In[38]:= SubstTest[composite, funpart[x], inverse[funpart[x]], x → temp]

Out[38]= composite[CUP, intersection[composite[inverse[SECOND], CART, cross[DIF, SINGLETON]],
           composite[inverse[FIRST], IMAGE[DUP], SECOND, FIRST]], intersection[
           composite[cross[inverse[DIF], inverse[SINGLETON]], inverse[CART], SECOND],
           composite[inverse[FIRST], inverse[SECOND], inverse[IMAGE[DUP]], FIRST]],
          inverse[CUP]] == id[image[CUP, composite[CART, id[cart[V, range[SINGLETON]]],
            inverse[FIRST], DISJOINT, inverse[IMAGE[DUP]]]]]

In[39]:= % /. Equal → SetDelayed
```

```
In[40]:= ImageComp[temp, inverse[temp], composite[inverse[E], SECOND]] // Reverse

Out[40]= image[CUP, composite[CART, intersection[composite[inverse[FIRST], DIF],
           composite[inverse[SECOND], SECOND, id[cart[V, range[SINGLETON]]],
            intersection[composite[inverse[FIRST], DIF], composite[inverse[CART],
              image[inverse[CUP], composite[inverse[E], SECOND]], IMAGE[DUP], SECOND]]],
          inverse[SECOND], inverse[IMAGE[DUP]]]] == intersection[
         composite[inverse[E], SECOND],
         image[CUP, composite[CART, id[cart[V, range[SINGLETON]]],
           inverse[FIRST], DISJOINT, inverse[IMAGE[DUP]]]]]

In[41]:= % /. Equal → SetDelayed

In[42]:= SubstTest[implies, subclass[x, image[inverse[funpart[y]], z]],
          subclass[image[funpart[y], x], z],
          {x → composite[inverse[E], SECOND], y → temp, z → PROJ}]

Out[42]= subclass[image[CUP, composite[CART,
           intersection[composite[inverse[FIRST], DISJOINT], composite[inverse[SECOND],
             inverse[E], IMAGE[SINGLETON]]], inverse[IMAGE[DUP]]]], PROJ] == True

In[43]:= % /. Equal → SetDelayed
```

The only thing left to do is to compute the image of **DORA** for this special class of projections. This is done in the next section.

---

## the final fort

First of three lemmas needed for the next step:

```
In[45]:= twist[composite[intersection[composite[inverse[FIRST], DISJOINT], composite[
              inverse[SECOND], inverse[E], IMAGE[SINGLETON]]], inverse[DUP]]] // TriNormality

Out[45]= twist[composite[intersection[composite[inverse[FIRST], DISJOINT],
           composite[inverse[SECOND], inverse[E], IMAGE[SINGLETON]]], inverse[DUP]]] ==
         composite[id[composite[inverse[E], IMAGE[SINGLETON]]],
          inverse[FIRST], FIRST, id[DISJOINT]]

In[46]:= % /. Equal → SetDelayed
```

Second lemma:

```
In[48]:= composite[CUP,
           id[composite[inverse[E], IMAGE[id[complement[set[0]]]], IMAGE[SINGLETON]]],
           inverse[FIRST], FIRST, id[composite[id[complement[set[0]]], DISJOINT]],
           inverse[CUP]] // RelnRenormality

Out[48]= composite[CUP,
           id[composite[inverse[E], IMAGE[id[complement[set[0]]]], IMAGE[SINGLETON]]],
           inverse[FIRST], FIRST, id[composite[id[complement[set[0]]], DISJOINT]],
           inverse[CUP]] == composite[id[complement[set[0]]], inverse[PS]]
```

```
% /. Equal → SetDelayed
```

Third lemma.

```
In[51]:= Map[inverse, union[composite[PS, id[complement[set[0]]]],
           id[complement[set[0]]]] // RelnRenormality]

Out[51]= union[composite[id[complement[set[0]]], inverse[PS]], id[complement[set[0]]]] ==
          composite[id[complement[set[0]]], inverse[S]]

In[52]:= % /. Equal → SetDelayed

In[53]:= ImageComp[DORA, temp, composite[inverse[E], SECOND]] // Reverse

Out[53]= image[DORA,
           image[CUP, composite[CART, intersection[composite[inverse[FIRST], DISJOINT],
              composite[inverse[SECOND], inverse[E], IMAGE[SINGLETON]]],
             inverse[IMAGE[DUP]]]]] == composite[id[complement[set[0]]], inverse[S]]

In[54]:= % /. Equal → SetDelayed

In[56]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → image[temp, composite[inverse[E], SECOND]], v → PROJ, w → DORA}]

Out[56]= subclass[composite[id[complement[set[0]]], inverse[S]], image[DORA, PROJ]] == True

In[57]:= % /. Equal → SetDelayed
```

The final step is to combine this with the inclusion in the reverse direction.

```
In[58]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → image[DORA, PROJ],
            v -> union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]}]

Out[58]= True == equal[image[DORA, PROJ],
            union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]]

In[59]:= image[DORA, PROJ] :=
          union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]
```

## a corollary

The same formula holds for the class of unary operations. This corollary requires only a few steps:

```
In[61]:= SubstTest[subclass, w, intersection[x, y],
          {w -> image[DORA, UNOPS], x → range[DORA], y -> inverse[S]}]

Out[61]= subclass[image[DORA, UNOPS],
            union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]] == True

In[62]:= % /. Equal → SetDelayed
```

```
In[64]:= SubstTest[implies, subclass[x, y],
          subclass[image[w, x], image[w, y]], {w → DORA, x → set[0], y → UNOPS}]

Out[64]= member[pair[0, 0], image[DORA, UNOPS]] == True

In[65]:= % /. Equal → SetDelayed
```

Since **PROJ** is a lower bound for **UNOPS**, one finds:

```
In[66]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → PROJ, v → UNOPS, w → DORA}]

Out[66]= subclass[composite[id[complement[set[0]]], inverse[S]], image[DORA, UNOPS]] == True

In[67]:= % /. Equal → SetDelayed
```

Putting all this together, one finds:

```
In[68]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → image[DORA, UNOPS],
           v -> union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]}]

Out[68]= True == equal[image[DORA, UNOPS],
           union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]]

In[70]:= image[DORA, UNOPS] :=
          union[cart[set[0], set[0]], composite[id[complement[set[0]]], inverse[S]]]
```