canonical projections for thin equivalences

Johan G. F. Belinfante
2006 October 16

```
In[1]:= SetDirectory["l:"]; << goedel86.14a; << tools.m

        :Package Title: goedel86.14a          2006 October 14 at 3:35 a.m.

        It is now:  2006 Oct 16 at 16:3

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 October 12

        weightlimit = 40
```

summary

The **canonical projection** for an equivalence relation **eqv[x]** is the function

```
In[2]:= composite[id[complement[set[0]]], VERTSECT[eqv[x]]]

Out[2]= composite[VERTSECT[eqv[x]], id[fix[eqv[x]]]]
```

If **eqv[x]** is thin, then the equivalence class **image[eqv[x], set[w]]** for any member **w** of **fix[eqv[x]]** is a set, and the canonical projection takes **w** to the equivalence class to which it belongs. It is shown in this notebook that one can write the canonical projection of any thin equivalence relation as the composite of the identity function restricted to the class of its equivalence classes with the membership relation **E**. For the special case of sets, a variable-free formulation of this equation can be derived.

a lemma

A general condition for the canonical projection to be a subclass of the membership relation **E** can be obtained using **AssertTest**.

```
In[3]:= subclass[composite[VERTSECT[x], id[domain[x]]], E] // AssertTest

Out[3]= subclass[composite[VERTSECT[x], id[domain[x]]], E] ==
         subclass[domain[thinpart[x]], fix[x]]

In[4]:= subclass[composite[VERTSECT[x_], id[domain[x_]]], E] :=
         subclass[domain[thinpart[x]], fix[x]]
```

For the special case of sets, a variable-free reformulation of this condition can be derived using **Normality**.

```
In[5]:= image[inverse[VS], P[E]] // Normality
Out[5]= image[inverse[VS], P[E]] == invar[composite[DUP, FIRST]]

In[6]:= image[inverse[VS], P[E]] := invar[composite[DUP, FIRST]]
```

For the special case of equivalence relations, one obtains this corollary.

```
In[7]:= SubstTest[subclass, EQV, image[inverse[VS], x], x → P[E]] // Reverse
Out[7]= subclass[U[image[VS, EQV]], E] == True

In[8]:= subclass[U[image[VS, EQV]], E] := True
```

a formula for the canonical projection

Lemma. Te domain of an equivalence relation is its fixed point set. The thinpart of an equivalence relation is an equivalence relation.

```
In[9]:= SubstTest[domain, eqv[y], y -> thinpart[eqv[x]]]
Out[9]= domain[thinpart[eqv[x]]] == fix[thinpart[eqv[x]]]

In[10]:= domain[thinpart[eqv[x_]]] := fix[thinpart[eqv[x]]]
```

Theorem. The result of the preceding section holds for equivalence relations: the canonical projection is a subclass of the membership relation.

```
In[11]:= SubstTest[subclass, composite[VERTSECT[y], id[domain[y]]], E, y → eqv[x]]
Out[11]= subclass[composite[VERTSECT[eqv[x]], id[fix[eqv[x]]]], E] == True

In[12]:= subclass[composite[VERTSECT[eqv[x_]], id[fix[eqv[x_]]]], E] := True
```

Lemma. The range of the canonical projection is a pairwise disjoint class of sets.

```
In[13]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → cartsq[image[VERTSECT[eqv[x]], fix[eqv[x]]]],
           v -> cartsq[range[VERTSECT[eqv[x]]]], w -> union[DISJOINT, Id]}]
Out[13]= subclass[cart[image[VERTSECT[eqv[x]], fix[eqv[x]]],
            image[VERTSECT[eqv[x]], fix[eqv[x]]]], union[DISJOINT, Id]] == True

In[14]:= subclass[cart[image[VERTSECT[eqv[x_]], fix[eqv[x_]]],
            image[VERTSECT[eqv[x_]], fix[eqv[x_]]]], union[DISJOINT, Id]] := True
```

Lemma.

```
In[15]:= equal[composite[id[image[VERTSECT[eqv[x]], fix[eqv[x]]]], E],
           composite[id[image[VERTSECT[eqv[x]], fix[eqv[x]]]],
            inverse[IMAGE[id[fix[eqv[x]]]]], E]] // AssertTest

Out[15]= equal[composite[id[image[VERTSECT[eqv[x]], fix[eqv[x]]]], E],
           composite[id[image[VERTSECT[eqv[x]], fix[eqv[x]]]],
            inverse[IMAGE[id[fix[eqv[x]]]]], E]] == True
```

This result can be made into a temporary rewrite rule:

```
In[16]:= composite[id[image[VERTSECT[eqv[x_]], fix[eqv[x_]]]],
           inverse[IMAGE[id[fix[eqv[x_]]]]], E] :=
          composite[id[image[VERTSECT[eqv[x]], fix[eqv[x]]]], E]
```

The following observation is the key fact used to derive the main theorem:

```
In[17]:= FUNCTION[composite[id[x], E]]

Out[17]= subclass[cart[x, x], union[DISJOINT, Id]]
```

Main theorem. A formula for the canonical projection of a thin equivalence relation.

```
In[18]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u → composite[id[complement[set[0]]], VERTSECT[eqv[y]]],
            v → composite[id[image[VERTSECT[eqv[y]], fix[eqv[y]]]], E]}] /. y → thinpart[x]

Out[18]= equal[composite[id[image[VERTSECT[eqv[thinpart[x]]], fix[eqv[thinpart[x]]]], E],
           composite[VERTSECT[eqv[thinpart[x]]], id[fix[eqv[thinpart[x]]]]]] == True

In[19]:= composite[id[image[VERTSECT[eqv[thinpart[x_]]], fix[eqv[thinpart[x_]]]], E] :=
          composite[VERTSECT[eqv[thinpart[x]]], id[fix[eqv[thinpart[x]]]]]
```

For the inverse of the canonical projection one has this corollary:

```
In[20]:= composite[inverse[E], id[image[VERTSECT[eqv[thinpart[x]]], fix[eqv[thinpart[x]]]]]] //
          DoubleInverse

Out[20]= composite[inverse[E], id[image[VERTSECT[eqv[thinpart[x]]], fix[eqv[thinpart[x]]]]]] ==
          composite[id[fix[eqv[thinpart[x]]]], inverse[VERTSECT[eqv[thinpart[x]]]]]

In[21]:= composite[inverse[E],
           id[image[VERTSECT[eqv[thinpart[x_]]], fix[eqv[thinpart[x_]]]]] :=
          composite[id[fix[eqv[thinpart[x]]]], inverse[VERTSECT[eqv[thinpart[x]]]]]
```

a variable-free corollary

For the special case of equivalence relations that are sets, one can derive variable-free versions of the theorems in the preceding section. Since any set is thin, the **thinpart** wrappers can be replaced with **setpart** wrappers. Doing so facilitates removing the variables.

```
In[22]:= SubstTest[composite,
          id[image[VERTSECT[eqv[thinpart[y]]], fix[eqv[thinpart[y]]]]], E, y → setpart[x]]

Out[22]= composite[id[image[VERTSECT[eqv[setpart[x]]], fix[eqv[setpart[x]]]]], E] ==
          composite[VERTSECT[eqv[setpart[x]]], id[fix[eqv[setpart[x]]]]]

In[23]:= composite[id[image[VERTSECT[eqv[setpart[x_]]], fix[eqv[setpart[x_]]]]], E] :=
          composite[VERTSECT[eqv[setpart[x]]], id[fix[eqv[setpart[x]]]]]
```

The variable **x** is now easily removed, but some cleaning up is in order.

```
In[24]:= SubstTest[class, x,
           equal[image[composite[u, v], set[setpart[x]]], image[v, set[setpart[x]]]],
           {u → composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND]],
            v -> composite[VS, EQUIV]}] // Reverse

Out[24]= image[inverse[EQUIV], image[inverse[VS],
            fix[composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND]]]]] == V

In[25]:= % /. Equal → SetDelayed
```

The only relation with an empty domain is the empty relation. Two relations are equal if their symmetric difference is empty.

```
In[26]:= SubstTest[composite, x, id[domain[x]],
           x -> symdif[composite[IMAGE[composite[id[E], inverse[SECOND]]],
              IMAGE[SECOND], VS, EQUIV], composite[VS, EQUIV]]] // Reverse

Out[26]= union[composite[intersection[complement[VS],
             composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS]], EQUIV],
           composite[id[complement[fix[composite[IMAGE[composite[id[E], inverse[SECOND]]],
                IMAGE[SECOND]]]], VS, EQUIV]] == 0

In[27]:= % /. Equal → SetDelayed
```

A clean variable-free formulation now emerges:

```
In[28]:= SubstTest[empty, symdif[u, v],
          {u -> composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS, EQUIV],
           v -> composite[VS, EQUIV]}]

Out[28]= True == equal[composite[VS, EQUIV],
           composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS, EQUIV]]

In[29]:= composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS, EQUIV] :=
          composite[VS, EQUIV]
```

It is often convenient to replace **EQUIV** with **id[EQV]**, which can be accomplished as follows:

```
In[30]:= Assoc[composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS],
          EQUIV, inverse[EQUIV]]

Out[30]= composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS, id[EQV]] ==
          composite[VS, id[EQV]]
```

```
In[31]:= composite[IMAGE[composite[id[E], inverse[SECOND]]], IMAGE[SECOND], VS, id[EQV]] :=
          composite[VS, id[EQV]]
```