canonical factorization for functions

Johan G. F. Belinfante
2009 December 4

```
In[1]:= SetDirectory["l:"]; << goedel.09dec03a; << tools.m

        :Package Title: goedel.09dec03a            2009 December 3 at 10:55 p.m.

        It is now:  2009 Dec 4 at 12:14

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 November 2

        weightlimit = 40
```

summary

Every function **f = funpart[x]** whose inverse is thin can be factored as the product of a bijection and the canonical projection of the associated equivalence relation **q = inverse[f] ∘ f**. The function **f** is constant on each of the equivalence classes of **q**. Each equivalence of class of the equivalence relation **q** corresponds to a unique value of the function **f**. The one-to-one correspondence **g = f ∘ inverse[p]** assigns to each equivalence class of **q** the value that **f** has for all members of that equivalence class.

When the function **f** is a set, the thin-ness condition holds, and the canonical projection associated with the equivalence relation **q** can be written as **p = APPLY[VS, q] = APPLY[VS, inverse[f]] ∘ f**. The function **f** has the canonical factorization **f = g ∘ p** where the function **g** is one-to-one. The bijection **g** in this case is given by the formula **g = id[range[f]] ∘ APPLY[VS, inverse[f]]**.

The derivation makes use of compound wrappers. Although this is convenient, it does tend to make expressions hard to read. The following translations may help make it easier for the reader to follow the details of the derivation. The function **f** is written as **funpart[setpart[x]]**. The equivalence relation **q** is **composite[inverse[funpart[setpart[x]]], funpart[setpart[x]]]**. The canonical projection **p** associated with **q** is **composite[-VERTSECT[inverse[funpart[setpart[x]]]], funpart[setpart[x]]]**. The one-to-one correspondence **g** is given by the expression **composite[id[range[funpart[setpart[x]]]], inverse[VERTSECT[inverse[funpart[setpart[x]]]]]]**.

No specific rewrite rule for the factorization theorem is required. The **GOEDEL** program simply recognizes the truth of its statement once a more general rewrite rule has been put into place. This more general rule can be viewed as a new simplification rule for the equivalence relation associated with the function **VERTSECT[inverse[funpart[x]]]**. A variable-free corollary of the canonical factorization is also derived which amounts to the somewhat weaker statement that any (small) function can be written as the composite of a bijection and a canonical projection for some equivalence relation.

This notebook was inspired by exercise 4.2 on page 32 in the following reference:

```
In[2]:= "Karel Hrbacek and Thomas Jech, Introduction to Set Theory, Third
         Edition, Revised and Expanded, Marcel Dekker, Inc., New York, 1999.";
```

Another standard treatment of this subject occurs on pages 22-23 in the following book:

```
In[3]:= "Saunders MacLane and Garrett Birkhoff,
          Algebra, The Macmillan Company, New York, 1967.";
```

---

factorization theorem

Lemma. Simplification rule.

```
In[5]:= SubstTest[intersection, complement[domain[t]],
          domain[VERTSECT[t]], t → inverse[x]] // Reverse
Out[5]= intersection[complement[range[x]], domain[VERTSECT[inverse[x]]]] == complement[range[x]]

In[6]:= intersection[complement[range[x_]], domain[VERTSECT[inverse[x_]]]] :=
         complement[range[x]]
```

Theorem. A simplification rule for the equivalence relation associated with the vertical section function corresponding to an inverse function.

```
In[7]:= composite[inverse[VERTSECT[inverse[funpart[x]]]],
          VERTSECT[inverse[funpart[x]]]] // RelnNormality
Out[7]= composite[inverse[VERTSECT[inverse[funpart[x]]]], VERTSECT[inverse[funpart[x]]]] ==
         union[cart[complement[range[funpart[x]]], complement[range[funpart[x]]]],
          id[domain[VERTSECT[inverse[funpart[x]]]]]]

In[8]:= composite[inverse[VERTSECT[inverse[funpart[x_]]]], VERTSECT[inverse[funpart[x_]]]] :=
         union[cart[complement[range[funpart[x]]], complement[range[funpart[x]]]],
          id[domain[VERTSECT[inverse[funpart[x]]]]]]
```

When no further assumptions are made, the rewrite rules of the **GOEDEL** program yields the following generalization of the canonical factorization theorem:

```
In[14]:= composite[id[range[funpart[x]]], inverse[VERTSECT[inverse[funpart[x]]]],
          VERTSECT[inverse[funpart[x]]], funpart[x]]
Out[14]= composite[id[domain[VERTSECT[inverse[funpart[x]]]]], funpart[x]]
```

This simplifies when a **setpart** wrapper is added.

```
In[15]:= % /. x → setpart[x]
Out[15]= funpart[setpart[x]]
```

a variable-free corollary

Lemma.

```
In[16]:= member[composite[VERTSECT[inverse[funpart[setpart[x]]]], funpart[setpart[x]]], V] //
          AssertTest
```

```
Out[16]= member[composite[VERTSECT[inverse[funpart[setpart[x]]]], funpart[setpart[x]]], V] ==
          True
```

```
In[17]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[18]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → VS, u → set[composite[inverse[funpart[setpart[x]]], funpart[setpart[x]]]],
            v → EQV}] // Reverse
```

```
Out[18]= member[composite[VERTSECT[inverse[funpart[setpart[x]]]], funpart[setpart[x]]],
           image[VS, EQV]] == True
```

```
In[19]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[20]:= SubstTest[implies, and[member[u, y], member[v, z]],
           member[composite[u, v], image[COMPOSE, cart[y, z]]],
           {u -> composite[id[range[funpart[setpart[x]]]],
              inverse[VERTSECT[inverse[funpart[setpart[x]]]]]],
            v -> composite[VERTSECT[inverse[funpart[setpart[x]]]], funpart[setpart[x]]],
            y → BIJ, z → image[VS, EQV]}] // Reverse
```

```
Out[20]= member[funpart[setpart[x]], image[COMPOSE, cart[BIJ, image[VS, EQV]]]] == True
```

```
In[21]:= (% /. x → x_) /. Equal→ SetDelayed
```

Eliminating the variable **x** yields the following inclusion.

Lemma.

```
In[22]:= Map[subclass[FUNS, image[FUNPART, #]] &, SubstTest[class, x, member[setpart[x], t],
           t → image[inverse[FUNPART], image[COMPOSE, cart[BIJ, image[VS, EQV]]]]]]
```

```
Out[22]= subclass[FUNS, image[COMPOSE, cart[BIJ, image[VS, EQV]]]] == True
```

```
In[23]:= % /. Equal→ SetDelayed
```

Lemma. Inclusion in the reverse direction.

```
In[24]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → COMPOSE, u→ cart[BIJ, image[VS, EQV]], v → cart[FUNS, FUNS]}] // Reverse

Out[24]= subclass[image[COMPOSE, cart[BIJ, image[VS, EQV]]], FUNS] == True

In[25]:= % /. Equal→ SetDelayed
```

Theorem. Every function can be factored into a bijection and a canonical projection of some equivalence relation.

```
In[26]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[COMPOSE, cart[BIJ, image[VS, EQV]]], v -> FUNS}]

Out[26]= equal[FUNS, image[COMPOSE, cart[BIJ, image[VS, EQV]]]] == True

In[27]:= image[COMPOSE, cart[BIJ, image[VS, EQV]]] := FUNS
```