COMPIMAG

Johan G. F. Belinfante
2009 September 14

```
In[1]:= SetDirectory["l:"]; << goedel.09sep12a; << tools.m

        :Package Title: goedel.09sep12a              2009 September 12 at 5:10 p.m.

        It is now:  2009 Sep 22 at 7:39

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 September 3

        weightlimit = 40
```

summary

Because **IMAGE[x]** is a proper class when **x** is a set, the function **λ x. IMAGE[x]** is empty. In this notebook a variable-free formulation of the fact that the constructor **IMAGE** preserves composition for sets is obtained by introducing a new binary function **COMPIMAG** which takes an ordered pair of sets **x** and **y** to the set **IMAGE[x] ∘ y**. This new function is closely related to the function **IPD** which was introduced recently in order to construct the covariant power set functor for the category of sets.

definition

The binary function **COMPIMAG** is defined by the following membership rule.

```
In[2]:= member[x_, COMPIMAG] :=
         and[equal[composite[IMAGE[first[first[x]]], second[first[x]]], second[x]],
          member[first[first[x]], V]]
```

Lemma.

```
In[3]:= Map[empty, dif[COMPIMAG, cart[cart[V, V], V]] // Normality]

Out[3]= subclass[COMPIMAG, cart[cart[V, V], V]] == True
```

```
In[4]:= subclass[COMPIMAG, cart[cart[V, V], V]] := True
```

Theorem. Simplification rule.

```
In[5]:= equal[composite[COMPIMAG, id[cart[V, V]]], COMPIMAG]

Out[5]= True
```

```
In[6]:= composite[COMPIMAG, id[cart[V, V]]] := COMPIMAG
```

A vertical section rule follows immediately from the definition.

Theorem.

```
In[7]:= image[COMPIMAG, set[PAIR[setpart[x], setpart[y]]]] // Normality

Out[7]= image[COMPIMAG, cart[set[setpart[x]], set[setpart[y]]]] ==
         set[composite[IMAGE[setpart[x]], setpart[y]]]

In[8]:= image[COMPIMAG, cart[set[setpart[x_]], set[setpart[y_]]]] :=
         set[composite[IMAGE[setpart[x]], setpart[y]]]
```

This vertical section rule implies that **COMPIMAG** is a function.

Theorem.

```
In[9]:= Map[FUNCTION[composite[#, id[cart[V, V]]]] &, SubstTest[reify, x,
          image[t, set[PAIR[setpart[first[x]], setpart[second[x]]]]], t → COMPIMAG]]

Out[9]= FUNCTION[COMPIMAG] == True

In[10]:= FUNCTION[COMPIMAG] := True
```

an explicit formula for COMPIMAG

An explicit formula for **COMPIMAG** can be derived by reifying on both variables in the vertical section rule. The first step yields the following useful fact.

Theorem. An explicit formula for left-composition.

```
In[11]:= SubstTest[reify, y, image[t, set[PAIR[setpart[x], setpart[y]]]], t → COMPIMAG]

Out[11]= composite[COMPIMAG, LEFT[setpart[x]]] == IMAGE[cross[Id, IMAGE[setpart[x]]]]

In[12]:= composite[COMPIMAG, LEFT[setpart[x_]]] := IMAGE[cross[Id, IMAGE[setpart[x]]]]
```

Theorem. An explicit formula for the binary function **COMPIMAG**.

```
In[13]:= Map[rotate[inverse[#]] &,
           SubstTest[reify, x, composite[t, LEFT[setpart[x]]], t → COMPIMAG]] // Reverse

Out[13]= composite[IMAGE[composite[SWAP, RIF,
             cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]], Id]]],
           CART, cross[SINGLETON, Id]] == COMPIMAG

In[14]:= composite[IMAGE[composite[SWAP, RIF,
             cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]], Id]]],
           CART, cross[SINGLETON, Id]] := COMPIMAG
```

domain

It is shown in this section that **COMPIMAG** is a total binary function. This amounts to the following fact.

Theorem. A sethood result.

```
In[15]:= member[composite[IMAGE[setpart[x]], setpart[y]], V] // AssertTest

Out[15]= member[composite[IMAGE[setpart[x]], setpart[y]], V] == True
```

```
In[16]:= member[composite[IMAGE[setpart[x_]], setpart[y_]], V] := True
```

Lemma. A simplification rule.

```
In[17]:= domain[VERTSECT[composite[FIRST,
             intersection[composite[inverse[rotate[composite[IMG, SWAP]]], FIRST],
              composite[inverse[SECOND], SECOND, SECOND]]]] // Normality

Out[17]= domain[VERTSECT[composite[FIRST,
             intersection[composite[inverse[rotate[composite[IMG, SWAP]]], FIRST],
              composite[inverse[SECOND], SECOND, SECOND]]]] == V
```

```
In[18]:= domain[VERTSECT[composite[FIRST,
             intersection[composite[inverse[rotate[composite[IMG, SWAP]]], FIRST],
              composite[inverse[SECOND], SECOND, SECOND]]]] := V
```

Theorem. The domain of **COMPIMAG** is the class of all ordered pairs of sets.

```
In[19]:= IminComp[composite[IMAGE[composite[SWAP, RIF,
             cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]], Id]]],
           CART], cross[SINGLETON, Id], V]

Out[19]= domain[COMPIMAG] == cart[V, V]
```

```
In[20]:= domain[COMPIMAG] := cart[V, V]
```

relation to the function IPD

The function **IPD** which takes any set **x** to the **composite[IMAGE[x], id[P[domain[x]]]** was introduced recently in order to construct the covariant power functor for the category of sets.

Lemma. A vertical section formula.

```
In[21]:= SubstTest[image, COMPIMAG, cart[set[setpart[x]], set[setpart[y]]],
           y → id[P[domain[setpart[x]]]]] // Reverse

Out[21]= image[COMPIMAG, cart[set[setpart[x]], set[id[P[domain[setpart[x]]]]]]] ==
          set[composite[IMAGE[setpart[x]], id[P[domain[setpart[x]]]]]]
```

```
In[22]:= image[COMPIMAG, cart[set[setpart[x_]], set[id[P[domain[setpart[x_]]]]]]] :=
          set[composite[IMAGE[setpart[x]], id[P[domain[setpart[x]]]]]]
```

Theorem. An explicit equation relating **COMPIMAG** to the function **IPD**.

```
In[23]:= SubstTest[reify, x,
          image[composite[t, id[composite[IDP, POWER, IMAGE[FIRST]]], inverse[FIRST]],
           set[setpart[x]]], t → COMPIMAG]

Out[23]= composite[COMPIMAG,
           id[composite[IMAGE[DUP], POWER, IMAGE[FIRST]]], inverse[FIRST]] == IPD

In[24]:= composite[COMPIMAG,
           id[composite[IMAGE[DUP], POWER, IMAGE[FIRST]]], inverse[FIRST]] := IPD
```

a normalization rule

A normalization rule us derived in this section which provides another explicit formula to **COMPIMAG**.

Lemma.

```
In[25]:= symdif[COMPIMAG, composite[VERTSECT[
             composite[SWAP, RIF, cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
               composite[inverse[E], IMAGE[id[cart[V, V]]]]]], id[cart[V, V]]]] // TriNormality

Out[25]= union[composite[intersection[COMPIMAG, complement[VERTSECT[composite[SWAP,
                RIF, cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
                 composite[inverse[E], IMAGE[id[cart[V, V]]]]]]]], id[cart[V, V]]],
           composite[intersection[complement[COMPIMAG], VERTSECT[composite[SWAP,
               RIF, cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
                composite[inverse[E], IMAGE[id[cart[V, V]]]]]]]], id[cart[V, V]]]] == 0

In[26]:= % /. Equal → SetDelayed
```

Theorem. Normalization rule.

```
In[27]:= SubstTest[empty, composite[symdif[u, v], id[cart[V, V]]],
          {u → COMPIMAG, v→ composite[VERTSECT[composite[SWAP, RIF,
               cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
                composite[inverse[E], IMAGE[id[cart[V, V]]]]]], id[cart[V, V]]]}]

Out[27]= equal[COMPIMAG, composite[VERTSECT[
             composite[SWAP, RIF, cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
               composite[inverse[E], IMAGE[id[cart[V, V]]]]]], id[cart[V, V]]]] == True

In[28]:= composite[VERTSECT[
             composite[SWAP, RIF, cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
               composite[inverse[E], IMAGE[id[cart[V, V]]]]]], id[cart[V, V]]] := COMPIMAG
```

vertical sections

A more general vertical section rule is derived in this section that does not require **setpart** wrappers. This result is based on the normalization rule derived in the preceding section.

Theorem. A general vertical section rule for **COMPIMAG**.

```
In[29]:= ImageComp[VERTSECT[
           composite[SWAP, RIF, cross[composite[SWAP, inverse[rotate[composite[IMG, SWAP]]]],
             composite[inverse[E], IMAGE[id[cart[V, V]]]]]], id[cart[V, V]], set[PAIR[x, y]]]

Out[29]= image[COMPIMAG, cart[set[x], set[y]]] == intersection[image[V, set[x]], image[V, set[y]],
           set[composite[IMAGE[x], id[image[V, set[x]]], setpart[composite[Id, y]]]]]

In[30]:= image[COMPIMAG, cart[set[x_], set[y_]]] :=
          intersection[image[V, set[x]], image[V, set[y]],
           set[composite[IMAGE[x], id[image[V, set[x]]], setpart[composite[Id, y]]]]]
```

quasi-associative law

In this section a variable-free formulation is derived for the fact that **IMAGE** preserves composites for sets.

```
In[31]:= IMAGE[composite[setpart[x], setpart[y]]]

Out[31]= composite[IMAGE[setpart[x]], IMAGE[setpart[y]]]
```

Comment. More generally, the constructor **IMAGE** preserves composites for proper classes, provided that the right-hand factor is thin. In particular, **IMAGE** preserves composites for arbitrary functions whether they are sets or not. The variable-free formula derived in this section only applies to the special case of sets. The idea is to introduce a third set variable, and to derive a quasi-associative law corresponding to the formula **IMAGE[x ∘ y] ∘ z = IMAGE[x] ∘ (IMAGE[y] ∘ z)**. The derivation uses reification and the fact that two classes are equal if and only if their symmetric difference is empty.

Lemma. The vertical sections of a symmetric difference are empty.

```
In[32]:= Map[composite[#, id[cart[cart[V, V], V]]] &, SubstTest[reify, x,
           image[t, set[PAIR[PAIR[setpart[first[first[x]]], setpart[second[first[x]]]],
              setpart[second[x]]]], t -> symdif[composite[COMPIMAG, cross[COMPOSE, Id]],
             composite[COMPIMAG, cross[Id, COMPIMAG], ASSOC]]]]

Out[32]= union[composite[intersection[composite[COMPIMAG, cross[COMPOSE, Id]], composite[
              complement[COMPIMAG], cross[Id, COMPIMAG], ASSOC]], id[cart[cart[V, V], V]]],
           composite[intersection[composite[complement[COMPIMAG], cross[COMPOSE, Id]],
             composite[COMPIMAG, cross[Id, COMPIMAG], ASSOC]], id[cart[cart[V, V], V]]]] == 0

In[33]:= % /. Equal → SetDelayed
```

Theorem. Quasi-associative law.

```
In[34]:= SubstTest[empty, composite[symdif[u, v], id[w]],
          {u -> composite[COMPIMAG, cross[COMPOSE, Id]],
           v -> composite[COMPIMAG, cross[Id, COMPIMAG], ASSOC], w → cart[cart[V, V], V]}]

Out[34]= equal[composite[COMPIMAG, cross[COMPOSE, Id]],
           composite[COMPIMAG, cross[Id, COMPIMAG], ASSOC]] == True

In[35]:= composite[COMPIMAG, cross[Id, COMPIMAG], ASSOC] :=
          composite[COMPIMAG, cross[COMPOSE, Id]]
```

---

composite with IMAGE[FIRST] and IMAGE[SECOND]

Since **IMAGE[x]** is a total function when **x** is a set, the domain of **IMAGE[x]** ∘ **y** is equal to the domain of **y**.

Theorem.

```
In[36]:= composite[IMAGE[FIRST], COMPIMAG] // ReifTriNormality

Out[36]= composite[IMAGE[FIRST], COMPIMAG] == composite[IMAGE[FIRST], SECOND]

In[37]:= composite[IMAGE[FIRST], COMPIMAG] := composite[IMAGE[FIRST], SECOND]
```

A somewhat more complicated result will be derived for the composite with **IMAGE[SECOND]**.

Lemma.

```
In[38]:= SubstTest[reify, x, image[t, set[PAIR[setpart[first[x]], setpart[second[x]]]]],
          t -> symdif[composite[IMAGE[SECOND], COMPIMAG],
            composite[IMAGE[IMG], CART, cross[SINGLETON, IMAGE[SECOND]]]]]

Out[38]= union[composite[intersection[composite[complement[IMAGE[SECOND]], COMPIMAG],
             composite[IMAGE[IMG], CART, cross[SINGLETON, IMAGE[SECOND]]]], id[cart[V, V]]],
           composite[intersection[composite[IMAGE[SECOND], COMPIMAG], composite[complement[
              IMAGE[IMG]], CART, cross[SINGLETON, IMAGE[SECOND]]]], id[cart[V, V]]]] == 0

In[39]:= % /. Equal→ SetDelayed
```

Theorem. An explicit formula for **IMAGE[SECOND]** ∘ **COMPIMAG**.

```
In[40]:= SubstTest[empty, composite[symdif[u, v], id[w]],
          {u -> composite[IMAGE[SECOND], COMPIMAG],
           v -> composite[IMAGE[IMG], CART, cross[SINGLETON, IMAGE[SECOND]]], w -> cart[V, V]}]

Out[40]= equal[composite[IMAGE[SECOND], COMPIMAG],
           composite[IMAGE[IMG], CART, cross[SINGLETON, IMAGE[SECOND]]]] == True

In[41]:= composite[IMAGE[SECOND], COMPIMAG] :=
          composite[IMAGE[IMG], CART, cross[SINGLETON, IMAGE[SECOND]]]
```