category endomorphisms

Johan G. F. Belinfante
2009 August 6

```
In[1]:= SetDirectory["l:"]; << goedel.09jul31a; << tools.m

        :Package Title: goedel.09jul31a              2009 July 31 at 5:55 p.m.

        It is now:  2009 Aug 6 at 9:58

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 July 2

        weightlimit = 40
```

summary

If **u · v** is an endomorphism in a category, then **v · u** is also an endomorphism.

endomorphisms

The class of **endomorphisms** in a category **cat[x]** is **image[hom[cat[x]], Id] = fix[domain[cat[x]]]**. For example, identity morphisms are endomorphisms: **ids[cat[x]] ⊂ fix[domain[cat[x]]]**. It will be shown below that one can equivalently characterize an endomorphism as a morphism that has equal **cod** and **dom**. The following variable-free statement of this is already available:

```
In[2]:= fix[composite[inverse[cod[cat[x]]], dom[cat[x]]]]

Out[2]= fix[domain[cat[x]]]
```

Theorem. The class of endomorphisms is contained in the class of all morphisms:

```
In[3]:= SubstTest[subclass, fix[t], domain[t], t → domain[cat[x]]] // Reverse

Out[3]= subclass[fix[domain[cat[x]]], range[cat[x]]] == True

In[4]:= subclass[fix[domain[cat[x_]]], range[cat[x_]]] := True
```

Corollary. A simplification rule needed later.

```
In[5]:= equal[intersection[range[cat[x]], fix[domain[cat[x]]]], fix[domain[cat[x]]]]

Out[5]= True

In[6]:= intersection[fix[domain[cat[x_]]], range[cat[x_]]] := fix[domain[cat[x]]]
```

Theorem. An endomorphism has equal **cod** and **dom**.

```
In[7]:= Map[implies[#, equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], u]]] &,
         (member[u, fix[composite[inverse[funpart[y]], funpart[z]]]] // AssertTest) /.
          {y → dom[cat[x]], z → cod[cat[x]]}]

Out[7]= or[equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], u]],
          not[member[u, fix[domain[cat[x]]]]]] == True

In[8]:= or[equal[APPLY[cod[cat[x_]], u_], APPLY[dom[cat[x_]], u_]],
          not[member[u_, fix[domain[cat[x_]]]]]] := True
```

Converse Theorem. A morphism with equal **cod** and **dom** is an endomorphism.

```
In[9]:= Map[implies[#, member[u, fix[domain[cat[x]]]]] &,
         (member[u, fix[composite[inverse[funpart[y]], funpart[z]]]] // AssertTest // Reverse) /.
          {y → dom[cat[x]], z → cod[cat[x]]}]

Out[9]= or[member[u, fix[domain[cat[x]]]],
          not[equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], u]]],
          not[member[u, range[cat[x]]]]] == True

In[10]:= or[member[u_, fix[domain[cat[x_]]]],
           not[equal[APPLY[cod[cat[x_]], u_], APPLY[dom[cat[x_]], u_]]],
           not[member[u_, range[cat[x_]]]]] := True
```

---

products of morphisms

If **u · v** is the product of any two morphisms in a category **x**, then **pair[u, v] ∈ domain[x]**. If **pair[u, v] ∈ domain[x]** in a category **x**, then the **cod** of **v** is equal to the **dom** of **u**. Putting these two facts together yields the following general statement.

Theorem. If **u · v** is any kind of morphism in a category **x**, then the **cod** of **v** is equal to the **dom** of **u**.

```
In[11]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 -> member[APPLY[cat[x], PAIR[u, v]], y], p2 -> member[pair[u, v], domain[cat[x]]],
             p3 -> equal[APPLY[dom[cat[x]], u], APPLY[cod[cat[x]], v]]}]] // Reverse

Out[11]= or[equal[APPLY[cod[cat[x]], v], APPLY[dom[cat[x]], u]],
           not[member[APPLY[cat[x], PAIR[u, v]], y]]] == True

In[12]:= or[equal[APPLY[cod[cat[x_]], v_], APPLY[dom[cat[x_]], u_]],
           not[member[APPLY[cat[x_], PAIR[u_, v_]], y_]]] := True
```

When the fact that endomorphisms have equal **cod** and **dom** is instantiated to the case of a product of two morphisms, rewrite rules about the **cod** and **dom** of a composite transform the statement as follows:

Lemma.

```
In[13]:= SubstTest[implies, member[t, fix[domain[cat[x]]]],
           equal[APPLY[cod[cat[x]], t], APPLY[dom[cat[x]], t]],
           t → APPLY[cat[x], PAIR[u, v]]] // Reverse

Out[13]= or[equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], v]],
           not[member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]]],
           not[member[pair[u, v], domain[cat[x]]]] == True

In[14]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

This result can be simplified. The composability literal is redundant.

Theorem. If **u · v** is an endomorphism, then the **cod** of **u** is equal to the **dom** of **v**.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
            {p1 -> member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]],
             p2 -> member[pair[u, v], domain[cat[x]]],
             p3 -> equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], v]]}]] // Reverse

Out[15]= or[equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], v]],
           not[member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]]]] == True

In[16]:= or[equal[APPLY[cod[cat[x_]], u_], APPLY[dom[cat[x_]], v_]],
           not[member[APPLY[cat[x_], PAIR[u_, v_]], fix[domain[cat[x_]]]]]] := True
```

If the product of two morphisms is an endomorphism, then they can also be composed on the opposite order.

Corollary. If **u · v ∈ fix[domain[cat[x]]]**, then **v · u** is a morphism.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 -> member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]],
             p2 → equal[APPLY[dom[cat[x]], v], APPLY[cod[cat[x]], u]],
             p3 → member[u, range[cat[x]]],
             p4 → member[pair[v, u], domain[cat[x]]]}]] // Reverse

Out[17]= or[member[pair[v, u], domain[cat[x]]],
           not[member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]]]] == True

In[18]:= or[member[pair[v_, u_], domain[cat[x_]]],
           not[member[APPLY[cat[x_], PAIR[u_, v_]], fix[domain[cat[x_]]]]]] := True
```

Corollary. (Obtained by eliminating the variables **u** and **v**.)

```
In[19]:= Map[empty[domain[complement[#]]] &,
          complement[dif[image[inverse[cat[x]], fix[domain[cat[x]]]],
            inverse[domain[cat[x]]]]] // RelnNormality]

Out[19]= subclass[inverse[image[inverse[cat[x]], fix[domain[cat[x]]]]], domain[cat[x]]] == True

In[20]:= subclass[inverse[image[inverse[cat[x_]], fix[domain[cat[x_]]]]],
           domain[cat[x_]]] := True
```

Corollary.

```
In[21]:= SubstTest[subclass, composite[Id, u], inverse[v],
           {u -> image[inverse[cat[x]], fix[domain[cat[x]]]], v -> domain[cat[x]]}] // Reverse

Out[21]= subclass[image[inverse[cat[x]], fix[domain[cat[x]]]], inverse[domain[cat[x]]]] == True

In[22]:= subclass[image[inverse[cat[x_]], fix[domain[cat[x_]]]],
           inverse[domain[cat[x_]]]] := True
```

A counterexample will now be presented to show that this inclusion cannot be improved upon.

Lemma.

```
In[27]:= Map[not, SubstTest[implies, equal[u, v], equal[image[u, w], image[v, w]],
            {u -> SWAP, v -> composite[inverse[FIRST], SECOND], w → id[set[0]]}]] // Reverse

Out[27]= equal[SWAP, composite[inverse[FIRST], SECOND]] == False

In[28]:= equal[SWAP, composite[inverse[FIRST], SECOND]] := False
```

Counterexample. In general the inclusion derived above cannot be strengthened to an equation.

```
In[29]:= equal[image[inverse[cat[x]], fix[domain[cat[x]]]], inverse[domain[cat[x]]]] /.
          x → composite[SWAP, RIF]

Out[29]= False
```

The following is an instantiation to products for the converse statement that a morphism with equal **cod** and **dom** is an endomorphism.

Theorem.

```
In[30]:= SubstTest[or, member[t, fix[domain[cat[x]]]],
           not[equal[APPLY[cod[cat[x]], t], APPLY[dom[cat[x]], t]]],
           not[member[t, range[cat[x]]]], t → APPLY[cat[x], PAIR[u, v]]] // Reverse

Out[30]= or[member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]],
           not[equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], v]]],
           not[member[pair[u, v], domain[cat[x]]]]] == True

In[31]:= or[member[APPLY[cat[x_], PAIR[u_, v_]], fix[domain[cat[x_]]]],
           not[equal[APPLY[cod[cat[x_]], u_], APPLY[dom[cat[x_]], v_]]],
           not[member[pair[u_, v_], domain[cat[x_]]]]] := True
```

The morphism variables **u** and **v** could be eliminated from this statement, but the same result can be obtained more simply is as follows.

Lemma. An inclusion.

```
In[32]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → cat[x], u → image[inverse[cat[x]], fix[domain[cat[x]]]],
            v → inverse[domain[cat[x]]]}] // Reverse

Out[32]= subclass[fix[domain[cat[x]]], image[cat[x], inverse[domain[cat[x]]]]] == True

In[33]:= (% /. x → x_) /. Equal → SetDelayed
```

It will be shown now that the reverse inclusion also holds.

Lemma.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[and[p1, p2, p3], p4], implies[and[p1, p4], p5], not[implies[p1, p5]],
            {p1 → and[member[pair[u, v], domain[cat[x]]], member[pair[v, u], domain[cat[x]]]],
             p2 → or[member[pair[u, v], domain[cat[x]]], not[member[v, range[cat[x]]]]],
             p3 → or[member[pair[u, v], domain[cat[x]]], not[member[u, range[cat[x]]]]],
             p4 → or[equal[APPLY[cod[cat[x]], u], APPLY[dom[cat[x]], v]],
               not[member[pair[u, v], domain[cat[x]]]]],
             p5 → member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]]}]] // Reverse

Out[34]= or[member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]],
           not[member[pair[u, v], domain[cat[x]]]],
           not[member[pair[v, u], domain[cat[x]]]]] == True

In[35]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Eliminating the morphism variables **u** and **v** yields an inclusion:

Lemma.

```
In[36]:= Map[empty[domain[complement[#]]] &,
          SubstTest[class, pair[u, v], implies[member[pair[u, v], y], member[pair[u, v], z]],
           {y → intersection[domain[cat[x]], inverse[domain[cat[x]]]],
            z → image[inverse[cat[x]], fix[domain[cat[x]]]]}]]

Out[36]= subclass[intersection[domain[cat[x]], inverse[domain[cat[x]]]],
           inverse[image[inverse[cat[x]], fix[domain[cat[x]]]]]] == True

In[37]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[38]:= SubstTest[subclass, composite[Id, u], inverse[v],
           {u -> intersection[domain[cat[x]], inverse[domain[cat[x]]]],
            v -> image[inverse[cat[x]], fix[domain[cat[x]]]]}]

Out[38]= subclass[intersection[domain[cat[x]], inverse[domain[cat[x]]]],
           image[inverse[cat[x]], fix[domain[cat[x]]]]] == True

In[39]:= (% /. x → x_) /. Equal → SetDelayed
```

An equation is obtained by combining this inclusion with the one in the opposite direction.

Theorem. An equation that can be made into a rewrite rule.

```
In[40]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[domain[cat[x]], inverse[domain[cat[x]]]],
           v -> image[inverse[cat[x]], fix[domain[cat[x]]]]}]

Out[40]= equal[image[inverse[cat[x]], fix[domain[cat[x]]]],
           intersection[domain[cat[x]], inverse[domain[cat[x]]]]] == True

In[41]:= image[inverse[cat[x_]], fix[domain[cat[x_]]]] :=
          intersection[domain[cat[x]], inverse[domain[cat[x]]]]
```

Corollary. A simpler equation.

```
In[42]:= ImageComp[cat[x], inverse[cat[x]], fix[domain[cat[x]]]] // Reverse

Out[42]= image[cat[x], inverse[domain[cat[x]]]] == fix[domain[cat[x]]]

In[43]:= image[cat[x_], inverse[domain[cat[x_]]]] := fix[domain[cat[x]]]
```

Corollary. If **u · v** is an endomorphism in a category, then **v · u** is also an endomorphism.

```
In[44]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[p1, p4]], {p1 -> member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]],
             p2 -> member[pair[v, u], domain[cat[x]]],
             p3 -> equal[APPLY[cod[cat[x]], v], APPLY[dom[cat[x]], u]],
             p4 -> member[APPLY[cat[x], PAIR[v, u]], fix[domain[cat[x]]]]}]] // Reverse

Out[44]= or[member[APPLY[cat[x], PAIR[v, u]], fix[domain[cat[x]]]],
           not[member[APPLY[cat[x], PAIR[u, v]], fix[domain[cat[x]]]]]] == True

In[45]:= or[member[APPLY[cat[x_], PAIR[v_, u_]], fix[domain[cat[x_]]]],
           not[member[APPLY[cat[x_], PAIR[u_, v_]], fix[domain[cat[x_]]]]]] := True
```