hom[cat[x]]

Johan G. F. Belinfante
2009 January 23

```
In[1]:= SetDirectory["l:"]; << goedel.09jan23a;<< tools.m

        :Package Title: goedel.09jan23a        2009 January 23 at 12:40 noon

        It is now:  2009 Jan 23 at 12:53

        Loading Simplification Rules

        TOOLS.M                                        Revised 2009 January 20

        weightlimit = 40
```

summary

Beyond the general properties satisfied by **hom[x]** are special ones that hold when **x** is a category. In this notebook the **cat[x]** wrapper is used for convenience to derive rewrite rules for such facts. In particular, it is shown that the function **inverse[hom[cat[x]]]** is a functor from the category **cat[x]** to the category **composite[SWAP, RIF**], known as the pair groupoid. This functor assigns to each morphism **y** of the category **cat[x]** the pair of identity morphisms **APPLY[dom[cat[x]], y]** and **APPLY[cod[cat[x]], y]**. The domain of **hom[cat[x]]** is shown to be a quasi-order (reflexive transitive relation). An explicit counterexample is presented showing that **domain[hom[x]]** need not be transitive when **x** is not a category.

inverse[hom[cat[x]]] as a functor

Lemma.

```
In[2]:= Map[composite[#, cross[DUP, DUP]] &,
          (composite[SWAP, RIF, cross[cross[u, v], cross[u, v]]] // ReifTriNormality) /.
           {u → dom[cat[x]], v → cod[cat[x]]}]

Out[2]= composite[SWAP, RIF, cross[inverse[hom[cat[x]]], inverse[hom[cat[x]]]]] ==
         composite[SWAP, cross[cod[cat[x]], dom[cat[x]]], id[domain[cat[x]]]]

In[3]:= composite[SWAP, RIF, cross[inverse[hom[cat[x_]]], inverse[hom[cat[x_]]]]] :=
         composite[SWAP, cross[cod[cat[x]], dom[cat[x]]], id[domain[cat[x]]]]
```

Lemma.

```
In[4]:= Assoc[cross[dom[cat[x]], cod[cat[x]]], cross[cat[x], cat[x]], DUP]

Out[4]= composite[inverse[hom[cat[x]]], cat[x]] ==
         composite[SWAP, cross[cod[cat[x]], dom[cat[x]]], id[domain[cat[x]]]]

In[5]:= composite[inverse[hom[cat[x_]]], cat[x_]] :=
         composite[SWAP, cross[cod[cat[x]], dom[cat[x]]], id[domain[cat[x]]]]
```

Lemma.

```
In[6]:= ImageComp[cross[dom[cat[x]], cod[cat[x]]], DUP, ids[cat[x]]]

Out[6]= image[inverse[hom[cat[x]]], ids[cat[x]]] == id[ids[cat[x]]]

In[7]:= image[inverse[hom[cat[x_]]], ids[cat[x_]]] := id[ids[cat[x]]]
```

Theorem. The function **inverse[hom[cat[x]]]** is a functor from any category **cat[x]** to the pair groupoid **composite[SWAP, RIF]**.

```
In[8]:= SubstTest[or, functor[w, u, v], not[equal[composite[w, u], composite[v, cross[w, w]]]],
          not[equal[domain[w], range[u]]], not[FUNCTION[w]],
          not[subclass[image[w, ids[u]], ids[v]]],
          {w → inverse[hom[cat[x]]], u → cat[x], v → composite[SWAP, RIF]}] // Reverse

Out[8]= functor[inverse[hom[cat[x]]], cat[x], composite[SWAP, RIF]] == True

In[9]:= functor[inverse[hom[cat[x_]]], cat[x_], composite[SWAP, RIF]] := True
```

---

examples

In this section, some examples are provided to illustrate the theorem derived in the preceding section.

Theorem. The function **DUP** is a functor from the discrete category **inverse[DUP]** to the pair groupoid **composite[SWAP,-RIF]**.

```
In[10]:= SubstTest[functor, inverse[hom[cat[x]]],
           cat[x], composite[SWAP, RIF], x → inverse[DUP]] // Reverse

Out[10]= functor[DUP, inverse[DUP], composite[SWAP, RIF]] == True

In[11]:= functor[DUP, inverse[DUP], composite[SWAP, RIF]] := True
```

Lemma. Formula for the ternary **hom** relation for the pair groupoid.

```
In[12]:= SubstTest[intersection, composite[inverse[cod[t]], SECOND],
          composite[inverse[dom[t]], FIRST], t → composite[SWAP, RIF]]

Out[12]= hom[composite[SWAP, RIF]] == cross[inverse[DUP], inverse[DUP]]

In[13]:= hom[composite[SWAP, RIF]] := cross[inverse[DUP], inverse[DUP]]
```

Theorem. The function **cross[DUP, DUP]** is a functor from the pair groupoid **composite[SWAP, RIF]** to itself.

```
In[14]:= SubstTest[functor, inverse[hom[cat[x]]], cat[x],
           composite[SWAP, RIF], x → composite[SWAP, RIF]] // Reverse

Out[14]= functor[cross[DUP, DUP], composite[SWAP, RIF], composite[SWAP, RIF]] == True

In[15]:= functor[cross[DUP, DUP], composite[SWAP, RIF], composite[SWAP, RIF]] := True
```

domain[hom[cat[x]]] is transitive

When **x** is a category, the relation **dom[hom[x]]** consists of all pairs of identities **u**, **v** for which there is a morphism from **u** to **v**. In this section it is shown that this relation is transitive as well as being reflexive.

Lemma.

```
In[16]:= ImageComp[composite[SWAP, RIF], cross[inverse[hom[cat[x]]], inverse[hom[cat[x]]]], V]

Out[16]= composite[cod[cat[x]], inverse[domain[cat[x]]], inverse[dom[cat[x]]]] ==
          composite[domain[hom[cat[x]]], domain[hom[cat[x]]]]

In[17]:= composite[cod[cat[x_]], inverse[domain[cat[x_]]], inverse[dom[cat[x_]]]] :=
          composite[domain[hom[cat[x]]], domain[hom[cat[x]]]]
```

Theorem. The relation **domain[hom[cat[x]]]** is idempotent.

```
In[18]:= ImageComp[inverse[hom[cat[x]]], cat[x], V]

Out[18]= composite[domain[hom[cat[x]]], domain[hom[cat[x]]]] == domain[hom[cat[x]]]

In[19]:= composite[domain[hom[cat[x_]]], domain[hom[cat[x_]]]] := domain[hom[cat[x]]]
```

Corollary.

```
In[20]:= SubstTest[subclass, composite[t, t], t, t → domain[hom[cat[x]]]]

Out[20]= TRANSITIVE[domain[hom[cat[x]]]] == True

In[21]:= TRANSITIVE[domain[hom[cat[x_]]]] := True
```

a counterexample

The relation **domain[hom[x]]** need not be transitive when **x** is not a category. A restriction of the pair groupoid is used in this section to provide a counterexample.

Lemma. A **hom** formula for restrictions of the pair groupoid.

```
In[22]:= SubstTest[intersection, composite[inverse[cod[t]], SECOND],
          composite[inverse[dom[t]], FIRST], t → composite[SWAP, RIF, id[cartsq[x]]]]

Out[22]= hom[composite[SWAP, RIF, id[cart[x, x]]]] == composite[
           cross[inverse[DUP], inverse[DUP]], id[composite[DUP, rfx[x], inverse[DUP]]]]

In[23]:= hom[composite[SWAP, RIF, id[cart[x_, x_]]]] := composite[
           cross[inverse[DUP], inverse[DUP]], id[composite[DUP, rfx[x], inverse[DUP]]]]
```

Lemma.

```
In[25]:= SubstTest[subclass, w, cartsq[fix[w]],
           w -> composite[oopart[t], x, inverse[oopart[t]]]] /. t → DUP

Out[25]= REFLEXIVE[composite[DUP, x, inverse[DUP]]] == REFLEXIVE[composite[Id, x]]

In[26]:= REFLEXIVE[composite[DUP, x_, inverse[DUP]]] := REFLEXIVE[composite[Id, x]]
```

The relation **domain[hom[x]]** is reflexive for any class **x**. For a restriction of **composite[SWAP, RIF]**, one obtains:

```
In[24]:= domain[hom[composite[SWAP, RIF, id[cart[x, x]]]]]

Out[24]= composite[DUP, rfx[x], inverse[DUP]]
```

Lemma. Transitivity result for **composite[DUP, x, inverse[DUP]]**.

```
In[29]:= TRANSITIVE[composite[DUP, x, inverse[DUP]]] // AssertTest

Out[29]= TRANSITIVE[composite[DUP, x, inverse[DUP]]] == TRANSITIVE[composite[Id, x]]

In[30]:= TRANSITIVE[composite[DUP, x_, inverse[DUP]]] := TRANSITIVE[composite[Id, x]]
```

Counterexample. There is a function **x** for which **domain[hom[x]]** is not transitive:

```
In[31]:= TRANSITIVE[domain[hom[composite[SWAP, RIF, id[cart[t, t]]]]]] /.
          t -> union[cart[set[0], set[set[0]]],
            cart[set[set[0]], set[succ[set[0]]]], id[succ[succ[set[0]]]]]

Out[31]= False
```

Comment. In general, the restriction **x = composite[SWAP, RIF, id[cart[t, t]]]** need not be a category. It is a category when **t** is a quasi-order. In this case the relation **domain[hom[x]]** is a quasi-order isomorphic to **t**.

```
In[32]:= domain[hom[composite[SWAP, RIF, id[cart[trv[rfx[x]], trv[rfx[x]]]]]]]

Out[32]= composite[DUP, trv[rfx[x]], inverse[DUP]]
```