special categories

Johan G. F. Belinfante
2009 January 11

```
In[1]:= SetDirectory["l:"]; << goedel.09jan09b;<< tools.m

        :Package Title: goedel.09jan09b         2009 January 9 at 5:50 p.m.

        It is now:  2009 Jan 11 at 10:31

        Loading Simplification Rules

        TOOLS.M                                      Revised 2008 December 26

        weightlimit = 40
```

summary

Some special examples of categories are considered in this notebook. The direct product of categories is a category. For any class **x**, there is a category whose class of morphisms is **x**. From any quasiorder, one can construct a category.

direct products

Lemma.

```
In[2]:= SubstTest[associative,
          composite[cross[assoc[u], assoc[v]], TWIST], {u → cat[x], v -> cat[y]}] // Reverse

Out[2]= associative[composite[cross[cat[x], cat[y]], TWIST]] == True

In[3]:= associative[composite[cross[cat[x_], cat[y_]], TWIST]] := True
```

Theorem. The direct product of two categories is a category.

```
In[4]:= SubstTest[implies,
          and[associative[t], FUNCTION[t], subclass[domain[t], cartsq[range[t]]],
           subclass[composite[t, inverse[FIRST], domain[t]], domain[t]],
           subclass[range[t], domain[cod[t]]], subclass[range[t], domain[dom[t]]]],
          category[t], t → direct[cat[x], cat[y]]] // Reverse

Out[4]= category[composite[cross[cat[x], cat[y]], TWIST]] == True

In[5]:= category[composite[cross[cat[x_], cat[y_]], TWIST]] := True
```

discrete categories

Definition: A category **x** is **discrete** if **range[x] = ids[x]**.

Lemma.

```
In[7]:= SubstTest[intersection, composite[t, id[s]],
          composite[id[s], t], {t → domain[x], s → ids[x]}] // Reverse

Out[7]= intersection[dom[x], inverse[cod[x]]] == id[ids[x]]

In[8]:= intersection[dom[x_], inverse[cod[x_]]] := id[ids[x]]
```

Lemma.

```
In[9]:= Assoc[cat[x], id[dom[cat[x]]], id[inverse[cod[cat[x]]]]]

Out[9]= composite[cat[x], id[id[ids[cat[x]]]]] == composite[id[ids[cat[x]]], inverse[DUP]]

In[10]:= composite[cat[x_], id[id[ids[cat[x_]]]]] := composite[id[ids[cat[x]]], inverse[DUP]]
```

Lemma.

```
In[11]:= Assoc[cat[x], id[domain[cat[x]]], id[cartsq[ids[cat[x]]]]] // Reverse

Out[11]= composite[cat[x], id[cart[ids[cat[x]], ids[cat[x]]]]] ==
          composite[id[ids[cat[x]]], inverse[DUP]]

In[12]:= composite[cat[x_], id[cart[ids[cat[x_]], ids[cat[x_]]]]] :=
          composite[id[ids[cat[x]]], inverse[DUP]]
```

Lemma.

```
In[13]:= Assoc[cat[x], id[domain[cat[x]]], id[cartsq[range[cat[x]]]]] // Reverse

Out[13]= composite[cat[x], id[cart[range[cat[x]], range[cat[x]]]]] == cat[x]

In[14]:= composite[cat[x_], id[cart[range[cat[x_]], range[cat[x_]]]]] := cat[x]
```

Theorem. If a category is discrete, then it is of the form **composite[id[t], inverse[DUP]]]**.

```
In[15]:= SubstTest[implies, equal[u, v],
           equal[restrict[t, cartsq[u], u], restrict[t, cartsq[v], v]],
           {t → cat[x], u → range[cat[x]], v → ids[cat[x]]}] // Reverse

Out[15]= or[equal[cat[x], composite[id[ids[cat[x]]], inverse[DUP]]],
           not[equal[ids[cat[x]], range[cat[x]]]]] == True

In[16]:= or[equal[cat[x_], composite[id[ids[cat[x_]]], inverse[DUP]]],
           not[equal[ids[cat[x_]], range[cat[x_]]]]] := True
```

Corollary. Restatement without the **cat** wrapper.

```
In[17]:= SubstTest[implies, equal[x, cat[t]], or[equal[x, composite[id[ids[x]], inverse[DUP]]],
           not[equal[ids[x], range[x]]]], t → x] // Reverse

Out[17]= or[equal[x, composite[id[ids[x]], inverse[DUP]]],
          not[category[x]], not[equal[ids[x], range[x]]]] == True

In[18]:= or[equal[x_, composite[id[ids[x_]], inverse[DUP]]],
          not[category[x_]], not[equal[ids[x_], range[x_]]]] := True
```

The remainder of this section is concerned with the converse: for every class **x**, the function **composite[id[x], inverse[-DUP]]** is (the composition law for) a discrete category.

Lemma.

```
In[19]:= ids[composite[id[x], inverse[DUP]]] // Normality

Out[19]= ids[composite[id[x], inverse[DUP]]] == x

In[20]:= ids[composite[id[x], inverse[DUP]]] := x
```

Lemma.

```
In[21]:= SubstTest[composite, id[ids[t]], domain[t], t -> composite[id[x], inverse[DUP]]]

Out[21]= dom[composite[id[x], inverse[DUP]]] == id[x]

In[22]:= dom[composite[id[x_], inverse[DUP]]] := id[x]
```

Lemma.

```
In[23]:= SubstTest[composite, id[ids[t]],
           inverse[domain[t]], t -> composite[id[x], inverse[DUP]]]

Out[23]= cod[composite[id[x], inverse[DUP]]] == id[x]

In[24]:= cod[composite[id[x_], inverse[DUP]]] := id[x]
```

Theorem. For any class **x** the function **composite[id[x], inverse[DUP]]** defines a discrete category.

```
In[25]:= SubstTest[implies,
           and[associative[t], FUNCTION[t], subclass[domain[t], cartsq[range[t]]],
            subclass[composite[t, inverse[FIRST], domain[t]], domain[t]],
            subclass[range[t], domain[cod[t]]], subclass[range[t], domain[dom[t]]]],
           category[t], t → composite[id[x], inverse[DUP]]] // Reverse

Out[25]= category[composite[id[x], inverse[DUP]]] == True

In[26]:= category[composite[id[x_], inverse[DUP]]] := True
```

The category **composite[id[x], inverse[DUP]]** is discrete:

```
In[27]:= implies[equal[t, composite[id[x], inverse[DUP]]], equal[range[t], ids[t]]]

Out[27]= True
```

Comment. It follows that any class **x** can be the class of morphisms for a category, and that any class **x** can be the class of identities for a category.

Observation. The direct product of discrete categories is discrete:

```
In[28]:= implies[and[equal[range[cat[x]], ids[cat[x]]], equal[range[cat[y]], ids[cat[y]]]],
           equal[range[direct[cat[x], cat[y]]], ids[direct[cat[x], cat[y]]]]]

Out[28]= True
```

Explicitly:

```
In[29]:= direct[composite[id[x], inverse[DUP]],
           composite[id[y], inverse[DUP]]] // VSTriNormality

Out[29]= composite[cross[composite[id[x], inverse[DUP]], composite[id[y], inverse[DUP]]],
           TWIST] == composite[id[cart[x, y]], inverse[DUP]]

In[30]:= composite[cross[composite[id[x_], inverse[DUP]], composite[id[y_], inverse[DUP]]],
           TWIST] := composite[id[cart[x, y]], inverse[DUP]]
```

singleton categories

Theorem. The cartesian cube of a class **x** is a category if and only if **x** is either empty or a singleton.

```
In[31]:= category[cart[cart[x, x], x]] // AssertTest

Out[31]= category[cart[cart[x, x], x]] == or[equal[0, x], member[x, range[SINGLETON]]]

In[32]:= category[cart[cart[x_, x_], x_]] := or[equal[0, x], member[x, range[SINGLETON]]]
```

Comment. The direct product of singleton caregories is another one:

```
In[33]:= direct[cart[cart[x, x], x], cart[cart[y, y], y]]

Out[33]= cart[cart[cart[x, y], cart[x, y]], cart[x, y]]
```

quasiorder categories

A quasiorder (or preorder) is a reflexive transitive relation. From any quasiorder one can construct a category.

Lemma.

```
In[34]:= (ids[composite[t, id[cartsq[x]]]] // Normality) /. t → composite[SWAP, RIF]
```

```
Out[34]= ids[composite[SWAP, RIF, id[cart[x, x]]]] == id[fix[x]]
```

```
In[35]:= ids[composite[SWAP, RIF, id[cart[x_, x_]]]] := id[fix[x]]
```

Lemma.

```
In[36]:= Map[inverse,
           SubstTest[composite, domain[t], id[ids[t]], t -> composite[SWAP, RIF, id[cart[x, x]]]]]
```

```
Out[36]= cod[composite[SWAP, RIF, id[cart[x, x]]]] ==
          composite[DUP, SECOND, id[composite[id[fix[x]], x]]]
```

```
In[37]:= cod[composite[SWAP, RIF, id[cart[x_, x_]]]] :=
          composite[DUP, SECOND, id[composite[id[fix[x]], x]]]
```

Lemma.

```
In[38]:= SubstTest[composite, id[ids[t]], domain[t], t -> composite[SWAP, RIF, id[cart[x, x]]]]
```

```
Out[38]= dom[composite[SWAP, RIF, id[cart[x, x]]]] ==
          composite[DUP, FIRST, id[composite[x, id[fix[x]]]]]
```

```
In[39]:= dom[composite[SWAP, RIF, id[cart[x_, x_]]]] :=
          composite[DUP, FIRST, id[composite[x, id[fix[x]]]]]
```

Theorem. From any quasiorder one can construct a category.

```
In[40]:= SubstTest[implies,
            and[associative[t], FUNCTION[t], subclass[domain[t], cartsq[range[t]]],
             subclass[composite[t, inverse[FIRST], domain[t]], domain[t]],
             subclass[range[t], domain[cod[t]]], subclass[range[t], domain[dom[t]]]],
            category[t], t → composite[SWAP, RIF, id[cartsq[trv[rfx[x]]]]] // Reverse
```

```
Out[40]= category[composite[SWAP, RIF, id[cart[trv[rfx[x]], trv[rfx[x]]]]]] == True
```

```
In[41]:= category[composite[SWAP, RIF, id[cart[trv[rfx[x_]], trv[rfx[x_]]]]]] := True
```

Note that the quasiorder itself can be recovered as the class of morphisms for this category. That is, each morphism for this category is an ordered pair **pair[u,v] ∈ trv[rfx[x]]**.

```
In[58]:= range[composite[SWAP, RIF, id[cart[trv[rfx[x]], trv[rfx[x]]]]]]
```

```
Out[58]= trv[rfx[x]]
```

Corollary. Restatement of the theorem without wrappers.

```
In[42]:= SubstTest[implies, equal[x, trv[rfx[t]]],
            category[composite[SWAP, RIF, id[cartsq[x]]], t → x] // Reverse
```

```
Out[42]= or[category[composite[SWAP, RIF, id[cart[x, x]]]],
            not[REFLEXIVE[x]], not[TRANSITIVE[x]]] == True
```

```
In[43]:= or[category[composite[SWAP, RIF, id[cart[x_, x_]]]],
           not[REFLEXIVE[x_]], not[TRANSITIVE[x_]]] := True
```

Corollary. In particular, one can construct a category from any partial order relation.

```
In[44]:= SubstTest[category,
           composite[SWAP, RIF, id[cart[trv[rfx[t]], trv[rfx[t]]]]], t → po[x]] // Reverse

Out[44]= category[composite[SWAP, RIF, id[cart[po[x], po[x]]]]] == True

In[45]:= category[composite[SWAP, RIF, id[cart[po[x_], po[x_]]]]] := True
```

Corollary. Also, every equivalence relation determines a category.

```
In[46]:= SubstTest[category,
           composite[SWAP, RIF, id[cart[trv[rfx[t]], trv[rfx[t]]]]], t → eqv[x]] // Reverse

Out[46]= category[composite[SWAP, RIF, id[cart[eqv[x], eqv[x]]]]] == True

In[47]:= category[composite[SWAP, RIF, id[cart[eqv[x_], eqv[x_]]]]] := True
```

Corollary. Any class **x** can be partially ordered by inclusion. The corresponding category is:

```
In[48]:= SubstTest[category,
           composite[SWAP, RIF, id[cart[po[t], po[t]]]], t → restrict[S, x, x]] // Reverse

Out[48]= category[composite[SWAP, RIF,
            id[cart[composite[id[x], S, id[x]], composite[id[x], S, id[x]]]]]] == True

In[49]:= category[composite[SWAP, RIF,
            id[cart[composite[id[x_], S, id[x_]], composite[id[x_], S, id[x_]]]]]] := True
```

Theorem. If **x** is a singleton, this reduces to a singleton category. For example:

```
In[50]:= (composite[SWAP, RIF, id[cartsq[po[t]]]] /. t → restrict[S, set[0], set[0]]) //
          VSTriNormality

Out[50]= composite[SWAP, RIF, id[cart[cart[set[0], set[0]], cart[set[0], set[0]]]]] ==
          cart[cart[cart[set[0], set[0]], cart[set[0], set[0]]], cart[set[0], set[0]]]

In[51]:= composite[SWAP, RIF, id[cart[cart[set[0], set[0]], cart[set[0], set[0]]]]] :=
          cart[cart[cart[set[0], set[0]], cart[set[0], set[0]]], cart[set[0], set[0]]]
```

Theorem. For the equivalence relation **id[x]**, one obtains a discrete category:

```
In[55]:= composite[SWAP, RIF, id[cartsq[id[x]]]] // ReifNormality

Out[55]= composite[SWAP, RIF, id[cart[id[x], id[x]]]] == composite[id[id[x]], inverse[DUP]]

In[56]:= composite[SWAP, RIF, id[cart[id[x_], id[x_]]]] := composite[id[id[x]], inverse[DUP]]
```