restrictions of categories

Johan G. F. Belinfante
2009 August 12

```
In[1]:= SetDirectory["l:"]; << goedel.09aug11a; << tools.m
```

```
:Package Title: goedel.09aug11a              2009 August 11 at 12:05 noon

It is now:  2009 Aug 12 at 9:12

Loading Simplification Rules

TOOLS.M                                    Revised 2009 July 2

weightlimit = 40
```

summary

This notebook contains a detailed formal derivation of the fact that the restriction **cat[x]** ∘ **id[y × y]** of an (arrows-only) category **cat[x]** to the cartesian square of a class **y** is a category if **y** is binary-closed under **cat[x]** and invariant under both **cod[cat[x]]** and **dom[cat[x]]**. This result is stated (without proof) in Mac Lane's book (on page 15).

```
In[2]:= "Saunders Mac Lane, Categories for the
          Working Mathematician, Springer-Verlag, New York, 1971.";
```

The following concise characterization of a category is the basis for the derivation:

```
In[3]:= and[associative[x], FUNCTION[x],
          subclass[composite[x, inverse[FIRST], domain[x]], domain[x]],
          subclass[domain[x], cart[range[x], range[x]]], subclass[range[x], range[hom[x]]]]
```

```
Out[3]= category[x]
```

No new rewrite rule is needed to show that a restriction of a category is a function, but new rules are needed for each of the other four conditions.

```
In[4]:= FUNCTION[composite[cat[x], id[cart[y, y]]]]
```

```
Out[4]= True
```

an additional hypothesis

It is sometimes convenient to add the additional hypothesis that **y** be a subclass of **range[cat[x]]**. This hypothesis can easily be eliminated at the end. In this section some simplification rules are derived to help with this elimination.

Lemma.

```
In[5]:= ImageComp[dom[cat[x]], id[range[cat[x]]], y] // Reverse

Out[5]= image[dom[cat[x]], intersection[y, range[cat[x]]]] == image[dom[cat[x]], y]

In[6]:= image[dom[cat[x_]], intersection[y_, range[cat[x_]]]] := image[dom[cat[x]], y]
```

Lemma.

```
In[7]:= ImageComp[cod[cat[x]], id[range[cat[x]]], y] // Reverse

Out[7]= image[cod[cat[x]], intersection[y, range[cat[x]]]] == image[cod[cat[x]], y]

In[8]:= image[cod[cat[x_]], intersection[y_, range[cat[x_]]]] := image[cod[cat[x]], y]
```

Lemma.

```
In[9]:= ImageComp[cat[x], id[cartsq[range[cat[x]]]], cart[y, y]] // Reverse

Out[9]= image[cat[x], cart[intersection[y, range[cat[x]]], intersection[y, range[cat[x]]]]] ==
         image[cat[x], cart[y, y]]

In[10]:= image[cat[x_], cart[intersection[y_, range[cat[x_]]],
            intersection[y_, range[cat[x_]]]]] := image[cat[x], cart[y, y]]
```

Lemma.

```
In[11]:= Assoc[cat[x], id[cartsq[range[cat[x]]]], id[cartsq[y]]]

Out[11]= composite[cat[x],
           id[cart[intersection[y, range[cat[x]]], intersection[y, range[cat[x]]]]] ==
          composite[cat[x], id[cart[y, y]]]

In[12]:= composite[cat[x_],
           id[cart[intersection[y_, range[cat[x_]]], intersection[y_, range[cat[x_]]]]] :=
          composite[cat[x], id[cart[y, y]]]
```

---

duality lemmas

To shorten the derivation, some use of duality is convenient. If **x** is a category, so is its opposite, **flip[x] = composite[x, SWAP]**. The class of identity morphisms is the same for a category and its opposite. For a restriction of a category, a rewrite rule transforms the opposite of a restriction to a restriction of the opposite. To cope with this, the following result is needed.

Theorem.

```
In[13]:= SubstTest[ids, flip[t], t → composite[x, id[cart[y, y]]]] // Reverse

Out[13]= ids[composite[x, SWAP, id[cart[y, y]]]] == ids[composite[x, id[cart[y, y]]]]
```

```
In[14]:= ids[composite[x_, SWAP, id[cart[y_, y_]]]] := ids[composite[x, id[cart[y, y]]]]
```

A similar phenomenon occurs for **cod** and **dom**. Only one of the following two rewrite rules is actually needed below. The other is supplied merely for the sake of completeness.

Theorem.

```
In[15]:= SubstTest[dom, flip[t], t → composite[x, id[cart[y, y]]]] // Reverse
Out[15]= dom[composite[x, SWAP, id[cart[y, y]]]] == cod[composite[x, id[cart[y, y]]]]

In[16]:= dom[composite[x_, SWAP, id[cart[y_, y_]]]] := cod[composite[x, id[cart[y, y]]]]
```

Theorem.

```
In[17]:= SubstTest[cod, flip[t], t → composite[x, id[cart[y, y]]]] // Reverse
Out[17]= cod[composite[x, SWAP, id[cart[y, y]]]] == dom[composite[x, id[cart[y, y]]]]

In[18]:= cod[composite[x_, SWAP, id[cart[y_, y_]]]] := dom[composite[x, id[cart[y, y]]]]
```

associativity

The following theorem suffices to deal with the associativity hypothesis.

Theorem. The restriction of a category to any binary-closed class is associative.

```
In[19]:= SubstTest[implies, and[associative[t], subclass[image[t, cart[y, y]], y]],
           associative[composite[t, id[cart[y, y]]]], t → cat[x]] // Reverse
Out[19]= or[associative[composite[cat[x], id[cart[y, y]]]],
           not[subclass[image[cat[x], cart[y, y]], y]]] == True

In[20]:= or[associative[composite[cat[x_], id[cart[y_, y_]]]],
           not[subclass[image[cat[x_], cart[y_, y_]], y_]]] := True
```

the divisibility axiom

Mac Lane's second arrows-only axiom for a category can be stated in the following form: if a morphism is composable with some left-divisor of another morphism, then it is composable with that morphism. Eliminating the morphism variables from this statement yields the requirement that the composite of the divisibility relation **composite[x, inverse[FIRST]]** for a category **x** and the composability relation **domain[x]** be contained in the composability relation. When this axiom is stated for a restriction of a category to a cartesian square, this axiom naturally breaks apart into two statements, one of which is always true.

Theorem.

```
In[21]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> composite[cat[x], id[cart[y, y]], inverse[FIRST], domain[cat[x]], id[y]], v ->
             composite[cat[x], inverse[FIRST], domain[cat[x]]], w → domain[cat[x]]}] // Reverse

Out[21]= subclass[composite[cat[x], id[cart[y, y]], inverse[FIRST], domain[cat[x]], id[y]],
           domain[cat[x]]] == True

In[22]:= subclass[composite[cat[x_], id[cart[y_, y_]],
            inverse[FIRST], domain[cat[x_]], id[y_]], domain[cat[x_]]] := True
```

The other statement holds when **y** is binary-closed under **cat[x]**. This requires no additional rewrite rule:

```
In[23]:= implies[subclass[image[cat[x], cart[y, y]], y],
           subclass[composite[t, inverse[FIRST], domain[t]], domain[t]]] /.
         t → composite[cat[x], id[cart[y, y]]]

Out[23]= True
```

the domain condition

This section deals with the requirement that the domain of a category be a subclass of the cartesian square of its range. This category axiom is satisfied for a restriction **composite[cat[x], id[cart[y, y]]** of a category **cat[x]** to a class **y** provided that one replaces the binary-closure condition on **y** with the slightly stronger condition that the equation **y = image[cat[x], cart[y, y]]** holds:

```
In[24]:= implies[equal[image[cat[x], cart[y, y]], y],
           subclass[domain[t], cart[range[t], range[t]]]] /.
         t → composite[cat[x], id[cart[y, y]]]

Out[24]= True
```

It will be shown below that this stronger condition follows from binary closure when one requires that **y** be invariant under **dom[cat[x]]**. The strategy will be to show that any morphism **u ∈ y** belongs to **image[cat[x], cart[y, y]]** under the condition that the class **y ⊂ range[cat[x]]** be binary-closed under **cat[x]** and invariant under **dom[cat[x]]**.

Lemma.

```
In[25]:= SubstTest[implies,
           and[invariant[funpart[t], y], member[u, y], member[u, domain[funpart[t]]]],
           member[APPLY[funpart[t], u], y], t → dom[cat[x]]] // Reverse

Out[25]= or[member[APPLY[dom[cat[x]], u], y], not[member[u, y]],
           not[member[u, range[cat[x]]]], not[subclass[image[dom[cat[x]], y], y]]] == True

In[26]:= (% /. {u → u_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma.

```
In[27]:= SubstTest[implies, member[w, intersection[z, domain[funpart[t]]]],
           member[APPLY[funpart[t], w], image[funpart[t], z]],
           {t → cat[x], w → pair[u, APPLY[dom[cat[x]], u]], z → cart[y, y]}] // Reverse

Out[27]= or[member[u, image[cat[x], cart[y, y]]], not[member[u, y]],
           not[member[u, range[cat[x]]]], not[member[APPLY[dom[cat[x]], u], y]]] == True

In[28]:= (% /. {u → u_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If **y ⊂ range[cat[x]]** is invariant under **dom[cat[x]]**, then any morphism **u ∈ y** belongs to **image[cat[x], cart[y, y]]**.

```
In[29]:= Map[not, SubstTest[and, implies[and[p1, p3], p4], implies[and[p1, p2, p4], p5],
            implies[and[p1, p4, p5], p6], not[implies[and[p1, p2, p3], p6]], {p1 → member[u, y],
             p2 → subclass[image[dom[cat[x]], y], y], p3 → subclass[y, range[cat[x]]],
             p4 -> member[u, range[cat[x]]], p5 → member[APPLY[dom[cat[x]], u], y],
             p6 → member[u, image[cat[x], cart[y, y]]]}]] // Reverse

Out[29]= or[member[u, image[cat[x], cart[y, y]]], not[member[u, y]],
           not[subclass[y, range[cat[x]]]], not[subclass[image[dom[cat[x]], y], y]]] == True

In[30]:= (% /. {u → u_, x → x_, y → y_}) /. Equal → SetDelayed
```

The next step is to eliminate the morphism variable **u** from this statement.

Theorem.

```
In[31]:= Map[equal[V, #] &, SubstTest[class, u, or[member[u, v],
            not[member[u, y]], not[subclass[y, w]], not[subclass[image[t, y], y]]],
           {v -> image[cat[x], cart[y, y]], w → range[cat[x]], t → dom[cat[x]]}]]

Out[31]= or[not[subclass[y, range[cat[x]]]], not[subclass[image[dom[cat[x]], y], y]],
           subclass[y, image[cat[x], cart[y, y]]]] == True

In[32]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 -> and[subclass[y, range[cat[x]]], subclass[image[dom[cat[x]], y], y]],
             p2 -> subclass[y, image[cat[x], cart[y, y]]], p3 -> subclass[intersection[
                y, image[domain[cat[x]], y]], image[cat[x], cart[y, y]]]}]] // Reverse

Out[34]= or[not[subclass[y, range[cat[x]]]], not[subclass[image[dom[cat[x]], y], y]],
           subclass[intersection[y, image[domain[cat[x]], y]], image[cat[x], cart[y, y]]]] == True

In[35]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary. (A similar result, with **domain[x]** replaced with its inverse.)

```
In[36]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> and[subclass[y, range[cat[x]]], subclass[image[dom[cat[x]], y], y]],
            p2 -> subclass[y, image[cat[x], cart[y, y]]], p3 -> subclass[intersection[y,
              image[inverse[domain[cat[x]]], y]], image[cat[x], cart[y, y]]]}]] // Reverse

Out[36]= or[not[subclass[y, range[cat[x]]]], not[subclass[image[dom[cat[x]], y], y]],
          subclass[intersection[y, image[inverse[domain[cat[x]]], y]],
           image[cat[x], cart[y, y]]]] == True

In[37]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Restatement. If **y ⊂ range[cat[x]]** is invariant under **dom[cat[x]]**, then the domain condition holds for **composite[cat[x], id[cart[y, y]]]**.

```
In[38]:= (implies[and[subclass[y, range[cat[x]]], subclass[image[dom[cat[x]], y], y]],
            subclass[domain[t], cart[range[t], range[t]]]] /.
           t → composite[cat[x], id[cart[y, y]]]) // not // not

Out[38]= True
```

the hom condition

The final axiom to be verified is the hom condition **range[x] ⊂ range[hom[x]]**. (This axiom assures that every morphism is composable on either side with an identity morphism.) Since **range[hom[x]]** is the intersection of the domains of **cod[x]** and **dom[x]**, this condition can be broken up into two conditions, one of which is the dual of the other.

```
In[39]:= SubstTest[subclass, range[x], intersection[u, v],
          {u → domain[cod[x]], v → domain[dom[x]]}]

Out[39]= and[subclass[range[x], domain[cod[x]]], subclass[range[x], domain[dom[x]]]] ==
          subclass[range[x], range[hom[x]]]
```

On account of this, it suffices to consider just one of these two conditions, and then one can apply duality to get the other one.

Lemma.

```
In[40]:= IminComp[id[ids[x]], domain[x], y] // Reverse

Out[40]= image[inverse[domain[x]], intersection[y, ids[x]]] == image[inverse[dom[x]], y]

In[41]:= image[inverse[domain[x_]], intersection[y_, ids[x_]]] := image[inverse[dom[x]], y]
```

Lemma. A connection between invariance and subvariance that holds for functions is specialized to the function **dom[cat[x]]**.

```
In[42]:= SubstTest[implies, and[invariant[funpart[t], y], subclass[y, domain[funpart[t]]]],
           subvariant[inverse[funpart[t]], y], t → dom[cat[x]]] // Reverse

Out[42]= or[not[subclass[y, range[cat[x]]]], not[subclass[image[dom[cat[x]], y], y]],
           subclass[y, image[inverse[dom[cat[x]]], y]]] == True

In[43]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The inclusion **y ⋂ ids[x] ⊂ ids[x ∘ id[y × y]]** suffices to eliminate the class of identities for the restriction.

Lemma.

```
In[44]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → inverse[domain[x]], u -> intersection[y, ids[x]],
            v → intersection[y, ids[composite[x, id[cart[y, y]]]]]}] // Reverse

Out[44]= subclass[image[inverse[dom[x]], y],
           image[inverse[domain[x]], intersection[y, ids[composite[x, id[cart[y, y]]]]]]] == True

In[45]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[46]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
            {p1 → and[subclass[y, range[cat[x]]], subclass[image[dom[cat[x]], y], y]],
             p2 -> subclass[y, image[inverse[dom[cat[x]]], y]],
             p3 -> subclass[y, image[inverse[domain[cat[x]]],
                intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]}]] // Reverse

Out[46]= or[not[subclass[y, range[cat[x]]]],
           not[subclass[image[dom[cat[x]], y], y]], subclass[y, image[inverse[domain[cat[x]]],
             intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]] == True

In[47]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Dual Theorem.

```
In[48]:= SubstTest[or, not[subclass[y, range[cat[t]]]], not[subclass[image[dom[cat[t]], y], y]],
           subclass[y, image[inverse[domain[cat[t]]], intersection[y,
              ids[composite[cat[t], id[cart[y, y]]]]]]], t → flip[cat[x]]] // Reverse

Out[48]= or[not[subclass[y, range[cat[x]]]],
           not[subclass[image[cod[cat[x]], y], y]], subclass[y, image[domain[cat[x]],
             intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]] == True

In[49]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[50]:= Map[implies[or[not[equal[y, image[cat[x], cart[y, y]]]],
           subclass[image[cat[x], cart[y, y]], image[inverse[domain[cat[x]]],
            intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]]], #] &,
         Map[implies[equal[image[cat[x], cart[y, y]], y],
           subclass[image[cat[x], cart[y, y]], #]] &, SubstTest[image,
          inverse[domain[t]], ids[t], t → composite[cat[x], id[cart[y, y]]]]] // MapNotNot]

Out[50]= or[not[equal[y, image[cat[x], cart[y, y]]]],
          not[subclass[image[cat[x], cart[y, y]], image[inverse[domain[cat[x]]],
             intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]], subclass[
           image[cat[x], cart[y, y]], domain[dom[composite[cat[x], id[cart[y, y]]]]]]] == True

In[51]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[52]:= Map[not,
          SubstTest[and, implies[p1, p2], implies[and[p0, p2], p3], implies[and[p0, p3], p4],
           not[implies[and[p0, p1], p4]], {p0 -> equal[y, image[cat[x], cart[y, y]]],
            p1 → and[subclass[y, range[cat[x]]], subclass[image[dom[cat[x]], y], y]],
            p2 -> subclass[y, image[inverse[domain[cat[x]]],
               intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]],
            p3 -> subclass[image[cat[x], cart[y, y]], image[inverse[domain[cat[x]]],
               intersection[y, ids[composite[cat[x], id[cart[y, y]]]]]]],
            p4 -> subclass[image[cat[x], cart[y, y]], domain[
               dom[composite[cat[x], id[cart[y, y]]]]]]}]] // Reverse

Out[52]= or[not[equal[y, image[cat[x], cart[y, y]]]], not[subclass[y, range[cat[x]]]],
          not[subclass[image[dom[cat[x]], y], y]], subclass[image[cat[x], cart[y, y]],
           domain[dom[composite[cat[x], id[cart[y, y]]]]]] == True

In[53]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Dual.

```
In[54]:= SubstTest[or, not[equal[y, image[cat[t], cart[y, y]]]],
          not[subclass[y, range[cat[t]]]], not[subclass[image[dom[cat[t]], y], y]],
          subclass[image[cat[t], cart[y, y]], domain[dom[composite[cat[t], id[cart[y, y]]]]]],
          t → flip[cat[x]]] // Reverse

Out[54]= or[not[equal[y, image[cat[x], cart[y, y]]]], not[subclass[y, range[cat[x]]]],
          not[subclass[image[cod[cat[x]], y], y]], subclass[image[cat[x], cart[y, y]],
           domain[cod[composite[cat[x], id[cart[y, y]]]]]] == True

In[55]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[56]:= Map[implies[#, subclass[image[cat[x], cart[y, y]],
             range[hom[composite[cat[x], id[cart[y, y]]]]]]] &,
          SubstTest[subclass, u, intersection[v, w], {u -> image[cat[x], cart[y, y]],
             v -> domain[dom[t]], w -> domain[cod[t]]}] /. t -> composite[cat[x], id[cart[y, y]]]]

Out[56]= or[not[subclass[image[cat[x], cart[y, y]],
             domain[cod[composite[cat[x], id[cart[y, y]]]]]]],
           not[subclass[image[cat[x], cart[y, y]],
             domain[dom[composite[cat[x], id[cart[y, y]]]]]], subclass[
            image[cat[x], cart[y, y]], range[hom[composite[cat[x], id[cart[y, y]]]]]]] == True

In[57]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[58]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 -> and[equal[y, image[cat[x], cart[y, y]]], subclass[y, range[cat[x]]],
               subclass[image[cod[cat[x]], y], y], subclass[image[dom[cat[x]], y], y]],
             p2 -> subclass[image[cat[x], cart[y, y]],
               domain[cod[composite[cat[x], id[cart[y, y]]]]]],
             p3 -> subclass[image[cat[x], cart[y, y]],
               domain[dom[composite[cat[x], id[cart[y, y]]]]]],
             p4 -> subclass[image[cat[x], cart[y, y]],
               range[hom[composite[cat[x], id[cart[y, y]]]]]]}] // Reverse

Out[58]= or[not[equal[y, image[cat[x], cart[y, y]]]],
           not[subclass[y, range[cat[x]]]], not[subclass[image[cod[cat[x]], y], y]],
           not[subclass[image[dom[cat[x]], y], y]], subclass[image[cat[x], cart[y, y]],
            range[hom[composite[cat[x], id[cart[y, y]]]]]] == True

In[59]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[60]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p2, p3], p4],
            implies[and[p1, p4], p5], not[implies[and[p1, p2], p5]],
            {p1 -> and[subclass[y, range[cat[x]]], subclass[image[cod[cat[x]], y], y],
               subclass[image[dom[cat[x]], y], y]], p2 → subclass[image[cat[x], cart[y, y]], y],
             p3 → subclass[y, image[cat[x], cart[y, y]]],
             p4 → equal[y, image[cat[x], cart[y, y]]], p5 -> subclass[image[cat[x], cart[y, y]],
               range[hom[composite[cat[x], id[cart[y, y]]]]]]}] // Reverse

Out[60]= or[not[subclass[y, range[cat[x]]]],
           not[subclass[image[cat[x], cart[y, y]], y]], not[subclass[image[cod[cat[x]], y], y]],
           not[subclass[image[dom[cat[x]], y], y]], subclass[image[cat[x], cart[y, y]],
            range[hom[composite[cat[x], id[cart[y, y]]]]]] == True

In[61]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Subcategory Theorem. (Mac Lane's sufficient conditions for a restriction of a category to be a category.)

```
In[62]:= Map[not, Map[
           not[implies[and[subclass[y, range[cat[x]]], subclass[image[cat[x], cart[y, y]], y],
               subclass[image[cod[cat[x]], y], y], subclass[image[dom[cat[x]], y], y]], #]] &,
           SubstTest[and, associative[t], FUNCTION[t], subclass[
             composite[t, inverse[FIRST], domain[t]], domain[t]],
            subclass[domain[t], cart[range[t], range[t]]], subclass[range[t], range[hom[t]]],
            t -> composite[cat[x], id[cart[y, y]]]]]]]
```

```
Out[62]= or[category[composite[cat[x], id[cart[y, y]]]], not[subclass[y, range[cat[x]]]],
           not[subclass[image[cat[x], cart[y, y]], y]], not[subclass[image[cod[cat[x]], y], y]],
           not[subclass[image[dom[cat[x]], y], y]]] == True
```

```
In[63]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The inessential hypothesis **y ⊂ range[cat[x]]** can be eliminated from Mac Lane's condition.

Corollary. The restriction of a category **cat[x]** to the cartesian square of a class **y** is a category if **y** is binary-closed under **cat[x]** and invariant under both **cod[cat[x]]** and **dom[cat[x]]**.

```
In[64]:= SubstTest[or, category[composite[cat[x], id[cart[t, t]]]],
           not[subclass[t, range[cat[x]]]], not[subclass[image[cat[x], cart[t, t]], t]],
           not[subclass[image[cod[cat[x]], t], t]], not[subclass[image[dom[cat[x]], t], t]],
           t → intersection[y, range[cat[x]]]] // Reverse
```

```
Out[64]= or[category[composite[cat[x], id[cart[y, y]]]],
           not[subclass[image[cat[x], cart[y, y]], y]], not[subclass[image[cod[cat[x]], y], y]],
           not[subclass[image[dom[cat[x]], y], y]]] == True
```

```
In[65]:= or[category[composite[cat[x_], id[cart[y_, y_]]]],
           not[subclass[image[cat[x_], cart[y_, y_]], y_]],
           not[subclass[image[cod[cat[x_]], y_], y_]],
           not[subclass[image[dom[cat[x_]], y_], y_]]] := True
```

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

```
In[66]:= SubstTest[implies, equal[x, cat[t]], or[category[composite[x, id[cart[y, y]]]],
            not[subclass[image[x, cart[y, y]], y]], not[subclass[image[cod[x], y], y]],
            not[subclass[image[dom[x], y], y]]], t → x] // Reverse
```

```
Out[66]= or[category[composite[x, id[cart[y, y]]]],
           not[category[x]], not[subclass[image[x, cart[y, y]], y]],
           not[subclass[image[cod[x], y], y]], not[subclass[image[dom[x], y], y]]] == True
```

```
In[68]:= or[category[composite[x_, id[cart[y_, y_]]]], not[category[x_]],
           not[subclass[image[cod[x_], y_], y_]], not[subclass[image[dom[x_], y_], y_]],
           not[subclass[image[x_, cart[y_, y_]], y_]]] := True
```

an application: the core of a category

Theorem. The restriction of a category to the class of its invertible morphisms (isomorphisms) is a category.

```
In[69]:= SubstTest[or, category[composite[cat[x], id[cart[y, y]]]],
           not[subclass[image[cat[x], cart[y, y]], y]], not[subclass[image[cod[cat[x]], y], y]],
           not[subclass[image[dom[cat[x]], y], y]], y → domain[inv[cat[x]]]] // Reverse

Out[69]= category[composite[cat[x], id[cart[domain[inv[cat[x]]], domain[inv[cat[x]]]]]]] == True

In[70]:= category[
           composite[cat[x_], id[cart[domain[inv[cat[x_]]], domain[inv[cat[x_]]]]]] := True
```

Corollary. (Restatement with the wrapper **cat[x]** replaced by the predicate **category[x]**.)

```
In[71]:= SubstTest[implies, equal[x, cat[t]],
           category[composite[x, id[cart[domain[inv[x]], domain[inv[x]]]]], t → x] // Reverse

Out[71]= or[category[composite[x, id[cart[domain[inv[x]], domain[inv[x]]]]]],
           not[category[x]]] == True

In[72]:= or[category[composite[x_, id[cart[domain[inv[x_]], domain[inv[x_]]]]]],
           not[category[x_]]] := True
```