identity elements for categories

Johan G. F. Belinfante
2009 January 13

```
In[1]:= SetDirectory["l:"]; << goedel.09jan12a;<< tools.m

        :Package Title: goedel.09jan12a        2009 January 12 at 1:55 p.m.

        It is now:  2009 Jan 13 at 12:20

        Loading Simplification Rules

        TOOLS.M                                        Revised 2008 December 26

        weightlimit = 40
```

summary

In MacLane's **arrows-only** axioms for category theory, an identity morphism is defined to be a morphism **u** such that **f · u = f** whenever **f** and **u** are composable, and **u · g = g** whenever **u** and **g** are composable.

```
In[2]:= "Saunders MacLane, Categories for the Working
         Mathematician, Springer–Verlag, New York, 1971.  See page 9.";
```

The **GOEDEL** program contains a simple variant of MacLane's definition. For any category composition law **x**, the class of morphisms is defined to be **range[x]**. Morphisms **u** and **v** are composable if **pair[u, v] ∈ domain[x]**. The composite **u · v** of composable morphisms **u** and **v** is the morphism **APPLY[x, PAIR[u, v]]**. For convenience, the class **ids[x]** of identity morphisms is defined for any class **x** (not just for categories) as the class of members **u ∈ fix[domain[x]]** that are both left and right neutral in the precise sense that **x ∘ RIGHT[u] ⊂ Id** and **x ∘ LEFT[u] ⊂ Id**. The additional condition **u ∈ fix[domain[x]]** implies that identity morphisms are idempotent **u · u = u**. In this notebook it is shown that weaker conditions suffice to characterize identity morphisms in a category. In particular, the idempotence condition can be omitted, and it suffices to require only one-sided neutrality.

derivation

Lemma.

```
In[3]:= SubstTest[implies, subclass[s, t], subclass[image[s, z], image[t, z]],
          {s → composite[cat[x], RIGHT[y]], t → Id, z → set[APPLY[cod[cat[x]], y]]}] // Reverse

Out[3]= or[equal[y, APPLY[cod[cat[x]], y]], not[member[y, range[cat[x]]]],
          not[subclass[composite[cat[x], RIGHT[y]], Id]]] == True

In[4]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. Every right-neutral morphism is an identity morphism.

```
In[5]:= Map[not, SubstTest[and, implies[p1, p2],
          implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
          {p1 → and[member[u, range[cat[x]]], subclass[composite[cat[x], RIGHT[u]], Id]],
           p2 -> equal[u, APPLY[cod[cat[x]], u]], p3 → member[u, V],
           p4 → member[u, ids[cat[x]]]}]] // Reverse

Out[5]= or[member[u, ids[cat[x]]], not[member[u, range[cat[x]]]],
          not[subclass[composite[cat[x], RIGHT[u]], Id]]] == True

In[6]:= or[member[u_, ids[cat[x_]]], not[member[u_, range[cat[x_]]]],
          not[subclass[composite[cat[x_], RIGHT[u_]], Id]]] := True
```

Lemma.

```
In[7]:= Map[equal[V, #] &, SubstTest[class, y,
          or[member[y, s], not[member[y, r]], not[subclass[composite[t, RIGHT[y]], Id]]],
          {r → range[cat[x]], s → ids[cat[x]], t → cat[x]}]]

Out[7]= subclass[range[cat[x]],
          union[ids[cat[x]], range[fix[composite[inverse[FIRST], Di, cat[x]]]]]] == True

In[8]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[9]:= SubstTest[implies, subclass[u, v], subclass[range[u], range[v]],
          {u -> fix[composite[inverse[FIRST], Di, cat[x]]], v → domain[cat[x]]}] // Reverse

Out[9]= subclass[range[fix[composite[inverse[FIRST], Di, cat[x]]]], range[cat[x]]] == True

In[10]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. A formula for the class **range[fix[composite[inverse[FIRST], Di, cat[x]]]]**.

```
In[11]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → dif[range[cat[x]], ids[cat[x]]],
           v → range[fix[composite[inverse[FIRST], Di, cat[x]]]]}]

Out[11]= equal[intersection[complement[ids[cat[x]]], range[cat[x]]],
           range[fix[composite[inverse[FIRST], Di, cat[x]]]]] == True

In[12]:= range[fix[composite[inverse[FIRST], Di, cat[x_]]]] :=
          intersection[complement[ids[cat[x]]], range[cat[x]]]
```

Dual result. A formula for **domain[fix[composite[inverse[SECOND], Di, cat[x]]]]**.

```
In[13]:= SubstTest[range,
           fix[composite[inverse[FIRST], Di, cat[t]]], t → flip[cat[x]]] // Reverse

Out[13]= domain[fix[composite[inverse[SECOND], Di, cat[x]]]] ==
          intersection[complement[ids[cat[x]]], range[cat[x]]]
```

```
In[14]:= domain[fix[composite[inverse[SECOND], Di, cat[x_]]]] :=
          intersection[complement[ids[cat[x]]], range[cat[x]]]
```

Corollary. Every left-neutral morphism is an identity morphism.

```
In[15]:= Map[implies[subclass[composite[cat[x], LEFT[u]], Id], not[#]] &, SubstTest[member,
           u, domain[fix[composite[inverse[SECOND], Di, t]]], t -> cat[x]]] // Reverse

Out[15]= or[member[u, ids[cat[x]]], not[member[u, range[cat[x]]]],
           not[subclass[composite[cat[x], LEFT[u]], Id]]] == True

In[16]:= or[member[u_, ids[cat[x_]]], not[member[u_, range[cat[x_]]]],
           not[subclass[composite[cat[x_], LEFT[u_]], Id]]] := True
```

MacLane's characterization

Temporary Lemma. (Standardization of fix-point expressions. This rule could lead to looping in general, but it is safe here.)

```
In[17]:= fix[composite[inverse[cat[x]], y]] // InvertFixTest

Out[17]= fix[composite[inverse[cat[x]], y]] == fix[composite[inverse[y], cat[x]]]

In[18]:= fix[composite[inverse[cat[x_]], y_]] := fix[composite[inverse[y], cat[x]]]
```

Lemma. (Simplification rule.)

```
In[19]:= equal[intersection[complement[domain[x]], fix[composite[w, x]]], 0]

Out[19]= True

In[20]:= intersection[complement[domain[x_]], fix[composite[w_, x_]]] := 0
```

In this section MacLane's characterization of identity morphisms is translated into a form compatible with the **GOEDEL** program. Again it suffices to require only one of MacLane's two conditions. When one holds, the other does also. First note that the class of pairs of morphisms satisfying $\mathbf{u \cdot g = g}$ is:

```
class[pair[u, g], equal[APPLY[funpart[t], PAIR[u, g]], g]] /. t → cat[x]

fix[composite[inverse[SECOND], cat[x]]]
```

The class of all morphisms **u** such that $\mathbf{u \cdot g = g}$ whenever **u** and **g** are composable is:

```
In[21]:= (intersection[range[cat[x]],
            class[u, forall[g, implies[member[pair[u, g], y], member[pair[u, g], z]]]]] /.
           {y → domain[cat[x]], z -> fix[composite[inverse[SECOND], cat[x]]]}) // InvertFix

Out[21]= intersection[complement[fix[composite[inverse[domain[cat[x]]],
            complement[fix[composite[inverse[SECOND], cat[x]]]]]]], range[cat[x]]]
```

Lemma. Simplification rule.

```
In[22]:= equal[intersection[ids[x], range[x]], ids[x]]

Out[22]= True

In[23]:= intersection[ids[x_], range[x_]] := ids[x]
```

Theorem. MacLane's characterization of identities.

```
In[24]:= Map[dif[range[cat[x]], domain[complement[#]]] &, SubstTest[fix, complement[t],
           t -> composite[inverse[SECOND], Di, cat[x]]]] // InvertFix // Reverse

Out[24]= intersection[complement[fix[composite[inverse[domain[cat[x]]], complement[
           fix[composite[inverse[SECOND], cat[x]]]]]]], range[cat[x]]] == ids[cat[x]]

In[25]:= intersection[complement[fix[composite[inverse[domain[cat[x_]]], complement[
           fix[composite[inverse[SECOND], cat[x_]]]]]]], range[cat[x_]]] := ids[cat[x]]
```

Dual result:

```
In[26]:= SubstTest[dif, range[cat[t]], fix[composite[inverse[domain[cat[t]]],
           complement[fix[composite[inverse[SECOND], cat[t]]]]]],
          t → flip[cat[x]]] // Reverse // InvertFix

Out[26]= intersection[
          complement[fix[composite[complement[fix[composite[inverse[FIRST], cat[x]]]],
            inverse[domain[cat[x]]]]]], range[cat[x]]] == ids[cat[x]]

In[27]:= intersection[
          complement[fix[composite[complement[fix[composite[inverse[FIRST], cat[x_]]]],
            inverse[domain[cat[x_]]]]]], range[cat[x_]]] := ids[cat[x]]
```

dom and cod rules

In the remainder of this notebook some special results involving **dom** and **cod** are derived.

Theorem.

```
In[28]:= Assoc[cat[x], id[domain[cat[x]]], id[cart[V, ids[cat[x]]]]] // Reverse

Out[28]= composite[cat[x], id[cart[V, ids[cat[x]]]]] == composite[FIRST, id[dom[cat[x]]]]

In[29]:= composite[cat[x_], id[cart[V, ids[cat[x_]]]]] := composite[FIRST, id[dom[cat[x]]]]
```

Dual result.

```
In[30]:= Map[flip, SubstTest[composite, cat[t], id[cart[V, ids[cat[t]]]], t → flip[cat[x]]]] //
          Reverse

Out[30]= composite[cat[x], id[cart[ids[cat[x]], V]]] ==
          composite[SECOND, id[inverse[cod[cat[x]]]]]
```

```
In[31]:= composite[cat[x_], id[cart[ids[cat[x_]], V]]] :=
          composite[SECOND, id[inverse[cod[cat[x]]]]]
```

Lemma. (The case that **w** is a morphism.)

```
In[32]:= SubstTest[implies, member[t, ids[cat[x]]],
           subclass[composite[cat[x], LEFT[t]], Id], t → APPLY[cod[cat[x]], w]] // Reverse

Out[32]= or[not[member[w, range[cat[x]]]],
           subclass[composite[cat[x], LEFT[APPLY[cod[cat[x]], w]]], Id]] == True

In[33]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma. (The case that **w** is not a morphism.)

```
In[34]:= Map[or[member[w, range[cat[x]]], #] &, SubstTest[implies, empty[t],
            subclass[t, Id], t -> composite[cat[x], LEFT[APPLY[cod[cat[x]], w]]]]] // Reverse

Out[34]= or[member[w, range[cat[x]]],
           subclass[composite[cat[x], LEFT[APPLY[cod[cat[x]], w]]], Id]] == True

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

Theorem. (Combine the above two cases.)

```
In[36]:= SubstTest[and, implies[p, q], or[p, q], {p -> member[w, range[cat[x]]],
           q -> subclass[composite[cat[x], LEFT[APPLY[cod[cat[x]], w]]], Id]}]

Out[36]= subclass[composite[cat[x], LEFT[APPLY[cod[cat[x]], w]]], Id] == True

In[37]:= subclass[composite[cat[x_], LEFT[APPLY[cod[cat[x_]], w_]]], Id] := True
```

Corollary. (Dual result.)

```
In[38]:= SubstTest[subclass,
           composite[cat[t], LEFT[APPLY[cod[cat[t]], w]]], Id, t → flip[cat[x]]] // Reverse

Out[38]= subclass[composite[cat[x], RIGHT[APPLY[dom[cat[x]], w]]], Id] == True

In[39]:= subclass[composite[cat[x_], RIGHT[APPLY[dom[cat[x_]], w_]]], Id] := True
```

An analogous result can be derived with **dom** and **cod** interchanged. The derivation is similar to that just given.

Lemma.

```
In[40]:= SubstTest[implies, member[t, ids[cat[x]]],
           subclass[composite[cat[x], RIGHT[t]], Id], t → APPLY[cod[cat[x]], w]] // Reverse

Out[40]= or[not[member[w, range[cat[x]]]],
           subclass[composite[cat[x], RIGHT[APPLY[cod[cat[x]], w]]], Id]] == True

In[41]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[42]:= Map[or[member[w, range[cat[x]]], #] &, SubstTest[implies, empty[t], subclass[t, Id],
           t -> composite[cat[x], RIGHT[APPLY[cod[cat[x]], w]]]] // Reverse

Out[42]= or[member[w, range[cat[x]]],
           subclass[composite[cat[x], RIGHT[APPLY[cod[cat[x]], w]]], Id]] == True

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

Theorem. Combine the two cases.

```
In[44]:= SubstTest[and, implies[p, q], or[p, q], {p -> member[w, range[cat[x]]],
           q -> subclass[composite[cat[x], RIGHT[APPLY[cod[cat[x]], w]]], Id]}]

Out[44]= subclass[composite[cat[x], RIGHT[APPLY[cod[cat[x]], w]]], Id] == True

In[45]:= subclass[composite[cat[x_], RIGHT[APPLY[cod[cat[x_]], w_]]], Id] := True
```

Corollary. Dual result.

```
In[46]:= SubstTest[subclass,
           composite[cat[t], RIGHT[APPLY[cod[cat[t]], w]]], Id, t → flip[cat[x]]] // Reverse

Out[46]= subclass[composite[cat[x], LEFT[APPLY[dom[cat[x]], w]]], Id] == True

In[47]:= subclass[composite[cat[x_], LEFT[APPLY[dom[cat[x_]], w_]]], Id] := True
```