a category wrapper

Johan G. F. Belinfante
2008 December 30

```
In[1]:= SetDirectory["l:"]; << goedel.08dec30a;<< tools.m
        :Package Title: goedel.08dec30a         2008 December 30 at 10:10 a.m.
        It is now:  2008 Dec 30 at 11:40
        Loading Simplification Rules
        TOOLS.M                                      Revised 2008 December 26
        weightlimit = 40
```

summary

Basic properties of the wrapper **cat[x]** for categories are derived. This wrapper is defined by a **class**-wrapped membership rule:

```
In[2]:= Begin["Goedel`Private`"];
In[3]:= FirstMatch[class[t_, member[w_, HoldPattern[cat[x_]]]]]
Out[3]= class[t_, member[w_, cat[x_]]] := class[t, and[member[w, x], category[x]]]
```

As an application, it is shown that for any category **x**, there is an opposite category **flip[x]**.

normalization

Theorem. This important normalization rule takes quite a while to derive.

```
In[4]:= cat[x] // Normality // Reverse

Out[4]= intersection[x, complement[image[V, fix[composite[x, inverse[x], Di]]],
          complement[image[V, fix[composite[cross[Id, x], ASSOC,
              complement[composite[cross[inverse[x], Id], inverse[x]]], x]]],
          complement[image[V, fix[composite[cross[x, Id], inverse[ASSOC], complement[
               composite[cross[Id, inverse[x]], inverse[x]]], x]]], complement[image[V,
            fix[composite[inverse[FIRST], domain[x], complement[inverse[domain[x]]], x]]],
          complement[image[V, intersection[x, complement[cart[V, V]]]],
          complement[image[V, intersection[x, complement[cart[cart[V, V], V]]]],
          complement[image[V, intersection[complement[cart[V, range[x]]], domain[x]]],
          complement[image[V, intersection[complement[cart[range[x], V]], domain[x]]],
          complement[image[V, intersection[complement[domain[cod[x]]], range[x]]],
          complement[image[V, intersection[complement[domain[dom[x]]], range[x]]],
          complement[image[V,
            intersection[domain[domain[x]], image[x, complement[cart[V, V]]]]]] == cat[x]

In[5]:= intersection[complement[image[V, fix[composite[x_, inverse[x_], Di]]],
          complement[image[V, fix[composite[cross[Id, x_], ASSOC,
              complement[composite[cross[inverse[x_], Id], inverse[x_]]], x_]]],
          complement[image[V, fix[composite[cross[x_, Id], inverse[ASSOC],
              complement[composite[cross[Id, inverse[x_]], inverse[x_]]], x_]]],
          complement[image[V, fix[composite[inverse[FIRST], domain[x_],
              complement[inverse[domain[x_]]], x_]]],
          complement[image[V, intersection[complement[cart[V, V]], x_]]],
          complement[image[V, intersection[complement[cart[V, range[x_]]], domain[x_]]]],
          complement[image[V, intersection[complement[cart[cart[V, V], V]], x_]]],
          complement[image[V, intersection[complement[cart[range[x_], V]], domain[x_]]]],
          complement[image[V, intersection[complement[domain[cod[x_]]], range[x_]]]],
          complement[image[V, intersection[complement[domain[dom[x_]]], range[x_]]]],
          complement[image[V,
            intersection[domain[domain[x_]], image[x_, complement[cart[V, V]]]]], x_] := cat[x]
```

---

a temporary definition

The following temporary definition will be used repeatedly to derive the properties of the **cat[x]** wrapper.

```
In[6]:= temp[x_] := union[fix[composite[x, inverse[x], Di]], fix[composite[cross[Id, x],
            ASSOC, complement[composite[cross[inverse[x], Id], inverse[x]]], x]],
          fix[composite[cross[x, Id], inverse[ASSOC],
            complement[composite[cross[Id, inverse[x]], inverse[x]]], x]],
          fix[composite[inverse[FIRST], domain[x], complement[inverse[domain[x]]], x]],
          intersection[x, complement[cart[V, V]]],
          intersection[x, complement[cart[cart[V, V], V]]],
          intersection[complement[cart[V, range[x]]], domain[x]],
          intersection[complement[cart[range[x], V]], domain[x]],
          intersection[complement[domain[cod[x]]], range[x]],
          intersection[complement[domain[dom[x]]], range[x]],
          intersection[domain[domain[x]], image[x, complement[cart[V, V]]]]]
```

FUNCTION

Theorem.

```
In[7]:= SubstTest[FUNCTION, intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse
Out[7]= FUNCTION[cat[x]] == True

In[8]:= FUNCTION[cat[x_]] := True
```

associative

Theorem.

```
In[9]:= SubstTest[associative, intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse
Out[9]= associative[cat[x]] == True

In[10]:= associative[cat[x_]] := True
```

Corollary.

```
In[11]:= SubstTest[composite, assoc[t], id[cart[V, V]], t → cat[x]] // Reverse
Out[11]= composite[cat[x], id[cart[V, V]]] == cat[x]

In[12]:= composite[cat[x_], id[cart[V, V]]] := cat[x]
```

Corollary.

```
In[13]:= ImageComp[cat[x], id[cart[V, V]], V] // Reverse
Out[13]= image[cat[x], cart[V, V]] == range[cat[x]]

In[14]:= image[cat[x_], cart[V, V]] := range[cat[x]]
```

Corollary.

```
In[15]:= SubstTest[composite, assoc[t], cross[Id, assoc[t]], ASSOC, t → cat[x]] // Reverse
Out[15]= composite[cat[x], cross[Id, cat[x]], ASSOC] == composite[cat[x], cross[cat[x], Id]]

In[16]:= composite[cat[x_], cross[Id, cat[x_]], ASSOC] := composite[cat[x], cross[cat[x], Id]]
```

Corollary.

```
In[17]:= SubstTest[composite, assoc[t],
           cross[assoc[t], Id], inverse[ASSOC], t → cat[x]] // Reverse

Out[17]= composite[cat[x], cross[cat[x], Id], inverse[ASSOC]] ==
          composite[cat[x], cross[Id, cat[x]]]

In[18]:= composite[cat[x_], cross[cat[x_], Id], inverse[ASSOC]] :=
          composite[cat[x], cross[Id, cat[x]]]
```

domain property

Temporary definition.

```
In[19]:= domainproperty[x_] := subclass[domain[x], cart[range[x], range[x]]]
```

Lemma.

```
In[20]:= subclass[x, cart[y, intersection[complement[image[V, t]], z]]] // AssertTest

Out[20]= subclass[x, cart[y, intersection[z, complement[image[V, t]]]]] ==
          or[and[equal[0, t], subclass[x, cart[y, z]]], equal[0, x]]

In[21]:= subclass[x_, cart[y_, intersection[z_, complement[image[V, t_]]]]] :=
          or[and[equal[0, t], subclass[x, cart[y, z]]], equal[0, x]]
```

Theorem.

```
In[22]:= SubstTest[domainproperty,
           intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse

Out[22]= subclass[domain[cat[x]], cart[range[cat[x]], range[cat[x]]]] == True

In[23]:= subclass[domain[cat[x_]], cart[range[cat[x_]], range[cat[x_]]]] := True
```

Corollary. (This will be improved upon later.)

```
In[24]:= SubstTest[implies, subclass[u, cartsq[v]],
           subclass[range[u], v], {u → domain[cat[x]], v → range[cat[x]]}] // Reverse

Out[24]= subclass[range[domain[cat[x]]], range[cat[x]]] == True

In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary. (This will also be improved upon later.)

```
In[26]:= SubstTest[implies, subclass[u, cartsq[v]],
           subclass[domain[u], v], {u → domain[cat[x]], v → range[cat[x]]}] // Reverse

Out[26]= subclass[domain[domain[cat[x]]], range[cat[x]]] == True

In[27]:= (% /. x → x_) /. Equal → SetDelayed
```

axiom 2

Temporary definition.

```
In[28]:= ax2[x_] := subclass[composite[x, inverse[FIRST], domain[x]], domain[x]]
```

Lemma.

```
In[29]:= ax2[intersection[x, complement[image[V, t]]]] // AssertTest
```

```
Out[29]= subclass[composite[intersection[x, complement[image[V, t]]], inverse[FIRST],
           intersection[complement[image[V, t]], domain[x]]], domain[x]] ==
         or[not[equal[0, t]], subclass[composite[x, inverse[FIRST], domain[x]], domain[x]]]
```

```
In[30]:= subclass[composite[intersection[x_, complement[image[V, t_]]], inverse[FIRST],
           intersection[complement[image[V, t_]], domain[x_]]], domain[x_]] :=
         or[not[equal[0, t]], subclass[composite[x, inverse[FIRST], domain[x]], domain[x]]]
```

Theorem. Axiom 2 is satsified by **cat[x]**.

```
In[31]:= SubstTest[ax2, intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse
```

```
Out[31]= subclass[composite[cat[x], inverse[FIRST], domain[cat[x]]], domain[cat[x]]] == True
```

```
In[32]:= subclass[composite[cat[x_], inverse[FIRST], domain[cat[x_]]], domain[cat[x_]]] := True
```

axiom 3

```
In[33]:= dom[intersection[x, complement[image[V, t]]]] // Normality
```

```
Out[33]= dom[intersection[x, complement[image[V, t]]]] ==
          intersection[complement[image[V, t]], dom[x]]
```

```
In[34]:= dom[intersection[x_, complement[image[V, t_]]]] :=
          intersection[complement[image[V, t]], dom[x]]
```

Similarly:

```
In[35]:= cod[intersection[x, complement[image[V, t]]]] // Normality
```

```
Out[35]= cod[intersection[x, complement[image[V, t]]]] ==
          intersection[cod[x], complement[image[V, t]]]
```

```
In[36]:= cod[intersection[x_, complement[image[V, t_]]]] :=
          intersection[cod[x], complement[image[V, t]]]
```

Temporary definition.

```
In[37]:= ax3a[t_] := subclass[range[t], domain[cod[t]]]
```

Lemma. (This will later be improved upon.)

```
In[38]:= SubstTest[ax3a, intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse
Out[38]= subclass[range[cat[x]], domain[cod[cat[x]]]] == True

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

Temporary definition.

```
In[40]:= ax3b[t_] := subclass[range[t], domain[dom[t]]]
```

Theorem.

```
In[41]:= SubstTest[ax3b, intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse
Out[41]= subclass[range[cat[x]], domain[dom[cat[x]]]] == True

In[42]:= (% /. x → x_) /. Equal → SetDelayed
```

---

wrapper introduction rule

Theorem.

```
In[43]:= category[cat[x]] // AssertTest
Out[43]= category[cat[x]] == True

In[44]:= category[cat[x_]] := True
```

---

wrapper removal rule

Lemma.

```
In[45]:= equiv[or[category[x], equal[0, x]], category[x]]
Out[45]= True

In[46]:= or[category[x_], equal[0, x_]] := category[x]
```

Theorem.

```
In[47]:= SubstTest[equal, x, intersection[x, complement[image[V, t]]], t → temp[x]] // Reverse
Out[47]= equal[x, cat[x]] == category[x]

In[48]:= equal[x_, cat[x_]] := category[x]
```

A conditional rule is often useful:

```
In[49]:= cat[x_] := x /; category[x]
```

range[domain[cat[x]]]

Lemma.

```
In[50]:= SubstTest[subclass, image[u, v], range[u], {u → domain[x], v → ids[x]}] // Reverse
Out[50]= subclass[domain[cod[x]], range[domain[x]]] == True

In[51]:= subclass[domain[cod[x_]], range[domain[x_]]] := True
```

Lemma.

```
In[52]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → range[cat[x]], v → domain[cod[cat[x]]], w → range[domain[cat[x]]]}] // Reverse
Out[52]= subclass[range[cat[x]], range[domain[cat[x]]]] == True

In[53]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[54]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u → range[domain[cat[x]]], v → range[cat[x]]}]
Out[54]= equal[range[cat[x]], range[domain[cat[x]]]] == True

In[55]:= range[domain[cat[x_]]] := range[cat[x]]
```

domain[domain[cat[x]]]

Lemma.

```
In[56]:= SubstTest[subclass, image[u, v],
           range[u], {u → inverse[domain[x]], v → ids[x]}] // Reverse
Out[56]= subclass[domain[dom[x]], domain[domain[x]]] == True

In[57]:= subclass[domain[dom[x_]], domain[domain[x_]]] := True
```

Lemma.

```
In[58]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → range[cat[x]], v → domain[dom[cat[x]]], w → domain[domain[cat[x]]]}] // Reverse
Out[58]= subclass[range[cat[x]], domain[domain[cat[x]]]] == True

In[59]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[60]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → domain[domain[cat[x]]], v → range[cat[x]]}]

Out[60]= equal[domain[domain[cat[x]]], range[cat[x]]] == True

In[61]:= domain[domain[cat[x_]]] := range[cat[x]]
```

sharper form of axiom 3

Lemma.

```
In[62]:= SubstTest[subclass, domain[cod[t]], range[domain[t]], t → cat[x]] // Reverse

Out[62]= subclass[domain[cod[cat[x]]], range[cat[x]]] == True

In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (Sharper form for axiom 3.)

```
In[64]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → domain[cod[cat[x]]], v → range[cat[x]]}]

Out[64]= equal[domain[cod[cat[x]]], range[cat[x]]] == True

In[65]:= domain[cod[cat[x_]]] := range[cat[x]]
```

Lemma,

```
In[66]:= SubstTest[subclass, domain[dom[t]], domain[domain[t]], t → cat[x]] // Reverse

Out[66]= subclass[domain[dom[cat[x]]], range[cat[x]]] == True

In[67]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[68]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → domain[dom[cat[x]]], v → range[cat[x]]}]

Out[68]= equal[domain[dom[cat[x]]], range[cat[x]]] == True

In[69]:= domain[dom[cat[x_]]] := range[cat[x]]
```

flipped axiom 2

```
In[70]:= SubstTest[implies, category[t],
           subclass[composite[domain[t], SECOND, inverse[t]], domain[t]], t → cat[x]] // Reverse

Out[70]= subclass[composite[domain[cat[x]], SECOND, inverse[cat[x]]], domain[cat[x]]] == True

In[71]:= subclass[composite[domain[cat[x_]], SECOND, inverse[cat[x_]]], domain[cat[x_]]] := True
```

opposite category

Theorem.

```
In[72]:= category[composite[cat[x], SWAP]] // AssertTest

Out[72]= category[composite[cat[x], SWAP]] == True

In[73]:= category[composite[cat[x_], SWAP]] := True
```

Corollary.

```
In[74]:= SubstTest[implies, equal[x, cat[t]], category[flip[x]], t → x] // Reverse

Out[74]= or[category[composite[x, SWAP]], not[category[x]]] == True

In[75]:= or[category[composite[x_, SWAP]], not[category[x_]]] := True
```