a functor equation

Johan G. F. Belinfante
2011 December 30

```
In[1]:= SetDirectory["l:"]; << goedel.11dec29a
```

```
:Package Title: goedel.11dec29a              2011 December 29 at 9:30 a.m.

Loading takes about thirteen minutes, half that time due to builtin pauses.

It is now:  2011 Dec 30 at 8:48

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Dec 30 at 9:1
```

summary

If **w** is a functor from **x** to **y**, then **w ∘ x ⊂ y ∘ (w ⊗ w)**. When **y** is a category, this is an inclusion of functions, and is equivalent to the equation **w ∘ x = y ∘ (w ⊗ w) ∘ id[domain[x]]**.

derivation

Lemma.

```
In[2]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]], {u → composite[funpart[w], x],
           v → composite[cat[y], cross[funpart[w], funpart[w]]]}] // Reverse
```

```
Out[2]= or[equal[composite[funpart[w], x], composite[cat[y], cross[funpart[w], funpart[w]],
            id[image[inverse[x], domain[funpart[w]]]]]], not[subclass[
            composite[funpart[w], x], composite[cat[y], cross[funpart[w], funpart[w]]]]]] == True
```

```
In[3]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[13]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p1, p2], p3], implies[p1, p4], (*implies[and[p3,p4],p5],*)
            not[implies[p1, p5]], {p1 → functor[funpart[w], x, cat[y]], p2 → subclass[
               composite[funpart[w], x], composite[cat[y], cross[funpart[w], funpart[w]]]],
             p3 → equal[composite[funpart[w], x], composite[cat[y],
                cross[funpart[w], funpart[w]], id[image[inverse[x], domain[funpart[w]]]]]],
             p4 → equal[domain[funpart[w]], range[x]], p5 → equal[composite[funpart[w], x],
               composite[cat[y], cross[funpart[w], funpart[w]], id[domain[x]]]]}]] // Reverse

Out[13]= or[equal[composite[funpart[w], x],
            composite[cat[y], cross[funpart[w], funpart[w]], id[domain[x]]]],
           not[functor[funpart[w], x, cat[y]]]] == True

In[14]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[17]:= SubstTest[implies, equal[w, funpart[t]],
           or[equal[composite[w, x], composite[cat[y], cross[w, w], id[domain[x]]]],
            not[functor[w, x, cat[y]]]], t → w] // Reverse

Out[17]= or[equal[composite[w, x], composite[cat[y], cross[w, w], id[domain[x]]]],
           not[FUNCTION[w]], not[functor[w, x, cat[y]]]] == True

In[18]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. An equation for functors to categories.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
            not[implies[p1, p3]], {p1 → functor[w, x, cat[y]], p2 → FUNCTION[w], p3 ->
              equal[composite[w, x], composite[cat[y], cross[w, w], id[domain[x]]]]}]] // Reverse

Out[19]= or[equal[composite[w, x], composite[cat[y], cross[w, w], id[domain[x]]]],
           not[functor[w, x, cat[y]]]] == True

In[21]:= or[equal[composite[w_, x_], composite[cat[y_], cross[w_, w_], id[domain[x_]]]],
           not[functor[w_, x_, cat[y_]]]] := True
```

Corollary. (Eliminating the **cat** wrapper.)

```
In[22]:= SubstTest[implies, equal[y, cat[t]], implies[functor[w, x, y],
            equal[composite[w, x], composite[y, cross[w, w], id[domain[x]]]]], t → y] // Reverse

Out[22]= or[equal[composite[w, x], composite[y, cross[w, w], id[domain[x]]]],
           not[category[y]], not[functor[w, x, y]]] == True

In[23]:= or[equal[composite[w_, x_], composite[y_, cross[w_, w_], id[domain[x_]]]],
           not[category[y_]], not[functor[w_, x_, y_]]] := True
```

A corollary involving ranges can be derived.

Lemma.

```
In[32]:= SubstTest[implies, equal[u, v], equal[range[u], range[v]],
           {u -> composite[w, x], v -> composite[y, cross[w, w], id[domain[x]]]}] // Reverse

Out[32]= or[equal[image[w, range[x]], image[y, composite[w, domain[x], inverse[w]]]],
           not[equal[composite[w, x], composite[y, cross[w, w], id[domain[x]]]]] == True

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

Theorem. An equation involving the range of a functor to a category.

```
In[37]:= Map[not, SubstTest[and, (*implies[and[p1,p2],p3],*)
            implies[p3, p4], implies[p1, p5], (*implies[and[p4,p5],p6],*)
            not[implies[and[p1, p2], p6]], {p1 → functor[w, x, y], p2 → category[y],
             p3 -> equal[composite[w, x], composite[y, cross[w, w], id[domain[x]]]],
             p4 -> equal[image[w, range[x]], image[y, composite[w, domain[x], inverse[w]]]],
             p5 → equal[domain[w], range[x]],
             p6 -> equal[image[y, composite[w, domain[x], inverse[w]]], range[w]]}]] // Reverse

Out[37]= or[equal[image[y, composite[w, domain[x], inverse[w]]], range[w]],
           not[category[y]], not[functor[w, x, y]]] == True

In[39]:= or[equal[image[y_, composite[w_, domain[x_], inverse[w_]]], range[w_]],
           not[category[y_]], not[functor[w_, x_, y_]]] := True
```

Theorem. If **w** is a functor from **x** to a category **y** and if **domain[x] = range[x] × range[x]**, then **range[w]** is binary closed under **y**.

```
In[47]:= Map[not, SubstTest[and, (*implies[and[p1,p3],p4],*) implies[and[p2, p4, p5], p6],
            not[implies[and[p1, p2, p3], p6]], {p1 → functor[w, x, y],
             p2 → equal[domain[x], cart[range[x], range[x]]], p3 → category[y],
             p4 -> equal[image[y, composite[w, domain[x], inverse[w]]], range[w]],
             p5 → equal[domain[w], range[x]],
             p6 -> equal[image[y, cart[range[w], range[w]]], range[w]]}]] // Reverse

Out[47]= or[equal[image[y, cart[range[w], range[w]]], range[w]], not[category[y]],
           not[equal[cart[range[x], range[x]], domain[x]]], not[functor[w, x, y]]] == True

In[49]:= or[equal[image[y_, cart[range[w_], range[w_]]], range[w_]], not[category[y_]],
           not[equal[cart[range[x_], range[x_]], domain[x_]]], not[functor[w_, x_, y_]]] := True
```