category of sets (functions)

Johan G. F. Belinfante
2005 August 1

```
In[1]:= SetDirectory["i:"]; << goedel71.31a; << tools.m

        :Package Title: goedel71.31a          2005 July 31 at 12:20 midnite

        It is now:  2005 Aug 1 at 12:57

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 July 31

        weightlimit = 40
```

summary

The morphisms in the category of sets are ordered pairs **PAIR[x,y]**, where **x** is a (small) function and **y** is a set (called the codomain of the morphism) which contains the range of **x**. The function **CATOFUNS** is the (partial) binary composition for morphisms of the category of sets. Some rules about this function are derived.

FUNCTION and domain rules

Since **CATOFUNS** is a restriction of **CATORELN**, it follows that it is a function.

```
In[8]:= SubstTest[FUNCTION, composite[funpart[x], id[y]],
          {x → CATORELN, y → cart[cart[FUNS, V], cart[FUNS, V]]}]

Out[8]= FUNCTION[CATOFUNS] == True

In[9]:= FUNCTION[CATOFUNS] := True
```

A similar argument yields a formula for the domain:

```
In[10]:= IminComp[CATORELN, id[cart[cart[FUNS, V], cart[FUNS, V]]], V]

Out[10]= domain[CATOFUNS] ==
          composite[id[composite[S, IMAGE[SECOND], id[FUNS]]], inverse[SECOND],
           IMAGE[FIRST], FIRST, id[composite[S, IMAGE[SECOND], id[FUNS]]]]
```

```
In[11]:= domain[CATOFUNS] :=
          composite[id[composite[S, IMAGE[SECOND], id[FUNS]]], inverse[SECOND],
           IMAGE[FIRST], FIRST, id[composite[S, IMAGE[SECOND], id[FUNS]]]]
```

---

an abbreviation

The following abbreviation is convenient.

```
In[3]:= direct[x_, y_] := composite[cross[x, y], TWIST]
```

The connection with **CATOFUNS** is:

```
In[26]:= equal[CATOFUNS, composite[direct[COMPOSE, FIRST], id[domain[CATOFUNS]]]]

Out[26]= True
```

---

range

```
In[12]:= Map[inverse,
          composite[FIRST, id[composite[IMAGE[DUP], IMAGE[FIRST], id[FUNS]]],
            inverse[COMPOSE]] // inverse // VSNormality]

Out[12]= composite[FIRST, id[composite[IMAGE[DUP], IMAGE[FIRST], id[FUNS]]],
           inverse[COMPOSE]] == id[FUNS]

In[13]:= composite[FIRST, id[composite[IMAGE[DUP], IMAGE[FIRST], id[FUNS]]],
           inverse[COMPOSE]] := id[FUNS]

In[14]:= Map[subclass[range[#], range[CATOFUNS]] &,
          Assoc[direct[COMPOSE, FIRST], id[domain[CATOFUNS]],
           cross[Id, composite[id[inverse[IMAGE[DUP]]], inverse[SECOND]]]]]

Out[14]= subclass[composite[S, IMAGE[SECOND], id[FUNS]], range[CATOFUNS]] == True

In[15]:= % /. Equal → SetDelayed

In[16]:= Map[implies[subclass[#, range[CATORELN]],
            subclass[range[CATOFUNS], composite[S, IMAGE[SECOND]]]] &,
           ImageComp[CATORELN, id[cart[cart[FUNS, V], cart[FUNS, V]]], V]] // Reverse

Out[16]= subclass[range[CATOFUNS], composite[S, IMAGE[SECOND]]] == True

In[17]:= % /. Equal → SetDelayed
```

```
In[18]:= Map[subclass[#, cart[FUNS, V]] &,
          ImageComp[direct[COMPOSE, FIRST], id[domain[CATOFUNS]], V]]

Out[18]= subclass[range[CATOFUNS], cart[FUNS, V]] == True

In[19]:= % /. Equal → SetDelayed

In[20]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → range[CATOFUNS], v → domain[domain[CATOFUNS]]}]

Out[20]= True == equal[composite[S, IMAGE[SECOND], id[FUNS]], range[CATOFUNS]]

In[21]:= range[CATOFUNS] := composite[S, IMAGE[SECOND], id[FUNS]]
```

---

endomorphisms

```
In[22]:= equal[intersection[FUNS, U[image[MAP, Id]]], U[image[MAP, Id]]]

Out[22]= True

In[23]:= intersection[FUNS, U[image[MAP, Id]]] := U[image[MAP, Id]]

In[24]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
          {u → fix[domain[CATOFUNS]], v → composite[IMAGE[FIRST], id[FUNS]]}]

Out[24]= equal[composite[IMAGE[FIRST], id[U[image[MAP, Id]]]], composite[
           intersection[composite[S, IMAGE[SECOND]], IMAGE[FIRST]], id[FUNS]]] == True

In[25]:= composite[intersection[composite[S, IMAGE[SECOND]], IMAGE[FIRST]],
           id[FUNS]] := composite[IMAGE[FIRST], id[U[image[MAP, Id]]]]
```

---

a consequence of associativity

The associative property of the (partial) binary composition for morphisms of the category of relations implies:

```
In[4]:= SubstTest[implies, associative[x], equal[
          composite[x, cross[Id, x], ASSOC], composite[x, cross[x, Id]]], x → CATORELN]

Out[4]= equal[composite[CATORELN, cross[CATORELN, Id]],
          composite[CATORELN, cross[Id, CATORELN], ASSOC]] == True

In[5]:= composite[CATORELN, cross[Id, CATORELN], ASSOC] :=
         composite[CATORELN, cross[CATORELN, Id]]
```

A similar rule holds in the category of sets.

```
In[6]:= SubstTest[implies, associative[x], equal[
          composite[x, cross[Id, x], ASSOC], composite[x, cross[x, Id]]], x → CATOFUNS]

Out[6]= equal[composite[CATOFUNS, cross[CATOFUNS, Id]],
          composite[CATOFUNS, cross[Id, CATOFUNS], ASSOC]] == True

In[7]:= composite[CATOFUNS, cross[Id, CATOFUNS], ASSOC] :=
         composite[CATOFUNS, cross[CATOFUNS, Id]]
```

---

membership rules

Lemma.

```
In[66]:= equiv[and[equal[z, composite[x, y]], member[x, V], member[y, V], member[z, V]],
          and[equal[z, composite[x, y]], member[x, V], member[y, V]]]

Out[66]= True

In[68]:= and[equal[z_, composite[x_, y_]], member[x_, V], member[y_, V],
           member[z_, V]] := and[equal[z, composite[x, y]], member[x, V], member[y, V]]
```

Lemma.

```
In[48]:= member[pair[pair[pair[u, v], pair[x, y]], pair[w, z]],
           composite[t, TWIST]] // AssertTest

Out[48]= member[pair[pair[pair[u, v], pair[x, y]], pair[w, z]], composite[t, TWIST]] ==
          and[member[u, V], member[v, V], member[x, V], member[y, V],
           member[pair[pair[pair[u, x], pair[v, y]], pair[w, z]], t]]

In[49]:= member[pair[pair[pair[u_, v_], pair[x_, y_]], pair[w_, z_]],
           composite[t_, TWIST]] := and[member[u, V], member[v, V], member[x, V],
           member[y, V], member[pair[pair[pair[u, x], pair[v, y]], pair[w, z]], t]]
```

Membership rule for **CATORELN**.

```
In[71]:= SubstTest[member, pair[pair[pair[u, v], pair[x, y]], pair[w, z]],
          composite[direct[COMPOSE, FIRST], id[t]], t → domain[CATORELN]]

Out[71]= member[pair[pair[pair[u, v], pair[x, y]], pair[w, z]], CATORELN] ==
          and[equal[v, z], equal[w, composite[u, x]], equal[y, domain[u]], member[u, V],
           member[v, V], member[x, V], member[y, V], subclass[u, cart[V, V]],
           subclass[x, cart[V, V]], subclass[range[u], v], subclass[range[x], y]]

In[72]:= member[pair[pair[pair[u_, v_], pair[x_, y_]], pair[w_, z_]], CATORELN] :=
          and[equal[v, z], equal[w, composite[u, x]], equal[y, domain[u]], member[u, V],
           member[v, V], member[x, V], member[y, V], subclass[u, cart[V, V]],
           subclass[x, cart[V, V]], subclass[range[u], v], subclass[range[x], y]]
```

Membership rule for **CATOFUNS**.

```
In[73]:= SubstTest[member, pair[pair[pair[u, v], pair[x, y]], pair[w, z]],
          composite[direct[COMPOSE, FIRST], id[t]], t → domain[CATOFUNS]]

Out[73]= member[pair[pair[pair[u, v], pair[x, y]], pair[w, z]], CATOFUNS] ==
          and[equal[v, z], equal[w, composite[u, x]], equal[y, domain[u]],
           FUNCTION[u], FUNCTION[x], member[u, V], member[v, V], member[x, V],
           member[y, V], subclass[range[u], v], subclass[range[x], y]]

In[74]:= member[pair[pair[pair[u_, v_], pair[x_, y_]], pair[w_, z_]], CATOFUNS] :=
          and[equal[v, z], equal[w, composite[u, x]], equal[y, domain[u]],
           FUNCTION[u], FUNCTION[x], member[u, V], member[v, V], member[x, V],
           member[y, V], subclass[range[u], v], subclass[range[x], y]]
```

---

normalization

The membership rules in the preceding section are too special to have any effect on the usual **Normality** tests. The following substitutes are provided.

```
In[81]:= class[pair[pair[pair[u, v], pair[x, y]], pair[w, z]],
          and[equal[v, z], equal[w, composite[u, x]], equal[y, domain[u]],
           FUNCTION[u], FUNCTION[x], member[u, V], member[v, V], member[x, V],
           member[y, V], subclass[range[u], v], subclass[range[x], y]]]

Out[81]= CATOFUNS

In[82]:= class[pair[pair[pair[u, v], pair[x, y]], pair[w, z]],
          and[equal[v, z], equal[w, composite[u, x]], equal[y, domain[u]], member[u, V],
           member[v, V], member[x, V], member[y, V], subclass[u, cart[V, V]],
           subclass[x, cart[V, V]], subclass[range[u], v], subclass[range[x], y]]]

Out[82]= CATORELN
```