hom[x]

Johan G. F. Belinfante
2009 January 21

```
In[1]:= SetDirectory["l:"]; << goedel.09jan20a;<< tools.m

        :Package Title: goedel.09jan20a        2009 January 20 at 5:25 p.m.

        It is now:  2009 Jan 21 at 22:33

        Loading Simplification Rules

        TOOLS.M                                        Revised 2009 January 20

        weightlimit = 40
```

summary

When **u** and **v** are identity morphisms of an (abstract arrows-only) category **x**, the class of morphisms from **u** to **v**, usually denoted by $\mathbf{hom_x[u, v]}$, consists of those morphisms **w** ∈ **range[x]** such that **u** is the dom of **w** and **v** is the cod of **w**. In other words, this is the class of morphisms **w** such that **u** is the unique identity morphism for which the composite **w** · **u** is defined and **v** is the unique identity morphism for which the composite **v** · **w** is defined. The commonly-used notation $\mathbf{hom_x[u, v]}$ will not be adopted in the **GOEDEL** program. Instead, a ternary relation **hom[x]** depending only on the category **x** is introduced, such that the statement that **w** is a morphism from **u** to **v** in the category **x** is rendered as **pair[pair[u, v], w]** ∈ **x**. Introducing such a ternary relation allows one to eliminate the variables **u**, **v** and **w** from many definitions and theorems. For example, the definition of a **locally small** category, that is, a category with the property that the class of morphisms from **u** to **v** is a set for all **u**, **v** can be succinctly characterized as a category for which the ternary relation **hom[x]** is thin. (Recall that a relation is **thin** if all its vertical sections are sets.)

This notebook is limited to deriving basic properties of the ternary relation **hom[x]**, as well as providing some examples. Although the introduction of the ternary relation **hom[x]** is primarily motivated by applications to category theory, its definition in the **GOEDEL** program makes sense for any class **x**.

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= ?? hom

        the ternary relation hom[x] holds pair[pair[
          u,v],w] if dom[x] holds pair[w,u]  and cod[x] holds pair[w,v]
```

The following **class**-wrapped membership rule serves to define **hom[x]**:

```
In[4]:= FirstMatch[class[t_, member[w_, HoldPattern[hom[x_]]]]]

Out[4]= class[t_, member[w_, hom[x_]]] := ReleaseHold[class[t, and[member[first[first[w]],
          ids[x]]], member[pair[second[w], first[first[w]]], domain[x]], member[pair[
          second[first[w]], second[w]], domain[x]], member[second[first[w]], ids[x]]]]]]
```

normalization

The main tool for deriving the properties of **hom[x]** is the normalization rule derived in this section. The derivation of the normalization rule itself is the most time-consuming task in this notebook. The execution time is reduced from 64 seconds to 37 seconds by clearing the **simplify** flag, and an additional 5 seconds is shaved off by clearing the **cond** flag as well. (These execution times are for a computer with a 3.4 GHz Pentium 4 Processor.)

```
In[5]:= simplify = False; cond = False;
```

Lemma. Normalization rewrite rule.

```
In[6]:= hom[x] // Normality // Reverse

Out[6]= composite[intersection[composite[domain[x], SECOND],
          composite[inverse[domain[x]], FIRST]], id[cart[ids[x], ids[x]]]] == hom[x]

In[7]:= composite[intersection[composite[domain[x_], SECOND],
          composite[inverse[domain[x_]], FIRST]], id[cart[ids[x_], ids[x_]]]] := hom[x]
```

Theorem. This simpler variant of the normalization rule suffices for most applications.

```
In[8]:= Assoc[inverse[DUP], cross[inverse[domain[x]], domain[x]], id[cartsq[ids[x]]]]

Out[8]= intersection[composite[inverse[cod[x]], SECOND],
          composite[inverse[dom[x]], FIRST]] == hom[x]

In[9]:= intersection[composite[inverse[cod[x_]], SECOND],
          composite[inverse[dom[x_]], FIRST]] := hom[x]
```

inverse

When **x** is a category, **inverse[hom[x]]** is a function.

```
In[10]:= Map[FUNCTION, composite[cross[dom[t], cod[t]], DUP] // DoubleInverse] /. t → cat[x] //
          Reverse

Out[10]= FUNCTION[inverse[hom[cat[x]]]] == True

In[11]:= FUNCTION[inverse[hom[cat[x_]]]] := True
```

Theorem. The following normalization rule holds for the inverse:

```
In[12]:= composite[cross[dom[x], cod[x]], DUP] // DoubleInverse

Out[12]= intersection[composite[inverse[FIRST], dom[x]],
          composite[inverse[SECOND], cod[x]]] == inverse[hom[x]]
```

```
In[13]:= intersection[composite[inverse[FIRST], dom[x_]],
           composite[inverse[SECOND], cod[x_]]] := inverse[hom[x]]
```

trivial examples

Trivial cases sometimes suffice to provide counter-examples.

```
In[14]:= hom[0] // Normality
```

```
Out[14]= hom[0] == 0
```

```
In[15]:= hom[0] := 0
```

Lemma.

```
In[16]:= hom[V] // Normality
```

```
Out[16]= hom[V] == 0
```

```
In[17]:= hom[V] := 0
```

For the remainder of this notebook, the **simplify** and **cond** flags can be restored.

```
In[18]:= simplify = True; cond = True;
```

an example in arithmetic

Theorem.

```
In[19]:= SubstTest[intersection, composite[inverse[cod[t]], SECOND],
           composite[inverse[dom[t]], FIRST], t → NATADD]
```

```
Out[19]= hom[NATADD] == cart[cart[set[0], set[0]], omega]
```

```
In[20]:= hom[NATADD] := cart[cart[set[0], set[0]], omega]
```

discrete categories

A category is said to be **discrete** if every morphism is an identity. It will be shown in this section that **hom[x] = x** for any discrete category **x**. Note that the simplest case **x = 0** has already been encountered above: **hom[0] = 0**.)

Theorem.

```
In[21]:= SubstTest[intersection, composite[inverse[cod[t]], SECOND],
          composite[inverse[dom[t]], FIRST], t → composite[id[x], inverse[DUP]]]
```

```
Out[21]= hom[composite[id[x], inverse[DUP]]] == composite[id[x], inverse[DUP]]
```

```
In[22]:= hom[composite[id[x_], inverse[DUP]]] := composite[id[x], inverse[DUP]]
```

In particular:

```
In[23]:= SubstTest[hom, composite[id[x], inverse[DUP]], x → V] // Reverse
```

```
Out[23]= hom[inverse[DUP]] == inverse[DUP]
```

```
In[24]:= hom[inverse[DUP]] := inverse[DUP]
```

Theorem. If **x** is a discrete category, then **hom[x] = x**.

```
In[25]:= Map[not, SubstTest[and, implies[p1, p2],
            not[implies[p1, p3]], {p1 → and[category[x], equal[range[x], ids[x]]],
             p2 -> equal[x, composite[id[ids[x]], inverse[DUP]]],
             p3 → equal[hom[x], x]}]] // Reverse
```

```
Out[25]= or[equal[x, hom[x]], not[category[x]], not[equal[ids[x], range[x]]]] == True
```

```
In[26]:= or[equal[x_, hom[x_]], not[category[x_]], not[equal[ids[x_], range[x_]]]] := True
```

---

domain

Theorem. Formula for the domain of **hom[x]**.

```
In[27]:= SubstTest[domain, intersection[composite[inverse[u], SECOND],
           composite[inverse[v], FIRST]], {u → cod[x], v → dom[x]}]
```

```
Out[27]= composite[cod[x], inverse[dom[x]]] == domain[hom[x]]
```

```
In[28]:= composite[cod[x_], inverse[dom[x_]]] := domain[hom[x]]
```

Corollary. The domain of **hom[x]** is a binary relation.

```
In[29]:= Assoc[Id, cod[x], inverse[dom[x]]]
```

```
Out[29]= composite[Id, domain[hom[x]]] == domain[hom[x]]
```

```
In[30]:= composite[Id, domain[hom[x_]]] := domain[hom[x]]
```

An ordered pair of identities **u** and **v** belongs to the relation **domain[hom[x]]** if and only if there exists a morphism from **u** to **v**.

```
In[31]:= class[pair[u, v], exists[w, member[pair[pair[u, v], w], t]]] /. t → hom[x]
```

```
Out[31]= domain[hom[x]]
```

Corollary. A formula for **inverse[domain[hom[x]]]**.

```
In[32]:= composite[dom[cat[x]], inverse[cod[cat[x]]]] // DoubleInverse

Out[32]= composite[dom[cat[x]], inverse[cod[cat[x]]]] == inverse[domain[hom[cat[x]]]]

In[33]:= composite[dom[cat[x_]], inverse[cod[cat[x_]]]] := inverse[domain[hom[cat[x]]]]
```

Theorem. An upper bound for **domain[hom[x]]**. Comment: A more general rewrite rule will be derived in the next section.

```
In[34]:= SubstTest[subclass, composite[u, inverse[v]],
           cart[range[u], range[v]], {u → cod[x], v → dom[x]}] // Reverse

Out[34]= subclass[domain[hom[x]], cart[ids[x], ids[x]]] == True

In[35]:= (% /. x → x_) /. Equal → SetDelayed
```

---

fix[domain[hom[x]]]

Lemma.

```
In[36]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → ids[x], v → fix[domain[x]], w → fix[composite[domain[x], domain[x]]]}] // Reverse

Out[36]= subclass[ids[x], fix[composite[domain[x], domain[x]]]] == True

In[37]:= subclass[ids[x_], fix[composite[domain[x_], domain[x_]]]] := True
```

Theorem.

```
In[38]:= Map[equal[ids[x], domain[#]] &, Assoc[intersection[composite[domain[x], SECOND],
             composite[inverse[domain[x]], FIRST]], id[cart[ids[x], ids[x]]], DUP]] // Reverse

Out[38]= equal[fix[domain[hom[x]]], ids[x]] == True

In[39]:= fix[domain[hom[x_]]] := ids[x]
```

Corollary. The relation **domain[hom[x]]** is reflexive.

```
In[40]:= SubstTest[subclass, t, cartsq[fix[t]], t → domain[hom[x]]]

Out[40]= REFLEXIVE[domain[hom[x]]] == True

In[41]:= REFLEXIVE[domain[hom[x_]]] := True
```

Corollary.

```
In[42]:= SubstTest[domain, rfx[t], t → domain[hom[x]]] // Reverse

Out[42]= domain[domain[hom[x]]] == ids[x]
```

```
In[43]:= domain[domain[hom[x_]]] := ids[x]
```

Corollary.

```
In[44]:= SubstTest[range, rfx[t], t → domain[hom[x]]] // Reverse
Out[44]= range[domain[hom[x]]] == ids[x]

In[45]:= range[domain[hom[x_]]] := ids[x]
```

Corollary. An improvement of a rewrite rule derived in the preceding section.

```
In[46]:= SubstTest[subclass, rfx[t], cart[y, z], t → domain[hom[x]]] // Reverse
Out[46]= subclass[domain[hom[x]], cart[y, z]] == and[subclass[ids[x], y], subclass[ids[x], z]]

In[47]:= subclass[domain[hom[x_]], cart[y_, z_]] := and[subclass[ids[x], y], subclass[ids[x], z]]
```

range[hom[x]]

Theorem.

```
In[48]:= SubstTest[range, intersection[composite[inverse[u], SECOND],
           composite[inverse[v], FIRST]], {u → cod[x], v → dom[x]}]
Out[48]= intersection[domain[cod[x]], domain[dom[x]]] == range[hom[x]]

In[49]:= intersection[domain[cod[x_]], domain[dom[x_]]] := range[hom[x]]
```

Theorem. A simpler rule for **range[hom[x]]** holds when **x** is a category.

```
In[50]:= SubstTest[intersection, domain[cod[t]], domain[dom[t]], t → cat[x]]
Out[50]= range[hom[cat[x]]] == range[cat[x]]

In[51]:= range[hom[cat[x_]]] := range[cat[x]]
```

Corollary. If **x** is a category, then **range[hom[x]] = range[x]**.

```
In[52]:= SubstTest[implies, equal[x, cat[t]], equal[range[hom[x]], range[x]], t → x] // Reverse
Out[52]= or[equal[range[x], range[hom[x]]], not[category[x]]] == True

In[53]:= or[equal[range[x_], range[hom[x_]]], not[category[x_]]] := True
```

Counterexample. The range of **hom[x]** need not be equal to **range[x]** when **x** is not a category.

```
In[54]:= equal[range[x], range[hom[x]]] /. x → V
Out[54]= False
```

Lemma.

```
In[55]:= Assoc[Id, intersection[composite[domain[x], SECOND],
           composite[inverse[domain[x]], FIRST]], id[cart[ids[x], ids[x]]]]

Out[55]= composite[Id, hom[x]] == hom[x]

In[56]:= composite[Id, hom[x_]] := hom[x]
```

Theorem. An upper bound for **hom[x]**.

```
In[57]:= SubstTest[subclass, composite[Id, t], cart[y, z], t → hom[x]] // Reverse

Out[57]= subclass[hom[x], cart[y, z]] ==
          and[subclass[domain[hom[x]], y], subclass[range[hom[x]], z]]

In[58]:= subclass[hom[x_], cart[y_, z_]] :=
          and[subclass[domain[hom[x]], y], subclass[range[hom[x]], z]]
```

flip rule

Lemma.

```
In[59]:= SubstTest[composite,
           intersection[composite[inverse[y], SECOND], composite[inverse[z], FIRST]],
           id[cart[V, V]], {y → cod[x], z → dom[x]}] // Reverse

Out[59]= composite[hom[x], id[cart[V, V]]] == hom[x]

In[60]:= composite[hom[x_], id[cart[V, V]]] := hom[x]
```

Theorem. The constructors **hom** and **flip** commute.

```
In[61]:= Map[flip, Map[flip, SubstTest[intersection, composite[inverse[cod[t]], SECOND],
            composite[inverse[dom[t]], FIRST], t → flip[x]]]]

Out[61]= hom[composite[x, SWAP]] == composite[hom[x], SWAP]

In[62]:= hom[composite[x_, SWAP]] := composite[hom[x], SWAP]
```

direct products

The domain of the hom of a direct product is equal to the cross product of the domains of their respective homs. The range of the hom of a direct product is equal to the cartesian product of the ranges of their respective homs.

Theorem. A formula for the domain of the hom of a direct product.

```
In[63]:= Map[domain, SubstTest[intersection, composite[inverse[cod[t]], SECOND],
           composite[inverse[dom[t]], FIRST], t → direct[x, y]]]

Out[63]= domain[hom[composite[cross[x, y], TWIST]]] == cross[domain[hom[x]], domain[hom[y]]]
```

```
In[64]:= domain[hom[composite[cross[x_, y_], TWIST]]] := cross[domain[hom[x]], domain[hom[y]]]
```

Theorem. A formula for the range of the hom of a direct product.

```
In[65]:= Map[range, SubstTest[intersection, composite[inverse[cod[t]], SECOND],
           composite[inverse[dom[t]], FIRST], t → direct[x, y]]]
```

```
Out[65]= range[hom[composite[cross[x, y], TWIST]]] == cart[range[hom[x]], range[hom[y]]]
```

```
In[66]:= range[hom[composite[cross[x_, y_], TWIST]]] := cart[range[hom[x]], range[hom[y]]]
```