inv[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 20:47

        Loading Simplification Rules

        TOOLS.M                                        Revised 2009 January 20

        weightlimit = 40
```

summary

This notebook introduces the relation **inv[x]** which will be needed for group theory, and more generally in category theory. The constructor **inv** is defined in the **GOEDEL** program for any class **x**, not just for groups and categories.

```
In[2]:= ?? inv

        inv[x] is the inverse relation for a partial binary composition x
```

The constructor **inv** is defined by the following **class**-wrapped membership rule.

```
In[3]:= Begin["Goedel`Private`"];

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

Out[4]= class[t_, member[w_, inv[x_]]] := class[t, and[member[w, image[
          inverse[x], ids[x]]], member[w, inverse[image[inverse[x], ids[x]]]]]]
```

If **x** is a group, **inv[x]** is the function that takes any element to its inverse. The relation **inv[x]** for a category **x** is the class of all pairs of morphisms **u** and **v** such that the composites **u · v** and **v · u** are both identity morphisms. It is shown in this notebook that when **x** is a category, **inv[x]** is a function.

normalization

Most of the properties of the constructor **inv[x]** will be derived using the following basic normalization rule.

```
In[5]:= inv[x] // Normality // Reverse

Out[5]= intersection[image[inverse[x], ids[x]], inverse[image[inverse[x], ids[x]]]] == inv[x]
```

```
In[6]:= intersection[image[inverse[x_], ids[x_]],
          inverse[image[inverse[x_], ids[x_]]]] := inv[x]
```

general properties

In this section, general properties of **inv[x]** are derived that hold for any class **x**.

Theorem. The class **inv[x]** is symmetric.

```
In[7]:= SubstTest[inverse, intersection[t, inverse[t]],
          t -> image[inverse[x], ids[x]]] // Reverse
```

```
Out[7]= inverse[inv[x]] == inv[x]
```

```
In[8]:= inverse[inv[x_]] := inv[x]
```

Corollary. The class **inv[x]** is a relation.

```
In[9]:= SubstTest[composite, Id, inverse[t], t → inv[x]] // Reverse
```

```
Out[9]= composite[Id, inv[x]] == inv[x]
```

```
In[10]:= composite[Id, inv[x_]] := inv[x]
```

Corollary. The domain and range of **inv[x]** are equal.

```
In[11]:= SubstTest[range, inverse[t], t → inv[x]] // Reverse
```

```
Out[11]= range[inv[x]] == domain[inv[x]]
```

```
In[12]:= range[inv[x_]] := domain[inv[x]]
```

Comment. The class **domain[inv[x]]** for a category holds all invertible morphisms, also known as **isomorphisms**.

Corollary. Cartesian product upper bounds for the relation **inv[x]**. (Note that, in particular, **inv[x]** is contained in the class **cart[V, V]** of all ordered pairs.)

```
In[13]:= SubstTest[subclass, inverse[t], cart[y, z], t → inv[x]] // Reverse
```

```
Out[13]= subclass[inv[x], cart[y, z]] ==
          and[subclass[domain[inv[x]], y], subclass[domain[inv[x]], z]]
```

```
In[14]:= subclass[inv[x_], cart[y_, z_]] :=
          and[subclass[domain[inv[x]], y], subclass[domain[inv[x]], z]]
```

Theorem.

```
In[15]:= SubstTest[subclass, intersection[t, inverse[t]],
           t, t -> image[inverse[x], ids[x]]] // Reverse
```

```
Out[15]= subclass[inv[x], image[inverse[x], ids[x]]] == True
```

```
In[16]:= subclass[inv[x_], image[inverse[x_], ids[x_]]] := True
```

Corollary.

```
In[17]:= SubstTest[subclass, inverse[u], inverse[v],
           {u → inv[x], v → image[inverse[x], ids[x]]}] // Reverse
Out[17]= subclass[inv[x], inverse[image[inverse[x], ids[x]]]] == True

In[18]:= subclass[inv[x_], inverse[image[inverse[x_], ids[x_]]]] := True
```

Corollary. Inverse pairs are composable.

```
In[19]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → inv[x], v -> image[inverse[x], ids[x]], w → domain[x]}] // Reverse
Out[19]= subclass[inv[x], domain[x]] == True

In[20]:= subclass[inv[x_], domain[x_]] := True
```

Theorem. A rewrite rule for the class of fixed points of **inv[x]**.

```
In[21]:= SubstTest[fix, intersection[t, inverse[t]], t -> image[inverse[x], ids[x]]]
Out[21]= fix[image[inverse[x], ids[x]]] == fix[inv[x]]

In[22]:= fix[image[inverse[x_], ids[x_]]] := fix[inv[x]]
```

---

examples

The following named classes are categories:

```
In[23]:= Select[NamedClasses, category]
Out[23]= {0, CATOFUNS, CATORELN, CUP, INTADD, INTMUL, JOIN, NATADD, NATMUL, SYMDIF}
```

Theorem. The empty category has no inverse pairs of morphisms.

```
In[24]:= SubstTest[intersection, image[inverse[x], ids[x]],
           inverse[image[inverse[x], ids[x]]], x → 0]
Out[24]= inv[0] == 0

In[25]:= inv[0] := 0
```

Theorem. For the category **CUP**, only the only invertible morphism is **0**.

```
In[26]:= SubstTest[intersection, image[inverse[x], ids[x]],
           inverse[image[inverse[x], ids[x]]], x → CUP]
Out[26]= inv[CUP] == cart[set[0], set[0]]
```

```
In[27]:= inv[CUP] := cart[set[0], set[0]]
```

Theorem. Every morphism in the category **SYMDIF** is its own inverse.

```
In[28]:= SubstTest[intersection, image[inverse[x], ids[x]],
           inverse[image[inverse[x], ids[x]]], x → SYMDIF]
Out[28]= inv[SYMDIF] == Id

In[29]:= inv[SYMDIF] := Id
```

S.-T. Hu defines a **groupoid** as a category in which every morphism is invertible, that is, for which **domain[inv[x]] = range[x]**.

```
In[30]:= "Sze-Tsen Hu, Elements of Modern Algebra, Holden-Day, Inc., San Francisco, 1965";
```

The category **SYMDIF** is a groupoid. (Note that, like a group, the category **SYMDIF** has only a single identity element, namely **0**. Some would call **SYMDIF** a 'big group' as it fails to be a group only in that **SYMDIF** is a proper class rather than a set.)

```
In[31]:= equal[domain[inv[x]], range[x]] /. x → SYMDIF
Out[31]= True
```

Theorem.

```
In[32]:= SubstTest[intersection, image[inverse[x], ids[x]],
           inverse[image[inverse[x], ids[x]]], x → inverse[DUP]]
Out[32]= inv[inverse[DUP]] == Id

In[33]:= inv[inverse[DUP]] := Id
```

Observation. The discrete category **inverse[DUP]** is a groupoid. (It is not a 'big group' because it has more than one identity morphism.)

```
In[34]:= equal[domain[inv[x]], range[x]] /. x → inverse[DUP]
Out[34]= True
```

Theorem.

```
In[35]:= SubstTest[intersection, image[inverse[x], ids[x]],
           inverse[image[inverse[x], ids[x]]], x → composite[SWAP, RIF]]
Out[35]= inv[composite[SWAP, RIF]] == SWAP

In[36]:= inv[composite[SWAP, RIF]] := SWAP
```

Observation. The category **composite[SWAP,RIF]** is a groupoid. It is sometimes called the **pair groupoid**.

```
In[37]:= equal[domain[inv[x]], range[x]] /. x → composite[SWAP, RIF]
Out[37]= True
```

Theorem. The additive inverse of an integer **n** is its negative **-n**. The binary operation **INTADD** is a group.

```
In[38]:= SubstTest[intersection, image[inverse[x], ids[x]],
          inverse[image[inverse[x], ids[x]]], x → INTADD]
Out[38]= inv[INTADD] == composite[id[Z], INVERSE]

In[39]:= inv[INTADD] := composite[id[Z], INVERSE]
```

Theorem. The only natural number with an additive inverse is **0**. The binary operation **NATADD** is a monoid, but not a group.

```
In[40]:= SubstTest[intersection, image[inverse[x], ids[x]],
          inverse[image[inverse[x], ids[x]]], x → NATADD]
Out[40]= inv[NATADD] == cart[set[0], set[0]]

In[41]:= inv[NATADD] := cart[set[0], set[0]]
```

Theorem. The only natural number with a multiplicative inverse is **1**. The binary operation **NATMUL** is a monoid, but not a group.

```
In[42]:= SubstTest[intersection, image[inverse[x], ids[x]],
          inverse[image[inverse[x], ids[x]]], x → NATMUL]
Out[42]= inv[NATMUL] == cart[set[set[0]], set[set[0]]]

In[43]:= inv[NATMUL] := cart[set[set[0]], set[set[0]]]
```

flip and direct product rules

Theorem. Flip rule.

```
In[44]:= SubstTest[intersection, image[inverse[t], ids[t]],
          inverse[image[inverse[t], ids[t]]], t → flip[x]]
Out[44]= inv[composite[x, SWAP]] == inv[x]

In[45]:= inv[composite[x_, SWAP]] := inv[x]
```

Observation. The opposite (= flip) of a groupoid is a groupoid.

```
In[46]:= implies[equal[domain[inv[cat[x]]], range[cat[x]]],
          equal[domain[inv[flip[cat[x]]]], range[flip[cat[x]]]]]
Out[46]= True
```

Theorem. Direct products.

```
In[47]:= SubstTest[intersection, image[inverse[t], ids[t]],
          inverse[image[inverse[t], ids[t]]], t → direct[x, y]]

Out[47]= inv[composite[cross[x, y], TWIST]] == cross[inv[x], inv[y]]

In[48]:= inv[composite[cross[x_, y_], TWIST]] := cross[inv[x], inv[y]]
```

Observation. The direct product of groupoids is a groupoid.

```
In[49]:= implies[and[equal[domain[inv[cat[x]]], range[cat[x]]],
            equal[domain[inv[cat[y]]], range[cat[y]]]],
           equal[domain[inv[direct[cat[x], cat[y]]]], range[direct[cat[x], cat[y]]]]]

Out[49]= True
```

the case of a category

When **x** is a category, some additional properties of **inv[x]** can be derived. For simplicity the **cat[x]** wrapper will be used to formulate such rules.

Theorem.

```
In[50]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → inv[cat[x]], v -> domain[cat[x]], w → cartsq[range[cat[x]]]}] // Reverse

Out[50]= subclass[domain[inv[cat[x]]], range[cat[x]]] == True

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

It will next be shown that for a category, any identity morphism is its own inverse.

Lemma.

```
In[52]:= SubstTest[subclass, composite[cat[x], id[t]], cat[x], t -> id[ids[cat[x]]]] // Reverse

Out[52]= subclass[composite[id[ids[cat[x]]], inverse[DUP]], cat[x]] == True

In[53]:= subclass[composite[id[ids[cat[x_]]], inverse[DUP]], cat[x_]] := True
```

Theorem. Any identity morphism is its own inverse.

```
In[54]:= SubstTest[implies, subclass[u, v],
           subclass[image[inverse[u], w], image[inverse[v], w]],
           {u -> composite[id[ids[cat[x]]], inverse[DUP]],
            v -> cat[x], w → ids[cat[x]]}] // Reverse

Out[54]= subclass[ids[cat[x]], fix[inv[cat[x]]]] == True

In[55]:= subclass[ids[cat[x_]], fix[inv[cat[x_]]]] := True
```

Theorem. The composite of a morphism and its inverse is an identity morphism.

```
In[56]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → cat[x], u → inv[cat[x]], v -> image[inverse[cat[x]], ids[cat[x]]]}] // Reverse

Out[56]= subclass[image[cat[x], inv[cat[x]]], ids[cat[x]]] == True

In[57]:= subclass[image[cat[x_], inv[cat[x_]]], ids[cat[x_]]] := True
```

In the case of categories, it is convenient to use **APPLY** rules.

Lemma.

```
In[58]:= (member[w, image[inverse[funpart[t]], y]] // AssertTest) /. {w → pair[u, v], t → cat[x]}

Out[58]= member[pair[u, v], image[inverse[cat[x]], y]] == member[APPLY[cat[x], PAIR[u, v]], y]

In[59]:= member[pair[u_, v_], image[inverse[cat[x_]], y_]] :=
          member[APPLY[cat[x], PAIR[u, v]], y]
```

Theorem. If **u** and **v** are an inverse pair of morphisms, then **u · v** is an identity morphism.

```
In[60]:= Map[implies[#, member[APPLY[cat[x], PAIR[u, v]], ids[cat[x]]]] &,
           SubstTest[member, pair[u, v], intersection[t, inverse[t]],
            t -> image[inverse[cat[x]], ids[cat[x]]]]] // Reverse

Out[60]= or[member[APPLY[cat[x], PAIR[u, v]], ids[cat[x]]],
           not[member[pair[u, v], inv[cat[x]]]]] == True

In[61]:= or[member[APPLY[cat[x_], PAIR[u_, v_]], ids[cat[x_]]],
           not[member[pair[u_, v_], inv[cat[x_]]]]] := True
```

Theorem. If **u** and **v** are an inverse pair of morphisms, then **v · u** is an identity morphism.

```
In[62]:= Map[implies[#, member[APPLY[cat[x], PAIR[v, u]], ids[cat[x]]]] &,
           SubstTest[member, pair[u, v], intersection[t, inverse[t]],
            t -> image[inverse[cat[x]], ids[cat[x]]]]] // Reverse

Out[62]= or[member[APPLY[cat[x], PAIR[v, u]], ids[cat[x]]],
           not[member[pair[u, v], inv[cat[x]]]]] == True

In[63]:= or[member[APPLY[cat[x_], PAIR[v_, u_]], ids[cat[x_]]],
           not[member[pair[u_, v_], inv[cat[x_]]]]] := True
```

uniqueness of inverses

It will be shown in this section that if **x** is a category, then **inv[x]** is a function. (This is Lemma 1.6 on page 191 in the book by S.-T. Hu, referenced earlier.)

Observation. The next lemma is needed to cope with the following rewrite rule which expresses the associative law.

```
In[64]:= APPLY[cat[x], PAIR[u, APPLY[cat[x], PAIR[v, w]]]]

Out[64]= APPLY[cat[x], PAIR[APPLY[cat[x], PAIR[u, v]], w]]
```

Technical Lemma. (Needed to accomodate rewriting due to the associative law.)

```
In[65]:= SubstTest[implies, and[member[pair[u, t], domain[cat[x]]], member[t, ids[cat[x]]]],
           equal[u, APPLY[cat[x], PAIR[u, t]]], t → APPLY[cat[x], PAIR[v, w]]] // Reverse

Out[65]= or[equal[u, APPLY[cat[x], PAIR[APPLY[cat[x], PAIR[u, v]], w]]],
           not[member[APPLY[cat[x], PAIR[v, w]], ids[cat[x]]]],
           not[member[pair[u, APPLY[cat[x], PAIR[v, w]]], domain[cat[x]]]]] == True

In[66]:= (% /. {u → u_, v → v_, w → w_, x → x_}) /. Equal→ SetDelayed
```

Lemma. If $\mathbf{u} \cdot \mathbf{v}$ and $\mathbf{v} \cdot \mathbf{w}$ are both identity morphisms in a category, then $\mathbf{u} = \mathbf{w}$.

```
In[67]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[and[p1, p2], p4], implies[and[p5, p6], p7], not[implies[p0, p7]],
            {p0 → and[member[APPLY[cat[x], PAIR[u, v]], ids[cat[x]]],
               member[APPLY[cat[x], PAIR[v, w]], ids[cat[x]]]],
             p1 -> member[pair[u, v], domain[cat[x]]],
             p2 -> member[pair[v, w], domain[cat[x]]],
             p3 → member[pair[APPLY[cat[x], PAIR[u, v]], w], domain[cat[x]]],
             p4 -> member[pair[u, APPLY[cat[x], PAIR[v, w]]], domain[cat[x]]],
             p5 → equal[APPLY[cat[x], PAIR[APPLY[cat[x], PAIR[u, v]], w]], w],
             p6 → equal[u, APPLY[cat[x], PAIR[u, APPLY[cat[x], PAIR[v, w]]]]],
             p7 → equal[u, w]}]] // Reverse

Out[67]= or[equal[u, w], not[member[APPLY[cat[x], PAIR[u, v]], ids[cat[x]]]],
           not[member[APPLY[cat[x], PAIR[v, w]], ids[cat[x]]]]] == True

In[68]:= or[equal[u_, w_], not[member[APPLY[cat[x_], PAIR[u_, v_]], ids[cat[x_]]]],
           not[member[APPLY[cat[x_], PAIR[v_, w_]], ids[cat[x_]]]]] := True
```

Corollary. If $\mathbf{u}$ and $\mathbf{v}$ are a pair of inverse morphisms in a category, and if $\mathbf{v}$ and $\mathbf{w}$ are also a pair of inverse morphisms, then $\mathbf{u} = \mathbf{w}$.

```
In[69]:= Map[not, SubstTest[and, implies[p0, p1],
            implies[p0, p2], implies[and[p1, p2], p3], not[implies[p0, p3]],
            {p0 -> and[member[pair[u, v], inv[cat[x]]], member[pair[v, w], inv[cat[x]]]],
             p1 -> member[APPLY[cat[x], PAIR[u, v]], ids[cat[x]]],
             p2 → member[APPLY[cat[x], PAIR[v, w]], ids[cat[x]]],
             p3 → equal[u, w]}]] // Reverse

Out[69]= or[equal[u, w], not[member[pair[u, v], inv[cat[x]]]],
           not[member[pair[v, w], inv[cat[x]]]]] == True

In[70]:= or[equal[u_, w_], not[member[pair[u_, v_], inv[cat[x_]]]],
           not[member[pair[v_, w_], inv[cat[x_]]]]] := True
```

Main Theorem. (Obtained by eliminating the three variables $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$.)

```
In[71]:= Map[empty[composite[complement[#], id[cart[V, V]]]] &,
          SubstTest[class, pair[pair[u, v], w], or[equal[u, w],
            not[member[pair[u, v], t]], not[member[pair[v, w], t]]], t → inv[cat[x]]]]

Out[71]= FUNCTION[inv[cat[x]]] == True

In[72]:= FUNCTION[inv[cat[x_]]] := True
```

Corollary. Restatement without the **cat** wrapper.

```
In[73]:= SubstTest[implies, equal[x, cat[t]], FUNCTION[inv[x]], t → x] // Reverse

Out[73]= or[FUNCTION[inv[x]], not[category[x]]] == True

In[74]:= or[FUNCTION[inv[x_]], not[category[x_]]] := True
```

Corollary. Uniqueness of inverses for monoids: If an element of a monoid has an inverse, then it only has only one inverse.

```
In[75]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 -> member[x, MONOIDS], p2 -> category[x], p3 -> FUNCTION[inv[x]]}]] // Reverse

Out[75]= or[FUNCTION[inv[x]], not[member[x, MONOIDS]]] == True

In[76]:= or[FUNCTION[inv[x_]], not[member[x_, MONOIDS]]] := True
```