inverse images of functions preserve intersections

Johan G. F. Belinfante and Tiffany Goble
2003 August 29

```
In[1]:= << goedel52.s82; << tools.m

        :Package Title: goedel52.s82       2003 August 27 at 5:15 a.m.

        It is now:  2003 Aug 29 at 9:3

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 August 9

        weightlimit = 40
```

summary

Inverse images of functions preserve intersections. If the inverse of a function **x** is thin, one obtains a simple equation involving **CAP** and **IMAGE[x]**. Both of these results were previously available as conditional rewrite rules, but not yet as statements of fact. Having these results in the form of assertions is expected to be useful for applications, especially in topology.

a curious derivation

The following curious derivation yields directly a basic result that was already available as a conditional rewrite rule, but not yet in the form of an assertion.

```
In[2]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
            {u -> intersection[composite[inverse[FIRST], inverse[x]],
            composite[inverse[SECOND], inverse[x]]],
          v -> composite[DUP, inverse[x]], w -> composite[FIRST, id[cart[V, cart[y, z]]]]}]

Out[2]= or[equal[image[x, intersection[y, z]], intersection[image[x, y], image[x, z]]],
          not[FUNCTION[inverse[x]]]] == True

In[3]:= or[equal[image[x_, intersection[y_, z_]], intersection[image[x_, y_], image[x_, z_]]],
          not[FUNCTION[inverse[x_]]]] := True
```

The converse is also true: if all intersections are preserved by images under **x**, then the inverse of **x** is a function. To show this, it suffices to consider the case that **y** and **z** are singletons. We begin with a lemma:

```
In[4]:= SubstTest[implies, subclass[y, Id],
         subclass[composite[x, y], x], y -> composite[inverse[x], x]]

Out[4]= or[not[FUNCTION[inverse[x]]], subclass[composite[x, inverse[x], x], x]] == True

In[5]:= or[not[FUNCTION[inverse[x_]]], subclass[composite[x_, inverse[x_], x_], x_]] := True
```

Thus:

```
In[6]:= equiv[and[FUNCTION[inverse[x]], subclass[composite[x, inverse[x], x], x]],
         FUNCTION[inverse[x]]]

Out[6]= True
```

This factis now added as a temporary rewrite rule:

```
In[7]:= and[FUNCTION[inverse[x_]], subclass[composite[x_, inverse[x_], x_], x_]] :=
         FUNCTION[inverse[x]]
```

The desired converse follows immediately:

```
In[8]:= assert[forall[y, z, equal[image[x, intersection[singleton[y], singleton[z]]],
           intersection[image[x, singleton[y]], image[x, singleton[z]]]]]]

Out[8]= FUNCTION[inverse[x]]
```

a corollary about IMAGE[x]

The function **IMAGE[x]** takes **y** to **image[x,y]** provided the latter is a set. The result derived in the preceding section about the preservation of iintersections implies a property of this function when **inverse[x]** is a function. In the remainder of this notebook, this formula is derived for the special case that **x** is thin. When **x** is thin, **image[x,y]** is a set whenever **y** is a set.

two lemmas

Lemma 1.

```
In[9]:= SubstTest[equal, 0, fix[composite[z, Di, inverse[z]]], z -> composite[x, id[y]]]

Out[9]= equal[0, intersection[y, fix[composite[Di, id[y], inverse[x], x]]]] ==
         FUNCTION[composite[id[y], inverse[x]]]

In[10]:= equal[0, intersection[y_, fix[composite[Di, id[y_], inverse[x_], x_]]]] :=
          FUNCTION[composite[id[y], inverse[x]]]
```

Lemma 2.

```
In[11]:= U[intersection[image[CAP, composite[SINGLETON, Di, inverse[SINGLETON]]], P[x]]] //
          Renormality

Out[11]= U[intersection[image[CAP, composite[SINGLETON, Di, inverse[SINGLETON]]], P[x]]] == 0

In[12]:= U[intersection[image[CAP, composite[SINGLETON, Di, inverse[SINGLETON]]], P[x_]]] := 0
```

derivations

The following argument shows that a certain simple equation relating **CAP** and **IMAGE[x]** implies that **x** must be thin.

```
In[13]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
             {u -> composite[IMAGE[x], CAP], v -> composite[CAP, cross[IMAGE[x], IMAGE[x]]],
                 w -> composite[inverse[E], FIRST, FIRST]}]

Out[13]= or[equal[V, domain[VERTSECT[x]]], not[
            equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]]]] == True

In[14]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The same equation also implies that **inverse[x]** is a function, but the argument is more involved. The following is a partial result:

```
In[15]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
             {u -> composite[IMAGE[x], CAP], v -> composite[CAP, cross[IMAGE[x], IMAGE[x]]],
                 w -> composite[inverse[E], SECOND,
               id[cart[composite[SINGLETON, Di, inverse[SINGLETON]], V]]]}]

Out[15]= or[FUNCTION[composite[id[domain[VERTSECT[x]]], inverse[x]]], not[
            equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]]]] == True

In[16]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The following lemma is needed:

```
In[17]:= Map[or[not[thin[x]], #] &,
           SubstTest[implies, and[equal[u, v], FUNCTION[v]], FUNCTION[u],
             {u -> inverse[x], v -> composite[id[domain[VERTSECT[x]]], inverse[x]]}]] //
          MapNotNot

Out[17]= or[FUNCTION[inverse[x]], not[equal[V, domain[VERTSECT[x]]]],
           not[FUNCTION[composite[id[domain[VERTSECT[x]]], inverse[x]]]]] == True

In[18]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The rest of the argument can now be given:

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
                {p1 -> equal[composite[IMAGE[x], CAP],
             composite[CAP, cross[IMAGE[x], IMAGE[x]]]],
                    p2 -> thin[x], p3 -> FUNCTION[
             composite[id[domain[VERTSECT[x]]], inverse[x]]],
                    p4 -> FUNCTION[inverse[x]]}]]

Out[19]= or[FUNCTION[inverse[x]], not[
            equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]]]] == True

In[20]:= (% /. x -> x_) /. Equal -> SetDelayed
```

other direction

The next goal is to show that the converse holds; if **x** is thin and its inverse is a function, then the equation relating **CAP** and **IMAGE[x]** holds. It is easy to get a partial result:

```
In[21]:= intersection[composite[CAP, cross[IMAGE[x], IMAGE[x]]],
           composite[complement[IMAGE[x]], CAP],
             complement[image[V, complement[domain[VERTSECT[x]]]]],
           complement[image[V, fix[composite[inverse[x], x, Di]]]]] // VSNormality

Out[21]= composite[intersection[
            composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[complement[IMAGE[x]], CAP]],
           id[intersection[complement[image[V, complement[domain[VERTSECT[x]]]]],
             complement[image[V, fix[composite[inverse[x], x, Di]]]]]] == 0

In[22]:= (% /. x -> x_) /. Equal -> SetDelayed
```

This yields an inclusion:

```
In[23]:= SubstTest[equal, 0, dif[u, v], {u -> composite[CAP, cross[IMAGE[x], IMAGE[x]]],
         v -> union[composite[IMAGE[x], CAP],     image[V, complement[domain[VERTSECT[x]]]],
              image[V, fix[composite[inverse[x], x, Di]]]]}] // Reverse

Out[23]= or[not[equal[V, domain[VERTSECT[x]]]], not[FUNCTION[inverse[x]]],
           subclass[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]]] == True

In[24]:= (% /. x -> x_) /. Equal -> SetDelayed
```

This inclusion involves functions, and can therefore be replaced with an equation:

```
In[25]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
             {u -> composite[CAP, cross[IMAGE[x], IMAGE[x]]], v -> composite[IMAGE[x], CAP]}]

Out[25]= or[equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP,
             id[cart[P[domain[VERTSECT[x]]], P[domain[VERTSECT[x]]]]]]], not[subclass[
             composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]]]] == True

In[26]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma:

```
In[27]:= Map[or[not[thin[x]], #] &,
           SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
             {u -> composite[CAP, cross[IMAGE[x], IMAGE[x]]],
                 v -> composite[IMAGE[x], CAP],
               id[cart[P[domain[VERTSECT[x]]], P[domain[VERTSECT[x]]]]],
                 w -> composite[IMAGE[x], CAP]}]] // MapNotNot

Out[27]= or[equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]],
           not[equal[V, domain[VERTSECT[x]]]],
           not[equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x],
              CAP, id[cart[P[domain[VERTSECT[x]]], P[domain[VERTSECT[x]]]]]]]]] == True

In[28]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Putting it all together:

```
In[29]:= Map[not,
          SubstTest[and, implies[and[p1, p2], p3], implies[p3, p4], implies[and[p1, p4], p5],
              not[implies[and[p1, p2], p5]],
            {p1 -> thin[x], p2 -> FUNCTION[inverse[x]], p3 ->
            subclass[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]],
              p4 -> equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x],
               CAP, id[cart[P[domain[VERTSECT[x]]], P[domain[VERTSECT[x]]]]]]],
              p5 -> equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]],
             composite[IMAGE[x], CAP]]}]]

Out[29]= or[equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]],
           not[equal[V, domain[VERTSECT[x]]]], not[FUNCTION[inverse[x]]]] == True

In[30]:= (% /. x -> x_) /. Equal -> SetDelayed
```

combining the theorem and its converse into a rewrite rule

The final step is to combine the theorem and its converse into an if−and−only−if statement:

```
In[31]:= equiv[equal[composite[CAP, cross[IMAGE[x], IMAGE[x]]], composite[IMAGE[x], CAP]],
             and[thin[x], FUNCTION[inverse[x]]]] // not // not

Out[31]= True
```

This justifies the following:

```
In[32]:= equal[composite[CAP, cross[IMAGE[x_], IMAGE[x_]]], composite[IMAGE[x_], CAP]] :=
          and[equal[V, domain[VERTSECT[x]]], FUNCTION[inverse[x]]]
```

This result was previously available as a conditional rewrite rule, but now we also have it as a statement of fact.

comments

There are other functions that satisfy the condition **composite[f, CAP] = composite[CAP, cross[f, f]]**.

```
In[33]:= Select[Funs[NamedClasses], Equal[composite[#, CAP], composite[CAP, cross[#, #]]] &]

Out[33]= {0, CLIQUES, IMAGE[inverse[DUP]], HC, Id, IMAGE[DUP], POWER}
```

Of these, the functions **IMAGE[inverse[DUP]]**, **Id** and **IMAGE[DUP]** are of the form considered in this notebook, but the others are not. The case of **HC** is of particular interest because the proper class **FULL** satisfies axioms similar to a topology and **HC** is the interior operator for this pseudo−topology.

```
In[34]:= CORE[FULL]

Out[34]= HC
```