images and inverse images for functions

Johan G. F. Belinfante and Lee Martie
2006 April 27

```
In[1]:= SetDirectory["l:"]; << goedel80.24a; << tools.m

        :Package Title: goedel80.24a          2006 April 24 at 6:50 p.m.

        It is now:  2006 Apr 27 at 16:5

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 March 7

        weightlimit = 40
```

summary

For functions there is a relation between subvariance and invariance. For example:

```
In[2]:= subvar[inverse[funpart[x]]]

Out[2]= intersection[invar[funpart[x]], P[domain[funpart[x]]]]
```

A general theorem about this is derived in this notebook.

derivation

Lemma.

```
In[3]:= SubstTest[implies, equal[y, intersection[w, domain[funpart[x]]]],
          subclass[y, image[inverse[funpart[x]], image[funpart[x], w]]], w → y]

Out[3]= or[not[subclass[y, domain[funpart[x]]]],
           subclass[y, image[inverse[funpart[x]], image[funpart[x], y]]]] == True

In[4]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A corollary of the lemma:

```
In[5]:= SubstTest[implies, and[equal[w, image[funpart[x], y]], subclass[y, domain[funpart[x]]]],
          subclass[y, image[inverse[funpart[x]], w]], w → intersection[z, image[funpart[x], y]]]

Out[5]= or[not[subclass[y, domain[funpart[x]]]], not[subclass[image[funpart[x], y], z]],
           subclass[y, image[inverse[funpart[x]], intersection[z, image[funpart[x], y]]]]] == True
```

```
In[6]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[7]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
          implies[p3, p4], not[implies[and[p1, p2], p4]],
          {p1 -> subclass[y, domain[funpart[x]]], p2 -> subclass[image[funpart[x], y], z],
           p3 -> subclass[y, image[inverse[funpart[x]], intersection[z, image[funpart[x], y]]]],
           p4 -> subclass[y, image[inverse[funpart[x]], z]]}]]

Out[7]= or[not[subclass[y, domain[funpart[x]]]], not[subclass[image[funpart[x], y], z]],
          subclass[y, image[inverse[funpart[x]], z]]] == True

In[8]:= or[not[subclass[image[funpart[x_], y_], z_]], not[subclass[y_, domain[funpart[x_]]]],
          subclass[y_, image[inverse[funpart[x_]], z_]]] := True
```

The **funpart** wrapper can be removed.

```
In[9]:= SubstTest[implies, and[equal[x, funpart[w]], subclass[y, domain[x]],
          subclass[image[x, y], z]], subclass[y, image[inverse[x], z]], w → x]

Out[9]= or[not[FUNCTION[x]], not[subclass[y, domain[x]]],
          not[subclass[image[x, y], z]], subclass[y, image[inverse[x], z]]] == True

In[10]:= or[not[FUNCTION[x_]], not[subclass[y_, domain[x_]]],
           not[subclass[image[x_, y_], z_]], subclass[y_, image[inverse[x_], z_]]] := True
```

variable-free

The variables **y** and **z** can be eliminated, yielding the following corollary:

```
In[28]:= SubstTest[implies, equal[x, funpart[y]],
          equal[composite[S, IMAGE[x], id[P[domain[x]]]], UB[composite[E, x]]], y → x]

Out[28]= or[equal[composite[S, IMAGE[x], id[P[domain[x]]]], UB[composite[E, x]]],
           not[FUNCTION[x]]] == True

In[29]:= or[equal[composite[S, IMAGE[x_], id[P[domain[x_]]]], UB[composite[E, x_]]],
           not[FUNCTION[x_]]] := True
```

some examples

If **x** is a function with a thin inverse, there is a relation between **composite[S, IMAGE[x]]** and **composite[-IMAGE[inverse[x]], S]**. This connection can be derived directly using **VSRenormality** in many cases:

```
In[60]:= composite[inverse[IMAGE[inverse[ACLOSURE]]], S] // VSRenormality

Out[60]= composite[inverse[IMAGE[inverse[ACLOSURE]]], S] == composite[S, IMAGE[ACLOSURE]]
```

```
In[61]:= composite[inverse[IMAGE[inverse[ACLOSURE]]], S] := composite[S, IMAGE[ACLOSURE]]

In[47]:= composite[inverse[IMAGE[inverse[BIGCUP]]], S] // VSRenormality

Out[47]= composite[inverse[IMAGE[inverse[BIGCUP]]], S] == composite[S, IMAGE[BIGCUP]]

In[62]:= composite[inverse[IMAGE[inverse[BIGCUP]]], S] := composite[S, IMAGE[BIGCUP]]

In[63]:= composite[S, IMAGE[POWER]] // VSRenormality // Reverse

Out[63]= composite[S, IMAGE[id[range[POWER]]], IMAGE[inverse[BIGCUP]]] ==
          composite[S, IMAGE[POWER]]

In[64]:= % /. Equal → SetDelayed

In[65]:= composite[inverse[IMAGE[inverse[POWER]]], S] // RelnRenormality

Out[65]= composite[inverse[IMAGE[inverse[POWER]]], S] == composite[S, IMAGE[POWER]]

In[66]:= composite[inverse[IMAGE[inverse[POWER]]], S] := composite[S, IMAGE[POWER]]

In[67]:= composite[inverse[IMAGE[inverse[SINGLETON]]], S] // VSRenormality

Out[67]= composite[inverse[IMAGE[inverse[SINGLETON]]], S] == composite[S, IMAGE[SINGLETON]]

In[68]:= composite[inverse[IMAGE[inverse[SINGLETON]]], S] := composite[S, IMAGE[SINGLETON]]

In[69]:= composite[inverse[IMAGE[inverse[UCLOSURE]]], S] // VSRenormality

Out[69]= composite[inverse[IMAGE[inverse[UCLOSURE]]], S] == composite[S, IMAGE[UCLOSURE]]

In[70]:= composite[inverse[IMAGE[inverse[UCLOSURE]]], S] := composite[S, IMAGE[UCLOSURE]]
```

---

serendipity: a formula for funpart[x]

Observation:

```
In[11]:= class[u,
           forall[v, w, implies[and[member[pair[u, v], x], member[pair[u, w], x]], equal[v, w]]]]

Out[11]= complement[fix[composite[inverse[x], Di, x]]]
```

Theorem.

```
In[13]:= Assoc[x, id[domain[x]], id[complement[fix[composite[inverse[x], Di, x]]]]] // Reverse

Out[13]= composite[x, id[complement[fix[composite[inverse[x], Di, x]]]]] == funpart[x]

In[14]:= composite[x_, id[complement[fix[composite[inverse[x_], Di, x_]]]]] := funpart[x]
```

Comment. This result can also be obtained using **VSNormality**.