two theorems about inverse images of functions

Johan G. F. Belinfante
2004 July 17

```
In[1]:= SetDirectory["p:"]; << goedel59.16a; << tools.m

        :Package Title: goedel59.16a    2004 July 16 at 3:15 p.m.

        It is now:  2004 Jul 17 at 19:38

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 June 22

        weightlimit = 40
```

summary

Two theorems about inverse images of functions are derived, which involve **APPLY**. Two other minor results are derived that are not needed for the main result, but which also have something to do with application.

lemmas

Lemma 1.

```
In[2]:= SubstTest[implies, member[y, w], member[y, V], w → domain[funpart[x]]]

Out[2]= or[member[y, V], not[member[image[x, singleton[y]], range[SINGLETON]]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma 2.

```
In[4]:= SubstTest[implies, member[w, z], member[w, V], w → APPLY[funpart[x], y]]

Out[4]= or[member[image[x, singleton[y]], range[SINGLETON]],
            not[member[APPLY[funpart[x], y], z]]] == True

In[5]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

```
In[6]:= Map[not,
         SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
          {p1 -> member[APPLY[funpart[x], y], z],
           p2 -> member[image[x, singleton[y]], range[SINGLETON]], p3 → member[y, V]}]]

Out[6]= or[member[y, V], not[member[APPLY[funpart[x], y], z]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Note that:

```
In[8]:= equiv[and[member[APPLY[funpart[x], y], z], member[y, V]],
         member[APPLY[funpart[x], y], z]]

Out[8]= True
```

This justifies the following temporary rewrite rule:

```
In[9]:= and[member[y_, V], member[APPLY[funpart[x_], y_], z_]] :=
         member[APPLY[funpart[x], y], z]
```

---

inverse images for functions

The main result of this notebook can be stated most succinctly using **funpart**. To prevent looping on the test suite example **ap3**, this rule must be oriented as folllows:

```
In[10]:= member[y, image[inverse[funpart[x]], z]] // AssertTest // Reverse

Out[10]= member[APPLY[funpart[x], y], z] == member[y, image[inverse[funpart[x]], z]]

In[11]:= member[APPLY[funpart[x_], y_], z_] := member[y, image[inverse[funpart[x]], z]]
```

---

eliminating the funpart wrapper

By eliminating the **funpart** wrapper, one can derive two theorems from this. The first of these is:

```
In[12]:= SubstTest[implies, and[equal[x, w], or[member[A[image[w, singleton[y]]], z],
            not[member[y, image[inverse[w], z]]]]],
          or[member[A[image[x, singleton[y]]], z],
           not[member[y, image[inverse[x], z]]]], w → funpart[x]]

Out[12]= or[member[APPLY[x, y], z], not[FUNCTION[x]],
           not[member[y, image[inverse[x], z]]]] == True

In[13]:= or[member[APPLY[x_, y_], z_], not[FUNCTION[x_]],
           not[member[y_, image[inverse[x_], z_]]]] := True
```

This is the second:

```
In[14]:= SubstTest[implies, and[equal[x, w],
           or[member[y, image[inverse[w], z]], not[member[APPLY[w, y], z]]]],
          or[member[y, image[inverse[x], z]], not[member[APPLY[x, y], z]]],
          w → funpart[x]]

Out[14]= or[member[y, image[inverse[x], z]],
           not[FUNCTION[x]], not[member[APPLY[x, y], z]]] == True

In[15]:= or[member[y_, image[inverse[x_], z_]],
           not[FUNCTION[x_]], not[member[APPLY[x_, y_], z_]]] := True
```

---

vertical sections of functions are sets

The result in this section is a simple corollary of the axiom of replacement.

```
In[16]:= SubstTest[implies, and[equal[x, z], member[image[z, singleton[y]], V]],
          member[image[x, singleton[y]], V], z → funpart[x]]

Out[16]= or[member[image[x, singleton[y]], V], not[FUNCTION[x]]] == True

In[17]:= or[member[image[x_, singleton[y_]], V], not[FUNCTION[x_]]] := True
```

---

a general inclusion for APPLY

The result in this final section is a general inclusion for **APPLY**, which is not limited to functions.

```
In[18]:= SubstTest[implies, member[y, w], subclass[A[w], y], w → image[z, singleton[x]]]

Out[18]= or[not[member[x, V]], not[member[y, V]],
           not[member[pair[x, y], z]], subclass[APPLY[z, x], y]] == True

In[19]:= or[not[member[x_, V]], not[member[y_, V]],
           not[member[pair[x_, y_], z_]], subclass[APPLY[z_, x_], y_]] := True
```