constant functions

Johan G. F. Belinfante
2006 June 7

```
In[1]:= SetDirectory["l:"]; << goedel82.06a; << tools.m

        :Package Title: goedel82.06a         2006 June 6 at 8:15 a.m.

        It is now:  2006 Jun 7 at 2:10

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 June 6

        weightlimit = 40
```

summary

What is a constant function? One possible definition is that it is a function that has the same value at every point of its domain. Another possible definition is that it is a function with a single value. The difference between these two definitions is that the former includes the empty function, and the latter excludes it. The class **CONST** is defined here as the class of all functions that belongs to **map[u, set[v]]** for some **u** and **v**. To prevent the statement **member[x, CONST]** from always being expanded, the definition is wrapped with **class**:

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

In[3]:= InfoMatch[class[w_, member[x_, HoldPattern[CONST]]]]

Out[3]//TableForm=
        class[w_, member[x_, CONST]] := Module[{u = Unique[], v = Unique[]}, class[w, exists[u,
```

With this definition, the empty function is constant:

```
In[4]:= member[0, CONST] // AssertTest

Out[4]= member[0, CONST] == True

In[5]:= member[0, CONST] := True
```

In this notebook, some basic rewrite rules for the class of constant functions are derived.

removal of an old rewrite rule

An old rewrite rule is here removed, and will be replaced with a new one in a later section.

```
In[6]:= intersection[FUNS, range[CART]] =.
```

normalization

Lemma.

```
In[14]:= equal[intersection[complement[set[0]],
           image[inverse[IMAGE[SECOND]], range[SINGLETON]]],
          image[inverse[IMAGE[SECOND]], range[SINGLETON]]]

Out[14]= True

In[16]:= intersection[complement[set[0]], image[inverse[IMAGE[SECOND]], range[SINGLETON]]] :=
          image[inverse[IMAGE[SECOND]], range[SINGLETON]]
```

An application of **Normality** yields a formula which says in effect that a nonempty constant function has only one value:

```
In[17]:= dif[CONST, set[0]] // Normality // Reverse

Out[17]= intersection[FUNS, image[inverse[IMAGE[SECOND]], range[SINGLETON]]] ==
          intersection[CONST, complement[set[0]]]

In[18]:= intersection[FUNS, image[inverse[IMAGE[SECOND]], range[SINGLETON]]] :=
          intersection[CONST, complement[set[0]]]
```

a related equation

In this section, a related equation is derived with **P[cart[V,V]]** replacing **FUN**S.

```
In[20]:= symdif[dif[CONST, set[0]], intersection[
            image[inverse[IMAGE[SECOND]], range[SINGLETON]], P[cart[V, V]]]] // Renormality

Out[20]= union[intersection[CONST,
            complement[image[inverse[IMAGE[SECOND]], range[SINGLETON]]], complement[set[0]]],
           intersection[CONST, complement[P[cart[V, V]]], complement[set[0]]],
           intersection[complement[CONST],
            image[inverse[IMAGE[SECOND]], range[SINGLETON]], P[cart[V, V]]] == 0

In[21]:= % /. Equal → SetDelayed

In[22]:= SubstTest[equal, 0, symdif[u, v],
          {u → intersection[image[inverse[IMAGE[SECOND]], range[SINGLETON]], P[cart[V, V]]],
           v → dif[CONST, set[0]]}]

Out[22]= True == equal[intersection[CONST, complement[set[0]]],
           intersection[image[inverse[IMAGE[SECOND]], range[SINGLETON]], P[cart[V, V]]]]

In[23]:= intersection[image[inverse[IMAGE[SECOND]], range[SINGLETON]], P[cart[V, V]]] :=
          intersection[CONST, complement[set[0]]]
```

other formulas for CONST

A simpler formula for the class of constant functions is obtained by using **reify** in a fairly tricky fashion. This formula says that a constant function is a cartesian product of a set with a singleton.

```
In[24]:= Map[range, SubstTest[reify, x, map[first[x], set[f[second[x]]]], f → setpart]]

Out[24]= image[CART, cart[V, range[SINGLETON]]] == CONST

In[25]:= image[CART, cart[V, range[SINGLETON]]] := CONST
```

Corollary.

```
In[26]:= SubstTest[subclass, image[x, y], range[x], {x → CART, y → cart[V, range[SINGLETON]]}]

Out[26]= subclass[CONST, range[CART]] == True

In[27]:= subclass[CONST, range[CART]] := True
```

Corollary.

```
In[28]:= SubstTest[U, image[CART, cart[V, w]], w → range[SINGLETON]]

Out[28]= U[CONST] == cart[V, V]

In[29]:= U[CONST] := cart[V, V]
```

Yet another characterization of a constant function is obtained by using a rewrite rule for the binary function **MAP**.

```
In[30]:= Map[range, Assoc[composite[id[FUNS], inverse[DORA]],
            cross[Id, inverse[S]], cross[Id, SINGLETON]]] // Reverse

Out[30]= U[image[MAP, cart[V, range[SINGLETON]]]] == CONST

In[31]:= U[image[MAP, cart[V, range[SINGLETON]]]] := CONST
```

Corollary.

```
In[32]:= SubstTest[subclass, U[image[MAP, x]], FUNS, x -> cart[V, range[SINGLETON]]]

Out[32]= subclass[CONST, FUNS] == True

In[33]:= subclass[CONST, FUNS] := True
```

replacement rule

In this section a replacement is derived for the rule that was removed earlier. A lemma is needed:

```
In[34]:= SubstTest[image, CART, union[u, v], {u → cart[V, x], v → cart[V, set[0]]}]
Out[34]= image[CART, cart[V, union[x, set[0]]]] == union[image[CART, cart[V, x]], set[0]]

In[35]:= image[CART, cart[V, union[x_, set[0]]]] := union[image[CART, cart[V, x]], set[0]]
```

The replacement rule is:

```
In[36]:= ImageComp[CART, inverse[CART], FUNS]
Out[36]= intersection[FUNS, range[CART]] == CONST

In[37]:= intersection[FUNS, range[CART]] := CONST
```

a hereditary property

One advantage of considering the empty function to be constant is that any subset of a constant function is constant.

```
In[38]:= Map[range, composite[inverse[S], CART, cross[Id, SINGLETON]] // VSTriNormality]
Out[38]= image[inverse[S], CONST] == CONST

In[39]:= image[inverse[S], CONST] := CONST
```

membership rule

The following membership rule is basic.

```
In[40]:= member[cart[x, set[y]], CONST] // AssertTest
Out[40]= member[cart[x, set[y]], CONST] == or[member[x, V], not[member[y, V]]]

In[41]:= member[cart[x_, set[y_]], CONST] := or[member[x, V], not[member[y, V]]]
```