map[set[x], y]

Johan G. F. Belinfante and Lee Martie
2006 July 6

```
In[1]:= SetDirectory["l:"]; << goedel83.04a; << tools.m

        :Package Title: goedel83.04a         2006 July 4 at 5:15 a.m.

        It is now:  2006 Jul 6 at 15:50

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 June 27

        weightlimit = 40
```

summary

A formula for the class of mappings with one-point domains is derived.

derivation

Lemma. If the domain of a function is a singleton, then its graph consists of a single point.

```
In[2]:= SubstTest[implies, equal[u, v], equal[composite[w, id[u]], composite[w, id[v]]],
         {u → domain[funpart[x]], v → set[setpart[y]], w → funpart[x]}]

Out[2]= or[equal[cart[set[setpart[y]], set[APPLY[funpart[x], setpart[y]]]], funpart[x]],
          not[equal[domain[funpart[x]], set[setpart[y]]]]] == True

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

In[4]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[and[p2, p3], p4],
          not[implies[p1, p4]], {p1 -> member[funpart[x], map[set[setpart[y]], z]],
           p2 -> equal[domain[funpart[x]], set[setpart[y]]],
           p3 -> equal[cart[set[setpart[y]], set[APPLY[funpart[x], setpart[y]]]], funpart[x]],
           p4 → member[funpart[x], range[SINGLETON]]}]]

Out[4]= or[member[funpart[x], range[SINGLETON]],
          not[member[funpart[x], map[set[setpart[y]], z]]]] == True

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

The variable **x** is removed as follows:

```
In[6]:= Map[equal[V, #] &,
          SubstTest[class, x, implies[member[funpart[x], y], member[funpart[x], t]],
           {y -> map[set[setpart[y]], z], t -> range[SINGLETON]}]] // Reverse

Out[6]= subclass[map[set[setpart[y]], z], range[SINGLETON]] == True

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

This result can be made more precise as follows:

```
In[8]:= SubstTest[subclass, w, intersection[u, v],
         {u → range[SINGLETON], v → P[cart[set[setpart[x]], y]], w → map[set[setpart[x]], y]}]

Out[8]= subclass[map[set[setpart[x]], y], image[SINGLETON, cart[set[setpart[x]], y]]] == True

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The reverse inclusion also holds:

```
In[10]:= SubstTest[subclass, u, image[inverse[SINGLETON], v],
           {u -> cart[set[setpart[x]], y], v -> map[set[setpart[x]], y]}] // Reverse

Out[10]= subclass[image[SINGLETON, cart[set[setpart[x]], y]], map[set[setpart[x]], y]] == True

In[11]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Combining the two inclusions yields an equation:

```
In[12]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[SINGLETON, cart[set[setpart[x]], y]], v -> map[set[setpart[x]], y]}]

Out[12]= True == equal[image[SINGLETON, cart[set[setpart[x]], y]], map[set[setpart[x]], y]]

In[13]:= map[set[setpart[x_]], y_] := image[SINGLETON, cart[set[setpart[x]], y]]
```

The **setpart** wrapper can be removed.

```
In[18]:= Map[implies[member[x, z], #] &, SubstTest[implies, equal[x, setpart[w]],
           equal[image[SINGLETON, cart[set[x], y]], map[set[x], y]], w → x]]

Out[18]= or[equal[image[SINGLETON, cart[set[x], y]], map[set[x], y]], not[member[x, z]]] == True

In[19]:= or[equal[image[SINGLETON, cart[set[x_], y_]], map[set[x_], y_]],
           not[member[x_, z_]]] := True
```

a variable-free version

Removing the variables yields an inclusion:

```
In[28]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], implies[and[member[x, V], member[y, V]],
             member[image[w, cart[set[x], y]], image[z, cart[set[set[x]], set[y]]]]],
            {w → SINGLETON, z → MAP}]] // Reverse

Out[28]= subclass[cart[range[SINGLETON], V],
           fix[composite[inverse[MAP], IMAGE[SINGLETON], CART]]] == True

In[29]:= % /. Equal → SetDelayed
```

This result can be cleaned up a bit as follows.

```
In[36]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → id[cart[range[SINGLETON], V]],
           v -> composite[inverse[MAP], IMAGE[SINGLETON], CART],
           w -> cross[cross[inverse[SINGLETON], Id], MAP]}]

Out[36]= subclass[composite[MAP, cross[SINGLETON, Id]],
           composite[IMAGE[SINGLETON], id[image[inverse[IMAGE[SINGLETON]], range[MAP]]],
            CART, cross[SINGLETON, Id]]] == True

In[37]:= % /. Equal → SetDelayed
```

Further simplification is possible:

```
In[40]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> composite[MAP, cross[SINGLETON, Id]], v -> composite[IMAGE[SINGLETON],
             id[image[inverse[IMAGE[SINGLETON]], range[MAP]]], CART, cross[SINGLETON, Id]],
           w -> composite[IMAGE[SINGLETON], CART, cross[SINGLETON, Id]]}]

Out[40]= subclass[composite[MAP, cross[SINGLETON, Id]],
           composite[IMAGE[SINGLETON], CART, cross[SINGLETON, Id]]] == True

In[41]:= % /. Equal → SetDelayed
```

An equation can be obtained as follows:

```
In[42]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
          {u -> composite[MAP, cross[SINGLETON, Id]],
           v -> composite[IMAGE[SINGLETON], CART, cross[SINGLETON, Id]]}]

Out[42]= equal[composite[MAP, cross[SINGLETON, Id]],
           composite[IMAGE[SINGLETON], CART, cross[SINGLETON, Id]]] == True

In[45]:= composite[MAP, cross[SINGLETON, Id]] :=
          composite[IMAGE[SINGLETON], CART, cross[SINGLETON, Id]]
```