CORE[image[S,singleton[x]]]

Johan G. F. Belinfante
2004 May 4

```
In[1]:= << goedel57.04b; << tools.m

        :Package Title: goedel57.04b        2004 May 4 at 4:20 a.m.

        It is now:  2004 May 5 at 10:45

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 April 26

        weightlimit = 40
```

summary

A characterization of **CORE** functions is applied in this notebook to the derivation of a formula for **CORE[image[-S,singleton[x]]]**. The result was originally discovered in a different, more roundabout way, but the derivation found earlier was quite messy and unenlightening. There is still some messiness in the present derivation, but the overall strategy is easier to understand.

definitions

```
In[2]:= idempotent[x_] := equal[composite[x, x], x]

In[3]:= total[x_] := equal[domain[x], V]
```

The strategy is to use this theorem which characterizes **CORE** functions:

```
In[4]:= implies[and[idempotent[x], total[x], subclass[x, inverse[S]],
          subcommute[x, S], FUNCTION[x]], equal[x, CORE[fix[x]]]]

Out[4]= True
```

two messy lemmas

```
In[5]:= union[cart[complement[image[S, singleton[x]]], singleton[0]],
           id[image[S, singleton[x]]]] // idempotent // AssertTest

Out[5]= equal[union[cart[complement[image[S, singleton[x]]], singleton[0]],
           id[image[S, singleton[x]]]], union[cart[complement[image[S, singleton[x]]],
            union[intersection[image[S, singleton[x]], singleton[0]],
             intersection[image[V, x], singleton[0]]]], id[image[S, singleton[x]]]]] == True

In[6]:= (% /. x -> x_) /. Equal -> SetDelayed
```

```
In[7]:= (subcommute[w, S] /. w -> union[cart[complement[image[S, singleton[x]]], singleton[0]],
          id[image[S, singleton[x]]]]) // AssertTest

Out[7]= and[equal[V,
          union[image[S, singleton[x]], lb[union[cart[complement[image[S, singleton[x]]], V],
             composite[S, id[image[S, singleton[x]]]]], singleton[0]]]],
         subclass[composite[id[image[S, singleton[x]]], S],
          union[cart[complement[image[S, singleton[x]]], V],
           composite[S, id[image[S, singleton[x]]]]]]] == True

In[8]:= (% /. x -> x_) /. Equal -> SetDelayed
```

---

results that can be made into permanent rules

```
In[9]:= FUNCTION[union[cart[complement[x], singleton[0]], id[x]]] // AssertTest

Out[9]= FUNCTION[union[cart[complement[x], singleton[0]], id[x]]] == True

In[10]:= FUNCTION[union[cart[complement[x_], singleton[0]], id[x_]]] := True

In[11]:= CORE[union[x, singleton[0]]] // RelnNormality

Out[11]= CORE[union[x, singleton[0]]] == CORE[x]

In[12]:= CORE[union[x_, singleton[0]]] := CORE[x]

In[13]:= SubstTest[implies, and[idempotent[w], total[w], subclass[w, inverse[S]],
           subcommute[w, S], FUNCTION[w]], equal[w, CORE[fix[w]]],
            w -> union[cart[complement[image[S, singleton[x]]], singleton[0]],
             id[image[S, singleton[x]]]]]

Out[13]= equal[CORE[image[S, singleton[x]]],
           union[cart[complement[image[S, singleton[x]]], singleton[0]],
            id[image[S, singleton[x]]]]] == True

In[14]:= CORE[image[S, singleton[x_]]] := union[
           cart[complement[image[S, singleton[x]]], singleton[0]], id[image[S, singleton[x]]]]
```