characterizing CORE[x]

Johan G. F. Belinfante
2003 November 12

```
In[1]:= << goedel52.t15; << tools.m

        :Package Title: goedel52.t15      2003 November 9 at 1:25 p.m.

        It is now:  2003 Nov 12 at 9:57

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 October 28

        weightlimit = 40
```

summary

A characterization of the function **CORE[x]** is derived. Execution time is greatly reduced by turning off the **simplify** flag.

```
In[2]:= simplify = False;
```

temporary definitions

Some temporary abbreviations are convenient:

```
In[3]:= core[x_, y_] := U[intersection[x, P[y]]]

In[4]:= idempotent[x_] := equal[composite[x, x], x]

In[5]:= total[x_] := equal[V, domain[x]]

In[6]:= hypotheses[x_] :=
         and[FUNCTION[x], total[x], idempotent[x], subcommute[x, S], subclass[x, inverse[S]]]
```

All hypotheses are satisfied by **CORE[x]**.

```
In[7]:= hypotheses[CORE[x]]

Out[7]= True
```

It will be shown in the sequel that if **x** satisfies the hypotheses, then **x = CORE[fix[x]]**.

Quaife's theorem (AP2)

An analog of Quaife's theorem **(AP2)** is derived in this section, but with **APPLY** in place of **apply**.

Lemma 1.

```
In[8]:= SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
            {u -> APPLY[z, x], v -> singleton[APPLY[z, x]], w -> image[z, singleton[x]]}]

Out[8]= or[member[pair[x, APPLY[z, x]], z],
          not[equal[image[z, singleton[x]], singleton[APPLY[z, x]]]],
          not[member[x, domain[z]]]] == True

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

Lemma 2 follows from Lemma 1:

```
In[10]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
                 {p1 -> FUNCTION[z], p2 -> member[x, domain[z]],
            p3 -> equal[image[z, singleton[x]], singleton[APPLY[z, x]]],
                      p4 -> member[pair[x, APPLY[z, x]], z]}]]

Out[10]= or[member[pair[x, APPLY[z, x]], z], not[FUNCTION[z]], not[member[x, domain[z]]]] == True

In[11]:= or[member[pair[x_, APPLY[z_, x_]], z_],
           not[FUNCTION[z_]], not[member[x_, domain[z_]]]] := True
```

Lemma 3.

```
In[12]:= SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
             {u -> pair[x, y], v -> z, w -> funpart[z]}]

Out[12]= or[and[equal[image[z, singleton[x]], singleton[y]], member[x, V], member[y, V]],
           not[FUNCTION[z]], not[member[pair[x, y], z]]] == True

In[13]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma 4.

```
In[14]:= SubstTest[implies, and[equal[u, v], subclass[v, w]], subclass[u, w],
             {u -> singleton[y], v -> image[z, singleton[x]], w -> range[z]}]

Out[14]= or[member[y, range[z]],
           not[equal[image[z, singleton[x]], singleton[y]]], not[member[y, V]]] == True

In[15]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma 5 combines Lemmas 3 and 4.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
                 {p1 -> and[FUNCTION[z], member[pair[x, y], z]],
            p2 -> and[equal[image[z, singleton[x]], singleton[y]], member[x, V], member[y, V]],
                      p3 -> member[y, range[z]]}]]

Out[16]= or[member[y, range[z]], not[FUNCTION[z]], not[member[pair[x, y], z]]] == True

In[17]:= or[member[y_, range[z_]], not[FUNCTION[z_]], not[member[pair[x_, y_], z_]]] := True
```

This implies an analog of Quaife's Theorem **(AP2)**:

```
In[18]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 -> member[x, domain[z]], p2 -> FUNCTION[z], p3 -> member[pair[x, APPLY[z, x]], z],
             p4 -> member[APPLY[z, x], range[z]]}]]

Out[18]= or[member[APPLY[z, x], range[z]], not[FUNCTION[z]], not[member[x, domain[z]]]] == True

In[19]:= or[member[APPLY[z_, x_], range[z_]],
           not[FUNCTION[z_]], not[member[x_, domain[z_]]]] := True
```

Uclosure lemma

The first goal is to show that if **z** is a total monotone function contained in **inverse[S],** then **fix[z]** is Uclosed. To this end, it is of interest to consider what happens when **z** is applied to the sum class of a subclass of **fix[z]**.

Lemma.

```
In[20]:= Map[not, SubstTest[and, implies[and[p1, p2], p5], implies[and[p3, p5], p6],
                 not[implies[and[p1, p2, p3], p6]], {p1 -> member[x, y],
            p2 -> subclass[y, fix[z]], p3 -> FUNCTION[z], p5 -> member[x, fix[z]],
                 p6 -> equal[APPLY[z, x], x]}]]

Out[20]= or[equal[x, APPLY[z, x]], not[FUNCTION[z]],
           not[member[x, y]], not[subclass[y, fix[z]]]] == True

In[21]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed

In[22]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p6],
           implies[p1, p7], implies[and[p3, p4, p7], p8], implies[and[p6, p8], p9],
                 not[implies[and[p1, p2, p3, p4], p9]], {p1 -> member[x, y],
            p2 -> subclass[y, fix[z]], p3 -> FUNCTION[z], p4 -> subcommute[z, S],
                 p6 -> equal[APPLY[z, x], x], p7 -> subclass[x, U[y]],
            p8 -> subclass[APPLY[z, x], APPLY[z, U[y]]],
                 p9 -> subclass[x, APPLY[z, U[y]]]}]]

Out[22]= or[not[FUNCTION[z]], not[member[x, y]], not[subclass[y, fix[z]]],
           not[subclass[composite[z, S], composite[S, z]]], subclass[x, APPLY[z, U[y]]]] == True

In[23]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

The second step is to eliminate the set variable **x**. The following is a new, faster way to do this:

```
In[24]:= SubstTest[assert[forall[w, #]] &, or[not[subclass[P[z], y]],
            not[member[w, x]], not[subclass[x, t]], not[subclass[u, s]], subclass[w, v]],
             {y -> FUNS, v -> APPLY[z, U[x]], t -> fix[z], u -> composite[z, S],
            s -> composite[S, z]}] // Reverse

Out[24]= or[not[FUNCTION[z]], not[subclass[x, fix[z]]],
           not[subclass[composite[z, S], composite[S, z]]],
           subclass[U[x], APPLY[z, U[x]]]] == True

In[25]:= (% /. {x -> x_, z -> z_}) /. Equal -> SetDelayed
```

The hypothesis that **z** is a subclass of **inverse[S]** yields the following lemma.

```
In[26]:= SubstTest[implies, subclass[z, w],
          subclass[image[z, y], image[w, y]], {w -> inverse[S], y -> singleton[x]}]

Out[26]= or[not[subclass[z, inverse[S]]], subclass[U[image[z, singleton[x]]], x]] == True
```

```
In[27]:= (% /. {x -> x_, z -> z_}) /. Equal -> SetDelayed
```

To replace **apply** by **APPLY** one needs to add the hypthesis that **x** is in the domain of **z**.

```
In[28]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
             {u -> APPLY[z, x], v -> apply[z, x]}]

Out[28]= or[not[member[x, domain[z]]],
           not[subclass[U[image[z, singleton[x]]], w]], subclass[APPLY[z, x], w]] == True

In[29]:= (% /. {w -> w_, x -> x_, z -> z_}) /. Equal -> SetDelayed

In[30]:= Map[not, SubstTest[and, implies[p2, p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 -> member[x, domain[z]], p2 -> subclass[z, inverse[S]],
            p3 -> subclass[U[image[z, singleton[x]]], x], p4 -> subclass[APPLY[z, x], x]}]]

Out[30]= or[not[member[x, domain[z]]],
           not[subclass[z, inverse[S]]], subclass[APPLY[z, x], x]] == True

In[31]:= (% /. {x -> x_, z -> z_}) /. Equal -> SetDelayed
```

Lemma.

```
In[32]:= SubstTest[implies, and[member[u, v], equal[v, z]], member[u, z], {u -> U[x], v -> V}]

Out[32]= or[member[U[x], z], not[equal[V, z]], not[member[x, V]]] == True

In[33]:= (% /. {x -> x_, z -> z_}) /. Equal -> SetDelayed
```

The totality hypothesis is now added. The argument uses the following facts:

```
In[34]:= {implies[and[p1, p2, p6], p7], implies[and[p4, p5], p8], implies[and[p3, p8], p9],
             implies[and[p7, p9], p10], implies[and[p1, p8, p10], p11]} /.
                 {p1 -> FUNCTION[z], p2 -> subcommute[z, S],
           p3 -> subclass[z, inverse[S]], p4 -> equal[domain[z], V], p5 -> member[x, V],
           p6 -> subclass[x, fix[z]], p7 -> subclass[U[x], APPLY[z, U[x]]],
         p8 -> member[U[x], domain[z]],     p9 -> subclass[APPLY[z, U[x]], U[x]],
           p10 -> equal[APPLY[z, U[x]], U[x]],
                    p11 -> member[U[x], fix[z]]}

Out[34]= {True, True, True, True, True}
```

To make it go faster, it helps to bundle the hypotheses.

```
In[35]:= Map[not, SubstTest[and, implies[and[p1, p6], p7],
           implies[and[p1, p5], p8], implies[and[p1, p8], p9],
             implies[and[p7, p9], p10], implies[and[p1, p8, p10], p11],
           not[implies[and[p1, p5, p6], p11]],
                 {p1 -> and[FUNCTION[z], subcommute[z, S], subclass[z, inverse[S]],
              equal[domain[z], V]], p5 -> member[x, V], p6 -> subclass[x, fix[z]],
            p7 -> subclass[U[x], APPLY[z, U[x]]], p8 -> member[U[x], domain[z]],
            p9 -> subclass[APPLY[z, U[x]], U[x]], p10 -> equal[APPLY[z, U[x]], U[x]],
         p11 -> member[U[x], fix[z]]}]]

Out[35]= or[member[U[x], fix[z]], not[equal[V, domain[z]]], not[FUNCTION[z]],
           not[member[x, V]], not[subclass[x, fix[z]]], not[subclass[z, inverse[S]]],
           not[subclass[composite[z, S], composite[S, z]]]] == True

In[36]:= (% /. {x -> x_, z -> z_}) /. Equal -> SetDelayed
```

Another variable is removed using heavy sequestering. Execution time is reduced dramatically by combining several conditions to reduce the number of literals:

```
In[37]:= subclass[union[P[z], image[V, complement[domain[z]]]],
          intersection[FUNS, P[inverse[S]]]]

Out[37]= and[equal[V, domain[z]], FUNCTION[z], subclass[z, inverse[S]]]

In[38]:= Map[equal[V, #] &, SubstTest[class, x,
            or[member[U[x], u], not[subclass[p, q]], not[member[x, y]], not[subclass[s, t]]],
             {u -> fix[z], p -> union[P[z], image[V, complement[domain[z]]]],
             q -> intersection[FUNS, P[inverse[S]]], s -> composite[z, S],
             t -> composite[S, z], y -> P[fix[z]]}]] // Reverse

Out[38]= or[not[equal[V, domain[z]]], not[FUNCTION[z]],
           not[subclass[z, inverse[S]]], not[subclass[composite[z, S], composite[S, z]]],
           subclass[Uclosure[fix[z]], fix[z]]] == True

In[39]:= (% /. z -> z_) /. Equal -> SetDelayed
```

This can be improved slightly:

```
In[40]:= SubstTest[and, implies[p, subclass[u, v]], subclass[v, u],
             {p -> and[FUNCTION[z], equal[V, domain[z]], subclass[z, inverse[S]],
              subcommute[z, S]], u -> Uclosure[fix[z]], v -> fix[z]}] // Reverse

Out[40]= or[equal[fix[z], Uclosure[fix[z]]], not[equal[V, domain[z]]], not[FUNCTION[z]],
           not[subclass[z, inverse[S]]], not[subclass[composite[z, S], composite[S, z]]]] == True

In[41]:= (% /. z -> z_) /. Equal -> SetDelayed
```

a lemma about idempotent functions

```
In[42]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
           implies[and[p2, p3], p5], implies[and[p4, p5], p6],
           not[implies[and[p1, p2, p3], p6]], {p1 -> member[x, domain[z]],
            p2 -> FUNCTION[z], p3 -> idempotent[z], p4 -> member[APPLY[z, x], range[z]],
            p5 -> equal[range[z], fix[z]], p6 -> member[APPLY[z, x], fix[z]]}]]

Out[42]= or[member[APPLY[z, x], fix[z]], not[equal[z, composite[z, z]]],
           not[FUNCTION[z]], not[member[x, domain[z]]]] == True

In[43]:= or[member[APPLY[z_, x_], fix[z_]], not[equal[z_, composite[z_, z_]]],
           not[FUNCTION[z_]], not[member[x_, domain[z_]]]] := True
```

Restatement:

```
In[44]:= implies[and[FUNCTION[z], idempotent[z], member[x, domain[z]]],
          member[APPLY[z, x], fix[z]]]

Out[44]= True
```

APPLY and CORE

Lemma.

```
In[45]:= SubstTest[implies, member[z, V], member[intersection[x, z], V], z -> P[y]]

Out[45]= or[member[intersection[x, P[y]], V], not[member[y, V]]] == True

In[46]:= or[member[intersection[x_, P[y_]], V], not[member[y_, V]]] := True
```

Note:

```
In[47]:= equal[union[complement[image[V, singleton[y]]],
           complement[image[V, singleton[intersection[x, P[y]]]]]],
          complement[image[V, singleton[y]]]]

Out[47]= True

In[48]:= union[complement[image[V, singleton[y_]]],
           complement[image[V, singleton[intersection[x_, P[y_]]]]]] :=
          complement[image[V, singleton[y]]]

In[49]:= SubstTest[A, image[w, singleton[y]], w -> CORE[x]] // Reverse

Out[49]= APPLY[CORE[x], y] == union[complement[image[V, singleton[y]]], U[intersection[x, P[y]]]]

In[50]:= APPLY[CORE[x_], y_] :=
          union[complement[image[V, singleton[y]]], U[intersection[x, P[y]]]]
```

Restatement:

```
In[51]:= equal[APPLY[CORE[x], y], union[complement[image[V, singleton[y]]], core[x, y]]]

Out[51]= True
```

Corollary:

```
In[52]:= implies[member[y, z], equal[APPLY[CORE[x], y], core[x, y]]]

Out[52]= True
```

Corollary of the result about idempotent functions.

```
In[53]:= SubstTest[implies, and[FUNCTION[z], idempotent[z], member[y, domain[z]]],
           member[APPLY[z, y], fix[z]], z -> CORE[x]]

Out[53]= or[member[U[intersection[x, P[y]]], Uclosure[x]], not[member[y, V]]] == True

In[54]:= or[member[U[intersection[x_, P[y_]]], Uclosure[x_]], not[member[y_, V]]] := True
```

Generally speaking, it is better to express statements in terms of **core[x,y]** than in terms of **APPLY[CORE[x],y]**.

the final steps

Previously proven facts are combined using double negation:

```
In[55]:= implies[and[member[y, domain[x]], FUNCTION[x], idempotent[x], subclass[x, inverse[S]]],
           member[APPLY[x, y], intersection[fix[x], P[y]]]] // NotNotTest

Out[55]= or[and[member[APPLY[x, y], fix[x]], subclass[APPLY[x, y], y]],
           not[equal[x, composite[x, x]]], not[FUNCTION[x]],
           not[member[y, domain[x]]], not[subclass[x, inverse[S]]]] == True

In[56]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[57]:= SubstTest[implies, member[u, v], subclass[u, U[v]], v -> intersection[x, P[y]]]

Out[57]= or[not[member[u, x]], not[subclass[u, y]], subclass[u, U[intersection[x, P[y]]]]] == True

In[58]:= or[not[member[u_, x_]], not[subclass[u_, y_]],
           subclass[u_, U[intersection[x_, P[y_]]]]] := True
```

Restatement:

```
In[59]:= implies[and[member[u, x], subclass[u, y]], subclass[u, core[x, y]]]

Out[59]= True

In[60]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
             {p1 ->
               and[member[y, domain[x]], FUNCTION[x], idempotent[x], subclass[x, inverse[S]]],
                 p2 -> member[APPLY[x, y], intersection[fix[x], P[y]]],
                 p3 -> subclass[APPLY[x, y], core[fix[x], y]]}]]

Out[60]= or[not[equal[x, composite[x, x]]], not[FUNCTION[x]],
           not[member[y, domain[x]]], not[subclass[x, inverse[S]]],
           subclass[APPLY[x, y], U[intersection[fix[x], P[y]]]]] == True

In[61]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[62]:= implies[and[member[y, domain[x]], FUNCTION[x], idempotent[x], subclass[x, inverse[S]]],
          subclass[APPLY[x, y], core[fix[x], y]]]

Out[62]= True

In[63]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4], p6],
           implies[p5, p7], implies[and[p6, p7], p8], implies[and[p1, p5, p8], p9],
                 not[implies[and[p1, p2, p3, p4, p5], p9]],
                 {p1 -> FUNCTION[x], p2 -> equal[V, domain[x]],
            p3 -> subclass[x, inverse[S]], p4 -> subcommute[x, S], p5 -> member[y, V],
                      p6 -> equal[fix[x], Uclosure[fix[x]]],
            p7 -> member[U[intersection[fix[x], P[y]]], Uclosure[fix[x]]],
            p8 -> member[U[intersection[fix[x], P[y]]], fix[x]],
                 p9 -> equal[APPLY[x, U[intersection[fix[x], P[y]]]],
              U[intersection[fix[x], P[y]]]]}]]

Out[63]= or[equal[APPLY[x, U[intersection[fix[x], P[y]]]], U[intersection[fix[x], P[y]]]],
           not[equal[V, domain[x]]], not[FUNCTION[x]], not[member[y, V]],
           not[subclass[x, inverse[S]]], not[subclass[composite[x, S], composite[S, x]]]] == True

In[64]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[65]:= SubstTest[implies, and[FUNCTION[x], subcommute[x, S], subclass[z, y]],
          subclass[APPLY[x, z], APPLY[x, y]], z -> core[fix[x], y]]

Out[65]= or[not[FUNCTION[x]], not[subclass[composite[x, S], composite[S, x]]],
           subclass[APPLY[x, U[intersection[fix[x], P[y]]]], APPLY[x, y]]] == True
```

```
In[66]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[67]:= Map[not, SubstTest[and, implies[and[p4, p6], p7],
           implies[and[p1, p3, p5, p7], p8], not[implies[and[p1, p2, p3, p4, p5, p6], p8]],
           {p1 -> FUNCTION[x], p2 -> subcommute[x, S], p3 -> idempotent[x],
            p4 -> equal[V, domain[x]], p5 -> subclass[x, inverse[S]], p6 -> member[y, V],
                  p7 -> member[y, domain[x]], p8 -> subclass[APPLY[x, y], core[fix[x], y]]}]]

Out[67]= or[not[equal[V, domain[x]]], not[equal[x, composite[x, x]]],
           not[FUNCTION[x]], not[member[y, V]], not[subclass[x, inverse[S]]],
           not[subclass[composite[x, S], composite[S, x]]],
           subclass[APPLY[x, y], U[intersection[fix[x], P[y]]]]] == True

In[68]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

These facts now need to be combined:

```
In[69]:= {implies[and[p4, p6], p10], implies[and[p1, p2, p4, p5, p6], p7],
           implies[and[p1, p2], p8], implies[and[p7, p8], p9],
             implies[and[p1, p3, p5, p10], p11], implies[and[p9, p11], p12]} /.
          {p1 -> FUNCTION[x], p2 -> subcommute[x, S], p3 -> idempotent[x],
           p4 -> equal[V, domain[x]], p5 -> subclass[x, inverse[S]], p6 -> member[y, V],
           p7 -> equal[APPLY[x, core[fix[x], y]], core[fix[x], y]],
                 p8 -> subclass[APPLY[x, core[fix[x], y]], APPLY[x, y]],
           p9 -> subclass[core[fix[x], y], APPLY[x, y]],
                 p10 -> member[y, domain[x]], p11 -> subclass[APPLY[x, y], core[fix[x], y]],
           p12 -> equal[APPLY[x, y], core[fix[x], y]]}

Out[69]= {True, True, True, True, True, True}
```

This goes faster if the hypotheses are gathered.

```
In[70]:= Map[not, SubstTest[and, implies[p1, p10],
           implies[p1, p7], implies[p1, p8], implies[and[p7, p8], p9],
             implies[and[p1, p10], p11], implies[and[p9, p11], p12],
           not[implies[p1, p12]], {p1 -> and[FUNCTION[x], subcommute[x, S],
              idempotent[x], equal[V, domain[x]], subclass[x, inverse[S]], member[y, V]],
            p7 -> equal[APPLY[x, core[fix[x], y]], core[fix[x], y]],
                  p8 -> subclass[APPLY[x, core[fix[x], y]], APPLY[x, y]],
            p9 -> subclass[core[fix[x], y], APPLY[x, y]],
                  p10 -> member[y, domain[x]], p11 -> subclass[APPLY[x, y], core[fix[x], y]],
            p12 -> equal[APPLY[x, y], core[fix[x], y]]}]]

Out[70]= or[equal[APPLY[x, y], U[intersection[fix[x], P[y]]]], not[equal[V, domain[x]]],
           not[equal[x, composite[x, x]]], not[FUNCTION[x]], not[member[y, V]],
           not[subclass[x, inverse[S]]], not[subclass[composite[x, S], composite[S, x]]]] == True

In[71]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

The final step is to eliminate the variable **y**. For this it is best to get rid of **APPLY**.

```
In[72]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 -> FUNCTION[x], p2 -> equal[APPLY[x, y], core[z, y]],
            p3 -> equal[image[x, singleton[y]], singleton[APPLY[x, y]]],
                  p4 -> equal[singleton[APPLY[x, y]], singleton[core[z, y]]],
            p5 -> equal[image[x, singleton[y]], singleton[core[z, y]]]}]]

Out[72]= or[equal[image[x, singleton[y]], singleton[U[intersection[z, P[y]]]]],
           not[equal[APPLY[x, y], U[intersection[z, P[y]]]]], not[FUNCTION[x]]] == True

In[73]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

```
In[74]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 -> FUNCTION[x], p2 -> and[subcommute[x, S], idempotent[x],
              equal[V, domain[x]], subclass[x, inverse[S]], member[y, V]],
            p3 -> equal[APPLY[x, y], U[intersection[fix[x], P[y]]]],
            p4 -> equal[image[x, singleton[y]], singleton[U[intersection[fix[x], P[y]]]]]}]]

Out[74]= or[equal[image[x, singleton[y]], singleton[U[intersection[fix[x], P[y]]]]],
           not[equal[V, domain[x]]], not[equal[x, composite[x, x]]],
           not[FUNCTION[x]], not[member[y, V]], not[subclass[x, inverse[S]]],
           not[subclass[composite[x, S], composite[S, x]]] == True

In[75]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[76]:= (implies[and[subclass[p, q], subclass[s, t], equal[x, v], member[y, V]],
            equal[image[x, singleton[y]], image[w, singleton[y]]]]) /.
          {p -> union[P[x], image[V, complement[domain[x]]]],
           q -> intersection[FUNS, P[inverse[S]]], s -> composite[x, S],
                t -> composite[S, x], v -> composite[x, x], w -> CORE[fix[x]]}

Out[76]= True

In[77]:= Map[or[subclass[composite[Id, x], CORE[fix[x]]], equal[V, #]] &, SubstTest[class,
            y, implies[and[subclass[p, q], subclass[s, t], equal[x, v], member[y, V]],
             equal[image[x, singleton[y]], image[w, singleton[y]]]],
            {p -> union[P[x], image[V, complement[domain[x]]]],
             q -> intersection[FUNS, P[inverse[S]]], s -> composite[x, S],
                  t -> composite[S, x], v -> composite[x, x], w -> CORE[fix[x]]}]] // Reverse

Out[77]= or[not[equal[V, domain[x]]], not[equal[x, composite[x, x]]], not[FUNCTION[x]],
           not[subclass[x, inverse[S]]], not[subclass[composite[x, S], composite[S, x]]],
           subclass[composite[Id, x], CORE[fix[x]]]] == True

In[78]:= (% /. x -> x_) /. Equal -> SetDelayed

In[79]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[composite[v, id[domain[u]]], u], {u -> composite[Id, x], v -> CORE[y]}]

Out[79]= or[equal[composite[Id, x], composite[CORE[y], id[domain[x]]]],
           not[subclass[composite[Id, x], CORE[y]]]] == True

In[80]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[81]:= Map[or[not[equal[V, y]], #] &, SubstTest[implies, and[equal[u, v], equal[v, w]],
            equal[u, w], {u -> CORE[x], v -> composite[CORE[x], id[y]]}]] // MapNotNot

Out[81]= or[equal[w, CORE[x]], not[equal[V, y]], not[equal[w, composite[CORE[x], id[y]]]]] == True

In[82]:= or[equal[w_, CORE[x_]], not[equal[V, y_]],
           not[equal[w_, composite[CORE[x_], id[y_]]]]] := True

In[83]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w], v -> composite[Id, u]]

Out[83]= or[equal[u, w], not[equal[w, composite[Id, u]]], not[subclass[u, cart[V, V]]]] == True

In[84]:= (% /. {u -> u_, v -> v_, w -> w_}) /. Equal -> SetDelayed
```

```
In[85]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           implies[p1, p5], implies[and[p1, p3], p4], implies[and[p4, p5], p6],
           not[implies[p1, p6]], {p1 -> and[FUNCTION[x], equal[V, domain[x]],
              idempotent[x], subclass[x, inverse[S]], subcommute[x, S]],
            p2 -> subclass[composite[Id, x], CORE[fix[x]]],
           p3 -> equal[composite[Id, x], composite[CORE[fix[x]], id[domain[x]]]],
            p4 -> equal[composite[Id, x], CORE[fix[x]]], p5 -> subclass[x, cart[V, V]],
            p6 -> equal[x, CORE[fix[x]]]}]]

Out[85]= or[equal[x, CORE[fix[x]]], not[equal[V, domain[x]]],
           not[equal[x, composite[x, x]]], not[FUNCTION[x]], not[subclass[x, inverse[S]]],
           not[subclass[composite[x, S], composite[S, x]]]] == True
```

This is the main result of this notebook.

```
In[86]:= or[equal[CORE[fix[x_]], x_], not[equal[V, domain[x_]]],
           not[equal[composite[x_, x_], x_]], not[FUNCTION[x_]],
           not[subclass[composite[x_, S], composite[S, x_]]],
           not[subclass[x_, inverse[S]]]] := True
```

Restatement:

```
In[87]:= implies[hypotheses[x], equal[x, CORE[fix[x]]]]

Out[87]= True
```