characterization of HULL[x]

Johan G. F. Belinfante
2003 November 3

```
In[1]:= << goedel52.t10; << tools.m

        :Package Title: goedel52.t10      2003 November 1 at 5:30 a.m.

        It is now:  2003 Nov 3 at 9:42

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 October 28

        weightlimit = 40
```

summary

The results in this notebook were inspired by Kuratowski's characterization of closure operators, but what is derived is more general. It is shown that if an idempotent function **x** is contained in and subcommutes with the subset relation **S**, then **x** is equal to **HULL[fix[x]]**. The following temporary abbreviations will be used:

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

```
In[3]:= hypotheses[x_] := and[FUNCTION[x], subclass[x, S], subcommute[x, S], idempotent[x]]
```

These hypotheses are satisfied by any **HULL[x]**:

```
In[4]:= hypotheses[HULL[x]]

Out[4]= True
```

The goal is to show that conversely, these hypotheses characterize the functions **HULL[x]**.

normalization for the hypotheses

The conditions in the hypotheses will be assert−normalized to prevent them from being rewritten over and over again.

```
In[5]:= and[FUNCTION[x], subclass[x, S]] // AssertTest // Reverse

Out[5]= and[FUNCTION[x], subclass[composite[Id, x], S]] == and[FUNCTION[x], subclass[x, S]]
```

```
In[6]:= and[FUNCTION[x_], subclass[composite[Id, x_], S]] := and[FUNCTION[x], subclass[x, S]]
```

The same goes for the negation of the hypotheses.

```
In[7]:= or[not[FUNCTION[x]], not[subclass[composite[Id, x], S]]] // NotNotTest

Out[7]= or[not[FUNCTION[x]], not[subclass[composite[Id, x], S]]] ==
         or[not[FUNCTION[x]], not[subclass[x, S]]]

In[8]:= or[not[FUNCTION[x_]], not[subclass[composite[Id, x_], S]]] :=
         or[not[FUNCTION[x]], not[subclass[x, S]]]
```

Testing this takes only a few seconds with the **simplify** flag off, but 240 times longer with the flag on.

```
In[9]:= simplify = False;

In[10]:= hypotheses[x] // AssertTest

Out[10]= True

In[11]:= not[hypotheses[x]] // AssertTest

Out[11]= True
```

domain considerations

In this section it is shown that the hyptheses imply that **x** has the same domain as **HULL[fix[x]]**.

```
In[12]:= SubstTest[implies, subclass[x, y],
          subclass[image[inverse[x], z], image[inverse[y], z]], {y -> S, z -> range[x]}]

Out[12]= or[not[subclass[x, S]], subclass[domain[x], image[inverse[S], range[x]]]] == True

In[13]:= or[not[subclass[x_, S]], subclass[domain[x_], image[inverse[S], range[x_]]]] := True

In[14]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> fix[x], v -> domain[x], w -> image[inverse[S], domain[x]]}]

Out[14]= subclass[fix[x], image[inverse[S], domain[x]]] == True

In[15]:= subclass[fix[x_], image[inverse[S], domain[x_]]] := True

In[16]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[inverse[S], x], v -> image[inverse[S], y]}]

Out[16]= and[subclass[x, image[inverse[S], y]], subclass[y, image[inverse[S], x]]] ==
          equal[image[inverse[S], x], image[inverse[S], y]]

In[17]:= and[subclass[x_, image[inverse[S], y_]], subclass[y_, image[inverse[S], x_]]] :=
          equal[image[inverse[S], x], image[inverse[S], y]]

In[18]:= Map[implies[subclass[domain[x], image[inverse[S], fix[x]]], #] &,
           SubstTest[and, subclass[u, image[inverse[S], v]],
            subclass[v, image[inverse[S], u]], {u -> fix[x], v -> domain[x]}]] // Reverse

Out[18]= or[equal[image[inverse[S], domain[x]], image[inverse[S], fix[x]]],
           not[subclass[domain[x], image[inverse[S], fix[x]]]]] == True

In[19]:= (% /. x -> x_) /. Equal -> SetDelayed
```

This took a long time when **simplify** was on, but not with the flag turned off.

```
In[20]:= Map[not, SubstTest[and, implies[and[p1, p4], p5], implies[p2, p6],
           implies[p5, p7], implies[and[p6, p7], p8], not[implies[and[p1, p2, p4], p8]],
                {p1 -> FUNCTION[x], p2 -> subclass[x, S],
            p4 -> idempotent[x], p5 -> equal[fix[x], range[x]],
                p6 -> subclass[domain[x], image[inverse[S], range[x]]],
            p7 -> equal[image[inverse[S], fix[x]], image[inverse[S], range[x]]],
                p8 -> subclass[domain[x], image[inverse[S], fix[x]]],
            p9 -> equal[image[inverse[S], domain[x]], image[inverse[S], fix[x]]],
                p10 -> equal[image[inverse[S], domain[x]], domain[x]],
            p11 -> equal[domain[x], image[inverse[S], fix[x]]]}]]

Out[20]= or[not[equal[x, composite[x, x]]], not[FUNCTION[x]],
           not[subclass[x, S]], subclass[domain[x], image[inverse[S], fix[x]]]] == True

In[21]:= or[not[equal[x_, composite[x_, x_]]], not[FUNCTION[x_]],
           not[subclass[x_, S]], subclass[domain[x_], image[inverse[S], fix[x_]]]] := True

In[22]:= Map[not,
          SubstTest[and, implies[and[p1, p2, p4], p8], implies[p8, p9], implies[p3, p10],
             implies[and[p9, p10], p11], not[implies[and[p1, p2, p3, p4], p11]],
                {p1 -> FUNCTION[x], p2 -> subclass[x, S],
            p3 -> subcommute[x, S], p4 -> idempotent[x], p5 -> equal[fix[x], range[x]],
                p6 -> subclass[domain[x], image[inverse[S], range[x]]],
            p7 -> equal[image[inverse[S], fix[x]], image[inverse[S], range[x]]],
                p8 -> subclass[domain[x], image[inverse[S], fix[x]]],
            p9 -> equal[image[inverse[S], domain[x]], image[inverse[S], fix[x]]],
                p10 -> equal[image[inverse[S], domain[x]], domain[x]],
            p11 -> equal[domain[x], image[inverse[S], fix[x]]]}]]

Out[22]= or[equal[domain[x], image[inverse[S], fix[x]]],
           not[equal[x, composite[x, x]]], not[FUNCTION[x]], not[subclass[x, S]],
           not[subclass[composite[x, S], composite[S, x]]]] == True

In[23]:= or[equal[domain[x_], image[inverse[S], fix[x_]]],
           not[equal[x_, composite[x_, x_]]], not[FUNCTION[x_]], not[subclass[x_, S]],
           not[subclass[composite[x_, S], composite[S, x_]]]] := True

In[24]:= implies[hypotheses[x], equal[domain[x], domain[HULL[fix[x]]]]]

Out[24]= True
```

monotonicity for functions with hereditary domains

Monotone functions with hereditary domains subcommute with the subclass relation **S**. In this sectiion, the monotonicity property for such functions is expressed in terms of function application.

```
In[25]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
          {u -> composite[x, S], v -> composite[S, x], w -> complement[inverse[E]]}]

Out[25]= or[not[subclass[composite[x, S], composite[S, x]]],
           subclass[composite[complement[inverse[E]], x, S],
            composite[complement[inverse[E]], x]]] == True

In[26]:= (% /. x -> x_) /. Equal -> SetDelayed

In[27]:= SubstTest[implies, subclass[u, v],
          subclass[composite[w, u], composite[w, v]], {u -> Id, v -> S, w -> composite[x, y]}]

Out[27]= subclass[composite[x, y], composite[x, y, S]] == True

In[28]:= subclass[composite[x_, y_], composite[x_, y_, S]] := True
```

```
In[29]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p -> subcommute[x, S], u -> composite[complement[inverse[E]], x, S],
            v -> composite[complement[inverse[E]], x]}] // Reverse

Out[29]= or[equal[composite[complement[inverse[E]], x],
           composite[complement[inverse[E]], x, S]],
          not[subclass[composite[x, S], composite[S, x]]]] == True

In[30]:= (% /. x -> x_) /. Equal -> SetDelayed

In[31]:= SubstTest[implies, equal[u, v], equal[image[u, w], image[v, w]],
           {u -> composite[complement[inverse[E]], x],
            v -> composite[complement[inverse[E]], x, S], w -> singleton[y]}]

Out[31]= or[equal[A[image[x, image[S, singleton[y]]]], APPLY[x, y]],
          not[equal[composite[complement[inverse[E]], x],
            composite[complement[inverse[E]], x, S]]]] == True

In[32]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[33]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
             {p1 -> subcommute[x, S], p2 -> equal[
              composite[complement[inverse[E]], x], composite[complement[inverse[E]], x, S]],
             p3 -> equal[A[image[x, image[S, singleton[y]]]], A[image[x, singleton[y]]]]}]]

Out[33]= or[equal[A[image[x, image[S, singleton[y]]]], APPLY[x, y]],
          not[subclass[composite[x, S], composite[S, x]]]] == True

In[34]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[35]:= Map[or[not[member[y, V]], #] &, SubstTest[implies,
           and[subclass[u, v], subclass[t, w]], subclass[composite[t, u], composite[w, v]],
            {u -> singleton[PAIR[x, y]], v -> S, t -> composite[complement[inverse[E]], z, S],
             w -> composite[complement[inverse[E]], z]}]] // MapNotNot

Out[35]= or[not[member[y, V]], not[subclass[x, y]],
          not[subclass[composite[complement[inverse[E]], z, S],
            composite[complement[inverse[E]], z]]], subclass[
           A[image[z, image[S, singleton[x]]]], A[image[z, image[S, singleton[y]]]]]] == True

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

In[37]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2, p3], p5]],
             {p1 -> equal[w, x], p2 -> subclass[x, y], p3 -> equal[y, z],
              p4 -> subclass[w, y], p5 -> subclass[w, z]}]]

Out[37]= or[not[equal[w, x]], not[equal[y, z]], not[subclass[x, y]], subclass[w, z]] == True

In[38]:= or[not[equal[w_, x_]], not[equal[y_, z_]],
           not[subclass[x_, y_]], subclass[w_, z_]] := True
```

The next step takes 12 seconds even with **simplify** turned off, and it does not help to turn off the **cond** flag.

```
In[39]:= Map[not, SubstTest[and, implies[p1, p4], implies[and[p2, p3, p4], p5], implies[p1, p6],
           implies[p1, p7], implies[and[p5, p6, p7], p8], not[implies[and[p1, p2, p3], p8]],
           {p1 -> subcommute[z, S], p2 -> subclass[x, y], p3 -> member[y, V], p4 -> subclass[
              composite[complement[inverse[E]], z, S], composite[complement[inverse[E]], z]],
                 p5 -> subclass[A[image[z, image[S, singleton[x]]]],
              A[image[z, image[S, singleton[y]]]]],
            p6 -> equal[A[image[z, image[S, singleton[x]]]], A[image[z, singleton[x]]],
                 p7 -> equal[A[image[z, image[S, singleton[y]]]], A[image[z, singleton[y]]]],
                 p8 -> subclass[APPLY[z, x], APPLY[z, y]]}]]

Out[39]= or[not[member[y, V]], not[subclass[x, y]],
           not[subclass[composite[z, S], composite[S, z]]],
           subclass[APPLY[z, x], APPLY[z, y]]] == True

In[40]:= or[not[member[y_, V]], not[subclass[x_, y_]],
           not[subclass[composite[z_, S], composite[S, z_]]],
           subclass[APPLY[z_, x_], APPLY[z_, y_]]] := True

In[41]:= SubstTest[implies, equal[V, z], subclass[x, z], z -> A[y]]

Out[41]= or[not[equal[0, y]], subclass[x, A[y]]] == True

In[42]:= or[not[equal[0, y_]], subclass[x_, A[y_]]] := True

In[43]:= SubstTest[implies, equal[0, w], subclass[z, A[w]], w -> image[x, singleton[y]]]

Out[43]= or[member[y, domain[x]], subclass[z, APPLY[x, y]]] == True

In[44]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed

In[45]:= Map[not, SubstTest[and, implies[p, q],
           implies[q, r], not[implies[p, r]], {p -> not[member[y, V]],
            q -> not[member[y, domain[z]]], r -> subclass[x, APPLY[z, y]]}]]

Out[45]= or[member[y, V], subclass[x, APPLY[z, y]]] == True

In[46]:= or[member[y_, V], subclass[x_, APPLY[z_, y_]]] := True

In[47]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4], implies[not[p3], p4],
           not[implies[and[p1, p2], p4]], {p1 -> subcommute[z, S], p2 -> subclass[x, y],
            p3 -> member[y, V], p4 -> subclass[APPLY[z, x], APPLY[z, y]]}]]

Out[47]= or[not[subclass[x, y]], not[subclass[composite[z, S], composite[S, z]]],
           subclass[APPLY[z, x], APPLY[z, y]]] == True

In[48]:= or[not[subclass[x_, y_]], not[subclass[composite[z_, S], composite[S, z_]]],
           subclass[APPLY[z_, x_], APPLY[z_, y_]]] := True
```

Restatement:

```
In[49]:= implies[and[subclass[x, y], subcommute[z, S]], subclass[APPLY[z, x], APPLY[z, y]]]

Out[49]= True
```

fixed points of functions

Lemma.

```
In[50]:= or[and[equal[y, APPLY[x, y]], member[y, V]],
            not[equal[0, y]], not[equal[0, APPLY[x, y]]]] // NotNotTest

Out[50]= or[and[equal[y, APPLY[x, y]], member[y, V]],
            not[equal[0, y]], not[equal[0, APPLY[x, y]]]] == True

In[51]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Note that:

```
In[52]:= equiv[or[and[equal[0, y], equal[0, APPLY[x, y]]],
             and[equal[y, APPLY[x, y]], member[y, V]]], and[equal[y, APPLY[x, y]], member[y, V]]]

Out[52]= True
```

This fact is added as a rewrite rule:

```
In[53]:= or[and[equal[0, y_], equal[0, APPLY[x_, y_]]],
             and[equal[y_, APPLY[x_, y_]], member[y_, V]]] :=
           and[equal[y, APPLY[x, y]], member[y, V]]

In[54]:= member[y, fix[VERTSECT[complement[composite[complement[inverse[E]], z]]]]] //
           AssertTest

Out[54]= member[y, fix[VERTSECT[complement[composite[complement[inverse[E]], z]]]]] ==
           and[equal[y, APPLY[z, y]], member[y, V]]

In[55]:= member[y_, fix[VERTSECT[complement[composite[complement[inverse[E]], z_]]]]] :=
           and[equal[y, APPLY[z, y]], member[y, V]]
```

From this one can derive a rule that prevents the appearance of **pair[y,y]**.

```
In[56]:= SubstTest[member, y,
           fix[VERTSECT[complement[composite[complement[inverse[E]], z]]]], z -> funpart[x]]

Out[56]= member[y, fix[funpart[x]]] == and[equal[y, APPLY[funpart[x], y]], member[y, V]]

In[57]:= member[y_, fix[funpart[x_]]] := and[equal[y, APPLY[funpart[x], y]], member[y, V]]
```

Lemma.

```
In[58]:= SubstTest[implies, equal[x, y], equal[fix[x], fix[y]], y -> funpart[x]]

Out[58]= or[equal[fix[x], fix[funpart[x]]], not[FUNCTION[x]]] == True

In[59]:= or[equal[fix[x_], fix[funpart[x_]]], not[FUNCTION[x_]]] := True

In[60]:= SubstTest[implies, and[member[x, u], equal[u, v]],
           member[x, v], {u -> fix[funpart[y]], v -> fix[y]}]

Out[60]= or[member[x, fix[y]], not[equal[x, APPLY[funpart[y], x]]],
            not[equal[fix[y], fix[funpart[y]]]], not[member[x, V]]] == True

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

```
In[62]:= Map[not, SubstTest[and, implies[p1, p4], implies[and[p3, p4], p5], implies[p1, p6],
           implies[and[p2, p5, p6], p7], not[implies[and[p1, p2, p3], p7]],
             {p1 -> FUNCTION[x], p2 -> member[y, V], p3 -> equal[y, APPLY[x, y]],
            p4 -> equal[APPLY[x, y], APPLY[funpart[x], y]],
                 p5 -> equal[y, APPLY[funpart[x], y]],
            p6 -> equal[fix[x], fix[funpart[x]]], p7 -> member[y, fix[x]]}]]

Out[62]= or[member[y, fix[x]], not[equal[y, APPLY[x, y]]],
           not[FUNCTION[x]], not[member[y, V]]] == True

In[63]:= or[member[y_, fix[x_]], not[equal[y_, APPLY[x_, y_]]],
           not[FUNCTION[x_]], not[member[y_, V]]] := True
```

Corollary:

```
In[64]:= SubstTest[implies, and[member[z, V], FUNCTION[x], equal[z, APPLY[x, z]]],
          member[z, fix[x]], z -> A[y]]

Out[64]= or[equal[0, y], member[A[y], fix[x]],
           not[equal[A[y], APPLY[x, A[y]]]], not[FUNCTION[x]]] == True

In[65]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[66]:= SubstTest[implies, and[member[u, v], equal[v, w]],
           member[u, w], {u -> y, v -> fix[x], w -> fix[funpart[x]]}] // MapNotNot

Out[66]= or[equal[y, APPLY[funpart[x], y]],
           not[equal[fix[x], fix[funpart[x]]]], not[member[y, fix[x]]]] == True

In[67]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[68]:= Map[not, SubstTest[and, implies[p1, p4],
           implies[and[p2, p4], p5], implies[p1, p6], implies[and[p5, p6], p3],
           not[implies[and[p1, p2], p3]], {p1 -> FUNCTION[x], p2 -> member[y, fix[x]],
            p3 -> equal[y, APPLY[x, y]], p4 -> equal[fix[x], fix[funpart[x]]],
                 p5 -> equal[y, APPLY[funpart[x], y]],
            p6 -> equal[APPLY[funpart[x], y], APPLY[x, y]]}]]

Out[68]= or[equal[y, APPLY[x, y]], not[FUNCTION[x]], not[member[y, fix[x]]]] == True

In[69]:= or[equal[y_, APPLY[x_, y_]], not[FUNCTION[x_]], not[member[y_, fix[x_]]]] := True
```

a lemma

In this section, the results of the preceding section are applied to the case of **HULL[x]**. The following lemma takes only 8 seconds with the **simplify** flag turned off, but ten times longer with that flag on.

```
In[70]:= Map[equal[V, class[w, not[#]]] &, SubstTest[and, implies[and[p0, p1], p6],
           implies[p1, p2], implies[and[p2, p3], p4], implies[and[p5, p6], p7],
           implies[and[p4, p7], p8], not[implies[and[p0, p1, p3, p5], p8]],
             {p0 -> subclass[z, fix[x]], p1 -> member[w, z], p2 -> subclass[A[z], w],
            p3 -> subcommute[x, S], p4 -> subclass[APPLY[x, A[z]], APPLY[x, w]],
                 p5 -> FUNCTION[x], p6 -> member[w, fix[x]],
            p7 -> equal[APPLY[x, w], w], p8 -> subclass[APPLY[x, A[z]], w]}]]

Out[70]= or[not[FUNCTION[x]], not[subclass[z, fix[x]]],
           not[subclass[composite[x, S], composite[S, x]]],
           subclass[APPLY[x, A[z]], A[z]]] == True
```

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

This will now be combined with the results of the preceding section.

```
In[72]:= Map[not,
          SubstTest[and, implies[and[p1, p2, p3], p5], implies[p4, p6], implies[and[p5, p6], p7],
           implies[and[p1, p7], p8], not[implies[and[p1, p2, p3, p4], p8]],
          {p1 -> FUNCTION[x], p2 -> subclass[z, fix[x]],
             p3 -> subcommute[x, S], p4 -> subclass[x, S],
              p5 -> subclass[A[image[x, singleton[A[z]]]], A[z]],
             p6 -> subclass[A[z], A[image[x, singleton[A[z]]]]],
             p7 -> equal[A[image[x, singleton[A[z]]]], A[z]],
                   p8 -> or[equal[0, z], member[A[z], fix[x]]]}]]

Out[72]= or[equal[0, z], member[A[z], fix[x]], not[FUNCTION[x]], not[subclass[x, S]],
           not[subclass[z, fix[x]]], not[subclass[composite[x, S], composite[S, x]]]] == True

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

The next step also took a long time when **simplify** was on..

```
In[74]:= Map[assert[forall[y, #]] &,
          SubstTest[implies, and[FUNCTION[x], subclass[x, S], subcommute[x, S],
            subclass[z, fix[x]], not[equal[0, z]]], member[A[z], fix[x]],
             z -> intersection[fix[x], image[S, singleton[y]]]]]

Out[74]= or[not[FUNCTION[x]], not[subclass[x, S]],
           not[subclass[composite[x, S], composite[S, x]]],
           subclass[image[inverse[S], fix[x]], image[inverse[HULL[fix[x]]], fix[x]]]] == True

In[75]:= (% /. x -> x_) /. Equal -> SetDelayed

In[76]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
             {u -> image[inverse[S], fix[x]],
           v -> image[inverse[HULL[fix[x]]], fix[x]], w -> HULL[fix[x]]}]

Out[76]= or[not[subclass[image[inverse[S], fix[x]], image[inverse[HULL[fix[x]]], fix[x]]]],
           subclass[fix[HULL[fix[x]]], fix[x]]] == True

In[77]:= (% /. x -> x_) /. Equal -> SetDelayed

In[78]:= or[equal[fix[x], fix[HULL[fix[x]]]],
           not[subclass[fix[HULL[fix[x]]], fix[x]]]] // AssertTest

Out[78]= or[equal[fix[x], fix[HULL[fix[x]]]], not[subclass[fix[HULL[fix[x]]], fix[x]]]] == True

In[79]:= (% /. x -> x_) /. Equal -> SetDelayed

In[80]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
           implies[p4, p5], implies[p5, p6], not[implies[and[p1, p2, p3], p6]],
             {p1 -> FUNCTION[x], p2 -> subclass[x, S], p3 -> subcommute[x, S],
            p4 -> subclass[image[inverse[S], fix[x]], image[inverse[HULL[fix[x]]], fix[x]]],
            p5 -> subclass[fix[HULL[fix[x]]], fix[x]],
                  p6 -> equal[fix[HULL[fix[x]]], fix[x]]}]]

Out[80]= or[equal[fix[x], fix[HULL[fix[x]]]], not[FUNCTION[x]],
           not[subclass[x, S]], not[subclass[composite[x, S], composite[S, x]]]] == True

In[81]:= or[equal[fix[x_], fix[HULL[fix[x_]]]], not[FUNCTION[x_]],
           not[subclass[x_, S]], not[subclass[composite[x_, S], composite[S, x_]]]] := True
```

some facts about HULL[x]

An application rule for **HULL[x]** is derived:

```
In[82]:= SubstTest[A, image[w, singleton[y]], w -> HULL[x]] // Reverse

Out[82]= APPLY[HULL[x], y] == A[intersection[x, image[S, singleton[y]]]]

In[83]:= APPLY[HULL[x_], y_] := A[intersection[x, image[S, singleton[y]]]]

In[84]:= SubstTest[implies, subclass[w, z], subclass[APPLY[z, y], APPLY[w, y]],
          {w -> composite[Id, x], z -> composite[id[range[x]], S]}]

Out[84]= or[not[subclass[composite[Id, x], S]],
           subclass[A[intersection[image[S, singleton[y]], range[x]]], APPLY[x, y]]] == True

In[85]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[86]:= SubstTest[implies, subclass[u, v], subclass[A[v], A[u]],
             {u -> image[x, singleton[y]], v -> intersection[image[S, singleton[y]], range[x]]}]

Out[86]= or[not[subclass[y, APPLY[x, y]]],
           subclass[A[intersection[image[S, singleton[y]], range[x]]], APPLY[x, y]]] == True

In[87]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[88]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
             {u -> composite[Id, x], v -> S, w -> singleton[y]}]

Out[88]= or[not[subclass[composite[Id, x], S]], subclass[y, APPLY[x, y]]] == True

In[89]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[90]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
             {p1 -> subclass[x, S], p2 -> subclass[composite[Id, x], S], p3 -> subclass[
              A[intersection[image[S, singleton[y]], range[x]]], A[image[x, singleton[y]]]]}]]

Out[90]= or[not[subclass[x, S]],
           subclass[A[intersection[image[S, singleton[y]], range[x]]], APPLY[x, y]]] == True

In[91]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[92]:= implies[subclass[x, S], subclass[APPLY[HULL[range[x]], y], APPLY[x, y]]]

Out[92]= True
```

The following is a consequence of monotonicity of **HULL[w]**.

```
In[93]:= SubstTest[implies, and[subclass[y, t], subcommute[x, S]],
          subclass[APPLY[x, y], APPLY[x, t]], t -> APPLY[HULL[w], y]]

Out[93]= or[not[subclass[composite[x, S], composite[S, x]]],
           subclass[APPLY[x, y], APPLY[x, A[intersection[w, image[S, singleton[y]]]]]]] == True

In[94]:= (% /. {w -> w_, x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[95]:= implies[subcommute[x, S], subclass[APPLY[x, y], APPLY[x, APPLY[HULL[w], y]]]]

Out[95]= True
```

a rule about idempotent functions

```
In[96]:= SubstTest[implies, equal[x, z], equal[image[x, w], image[z, w]],
           {z -> composite[x, x], w -> singleton[y]}]

Out[96]= or[equal[image[x, image[x, singleton[y]]], image[x, singleton[y]]],
           not[equal[x, composite[x, x]]]] == True

In[97]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[98]:= SubstTest[implies, equal[u, v], equal[A[u], A[v]],
           {u -> image[x, singleton[y]], v -> image[x, singleton[z]]}]

Out[98]= or[equal[APPLY[x, y], APPLY[x, z]],
           not[equal[image[x, singleton[y]], image[x, singleton[z]]]]] == True

In[99]:= or[equal[APPLY[x_, y_], APPLY[x_, z_]],
           not[equal[image[x_, singleton[y_]], image[x_, singleton[z_]]]]] := True

In[100]:=
        Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p0, p4],
          implies[and[p3, p4], p5], implies[p5, p6], not[implies[and[p0, p1], p6]],
            {p0 -> idempotent[x], p1 -> FUNCTION[x],
           p2 -> equal[image[x, singleton[y]], singleton[APPLY[x, y]]],
           p3 -> equal[image[x, image[x, singleton[y]]], image[x, singleton[APPLY[x, y]]]],
            p4 -> equal[image[x, image[x, singleton[y]]], image[x, singleton[y]]],
           p5 -> equal[image[x, singleton[APPLY[x, y]]], image[x, singleton[y]]],
            p6 -> equal[APPLY[x, APPLY[x, y]], APPLY[x, y]]}]]

Out[100]=
        or[equal[APPLY[x, y], APPLY[x, APPLY[x, y]]],
          not[equal[x, composite[x, x]]], not[FUNCTION[x]]] == True

In[101]:=
        (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[102]:=
        implies[and[FUNCTION[x], idempotent[x]], equal[APPLY[x, APPLY[x, y]], APPLY[x, y]]]

Out[102]=
        True
```

the final steps

Applying **HULL[w]** to an argument in its domain produces a value in its range, which is the same thing as its fixed−point set.

```
In[103]:=
SubstTest[subclass, image[z, u], range[z], {z -> HULL[x], u -> singleton[y]}]

Out[103]=
or[member[A[intersection[x, image[S, singleton[y]]]], fix[HULL[x]]],
  not[member[y, image[inverse[S], x]]]] == True

In[104]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[105]:=
implies[member[y, domain[HULL[w]]], member[APPLY[HULL[w], y], fix[HULL[w]]]]

Out[105]=
True
```

The following is needed:

```
In[106]:=
SubstTest[implies, equal[v, w],
 equal[APPLY[v, z], APPLY[w, z]], {v -> HULL[x], w -> HULL[y]}]

Out[106]=
or[equal[A[intersection[x, image[S, singleton[z]]]],
   A[intersection[y, image[S, singleton[z]]]]], not[equal[HULL[x], HULL[y]]]] == True

In[107]:=
(% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed

In[108]:=
Map[not, SubstTest[and, implies[and[p1, p2], p11],
  implies[p11, p18], implies[p18, p19], implies[p8, p9],
 implies[and[p9, p19], p20], not[implies[and[p1, p2, p8], p20]],
    {p1 -> FUNCTION[x], p2 -> idempotent[x],
  p8 -> subclass[x, S], p9 -> subclass[APPLY[HULL[range[x]], y], APPLY[x, y]],
 p11 -> equal[range[x], fix[x]], p18 -> equal[HULL[fix[x]], HULL[range[x]]],
   p19 -> equal[APPLY[HULL[fix[x]], y], APPLY[HULL[range[x]], y]],
 p20 -> subclass[APPLY[HULL[fix[x]], y], APPLY[x, y]]}]]

Out[108]=
or[not[equal[x, composite[x, x]]], not[FUNCTION[x]], not[subclass[x, S]],
  subclass[A[intersection[fix[x], image[S, singleton[y]]]], APPLY[x, y]]] == True

In[109]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[110]:=
Map[not, SubstTest[and, implies[and[p1, p5, p8], p10],
  implies[p12, p13], implies[and[p10, p13], p14],
  implies[and[p1, p14], p15], not[implies[and[p1, p5, p8, p12], p15]],
 {p1 -> FUNCTION[x], p5 -> subcommute[x, S], p8 -> subclass[x, S],
   p10 -> equal[fix[x], fix[HULL[fix[x]]]], p12 -> member[y, domain[HULL[fix[x]]]],
   p13 -> member[APPLY[HULL[fix[x]], y], fix[HULL[fix[x]]]],
   p14 -> member[APPLY[HULL[fix[x]], y], fix[x]],
   p15 -> equal[APPLY[x, APPLY[HULL[fix[x]], y]], APPLY[HULL[fix[x]], y]]}]]

Out[110]=
or[equal[A[intersection[fix[x], image[S, singleton[y]]]],
   APPLY[x, A[intersection[fix[x], image[S, singleton[y]]]]]],
  not[FUNCTION[x]], not[member[y, image[inverse[S], fix[x]]]],
  not[subclass[x, S]], not[subclass[composite[x, S], composite[S, x]]]] == True
```

```
In[111]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

This does not take long:

```
In[112]:=
Map[not, SubstTest[and, implies[and[p1, p5, p8, p12], p15],
  implies[p5, p16], implies[and[p15, p16], p17],
    not[implies[and[p1, p5, p8, p12], p17]], {p1 -> FUNCTION[x], p5 -> subcommute[x, S],
   p8 -> subclass[x, S], p10 -> equal[fix[x], fix[HULL[fix[x]]]],
  p12 -> member[y, domain[HULL[fix[x]]]],
    p13 -> member[APPLY[HULL[fix[x]], y], fix[HULL[fix[x]]]],
  p14 -> member[APPLY[HULL[fix[x]], y], fix[x]],
    p15 -> equal[APPLY[x, APPLY[HULL[fix[x]], y]], APPLY[HULL[fix[x]], y]],
  p16 -> subclass[APPLY[x, y], APPLY[x, APPLY[HULL[fix[x]], y]]],
  p17 -> subclass[APPLY[x, y], APPLY[HULL[fix[x]], y]]}]]

Out[112]=
or[not[FUNCTION[x]], not[member[y, image[inverse[S], fix[x]]]],
  not[subclass[x, S]], not[subclass[composite[x, S], composite[S, x]]],
  subclass[APPLY[x, y], A[intersection[fix[x], image[S, singleton[y]]]]]] == True

In[113]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

This is combined:

```
In[114]:=
Map[not,
 SubstTest[and, implies[and[p1, p2, p8], p20], implies[and[p1, p5, p8, p12], p17],
  implies[and[p17, p20], p21], not[implies[and[p1, p2, p5, p8, p12], p21]],
  {p1 -> FUNCTION[x], p2 -> idempotent[x], p5 -> subcommute[x, S], p8 -> subclass[x, S],
  p12 -> member[y, domain[HULL[fix[x]]]],
    p17 -> subclass[APPLY[x, y], APPLY[HULL[fix[x]], y]],
  p20 -> subclass[APPLY[HULL[fix[x]], y], APPLY[x, y]],
    p21 -> equal[APPLY[HULL[fix[x]], y], APPLY[x, y]]}]]

Out[114]=
or[equal[A[intersection[fix[x], image[S, singleton[y]]]], APPLY[x, y]],
  not[equal[x, composite[x, x]]], not[FUNCTION[x]],
  not[member[y, image[inverse[S], fix[x]]]], not[subclass[x, S]],
  not[subclass[composite[x, S], composite[S, x]]]] == True

In[115]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[116]:=
implies[and[FUNCTION[x], idempotent[x],
  subclass[x, S], subcommute[x, S], member[y, domain[HULL[fix[x]]]]],
    equal[APPLY[x, y], APPLY[HULL[fix[x]], y]]]

Out[116]=
True
```

```
In[117]:=
Map[not, SubstTest[and, implies[and[p1, p2, p3, p4, p5], p6], implies[p6, p7],
  implies[p1, p8], implies[and[p7, p8], p9], not[implies[and[p1, p2, p3, p4, p5], p9]],
    {p1 -> FUNCTION[x], p2 -> idempotent[x], p3 -> subclass[x, S],
   p4 -> subcommute[x, S], p5 -> member[y, domain[HULL[fix[x]]]],
   p6 -> equal[APPLY[x, y], APPLY[HULL[fix[x]], y]],
   p7 -> equal[singleton[APPLY[x, y]], singleton[APPLY[HULL[fix[x]], y]]],
   p8 -> equal[image[x, singleton[y]], singleton[APPLY[x, y]]],
   p9 -> equal[image[x, singleton[y]], singleton[APPLY[HULL[fix[x]], y]]]}]]
Out[117]=
or[equal[image[x, singleton[y]],
    singleton[A[intersection[fix[x], image[S, singleton[y]]]]]],
  not[equal[x, composite[x, x]]], not[FUNCTION[x]],
  not[member[y, image[inverse[S], fix[x]]]], not[subclass[x, S]],
  not[subclass[composite[x, S], composite[S, x]]]] == True

In[118]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[119]:=
implies[and[FUNCTION[x], idempotent[x],
  subclass[x, S], subcommute[x, S], member[y, domain[HULL[fix[x]]]]],
 equal[image[x, singleton[y]], image[HULL[fix[x]], singleton[y]]]]
Out[119]=
True

In[120]:=
SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
 {u -> image[x, singleton[z]], v -> 0, w -> image[y, singleton[z]]}]
Out[120]=
or[equal[image[x, singleton[z]], image[y, singleton[z]]],
  member[z, domain[x]], member[z, domain[y]]] == True

In[121]:=
(% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

In particular, the case of interest:

```
In[122]:=
SubstTest[implies, and[not[member[y, domain[x]]], not[member[y, domain[z]]]],
 equal[image[x, singleton[y]], image[z, singleton[y]]],
    z -> HULL[fix[x]]]
Out[122]=
or[equal[image[x, singleton[y]],
    singleton[A[intersection[fix[x], image[S, singleton[y]]]]]],
  member[y, domain[x]], member[y, image[inverse[S], fix[x]]]] == True

In[123]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

```
In[124]:=
Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p4], p5],
  implies[and[p2, p3], p4], implies[and[not[p3], not[p4]], p5], not[implies[p1, p5]],
  {p1 -> hypotheses[x], p2 -> equal[domain[x], image[inverse[S], fix[x]]],
   p3 -> member[y, domain[x]], p4 -> member[y, image[inverse[S], fix[x]]],
   p5 -> equal[image[x, singleton[y]],
     singleton[A[intersection[fix[x], image[S, singleton[y]]]]]]}]]
```

```
Out[124]=
or[equal[image[x, singleton[y]],
   singleton[A[intersection[fix[x], image[S, singleton[y]]]]],
  not[equal[x, composite[x, x]]], not[FUNCTION[x]], not[subclass[x, S]],
  not[subclass[composite[x, S], composite[S, x]]]] == True
```

```
In[125]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[126]:=
implies[hypotheses[x], equal[image[x, singleton[y]],
  singleton[A[intersection[fix[x], image[S, singleton[y]]]]]]]
```

```
Out[126]=
True
```

The negation of this statement is needed:

```
In[127]:=
and[equal[x, composite[x, x]], FUNCTION[x], not[equal[image[x, singleton[y]],
    singleton[A[intersection[fix[x], image[S, singleton[y]]]]]]],
  subclass[x, S], subclass[composite[x, S], composite[S, x]]] // NotNotTest
```

```
Out[127]=
and[equal[x, composite[x, x]], FUNCTION[x], not[equal[image[x, singleton[y]],
    singleton[A[intersection[fix[x], image[S, singleton[y]]]]]]],
  subclass[x, S], subclass[composite[x, S], composite[S, x]]] == False
```

```
In[128]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

```
In[129]:=
SubstTest[assert,
  forall[y, implies[p[x], equal[image[x, singleton[y]], image[z, singleton[y]]]]],
 {z -> HULL[fix[x]], p[x] -> hypotheses[x]}] // Reverse
```

```
Out[129]=
or[equal[composite[Id, x], HULL[fix[x]]],
  not[equal[x, composite[x, x]]], not[FUNCTION[x]], not[subclass[x, S]],
  not[subclass[composite[x, S], composite[S, x]]]] == True
```

```
In[130]:=
(% /. x -> x_) /. Equal -> SetDelayed
```

```
In[131]:=
SubstTest[implies,
 and[equal[x, composite[Id, y]], equal[composite[Id, y], z]], equal[x, z], z -> y]
```

```
Out[131]=
or[equal[x, y], not[equal[x, composite[Id, y]]], not[subclass[y, cart[V, V]]]] == True
```

```
In[132]:=
(% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Main result:

In[133]:=
```
Map[not, SubstTest[and, implies[p1, p2],
  implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
        {p1 -> hypotheses[x], p2 -> equal[composite[Id, x], HULL[fix[x]]],
   p3 -> equal[x, composite[Id, x]], p4 -> equal[x, HULL[fix[x]]]}]]
```

Out[133]=
```
or[equal[x, HULL[fix[x]]], not[equal[x, composite[x, x]]], not[FUNCTION[x]],
  not[subclass[x, S]], not[subclass[composite[x, S], composite[S, x]]]] == True
```

In[134]:=
```
or[equal[x_, HULL[fix[x_]]], not[equal[x_, composite[x_, x_]]], not[FUNCTION[x_]],
  not[subclass[x_, S]], not[subclass[composite[x_, S], composite[S, x_]]]] := True
```

Restatement:

In[135]:=
```
implies[hypotheses[x], equal[x, HULL[fix[x]]]]
```

Out[135]=
```
True
```