two HULL functions

Johan G. F. Belinfante
2011 November 5

```
In[1]:= SetDirectory["l:"]; << goedel.11nov03a

        :Package Title: goedel.11nov03a               2011 November 3 at 9:40 a.m.

        Loading takes about thirteen minutes, half that time due to builtin pauses.

        It is now:  2011 Nov 5 at 1:52

        Loading Simplification Rules

        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

        weightlimit = 40

        Loading completed.

        It is now:  2011 Nov 5 at 2:5
```

summary

A new characterization of **HULL** functions is derived and applied to two special cases involving the function **IMAGE[INVERSE]**.

characterizing HULL functions

The following new characterization replaces the idempotence condition with the condition that the range be equal to the fixed point class.

Theorem. A new characterization of **HULL** functions.

```
In[2]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p5], p6],
           implies[and[p1, p4], p5], not[implies[and[p1, p2, p3, p4], p6]], {p1 → FUNCTION[x],
            p2 → subclass[x, S], p3 → subcommute[x, S], p4 → equal[fix[x], range[x]],
            p5 → idempotent[x], p6 → equal[x, HULL[fix[x]]]}]] // Reverse

Out[2]= or[equal[x, HULL[fix[x]]], not[equal[fix[x], range[x]]], not[FUNCTION[x]],
          not[subclass[x, S]], not[subclass[composite[x, S], composite[S, x]]]] == True

In[3]:= or[equal[x_, HULL[fix[x_]]], not[equal[fix[x_], range[x_]]], not[FUNCTION[x_]],
          not[subclass[x_, S]], not[subclass[composite[x_, S], composite[S, x_]]]] := True
```

HULL[invar[INVERSE]]

Lemma.

```
In[4]:= Map[empty[composite[Id, complement[#]]] &,
         dif[IMAGE[INVERSE], image[inverse[CUP], invar[INVERSE]]] // complement//
          FastReifNormality]
```

```
Out[4]= subclass[image[CUP, IMAGE[INVERSE]], invar[INVERSE]] == True
```

```
In[5]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[6]:= SubstTest[and, subclass[u, v], subclass[v, u],
         {u -> image[CUP, IMAGE[INVERSE]], v -> invar[INVERSE]}]
```

```
Out[6]= equal[image[CUP, IMAGE[INVERSE]], invar[INVERSE]] == True
```

```
In[7]:= image[CUP, IMAGE[INVERSE]] := invar[INVERSE]
```

Lemma.

```
In[8]:= member[pair[x, y], fix[composite[inverse[CUP], S, IMAGE[INVERSE], FIRST]]] // AssertTest
```

```
Out[8]= member[pair[x, y], fix[composite[inverse[CUP], S, IMAGE[INVERSE], FIRST]]] ==
         and[member[x, V], member[y, V], subclass[image[INVERSE, x], union[x, y]]]
```

```
In[9]:= member[pair[x_, y_], fix[composite[inverse[CUP], S, IMAGE[INVERSE], FIRST]]] :=
         and[member[x, V], member[y, V], subclass[image[INVERSE, x], union[x, y]]]
```

Lemma.

```
In[10]:= Map[empty[composite[Id, complement[#]]] &,
          dif[IMAGE[INVERSE], fix[composite[inverse[CUP], S, IMAGE[INVERSE], FIRST]]] //
            complement// FastReifNormality]
```

```
Out[10]= subclass[IMAGE[INVERSE], fix[composite[inverse[CUP], S, IMAGE[INVERSE], FIRST]]] == True
```

```
In[11]:= % /. Equal→ SetDelayed
```

Theorem. A subcommutativity property.

```
In[12]:= ((subclass[composite[CUP, id[x], inverse[FIRST], S], composite[S, CUP, id[x],
                inverse[FIRST]]] // AssertTest) /. x -> IMAGE[INVERSE]) // InvertFix
```

```
Out[12]= subclass[composite[CUP, id[IMAGE[INVERSE]], inverse[FIRST], S],
           composite[S, CUP, id[IMAGE[INVERSE]], inverse[FIRST]]] == True
```

```
In[13]:= % /. Equal→ SetDelayed
```

Theorem. An application of the new characterization of **HULL** functions.

```
In[14]:= SubstTest[implies, and[equal[fix[x], range[x]],
            FUNCTION[x], subclass[x, S], subcommute[x, S]], equal[x, HULL[fix[x]]],
          x -> composite[CUP, id[IMAGE[INVERSE]], inverse[FIRST]]] // Reverse

Out[14]= equal[composite[CUP, id[IMAGE[INVERSE]], inverse[FIRST]], HULL[invar[INVERSE]]] == True

In[15]:= composite[CUP, id[IMAGE[INVERSE]], inverse[FIRST]] := HULL[invar[INVERSE]]
```

Corollary.

```
In[16]:= ImageComp[CUP, composite[id[IMAGE[INVERSE]], inverse[FIRST]], x] // Reverse

Out[16]= image[CUP, composite[IMAGE[INVERSE], id[x]]] == image[HULL[invar[INVERSE]], x]

In[17]:= image[CUP, composite[IMAGE[INVERSE], id[x_]]] := image[HULL[invar[INVERSE]], x]
```

Observation. The function **HULL[invar[INVERSE]]** is total.

```
In[21]:= domain[HULL[invar[INVERSE]]]

Out[21]= V
```

Theorem.

```
In[23]:= SubstTest[composite, BIGCUP, IMAGE[HULL[t]],
           POWER, id[image[inverse[S], t]], t -> invar[INVERSE]] // Reverse

Out[23]= composite[BIGCUP, IMAGE[HULL[invar[INVERSE]]], POWER] == HULL[invar[INVERSE]]

In[24]:= composite[BIGCUP, IMAGE[HULL[invar[INVERSE]]], POWER] := HULL[invar[INVERSE]]
```

Theorem. A formula for restrictions of **HULL[invar[INVERSE]]**.

```
In[34]:= Assoc[composite[CUP, id[IMAGE[INVERSE]]], inverse[FIRST], id[x]]

Out[34]= composite[CUP, id[composite[IMAGE[INVERSE], id[x]]], inverse[FIRST]] ==
          composite[HULL[invar[INVERSE]], id[x]]

In[35]:= composite[CUP, id[composite[IMAGE[INVERSE], id[x_]]], inverse[FIRST]] :=
          composite[HULL[invar[INVERSE]], id[x]]
```

HULL[fix[IMAGE[INVERSE]]]

The function **HULL[fix[IMAGE[INVERSE]]]** is not total. A formula for its domain will now be derived.

Lemma.

```
In[26]:= Map[equal[V, domain[reify[x, case[#]]]] &,
           SubstTest[implies, and[subclass[x, y], member[y, z]], member[x, image[inverse[S], z]],
            {y -> union[x, image[INVERSE, x]], z -> fix[IMAGE[INVERSE]]}]] // Reverse

Out[26]= subclass[P[P[cart[V, V]]], image[inverse[S], fix[IMAGE[INVERSE]]]] == True

In[27]:= % /. Equal→ SetDelayed
```

Theorem. A formula for the domain of **HULL[fix[IMAGE[INVERSE]]]**.

```
In[28]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[inverse[S], fix[IMAGE[INVERSE]]], v -> P[P[cart[V, V]]]}]

Out[28]= equal[image[inverse[S], fix[IMAGE[INVERSE]]], P[P[cart[V, V]]]] == True

In[29]:= image[inverse[S], fix[IMAGE[INVERSE]]] := P[P[cart[V, V]]]
```

The function **IMAGE[INVERSE]** is not one-to-one, but a restriction of it is.

Theorem.

```
In[32]:= SubstTest[ONEONE,
           composite[IMAGE[oopart[x]], id[P[domain[oopart[x]]]]], x → INVERSE] // Reverse

Out[32]= FUNCTION[composite[id[P[P[cart[V, V]]]], inverse[IMAGE[INVERSE]]]] == True

In[33]:= FUNCTION[composite[id[P[P[cart[V, V]]]], inverse[IMAGE[INVERSE]]]] := True
```

Lemma.

```
In[37]:= ((member[pair[u, v], composite[z, IMAGE[INVERSE]]] // AssertTest) /.
           z -> image[inverse[CUP], fix[IMAGE[INVERSE]]]) /. {u → setpart[x], v → setpart[y]}

Out[37]= member[pair[setpart[x], setpart[y]],
           composite[image[inverse[CUP], fix[IMAGE[INVERSE]]], IMAGE[INVERSE]]] ==
          equal[union[image[INVERSE, setpart[x]], setpart[y]],
           union[image[INVERSE, setpart[y]], intersection[P[cart[V, V]], setpart[x]]]]

In[38]:= member[pair[setpart[x_], setpart[y_]],
           composite[image[inverse[CUP], fix[IMAGE[INVERSE]]], IMAGE[INVERSE]]] :=
          equal[union[image[INVERSE, setpart[x]], setpart[y]],
           union[image[INVERSE, setpart[y]], intersection[P[cart[V, V]], setpart[x]]]]
```

Theorem.

```
In[41]:= SubstTest[class, x, member[pair[setpart[x], setpart[x]], y],
           y -> composite[image[inverse[CUP], fix[IMAGE[INVERSE]]], IMAGE[INVERSE]]]

Out[41]= fix[composite[image[inverse[CUP], fix[IMAGE[INVERSE]]], IMAGE[INVERSE]]] ==
          P[P[cart[V, V]]]

In[42]:= fix[composite[image[inverse[CUP], fix[IMAGE[INVERSE]]], IMAGE[INVERSE]]] :=
          P[P[cart[V, V]]]
```

Theorem.

```
In[43]:= ImInComp[CUP, composite[id[IMAGE[INVERSE]], inverse[FIRST]], fix[IMAGE[INVERSE]]]

Out[43]= image[inverse[HULL[invar[INVERSE]]], fix[IMAGE[INVERSE]]] == P[P[cart[V, V]]]

In[44]:= image[inverse[HULL[invar[INVERSE]]], fix[IMAGE[INVERSE]]] := P[P[cart[V, V]]]
```

Corollary.

```
In[45]:= ImageComp[HULL[invar[INVERSE]],
           inverse[HULL[invar[INVERSE]]], fix[IMAGE[INVERSE]]] // Reverse

Out[45]= image[HULL[invar[INVERSE]], P[P[cart[V, V]]]] == fix[IMAGE[INVERSE]]

In[46]:= image[HULL[invar[INVERSE]], P[P[cart[V, V]]]] := fix[IMAGE[INVERSE]]
```

Theorem. Another application of the new characterization of **HULL** functions.

```
In[47]:= SubstTest[implies, and[equal[fix[x], range[x]],
            FUNCTION[x], subclass[x, S], subcommute[x, S]], equal[x, HULL[fix[x]]],
           x -> composite[HULL[invar[INVERSE]], id[P[P[cart[V, V]]]]]] // Reverse

Out[47]= equal[composite[HULL[invar[INVERSE]], id[P[P[cart[V, V]]]]],
           HULL[fix[IMAGE[INVERSE]]]] == True

In[48]:= composite[HULL[invar[INVERSE]], id[P[P[cart[V, V]]]]] := HULL[fix[IMAGE[INVERSE]]]
```

Observation.

```
In[49]:= composite[CUP, id[composite[IMAGE[INVERSE], id[P[P[cart[V, V]]]]]], inverse[FIRST]]

Out[49]= HULL[fix[IMAGE[INVERSE]]]
```

Theorem.

```
In[54]:= ImageComp[composite[CUP, id[composite[IMAGE[INVERSE], id[P[P[cart[V, V]]]]]]],
           inverse[FIRST], V]

Out[54]= fix[HULL[fix[IMAGE[INVERSE]]]] == fix[IMAGE[INVERSE]]

In[55]:= fix[HULL[fix[IMAGE[INVERSE]]]] := fix[IMAGE[INVERSE]]
```

serendipity

The theory of the two new **HULL** functions closely resembles that of the functions **HULL[invar[SWAP]]** and **HULL[-SYM]**. This analogy was a source of inspiration for much of the material in this notebook. New properties of these functions were discovered in the course of this work.

Theorem.

```
In[65]:= ImageComp[composite[CUP, id[IMAGE[SWAP]]], inverse[FIRST], V] // Reverse
Out[65]= image[CUP, IMAGE[SWAP]] == invar[SWAP]

In[66]:= image[CUP, IMAGE[SWAP]] := invar[SWAP]
```

Theorem.

```
In[70]:= SubstTest[composite, BIGCUP, IMAGE[HULL[t]],
           POWER, id[image[inverse[S], t]], t → invar[SWAP]] // Reverse
Out[70]= composite[BIGCUP, IMAGE[HULL[invar[SWAP]]], POWER] == HULL[invar[SWAP]]

In[72]:= composite[BIGCUP, IMAGE[HULL[invar[SWAP]]], POWER] := HULL[invar[SWAP]]
```