characterizing HULL[x] functions

Johan G. F. Belinfante
2014 January 4

```
In[1]:= SetDirectory["l:"]; << goedel.14jan03a
```

```
:Package Title: goedel.14jan03a                    2014 January 3 at 4:20 p.m.

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

It is now:  2014 Jan 4 at 7:24

Loading Simplification Rules

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

weightlimit = 40

Loading completed.

It is now:  2014 Jan 4 at 7:41
```

summary

Characterizations **HULL[x]** functions are studied.

intersections

Some simplification rules for intersections with **monotone[S, S]** are derived in this section.

Theorem.

```
In[9]:= AssInt[monotone[S, S], FUNS, invar[composite[DUP, SECOND]]] // Reverse
```

```
Out[9]= intersection[invar[composite[DUP, SECOND]], monotone[S, S]] ==
         intersection[PROJ, monotone[S, S]]
```

```
In[10]:= intersection[invar[composite[DUP, SECOND]], monotone[S, S]] :=
           intersection[PROJ, monotone[S, S]]
```

It is not immediately clear how to orient the following rewrite rule. The choice made here is somewhat arbitrary.

Theorem.

```
In[11]:= AssInt[monotone[S, S], FUNS, IDEM] // Reverse
```

```
Out[11]= intersection[IDEM, monotone[S, S]] == intersection[PROJ, monotone[S, S]]
```

```
In[12]:= intersection[IDEM, monotone[S, S]] := intersection[PROJ, monotone[S, S]]
```

Theorem.

```
In[37]:= equal[intersection[PROJ, invar[composite[DUP, SECOND]]], PROJ]

Out[37]= True

In[39]:= intersection[PROJ, invar[composite[DUP, SECOND]]] := PROJ
```

subcommutant[inverse[S]]

Lemma.

```
In[14]:= Map[equal[V, #] &,
          dif[intersection[PROJ, image[INVERSE, subcommutant[inverse[S]]], P[S]],
            range[LAMBHULL]] // complement // Normality]

Out[14]= subclass[intersection[PROJ, image[INVERSE, subcommutant[inverse[S]]], P[S]],
           range[LAMBHULL]] == True

In[15]:= % /. Equal → SetDelayed
```

Theorem.

```
In[16]:= Map[equal[V, #] &,
          dif[range[LAMBHULL], image[INVERSE, subcommutant[inverse[S]]]] // complement //
           Normality]

Out[16]= subclass[image[INVERSE, range[LAMBHULL]], subcommutant[inverse[S]]] == True

In[17]:= subclass[image[INVERSE, range[LAMBHULL]], subcommutant[inverse[S]]] := True
```

Theorem.

```
In[19]:= equal[intersection[PROJ, image[INVERSE, subcommutant[inverse[S]]], P[S]],
           range[LAMBHULL]] // AssertTest

Out[19]= equal[intersection[PROJ, image[INVERSE, subcommutant[inverse[S]]], P[S]],
           range[LAMBHULL]] == True

In[20]:= intersection[PROJ, image[INVERSE, subcommutant[inverse[S]]], P[S]] := range[LAMBHULL]
```

monotone[S, S]

The condition $\mathbf{x} \circ \mathbf{S} \subset \mathbf{S} \circ \mathbf{x}$ in the characterization of **HULL[x]** can be replaced with the monotonicity condition $\mathbf{x} \circ \mathbf{S} \circ \mathbf{inverse[x]} \subset \mathbf{S}$ if one adds the condition that **domain[x]** is invariant under **inverse[S]**.

Theorem.

```
In[22]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4], p9],
           implies[and[p5, p6], p4], implies[p3, p7], implies[and[p5, p7], p1],
           not[implies[and[p2, p3, p5, p6], p9]], {p1 → FUNCTION[x], p2 → subclass[x, S],
            p3 → equal[composite[x, x], x], p4 → subclass[composite[x, S], composite[S, x]],
            p5 -> subclass[composite[x, S, inverse[x]], S],
            p6 → equal[image[inverse[S], domain[x]], domain[x]],
            p7 → subclass[x, cart[V, V]], p9 → equal[x, HULL[fix[x]]]}]] // Reverse

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

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

Corollary. A characterization of **HULL** functions using the monotonicity condition $\mathbf{x \circ S \circ inverse[x] \subset S}$.

```
In[24]:= equiv[and[equal[x, composite[x, x]], equal[domain[x], image[inverse[S], domain[x]]],
           subclass[x, S], subclass[composite[x, S, inverse[x]], S]], equal[x, HULL[fix[x]]]]

Out[24]= True

In[25]:= and[equal[x_, composite[x_, x_]], equal[domain[x_], image[inverse[S], domain[x_]]],
           subclass[x_, S], subclass[composite[x_, S, inverse[x_]], S]] := equal[x, HULL[fix[x]]]
```

Observation. The following variable-free restatement is already recognized as true by the **GOEDEL** program via available rewrite rules.

```
In[30]:= intersection[IDEM,
          image[inverse[IMAGE[FIRST]], fix[IMAGE[inverse[S]]]], monotone[S, S], P[S]]

Out[30]= range[LAMBHULL]
```

One can also replace **IDEM** with **PROJ**.

```
In[31]:= intersection[PROJ,
          image[inverse[IMAGE[FIRST]], fix[IMAGE[inverse[S]]]], monotone[S, S], P[S]]

Out[31]= range[LAMBHULL]
```

One can also replace **IDEM** with **invar[DUP ∘ SECOND]**.

```
In[32]:= intersection[invar[composite[DUP, SECOND]],
          image[inverse[IMAGE[FIRST]], fix[IMAGE[inverse[S]]]], monotone[S, S], P[S]]

Out[32]= range[LAMBHULL]
```

serendipity

The following rewrite rule was also discovered in the course of this study, but does not specifically involve **HULL** functions.

Theorem.

```
In[41]:= SubstTest[subclass,
          intersection[image[inverse[IMAGE[FIRST]], fix[IMAGE[inverse[S]]]], monotone[S, S]],
          image[INVERSE, subcommutant[t]], t -> inverse[S]]

Out[41]= subclass[image[INVERSE, intersection[FUNS, image[INVERSE, subcommutant[inverse[S]]]]],
           subcommutant[inverse[S]]] == True

In[42]:= subclass[image[INVERSE, intersection[FUNS, image[INVERSE, subcommutant[inverse[S]]]]],
           subcommutant[inverse[S]]] := True
```