HULL[x] and INVERSE

Johan G. F. Belinfante
2006 September 30

```
In[1]:= SetDirectory["l:"]; << goedel85.28a; << tools.m

        :Package Title: goedel85.28a          2006 September 28 at 2:45 p.m.

        It is now:  2006 Sep 30 at 13:53

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 September 24

        weightlimit = 40
```

summary

A theorem about composites of **HULL[x]** with **INVERSE** is derived.

derivation

The following characterization of **HULL** functions will be used.

```
In[2]:= implies[and[FUNCTION[x], idempotent[x], subclass[x, S], subcommute[x, S]],
          equal[x, HULL[fix[x]]]]

Out[2]= True
```

The function to which this theorem will be applied is **composite[INVERSE, HULL[x], INVERSE]**.

```
In[3]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → HULL[x], v → S, w → cross[INVERSE, INVERSE]}]

Out[3]= subclass[composite[INVERSE, HULL[x], INVERSE], S] == True

In[4]:= subclass[composite[INVERSE, HULL[x_], INVERSE], S] := True
```

The following rewrite rule interferes slightly with our work:

```
In[5]:= composite[S, INVERSE]

Out[5]= composite[inverse[IMAGE[SWAP]], S]
```

The following new rule helps in this regard.

```
In[6]:= subclass[INVERSE, inverse[IMAGE[SWAP]]] // AssertTest

Out[6]= subclass[INVERSE, inverse[IMAGE[SWAP]]] == True

In[7]:= subclass[INVERSE, inverse[IMAGE[SWAP]]] := True
```

A corollary is that **INVERSE** subcommutes with **S**.

```
In[8]:= subcommute[INVERSE, S]

Out[8]= True
```

Since **INVERSE** and **HULL[x]** both subcommute with **S**, so does their composite:

```
In[9]:= SubstTest[implies, and[subcommute[u, S], subcommute[v, S]],
          subcommute[composite[u, v], S], {u → HULL[x], v → INVERSE}]

Out[9]= subclass[composite[HULL[x], INVERSE, S], composite[S, HULL[x], INVERSE]] == True

In[10]:= (% /. x → x_) /. Equal → SetDelayed
```

A second application yields:

```
In[11]:= SubstTest[implies, and[subcommute[u, S], subcommute[v, S]],
          subcommute[composite[u, v], S], {u → INVERSE, v → composite[HULL[x], INVERSE]}]

Out[11]= subclass[composite[INVERSE, HULL[x], INVERSE, S],
          composite[inverse[IMAGE[SWAP]], S, HULL[x], INVERSE]] == True

In[12]:= (% /. x → x_) /. Equal → SetDelayed
```

At this point, one obtains:

```
In[13]:= SubstTest[or, equal[HULL[fix[y]], y], not[equal[composite[y, y], y]],
          not[FUNCTION[y]], not[subclass[composite[y, S], composite[S, y]]],
          not[subclass[y, S]], y → composite[INVERSE, HULL[x], INVERSE]]

Out[13]= or[equal[composite[INVERSE, HULL[x], INVERSE], HULL[image[INVERSE, fix[HULL[x]]]]],
          not[equal[composite[INVERSE, HULL[x], INVERSE],
            composite[INVERSE, HULL[x], id[P[cart[V, V]]], HULL[x], INVERSE]]]] == True

In[14]:= (% /. x → x_) /. Equal → SetDelayed
```

The following theorem will be used to obtain a simple corollary.

```
In[15]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
          {u -> composite[id[P[cart[V, V]]], HULL[x]],
           v -> HULL[x], w -> cross[INVERSE, composite[INVERSE, HULL[x]]]}]

Out[15]= or[equal[composite[INVERSE, HULL[x], INVERSE],
           composite[INVERSE, HULL[x], id[P[cart[V, V]]], HULL[x], INVERSE]],
          not[subclass[U[x], cart[V, V]]]] == True

In[16]:= (% /. x → x_) /. Equal → SetDelayed
```

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
           {p1 -> subclass[U[x], cart[V, V]], p2 -> equal[composite[INVERSE, HULL[x], INVERSE],
              composite[INVERSE, HULL[x], id[P[cart[V, V]]], HULL[x], INVERSE]],
            p3 -> equal[composite[INVERSE, HULL[x], INVERSE],
              HULL[image[INVERSE, fix[HULL[x]]]]]}]]

Out[17]= or[equal[composite[INVERSE, HULL[x], INVERSE], HULL[image[INVERSE, fix[HULL[x]]]]],
           not[subclass[U[x], cart[V, V]]]] == True

In[18]:= or[equal[composite[INVERSE, HULL[x_], INVERSE], HULL[image[INVERSE, fix[HULL[x_]]]]],
           not[subclass[U[x_], cart[V, V]]]] := True
```

---

an application to integer arithmetic

An application of interest is the case that **x** is the class **Z** of integers. The following simplication rules will be used:

```
In[19]:= SubstTest[HULL, fix[HULL[x]], x → Z]

Out[19]= HULL[union[Z, set[0]]] == HULL[Z]

In[20]:= HULL[union[Z, set[0]]] := HULL[Z]

In[21]:= Assoc[id[P[cart[V, V]]], id[union[Z, set[0]]], HULL[Z]]

Out[21]= composite[id[P[cart[V, V]]], HULL[Z]] == HULL[Z]

In[22]:= composite[id[P[cart[V, V]]], HULL[Z]] := HULL[Z]
```

Corollary.

```
In[23]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], HULL[Z]]

Out[23]= composite[IMAGE[SWAP], HULL[Z]] == composite[INVERSE, HULL[Z]]

In[24]:= composite[IMAGE[SWAP], HULL[Z]] := composite[INVERSE, HULL[Z]]
```

The theorem derived in the preceding section yields:

```
In[25]:= SubstTest[implies, subclass[U[x], cart[V, V]],
          equal[composite[INVERSE, HULL[x], INVERSE], HULL[image[INVERSE, fix[HULL[x]]]]], x → Z]

Out[25]= equal[composite[INVERSE, HULL[Z], INVERSE], HULL[Z]] == True

In[26]:= % /. Equal → SetDelayed
```

Corollary. The functions **HULL[Z]** and **INVERSE** commute.

```
In[27]:= SubstTest[implies, equal[u, v], equal[composite[w, u], composite[w, v]],
          {u -> composite[INVERSE, HULL[Z], INVERSE], v -> HULL[Z], w → INVERSE}]

Out[27]= equal[composite[INVERSE, HULL[Z]], composite[HULL[Z], INVERSE]] == True
```

The following orientation of this equation as a rewrite rule is tentative, based on an analogy with a similar rule for **HULL[TRV]**.

```
In[28]:= composite[HULL[Z], INVERSE] := composite[INVERSE, HULL[Z]]
```