U[A[image[IMAGE[inverse[S]], x]]]

Johan G. F. Belinfante
2011 March 18

```
In[1]:= SetDirectory["l:"]; << goedel.11mar17a

        :Package Title: goedel.11mar17a               2011 March 17 at 1:40 p.m.

        It is now:  2011 Mar 18 at 14:17

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

A class **w** is **hereditary** if every subset of a member is a member. Any class is a subclass of a hereditary class. The least such class **image[inverse[S], x]** is called its **hereditary hull**. The sum class of any class is the same as that of its hereditary hull. Recently it was shown using the **GOEDEL** program that the sum class of an intersection of hereditary hulls is the intersection of their sum classes.

```
In[2]:= U[intersection[image[inverse[S], x], image[inverse[S], y]]]

Out[2]= intersection[U[x], U[y]]
```

In this notebook a generalization of this result is derived for an arbitrary collection of hereditary sets.

derivation

The class of all hereditary sets is **fix[IMAGE[inverse[S]]]**. More generally, any collection of hereditary sets can be written as the class **image[IMAGE[inverse[S]], x]** of all hereditary hulls of members of a class **x**. The result to be shown below is currently recognized by the **GOEDEL** program for the case of two sets:

```
In[3]:= U[A[image[IMAGE[inverse[S]], x]]] == A[image[BIGCUP, x]] /. x → set[a, b]

Out[3]= True
```

But not yet for three sets:

```
In[4]:= U[A[image[IMAGE[inverse[S]], x]]] == A[image[BIGCUP, x]] /. x → set[a, b, c]

Out[4]= U[intersection[image[inverse[S], a], image[inverse[S], b], image[inverse[S], c]]] ==
         intersection[U[a], U[b], U[c]]
```

One coulld derive the special case of three sets from that of two sets, and presumably a similar technique using induction would suffice for any finite collection of hereditary classes.

```
In[6]:= SubstTest[U, intersection[image[inverse[S], t], image[inverse[S], z]],
          t → intersection[image[inverse[S], x], image[inverse[S], y]]] // Reverse

Out[6]= U[intersection[image[inverse[S], x], image[inverse[S], y], image[inverse[S], z]]] ==
         intersection[U[x], U[y], U[z]]
```

However, a different technique is needed for the general case.

Theorem. If **w ∈ x**, then **A[image[BIGCUP, x]] ⊂ U[w]**.

```
In[7]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
          {t → composite[complement[inverse[E]], BIGCUP], u → set[w], v → x}] // Reverse

Out[7]= or[not[member[w, x]], subclass[A[image[BIGCUP, x]], U[w]]] == True

In[8]:= or[not[member[w_, x_]], subclass[A[image[BIGCUP, x_]], U[w_]]] := True
```

Corollary.

```
In[9]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           not[implies[p1, p3]], {p1 → and[member[w, x], member[t, A[image[BIGCUP, x]]]],
            p2 -> subclass[A[image[BIGCUP, x]], U[w]], p3 → member[t, U[w]]}]] // Reverse

Out[9]= or[member[t, U[w]], not[member[t, A[image[BIGCUP, x]]]], not[member[w, x]]] == True

In[10]:= (% /. {t → t_, w → w_, x → x_}) /. Equal → SetDelayed
```

If **t ∈ A[image[BIGCUP, x]** and **w ∈ x**, there exists an element **v ∈ w** with **t ∈ w**. The key idea of the derivation is to consider the intersection **hull[U[x], set[t]]** of all such sets **v**.

Lemma.

```
In[11]:= SubstTest[implies, member[y, t], subclass[A[t], y],
           {t → intersection[z, image[E, set[x]]]}] // Reverse

Out[11]= or[not[member[x, y]], not[member[y, z]], subclass[hull[z, set[x]], y]] == True

In[12]:= or[not[member[x_, y_]], not[member[y_, z_]], subclass[hull[z_, set[x_]], y_]] := True
```

Lemma.

```
In[13]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
            implies[and[p3, p4], p5], not[implies[and[p1, p2, p3], p5]], {p1 → member[t, v],
             p2 → member[v, U[x]], p3 → member[v, w], p4 → subclass[hull[U[x], set[t]], v],
             p5 → member[hull[U[x], set[t]], image[inverse[S], w]]}]] // Reverse

Out[13]= or[member[hull[U[x], set[t]], image[inverse[S], w]],
           not[member[t, v]], not[member[v, w]], not[member[v, U[x]]]] == True

In[14]:= (% /. {t → t_, v → v_, w → w_, x → x_}) /. Equal → SetDelayed
```

The following corollary eliminates reference to the class **U[x]**.

Corollary.

```
In[15]:= Map[not, SubstTest[and, implies[and[p2, p3], p4],
            implies[and[p1, p2, p4], p5], not[implies[and[p1, p2, p3], p5]],
            {p1 → member[t, v], p2 → member[v, w], p3 → member[w, x], p4 → member[v, U[x]],
             p5 → member[hull[U[x], set[t]], image[inverse[S], w]]}]] // Reverse

Out[15]= or[member[hull[U[x], set[t]], image[inverse[S], w]],
           not[member[t, v]], not[member[v, w]], not[member[w, x]]] == True

In[16]:= (% /. {t → t_, v → v_, w → w_, x → x_}) /. Equal → SetDelayed
```

Eliminating the variable **v** yields the following theorem.

Theorem. If **t ∈ U[w]** and **w ∈ x**, then **hull[U[x], set[t]]** is a subset of some member of **w**.

```
In[17]:= Map[equal[V, #] &, SubstTest[class, v,
           or[member[y, z], not[member[t, v]], not[member[v, w]], not[member[w, x]]],
           {y -> hull[U[x], set[t]], z -> image[inverse[S], w]}]]

Out[17]= or[member[hull[U[x], set[t]], image[inverse[S], w]],
           not[member[t, U[w]]], not[member[w, x]]] == True

In[19]:= or[member[hull[U[x_], set[t_]], image[inverse[S], w_]],
           not[member[t_, U[w_]]], not[member[w_, x_]]] := True
```

Lemma. (Introduce the class **A[image[BIGCUP, x]]**.)

```
In[20]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 → member[t, A[image[BIGCUP, x]]], p2 → member[w, x], p3 → member[t, U[w]],
             p4 -> member[hull[U[x], set[t]], image[inverse[S], w]]}]] // Reverse

Out[20]= or[member[hull[U[x], set[t]], image[inverse[S], w]],
           not[member[t, A[image[BIGCUP, x]]]], not[member[w, x]]] == True

In[21]:= (% /. {t → t_, w → w_, x → x_}) /. Equal → SetDelayed
```

Theorem. Eliminate **w**.

```
In[22]:= Map[equal[V, #] &, SubstTest[class, w,
           or[member[y, image[inverse[S], w]], not[member[t, z]], not[member[w, x]]],
           {y -> hull[U[x], set[t]], z -> A[image[BIGCUP, x]]}]]

Out[22]= or[equal[0, intersection[x, P[complement[image[S, set[hull[U[x], set[t]]]]]]]],
           not[member[t, A[image[BIGCUP, x]]]]] == True

In[23]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

A better result can be obtained by using the following lemma.

Lemma.

```
In[24]:= Map[implies[and[not[empty[x]], #], member[t, A[image[IMAGE[inverse[S]], x]]]] &,
           SubstTest[subclass, x, image[inverse[IMAGE[inverse[S]]], y],
            y → image[E, set[t]]]] // Reverse

Out[24]= or[equal[0, x], member[t, A[image[IMAGE[inverse[S]], x]]],
           not[equal[0, intersection[x, P[complement[image[S, set[t]]]]]]]] == True

In[25]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Theorem. If **t** belongs to the sum class of every member of **x**, and **x** is not empty, then **hull[U[x], set[t]]** belongs to the hereditary hull of every member of **x**.

```
In[26]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p0, p2], p3], not[implies[and[p0, p1], p3]],
            {p0 → not[empty[x]], p1 -> member[t, A[image[BIGCUP, x]]],
             p2 → disjoint[x, P[complement[image[S, set[hull[U[x], set[t]]]]]]],
             p3 → member[hull[U[x], set[t]], A[image[IMAGE[inverse[S]], x]]]}]] // Reverse

Out[26]= or[equal[0, x], member[hull[U[x], set[t]], A[image[IMAGE[inverse[S]], x]]],
           not[member[t, A[image[BIGCUP, x]]]]] == True

In[28]:= or[equal[0, x_], member[hull[U[x_], set[t_]], A[image[IMAGE[inverse[S]], x_]]],
           not[member[t_, A[image[BIGCUP, x_]]]]] := True
```

Lemma.

```
In[29]:= Map[implies[member[t, y], #] &,
           SubstTest[implies, and[member[t, u], member[u, v]], member[t, U[v]],
            {u -> hull[U[x], set[t]], v -> A[image[IMAGE[inverse[S]], x]]}]] // Reverse

Out[29]= or[member[t, U[A[image[IMAGE[inverse[S]], x]]]], not[member[t, y]],
           not[member[hull[U[x], set[t]], A[image[IMAGE[inverse[S]], x]]]]] == True

In[30]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[31]:= Map[not, SubstTest[and, implies[and[p0, p1], p2],
            implies[and[p1, p2], p3], not[implies[and[p0, p1], p3]],
            {p0 → not[empty[x]], p1 -> member[t, A[image[BIGCUP, x]]],
             p2 -> member[hull[U[x], set[t]], A[image[IMAGE[inverse[S]], x]]],
             p3 → member[t, U[A[image[IMAGE[inverse[S]], x]]]]}]] // Reverse

Out[31]= or[equal[0, x], member[t, U[A[image[IMAGE[inverse[S]], x]]]],
           not[member[t, A[image[BIGCUP, x]]]]] == True

In[32]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma. Eliminate **t**.

```
In[33]:= Map[equal[V, #] &, SubstTest[class, t, or[equal[0, x], member[t, u], not[member[t, v]]],
           {u → U[A[image[IMAGE[inverse[S]], x]]], v -> A[image[BIGCUP, x]]}]]

Out[33]= or[equal[0, x], subclass[A[image[BIGCUP, x]], U[A[image[IMAGE[inverse[S]], x]]]]] == True

In[34]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. Eliminate a redundant literal.

```
In[35]:= SubstTest[and, implies[p, q], or[p, q],
           {p → empty[x], q -> subclass[A[image[BIGCUP, x]], U[A[image[IMAGE[inverse[S]], x]]]]}]

Out[35]= subclass[A[image[BIGCUP, x]], U[A[image[IMAGE[inverse[S]], x]]]] == True

In[36]:= (% /. x → x_) /. Equal→ SetDelayed
```

The reverse inclusion was derived previously. Combining the two inclusions yields an equation that can be made into a rewrite rule.

Main Theorem. The sum class of the intersection of the class of hereditary hulls of members of any class **x** is the intersection of the class of all sum classes of members of **x**.

```
In[37]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> A[image[BIGCUP, x]], v -> U[A[image[IMAGE[inverse[S]], x]]]}]

Out[37]= equal[A[image[BIGCUP, x]], U[A[image[IMAGE[inverse[S]], x]]]] == True

In[38]:= U[A[image[IMAGE[inverse[S]], x_]]] := A[image[BIGCUP, x]]
```

The variable **x** can be eliminated using **reify**.

Theorem. A variable-free formulation of the main theorem.

```
In[39]:= Map[VERTSECT, SubstTest[reify, x, U[A[image[t, x]]], t → IMAGE[inverse[S]]]]

Out[39]= composite[BIGCUP, BIGCAP, IMAGE[IMAGE[inverse[S]]]] == composite[BIGCAP, IMAGE[BIGCUP]]

In[40]:= composite[BIGCUP, BIGCAP, IMAGE[IMAGE[inverse[S]]]] := composite[BIGCAP, IMAGE[BIGCUP]]
```

The following corollary is a variable-free statement of the result for the special case of two sets, which initiated this entire line of inquiry.

Corollary.

```
In[41]:= Assoc[composite[BIGCUP, BIGCAP], IMAGE[IMAGE[inverse[S]]], PAIRSET]

Out[41]= composite[BIGCUP, CAP, cross[IMAGE[inverse[S]], IMAGE[inverse[S]]]] ==
          composite[CAP, cross[BIGCUP, BIGCUP]]

In[42]:= composite[BIGCUP, CAP, cross[IMAGE[inverse[S]], IMAGE[inverse[S]]]] :=
          composite[CAP, cross[BIGCUP, BIGCUP]]
```

The following corollary seems interesting in view of the open question whether **ACLOSURE** commutes with **UCLOSURE**.

Corollary. The functions **IMAGE[BIGCUP]** and **ACLOSURE** almost commute in a certain sense.

```
In[43]:= Map[VERTSECT, Assoc[composite[BIGCUP, BIGCAP], IMAGE[IMAGE[inverse[S]]], inverse[S]]]

Out[43]= composite[IMAGE[BIGCUP], ACLOSURE, IMAGE[IMAGE[inverse[S]]]] ==
          composite[ACLOSURE, IMAGE[BIGCUP]]

In[44]:= composite[IMAGE[BIGCUP], ACLOSURE, IMAGE[IMAGE[inverse[S]]]] :=
          composite[ACLOSURE, IMAGE[BIGCUP]]
```

serendipity

Theorem. An inclusion.

```
In[45]:= Map[assert, SubstTest[subclass, A[t], U[t], t -> image[BIGCUP, x]] // Reverse]

Out[45]= subclass[A[image[BIGCUP, x]], U[U[x]]] == not[equal[0, x]]

In[46]:= subclass[A[image[BIGCUP, x_]], U[U[x_]]] := not[equal[0, x]]
```