HULL[allclosed[BIGCUP ∘ id[{0}']]]

Johan G. F. Belinfante
2014 March 7

```
In[1]:= SetDirectory["l:"]; << goedel.14mar06a
        :Package Title: goedel.14mar06a                    2014 March 6 at 1:00 p.m.
        Loading takes about seventeen minutes, half that time due to builtin pauses.
        It is now:  2014 Mar 6 at 23:27
        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 Mar 6 at 23:44
```

summary

The function **IRC = λ t. image[RC[U[t]], t]** takes any collection of sets **t** to their relative complements in **U[t]**. In topology, this function takes the collection of open sets of a topological space to the set of closed sets, and vice versa. If **t** is any collection of sets, then **Aclosure[t]** is the collection of all intersections of (nonempty) subsets of **t**, and **Uclosure[t]** is the collection of all unions of subsets of **t**.

```
In[2]:= class[v, exists[u, and[subclass[u, t], equal[v, A[u]]]]]
Out[2]= Aclosure[t]

In[3]:= class[v, exists[u, and[subclass[u, t], equal[v, U[u]]]]]
Out[3]= Uclosure[t]
```

In general when one applies **Aclosure** to **image[RC[U[t]], t]** one does not quite get the **image[RC[U[t]], Uclosure[t]]**, but something very near to it:

```
In[4]:= Aclosure[image[RC[U[t]], t]]
Out[4]= image[RC[U[t]], image[BIGCUP, intersection[complement[set[0]], P[t]]]]
```

The class **image[BIGCUP, P[t] – {0}]** that appears here differs from **Uclosure[t]** by at most the element **0**. If **0 ∈ t**, then these classes are identical, which is the case for the set of open sets or the set of all closed sets of a topological space. But if **0 ∉ t**, then **image[BIGCUP, P[t] – {0}] = Uclosure[t] – {0}**. The class **image[BIGCUP, P[t] – {0}]** can also be expressed as the intersection of **Uclosure[t]** with the class **image[S, t]**.

```
In[5]:= intersection[image[S, t], Uclosure[t]]

Out[5]= image[BIGCUP, intersection[complement[set[0]], P[t]]]
```

In this notebook it is shown that the function **IMAGE[BIGCUP] ∘ IMAGE[id[{0}']] ∘ POWER** that takes any set **t** to the set **image[BIGCUP, P[t] – {0}]** is the **HULL** function of the class **allclosed[BIGCUP ∘ id[{0}']]**. It should be noted that a similar result holds for the more familiar function **UCLOSURE = λ t. Uclosure[t]**.

```
In[6]:= HULL[allclosed[BIGCUP]]

Out[6]= UCLOSURE
```

derivation

Recall that a function is a **HULL** function if it is idempotent, increasing and subcommutes with the subset relation **S**.

```
In[7]:= implies[and[FUNCTION[f], idempotent[f], subclass[f, S], subcommute[f, S]],
          equal[f, HULL[fix[f]]]]

Out[7]= True
```

A function **f** is said to **subcommute** with **S** if **f ∘ S ⊂ S ∘ f**. Here this condition follows immediately from the following more general rule.

Theorem. A general subcommute rule.

```
In[8]:= SubstTest[implies,
          and[subcommute[u, S], subcommute[v, S]], subcommute[composite[u, v], S],
          {u → IMAGE[x], v → composite[IMAGE[y], POWER]}] // Reverse

Out[8]= subclass[composite[IMAGE[x], IMAGE[y], id[range[POWER]], S, id[range[POWER]]],
          composite[S, IMAGE[x], IMAGE[y]]] == True

In[9]:= subclass[composite[IMAGE[x_], IMAGE[y_], id[range[POWER]], S, id[range[POWER]]],
          composite[S, IMAGE[x_], IMAGE[y_]]] := True
```

A function **f** is **increasing** if **f(x) ⊂ x**. This is equivalent to the condition **f ⊂ S**. In the present case, this can be derived using **FastReifNormality**.

Theorem. The increasing property.

```
In[10]:= Map[empty, dif[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER], S] //
           FastReifNormality]

Out[10]= subclass[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER], S] == True

In[11]:= % /. Equal → SetDelayed
```

A function **f** is idempotent if **f(f(x)) = f(x)**, that is, if **f ∘ f = f**. The idempotence of the function **IMAGE[BIGCUP] ∘ IMAGE[id[{0}']] ∘ POWER** will be shown below by considering separately its restrictions to the class of collections that do or do not hold the empty set.

restriction to the class of sets holding 0

If **0 ∈ x** there is no difference at all between **image[BIGCUP, P[x] – {0}]** and **Uclosure[x]**. A variable-free expression of this will now be derived.

Lemma. (Eliminate the variable **x**.)

```
In[12]:= Map[composite[Id, complement[#]] &,
          dif[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER,
              id[complement[P[complement[set[0]]]]]], UCLOSURE] // complement // RelnNormality]
```

```
Out[12]= composite[intersection[complement[UCLOSURE],
           composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]],
          id[complement[P[complement[set[0]]]]]] == 0
```

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

Lemma. (An inclusion.)

```
In[14]:= SubstTest[empty, dif[u, v],
          {u -> composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]],
             POWER, id[complement[P[complement[set[0]]]]], v -> UCLOSURE}]
```

```
Out[14]= subclass[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]],
            POWER, id[complement[P[complement[set[0]]]]], UCLOSURE] == True
```

```
In[15]:= % /. Equal → SetDelayed
```

Theorem. (An equation that can be made into a temporary rewrite rule.)

```
In[16]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u -> composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER,
              id[complement[P[complement[set[0]]]]], v -> UCLOSURE}] // Reverse
```

```
Out[16]= equal[composite[UCLOSURE, id[complement[P[complement[set[0]]]]]],
           composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]],
            POWER, id[complement[P[complement[set[0]]]]]]] == True
```

```
In[17]:= composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]],
           POWER, id[complement[P[complement[set[0]]]]]] :=
          composite[UCLOSURE, id[complement[P[complement[set[0]]]]]]
```

the case $0 \notin x$

When $0 \notin x$ one has **image[BIGCUP, P[x] – {0}] = Uclosure[x] - {0}**. Again, a variable-free formulation will be derived.

Lemma. (Eliminate the variable **x**.)

```
In[18]:= Map[composite[Id, complement[#]] &,
          dif[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER,
              id[P[complement[set[0]]]]], composite[IMAGE[id[complement[set[0]]]],
              UCLOSURE]] // complement // FastReifNormality]
```

```
Out[18]= composite[
           intersection[composite[complement[IMAGE[id[complement[set[0]]]], UCLOSURE],
            composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]],
           id[P[complement[set[0]]]]] == 0
```

```
In[19]:= % /. Equal → SetDelayed
```

Lemma. (An inclusion.)

```
In[20]:= SubstTest[empty, dif[u, v], {u -> composite[IMAGE[BIGCUP],
             IMAGE[id[complement[set[0]]]], POWER, id[P[complement[set[0]]]]],
           v → composite[IMAGE[id[complement[set[0]]]], UCLOSURE]}]
```

```
Out[20]= subclass[composite[IMAGE[BIGCUP],
            IMAGE[id[complement[set[0]]]], POWER, id[P[complement[set[0]]]]],
           composite[IMAGE[id[complement[set[0]]]], UCLOSURE]] == True
```

```
In[21]:= % /. Equal → SetDelayed
```

Theorem. (An equation that can be made into a temporary rewrite rule.)

```
In[22]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]], {u -> composite[IMAGE[BIGCUP],
              IMAGE[id[complement[set[0]]]], POWER, id[P[complement[set[0]]]]],
            v -> composite[IMAGE[id[complement[set[0]]]], UCLOSURE]}] // Reverse
```

```
Out[22]= equal[composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]],
           composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]],
            POWER, id[P[complement[set[0]]]]]] == True
```

```
In[23]:= composite[IMAGE[BIGCUP],
           IMAGE[id[complement[set[0]]]], POWER, id[P[complement[set[0]]]]] :=
          composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]
```

idempotence property

Theorem. Rewrite rule combining the special cases.

```
In[24]:= SubstTest[union, composite[x, id[y]], composite[x, id[complement[y]]],
           {x -> composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER],
            y → P[complement[set[0]]]}] // Reverse

Out[24]= union[composite[UCLOSURE, id[complement[P[complement[set[0]]]]]],
           composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]] ==
         composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]

In[25]:= % /. Equal → SetDelayed
```

Lemma. A simplification rule.

```
In[27]:= composite[id[complement[P[complement[set[0]]]]], UCLOSURE] // FastReifNormality

Out[27]= composite[id[complement[P[complement[set[0]]]]], UCLOSURE] == UCLOSURE

In[28]:= composite[id[complement[P[complement[set[0]]]]], UCLOSURE] := UCLOSURE
```

Theorem. The function **IMAGE[BIGCUP] ∘ IMAGE[id[complement[{0}]]] ∘ POWER** is idempotent.

```
In[30]:= SubstTest[composite, union[u, v], union[u, v],
           {u -> composite[UCLOSURE, id[complement[P[complement[set[0]]]]]], v -> composite[
              IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]}] // Reverse

Out[30]= composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]],
           POWER, IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER] ==
         composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]

In[31]:= % /. Equal → SetDelayed
```

allclosed rules

The class **allclosed[x]** is the collection of all sets **t** that satisfy **image[x, P[t]] ⊂ t**.

```
In[37]:= class[t, subclass[image[x, P[t]], t]]

Out[37]= allclosed[x]
```

Theorem.

```
In[41]:= SubstTest[intersection, fix[composite[S, funpart[t]]],
          fix[composite[inverse[S], funpart[t]]],
          t -> composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]]

Out[41]= fix[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]] ==
          allclosed[composite[BIGCUP, id[complement[set[0]]]]]

In[42]:= % /. Equal → SetDelayed
```

Corollary. A simplification rule.

```
In[43]:= SubstTest[implies, and[FUNCTION[x], idempotent[x]], equal[range[x], fix[x]],
           x -> composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]] // Reverse

Out[43]= equal[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
           image[IMAGE[BIGCUP], image[IMAGE[id[complement[set[0]]]], range[POWER]]]] == True

In[44]:= image[IMAGE[BIGCUP], image[IMAGE[id[complement[set[0]]]], range[POWER]]] :=
          allclosed[composite[BIGCUP, id[complement[set[0]]]]]
```

HULL rule

Theorem. Application of a criterion for a **HULL** function.

```
In[45]:= SubstTest[implies, and[FUNCTION[t], equal[range[t], fix[t]],
             subclass[t, S], subcommute[t, S]], equal[t, HULL[fix[t]]],
           t -> composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER]] // Reverse

Out[45]= equal[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER],
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]] == True

In[46]:= composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER] :=
          HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]
```

Corollary. A simplification rule.

```
In[47]:= IminComp[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]]], POWER, V]

Out[47]= image[inverse[S], allclosed[composite[BIGCUP, id[complement[set[0]]]]]] == V

In[48]:= image[inverse[S], allclosed[composite[BIGCUP, id[complement[set[0]]]]]] := V
```

Lemma. A simplification rule.

```
In[49]:= image[V, set[intersection[complement[set[0]], P[x]]]] // Normality

Out[49]= image[V, set[intersection[complement[set[0]], P[x]]]] == image[V, set[x]]

In[50]:= image[V, set[intersection[complement[set[0]], P[x_]]]] := image[V, set[x]]
```

Theorem. An application rule for a **HULL** function.

```
In[51]:= ApComp[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]]], POWER, x] // Reverse

Out[51]= hull[allclosed[composite[BIGCUP, id[complement[set[0]]]], x] == union[
           complement[image[V, set[x]]], image[BIGCUP, intersection[complement[set[0]], P[x]]]]

In[52]:= hull[allclosed[composite[BIGCUP, id[complement[set[0]]]], x_] := union[
           complement[image[V, set[x]]], image[BIGCUP, intersection[complement[set[0]], P[x]]]]
```

adjoining the empty set

A variable-free reformulation of the following existing rewrite rule will now be derived.

```
In[56]:= union[image[BIGCUP, dif[P[x], set[0]]], set[0]]

Out[56]= Uclosure[x]
```

Theorem.

```
In[54]:= Map[VERTSECT,
          SubstTest[reify, x, union[image[BIGCUP, dif[P[x], set[0]]], t], t → set[0]]]

Out[54]= composite[ADJOIN[set[0]],
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] == UCLOSURE

In[55]:= composite[ADJOIN[set[0]],
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] := UCLOSURE
```

The same result is obtained when one composes these same two functions in the reverse order.

Theorem.

```
In[58]:= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
           ADJOIN[set[0]]] // RelnNormality

Out[58]= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
           ADJOIN[set[0]]] == UCLOSURE

In[59]:= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
           ADJOIN[set[0]]] := UCLOSURE
```

Replacement rules will now be derived for temporary rewrite rules derived above.

Theorem. Formula for the restriction to collections that hold the empty set.

```
In[61]:= Assoc[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]]],
           POWER, id[complement[P[complement[set[0]]]]]] // Reverse

Out[61]= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
           id[complement[P[complement[set[0]]]]] ==
          composite[UCLOSURE, id[complement[P[complement[set[0]]]]]
```

```
In[62]:= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
           id[complement[P[complement[set[0]]]]]] :=
          composite[UCLOSURE, id[complement[P[complement[set[0]]]]]]
```

Theorem. Formula for the restriction to collections that do not hold the empty set.

```
In[64]:= Assoc[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]]],
           POWER, id[P[complement[set[0]]]]] // Reverse

Out[64]= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
           id[P[complement[set[0]]]]] ==
          composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]
```

```
In[65]:= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
           id[P[complement[set[0]]]]] :=
          composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]
```

The rule for combining the two restrictions needs to be replaced. First the old rule needs to be replaced.

```
In[71]:= union[composite[UCLOSURE, id[complement[P[complement[set[0]]]]]],
           composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]] =.
```

Theorem. Replacement rule.

```
In[72]:= SubstTest[composite,
          HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]], union[u, v],
          {u → id[P[complement[set[0]]]], v → id[complement[P[complement[set[0]]]]]}]

Out[72]= union[composite[UCLOSURE, id[complement[P[complement[set[0]]]]]],
           composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]] ==
          HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]
```

```
In[73]:= union[composite[UCLOSURE, id[complement[P[complement[set[0]]]]]],
           composite[IMAGE[id[complement[set[0]]]], UCLOSURE, id[P[complement[set[0]]]]]] :=
          HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]
```

When one composes this with **ADJOIN[{0}]**, one term drops out. The following rewrite rule is needed for that.

Theorem.

```
In[74]:= composite[id[P[complement[set[0]]]], ADJOIN[set[0]]] // RelnNormality

Out[74]= composite[id[P[complement[set[0]]]], ADJOIN[set[0]]] == 0
```

```
In[75]:= composite[id[P[complement[set[0]]]], ADJOIN[set[0]]] := 0
```

other composites

Theorem.

```
In[76]:= Assoc[composite[IMAGE[BIGCUP], IMAGE[id[complement[set[0]]]], POWER],
          id[complement[P[complement[set[0]]]]], UCLOSURE]

Out[76]= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]], UCLOSURE] ==
          UCLOSURE

In[77]:= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]], UCLOSURE] :=
          UCLOSURE
```

Corollary.

```
In[101]:=
        Assoc[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]], UCLOSURE, POWER]

Out[101]=
        composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]], POWER] == POWER

In[102]:=
        composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]], POWER] := POWER
```

Theorem.

```
In[78]:= Assoc[UCLOSURE, ADJOIN[set[0]],
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] // Reverse

Out[78]= composite[UCLOSURE,
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] == UCLOSURE

In[79]:= composite[UCLOSURE,
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] := UCLOSURE
```

Theorem.

```
In[80]:= Map[VERTSECT, SubstTest[reify, x, U[image[u, dif[P[x], v]]], {u → BIGCUP, v → set[0]}]]

Out[80]= composite[BIGCUP, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] == BIGCUP

In[81]:= composite[BIGCUP, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]] := BIGCUP
```

Theorem.

```
In[85]:= Assoc[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
          id[P[complement[set[0]]]], IMAGE[id[complement[set[0]]]]

Out[85]= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
           IMAGE[id[complement[set[0]]]] == composite[IMAGE[id[complement[set[0]]]], UCLOSURE]

In[86]:= composite[HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
           IMAGE[id[complement[set[0]]]] := composite[IMAGE[id[complement[set[0]]]], UCLOSURE]
```

IRC rule

Lemma.

```
In[89]:= symdif[composite[ACLOSURE, IRC], composite[IRC,
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]]] // FastReifNormality

Out[89]= union[intersection[composite[ACLOSURE, IRC], composite[complement[IRC],
           HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]], intersection[
          composite[IRC, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
          composite[complement[ACLOSURE], IRC]]] == 0

In[90]:= % /. Equal → SetDelayed
```

Theorem.

```
In[91]:= SubstTest[empty, symdif[u, v], {u -> composite[ACLOSURE, IRC],
           v -> composite[IRC, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]]}]

Out[91]= equal[composite[ACLOSURE, IRC],
           composite[IRC, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]]] == True

In[92]:= composite[IRC, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]] :=
          composite[ACLOSURE, IRC]
```

Corollary.

```
In[103]:=
        Assoc[IRC, HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]],
         id[complement[P[complement[set[0]]]]]]

Out[103]=
        composite[IRC, UCLOSURE, id[complement[P[complement[set[0]]]]]] ==
         composite[ACLOSURE, IRC, id[complement[P[complement[set[0]]]]]]

In[104]:=
        composite[IRC, UCLOSURE, id[complement[P[complement[set[0]]]]]] :=
         composite[ACLOSURE, IRC, id[complement[P[complement[set[0]]]]]]
```

more allclosed rules

Theorem.

```
In[94]:= allclosed[composite[BIGCUP, id[P[complement[set[0]]]]]] // Normality

Out[94]= allclosed[composite[BIGCUP, id[P[complement[set[0]]]]]] == fix[UCLOSURE]

In[95]:= allclosed[composite[BIGCUP, id[P[complement[set[0]]]]]] := fix[UCLOSURE]
```

Theorem.

```
In[96]:= SubstTest[allclosed, union[u, v],
          {u -> composite[BIGCUP, id[intersection[FINITE, chains[S]]]],
           v -> composite[BIGCUP, id[complement[set[0]]]]}]

Out[96]= intersection[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
          complement[P[complement[set[0]]]] == fix[UCLOSURE]

In[97]:= intersection[allclosed[composite[BIGCUP, id[complement[set[0]]]]],
          complement[P[complement[set[0]]]] := fix[UCLOSURE]
```

image[S,x] ⋂ Uclosure[x]

Theorem. A variable-free rule for the intersection of **image[S, x]** with **Uclosure[x]**.

```
In[98]:= Map[VERTSECT, SubstTest[reify, x, intersection[image[s, x], Uclosure[x]], s → S]]

Out[98]= composite[IMAGE[SECOND], IMAGE[id[S]], CART, id[UCLOSURE], inverse[FIRST]] ==
          HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]

In[99]:= composite[IMAGE[SECOND], IMAGE[id[S]], CART, id[UCLOSURE], inverse[FIRST]] :=
          HULL[allclosed[composite[BIGCUP, id[complement[set[0]]]]]]
```