image[inverse[S], binclosed[thinpart[x]]]

Johan G. F. Belinfante
2011 January 24

```
In[1]:= SetDirectory["l:"]; << goedel.11jan20a
```

```
:Package Title: goedel.11jan20a           2011 January 20 at 6:20 p.m.

It is now:  2011 Jan 24 at 5:8

Loading Simplification Rules

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

weightlimit = 40
```

summary

It is shown in this notebook that any set is a subset of a set that is binary closed under any given thin relation.

introduction

Hrbacek and Jech discuss closure under any set of ***n***-ary operations on page 61 of the following book.

```
In[2]:= "Karel Hrbacek and Thomas Jech,Introduction to Set
         Theory, Third Edition, Marcel Dekker, Inc., New York, 1999.";
```

The derivation presented in this notebook is an adaptation of their proof, but all that will be considered here is closure under a set of binary operations. It is convenient to replace the set of binary operations by a single relation. Since the constructor **binclosed** has the following property, all one needs to do is replace the set of binary operations by its sum class.

```
In[3]:= binclosed[union[x, y]]
Out[3]= intersection[binclosed[x], binclosed[y]]
```

Lemma. (A compound wrapper removal rule.)

```
In[4]:= equal[x, intersection[y, setpart[x]]] // AssertTest
Out[4]= equal[x, intersection[y, setpart[x]]] == and[member[x, V], subclass[x, y]]

In[5]:= equal[x_, intersection[y_, setpart[x_]]] := and[member[x, V], subclass[x, y]]
```

Theorem. The sum class of any set of functions is a thin relation.

```
In[6]:= Map[implies[member[x, y], #] &,
          SubstTest[implies, equal[x, intersection[FUNS, setpart[t]]],
           equal[V, domain[VERTSECT[U[x]]]], t → x]] // Reverse

Out[6]= or[equal[V, domain[VERTSECT[U[x]]]], not[member[x, y]], not[subclass[x, FUNS]]] == True

In[7]:= or[equal[V, domain[VERTSECT[U[x_]]]],
          not[member[x_, y_]], not[subclass[x_, FUNS]]] := True
```

---

an abbreviation

The basic idea in the proof given by Hrbacek and Jech is to consider the following iteration, for which a temporary abbreviation will be introduced to save some writing.

```
In[8]:= it[x_, y_] := iterate[composite[CUP,
           id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]], set[setpart[y]]]
```

The class **it[x, y]** is a function, and its range is a **hull** expression.

Theorem. A formula for **range[it[x, y]]**.

```
In[9]:= SubstTest[range, iterate[funpart[t], set[y]],
          t -> composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]] // Reverse

Out[9]= range[
          iterate[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]], set[y]]] ==
         hull[invar[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]], set[y]]

In[10]:= range[iterate[composite[CUP,
             id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST]], set[y_]]] := hull[
           invar[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]], set[y]]
```

The following theorem implies that **setpart[y] ⊂ U[range[it[x, y]]**.

Theorem.

```
In[11]:= SubstTest[implies, subclass[u, v], subclass[U[u], U[v]],
           {u → set[setpart[y]], v → hull[x, set[setpart[y]]]}] // Reverse

Out[11]= subclass[setpart[y], U[hull[x, set[setpart[y]]]]] == True

In[12]:= subclass[setpart[y_], U[hull[x_, set[setpart[y_]]]]] := True
```

The function **CUP ∘ id[IMAGE[x] ∘ CART ∘ DUP] ∘ inverse[FIRST]** is a function that is monotone, increasing, and sucommutes with the subset relaton.

```
In[13]:= {FUNCTION[#], subclass[P[#], monotone[S, S]], subclass[#, S], subcommute[#, S]} & @
          composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]

Out[13]= {True, True, True, True}
```

The function **it[x, y]** is monotone with respect to inclusion.

Theorem. **P[it[x, y]] ⊂ monotone[S,S]**.

```
In[14]:= SubstTest[implies, and[FUNCTION[u], subclass[u, S]],
           subclass[composite[iterate[u, set[v]], S, inverse[iterate[u, set[v]]]], S],
           {u → composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]],
            v → y}] // Reverse

Out[14]= subclass[composite[
            iterate[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]], set[y]],
            S, inverse[iterate[composite[CUP,
               id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]], set[y]]]], S] == True

In[15]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

APPLY rules

Theorem. A vertical section rule: **image[it[x,y], {z}]] = {APPLY[it[x,y], z]}**.

```
In[16]:= SubstTest[image, iterate[funpart[t], set[y]], set[z],
           t -> composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]] // Reverse

Out[16]= image[iterate[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]],
            set[y]], set[z]] == set[APPLY[iterate[
             composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]], set[y]], z]]

In[17]:= image[iterate[composite[CUP, id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST]],
            set[y_]], set[z_]] := set[APPLY[iterate[
             composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]], set[y]], z]]
```

Theorem. The domain of **it[x, y]** is ω.

```
In[18]:= SubstTest[implies, and[member[setpart[y], domain[funpart[t]]],
            invariant[funpart[t], domain[funpart[t]]]],
           equal[omega, domain[iterate[funpart[t], set[setpart[y]]]]], t -> composite[
             CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]]] // Reverse

Out[18]= equal[omega,
           domain[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
              inverse[FIRST]], set[setpart[y]]]]] == True

In[19]:= domain[iterate[composite[CUP, id[composite[IMAGE[thinpart[x_]], CART, DUP]],
             inverse[FIRST]], set[setpart[y_]]]] := omega
```

Theorem. The values of **it[x, y]** belong to its range.

```
In[20]:= SubstTest[subclass, image[u, v], range[u], {u → it[x, y], v → set[nat[i]]}] // Reverse

Out[20]= member[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]], nat[i]],
           hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]]], set[setpart[y]]]] == True

In[21]:= (% /. {x → x_, y → y_, i → i_}) /. Equal → SetDelayed
```

---

induction step

Lemma.

```
In[22]:= Map[A, SubstTest[image, iterate[t, set[y]], set[succ[z]], t → funpart[x]]] // Reverse

Out[22]= APPLY[iterate[funpart[x], set[y]], succ[z]] ==
          APPLY[funpart[x], APPLY[iterate[funpart[x], set[y]], z]]

In[23]:= APPLY[iterate[funpart[x_], set[y_]], succ[z_]] :=
          APPLY[funpart[x], APPLY[iterate[funpart[x], set[y]], z]]
```

The each value of **it[x, y]** is a subset of the next.

Theorem. **APPLY[it[x, y], nat[i]] ⊂ APPLY[it[x, y], succ[nat[i]]]**.

```
In[24]:= Map[
           subclass[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[y]], nat[i]], #] &,
           SubstTest[APPLY, iterate[funpart[t], set[y]], succ[nat[i]], t -> composite[
             CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]]]] // Reverse

Out[24]= subclass[APPLY[iterate[composite[CUP,
             id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]], set[y]], nat[i]],
           APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[y]], succ[nat[i]]]] == True

In[25]:= (% /. {x → x_, y → y_, i → i_}) /. Equal → SetDelayed
```

A more general result holds.

Theorem.

```
In[26]:= SubstTest[implies, and[subclass[P[t], monotone[S, S]],
            subclass[u, v], member[u, domain[t]]], subclass[APPLY[t, u], APPLY[t, v]],
           {t → it[x, y], u → nat[i], v → natadd[nat[i], nat[j]]}] // Reverse

Out[26]= subclass[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]], nat[i]],
           APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]], natadd[nat[i], nat[j]]]] == True
```

```
In[27]:= (% /. {x → x_, y → y_, i → i_, j → j_}) /. Equal → SetDelayed
```

binary closure

Theorem. **image[thinpart[x], cartsq[APPLY[it[x, y], nat[i]]]] ⊂ APPLY[it[x, y], succ[nat[i]]]**.

```
In[28]:= Map[subclass[#,
            APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]], set[setpart[y]]], succ[nat[i]]]] &,
          SubstTest[APPLY, iterate[funpart[t], set[setpart[y]]], succ[nat[i]],
           t -> composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]]]]
```

```
Out[28]= subclass[image[thinpart[x],
            cart[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], nat[i]],
             APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], nat[i]]]],
           APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]], succ[nat[i]]]] == True
```

```
In[29]:= (% /. {x → x_, y → y_, i → i_}) /. Equal → SetDelayed
```

Theorem. **APPLY[it[x, y], nat[i]] ⊂ U[range[it[x, y]]]**.

```
In[30]:= SubstTest[implies, member[u, v], subclass[u, U[v]],
           {u -> APPLY[it[x, y], nat[i]], v -> range[it[x, y]]}] // Reverse
```

```
Out[30]= subclass[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]], nat[i]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
              inverse[FIRST]]], set[setpart[y]]]]] == True
```

```
In[31]:= (% /. {x → x_, y → y_, i → i_}) /. Equal → SetDelayed
```

Corollary.

```
In[32]:= SubstTest[subclass, APPLY[it[x, y], nat[j]],
           U[range[it[x, y]]], j → succ[nat[i]]] // Reverse
```

```
Out[32]= subclass[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]], succ[nat[i]]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
              inverse[FIRST]]], set[setpart[y]]]]] == True
```

```
In[33]:= (% /. {x → x_, y → y_, i → i_}) /. Equal → SetDelayed
```

Corollary. **image[thinpart[x], cartsq[APPLY[it[x, y], nat[i]]]] ⊂ U[range[it[x, y]]]**.

```
In[34]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u -> image[thinpart[x], cartsq[APPLY[it[x, y], nat[i]]]],
            v -> APPLY[it[x, y], succ[nat[i]]], w → U[range[it[x, y]]]}] // Reverse

Out[34]= subclass[image[thinpart[x],
            cart[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], nat[i]],
             APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], nat[i]]]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]]], set[setpart[y]]]]] == True

In[35]:= (% /. {x → x_, y → y_, i → i_}) /. Equal→ SetDelayed
```

---

polarization

Lemma. **image[thinpart[x], cartsq[APPLY[it[x, y], nat[i] + nat[j]]]] ⊂ U[range[it[x, y]]]**.

```
In[36]:= SubstTest[subclass, image[thinpart[x], cartsq[APPLY[it[x, y], nat[k]]]],
           U[range[it[x, y]]], k → natadd[nat[i], nat[j]]] // Reverse

Out[36]= subclass[image[thinpart[x],
            cart[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], natadd[nat[i], nat[j]]],
             APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], natadd[nat[i], nat[j]]]]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]]], set[setpart[y]]]]] == True

In[37]:= (% /. {x → x_, y → y_, i → i_, j → j_}) /. Equal→ SetDelayed
```

Polarization Lemma. **image[thinpart[x], cart[APPLY[it[x,y],nat[i]], APPLY[it[x,y],nat[j]]]] ⊂ U[range[it[x,y]]]**.

```
In[38]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → image[thinpart[x], cart[APPLY[it[x, y], nat[i]], APPLY[it[x, y], nat[j]]]],
            v → image[thinpart[x], cartsq[APPLY[it[x, y], natadd[nat[i], nat[j]]]]],
            w → U[range[it[x, y]]]}] // Reverse

Out[38]= subclass[image[thinpart[x],
            cart[APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], nat[i]],
             APPLY[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]], nat[j]]]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]]], set[setpart[y]]]]] == True

In[39]:= (% /. {x → x_, y → y_, i → i_, j → j_}) /. Equal→ SetDelayed
```

eliminating i and j

Observation.

```
In[40]:= class[pair[i, j], subclass[image[t, cart[i, j]], u]]

Out[40]= LB[LB[complement[image[inverse[t], complement[u]]]]]
```

Lemma.

```
In[41]:= SubstTest[subclass, image[inverse[s], cart[u, v]],
          complement[t], {s → inverse[x], t → complement[y]}]

Out[41]= equal[0, intersection[v, image[image[inverse[x], complement[y]], u]]] ==
          subclass[image[x, cart[u, v]], y]

In[42]:= equal[0, intersection[v_, image[image[inverse[x_], complement[y_]], u_]]] :=
          subclass[image[x, cart[u, v]], y]
```

Lemma.

```
In[43]:= (member[pair[u, v], composite[inverse[funpart[t]], z, funpart[t]]] // AssertTest) /.
          {t → it[x, y], u → nat[i], v → nat[j],
           z -> LB[LB[complement[image[inverse[thinpart[x]], complement[
                 U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                     inverse[FIRST]]], set[setpart[y]]]]]]]]]}

Out[43]= member[pair[nat[i], nat[j]], composite[
           inverse[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
              inverse[FIRST]], set[setpart[y]]]],
           LB[LB[complement[image[inverse[thinpart[x]], complement[
                U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                    inverse[FIRST]]], set[setpart[y]]]]]]]],
           iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[setpart[y]]]]] == True

In[44]:= (% /. {x → x_, y → y_, i → i_, j → j_}) /. Equal → SetDelayed
```

Lemma. (Removing the **nat** wrappers.)

```
In[45]:= SubstTest[implies, and[equal[u, nat[i]], equal[v, nat[j]]],
           member[pair[u, v], composite[inverse[iterate[
              composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]],
              set[setpart[y]]]], LB[LB[complement[image[inverse[thinpart[x]], complement[
                 U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                     inverse[FIRST]]], set[setpart[y]]]]]]]]],
             iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
              inverse[FIRST]], set[setpart[y]]]]], {i → u, j → v}] // Reverse

Out[45]= or[member[pair[u, v], composite[
            inverse[iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]], set[setpart[y]]]],
            LB[LB[complement[image[inverse[thinpart[x]], complement[
                 U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                     inverse[FIRST]]], set[setpart[y]]]]]]]]],
            iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
              inverse[FIRST]], set[setpart[y]]]]],
           not[member[u, omega]], not[member[v, omega]]] ==
          True

In[46]:= (% /. {x → x_, y → y_, u → u_, v → v_}) /. Equal→ SetDelayed
```

The following derivation takes a minute or so.

Main theorem. **U[range[it[x, y]]] ∈ binclosed[thinpart[x]]**.

```
In[47]:= Map[empty[range[complement[#]]] &,
          SubstTest[class, pair[u, v], implies[member[pair[u, v], w], member[pair[u, v], z]],
           {w -> cart[omega, omega], z -> composite[inverse[iterate[
                composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]],
                set[setpart[y]]]], LB[LB[complement[image[inverse[thinpart[x]],
                  complement[U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]],
                         CART, DUP]], inverse[FIRST]]], set[setpart[y]]]]]]]]],
              iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]], set[setpart[y]]]]}]]

Out[47]= subclass[image[thinpart[x],
             cart[U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                  inverse[FIRST]]], set[setpart[y]]]],
              U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                  inverse[FIRST]]], set[setpart[y]]]]],
            U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]]], set[setpart[y]]]]] == True

In[48]:= (% /. x → x_) /. Equal→ SetDelayed
```

The details of the construction are not important. The following corollary generally suffices.

Corollary.

```
In[49]:= SubstTest[implies, and[subclass[u, v], member[v, w]], member[u, image[inverse[S], w]],
           {u → setpart[y], v → U[range[it[x, y]]], w -> binclosed[thinpart[x]]}] // Reverse

Out[49]= member[setpart[y], image[inverse[S], binclosed[thinpart[x]]]] == True

In[50]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The variable **y** can be eliminated.

Theorem. Every set is a subset of a set that is binary closed under any given thin relation.

```
In[51]:= SubstTest[class, y, member[setpart[y], z],
           z -> image[inverse[S], binclosed[thinpart[x]]]]

Out[51]= image[inverse[S], binclosed[thinpart[x]]] == V

In[52]:= image[inverse[S], binclosed[thinpart[x_]]] := V
```

---

the binary closure

For any set **y**, the set **hull[binclosed[thinpart[x], y]** is called its **binary closure** with respect to **thinpart[x]**.

Theorem. Any binary closed set is its own binary closure.

```
In[53]:= Map[implies[member[y, z], #] &, SubstTest[implies, member[y, t],
           equal[y, hull[t, y]], t → binclosed[thinpart[x]]]] // Reverse

Out[53]= or[equal[y, hull[binclosed[thinpart[x]], y]],
           not[member[y, z]], not[subclass[image[thinpart[x], cart[y, y]], y]]] == True

In[54]:= or[equal[y_, hull[binclosed[thinpart[x_]], y_]], not[member[y_, z_]],
           not[subclass[image[thinpart[x_], cart[y_, y_]], y_]]] := True
```

The following inclusion will later be strengthened to an equation.

Lemma. **hull[binclosed[thinpart[x], setpart[y]] ⊂ U[range[it[x, y]]**.

```
In[55]:= SubstTest[subclass, hull[t, u], hull[t, v], {t → binclosed[thinpart[x]], u → setpart[y],
           v -> U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]]], set[setpart[y]]]]}] // Reverse

Out[55]= subclass[hull[binclosed[thinpart[x]], setpart[y]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]]], set[setpart[y]]]]] == True

In[56]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The binary closure of a set is binary closed.

```
In[57]:= SubstTest[member, hull[t, setpart[y]],
           fix[HULL[t]], t → binclosed[thinpart[x]]] // Reverse

Out[57]= subclass[image[thinpart[x], cart[hull[binclosed[thinpart[x]], setpart[y]],
             hull[binclosed[thinpart[x]], setpart[y]]]],
           hull[binclosed[thinpart[x]], setpart[y]]] == True

In[58]:= subclass[image[thinpart[x_], cart[hull[binclosed[thinpart[x_]], setpart[y_]],
             hull[binclosed[thinpart[x_]], setpart[y_]]]],
           hull[binclosed[thinpart[x_]], setpart[y_]]] := True
```

Corollary. If one starts the iteration with the binary closure, the iteration is a constant function.

```
In[59]:= SubstTest[implies, and[FUNCTION[t], member[u, fix[t]]],
           equal[cart[omega, set[u]], iterate[t, set[u]]],
           {t -> composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]],
            u → hull[binclosed[thinpart[x]], setpart[y]]}] // Reverse

Out[59]= equal[cart[omega, set[hull[binclosed[thinpart[x]], setpart[y]]]],
           iterate[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
             inverse[FIRST]], set[hull[binclosed[thinpart[x]], setpart[y]]]]] == True

In[60]:= iterate[composite[CUP, id[composite[IMAGE[thinpart[x_]], CART, DUP]], inverse[FIRST]],
           set[hull[binclosed[thinpart[x_]], setpart[y_]]]] :=
          cart[omega, set[hull[binclosed[thinpart[x]], setpart[y]]]]
```

Lemma. Inclusion in the reverse direction.

```
In[61]:= SubstTest[implies, and[subclass[u, v], subcommute[t, S]],
           subclass[iterate[t, set[v]], composite[S, iterate[t, set[u]]]],
           {t → composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]],
            u → setpart[y], v → hull[binclosed[thinpart[x]], setpart[y]]}] // Reverse

Out[61]= subclass[U[hull[invar[
              composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]]],
             set[setpart[y]]]], hull[binclosed[thinpart[x]], setpart[y]]] == True

In[62]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The two inclusions are combined to derive an equation.

Theorem. **U[range[it[x, y]] = hull[binclosed[thinpart[x], setpart[y]]]**.

```
In[63]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
                inverse[FIRST]]], set[setpart[y]]]],
           v -> hull[binclosed[thinpart[x]], setpart[y]]}]

Out[63]= equal[hull[binclosed[thinpart[x]], setpart[y]],
           U[hull[invar[composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]],
               inverse[FIRST]]], set[setpart[y]]]]] == True
```

```
In[64]:= U[hull[invar[
             composite[CUP, id[composite[IMAGE[thinpart[x_]], CART, DUP]], inverse[FIRST]]],
           set[setpart[y_]]]] := hull[binclosed[thinpart[x]], setpart[y]]
```

A variable-free formulation can be obtained using reification.

Theorem.

```
In[65]:= Map[VERTSECT, SubstTest[reify, y, U[hull[t, set[setpart[y]]]], t -> invar[
             composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]]]]]
```

```
Out[65]= composite[BIGCUP, HULL[invar[
             composite[CUP, id[composite[IMAGE[thinpart[x]], CART, DUP]], inverse[FIRST]]],
           SINGLETON]] == HULL[binclosed[thinpart[x]]]
```

```
In[66]:= composite[BIGCUP, HULL[invar[
             composite[CUP, id[composite[IMAGE[thinpart[x_]], CART, DUP]], inverse[FIRST]]],
           SINGLETON]] := HULL[binclosed[thinpart[x]]]
```

Corollary. A special case.

```
In[68]:= SubstTest[composite, BIGCUP, HULL[invar[
             composite[CUP, id[composite[IMAGE[thinpart[t]], CART, DUP]], inverse[FIRST]]],
           SINGLETON, t→ composite[x, inverse[DUP]]] // Reverse
```

```
Out[68]= composite[BIGCUP, HULL[invar[composite[CUP, id[IMAGE[thinpart[x]]], inverse[FIRST]]],
           SINGLETON]] == HULL[invar[thinpart[x]]]
```

```
In[70]:= composite[BIGCUP,
           HULL[invar[composite[CUP, id[IMAGE[thinpart[x_]]], inverse[FIRST]]]],
           SINGLETON]] := HULL[invar[thinpart[x]]]
```