ub[x, lb[x, y]] as a hull operation

Johan G. F. Belinfante
2004 February 27

In[1]:=
```
<< goedel54.23a; << tools.m
```

```
:Package Title: goedel54.23a     2004 February 23 at 6:00 a.m.

It is now:  2004 Feb 27 at 15:38

Loading Simplification Rules

TOOLS.M                      Revised 2004 February 21

weightlimit = 40
```

summary

It is shown in this notebook that the operation of taking the upper bound of the lower bound is a hull operation. The function under consideration is:

In[2]:=
```
class[pair[y, z], equal[z, ub[w, lb[w, y]]]] /. w -> composite[Id, x]
```

Out[2]=
```
VERTSECT[complement[composite[complement[x], LB[x]]]]
```

The primary tool is a characterization of **HULL[x]** that is reminiscent of Kuratowski's characterization of closure operators in topology. (For details, see the notebook **CHARHULL.NB**.)

derivation

Increasing property.

In[3]:=
```
Map[equal[0, #] &,
  dif[VERTSECT[complement[composite[complement[x], LB[x]]]], S] // VSRenormality]
```

Out[3]=
```
subclass[VERTSECT[complement[composite[complement[x], LB[x]]]], S] == True
```

In[4]:=
```
subclass[VERTSECT[complement[composite[complement[x_], LB[x_]]]], S] := True
```

Idempotence.

In[5]:=
```
(composite[VERTSECT[complement[composite[complement[y], LB[y]]]],
     VERTSECT[complement[composite[complement[y], LB[y]]]]] //
    VSNormality) /. y -> composite[Id, x]
```

Out[5]=
```
composite[VERTSECT[complement[composite[complement[x], LB[x]]]],
  VERTSECT[complement[composite[complement[x], LB[x]]]]] ==
 VERTSECT[complement[composite[complement[x], LB[x]]]]
```

```
In[6]:= composite[VERTSECT[complement[composite[complement[x_], LB[x_]]]],
          VERTSECT[complement[composite[complement[x_], LB[x_]]]]] :=
         VERTSECT[complement[composite[complement[x], LB[x]]]]
```

Corollary.

```
In[7]:= SubstTest[implies, and[FUNCTION[y], equal[composite[y, y], y]],
         equal[range[y], fix[y]], y -> VERTSECT[complement[composite[complement[x], LB[x]]]]]

Out[7]= equal[fix[VERTSECT[complement[composite[complement[x], LB[x]]]]],
          range[VERTSECT[complement[composite[complement[x], LB[x]]]]]] == True

In[8]:= range[VERTSECT[complement[composite[complement[x_], LB[x_]]]]] :=
         fix[VERTSECT[complement[composite[complement[x], LB[x]]]]]
```

monotone property

A new rewrite rule is needed for the **GOEDEL** program to be able to recognize that the combination of the two antitone operations **ub** and **lb** is monotone.

```
In[9]:= SubstTest[subclass, lb[complement[w], ub[complement[w], y]],
         lb[complement[w], ub[complement[w], z]],
            w -> complement[x]]

Out[9]= subclass[cart[lb[x, ub[x, y]], ub[x, z]], x] == subclass[cart[y, ub[x, z]], x]

In[10]:= subclass[cart[lb[x_, ub[x_, y_]], ub[x_, z_]], x_] := subclass[cart[y, ub[x, z]], x]

In[11]:= SubstTest[subclass, cart[lb[w, ub[w, y]], ub[w, z]], w, w -> inverse[x]]

Out[11]= subclass[cart[lb[x, z], ub[x, lb[x, y]]], x] == subclass[cart[lb[x, z], y], x]

In[12]:= subclass[cart[lb[x_, z_], ub[x_, lb[x_, y_]]], x_] := subclass[cart[lb[x, z], y], x]
```

hereditary property of domain

Lemma.

```
In[13]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
            {u -> ub[x, lb[x, y]], v -> ub[x, lb[x, z]]}]

Out[13]= or[member[ub[x, lb[x, y]], V],
           not[member[ub[x, lb[x, z]], V]], not[subclass[cart[lb[x, z], y], x]]] == True

In[14]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma.

```
In[15]:= ub[x, intersection[y, image[V, z]]] // Normality

Out[15]= ub[x, intersection[y, image[V, z]]] == union[complement[image[V, z]], ub[x, y]]

In[16]:= ub[x_, intersection[y_, image[V, z_]]] := union[complement[image[V, z]], ub[x, y]]
```

Lemma.

```
In[17]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[p1, p4], implies[and[p2, p4], p5], implies[and[p3, p5], p6],
                 not[implies[and[p1, p2], p6]], {p1 -> subclass[y, z],
            p2 -> member[z, domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]],
                 p3 -> member[y, V], p4 -> subclass[ub[x, lb[x, y]], ub[x, lb[x, z]]],
            p5 -> member[ub[x, lb[x, y]], V],
                 p6 -> member[y, domain[VERTSECT[
                complement[composite[complement[x], LB[x]]]]]]}]]

Out[17]= or[member[ub[x, lb[x, y]], V], not[member[z, V]],
           not[member[ub[x, lb[x, z]], V]], not[subclass[y, z]]] == True

In[18]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma.

```
In[19]:= or[and[member[y, V], member[ub[x, lb[x, y]], V]], not[member[z, V]],
           not[member[ub[x, lb[x, z]], V]], not[subclass[y, z]]] // NotNotTest

Out[19]= or[and[member[y, V], member[ub[x, lb[x, y]], V]], not[member[z, V]],
           not[member[ub[x, lb[x, z]], V]], not[subclass[y, z]]] == True

In[20]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

The vaiables **y** and **z** are eliminated:

```
In[21]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class,
            pair[y, z], implies[and[subclass[y, z], member[z, w]], member[y, w]],
                 w -> domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]]] // Reverse

Out[21]= subclass[
           image[inverse[S], domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]],
           domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]] == True

In[22]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The main result of this section is the following. This will later be replaced by a different rewrite rule.

```
In[23]:= equal[
           image[inverse[S], domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]],
           domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]] // AssertTest

Out[23]= equal[domain[VERTSECT[complement[composite[complement[x], LB[x]]]]], image[
            inverse[S], domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]]] == True

In[24]:= (% /. x -> x_) /. Equal -> SetDelayed
```

the monotonicity property

The monotonicity property was the most difficult to establish. The following procedure finally succeeded:

```
In[25]:= composite[id[complement[S]],
           cross[VERTSECT[complement[composite[complement[x], LB[x]]]],
            VERTSECT[complement[composite[complement[x], LB[x]]]]], id[S]] // TriRenormality

Out[25]= composite[cross[VERTSECT[complement[composite[complement[x], LB[x]]]],
            VERTSECT[complement[composite[complement[x], LB[x]]]]], id[composite[intersection[
              S, composite[inverse[VERTSECT[complement[composite[complement[x], LB[x]]]]],
               complement[E], complement[composite[complement[x], LB[x]]]]],
             id[domain[VERTSECT[complement[composite[complement[x], LB[x]]]]]]]]] == 0
```

```
In[26]:= (% /. x -> x_) /. Equal -> SetDelayed
```

To recognize this result, it is useful to transform it as follows:

```
In[27]:= SubstTest[equal, 0, composite[id[complement[S]], cross[w, w], id[S]],
           w -> VERTSECT[complement[composite[complement[x], LB[x]]]]] // Reverse

Out[27]= subclass[composite[VERTSECT[complement[composite[complement[x], LB[x]]]], S,
            inverse[VERTSECT[complement[composite[complement[x], LB[x]]]]]], S] == True

In[28]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The hereditary property of the domain can be incorporated into this formula as follows:

```
In[29]:= SubstTest[implies, and[equal[domain[w], image[inverse[S], domain[w]]],
           FUNCTION[w], subclass[composite[w, S, inverse[w]], S]],
          subclass[composite[w, S], composite[S, w]],
          w -> VERTSECT[complement[composite[complement[x], LB[x]]]]]

Out[29]= subclass[composite[VERTSECT[complement[composite[complement[x], LB[x]]]], S],
           composite[S, VERTSECT[complement[composite[complement[x], LB[x]]]]]] == True

In[30]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The theorem characterizing **HULL** functions can now be applied:

```
In[31]:= SubstTest[implies,
          and[FUNCTION[w], equal[composite[w, w], w], subclass[w, S], subcommute[w, S]],
          equal[w, HULL[fix[w]]], w -> VERTSECT[complement[composite[complement[x], LB[x]]]]]

Out[31]= equal[HULL[fix[VERTSECT[complement[composite[complement[x], LB[x]]]]]],
           VERTSECT[complement[composite[complement[x], LB[x]]]]] == True

In[32]:= HULL[fix[VERTSECT[complement[composite[complement[x_], LB[x_]]]]]] :=
          VERTSECT[complement[composite[complement[x], LB[x]]]]
```

---

some corollaries

The domain of this functon is the hereditary closure of its range.

```
In[33]:= SubstTest[domain, HULL[w],
          w -> fix[VERTSECT[complement[composite[complement[x], LB[x]]]]]]

Out[33]= domain[VERTSECT[complement[composite[complement[x], LB[x]]]]] ==
          image[inverse[S], fix[VERTSECT[complement[composite[complement[x], LB[x]]]]]]

In[34]:= domain[VERTSECT[complement[composite[complement[x_], LB[x_]]]]] :=
          image[inverse[S], fix[VERTSECT[complement[composite[complement[x], LB[x]]]]]]
```

A similar result holds for lower bounds of upper bounds.

```
In[35]:= SubstTest[HULL,
          fix[VERTSECT[complement[composite[complement[y], LB[y]]]]], y -> inverse[x]]

Out[35]= HULL[fix[VERTSECT[complement[composite[complement[inverse[x]], UB[x]]]]]] ==
          VERTSECT[complement[composite[complement[inverse[x]], UB[x]]]]

In[36]:= HULL[fix[VERTSECT[complement[composite[complement[inverse[x_]], UB[x_]]]]]] :=
          VERTSECT[complement[composite[complement[inverse[x]], UB[x]]]]
```

an example

For the case **x = inverse[S]**, the function under consideration is:

```
In[41]:= VERTSECT[complement[composite[complement[x], LB[x]]]] /. x → inverse[S]

Out[41]= composite[POWER, BIGCUP]
```

This is the **HULL** of its fixed point class:

```
In[44]:= HULL[range[POWER]]

Out[44]= composite[POWER, BIGCUP]
```