The functions GLB[DIV] and LUB[DIV]

Johan G. F. Belinfante
2005 October 5

```
In[1]:= SetDirectory["l:"]; << goedel74.02c; << tools.m

        :Package Title: goedel74.02c            2005 October 2 at 8:15 p.m.

        It is now:  2005 Oct 5 at 8:37

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 October 2

        weightlimit = 40
```

summary

Divisibility is usually studied using the domain of integers instead of the set of natural numbers in order to take advantage of the fact that the integers form a principal ideal domain. The principal ideal corresponding to the greatest lower bound of a nonempty set of integers is the intersection of the set of principal ideals generated by the members of that set. In this notebook a rather similar theory is developed, but always staying within the framework of the natural numbers. Formulas for the domain and range of the functions **GLB[DIV]** and **LUB[DIV]** are derived. It is shown that every set of natural numbers has a greatest lower bound by explicitly constructing it. It follows that set of the natural numbers, ordered by divisibility, forms a complete lattice. The formula for the domain of **LUB[DIV]** is derived from a general fact about complete lattices. The explicit construction of greatest lower bounds yields simple formulas connecting the functions **GLB[DIV]** and **HULL[image[VERTSECT[DIV],omega]]**. This connection can be considered to be the analog for natural numbers of the usual characterization of greatest lower bounds of a set of numbers in terms of the principal ideal generated by that set.

range of GLB[DIV] and LUB[DIV]

The range formulas follow directly from general results applicable to any partial order relation.

```
In[2]:= SubstTest[range, GLB[po[x]], x → DIV]

Out[2]= range[GLB[DIV]] == omega

In[3]:= range[GLB[DIV]] := omega

In[4]:= SubstTest[range, LUB[po[x]], x → DIV]

Out[4]= range[LUB[DIV]] == omega

In[5]:= range[LUB[DIV]] := omega
```

an upper bound for domain[GLB[DIV]]

An upper bound for **domain[GLB[DIV]]** follows from a general result.

```
In[6]:= SubstTest[subclass, domain[GLB[x]], domain[LB[x]], x → DIV]
Out[6]= subclass[U[domain[GLB[DIV]]], omega] == True
In[7]:= % /. Equal → SetDelayed
```

range[VERTSECT[DIV]] and image[VERTSECT[DIV],omega]

Although the set **omega** is not a principal ideal domain, it behaves rather similarly. The natural analog for **omega** of the principal ideals of the domain **Z** of integers are the sets of the form **image[DIV,set[x]]**. If **x** is a natural number, the set **image[DIV,set[x]]** holds all multiples of **x**. When **x** is not a natural number, one obtains the empty set. The members of the set **range[VERTSECT[DIV]]** are all these sets, including the empty set. The empty set is omitted by working with **image[VERTSECT[DIV], omega]**.

```
In[8]:= member[image[DIV, set[x]], image[VERTSECT[DIV], omega]] // AssertTest
Out[8]= member[image[DIV, set[x]], image[VERTSECT[DIV], omega]] == member[x, omega]
In[9]:= member[image[DIV, set[x_]], image[VERTSECT[DIV], omega]] := member[x, omega]
```

Both of the sets **range[VERTSECT[DIV]]** and **image[VERTSECT[DIV], omega]** are closed under arbitrary intersections. They are related as follows:

```
In[10]:= union[set[0], image[VERTSECT[DIV], omega]]
Out[10]= range[VERTSECT[DIV]]
```

HULL[range[VERTSECT[DIV]]] and HULL[image[VERTSECT[DIV],omega]]

Both of the sets are closed under arbitrary intersections. The set **hull[image[VERTSECT[DIV], omega], x]** is the intersection of all sets of the form **image[DIV,set[y]]**, where **y** is a natural number, that contain a given set **x** of natural numbers. (This hull is akin to the principal ideal generated by a set of integers.) Note that the hull is a set because any set of natural numbers is contained in at least one of the sets **image[DIV,set[y]]** , namely the set of multiples of **1 = set[0]**.

These two functions have the same domain, but different ranges.

```
In[11]:= domain[HULL[range[VERTSECT[DIV]]]]
Out[11]= P[omega]
```

```
In[12]:= domain[HULL[image[VERTSECT[DIV], omega]]]

Out[12]= P[omega]
```

Lemma.

```
In[13]:= Assoc[HULL[image[VERTSECT[DIV], omega]], id[P[omega]], id[x]]

Out[13]= composite[HULL[image[VERTSECT[DIV], omega]], id[intersection[x, P[omega]]]] ==
          composite[HULL[image[VERTSECT[DIV], omega]], id[x]]

In[14]:= composite[HULL[image[VERTSECT[DIV], omega]], id[intersection[x_, P[omega]]]] :=
          composite[HULL[image[VERTSECT[DIV], omega]], id[x]]
```

The two functions agree on all nonempty sets.

```
In[15]:= Map[composite[VERTSECT[reify[x, #]], id[complement[set[0]]]] &,
          SubstTest[hull, union[u, v], x, {u → image[VERTSECT[DIV], omega], v → set[0]}]]

Out[15]= composite[HULL[range[VERTSECT[DIV]]], id[complement[set[0]]]] ==
          composite[HULL[image[VERTSECT[DIV], omega]], id[complement[set[0]]]]
```

The only difference is the value assigned to the empty set:

```
In[16]:= APPLY[HULL[range[VERTSECT[DIV]]], 0]

Out[16]= 0

In[17]:= APPLY[HULL[image[VERTSECT[DIV], omega]], 0]

Out[17]= set[0]
```

temporary rewrite rules

It is easy to write either one of the two functions in terms of the other. In this notebook the function **HULL[image[-VERTSECT[DIV],omega]]** is favored because it is the one that appears in the final results.

```
In[18]:= SubstTest[implies, and[member[x, domain[y]], FUNCTION[y]],
          member[APPLY[y, x], range[y]], y → HULL[image[VERTSECT[DIV], omega]]]

Out[18]= or[member[hull[image[VERTSECT[DIV], omega], x], image[VERTSECT[DIV], omega]],
           not[subclass[x, omega]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Conversely:

```
In[20]:= Map[or[#, subclass[x, omega]] &, SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u -> hull[image[VERTSECT[DIV], omega], x],
            v -> image[VERTSECT[DIV], omega], w → P[omega]}]]

Out[20]= or[not[member[hull[image[VERTSECT[DIV], omega], x], image[VERTSECT[DIV], omega]]],
           subclass[x, omega]] == True

In[21]:= (% /. x → x_) /. Equal → SetDelayed
```

These results can be combined:

```
In[22]:= equiv[member[hull[image[VERTSECT[DIV], omega], x], image[VERTSECT[DIV], omega]],
           subclass[x, omega]]

Out[22]= True

In[23]:= member[hull[image[VERTSECT[DIV], omega], x_], image[VERTSECT[DIV], omega]] :=
           subclass[x, omega]
```

Lemma.

```
In[24]:= hull[set[0], x] // Normality

Out[24]= hull[set[0], x] == image[V, x]

In[25]:= hull[set[0], x_] := image[V, x]
```

The following temporary rewrite rules force the elimination of hulls involving **range[VERTSECT[DIV]]** in favor of those for **image[VERTSECT[DIV], omega]**.

```
In[26]:= SubstTest[hull, union[u, v], x, {u → image[VERTSECT[DIV], omega], v → set[0]}]

Out[26]= hull[range[VERTSECT[DIV]], x] ==
           intersection[hull[image[VERTSECT[DIV], omega], x], image[V, x]]

In[27]:= hull[range[VERTSECT[DIV]], x_] :=
           intersection[hull[image[VERTSECT[DIV], omega], x], image[V, x]]

In[28]:= Map[VERTSECT, SubstTest[reify, x, hull[y, x], y → range[VERTSECT[DIV]]]] // Reverse

Out[28]= HULL[range[VERTSECT[DIV]]] == union[cart[set[0], set[0]],
           composite[HULL[image[VERTSECT[DIV], omega]], id[complement[set[0]]]]]

In[29]:= HULL[range[VERTSECT[DIV]]] := union[cart[set[0], set[0]],
           composite[HULL[image[VERTSECT[DIV], omega]], id[complement[set[0]]]]]
```

equal[0,hull[x,y]]

This section is concerned with a general characterization of the condition that a hull is empty. Two lemmas are needed:

```
In[30]:= Map[implies[#, equal[0, y]] &,
          SubstTest[and, subclass[y, z], equal[0, z], z → hull[x, y]]]

Out[30]= or[equal[0, y], not[equal[0, hull[x, y]]]] == True

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

In[32]:= Map[not,
          SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> equal[0, hull[x, y]], p2 → equal[0, y], p3 → equal[0, A[x]]}]]

Out[32]= or[equal[0, A[x]], not[equal[0, hull[x, y]]]] == True

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

This yields the following rewrite rule:

```
In[34]:= equiv[equal[0, hull[x, y]], and[equal[0, y], equal[A[x], 0]]] // not // not

Out[34]= True

In[35]:= equal[0, hull[x_, y_]] := and[equal[0, y], equal[0, A[x]]]
```

Corollary 1.

```
In[36]:= image[inverse[HULL[x]], set[0]] // Renormality

Out[36]= image[inverse[HULL[x]], set[0]] == intersection[complement[image[V, A[x]]], set[0]]

In[37]:= image[inverse[HULL[x_]], set[0]] := intersection[complement[image[V, A[x]]], set[0]]
```

Corollary 2.

```
In[38]:= equal[composite[id[complement[set[0]]], HULL[image[VERTSECT[DIV], omega]]],
          HULL[image[VERTSECT[DIV], omega]]]

Out[38]= True

In[39]:= composite[id[complement[set[0]]], HULL[image[VERTSECT[DIV], omega]]] :=
          HULL[image[VERTSECT[DIV], omega]]
```

---

using hulls to construct greatest lower bounds

Lemma. The set **hull[[image[VERTSECT[DIV], omega], x]** is contained in any upper bound **y** for the set **x**.

```
In[40]:= SubstTest[implies, member[u, v], subclass[A[v], u],
          {v → intersection[image[VERTSECT[DIV], omega], image[S, set[x]]],
           u → image[DIV, set[y]]}]

Out[40]= or[not[member[x, V]], not[member[y, omega]], not[subclass[x, image[DIV, set[y]]]],
           subclass[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]] == True
```

```
In[41]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The above statement contains a redundant literal:

```
In[42]:= SubstTest[implies, and[subclass[x, z], member[z, V]],
          member[x, V], z → image[DIV, set[y]]]

Out[42]= or[member[x, V], not[subclass[x, image[DIV, set[y]]]]] == True
```

```
In[43]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the redundant literal yields:

```
In[44]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → and[member[y, omega], subclass[x, image[DIV, set[y]]]], p2 → member[x, V],
            p3 -> subclass[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]}]]

Out[44]= or[not[member[y, omega]], not[subclass[x, image[DIV, set[y]]]],
           subclass[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]] == True
```

```
In[45]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Restatement:

```
In[46]:= implies[member[z, intersection[omega, lb[DIV, x]]],
          subclass[hull[range[VERTSECT[DIV]], x], image[DIV, set[z]]]]

Out[46]= True
```

Lemma.

```
In[47]:= Map[or[not[member[pair[y, z], DIV]], #] &,
          SubstTest[implies, and[subclass[x, u], subclass[u, v]],
           subclass[x, v], {u -> image[DIV, set[z]], v → image[DIV, set[y]]}]]

Out[47]= or[not[member[pair[y, z], DIV]],
           not[subclass[x, image[DIV, set[z]]]], subclass[x, image[DIV, set[y]]]] == True
```

```
In[48]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. If the hull of a set **x** is the set of multiples of a natural number **y**, and if every member of **x** is divisible by **z**, then **z** divides **y**.

```
In[49]:= Map[not, SubstTest[and, implies[and[p2, p4], p5],
           implies[and[p1, p5, p6], p3], not[implies[and[p1, p2, p4, p6], p3]],
           {p1 → equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]],
            p2 → member[z, omega], p3 → member[pair[z, y], DIV],
            p4 -> subclass[x, image[DIV, set[z]]],
            p5 -> subclass[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[z]]],
            p6 → member[y, omega]}]]

Out[49]= or[member[pair[z, y], DIV],
           not[equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]],
           not[member[y, omega]], not[member[z, omega]],
           not[subclass[x, image[DIV, set[z]]]]] == True

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

Restatement, removing the variable **z**.

```
In[51]:= Map[equal[V, #] &, SubstTest[class, z,
            implies[and[member[y, omega], equal[r, s], member[z, u]], member[z, v]],
            {r -> hull[image[VERTSECT[DIV], omega], x], s -> image[DIV, set[y]],
             u → intersection[omega, lb[DIV, x]], v -> image[inverse[DIV], set[y]]}]] // Reverse

Out[51]= or[not[equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]],
           not[member[y, omega]],
           subclass[intersection[omega, lb[DIV, x]], image[inverse[DIV], set[y]]]] == True

In[52]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Conversely, if the hull of a set **x** is the set of multiples of a natural number **y**, and if **z** divides **y**, then **z** divides every member of **x**.

```
In[53]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[and[p2, p3], p4], not[implies[and[p1, p3], p4]],
           {p1 → equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]],
            p2 → subclass[x, image[DIV, set[y]]], p3 → member[pair[z, y], DIV],
            p4 -> subclass[x, image[DIV, set[z]]]}]]

Out[53]= or[not[equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]],
           not[member[pair[z, y], DIV]], subclass[x, image[DIV, set[z]]]] == True

In[54]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

removing the remaining variables

Lemma.

```
In[55]:= ImageComp[inverse[VERTSECT[DIV]], VERTSECT[DIV], omega] // Reverse

Out[55]= image[inverse[VERTSECT[DIV]], image[VERTSECT[DIV], omega]] == omega

In[56]:= image[inverse[VERTSECT[DIV]], image[VERTSECT[DIV], omega]] := omega
```

```
In[57]:= composite[inverse[VERTSECT[DIV]], id[image[VERTSECT[DIV], omega]]] // DoubleInverse

Out[57]= composite[inverse[VERTSECT[DIV]], id[image[VERTSECT[DIV], omega]]] ==
          composite[id[omega], inverse[VERTSECT[DIV]]]

In[58]:= composite[inverse[VERTSECT[DIV]], id[image[VERTSECT[DIV], omega]]] :=
          composite[id[omega], inverse[VERTSECT[DIV]]]
```

Lemma.

```
In[59]:= or[and[subclass[x, image[DIV, set[y]]],
             subclass[intersection[omega, lb[DIV, x]], image[inverse[DIV], set[y]]]],
           not[equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]],
           not[member[x, V]], not[member[y, omega]]] // NotNotTest

Out[59]= or[and[subclass[x, image[DIV, set[y]]],
             subclass[intersection[omega, lb[DIV, x]], image[inverse[DIV], set[y]]]],
           not[equal[hull[image[VERTSECT[DIV], omega], x], image[DIV, set[y]]]],
           not[member[x, V]], not[member[y, omega]]] == True

In[60]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[61]:= Map[implies[equal[0, composite[Id, complement[#]]], subclass[composite[id[omega],
            inverse[VERTSECT[DIV]], HULL[image[VERTSECT[DIV], omega]]], GLB[DIV]]] &,
          SubstTest[class, pair[x, y], implies[and[member[x, V], member[y, omega],
            equal[hull[r, x], image[v, set[y]]]], member[pair[x, y], z]],
           {r → image[VERTSECT[DIV], omega], v → DIV, z → GLB[DIV]}]]

Out[61]= subclass[composite[inverse[VERTSECT[DIV]],
            HULL[image[VERTSECT[DIV], omega]]], GLB[DIV]] == True

In[62]:= % /. Equal → SetDelayed
```

domain formulas

The hull construction yields a lower bound for **domain[GLB[DIV]]**.

```
In[63]:= SubstTest[implies, subclass[u, v], subclass[domain[u], domain[v]],
          {u -> composite[id[omega], inverse[VERTSECT[DIV]], HULL[image[VERTSECT[DIV], omega]]],
           v -> GLB[DIV]}]

Out[63]= subclass[P[omega], domain[GLB[DIV]]] == True

In[64]:= subclass[P[omega], domain[GLB[DIV]]] := True
```

The final step is to combine this result with the upper bound found earlier. This results in an equation that is made into a rewrite rule.

```
In[65]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → domain[GLB[DIV]], v → P[omega]}]

Out[65]= True == equal[domain[GLB[DIV]], P[omega]]

In[66]:= domain[GLB[DIV]] := P[omega]
```

It follows from this that the natural numbers, ordered by divisibility, forms a complete lattice. Since **DIV** is a complete lattice, one can get the formula for the domain of **LUB[DIV]** as a corollary of the formula for **domain[GLB[DIV]]**.

```
In[67]:= SubstTest[implies, and[member[x, PO], equal[domain[GLB[x]], P[fix[x]]]],
          equal[domain[LUB[x]], P[fix[x]]], x → DIV]

Out[67]= equal[domain[LUB[DIV]], P[omega]] == True

In[68]:= domain[LUB[DIV]] := P[omega]
```

formulas relating GLB[DIV] to HULL[image[VERTSECT[DIV],omega]]

Lemma.

```
In[69]:= Assoc[GLB[DIV], id[P[omega]], id[x]]

Out[69]= composite[GLB[DIV], id[intersection[x, P[omega]]]] == composite[GLB[DIV], id[x]]

In[70]:= composite[GLB[DIV], id[intersection[x_, P[omega]]]] := composite[GLB[DIV], id[x]]
```

Any subclass of a function is a restriction.

```
In[71]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
          {u -> composite[id[omega], inverse[VERTSECT[DIV]], HULL[image[VERTSECT[DIV], omega]]],
           v → GLB[DIV]}]

Out[71]= equal[composite[inverse[VERTSECT[DIV]], HULL[image[VERTSECT[DIV], omega]]],
           GLB[DIV]] == True

In[72]:= composite[inverse[VERTSECT[DIV]], HULL[image[VERTSECT[DIV], omega]]] := GLB[DIV]
```

Lemma.

```
In[73]:= Assoc[id[range[VERTSECT[DIV]]],
          id[image[VERTSECT[DIV], omega]], HULL[image[VERTSECT[DIV], omega]]]

Out[73]= composite[id[range[VERTSECT[DIV]]], HULL[image[VERTSECT[DIV], omega]]] ==
          HULL[image[VERTSECT[DIV], omega]]

In[74]:= composite[id[range[VERTSECT[DIV]]], HULL[image[VERTSECT[DIV], omega]]] :=
          HULL[image[VERTSECT[DIV], omega]]
```

Corollary.

```
In[75]:= Assoc[VERTSECT[DIV], inverse[VERTSECT[DIV]], HULL[image[VERTSECT[DIV], omega]]]
Out[75]= composite[VERTSECT[DIV], GLB[DIV]] == HULL[image[VERTSECT[DIV], omega]]

In[76]:= composite[VERTSECT[DIV], GLB[DIV]] := HULL[image[VERTSECT[DIV], omega]]
```

serendipity

The following results were discovered in the course of this work, but were not used in the preceding sections.

```
In[77]:= equal[0, lb[setpart[x], V]] // AssertTest
Out[77]= equal[0, lb[setpart[x], V]] == True

In[78]:= lb[setpart[x_], V] := 0
```

In particular:

```
In[79]:= SubstTest[lb, setpart[x], V, x → DIV]
Out[79]= lb[DIV, V] == 0

In[80]:= lb[DIV, V] := 0
```

Dual results:

```
In[81]:= equal[0, ub[setpart[x], V]] // AssertTest
Out[81]= equal[0, ub[setpart[x], V]] == True

In[85]:= ub[setpart[x_], V] := 0

In[86]:= SubstTest[ub, setpart[x], V, x → DIV]
Out[86]= ub[DIV, V] == 0

In[87]:= ub[DIV, V] := 0
```