range[VERTSECT[INTDIV]]

Johan G. F. Belinfante
2011 November 14

```
In[1]:= SetDirectory["l:"]; << goedel.11nov12a

        :Package Title: goedel.11nov12a              2011 November 12 at 8:30 p.m.

        Loading takes about thirteen minutes, half that time due to builtin pauses.

        It is now:  2011 Nov 14 at 14:45

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2011 Nov 14 at 14:58
```

summary

The main result derived in this notebook is that the range of every subgroup of the integers under addition has the form **image[INTDIV, {x}]** where **x** is an integer. That is, the ranges of the subgroups of **INTADD** are precisely the non-empty vertical sections of the integer divisibility relation **INTDIV**. Much of this notebook is concerned with the connection between the divisibility relation **DIV** for natural numbers, the divisibility relation **INTDIV** for integers and the mixed divisibility relation **MIXDIV**. Of particular interest is how their vertical section functions are related.

derivation

The following variable-free statement is already available, but it might strike one as being somewhat cryptic in that it introduces the mixed divisibility relation.

```
In[2]:= union[composite[PLUS, DIV], composite[INVERSE, PLUS, DIV]]

Out[2]= MIXDIV
```

A more transparent result is obtained by examining vertical sections.

Theorem. A direct connection between vertical sections of **DIV** and **INTDIV** that does not involve **MIXDIV**.

```
In[3]:= SubstTest[image, union[u, v], set[x],
         {u -> composite[PLUS, DIV], v -> composite[INVERSE, PLUS, DIV]}]

Out[3]= union[image[INVERSE, image[PLUS, image[DIV, set[x]]]],
          image[PLUS, image[DIV, set[x]]]] == image[INTDIV, set[plus[x]]]

In[4]:= union[image[INVERSE, image[PLUS, image[DIV, set[x_]]]],
          image[PLUS, image[DIV, set[x_]]]] := image[INTDIV, set[plus[x]]]
```

VERTSECT formulas

Two different formulas for **VERTSECT[MIXDIV]** can be derived, one involving **DIV** and the other **INTDIV**.

Theorem. A formula for **VERTSECT[MIXDIV]** involving the divisibility relation **DIV** for natural numbers.

```
In[5]:= SubstTest[VERTSECT, union[u, v],
         {u -> composite[PLUS, DIV], v -> composite[INVERSE, PLUS, DIV]}]

Out[5]= composite[HULL[invar[INVERSE]], IMAGE[PLUS], VERTSECT[DIV]] == VERTSECT[MIXDIV]

In[6]:= composite[HULL[invar[INVERSE]], IMAGE[PLUS], VERTSECT[DIV]] := VERTSECT[MIXDIV]
```

Theorem. A formula for **VERTSECT[MIXDIV]** involving the divisibility relation **INTDIV** for integers.

```
In[7]:= Map[composite[#, id[omega]] &,
         SubstTest[VERTSECT, composite[setpart[x], setpart[y]], {x → INTDIV, y → PLUS}]]

Out[7]= composite[VERTSECT[INTDIV], PLUS] == composite[VERTSECT[MIXDIV], id[omega]]

In[8]:= composite[VERTSECT[INTDIV], PLUS] := composite[VERTSECT[MIXDIV], id[omega]]
```

Lemma. A temporary simplification rule.

```
In[9]:= Map[image[#, range[PLUS]] &,
         SubstTest[VERTSECT, composite[INTDIV, thinpart[t]], t → INVERSE]]

Out[9]= image[VERTSECT[INTDIV], image[INVERSE, range[PLUS]]] ==
         image[VERTSECT[INTDIV], range[PLUS]]

        (% /. x → x_) /. Equal → SetDelayed
```

Lemma. A simplification rule.

```
In[11]:= SubstTest[image, VERTSECT[INTDIV], union[u, v],
          {u -> range[PLUS], v -> image[INVERSE, range[PLUS]]}]

Out[11]= image[VERTSECT[INTDIV], range[PLUS]] == image[VERTSECT[INTDIV], Z]

In[12]:= image[VERTSECT[INTDIV], range[PLUS]] := image[VERTSECT[INTDIV], Z]
```

Theorem. A simplification rule.

```
In[13]:= ImageComp[VERTSECT[INTDIV], PLUS, V]

Out[13]= image[VERTSECT[MIXDIV], omega] == image[VERTSECT[INTDIV], Z]

In[14]:= image[VERTSECT[MIXDIV], omega] := image[VERTSECT[INTDIV], Z]
```

Corollary. A simplification rule.

```
In[15]:= SubstTest[union, set[0], image[VERTSECT[setpart[x]], domain[setpart[x]]], x → MIXDIV]

Out[15]= range[VERTSECT[MIXDIV]] == range[VERTSECT[INTDIV]]

In[16]:= range[VERTSECT[MIXDIV]] := range[VERTSECT[INTDIV]]
```

Theorem.

```
In[17]:= ImageComp[composite[HULL[invar[INVERSE]], IMAGE[PLUS]], VERTSECT[DIV], V] // Reverse

Out[17]= image[HULL[invar[INVERSE]], image[IMAGE[PLUS], range[VERTSECT[DIV]]]] ==
          range[VERTSECT[INTDIV]]

In[18]:= image[HULL[invar[INVERSE]], image[IMAGE[PLUS], range[VERTSECT[DIV]]]] :=
          range[VERTSECT[INTDIV]]
```

Theorem.

```
In[19]:= ImageComp[composite[HULL[invar[INVERSE]], IMAGE[PLUS]],
           VERTSECT[DIV], omega] // Reverse

Out[19]= image[HULL[invar[INVERSE]], image[IMAGE[PLUS], image[VERTSECT[DIV], omega]]] ==
          image[VERTSECT[INTDIV], Z]

In[20]:= image[HULL[invar[INVERSE]], image[IMAGE[PLUS], image[VERTSECT[DIV], omega]]] :=
          image[VERTSECT[INTDIV], Z]
```

---

a similar result

This section is not used in the sequel, and could be skipped over if desired. The results derived here are similar to those in the previous section, but involve the function **HULL[fix[IMAGE[INVERSE]]]** instead of **HULL[invar[INVERSE]]**.

Lemma.

```
In[21]:= Assoc[id[P[P[cart[V, V]]]], id[range[IMAGE[PLUS]]], IMAGE[PLUS]]

Out[21]= composite[id[P[P[cart[V, V]]]], IMAGE[PLUS]] == IMAGE[PLUS]

In[22]:= composite[id[P[P[cart[V, V]]]], IMAGE[PLUS]] := IMAGE[PLUS]
```

Lemma.

```
In[23]:= Assoc[HULL[invar[INVERSE]], id[P[P[cart[V, V]]]],
           composite[IMAGE[PLUS], VERTSECT[DIV]]] // Reverse

Out[23]= composite[HULL[fix[IMAGE[INVERSE]]], IMAGE[PLUS], VERTSECT[DIV]] == VERTSECT[MIXDIV]

In[24]:= composite[HULL[fix[IMAGE[INVERSE]]], IMAGE[PLUS], VERTSECT[DIV]] := VERTSECT[MIXDIV]
```

Theorem.

```
In[25]:= ImageComp[HULL[fix[IMAGE[INVERSE]]],
           composite[IMAGE[PLUS], VERTSECT[DIV]], V] // Reverse

Out[25]= image[HULL[fix[IMAGE[INVERSE]]], image[IMAGE[PLUS], range[VERTSECT[DIV]]]] ==
          range[VERTSECT[INTDIV]]

In[26]:= image[HULL[fix[IMAGE[INVERSE]]], image[IMAGE[PLUS], range[VERTSECT[DIV]]]] :=
          range[VERTSECT[INTDIV]]
```

Theorem.

```
In[27]:= ImageComp[HULL[fix[IMAGE[INVERSE]]],
           composite[IMAGE[PLUS], VERTSECT[DIV]], omega] // Reverse

Out[27]= image[HULL[fix[IMAGE[INVERSE]]], image[IMAGE[PLUS], image[VERTSECT[DIV], omega]]] ==
          image[VERTSECT[INTDIV], Z]

In[28]:= image[HULL[fix[IMAGE[INVERSE]]], image[IMAGE[PLUS], image[VERTSECT[DIV], omega]]] :=
          image[VERTSECT[INTDIV], Z]
```

---

formulas for subsets of Z invariant under INVERSE

Any subset of **Z** that is invariant under **INVERSE** can be reconstructed from its intersection with **range[PLUS]**.

Lemma.

```
In[29]:= Map[implies[subclass[image[INVERSE, x], x], #] &,
           SubstTest[subclass, image[INVERSE, intersection[x, t]],
            intersection[x, image[INVERSE, t]], t → image[INVERSE, range[PLUS]]]] // Reverse

Out[29]= or[not[subclass[image[INVERSE, x], x]], subclass[
            image[INVERSE, intersection[x, image[INVERSE, range[PLUS]]]], range[PLUS]]] == True

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

Theorem.

```
In[31]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
          {p -> and[subclass[x, Z], subclass[image[INVERSE, x], x]],
           u -> image[INVERSE, intersection[x, range[PLUS]]],
           v -> intersection[x, image[INVERSE, range[PLUS]]]}]

Out[31]= or[equal[image[INVERSE, intersection[x, range[PLUS]]],
            intersection[x, image[INVERSE, range[PLUS]]]],
           not[subclass[x, Z]], not[subclass[image[INVERSE, x], x]]] == True

In[32]:= (% /. x → x_) /. Equal → SetDelayed

In[33]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           not[implies[p1, p3]], {p1 -> and[subclass[x, Z], subclass[image[INVERSE, x], x]],
            p2 -> equal[image[INVERSE, intersection[x, range[PLUS]]],
              intersection[x, image[INVERSE, range[PLUS]]]],
            p3 → equal[x, union[image[INVERSE, intersection[x, range[PLUS]]],
               intersection[x, range[PLUS]]]]}]] // Reverse

Out[33]= or[equal[x,
            union[image[INVERSE, intersection[x, range[PLUS]]], intersection[x, range[PLUS]]]],
           not[subclass[x, Z]], not[subclass[image[INVERSE, x], x]]] == True

In[34]:= or[equal[x_, union[
            image[INVERSE, intersection[x_, range[PLUS]]], intersection[x_, range[PLUS]]]],
           not[subclass[x_, Z]], not[subclass[image[INVERSE, x_], x_]]] := True
```

Theorem.

```
In[35]:= Map[A, ImageComp[composite[CUP, id[IMAGE[INVERSE]]], inverse[FIRST], set[setpart[x]]]]

Out[35]= hull[invar[INVERSE], setpart[x]] == union[image[INVERSE, setpart[x]], setpart[x]]

In[36]:= hull[invar[INVERSE], setpart[x_]] := union[image[INVERSE, setpart[x]], setpart[x]]
```

Corollary.

```
In[37]:= SubstTest[hull, invar[INVERSE], setpart[t], t → intersection[y, setpart[x]]] // Reverse

Out[37]= hull[invar[INVERSE], intersection[y, setpart[x]]] ==
          union[image[INVERSE, intersection[y, setpart[x]]], intersection[y, setpart[x]]]

In[38]:= hull[invar[INVERSE], intersection[y_, setpart[x_]]] :=
          union[image[INVERSE, intersection[y, setpart[x]]], intersection[y, setpart[x]]]
```

Lemma.

```
In[39]:= (member[u, fix[v]] // AssertTest) /.
          {u → setpart[x], v -> composite[HULL[invar[INVERSE]], IMAGE[id[y]]]}

Out[39]= member[setpart[x], fix[composite[HULL[invar[INVERSE]], IMAGE[id[y]]]]] ==
          equal[setpart[x],
           union[image[INVERSE, intersection[y, setpart[x]]], intersection[y, setpart[x]]]]
```

```
In[40]:= member[setpart[x_], fix[composite[HULL[invar[INVERSE]], IMAGE[id[y_]]]]] :=
          equal[setpart[x],
           union[image[INVERSE, intersection[y, setpart[x]]], intersection[y, setpart[x]]]]

In[41]:= Map[equal[V, #] &, SubstTest[class, x,
           not[member[setpart[x], t]], t -> dif[intersection[invar[INVERSE], P[Z]],
             fix[composite[HULL[invar[INVERSE]], IMAGE[id[range[PLUS]]]]]]]]

Out[41]= subclass[intersection[invar[INVERSE], P[Z]],
           fix[composite[HULL[invar[INVERSE]], IMAGE[id[range[PLUS]]]]]] == True

In[42]:= subclass[intersection[invar[INVERSE], P[Z]],
           fix[composite[HULL[invar[INVERSE]], IMAGE[id[range[PLUS]]]]]] := True
```

Theorem.

```
In[43]:= SubstTest[implies, and[FUNCTION[x], subclass[y, fix[x]]],
           equal[composite[x, id[y]], id[y]],
           {x -> composite[HULL[invar[INVERSE]], IMAGE[id[range[PLUS]]]],
            y -> intersection[invar[INVERSE], P[Z]]}] // Reverse

Out[43]= equal[composite[HULL[invar[INVERSE]],
            IMAGE[id[range[PLUS]]], id[intersection[invar[INVERSE], P[Z]]]],
           id[intersection[invar[INVERSE], P[Z]]]] == True

In[44]:= composite[HULL[invar[INVERSE]], IMAGE[id[range[PLUS]]],
           id[intersection[invar[INVERSE], P[Z]]]] := id[intersection[invar[INVERSE], P[Z]]]
```

Corollary.

```
In[45]:= ImageComp[composite[HULL[invar[INVERSE]], IMAGE[id[range[PLUS]]]],
           id[intersection[invar[INVERSE], P[Z]]], x] // Reverse

Out[45]= image[HULL[invar[INVERSE]],
           image[IMAGE[id[range[PLUS]]], intersection[x, invar[INVERSE], P[Z]]]] ==
          intersection[x, invar[INVERSE], P[Z]]

In[46]:= image[HULL[invar[INVERSE]],
           image[IMAGE[id[range[PLUS]]], intersection[x_, invar[INVERSE], P[Z]]]] :=
          intersection[x, invar[INVERSE], P[Z]]
```

subgroups of INTADD

In this section it is shown that the range of every subgroup of **INTADD** has the form **image[INTDIV, {int[x]}]**. The idea is to deduce this fact from a corresponding theorem about natural numbers. The natural numbers do not form a group under addition, but the following result is known:

```
In[47]:= intersection[binclosed[NATADD], binclosed[rotate[NATADD]], P[omega]]

Out[47]= range[VERTSECT[DIV]]
```

The function **PLUS ∈ binhom[NATADD, INTADD]** will be used to transform each of the quantities in this rewrite rule to integer counterparts. In fact most of these translations have already been derived earlier, and all that remains at this point is to do the translation for the term **range[VERTSECT[DIV]]**. Even in this case, an inclusion is already available, and it is only necessary to strengthen this to an equation.

Lemma. A temporary simplification rule.

```
In[48]:= SubstTest[subclass, image[t, intersection[u, v]],
           intersection[image[t, u], image[t, v]],
           {t -> IMAGE[inverse[PLUS]], u → binclosed[rotate[INTADD]], v → P[Z]}] // Reverse

Out[48]= subclass[
           image[IMAGE[inverse[PLUS]], intersection[binclosed[INTADD], invar[INVERSE], P[Z]]],
           image[IMAGE[inverse[PLUS]], binclosed[composite[INTADD, cross[Id, INVERSE]]]]] == True

In[49]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[50]:= SubstTest[subclass, image[t, intersection[u, v, w]],
           intersection[image[t, u], image[t, v], image[t, w]], {t -> IMAGE[inverse[PLUS]],
            u → binclosed[rotate[INTADD]], v → P[Z], w → binclosed[INTADD]}] // Reverse

Out[50]= subclass[image[IMAGE[inverse[PLUS]],
            intersection[binclosed[INTADD], invar[INVERSE], P[Z]]], binclosed[NATADD]] == True

In[51]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[52]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u -> image[IMAGE[inverse[PLUS]],
              intersection[binclosed[INTADD], invar[INVERSE], P[Z]]],
            v -> image[IMAGE[inverse[PLUS]], binclosed[rotate[INTADD]]],
            w -> binclosed[rotate[NATADD]]}] // Reverse

Out[52]= subclass[
           image[IMAGE[inverse[PLUS]], intersection[binclosed[INTADD], invar[INVERSE], P[Z]]],
           binclosed[rotate[NATADD]]] == True

In[53]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[54]:= SubstTest[subclass, t, intersection[u, v, w], {t -> image[
              IMAGE[inverse[PLUS]], intersection[binclosed[INTADD], invar[INVERSE], P[Z]]],
            u -> binclosed[NATADD], v → binclosed[rotate[NATADD]], w → P[omega]}] // Reverse

Out[54]= subclass[image[IMAGE[inverse[PLUS]],
            intersection[binclosed[INTADD], invar[INVERSE], P[Z]]], range[VERTSECT[DIV]]] == True

In[55]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[56]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → composite[HULL[invar[INVERSE]], IMAGE[PLUS]],
            u -> image[IMAGE[inverse[PLUS]], intersection[binclosed[INTADD],
               invar[INVERSE], P[Z]]], v -> range[VERTSECT[DIV]]}] // Reverse

Out[56]= subclass[intersection[binclosed[INTADD], invar[INVERSE], P[Z]],
           range[VERTSECT[INTDIV]]] == True

In[57]:= % /. Equal→ SetDelayed
```

Main theorem.

```
In[58]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[binclosed[INTADD], invar[INVERSE], P[Z]],
           v -> range[VERTSECT[INTDIV]]}]

Out[58]= equal[intersection[binclosed[INTADD], invar[INVERSE], P[Z]],
           range[VERTSECT[INTDIV]]] == True

In[59]:= intersection[binclosed[INTADD], invar[INVERSE], P[Z]] := range[VERTSECT[INTDIV]]
```

Corollary. The ranges of the subgroups of **INTADD** are the non-empty vertical sections of **INTDIV**.

```
In[60]:= AssInt[intersection[binclosed[INTADD], invar[INVERSE]], P[Z], complement[set[0]]]

Out[60]= image[IMAGE[SECOND], intersection[GROUPS, P[INTADD]]] == image[VERTSECT[INTDIV], Z]

In[61]:= image[IMAGE[SECOND], intersection[GROUPS, P[INTADD]]] := image[VERTSECT[INTDIV], Z]
```