binclosed[rotate[gp[x]]]

Johan G. F. Belinfante
2011 June 11

```
In[1]:= SetDirectory["l:"]; << goedel.11jun10a

        :Package Title: goedel.11jun10a               2011 June 10 at 10:35 a.m.

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

        It is now:  2011 Jun 11 at 9:34

        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 Jun 11 at 9:45
```

summary

If **x** is a group, binary closure under **rotate[x]** is equivalent to binary closure under **x** plus invariance under **inv[x]**.

derivation

The **gp[x]** wrapper helps provide automatic simplifications. What needs to be shown is that **binclosed[rotate[gp[x]]]** is equal to the intersection of **binclosed[gp[x]]** and **invar[gp[x]]**. This equation will be derived by combining inclusions in two directions. The easy direction will be done first.

Lemma

```
In[6]:= Map[implies[subclass[image[inv[gp[x]], y], y], #] &,
          SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → gp[x], u → cart[image[inv[gp[x]], y], y], v → cart[y, y]}]] // Reverse

Out[6]= or[not[subclass[image[inv[gp[x]], y], y]],
          subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], image[gp[x], cart[y, y]]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[8]:= Map[not, SubstTest[and, implies[and[p1, p3], p4],
           implies[p2, p3], not[implies[and[p1, p2], p4]],
           {p1 -> subclass[image[gp[x], cart[y, y]], y], p2 → invariant[inv[gp[x]], y],
            p3 → subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], image[gp[x], cart[y, y]]],
            p4 -> subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]}]] // Reverse

Out[8]= or[not[subclass[image[gp[x], cart[y, y]], y]], not[subclass[image[inv[gp[x]], y], y]],
          subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]] == True

In[10]:= or[not[subclass[image[gp[x_], cart[y_, y_]], y_]],
           not[subclass[image[inv[gp[x_]], y_], y_]],
           subclass[image[gp[x_], cart[image[inv[gp[x_]], y_], y_]], y_]] := True
```

Theorem. (Eliminating the variable **y**.)

```
In[11]:= Map[equal[V, #] &, complement[dif[intersection[binclosed[gp[x]], invar[inv[gp[x]]]],
             binclosed[composite[gp[x], SWAP, cross[Id, inv[gp[x]]]]]]] // Normality]

Out[11]= subclass[intersection[binclosed[gp[x]], invar[inv[gp[x]]]],
           binclosed[composite[gp[x], SWAP, cross[Id, inv[gp[x]]]]]] == True

In[12]:= (% /. x → x_) /. Equal → SetDelayed
```

---

inclusion in the opposite direction

To derive an inclusion in the opposite direction, it is convenient to consider first subsets of **range[gp[x]]**, and only later to remove that limitation. The first step will be to show that if a subset **y ⊂ range[gp[x]]** is binary closed under **rotate[gp[x]]**, then it holds the neutral element **e[gp[x]]**. The strategy will be to consider an element **u ∈ y** and its inverse **APPLY[inv[gp[x]],u]**.

Theorem. If **u ∈ y** and **y ⊂ range[gp[x]]**, then **APPLY[inv[gp[x]], u] ∈ image[inv[gp[x]], y]**.

```
In[13]:= SubstTest[implies, and[member[u, y], subclass[y, domain[funpart[t]]]],
           member[APPLY[funpart[t], u], image[funpart[t], y]], t → inv[gp[x]]] // Reverse

Out[13]= or[member[APPLY[inv[gp[x]], u], image[inv[gp[x]], y]],
           not[member[u, y]], not[subclass[y, range[gp[x]]]]] == True

In[14]:= or[member[APPLY[inv[gp[x_]], u_], image[inv[gp[x_]], y_]],
           not[member[u_, y_]], not[subclass[y_, range[gp[x_]]]]] := True
```

Lemma.

```
In[15]:= Map[or[#, not[subclass[y, range[gp[x]]]]] &, SubstTest[implies, and[member[w, r],
           subclass[r, domain[funpart[t]]], subclass[image[funpart[t], r], y]],
          member[APPLY[funpart[t], w], y], {w → PAIR[APPLY[inv[gp[x]], u], u],
           r → cart[image[inv[gp[x]], y], y], t → gp[x]}]] // Reverse

Out[15]= or[member[e[gp[x]], y], not[member[u, y]], not[member[u, range[gp[x]]]],
          not[member[APPLY[inv[gp[x]], u], image[inv[gp[x]], y]]],
          not[subclass[y, range[gp[x]]]],
          not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]]] == True

In[16]:= (% /. {u → u_, x → x_, y → y_}) /. Equal → SetDelayed
```

The preceding lemma can be improved upon by eliminating two redundant literals.

Lemma. If a subset **y ∈ range[gp[x]]** is binary closed under **rotate[gp[x]]** and **u ∈ y**, then **e[gp[x]] ∈ y**.

```
In[17]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
           implies[and[p1, p2], p5], (*implies[and[p1,p2,p3,p4,p5],p6],*)
           not[implies[and[p1, p2, p3], p6]], {p1 → member[u, y], p2 → subclass[y, range[gp[x]]],
            p3 -> subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y],
            p4 → member[u, range[gp[x]]], p5 -> member[APPLY[inv[gp[x]], u],
              image[inv[gp[x]], y]], p6 → member[e[gp[x]], y]}]] // Reverse

Out[17]= or[member[e[gp[x]], y], not[member[u, y]], not[subclass[y, range[gp[x]]]],
          not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]]] == True

In[18]:= (% /. {u → u_, x → x_, y → y_}) /. Equal → SetDelayed
```

The element **u ∈ y** has served its purpose and can now be eliminated.

Theorem. If a non-empty subset **y ∈ range[gp[x]]** is binary closed under **rotate[gp[x]]**, then **e[gp[x]] ∈ y**.

```
In[19]:= Map[equal[V, #] &, SubstTest[class, u,
           or[member[t, y], not[member[u, y]], not[subclass[y, r]], not[subclass[w, y]]],
           {t → e[gp[x]], r → range[gp[x]], w -> image[gp[x], cart[image[inv[gp[x]], y], y]]}]]

Out[19]= or[equal[0, y], member[e[gp[x]], y], not[subclass[y, range[gp[x]]]],
          not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]]] == True

In[20]:= or[equal[0, y_], member[e[gp[x_]], y_], not[subclass[y_, range[gp[x_]]]],
          not[subclass[image[gp[x_], cart[image[inv[gp[x_]], y_], y_]], y_]]] := True
```

The next step is to show that **y** is invariant under inversion. Again, a particular element **u ∈ y** will be chosen, and later eliminated.

Lemma.

```
In[21]:= Map[or[#, not[subclass[y, range[gp[x]]]]] &, SubstTest[implies, and[member[w, r],
           subclass[r, domain[funpart[t]]], subclass[image[funpart[t], r], y]],
          member[APPLY[funpart[t], w], y], {w → PAIR[APPLY[inv[gp[x]], u], e[gp[x]]],
           r → cart[image[inv[gp[x]], y], y], t → gp[x]}]] // Reverse

Out[21]= or[equal[0, gp[x]], member[APPLY[inv[gp[x]], u], y], not[member[u, range[gp[x]]]],
          not[member[APPLY[inv[gp[x]], u], image[inv[gp[x]], y]]],
          not[member[e[gp[x]], y]], not[subclass[y, range[gp[x]]]],
          not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]]] == True

In[22]:= (% /. {u → u_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If **y ⊂ range[gp[x]]** is binary closed under **rotate[gp[x]]** and **u ∈ y**, then **APPLY[inv[gp[x]], u] ∈ y**.

```
In[23]:= Map[not, SubstTest[and, implies[and[p0, p2, p3], p1],
           implies[p1, p4], implies[and[p0, p2], p5], implies[and[p0, p2], p6],
           (*implies[and[p1,p2,p3,p4,p5,p6],p7]*) not[implies[and[p0, p2, p3], p7]],
           {p0 → member[u, y], p1 -> member[e[gp[x]], y], p2 → subclass[y, range[gp[x]]],
            p3 -> subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y],
            p4 → not[empty[gp[x]]], p5 -> member[APPLY[inv[gp[x]], u], image[inv[gp[x]], y]],
            p6 → member[u, range[gp[x]]], p7 -> member[APPLY[inv[gp[x]], u], y]}]] // Reverse

Out[23]= or[member[APPLY[inv[gp[x]], u], y], not[member[u, y]], not[subclass[y, range[gp[x]]]],
          not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]]] == True

In[25]:= or[member[APPLY[inv[gp[x_]], u_], y_], not[member[u_, y_]],
          not[subclass[image[gp[x_], cart[image[inv[gp[x_]], y_], y_]], y_]],
          not[subclass[y_, range[gp[x_]]]]] := True
```

Eliminating the variable **u** yields a subvariance statement.

Theorem. If **y ⊂ range[gp[x]]** is binary closed under **rotate[gp[x]]**, then **y** is subvariant under **inv[gp[x]]**.

```
In[26]:= Map[equal[V, #] &, SubstTest[class, u, or[member[APPLY[t, u], y],
           not[member[u, y]], not[subclass[y, r]], not[subclass[w, y]]],
          {t → inv[gp[x]], r -> range[gp[x]], w -> image[gp[x], cart[image[inv[gp[x]], y], y]]}]]

Out[26]= or[not[subclass[y, range[gp[x]]]],
          not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]],
          subclass[y, image[inv[gp[x]], y]]] == True

In[27]:= or[not[subclass[y_, range[gp[x_]]]],
          not[subclass[image[gp[x_], cart[image[inv[gp[x_]], y_], y_]], y_]],
          subclass[y_, image[inv[gp[x_]], y_]]] := True
```

Lemma. If a subset **y ⊂ range[gp[x]]** is subvariant under **inv[gp[x]]**, then **y** is also invariant under **inv[gp[x]]**.

```
In[34]:= Map[implies[and[subclass[y, range[gp[x]]], #], invariant[inv[gp[x]], y]] &,
           SubstTest[member, y, subvar[t], t → inv[gp[x]]]]

Out[34]= or[not[subclass[y, image[inv[gp[x]], y]]],
          not[subclass[y, range[gp[x]]]], subclass[image[inv[gp[x]], y], y]] == True
```

```
In[36]:= or[not[subclass[y_, image[inv[gp[x_]], y_]]],
           not[subclass[y_, range[gp[x_]]]], subclass[image[inv[gp[x_]], y_], y_]] := True
```

Corollary. If **y ⊂ range[gp[x]]** is binary closed under **rotate[gp[x]]**, then **y** is invariant under **inv[gp[x]]**.

```
In[37]:= Map[not, SubstTest[and, implies[p1, p2],
            not[implies[p1, p3]], {p1 → and[subclass[y, range[gp[x]]],
               subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]],
             p2 → subvariant[inv[gp[x]], y], p3 → invariant[inv[gp[x]], y]}]] // Reverse

Out[37]= or[not[subclass[y, range[gp[x]]]],
           not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]],
           subclass[image[inv[gp[x]], y], y]] == True
```

```
In[38]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The condition **y ⊂ range[gp[x]]** is not needed.

Theorem. Any class that is binary closed under **rotate[gp[x]]** is invariant under **inv[gp[x]]**.

```
In[40]:= SubstTest[or, not[subclass[t, range[gp[x]]]],
           not[subclass[image[gp[x], cart[image[inv[gp[x]], t], t]], t]],
           subclass[image[inv[gp[x]], t], t], t → intersection[y, range[gp[x]]]] // Reverse

Out[40]= or[not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]],
           subclass[image[inv[gp[x]], y], y]] == True
```

```
In[42]:= or[not[subclass[image[gp[x_], cart[image[inv[gp[x_]], y_], y_]], y_]],
           subclass[image[inv[gp[x_]], y_], y_]] := True
```

Corollary. The class **binclosed[rotate[gp[x]]** is a subclass of **invar[inv[gp[x]]]**.

```
In[43]:= Map[equal[V, #] &,
           complement[dif[binclosed[rotate[gp[x]]], invar[inv[gp[x]]]]] // Normality]

Out[43]= subclass[binclosed[composite[gp[x], SWAP, cross[Id, inv[gp[x]]]]], invar[inv[gp[x]]]] ==
           True
```

```
In[44]:= (% /. x → x_) /. Equal → SetDelayed
```

binary closure

It remains to be shown that a set that is binary closed under **rotate[gp[x]]** is also binary closed under **gp[x]** itself.

Lemma.

```
In[3]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
          {t → gp[x], u → cart[y, y], v → cart[image[inv[gp[x]], y], y]}] // Reverse

Out[3]= or[not[subclass[y, image[inv[gp[x]], y]]],
          subclass[image[gp[x], cart[y, y]], image[gp[x], cart[image[inv[gp[x]], y], y]]]] == True
```

```
In[4]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If a subset **y ⊂ range[gp[x]]** is binary closed under **rotate[gp[x]]**, then **y** is binary closed under **gp[x]**.

```
In[45]:= Map[not,
           SubstTest[and, implies[and[p1, p2], p3], implies[p3, p4], implies[and[p2, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 → subclass[y, range[gp[x]]],
             p2 -> subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y],
             p3 → subclass[y, image[inv[gp[x]], y]], p4 →
              subclass[image[gp[x], cart[y, y]], image[gp[x], cart[image[inv[gp[x]], y], y]]],
             p5 -> subclass[image[gp[x], cart[y, y]], y]}]] // Reverse

Out[45]= or[not[subclass[y, range[gp[x]]]],
           not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]],
           subclass[image[gp[x], cart[y, y]], y]] == True

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

Again, the condition **y ⊂ range[gp[x]]** is not needed.

Corollary. Any class that is binary closed under **rotate[gp[x]]** is binary closed under **gp[x]**.

```
In[47]:= SubstTest[or, not[subclass[t, range[gp[x]]]],
           not[subclass[image[gp[x], cart[image[inv[gp[x]], t], t]], t]],
           subclass[image[gp[x], cart[t, t]], t], t → intersection[y, range[gp[x]]]] // Reverse

Out[47]= or[not[subclass[image[gp[x], cart[image[inv[gp[x]], y], y]], y]],
           subclass[image[gp[x], cart[y, y]], y]] == True

In[48]:= or[not[subclass[image[gp[x_], cart[image[inv[gp[x_]], y_], y_]], y_]],
           subclass[image[gp[x_], cart[y_, y_]], y_]] := True
```

The variable **y** can be eliminated.

Theorem.

```
In[49]:= Map[equal[V, #] &,
          complement[dif[binclosed[rotate[gp[x]]], binclosed[gp[x]]]] // Normality]

Out[49]= subclass[binclosed[composite[gp[x], SWAP, cross[Id, inv[gp[x]]]]], binclosed[gp[x]]] ==
          True

In[50]:= (% /. x → x_) /. Equal → SetDelayed
```

Main Theorem. An equation for **binclosed[rotate[gp[x]]]**.

```
In[51]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → binclosed[rotate[gp[x]]], v → intersection[binclosed[gp[x]], invar[inv[gp[x]]]]}]

Out[51]= equal[binclosed[composite[gp[x], SWAP, cross[Id, inv[gp[x]]]]],
           intersection[binclosed[gp[x]], invar[inv[gp[x]]]]] == True
```

```
In[53]:= binclosed[composite[gp[x_], SWAP, cross[Id, inv[gp[x_]]]]] :=
          intersection[binclosed[gp[x]], invar[inv[gp[x]]]]
```

Corollary. (Remove the **gp** wrapper.)

```
In[55]:= SubstTest[implies, equal[x, gp[t]], equal[binclosed[rotate[x]],
             intersection[binclosed[x], invar[inv[x]]]], t → x] // Reverse // MapNotNot

Out[55]= or[equal[binclosed[rotate[x]], intersection[binclosed[x], invar[inv[x]]]],
           not[member[x, GROUPS]]] == True

In[56]:= or[equal[binclosed[rotate[x_]], intersection[binclosed[x_], invar[inv[x_]]]],
           not[member[x_, GROUPS]]] := True
```