ranges of subgroups of a group

Johan G. F. Belinfante
2011 June 5

```
In[1]:= SetDirectory["l:"]; << goedel.11jun02a

        :Package Title: goedel.11jun02a                2011 June 2 at 10:40 a.m.

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

        It is now:  2011 Jun 5 at 0:59

        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 5 at 1:10
```

summary

In this notebook it is shown that the class of all subgroups of a group is determined by the class of binary closed sets that hold the neutral element and are both invariant and subvariant under inversion.

introduction

The terminology and notations used in the **GOEDEL** program differ only slightly from standard usage in group theory. The term **group** here refers to what is usually called a group composition law, that is, an associative binary operation **x: range[x] × range[x] → range [x]** on the set **range[x]**, satisfying the usual group axioms. The class of all groups is denoted by **GROUPS**. If **u** and **v** are any elements of **range[x]**, their **product** often denoted **u · v = w** is the unique element **w ∈ range[x]** satisfying **pair[pair[u, v], w] ∈ x**. For each group **x ∈ GROUPS**, there is a unique **neutral element** denoted **e[x] ∈ range[x]** satisfying **x ∘ LEFT[e[x]] = id[range[x]] = x ∘ RIGHT[e[x]]**, and there is an involution **inv[x]** on **range[x]** called **inversion**. The latter is a function **inv[x]: range[x] → range[x]** sending each element of **u ∈ range[x]** to the unique inverse element **v ∈ range[x]** satisfying **u · v = e[x] = v · u**

Lemma.

```
In[18]:= SubstTest[member, inv[setpart[t]], V, t → gp[x]] // Reverse

Out[18]= member[inv[gp[x]], V] == True

In[19]:= member[inv[gp[x_]], V] := True
```

Theorem.

```
In[20]:= SubstTest[and, member[t, FUNS], equal[domain[t], range[gp[x]]],
          subclass[range[t], range[gp[x]]], t → inv[gp[x]]]

Out[20]= member[inv[gp[x]], map[range[gp[x]], range[gp[x]]]] == True

In[21]:= member[inv[gp[x_]], map[range[gp[x_]], range[gp[x_]]]] := True
```

Corollary.

```
In[23]:= SubstTest[implies, equal[x, gp[t]],
            member[inv[x], map[range[x], range[x]]], t → x] // Reverse // MapNotNot

Out[23]= or[member[inv[x], map[range[x], range[x]]], not[member[x, GROUPS]]] == True

In[24]:= or[member[inv[x_], map[range[x_], range[x_]]], not[member[x_, GROUPS]]] := True
```

It will be said that **s** is a **subgroup** of a group **x** if **s** is also a group, and **s ⊂ x**. If **s** is a subgroup of **x**, then both **s** and **x** are functions, and since **s ⊂ x**, it follows that the function **s** is the restriction of **x** to **domain[s] = range[s] × range[s]**. Consequently, any subgroup of a given group is completely determined once its range is known. The class of all subgroups of a group **x** is **GROUPS ⋂ P[x]**. Of immediate interest therefore is the question, given a group, which sets can be the range of a subgroup? In this notebook certain well-known necessary and sufficient conditions on the range are identified: if **s** is a subgroup of **x**, then **y = range[s]** must be a subset of **range[x]** that is binary closed under **x**, the neutral element **e[x]** must belong to **y**, and **y** must be invariant under **inv[x]**.

The two conditions **y ⊂ range[x]** and **image[inv[x], y] ⊂ y** can be replaced with the single condition **image[inv[x], y] = y**. The main result derived in this notebook is a formula that expresses the class **image[IMAGE[SECOND], GROUPS ⋂ P[x]]** of all ranges of subgroups of a group **x** as the intersection of three classes, namely, the class **binclosed[x]** of all sets **y** satisfying **image[x, y × y] ⊂ y**, the class **P[{x}']'** of all sets that hold the neutral element **e[x]** of the group **x**, and the class **fix[IMAGE[inv[x]]]** of all sets **y** satisfying **image[inv[x], y] = y**.

necessary conditions

It has already been shown earlier that a subgroup of a group is a unital submonoid, that is, their neutral elements are the same. If **x** is a subgroup of a group **y**, the neutral element **e[x]** for the subgroup **x** is the same as the neutral element **e[y]** for the group **y**. From this fact it follows immediately that **e[y] ∈ range[x]**.

Theorem. If **x** is a subgroup of a group **y**, then **e[y] ∈ range[x]**.

```
In[28]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
            implies[and[p1, p4], p5], not[implies[and[p1, p2, p3], p5]],
            {p1 → member[x, GROUPS], p2 → subclass[x, y], p3 → member[y, GROUPS],
             p4 → equal[e[x], e[y]], p5 → member[e[y], range[x]]}]] // Reverse

Out[28]= or[member[e[y], range[x]], not[member[x, GROUPS]],
           not[member[y, GROUPS]], not[subclass[x, y]]] == True
```

```
In[29]:= or[member[e[y_], range[x_]], not[member[x_, GROUPS]],
           not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

If **x** is a subgroup of a group **y**, the inversion function **inv[x]** for the subgroup **x** is the restriction of the inversion function **inv[y]** for the group **y** to the set **range[x]**. From this it follows that the range of **x** is both invariant and subvariant under **inv[y]**.

Lemma.

```
In[30]:= SubstTest[implies, equal[u, v], equal[range[u], range[v]],
           {u → inv[x], v → composite[inv[y], id[z]]}] // Reverse

Out[30]= or[equal[domain[inv[x]], image[inv[y], z]],
           not[equal[composite[inv[y], id[z]], inv[x]]]] == True

In[31]:= or[equal[domain[inv[x_]], image[inv[y_], z_]],
           not[equal[composite[inv[y_], id[z_]], inv[x_]]]] := True
```

Theorem. If **x** is a subgroup of a group **y**, then **image[inv[y], range[x]] = range[x]**.

```
In[32]:= Map[not, SubstTest[and, (* implies[and[p1,p2,p3],p4],implies[and[p1,p2,p3],p5],
             implies[p1,p6], *) implies[and[p4, p5, p6], p7], implies[and[p6, p7], p8],
            not[implies[and[p1, p2, p3], p8]], {p1 → member[x, GROUPS],
             p2 → subclass[x, y], p3 → member[y, GROUPS], p4 → subclass[inv[x], inv[y]],
             p5 → FUNCTION[inv[y]], p6 → equal[domain[inv[x]], range[x]],
             p7 → equal[inv[x], composite[inv[y], id[range[x]]]],
             p8 → equal[image[inv[y], range[x]], range[x]]}]] // Reverse

Out[32]= or[equal[image[inv[y], range[x]], range[x]],
           not[member[x, GROUPS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[33]:= or[equal[image[inv[y_], range[x_]], range[x_]], not[member[x_, GROUPS]],
           not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Theorem. Necessary conditions for a set to be the range of a subgroup.

```
In[34]:= implies[and[member[t, intersection[GROUPS, P[x]]], member[x, GROUPS]], member[range[t],
            intersection[binclosed[x], image[E, set[e[x]]], fix[IMAGE[inv[x]]]]]] // NotNotTest

Out[34]= or[and[equal[image[inv[x], range[t]], range[t]], member[e[x], range[t]],
            subclass[image[x, cart[range[t], range[t]]], range[t]]],
           not[member[t, GROUPS]], not[member[x, GROUPS]], not[subclass[t, x]]] == True

In[35]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

In order to eliminate the variable **t** the following lemma is convenient.

Temporary Lemma.

```
In[36]:= SubstTest[subclass, u, image[inverse[IMAGE[SECOND]], v],
           {u -> intersection[GROUPS, P[x]], v -> image[E, set[e[x]]]}] // Reverse

Out[36]= equal[0, intersection[GROUPS, P[intersection[x, complement[cart[V, set[e[x]]]]]]]] ==
          or[and[not[member[e[x], V]], subclass[intersection[GROUPS, P[x]], 0]],
           member[e[x], A[image[IMAGE[SECOND], intersection[GROUPS, P[x]]]]]]

In[37]:= equal[0, intersection[GROUPS, P[intersection[x_, complement[cart[V, set[e[x_]]]]]]]] :=
          or[and[not[member[e[x], V]], subclass[intersection[GROUPS, P[x]], 0]],
           member[e[x], A[image[IMAGE[SECOND], intersection[GROUPS, P[x]]]]]]
```

Lemma. A simplification rule.

```
In[38]:= equiv[and[member[x, GROUPS], member[e[x], V]], member[x, GROUPS]]

Out[38]= True

In[39]:= and[member[x_, GROUPS], member[e[x_], V]] := member[x, GROUPS]
```

Theorem. The neutral element **e[x]** is common to all ranges of subgroups of **x**.

```
In[40]:= Map[equal[V, #] &,
           SubstTest[class, t, implies[and[member[t, u], member[x, w]], member[t, v]],
            {u -> intersection[GROUPS, P[x]],
             v → image[inverse[IMAGE[SECOND]], image[E, set[e[x]]]], w → GROUPS}]] // MapNotNot

Out[40]= or[member[e[x], A[image[IMAGE[SECOND], intersection[GROUPS, P[x]]]],
           not[member[x, GROUPS]]] == True

In[41]:= or[member[e[x_], A[image[IMAGE[SECOND], intersection[GROUPS, P[x_]]]]],
           not[member[x_, GROUPS]]] := True
```

Theorem. The range of any subgroup of a group **x** is fixed by the function **IMAGE[inv[x]]**.

```
In[42]:= Map[equal[V, #] &,
          SubstTest[class, t, implies[and[member[t, u], member[x, w]], member[t, v]],
           {u -> intersection[GROUPS, P[x]],
            v → image[inverse[IMAGE[SECOND]], fix[IMAGE[inv[x]]]], w → GROUPS}]]

Out[42]= or[not[member[x, GROUPS]], subclass[
            image[IMAGE[SECOND], intersection[GROUPS, P[x]]], fix[IMAGE[inv[x]]]] == True

In[43]:= or[not[member[x_, GROUPS]], subclass[
            image[IMAGE[SECOND], intersection[GROUPS, P[x_]]], fix[IMAGE[inv[x_]]]] := True
```

Comment. It has already been shown earlier that the range of any subgroup of a group **x** is binary closed under **x**.

```
In[44]:= or[not[member[x, GROUPS]],
          subclass[image[IMAGE[SECOND], intersection[GROUPS, P[x]]], binclosed[x]]]

Out[44]= True
```

sufficient conditions

In the recently posted notebook **SBGPS-RS.NB** it was shown that if **x** is a group, and if a class **y** is binary closed under **x**, holds **e[x]**, and is invariant under **inv[x]**, then **x ∘ id[y × y]** is a subgroup of **x**.

```
In[45]:= implies[and[member[x, GROUPS], subclass[image[x, cart[y, y]], y], member[e[x], y],
           invariant[inv[x], y]], member[composite[x, id[cart[y, y]]], GROUPS]]
Out[45]= True
```

The following corollary just replaces the hypothesis that **y** be invariant under **inv[x]** with a stronger equational hypothesis.

Corollary.

```
In[46]:= Map[not, SubstTest[and, implies[p2, p3], not[implies[p1, p3]],
           {p1 -> equal[image[inv[x], y], y], p2 → subclass[image[inv[x], y], y],
            p3 -> or[member[composite[x, id[cart[y, y]]], GROUPS], not[member[x, GROUPS]],
              not[member[e[x], y]], not[subclass[image[x, cart[y, y]], y]]]}]] // Reverse
Out[46]= or[member[composite[x, id[cart[y, y]]], GROUPS],
           not[equal[y, image[inv[x], y]]], not[member[x, GROUPS]],
           not[member[e[x], y]], not[subclass[image[x, cart[y, y]], y]]] == True

In[47]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem. (Introducing a **gp** wrapper.)

```
In[48]:= SubstTest[or, member[composite[t, id[cart[y, y]]], GROUPS],
           not[equal[y, image[inv[t], y]]], not[member[t, GROUPS]], not[member[e[t], y]],
           not[subclass[image[t, cart[y, y]], y]], t → gp[x]] // Reverse
Out[48]= or[equal[0, gp[x]], member[composite[gp[x], id[cart[y, y]]], GROUPS],
           not[equal[y, image[inv[gp[x]], y]]], not[member[e[gp[x]], y]],
           not[subclass[image[gp[x], cart[y, y]], y]]] == True

In[49]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

A redundant literal can be eliminated.

Theorem. If a class **y** is binary closed under **gp[x]**, holds **e[gp[x]]**, and is subvariant and invariant under **inv[gp[x]]**, then the restriction of **gp[x]** to **y × y** is a group.

```
In[50]:= SubstTest[and, implies[p, q], or[p, q],
           {p → empty[gp[x]], q -> or[member[composite[gp[x], id[cart[y, y]]], GROUPS],
              not[equal[y, image[inv[gp[x]], y]]], not[member[e[gp[x]], y]],
              not[subclass[image[gp[x], cart[y, y]], y]]]}]
Out[50]= or[member[composite[gp[x], id[cart[y, y]]], GROUPS],
           not[equal[y, image[inv[gp[x]], y]]], not[member[e[gp[x]], y]],
           not[subclass[image[gp[x], cart[y, y]], y]]] == True
```

```
In[51]:= or[member[composite[gp[x_], id[cart[y_, y_]]], GROUPS],
           not[equal[image[inv[gp[x_]], y_], y_]], not[member[e[gp[x_]], y_]],
           not[subclass[image[gp[x_], cart[y_, y_]], y_]]] := True
```

Lemma.

```
In[52]:= SubstTest[implies, member[t, GROUPS], equal[range[t], fix[domain[t]]],
           t -> composite[gp[x], id[cart[y, y]]]] // Reverse

Out[52]= or[equal[image[gp[x], cart[y, y]], intersection[y, range[gp[x]]]],
           not[member[composite[gp[x], id[cart[y, y]]], GROUPS]]] == True
```

```
(% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma.

```
In[54]:= SubstTest[implies, member[u, v],
           member[range[u], image[IMAGE[SECOND], v]], {u -> composite[gp[x], id[cart[y, y]]],
            v -> intersection[GROUPS, P[gp[x]]]}] // Reverse

Out[54]= or[member[image[gp[x], cart[y, y]],
             image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[member[composite[gp[x], id[cart[y, y]]], GROUPS]]] == True
```

```
In[55]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem.

```
In[56]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[p1, p4]], {p1 -> member[composite[gp[x], id[cart[y, y]]], GROUPS],
             p2 → equal[image[gp[x], cart[y, y]], intersection[y, range[gp[x]]]],
             p3 → member[image[gp[x], cart[y, y]], image[IMAGE[SECOND],
                intersection[GROUPS, P[gp[x]]]]], p4 → member[intersection[y, range[gp[x]]],
               image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]]}]] // Reverse

Out[56]= or[member[intersection[y, range[gp[x]]],
             image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[member[composite[gp[x], id[cart[y, y]]], GROUPS]]] == True
```

```
In[57]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem.

```
In[58]:= SubstTest[implies, equal[z, intersection[y, range[gp[x]]]],
           or[member[z, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
            not[member[composite[gp[x], id[cart[y, y]]], GROUPS]]], z → y] // Reverse

Out[58]= or[member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[member[composite[gp[x], id[cart[y, y]]], GROUPS]],
           not[subclass[y, range[gp[x]]]]] == True
```

```
In[59]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem. A sufficient condition for **y** to be the range of a subgroup of **gp[x]**.

```
In[60]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 -> and[equal[y, image[inv[gp[x]], y]],
              member[e[gp[x]], y], subclass[image[gp[x], cart[y, y]], y]], p2 ->
             member[composite[gp[x], id[cart[y, y]]], GROUPS], p3 → subclass[y, range[gp[x]]],
            p4 -> member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]]}]] // Reverse
```

```
Out[60]= or[member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[equal[y, image[inv[gp[x]], y]]], not[member[e[gp[x]], y]],
           not[subclass[image[gp[x], cart[y, y]], y]]] == True
```

```
In[61]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

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

```
In[62]:= Map[equal[V, #] &, SubstTest[class, y, implies[member[y, u], member[y, v]],
           {u → intersection[binclosed[gp[x]], image[E, set[e[gp[x]]]], fix[IMAGE[inv[gp[x]]]]],
            v -> image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]}]]
```

```
Out[62]= subclass[intersection[binclosed[gp[x]], fix[IMAGE[inv[gp[x]]]]],
           union[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]],
            P[complement[set[e[gp[x]]]]]]] == True
```

```
In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (Eliminating the **gp** wrapper.)

```
In[64]:= SubstTest[implies, equal[x, gp[t]],
           subclass[intersection[binclosed[x], fix[IMAGE[inv[x]]]],
            union[image[IMAGE[SECOND], intersection[GROUPS, P[x]]],
             P[complement[set[e[x]]]]]], t → x] // Reverse // MapNotNot
```

```
Out[64]= or[not[member[x, GROUPS]],
           subclass[intersection[binclosed[x], fix[IMAGE[inv[x]]]], union[image[
              IMAGE[SECOND], intersection[GROUPS, P[x]]], P[complement[set[e[x]]]]]] == True
```

```
In[65]:= (% /. x → x_) /. Equal → SetDelayed
```

a formula for the class of ranges of subgroups of a group

The main theorem of this notebook is obtained by combining the necessary and the sufficient conditions for a class to be the range of a subgroup of a given group.

Main Theorem. An explicit formula for the class of ranges of subgroups of a group **x**.

```
In[66]:= Map[implies[member[x, GROUPS], #] &, SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[IMAGE[SECOND], intersection[GROUPS, P[x]]], v -> intersection[
              binclosed[x], image[E, set[e[x]]], fix[IMAGE[inv[x]]]]}]] // MapNotNot

Out[66]= or[equal[image[IMAGE[SECOND], intersection[GROUPS, P[x]]],
           intersection[binclosed[x], complement[P[complement[set[e[x]]]]],
            fix[IMAGE[inv[x]]]], not[member[x, GROUPS]]] == True

In[67]:= or[equal[image[IMAGE[SECOND], intersection[GROUPS, P[x_]]],
           intersection[binclosed[x_], complement[P[complement[set[e[x_]]]]],
            fix[IMAGE[inv[x_]]]], not[member[x_, GROUPS]]] := True
```

A better rewrite rule can be derived by reintroducing the **gp** wrapper.

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

```
In[68]:= SubstTest[or, equal[image[IMAGE[SECOND], intersection[GROUPS, P[t]]],
           intersection[binclosed[t], complement[P[complement[set[e[t]]]]],
            fix[IMAGE[inv[t]]]], not[member[t, GROUPS]], t → gp[x]] // Reverse

Out[68]= or[equal[0, gp[x]], equal[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]],
           intersection[binclosed[gp[x]],
            complement[P[complement[set[e[gp[x]]]]]], fix[IMAGE[inv[gp[x]]]]]] == True

In[69]:= (% /. x → x_) /. Equal → SetDelayed
```

A redundant literal can be eliminated.

Theorem. A rewrite rule for the class of ranges of a group **gp[x]**.

```
In[70]:= SubstTest[and, implies[p, q], or[p, q],
          {p -> equal[0, gp[x]], q -> equal[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]],
             intersection[binclosed[gp[x]],
              complement[P[complement[set[e[gp[x]]]]]], fix[IMAGE[inv[gp[x]]]]]]}]

Out[70]= equal[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]],
           intersection[binclosed[gp[x]],
            complement[P[complement[set[e[gp[x]]]]]], fix[IMAGE[inv[gp[x]]]]] == True

In[71]:= intersection[binclosed[gp[x_]], complement[P[complement[set[e[gp[x_]]]]]],
           fix[IMAGE[inv[gp[x_]]]]] := image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]
```