subgroups of a group

Johan G. F. Belinfante
2011 June 7

```
In[1]:= SetDirectory["l:"]; << goedel.11jun06a

        :Package Title: goedel.11jun06a                 2011 June 6 at 3:40 p.m.

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

        It is now:  2011 Jun 7 at 11:29

        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 7 at 11:40
```

summary

A necessary and sufficient condition for a subset **y** of the range of a group **x** to be the range of a subgroup is that it be non-empty, binary closed under **x**, and invariant under **inv[x]**.

derivation

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

```
In[5]:= Map[implies[#, member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]]] &,
         SubstTest[member, y, intersection[t, u, v, w], {t -> P[range[gp[x]]],
           u -> binclosed[gp[x]], v -> image[E, set[e[gp[x]]]], w -> invar[inv[gp[x]]]}]]

Out[5]= or[member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
          not[member[y, V]], not[member[e[gp[x]], y]],
          not[subclass[y, range[gp[x]]]], not[subclass[image[gp[x], cart[y, y]], y]],
          not[subclass[image[inv[gp[x]], y], y]] == True

In[6]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The sethood requirement can be waived since it follows from the condition **y ⊂ range[gp[x]]**.

Lemma. Any subclass of **range[gp[x]]** is a set.

```
In[12]:= SubstTest[implies, and[subclass[y, z], member[z, V]],
           member[y, V], z → range[gp[x]]] // Reverse

Out[12]= or[member[y, V], not[subclass[y, range[gp[x]]]]] == True

In[14]:= or[member[y_, V], not[subclass[y_, range[gp[x_]]]]] := True
```

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

```
In[17]:= Map[not, SubstTest[and, implies[and[p0, p1, p2, p3, p4], p5],
           implies[p2, p0], not[implies[and[p1, p2, p3, p4], p5]],
           {p0 → member[y, V], p1 -> member[e[gp[x]], y], p2 -> subclass[y, range[gp[x]]],
            p3 -> subclass[image[gp[x], cart[y, y]], y], p4 -> subclass[image[inv[gp[x]], y], y],
            p5 -> member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]]}]] // Reverse

Out[17]= or[member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[member[e[gp[x]], y]], not[subclass[y, range[gp[x]]]],
           not[subclass[image[gp[x], cart[y, y]], y]],
           not[subclass[image[inv[gp[x]], y], y]]] == True

In[19]:= or[member[y_, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x_]]]]],
           not[member[e[gp[x_]], y_]], not[subclass[image[gp[x_], cart[y_, y_]], y_]],
           not[subclass[image[inv[gp[x_]], y_], y_]], not[subclass[y_, range[gp[x_]]]]] := True
```

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

```
In[21]:= SubstTest[implies, equal[x, gp[t]], or[
            member[y, image[IMAGE[SECOND], intersection[GROUPS, P[x]]]], not[member[e[x], y]],
            not[subclass[y, range[x]]], not[subclass[image[x, cart[y, y]], y]],
            not[subclass[image[inv[x], y], y]]], t → x] // Reverse // MapNotNot

Out[21]= or[member[y, image[IMAGE[SECOND], intersection[GROUPS, P[x]]]],
           not[member[x, GROUPS]], not[member[e[x], y]], not[subclass[y, range[x]]],
           not[subclass[image[x, cart[y, y]], y]], not[subclass[image[inv[x], y], y]]] == True

In[22]:= or[member[y_, image[IMAGE[SECOND], intersection[GROUPS, P[x_]]]],
           not[member[x_, GROUPS]], not[member[e[x_], y_]],
           not[subclass[y_, range[x_]]], not[subclass[image[x_, cart[y_, y_]], y_]],
           not[subclass[image[inv[x_], y_], y_]]] := True
```

Comment. The condition **y ⊂ range[x]** can also be waived if all one wants to know is that the restriction of **x** to **y × y** is a group, and not the additional information that **y** is the range of that restriction. This fact is already available:

```
In[24]:= 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]], not[subclass[image[inv[x], y], y]]]

Out[24]= True
```

a lemma about binary closed classes

Lemma.

```
In[61]:= SubstTest[subclass, cart[x, x], cart[t, t], t → fix[domain[binop[y]]]] // Reverse

Out[61]= subclass[cart[x, x], domain[binop[y]]] == subclass[x, fix[domain[binop[y]]]]

In[62]:= subclass[cart[x_, x_], domain[binop[y_]]] := subclass[x, fix[domain[binop[y]]]]
```

Theorem.

```
In[65]:= SubstTest[implies,
           and[FUNCTION[t], subclass[z, domain[t]], member[w, z], subclass[image[t, z], y]],
           member[APPLY[t, w], y], {t → binop[x], w → PAIR[u, v], z → cart[y, y]}] // Reverse

Out[65]= or[member[APPLY[binop[x], PAIR[u, v]], y], not[member[u, y]],
           not[member[v, y]], not[subclass[y, fix[domain[binop[x]]]]],
           not[subclass[image[binop[x], cart[y, y]], y]]] == True

In[66]:= or[member[APPLY[binop[x_], PAIR[u_, v_]], y_], not[member[u_, y_]],
           not[member[v_, y_]], not[subclass[y_, fix[domain[binop[x_]]]]],
           not[subclass[image[binop[x_], cart[y_, y_]], y_]]] := True
```

Corollary. Analogous result for groups, using **gp** wrapper.

```
In[68]:= SubstTest[or, member[APPLY[binop[t], PAIR[u, v]], y], not[member[u, y]],
           not[member[v, y]], not[subclass[y, fix[domain[binop[t]]]]],
           not[subclass[image[binop[t], cart[y, y]], y]], t → gp[x]] // Reverse

Out[68]= or[member[APPLY[gp[x], PAIR[u, v]], y], not[member[u, y]], not[member[v, y]],
           not[subclass[y, range[gp[x]]]], not[subclass[image[gp[x], cart[y, y]], y]]] == True

In[70]:= or[member[APPLY[gp[x_], PAIR[u_, v_]], y_], not[member[u_, y_]],
           not[member[v_, y_]], not[subclass[image[gp[x_], cart[y_, y_]], y_]],
           not[subclass[y_, range[gp[x_]]]]] := True
```

Corollary. Removing the **gp** wrapper.

```
In[73]:= Map[or[not[member[x, GROUPS]], #] &,
           SubstTest[implies, equal[x, gp[t]], or[member[APPLY[x, PAIR[u, v]], y],
             not[member[u, y]], not[member[v, y]], not[subclass[y, range[x]]],
             not[subclass[image[x, cart[y, y]], y]]], t → x]] // Reverse

Out[73]= or[member[APPLY[x, PAIR[u, v]], y],
           not[member[u, y]], not[member[v, y]], not[member[x, GROUPS]],
           not[subclass[y, range[x]]], not[subclass[image[x, cart[y, y]], y]]] == True

In[74]:= or[member[APPLY[x_, PAIR[u_, v_]], y_], not[member[u_, y_]],
           not[member[v_, y_]], not[member[x_, GROUPS]], not[subclass[y_, range[x_]]],
           not[subclass[image[x_, cart[y_, y_]], y_]]] := True
```

omitting the condition e[x] ∈ y

In this section a slightly simpler sufficient conditions are derived for a class **y** to be the range of a subgroup of a group **x**. The condition **e[x] ∈ y** considered in the preceding section implies that **y** is not empty. Although the latter condition is weaker, itsufficient in combination with the other conditions placed on **y**. Indeed, if **u ∈ y** is any element, then the condition that **y** be invariant under **inv[x]** implies that the inverse element **v = APPLY[inv[x], u]** also belongs to **y**, and the condition that **y** be binary closed under **x** then implies that **u · v = e[x]** belongs to **y**.

Lemma. If **x** is a group, if **y ⊂ range[x]** is invariant under **inv[x]**, and if **u ∈ y**, then **APPLY[inv[x], u] ∈ y**.

```
In[82]:= Map[not, SubstTest[and, implies[and[p0, p1, p2], p6],
            implies[and[p1, p6], p7], implies[p1, p8], (* implies[and[p0,p5,p7,p8],p9], *)
            not[implies[and[p0, p1, p2, p5], p9]], {p0 → member[u, y], p1 → member[x, GROUPS],
             p2 → subclass[y, range[x]], p5 → subclass[image[inv[x], y], y],
             p6 → member[u, range[x]], p7 → member[u, domain[inv[x]]],
             p8 → FUNCTION[inv[x]], p9 → member[APPLY[inv[x], u], y]}]] // Reverse

Out[82]= or[member[APPLY[inv[x], u], y], not[member[u, y]], not[member[x, GROUPS]],
           not[subclass[y, range[x]]], not[subclass[image[inv[x], y], y]]] == True

In[84]:= or[member[APPLY[inv[x_], u_], y_], not[member[u_, y_]], not[member[x_, GROUPS]],
           not[subclass[image[inv[x_], y_], y_]], not[subclass[y_, range[x_]]]] := True
```

Lemma.

```
In[95]:= Map[not,
           SubstTest[and, implies[and[p1, p2], p6], implies[p1, p4], implies[and[p1, p3], p5],
            implies[p1, p3], implies[and[p1, p4, p5], p2], not[implies[p1, p6]],
            {p1 → and[member[u, y], member[x, GROUPS], subclass[y, range[x]],
               subclass[image[x, cart[y, y]], y], subclass[image[inv[x], y], y]],
             p2 → member[e[x], y], p3 -> member[u, range[x]], p4 → member[APPLY[inv[x], u], y],
             p5 → equal[APPLY[x, PAIR[u, APPLY[inv[x], u]]], e[x]],
             p6 → member[y, image[IMAGE[SECOND], intersection[GROUPS, P[x]]]]}]] // Reverse

Out[95]= or[member[y, image[IMAGE[SECOND], intersection[GROUPS, P[x]]]],
           not[member[u, y]], not[member[x, GROUPS]], not[subclass[y, range[x]]],
           not[subclass[image[x, cart[y, y]], y]], not[subclass[image[inv[x], y], y]]] == True

In[96]:= (% /. {u → u_, x → x_, y → y_}) /. Equal → SetDelayed
```

The final step is to eliminate the variable **u**.

Theorem. If **x** is a group and if **y ⊂ range[x]** is non-empty, binary closed under **x**, and invariant under **inv[x]**, then **y** is the range of a subgroup of **x**.

```
In[99]:= Map[equal[V, #] &, SubstTest[class, u,
           or[member[y, z], not[member[u, y]], not[member[x, v]], not[subclass[y, range[x]]],
            not[subclass[image[x, cart[y, y]], y]], not[subclass[w, y]]], {v → GROUPS,
            w -> image[inv[x], y], z -> image[IMAGE[SECOND], intersection[GROUPS, P[x]]]}]]

Out[99]= or[equal[0, y], member[y, image[IMAGE[SECOND], intersection[GROUPS, P[x]]]],
           not[member[x, GROUPS]], not[subclass[y, range[x]]],
           not[subclass[image[x, cart[y, y]], y]], not[subclass[image[inv[x], y], y]] == True

In[101]:=
         or[equal[0, y_], member[y_, image[IMAGE[SECOND], intersection[GROUPS, P[x_]]]],
           not[member[x_, GROUPS]], not[subclass[image[inv[x_], y_], y_]],
           not[subclass[image[x_, cart[y_, y_]], y_]], not[subclass[y_, range[x_]]]] := True
```

Corollary. Restatement using a **gp[x]** wrapper.

```
In[105]:=
         SubstTest[or, equal[0, y],
           member[y, image[IMAGE[SECOND], intersection[GROUPS, P[t]]]], not[member[t, GROUPS]],
           not[subclass[y, range[t]]], not[subclass[image[t, cart[y, y]], y]],
           not[subclass[image[inv[t], y], y]], t → gp[x]] // Reverse

Out[105]=
         or[equal[0, y], equal[0, gp[x]],
           member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[subclass[y, range[gp[x]]]], not[subclass[image[gp[x], cart[y, y]], y]],
           not[subclass[image[inv[gp[x]], y], y]] == True

In[106]:=
         (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

A redundant literal can be removed.

Theorem.

```
In[108]:=
         SubstTest[and, implies[p, q], or[p, q], {p → equal[0, gp[x]],
           q -> or[equal[0, y], member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
            not[subclass[y, range[gp[x]]]], not[subclass[image[gp[x], cart[y, y]], y]],
            not[subclass[image[inv[gp[x]], y], y]]]}]

Out[108]=
         or[equal[0, y], member[y, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]],
           not[subclass[y, range[gp[x]]]], not[subclass[image[gp[x], cart[y, y]], y]],
           not[subclass[image[inv[gp[x]], y], y]] == True

In[110]:=
         or[equal[0, y_], member[y_, image[IMAGE[SECOND], intersection[GROUPS, P[gp[x_]]]]],
           not[subclass[image[gp[x_], cart[y_, y_]], y_]],
           not[subclass[image[inv[gp[x_]], y_], y_]], not[subclass[y_, range[gp[x_]]]]] := True
```

The variable **y** can also be eliminated.

Theorem.

In[113]:=
```
Map[equal[V, #] &,
 SubstTest[class, y, or[empty[y], member[y, z], not[subclass[y, range[t]]],
   not[subclass[image[t, cart[y, y]], y]], not[subclass[image[w, y], y]]],
  {t → gp[x], w → inv[gp[x]], z -> image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]}]]
```

Out[113]=
```
subclass[intersection[binclosed[gp[x]], fix[IMAGE[inv[gp[x]]]]],
  union[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]], set[0]]] == True
```

In[114]:=
```
subclass[intersection[binclosed[gp[x_]], fix[IMAGE[inv[gp[x_]]]]],
  union[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x_]]]], set[0]]] := True
```

Lemma.

In[116]:=
```
SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
  {t → id[intersection[binclosed[gp[x]], fix[IMAGE[inv[gp[x]]]]]],
   u → image[E, set[e[gp[x]]]], v → complement[set[0]]}] // Reverse
```

Out[116]=
```
subclass[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]],
  fix[IMAGE[inv[gp[x]]]]] == True
```

In[118]:=
```
subclass[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x_]]],
  fix[IMAGE[inv[gp[x_]]]]] := True
```

Theorem. A better rewrite rule.

In[119]:=
```
SubstTest[and, subclass[u, v], subclass[v, u],
 {u -> image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]],
  v -> intersection[binclosed[gp[x]], fix[IMAGE[inv[gp[x]]]], complement[set[0]]]}]
```

Out[119]=
```
equal[image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]],
  intersection[binclosed[gp[x]], complement[set[0]], fix[IMAGE[inv[gp[x]]]]]] == True
```

In[121]:=
```
intersection[binclosed[gp[x_]], complement[set[0]], fix[IMAGE[inv[gp[x_]]]]] :=
 image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]
```