finite subsemigroups of a group

Johan G. F. Belinfante
2012 March 5

```
In[1]:= SetDirectory["l:"]; << goedel.12mar03a
```

```
:Package Title: goedel.12mar03a             2012 March 4 at 12:55 midnite

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

It is now:  2012 Mar 5 at 8:5

Loading Simplification Rules

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

weightlimit = 40

Loading completed.

It is now:  2012 Mar 5 at 8:22
```

summary

A nonempty finite subsemigroup of a group is a subgroup. This fact is posed as exercise 3 on page 25 in the following reference. The derivation presented in this notebook uses the hint given there.

```
In[2]:= "Nathan Jacobson, Lectures in Abstract Algebra. Volume 1- Basic
          Concepts, D. Van Nostrand Company, Inc., Princeton, New Jersey, 1951";
```

Several versions of this result are obtained, inclusing a variable-free statement.

preliminaries

Two general results about rotated functions are derived in this section.

Theorem. If **x ⊂ y** and **rotate[y]** is a function, then so is **rotate[x]**.

```
In[11]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → P[x], v → P[y], w → image[inverse[IMAGE[ROT]], FUNS]}] // Reverse
```

```
Out[11]= or[FUNCTION[rotate[x]], not[FUNCTION[rotate[y]]], not[subclass[x, y]]] == True
```

```
In[12]:= or[FUNCTION[rotate[x_]], not[FUNCTION[rotate[y_]]], not[subclass[x_, y_]]] := True
```

Dual Theorem.

```
In[13]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → P[x], v → P[y], w → image[inverse[IMAGE[inverse[ROT]]], FUNS]}] // Reverse

Out[13]= or[FUNCTION[rotate[composite[x, SWAP]]],
           not[FUNCTION[rotate[composite[y, SWAP]]]], not[subclass[x, y]]] == True

In[14]:= or[FUNCTION[rotate[composite[x_, SWAP]]],
           not[FUNCTION[rotate[composite[y_, SWAP]]]], not[subclass[x_, y_]]] := True
```

---

main theorem

Lemma.

```
In[27]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → intersection[x, P[gp[y]]],
            v → P[gp[y]], w -> image[inverse[IMAGE[ROT]], FUNS]}] // Reverse

Out[27]= subclass[image[IMAGE[ROT], intersection[x, P[gp[y]]], FUNS] == True

In[28]:= subclass[image[IMAGE[ROT], intersection[x_, P[gp[y_]]], FUNS] := True
```

Dual Lemma.

```
In[29]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → intersection[x, P[gp[y]]], v → P[gp[y]],
            w -> image[inverse[IMAGE[inverse[ROT]]], FUNS]}] // Reverse

Out[29]= subclass[image[IMAGE[inverse[ROT]], intersection[x, P[gp[y]]], FUNS] == True

In[30]:= subclass[image[IMAGE[inverse[ROT]], intersection[x_, P[gp[y_]]], FUNS] := True
```

Lemma. (This has a reundant literal.)

```
In[38]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
            {u → x, v -> P[gp[y]], w -> image[inverse[IMAGE[ROT]], FUNS]}] // Reverse // MapNotNot

Out[38]= or[FUNCTION[rotate[x]], not[member[x, V]], not[subclass[x, gp[y]]]] == True

In[39]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. Any subclass of a group is a set.

```
In[42]:= SubstTest[implies, and[subclass[x, t], member[t, V]], member[x, V], t → gp[y]] // Reverse

Out[42]= or[member[x, V], not[subclass[x, gp[y]]]] == True

In[44]:= or[member[x_, V], not[subclass[x_, gp[y_]]]] := True
```

Theorem. The rotation of any subset of a group is a function.

```
In[46]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → subclass[x, gp[y]], p2 → member[x, V], p3 → FUNCTION[rotate[x]]}]] // Reverse

Out[46]= or[FUNCTION[rotate[x]], not[subclass[x, gp[y]]]] == True

In[47]:= or[FUNCTION[rotate[x_]], not[subclass[x_, gp[y_]]]] := True
```

Lemma.

```
In[54]:= SubstTest[implies, subclass[u, gp[v]],
           FUNCTION[rotate[u]], {u → flip[x], v → flip[gp[y]]}] // Reverse

Out[54]= or[FUNCTION[rotate[composite[x, SWAP]]],
           not[subclass[composite[x, id[cart[V, V]]], gp[y]]]] == True

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

Dual Theorem. The reverse rotation of any subset of a group is a function.

```
In[57]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> subclass[x, gp[y]], p2 -> subclass[composite[x, id[cart[V, V]]], gp[y]],
            p3 -> FUNCTION[rotate[composite[x, SWAP]]]}]] // Reverse

Out[57]= or[FUNCTION[rotate[composite[x, SWAP]]], not[subclass[x, gp[y]]]] == True

In[59]:= or[FUNCTION[rotate[composite[x_, SWAP]]], not[subclass[x_, gp[y_]]]] := True
```

Wrapper free versions can be derived.

Theorem. If $\mathbf{x} \subset \mathbf{y}$ and if $\mathbf{y}$ is a group, then **rotate[x]** is a function.

```
In[61]:= SubstTest[implies, equal[y, gp[t]],
           or[FUNCTION[rotate[x]], not[subclass[x, y]]], t → y] // Reverse // MapNotNot

Out[61]= or[FUNCTION[rotate[x]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[62]:= or[FUNCTION[rotate[x_]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Dual Theorem.

```
In[63]:= SubstTest[implies, equal[y, gp[t]], or[FUNCTION[rotate[composite[x, SWAP]]],
            not[subclass[x, y]]], t → y] // Reverse // MapNotNot

Out[63]= or[FUNCTION[rotate[composite[x, SWAP]]],
           not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[64]:= or[FUNCTION[rotate[composite[x_, SWAP]]],
           not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

The main theorem follows from the fact that any nonempty finite cancellative semigroup is a group. This result was derived in the recently posted notebook **cxlfinbo.nb**.

Main Theorem.

```
In[70]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p1, p2, p3], p4], not[implies[p1, p4]],
           {p1 → member[x, intersection[FINITE, SEMIGPS, P[gp[y]]]], p2 → FUNCTION[rotate[x]],
            p3 -> FUNCTION[rotate[flip[x]]], p4 → or[empty[x], member[x, GROUPS]]}]] // Reverse

Out[70]= or[equal[0, x], member[x, GROUPS], not[member[x, FINITE]],
           not[member[x, SEMIGPS]], not[subclass[x, gp[y]]]] == True

In[72]:= or[equal[0, x_], member[x_, GROUPS], not[member[x_, FINITE]],
           not[member[x_, SEMIGPS]], not[subclass[x_, gp[y_]]]] := True
```

Corollary. (Remove the **gp** wrapper.) Any nonempty finite subsemigroup of a group is a group.

```
In[74]:= SubstTest[implies, equal[y, gp[t]],
           or[equal[0, x], member[x, GROUPS], not[member[x, FINITE]],
            not[member[x, SEMIGPS]], not[subclass[x, y]]], t → y] // Reverse // MapNotNot

Out[74]= or[equal[0, x], member[x, GROUPS], not[member[x, FINITE]],
           not[member[x, SEMIGPS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[75]:= or[equal[0, x_], member[x_, GROUPS], not[member[x_, FINITE]],
           not[member[x_, SEMIGPS]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

eliminating variables

In this section the variables **x** and **y** in the main theorem are eliminated. One can eliminate both variables using **reify**, but it is slightly faster to eliminate one of the variables using the **Normality** test.

Theorem. (Eliminate one of the two variables.)

```
In[80]:= Map[equal[V, #] &, complement[
           dif[intersection[FINITE, SEMIGPS, P[gp[x]]], union[GROUPS, set[0]]]] // Normality]

Out[80]= subclass[intersection[FINITE, SEMIGPS, P[gp[x]]], union[GROUPS, set[0]]] == True

In[81]:= subclass[intersection[FINITE, SEMIGPS, P[gp[x_]]], union[GROUPS, set[0]]] := True
```

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

```
In[84]:= SubstTest[implies, equal[x, gp[t]], subclass[intersection[FINITE, SEMIGPS, P[x]],
            union[GROUPS, set[0]]], t → x] // Reverse // MapNotNot

Out[84]= or[not[member[x, GROUPS]],
           subclass[intersection[FINITE, SEMIGPS, P[x]], union[GROUPS, set[0]]]] == True

In[85]:= or[not[member[x_, GROUPS]],
           subclass[intersection[FINITE, SEMIGPS, P[x_]], union[GROUPS, set[0]]]] := True
```

Corollary. A variable-free restatement.

```
In[86]:= Map[equal[V, domain[#]] &, SubstTest[reify, x,
           case[or[not[member[x, GROUPS]], subclass[intersection[FINITE, SEMIGPS, P[x]], t]]],
           t -> union[GROUPS, set[0]]]]

Out[86]= subclass[intersection[FINITE, SEMIGPS, image[inverse[S], GROUPS]],
           union[GROUPS, set[0]]] == True

In[87]:= subclass[intersection[FINITE, SEMIGPS, image[inverse[S], GROUPS]],
           union[GROUPS, set[0]]] := True
```