groups as monoids with inverses

Johan G. F. Belinfante
2011 May 17

```
In[1]:= SetDirectory["l:"]; << goedel.11may16a

        :Package Title: goedel.11may16a              2011 May 16 at 11:40 a.m.

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

        It is now:  2011 May 17 at 12:14

        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 May 17 at 12:25
```

summary

There are many equivalent ways to define groups. Several of these are described in the following reference, pages 1-7.

```
In[2]:= "Marshall Hall, Jr., The Theory of Groups, The MacNillan Company, New York, 1959.";
```

The **GOEDEL** program follows Kurosh in defining a **group** as a semigroup that is also a quasigroup. This definition has the advantage that it is does not involve quantified variables.

```
In[3]:= "A. G. Kurosh, The Theory of Groups, Second Edition, translated from the Russian
         by K. A. Hirsch, two volumes, Chelsea Publishing Co., New York, 1955.";
```

For many purposes, the more conventional definition is convenient. Only a few rewrite rules concerning the equivalence of the Kurosh definition with other characterizations of groups are available in the **GOEDEL** program. At this point it has only been established that a group is a monoid in which every element has a (two-sided) inverse.

```
In[4]:= implies[member[x, GROUPS],
          and[member[x, MONOIDS], equal[domain[inv[x]], range[x]]]] // not // not

Out[4]= True
```

In this notebook various forms of the converse are derived, both with and without variables.

introduction

The statement **x ∈ MONOIDS** means that **x** is the composition law for a monoid. It will be said for short that **x** is a **monoid** even though it is more traditional to reserve the term monoid for **range[x]**. If **u ∈ range[x]**, one says that **u** is an **element** of the monoid. An element **u ∈ domain[inv[x]]** is said to be (two-sided) **invertible**. An element **u** of a monoid is said to be **left-cancellable** if the function **x ∘ LEFT[u]** is one-to-one, and **right-cancellable** if **x ∘ RIGHT[u]** is one-to-one. Every element of a monoid is left cancellable if and only if **rotate[x]** is a function. This fact is not limited to monoids.

```
In[5]:= assert[forall[u, FUNCTION[inverse[composite[x, LEFT[u]]]]]]
Out[5]= FUNCTION[rotate[x]]
```

The **left** and **right divisibility** relations for a binary operation **x** are defined as follows.

```
In[6]:= leftdivisibility[x]
Out[6]= composite[x, inverse[FIRST]]

In[7]:= rightdivisibility[x]
Out[7]= composite[x, inverse[SECOND]]
```

It will be shown below that if **x** is a monoid in which every element is invertible, then **x** is a quasigroup, that is, both **rotate[x]** and **rotate[flip[x]]** are binary operations. (It suffices to show only one of these statements because one can use duality to establish the other.) To further simplify the task, one can make used of the fact that if every element of a binary operation is left and right cancellable, and if both the left and right divisibility relations of the binary operation are cartesian squares, then it is a quasigroup operation.

```
In[8]:= or[member[x, QUASIGPS], not[equal[cart[y, y], composite[x, inverse[FIRST]]]],
          not[equal[cart[z, z], composite[x, inverse[SECOND]]]], not[FUNCTION[rotate[x]]],
          not[FUNCTION[rotate[composite[x, SWAP]]]], not[member[x, BINOPS]]]
Out[8]= True
```

left and right cancellability

Lemma.

```
In[9]:= SubstTest[implies, and[equal[u, v], FUNCTION[v]], FUNCTION[u],
           {u → rotate[x], v -> composite[rotate[x], id[cart[V, range[x]]]]}] // Reverse
Out[9]= or[FUNCTION[rotate[x]], not[FUNCTION[composite[rotate[x], id[cart[V, range[x]]]]]],
           not[subclass[domain[domain[x]], range[x]]] == True

In[10]:= (% /. x → x_) /. Equal → SetDelayed
```

The derivation of the following theorem is speeded up by leaving out three steps as indicated by **(* ⋯ *)**. These missing steps are provided by rewrite rules in the **GOEDEL** program.

Theorem. If every element of a monoid is invertible, then every element is left-cancellable.

```
In[11]:= Map[not,
           SubstTest[and, (* implies[p1,p3], implies[and[p2,p3],p4], *) implies[p1, p5],
            (* implies[p5,p6], *) implies[and[p4, p6], p7], not[implies[and[p1, p2], p7]],
            {p1 -> member[x, MONOIDS], p2 → equal[domain[inv[x]], range[x]],
             p3 -> FUNCTION[composite[rotate[x], id[cart[V, domain[inv[x]]]]]],
             p4 → FUNCTION[composite[rotate[x], id[cart[V, range[x]]]]],
             p5 → equal[domain[domain[x]], range[x]],
             p6 -> subclass[domain[domain[x]], range[x]], p7 → FUNCTION[rotate[x]]}]] // Reverse

Out[11]= or[FUNCTION[rotate[x]],
           not[equal[domain[inv[x]], range[x]]], not[member[x, MONOIDS]]] == True

In[12]:= or[FUNCTION[rotate[x_]],
           not[equal[domain[inv[x_]], range[x_]]], not[member[x_, MONOIDS]]] := True
```

The dual statement is derived in two steps, first replacing **x** with **flip[semigp[x]]** and then eliminating the **semigp** wrapper.

Lemma. (The first step toward the dual theorem.)

```
In[13]:= SubstTest[or, FUNCTION[rotate[t]], not[equal[domain[inv[t]], range[t]]],
           not[member[t, MONOIDS]], t → flip[semigp[x]]] // Reverse

Out[13]= or[FUNCTION[rotate[composite[semigp[x], SWAP]]],
           not[equal[domain[inv[semigp[x]]], range[semigp[x]]]],
           not[member[e[semigp[x]], V]]] == True

In[14]:= (% /. x → x_) /. Equal → SetDelayed
```

Dual Theorem. If every element of a monoid is invertible, then every element is right-cancellable.

```
In[15]:= SubstTest[implies, equal[x, semigp[t]],
            or[FUNCTION[rotate[composite[x, SWAP]]], not[equal[domain[inv[x]], range[x]]],
             not[member[e[x], V]]], t → x] // Reverse // MapNotNot

Out[15]= or[FUNCTION[rotate[composite[x, SWAP]]],
           not[equal[domain[inv[x]], range[x]]], not[member[x, MONOIDS]]] == True

In[16]:= or[FUNCTION[rotate[composite[x_, SWAP]]],
           not[equal[domain[inv[x_]], range[x_]]], not[member[x_, MONOIDS]]] := True
```

left and right divisibility

Lemma. A group is a monoid that is also a quasigroup.

```
In[17]:= SubstTest[member, x, intersection[u, v], {u → QUASIGPS, v → MONOIDS}]

Out[17]= and[member[x, MONOIDS], member[x, QUASIGPS]] == member[x, GROUPS]

In[18]:= and[member[x_, MONOIDS], member[x_, QUASIGPS]] := member[x, GROUPS]
```

The derivation of the main theorem is speeded up by omitting most of the proof steps.

Main Theorem. If every element of a monoid is invertible, then the monoid is a group.

```
In[19]:= Map[not,
           SubstTest[and, (* implies[p1,p2],implies[p1,p3],implies[p1,p4],implies[p1,p5],
             implies[p1,p6], *) implies[and[p2, p3, p4, p5, p6], p7], not[implies[p1, p8]],
            {p1 -> and[member[x, MONOIDS], equal[range[x], domain[inv[x]]]],
             p2 -> equal[cart[range[x], range[x]], composite[x, inverse[FIRST]]],
             p3 -> equal[cart[range[x], range[x]], composite[x, inverse[SECOND]]],
             p4 → member[x, BINOPS], p5 → FUNCTION[rotate[x]], p6 → FUNCTION[rotate[flip[x]]],
             p7 → member[x, QUASIGPS], p8 → member[x, GROUPS]}]] // Reverse

Out[19]= or[member[x, GROUPS],
           not[equal[domain[inv[x]], range[x]]], not[member[x, MONOIDS]]] == True

In[20]:= or[member[x_, GROUPS],
           not[equal[domain[inv[x_]], range[x_]]], not[member[x_, MONOIDS]]] := True
```

Corollary. (Combining the main theorem with its converse.)

```
In[21]:= equiv[and[member[x, MONOIDS], equal[domain[inv[x]], range[x]]],
            member[x, GROUPS]] // not // not

Out[21]= True

In[24]:= and[equal[domain[inv[x_]], range[x_]], member[x_, MONOIDS]] := member[x, GROUPS]
```

variable-free expression of invertibility

Lemma.

```
In[25]:= ((member[pair[x, y], composite[z, funpart[t]]]) // AssertTest) /. t → INV

Out[25]= member[pair[x, y], composite[z, INV]] ==
          and[member[x, V], member[y, V], member[pair[inv[x], y], z]]

In[26]:= member[pair[x_, y_], composite[z_, INV]] :=
          and[member[x, V], member[y, V], member[pair[inv[x], y], z]]
```

Lemma. Simplification rule.

```
In[27]:= equiv[and[member[x, y], member[inv[x], V]], member[x, y]]

Out[27]= True
```

```
In[28]:= and[member[x_, y_], member[inv[x_], V]] := member[x, y]
```

Theorem.

```
In[29]:= (member[x, fix[y]] // AssertTest) /.
          y → composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]
Out[29]= member[x, fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] ==
          and[equal[domain[inv[x]], range[x]], member[x, V]]

In[30]:= member[x_, fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] :=
          and[equal[domain[inv[x]], range[x]], member[x, V]]
```

Theorem. Variable-free characterization of groups as monoids whose elements are all invertible.

```
In[31]:= intersection[MONOIDS,
           fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] // Normality
Out[31]= intersection[MONOIDS,
           fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] == GROUPS

In[32]:= intersection[MONOIDS,
           fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] := GROUPS
```

Corollary.

```
In[33]:= SubstTest[subclass, intersection[u, v], v, {u -> MONOIDS,
            v -> fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]}] // Reverse
Out[33]= subclass[GROUPS, fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] == True

In[34]:= subclass[GROUPS, fix[composite[inverse[IMAGE[SECOND]], IMAGE[FIRST], INV]]] := True
```