# finite cancellative binary operations

Johan G. F. Belinfante
2012 March 3

```
In[1]:= SetDirectory["l:"]; << goedel.12mar02a

        :Package Title: goedel.12mar02a                 2012 March 2 at 3:30 p.m.

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

        It is now:  2012 Mar 3 at 9:49

        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 3 at 10:6
```

## summary

A cancellative finite binary operation is a quasigroup. As a corollary it follows that a nonempty finite cancellative semigroup is a group.

## finite binary operations

Lemma.

```
In[6]:= SubstTest[member, domain[funpart[t]], FINITE, t → binop[x]] // Reverse

Out[6]= member[domain[binop[x]], FINITE] == member[binop[x], FINITE]

In[7]:= member[domain[binop[x_]], FINITE] := member[binop[x], FINITE]
```

Theorem.

```
In[9]:= SubstTest[member, cartsq[t], FINITE, t → fix[domain[binop[x]]]]

Out[9]= member[fix[domain[binop[x]]], FINITE] == member[binop[x], FINITE]

In[10]:= member[fix[domain[binop[x_]]], FINITE] := member[binop[x], FINITE]
```

Theorem.

```
In[24]:= SubstTest[member, domain[binop[t]], FINITE, t → flip[binop[x]]]

Out[24]= member[composite[binop[x], SWAP], FINITE] == member[binop[x], FINITE]

In[25]:= member[composite[binop[x_], SWAP], FINITE] := member[binop[x], FINITE]
```

## double wrapper binop[fin[x]]

The compound wrapper **binop[fin[x]]** is studied in this section.

Theorem.

```
In[16]:= SubstTest[implies, member[t, FINITE],
           member[range[t], FINITE], t → binop[fin[x]]] // Reverse

Out[16]= member[range[binop[fin[x]]], FINITE] == True

In[17]:= member[range[binop[fin[x_]]], FINITE] := True
```

Theorem.

```
In[19]:= SubstTest[member, domain[funpart[t]], FINITE, t -> composite[binop[fin[x]], LEFT[y]]]

Out[19]= member[composite[binop[fin[x]], LEFT[y]], FINITE] == True

In[20]:= member[composite[binop[fin[x_]], LEFT[y_]], FINITE] := True
```

Theorem.

```
In[21]:= SubstTest[member, domain[funpart[t]], FINITE, t -> composite[binop[fin[x]], RIGHT[y]]]

Out[21]= member[composite[binop[fin[x]], RIGHT[y]], FINITE] == True

In[22]:= member[composite[binop[fin[x_]], RIGHT[y_]], FINITE] := True
```

## cancellative finite binary operations

A binary operation **x** is cancellative if **rotate[x]** and **rotate[flip[x]]** are both functions. This implies that left and right multiplications by any element **y** of **x** are both one-to-one.

```
In[2]:= implies[FUNCTION[rotate[x]], FUNCTION[inverse[composite[x, LEFT[y]]]]]

Out[2]= True
```

An **element** of a binary operation **x** is a member of **fix[domain[x]]**.

Theorem. An application of the pigeon hole principle for unary operations to left-multiplications by elements of a finite binary operation.

```
In[27]:= SubstTest[implies, and[member[t, UNOPS], member[t, FINITE], FUNCTION[inverse[t]]],
           equal[domain[t], range[t]], t -> composite[binop[fin[x]], LEFT[y]]] // Reverse

Out[27]= or[equal[fix[domain[binop[fin[x]]]], image[binop[fin[x]], cart[set[y], V]]],
           not[FUNCTION[composite[inverse[LEFT[y]], inverse[binop[fin[x]]]]]],
           not[member[y, fix[domain[binop[fin[x]]]]]]] == True

In[28]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p3],
            implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 → FUNCTION[rotate[binop[fin[x]]]], p2 -> member[y, fix[domain[binop[fin[x]]]]],
             p3 -> FUNCTION[composite[inverse[LEFT[y]], inverse[binop[fin[x]]]], p4 -> equal[
               image[binop[fin[x]], cart[set[y], V]], fix[domain[binop[fin[x]]]]]}]] // Reverse

Out[29]= or[equal[fix[domain[binop[fin[x]]]], image[binop[fin[x]], cart[set[y], V]]],
           not[FUNCTION[rotate[binop[fin[x]]]]],
           not[member[y, fix[domain[binop[fin[x]]]]]]] == True

In[30]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

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

```
In[31]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, y, case[or[equal[image[binop[fin[x]], cart[set[y], t]],
              fix[domain[binop[fin[x]]]]], not[FUNCTION[rotate[binop[fin[x]]]]],
             not[member[y, fix[domain[binop[fin[x]]]]]]], t → V]]

Out[31]= or[not[FUNCTION[rotate[binop[fin[x]]]]],
           subclass[domain[binop[fin[x]]], composite[FIRST, inverse[binop[fin[x]]]]]] == True

In[32]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[34]:= SubstTest[subclass, composite[FIRST, inverse[t]],
           cartsq[fix[domain[t]]], t → binop[x]] // Reverse

Out[34]= subclass[composite[FIRST, inverse[binop[x]]], domain[binop[x]]] == True

In[35]:= subclass[composite[FIRST, inverse[binop[x_]]], domain[binop[x_]]] := True
```

Corollary.

```
In[36]:= or[not[FUNCTION[rotate[binop[fin[x]]]]],
           equal[domain[binop[fin[x]]], composite[FIRST, inverse[binop[fin[x]]]]]] // AssertTest

Out[36]= or[equal[composite[FIRST, inverse[binop[fin[x]]]], domain[binop[fin[x]]]],
           not[FUNCTION[rotate[binop[fin[x]]]]]] == True

In[37]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[38]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> FUNCTION[rotate[binop[fin[x]]]],
            p2 → equal[composite[FIRST, inverse[binop[fin[x]]]], domain[binop[fin[x]]]],
            p3 -> equal[fix[composite[FIRST, inverse[binop[fin[x]]]]],
              fix[domain[binop[fin[x]]]]]}]] // Reverse

Out[38]= or[equal[fix[composite[FIRST, inverse[binop[fin[x]]]]], fix[domain[binop[fin[x]]]]],
          not[FUNCTION[rotate[binop[fin[x]]]]] == True

In[39]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[40]:= SubstTest[implies, and[member[t, V], equal[domain[t], cartsq[range[t]]], FUNCTION[t]],
          member[t, BINOPS], t → rotate[binop[fin[x]]]] // Reverse

Out[40]= or[member[rotate[binop[fin[x]]], BINOPS],
          not[member[rotate[binop[fin[x]]], map[domain[binop[fin[x]]], V]]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[42]:= Map[implies[#, member[rotate[binop[fin[x]]], BINOPS]] &,
          member[rotate[binop[fin[x]]], map[domain[binop[fin[x]]], V]] // AssertTest] // Reverse

Out[42]= or[member[rotate[binop[fin[x]]], BINOPS], not[FUNCTION[rotate[binop[fin[x]]]]], not[
           subclass[domain[binop[fin[x]]], composite[FIRST, inverse[binop[fin[x]]]]]]] == True

In[43]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[44]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           not[implies[p1, p3]], {p1 -> FUNCTION[rotate[binop[fin[x]]]],
            p2 -> subclass[domain[binop[fin[x]]], composite[FIRST, inverse[binop[fin[x]]]]],
            p3 -> member[rotate[binop[fin[x]]], BINOPS]}]] // Reverse

Out[44]= or[member[rotate[binop[fin[x]]], BINOPS], not[FUNCTION[rotate[binop[fin[x]]]]]] == True

In[45]:= or[member[rotate[binop[fin[x_]]], BINOPS],
          not[FUNCTION[rotate[binop[fin[x_]]]]]] := True
```

Dual Theorem.

```
In[47]:= SubstTest[implies, FUNCTION[rotate[binop[fin[t]]]],
          member[rotate[binop[fin[t]]], BINOPS], t → flip[binop[fin[x]]]] // Reverse

Out[47]= or[member[rotate[composite[binop[fin[x]], SWAP]], BINOPS],
          not[FUNCTION[rotate[composite[binop[fin[x]], SWAP]]]]] == True
```

```
In[48]:= or[member[rotate[composite[binop[fin[x_]], SWAP]], BINOPS],
           not[FUNCTION[rotate[composite[binop[fin[x_]], SWAP]]]]] := True
```

Theorem. (Eliminate the double wrappers.)

```
In[49]:= SubstTest[implies, equal[x, binop[fin[t]]],
           or[member[rotate[x], BINOPS], not[FUNCTION[rotate[x]]], t → x] // Reverse

Out[49]= or[member[rotate[x], BINOPS], not[FUNCTION[rotate[x]]],
           not[member[x, BINOPS]], not[member[x, FINITE]]] == True

In[50]:= or[member[rotate[x_], BINOPS], not[FUNCTION[rotate[x_]]],
           not[member[x_, BINOPS]], not[member[x_, FINITE]]] := True
```

Dual Theorem.

```
In[51]:= SubstTest[implies, equal[x, binop[fin[t]]],
           or[member[rotate[flip[x]], BINOPS], not[FUNCTION[rotate[flip[x]]]], t → x] // Reverse

Out[51]= or[member[rotate[composite[x, SWAP]], BINOPS],
           not[FUNCTION[rotate[composite[x, SWAP]]]],
           not[member[x, BINOPS]], not[member[x, FINITE]]] == True

In[52]:= or[member[rotate[composite[x_, SWAP]], BINOPS],
           not[FUNCTION[rotate[composite[x_, SWAP]]]],
           not[member[x_, BINOPS]], not[member[x_, FINITE]]] := True
```

Main Theorem. A non-empty finite cancellative binary operation is a quasigroup.

```
In[53]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p1, p2, p3], p4], not[implies[p1, p4]],
            {p1 → and[member[x, FINITE], member[x, BINOPS], FUNCTION[rotate[x]],
               FUNCTION[rotate[flip[x]]]], p2 → member[rotate[x], BINOPS],
             p3 → member[rotate[flip[x]], BINOPS], p4 → member[x, QUASIGPS]}]] // Reverse

Out[53]= or[member[x, QUASIGPS], not[FUNCTION[rotate[x]]],
           not[FUNCTION[rotate[composite[x, SWAP]]]],
           not[member[x, BINOPS]], not[member[x, FINITE]]] == True

In[55]:= or[member[x_, QUASIGPS], not[FUNCTION[rotate[composite[x_, SWAP]]]],
           not[FUNCTION[rotate[x_]]], not[member[x_, BINOPS]], not[member[x_, FINITE]]] := True
```

Technical Lemma.

```
In[60]:= member[x, dif[intersection[FINITE, BINOPS, image[inverse[IMAGE[ROT]], FUNS],
            image[inverse[IMAGE[inverse[ROT]]], FUNS]], QUASIGPS]] // NotNotTest

Out[60]= and[FUNCTION[rotate[x]], FUNCTION[rotate[composite[x, SWAP]]], member[x, BINOPS],
           member[x, FINITE], member[composite[rotate[composite[x, SWAP]], SWAP], V],
           not[member[x, QUASIGPS]]] == False

In[61]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary. (Variable-free formulation.)

```
In[63]:= Map[empty, SubstTest[class, x, member[x, t],
           t -> dif[intersection[FINITE, BINOPS, image[inverse[IMAGE[ROT]], FUNS],
              image[inverse[IMAGE[inverse[ROT]]], FUNS]], QUASIGPS]]]

Out[63]= subclass[intersection[BINOPS, FINITE, image[inverse[IMAGE[ROT]], FUNS],
           image[inverse[IMAGE[inverse[ROT]]], FUNS]], QUASIGPS] == True

In[64]:= subclass[intersection[BINOPS, FINITE, image[inverse[IMAGE[ROT]], FUNS],
           image[inverse[IMAGE[inverse[ROT]]], FUNS]], QUASIGPS] := True
```

## a corollary

In this section a corollary is derived that follows immediately from Kurosh's characterization of a group as a nonempty associative quasigroup. This was in fact taken as the definition of a group in the **GOEDEL** program in the posted notebook **groups.nb** dated 2008 October 18. In this approach, the usual axioms for a group become theorems that require proof. In the posted notebook **grp-ids.nb** dated 2009 January 26 the fact that groups have a neutral element was derived in a manner akin to the proof given on page 31 in the book by Kurosh.

```
"A. G. Kurosh, Lectures on General Algebra, English
 Translation by K. A. Hirsch, Chelsea Publishing Co., New York, 1963.";
```

The fact that every element of the range of a group has an inverse satisfying the usual axioms was shown in in the posted notebook **gp-inv-1.nb** dated 2009 Feb 1. In this notebook, the explicit use of the concept of quasigroup is eliminated from Kurosh's characterization of a group.

Corollary. A nonempty finite cancellative semigroup is a group.

```
In[66]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → id[SEMIGPS], u -> intersection[BINOPS, FINITE, image[inverse[IMAGE[ROT]], FUNS],
              image[inverse[IMAGE[inverse[ROT]]], FUNS]], v → QUASIGPS}] // Reverse

Out[66]= subclass[intersection[FINITE, SEMIGPS, image[inverse[IMAGE[ROT]], FUNS],
           image[inverse[IMAGE[inverse[ROT]]], FUNS]], union[GROUPS, set[0]]] == True

In[67]:= subclass[intersection[FINITE, SEMIGPS, image[inverse[IMAGE[ROT]], FUNS],
           image[inverse[IMAGE[inverse[ROT]]], FUNS]], union[GROUPS, set[0]]] := True
```