direct products of groups

Johan G. F. Belinfante
2008 December 28

```
In[1]:= SetDirectory["l:"]; << goedel.08dec26a;<< tools.m

        :Package Title: goedel.08dec26a         2008 December 26 at 6:45 a.m.

        It is now:  2008 Dec 28 at 6:15

        Loading Simplification Rules

        TOOLS.M                                      Revised 2008 December 26

        weightlimit = 40
```

summary

The direct product of two groups is a group.

reference

Following Kurosh, a group is defined in the **GOEDEL** program as a nonempty associative quasigroup. The theorem about direct products of groups is accordingly derived as a corollary of an analogous result about quasigroups.

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

direct products of quasigroups

The **GOEDEL** program already has a rewrite rule about direct products of binary operations. It is convenient to introduce an additional version of this with **binop** wrappers.

Theorem. The direct product of two binary operations is a binary operation.

```
In[2]:= SubstTest[implies, and[member[u, BINOPS], member[v, BINOPS]],
          member[direct[u, v], BINOPS], {u → binop[x], v → binop[y]}] // Reverse

Out[2]= member[composite[cross[binop[x], binop[y]], TWIST], BINOPS] == True

In[3]:= member[composite[cross[binop[x_], binop[y_]], TWIST], BINOPS] := True
```

Observation. Direct products preserve rotations:

```
In[4]:= rotate[direct[x, y]] == direct[rotate[x], rotate[y]]

Out[4]= True
```

Lemma.

```
In[5]:= SubstTest[member, composite[cross[binop[u], binop[v]], TWIST],
          BINOPS, {u → quasigp[x], v → quasigp[y]}] // Reverse

Out[5]= member[composite[cross[quasigp[x], quasigp[y]], TWIST], BINOPS] == True

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

Lemma.

```
In[7]:= SubstTest[member, composite[cross[binop[u], binop[v]], TWIST],
          BINOPS, {u → rotate[quasigp[x]], v → rotate[quasigp[y]]}] // Reverse

Out[7]= member[composite[cross[rotate[quasigp[x]], rotate[quasigp[y]]], TWIST], BINOPS] == True

In[8]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[9]:= SubstTest[member, composite[cross[binop[u], binop[v]], TWIST], BINOPS,
          {u → rotate[flip[quasigp[x]]], v → rotate[flip[quasigp[y]]]}] // Reverse

Out[9]= member[composite[cross[rotate[composite[quasigp[x], SWAP]],
            rotate[composite[quasigp[y], SWAP]]], TWIST], BINOPS] == True

In[10]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The direct product of quasigroups is a quasigroup.

```
In[11]:= SubstTest[implies,
           and[member[t, BINOPS], member[rotate[t], BINOPS], member[rotate[flip[t]], BINOPS],
           member[t, QUASIGPS], t → direct[quasigp[x], quasigp[y]]] // Reverse

Out[11]= member[composite[cross[quasigp[x], quasigp[y]], TWIST], QUASIGPS] == True

In[12]:= member[composite[cross[quasigp[x_], quasigp[y_]], TWIST], QUASIGPS] := True
```

Corollary. A restatement sans the **quasigp** wrappers.

```
In[13]:= SubstTest[implies, and[equal[x, quasigp[u]], equal[y, quasigp[v]]],
           member[composite[cross[x, y], TWIST], QUASIGPS], {u → x, v → y}] // Reverse

Out[13]= or[member[composite[cross[x, y], TWIST], QUASIGPS],
           not[member[x, QUASIGPS]], not[member[y, QUASIGPS]]] == True

In[14]:= or[member[composite[cross[x_, y_], TWIST], QUASIGPS],
           not[member[x_, QUASIGPS]], not[member[y_, QUASIGPS]]] := True
```

Observation. The function that takes a pair of sets to its direct product is:

```
In[15]:= VERTSECT[reify[x, direct[first[x], second[x]]]]

Out[15]= composite[IMAGE[cross[TWIST, Id]], CROSS]
```

Corollary. A variable free restatement of the theorem about direct products of quasigroups.

```
In[16]:= Map[empty[composite[Id, complement[#]]] &,
          dif[cartsq[QUASIGPS], image[inverse[composite[IMAGE[cross[TWIST, Id]], CROSS]],
              QUASIGPS]] // complement // RelnNormality]

Out[16]= subclass[image[IMAGE[cross[TWIST, Id]],
            image[CROSS, cart[QUASIGPS, QUASIGPS]]], QUASIGPS] == True

In[17]:= subclass[image[IMAGE[cross[TWIST, Id]],
            image[CROSS, cart[QUASIGPS, QUASIGPS]]], QUASIGPS] := True
```

serendipity: a mapping lemma

In the course of finding an alternate (longer) derivation of the theorem in the preceding section, the following lemma was needed. This result may be useful for other applications.

Lemma.

```
In[18]:= Map[implies[subclass[v, w], #] &, SubstTest[implies, and[member[t, x], subclass[x, y]],
            member[t, y], {x → map[u, v], y → map[u, w]}]] // Reverse

Out[18]= or[member[t, map[u, w]], not[member[t, map[u, v]]], not[subclass[v, w]]] == True

In[19]:= or[member[t_, map[u_, w_]], not[member[t_, map[u_, v_]]], not[subclass[v_, w_]]] := True
```

direct products of semigroups

The **GOEDEL** program already has a rewrite rule about direct products of semigroups. It may be convenient to introduce an additional version of this with **semigp** wrappers.

Theorem. The direct product of two semigroups is a semigroup.

```
In[20]:= SubstTest[implies, and[member[u, SEMIGPS], member[v, SEMIGPS]],
            member[direct[u, v], SEMIGPS], {u → semigp[x], v → semigp[y]}] // Reverse

Out[20]= member[composite[cross[semigp[x], semigp[y]], TWIST], SEMIGPS] == True

In[21]:= member[composite[cross[semigp[x_], semigp[y_]], TWIST], SEMIGPS] := True
```

direct products of groups

Lemma.

```
In[31]:= SubstTest[implies, and[equal[x, binop[t]], not[empty[x]]],
           not[empty[domain[domain[x]]]], t → x] // Reverse

Out[31]= or[equal[0, x], not[equal[0, domain[domain[x]]]], not[member[x, BINOPS]]] == True

In[32]:= or[equal[0, x_], not[equal[0, domain[domain[x_]]]], not[member[x_, BINOPS]]] := True

In[38]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[p1, p4]], {p1 → member[x, GROUPS], p2 → not[empty[x]],
             p3 → member[x, BINOPS], p4 → not[empty[domain[domain[x]]]]}]] // Reverse

Out[38]= or[not[equal[0, domain[domain[x]]]], not[member[x, GROUPS]]] == True

In[39]:= or[not[equal[0, domain[domain[x_]]]], not[member[x_, GROUPS]]] := True
```

Lemma.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
            implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
            {p1 → member[x, GROUPS], p2 -> member[y, GROUPS], p3 → member[x, SEMIGPS],
             p4 → member[y, SEMIGPS], p5 → member[direct[x, y], SEMIGPS]}]] // Reverse

Out[34]= or[member[composite[cross[x, y], TWIST], SEMIGPS],
           not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

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

Lemma.

```
In[36]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
            implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
            {p1 → member[x, GROUPS], p2 -> member[y, GROUPS], p3 → member[x, QUASIGPS],
             p4 → member[y, QUASIGPS], p5 → member[direct[x, y], QUASIGPS]}]] // Reverse

Out[36]= or[member[composite[cross[x, y], TWIST], QUASIGPS],
           not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[37]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The direct product of two groups is a group.

```
In[44]:= (Map[not, SubstTest[and, implies[p0, p1], implies[p0, p2], implies[p0, p3],
           implies[p0, p4], implies[and[p1, p2, p5], p6], not[implies[p0, p6]],
           {p0 → and[equal[t, direct[x, y]], member[x, GROUPS], member[y, GROUPS]],
            p1 → member[t, SEMIGPS], p2 → member[t, QUASIGPS],
            p3 → not[empty[domain[domain[x]]]], p4 → not[empty[domain[domain[x]]]],
            p5 → not[empty[t]], p6 → member[t, GROUPS]}]] // Reverse) /. t → direct[x, y]

Out[44]= or[member[composite[cross[x, y], TWIST], GROUPS],
           not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[45]:= or[member[composite[cross[x_, y_], TWIST], GROUPS],
           not[member[x_, GROUPS]], not[member[y_, GROUPS]]] := True
```

Corollary. (Variable-free restatement of the theorem about direct products of groups.)

```
In[46]:= Map[empty[composite[Id, complement[#]]] &,
          dif[cartsq[GROUPS], image[inverse[composite[IMAGE[cross[TWIST, Id]], CROSS]],
           GROUPS]] // complement // RelnNormality]

Out[46]= subclass[image[IMAGE[cross[TWIST, Id]], image[CROSS, cart[GROUPS, GROUPS]]], GROUPS] ==
          True

In[47]:= subclass[
           image[IMAGE[cross[TWIST, Id]], image[CROSS, cart[GROUPS, GROUPS]]], GROUPS] := True
```