unions of chains of groups

Johan G. F. Belinfante
2009 August 25

```
In[1]:= SetDirectory["l:"]; << goedel.09aug24a; << tools.m
```

```
:Package Title: goedel.09aug24a              2009 August 24 at 6:00 p.m.

It is now:  2009 Aug 25 at 11:40

Loading Simplification Rules

TOOLS.M                                    Revised 2009 July 2

weightlimit = 40
```

summary

It is shown in this notebook that the union of a chain of groups is either empty or a group. The derivation uses the definition of a group as a nonempty semigroup that is also a quasigroup. It has already been shown earlier that the union of a chain of semigroups is a semigroup. Most of the work in this notebook is to show that a similar result holds for quasigroups. The result for groups follows as an elementary corollary.

quasigroups

By definition, a quasigroup **x** is a binary operation such that both **rotate[x]** and **rotate[rotate[x]]** are also binary operations. This definition implies that the class of quasigroups is the intersection of three classes:

```
In[2]:= intersection[BINOPS, image[inverse[IMAGE[ROT]], BINOPS],
         image[inverse[IMAGE[inverse[ROT]]], BINOPS]]
```

```
Out[2]= QUASIGPS
```

For the derivation of the theorem about unions of chains of quasigroups it is desirable to have a similar formula for the class of quasigroups that uses direct images in place of inverse images. Such a formula is derived in this section.

Lemma.

```
In[8]:= Map[equal[intersection[BINOPS, image[IMAGE[ROT], BINOPS]], #] &,
          Map[intersection[BINOPS, image[inverse[#], BINOPS]] &,
           Assoc[inverse[IMAGE[ROT]], IMAGE[ROT], IMAGE[inverse[ROT]]]]] // Reverse
```

```
Out[8]= equal[intersection[BINOPS, image[IMAGE[ROT], BINOPS]],
          intersection[BINOPS, image[inverse[IMAGE[inverse[ROT]]], BINOPS]]] == True
```

```
In[9]:= % /. Equal → SetDelayed
```

Lemma.

```
In[12]:= Map[equal[intersection[BINOPS, image[IMAGE[inverse[ROT]], BINOPS]], #] &,
          Map[intersection[BINOPS, image[#, BINOPS]] &,
           Assoc[inverse[IMAGE[ROT]], IMAGE[ROT], IMAGE[inverse[ROT]]]]]
```

```
Out[12]= equal[intersection[BINOPS, image[IMAGE[inverse[ROT]], BINOPS]],
           intersection[BINOPS, image[inverse[IMAGE[ROT]], BINOPS]]] == True
```

```
In[13]:= % /. Equal → SetDelayed
```

Theorem. A formula for the class of quasigroups that uses direct images.

```
In[16]:= SubstTest[implies, and[equal[intersection[t, u], intersection[t, v]],
             equal[intersection[t, x], intersection[t, y]]],
            equal[intersection[t, u, x], intersection[t, v, y]],
            {t → BINOPS, u -> image[IMAGE[inverse[ROT]], BINOPS],
             v -> image[inverse[IMAGE[ROT]], BINOPS], x -> image[IMAGE[ROT], BINOPS],
             y -> image[inverse[IMAGE[inverse[ROT]]], BINOPS]}] // Reverse
```

```
Out[16]= equal[QUASIGPS, intersection[BINOPS,
             image[IMAGE[ROT], BINOPS], image[IMAGE[inverse[ROT]], BINOPS]]] == True
```

```
In[17]:= intersection[BINOPS, image[IMAGE[ROT], BINOPS],
           image[IMAGE[inverse[ROT]], BINOPS]] := QUASIGPS
```

temporary simplification rules

This section contains two temporary simplification rules that will be needed in the next section, where better rewrite rules are derived.

Theorem. Simplification rule.

```
In[18]:= equal[image[IMAGE[id[cart[cart[V, V], V]]], Uchains[image[IMAGE[ROT], BINOPS]]],
          Uchains[image[IMAGE[ROT], BINOPS]]]
```

```
Out[18]= True
```

```
In[19]:= image[IMAGE[id[cart[cart[V, V], V]]], Uchains[image[IMAGE[ROT], BINOPS]]] :=
          Uchains[image[IMAGE[ROT], BINOPS]]
```

Theorem. A similar rule.

```
In[20]:= equal[image[IMAGE[id[cart[cart[V, V], V]]],
           Uchains[image[IMAGE[inverse[ROT]], BINOPS]]],
          Uchains[image[IMAGE[inverse[ROT]], BINOPS]]]
```

```
Out[20]= True
```

```
In[21]:= image[IMAGE[id[cart[cart[V, V], V]]], Uchains[image[IMAGE[inverse[ROT]], BINOPS]]] :=
          Uchains[image[IMAGE[inverse[ROT]], BINOPS]]
```

Uchains theorem

Lemma.

```
In[22]:= ImageComp[BIGCUP, IMAGE[IMAGE[ROT]], x] // Reverse

Out[22]= image[BIGCUP, image[IMAGE[IMAGE[ROT]], x]] == image[IMAGE[ROT], image[BIGCUP, x]]

In[23]:= image[BIGCUP, image[IMAGE[IMAGE[ROT]], x_]] := image[IMAGE[ROT], image[BIGCUP, x]]
```

Lemma.

```
In[24]:= ImageComp[BIGCUP, IMAGE[IMAGE[inverse[ROT]]], x] // Reverse

Out[24]= image[BIGCUP, image[IMAGE[IMAGE[inverse[ROT]]], x]] ==
          image[IMAGE[inverse[ROT]], image[BIGCUP, x]]

In[25]:= image[BIGCUP, image[IMAGE[IMAGE[inverse[ROT]]], x_]] :=
          image[IMAGE[inverse[ROT]], image[BIGCUP, x]]
```

Lemma.

```
In[26]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u -> composite[IMAGE[IMAGE[ROT]], id[chains[S]]],
            v -> composite[id[chains[S]], IMAGE[IMAGE[ROT]]], w → P[x]}] // Reverse

Out[26]= subclass[image[IMAGE[IMAGE[ROT]], intersection[chains[S], P[x]]], chains[S]] == True

In[27]:= subclass[image[IMAGE[IMAGE[ROT]], intersection[chains[S], P[x_]]], chains[S]] := True
```

Theorem.

```
In[28]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → BIGCUP, u -> image[IMAGE[IMAGE[ROT]], intersection[chains[S], P[x]]],
            v -> intersection[chains[S], P[image[IMAGE[ROT], x]]]}] // Reverse

Out[28]= subclass[image[IMAGE[ROT], Uchains[x]], Uchains[image[IMAGE[ROT], x]]] == True

In[29]:= subclass[image[IMAGE[ROT], Uchains[x_]], Uchains[image[IMAGE[ROT], x_]]] := True
```

Similar results hold for **IMAGE[inverse[ROT]]**.

```
In[30]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u -> composite[IMAGE[IMAGE[inverse[ROT]]], id[chains[S]]],
            v -> composite[id[chains[S]], IMAGE[IMAGE[inverse[ROT]]]], w → P[x]}] // Reverse

Out[30]= subclass[
           image[IMAGE[IMAGE[inverse[ROT]]], intersection[chains[S], P[x]]], chains[S]] == True
```

```
In[31]:= subclass[
           image[IMAGE[IMAGE[inverse[ROT]]], intersection[chains[S], P[x_]]], chains[S]] := True
```

Theorem.

```
In[32]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → BIGCUP, u -> image[IMAGE[IMAGE[inverse[ROT]]], intersection[chains[S], P[x]]],
            v -> intersection[chains[S], P[image[IMAGE[inverse[ROT]], x]]]}] // Reverse

Out[32]= subclass[image[IMAGE[inverse[ROT]], Uchains[x]],
           Uchains[image[IMAGE[inverse[ROT]], x]]] == True

In[33]:= subclass[image[IMAGE[inverse[ROT]], Uchains[x_]],
           Uchains[image[IMAGE[inverse[ROT]], x_]]] := True
```

Lemma.

```
In[34]:= SubstTest[subclass, image[IMAGE[inverse[ROT]], Uchains[t]],
           Uchains[image[IMAGE[inverse[ROT]], t]], t → image[IMAGE[ROT], BINOPS]] // Reverse

Out[34]= subclass[image[IMAGE[inverse[ROT]], Uchains[image[IMAGE[ROT], BINOPS]]], BINOPS] == True

In[35]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[36]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IMAGE[ROT], u -> image[IMAGE[inverse[ROT]], Uchains[image[IMAGE[ROT], BINOPS]]],
            v → BINOPS}] // Reverse

Out[36]= subclass[Uchains[image[IMAGE[ROT], BINOPS]], image[IMAGE[ROT], BINOPS]] == True

In[37]:= % /. Equal→ SetDelayed
```

Theorem. The union of a chain of rotated binary operations is a rotated binary operation.

```
In[38]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> Uchains[image[IMAGE[ROT], BINOPS]], v -> image[IMAGE[ROT], BINOPS]}]

Out[38]= equal[image[IMAGE[ROT], BINOPS], Uchains[image[IMAGE[ROT], BINOPS]]] == True

In[39]:= Uchains[image[IMAGE[ROT], BINOPS]] := image[IMAGE[ROT], BINOPS]
```

The roles of **ROT** and **inverse[ROT**] can be exchanged.

Lemma.

```
In[40]:= SubstTest[subclass, image[IMAGE[ROT], Uchains[t]],
           Uchains[image[IMAGE[ROT], t]], t → image[IMAGE[inverse[ROT]], BINOPS]] // Reverse

Out[40]= subclass[image[IMAGE[ROT], Uchains[image[IMAGE[inverse[ROT]], BINOPS]]], BINOPS] == True

In[41]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[42]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IMAGE[inverse[ROT]],
            u -> image[IMAGE[ROT], Uchains[image[IMAGE[inverse[ROT]], BINOPS]]],
            v → BINOPS}] // Reverse

Out[42]= subclass[Uchains[image[IMAGE[inverse[ROT]], BINOPS]],
           image[IMAGE[inverse[ROT]], BINOPS]] == True

In[43]:= % /. Equal→ SetDelayed
```

Theorem. The union of a chain of reverse-rotated binary operations is a reverse-rotated binary operation.

```
In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> Uchains[image[IMAGE[inverse[ROT]], BINOPS]],
           v -> image[IMAGE[inverse[ROT]], BINOPS]}]

Out[44]= equal[image[IMAGE[inverse[ROT]], BINOPS],
           Uchains[image[IMAGE[inverse[ROT]], BINOPS]]] == True

In[45]:= Uchains[image[IMAGE[inverse[ROT]], BINOPS]] := image[IMAGE[inverse[ROT]], BINOPS]
```

Corollary. The union of a chain of quasigroups is a quasigroup.

```
In[46]:= (Map[not,
            SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
             {p1 -> and[equal[Uchains[x], x], equal[Uchains[y], y], equal[Uchains[z], z]],
              p2 → equal[intersection[x, y], Uchains[intersection[x, y]]],
              p3 -> equal[intersection[x, y, z], Uchains[intersection[x, y, z]]]}]] //
            Reverse) /. {x → BINOPS, y→ image[IMAGE[ROT], BINOPS],
           z → image[IMAGE[inverse[ROT]], BINOPS]}

Out[46]= equal[QUASIGPS, Uchains[QUASIGPS]] == True

In[47]:= Uchains[QUASIGPS] := QUASIGPS
```

Corollary. The union of a chain of groups is either empty or a group.

```
In[48]:= SubstTest[implies, and[equal[Uchains[x], x], equal[Uchains[y], y]],
           equal[intersection[x, y], Uchains[intersection[x, y]]],
           {x → QUASIGPS, y→ SEMIGPS}] // Reverse

Out[48]= equal[Uchains[GROUPS], union[GROUPS, set[0]]] == True

In[49]:= Uchains[GROUPS] := union[GROUPS, set[0]]
```

Comments. For convenience, the empty set is considered to be both a semigroup and a quasigroup, but tradition has been followed in requiring groups to be non-empty. Since the union of an empty chain is empty, the empty set is always a member of **Uchains[x]** for any **x**.