rotation-invariant quasigroups

Johan G. F. Belinfante
2008 December 2

```
In[1]:= SetDirectory["l:"]; << goedel.08nov30a;<< tools.m

        :Package Title: goedel.08nov30a          2008 November 30 at 9:55 p.m.

        It is now:  2008 Dec 2 at 8:44

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 21

        weightlimit = 40
```

summary

Invariance under **ROT** is equivalent to invariance under **inverse[ROT]**. If **x** is rotation-invariant, so is **flip[x]**. Any rotation-invariant binary operation is a quasigroup.

rotation-invariance

Temporary Lemma. (The orientation chosen for this rewrite rule is arbitrary.)

```
In[2]:= SubstTest[subclass, rotate[x], rotate[t], t → rotate[x]] // Reverse

Out[2]= subclass[rotate[x], composite[rotate[composite[x, SWAP]], SWAP]] ==
         subclass[composite[x, id[cart[V, V]]], rotate[x]]

In[3]:= subclass[rotate[x_], composite[rotate[composite[x_, SWAP]], SWAP]] :=
         subclass[composite[x, id[cart[V, V]]], rotate[x]]
```

Temporary Lemma.

```
In[4]:= SubstTest[subclass, rotate[u], rotate[v], {u → rotate[rotate[x]], v → x}]

Out[4]= subclass[composite[rotate[composite[x, SWAP]], SWAP], x] ==
         subclass[composite[x, id[cart[V, V]]], rotate[x]]

In[5]:= subclass[composite[rotate[composite[x_, SWAP]], SWAP], x_] :=
         subclass[composite[x, id[cart[V, V]]], rotate[x]]
```

Lemma.

```
In[6]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {w → composite[x, id[cart[V, V]]], u → rotate[x], v → rotate[rotate[x]]}] // Reverse

Out[6]= or[not[subclass[composite[x, id[cart[V, V]]], rotate[x]]],
          subclass[rotate[x], x]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Temporary Lemma.

```
In[8]:= SubstTest[subclass, rotate[u], rotate[v],
         {u → composite[x, id[cart[V, V]]], v → rotate[rotate[x]]}]

Out[8]= subclass[composite[x, id[cart[V, V]]], composite[rotate[composite[x, SWAP]], SWAP]] ==
         subclass[rotate[x], x]

In[9]:= subclass[composite[x_, id[cart[V, V]]], composite[rotate[composite[x_, SWAP]], SWAP]] :=
         subclass[rotate[x], x]
```

Temporary Lemma.

```
In[10]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {v → composite[x, id[cart[V, V]]], u → rotate[x], w → rotate[rotate[x]]}] // Reverse

Out[10]= or[not[subclass[rotate[x], x]],
           subclass[composite[x, id[cart[V, V]]], rotate[x]]] == True

In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

Temporary Lemma.

```
In[12]:= equiv[subclass[rotate[x], x], subclass[composite[x, id[cart[V, V]]], rotate[x]]]

Out[12]= True

In[13]:= subclass[composite[x_, id[cart[V, V]]], rotate[x_]] := subclass[rotate[x], x]
```

Theorem.

```
In[14]:= invar[inverse[ROT]] // Normality

Out[14]= invar[inverse[ROT]] == invar[ROT]

In[15]:= invar[inverse[ROT]] := invar[ROT]
```

Corollary.

```
In[16]:= SubstTest[intersection, subvar[t], invar[t], t → ROT] // Reverse

Out[16]= intersection[invar[ROT], P[cart[cart[V, V], V]]] == fix[IMAGE[ROT]]

In[17]:= intersection[invar[ROT], P[cart[cart[V, V], V]]] := fix[IMAGE[ROT]]
```

if x is rotation-invariant, so is flip[x]

Lemma.

```
In[18]:= SubstTest[or, equal[composite[x, id[cart[V, V]]], rotate[y]],
           not[equal[x, rotate[y]]], y → x] // Reverse

Out[18]= or[equal[composite[x, id[cart[V, V]]], rotate[x]], not[equal[x, rotate[x]]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If **x** is rotation-invariant, so is **flip[x]**.

```
In[20]:= Map[implies[equal[x, rotate[x]], #] &,
           SubstTest[equal, rotate[flip[x]], rotate[flip[y]], y → rotate[x]]] // Reverse

Out[20]= or[equal[composite[x, SWAP], rotate[composite[x, SWAP]]],
           not[equal[x, rotate[x]]]] == True

In[21]:= or[equal[composite[x_, SWAP], rotate[composite[x_, SWAP]]],
           not[equal[x_, rotate[x_]]]] := True
```

Lemma.

```
In[22]:= implies[member[x, fix[IMAGE[ROT]]], member[flip[x], fix[IMAGE[ROT]]]] // NotNotTest

Out[22]= or[and[equal[composite[x, SWAP], rotate[composite[x, SWAP]]],
            member[composite[x, SWAP], V]], not[equal[x, rotate[x]]], not[member[x, V]]] == True

In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[24]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, t], member[image[z, x], t]],
           {t -> fix[IMAGE[ROT]], z → cross[SWAP, Id]}]]

Out[24]= subclass[image[IMAGE[cross[SWAP, Id]], fix[IMAGE[ROT]]], fix[IMAGE[ROT]]] == True

In[25]:= % /. Equal → SetDelayed
```

Lemma.

```
In[26]:= SubstTest[subclass, fix[t], range[t], t → IMAGE[ROT]] // Reverse

Out[26]= subclass[U[fix[IMAGE[ROT]]], cart[cart[V, V], V]] == True

In[27]:= % /. Equal → SetDelayed
```

Theorem.

```
In[28]:= equal[image[IMAGE[id[cart[cart[V, V], V]]], fix[IMAGE[ROT]]], fix[IMAGE[ROT]]]

Out[28]= True

In[29]:= image[IMAGE[id[cart[cart[V, V], V]]], fix[IMAGE[ROT]]] := fix[IMAGE[ROT]]
```

Lemma.

```
In[30]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> IMAGE[cross[SWAP, Id]],
            u -> image[IMAGE[cross[SWAP, Id]], fix[IMAGE[ROT]]], v -> fix[IMAGE[ROT]]}] // Reverse

Out[30]= subclass[fix[IMAGE[ROT]], image[IMAGE[cross[SWAP, Id]], fix[IMAGE[ROT]]]] == True

In[31]:= % /. Equal → SetDelayed

Out[31]= True
```

Theorem.

```
In[32]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> fix[IMAGE[ROT]], v -> image[IMAGE[cross[SWAP, Id]], fix[IMAGE[ROT]]]}]

Out[32]= equal[fix[IMAGE[ROT]], image[IMAGE[cross[SWAP, Id]], fix[IMAGE[ROT]]]] == True

In[33]:= image[IMAGE[cross[SWAP, Id]], fix[IMAGE[ROT]]] := fix[IMAGE[ROT]]
```

rotation-invariant binary operations

Theorem.

```
In[34]:= implies[equal[binop[x], rotate[binop[x]]], member[binop[x], QUASIGPS]] // AssertTest //
           MapNotNot

Out[34]= or[member[binop[x], QUASIGPS], not[equal[binop[x], rotate[binop[x]]]]] == True

In[35]:= or[member[binop[x_], QUASIGPS], not[equal[binop[x_], rotate[binop[x_]]]]] := True
```

Corollary obtained by eliminating the **binop** wrapper.

```
In[36]:= SubstTest[implies, equal[x, binop[t]],
           or[member[x, QUASIGPS], not[equal[x, rotate[x]]]], t → x] // Reverse

Out[36]= or[member[x, QUASIGPS], not[equal[x, rotate[x]]], not[member[x, BINOPS]]] == True

In[37]:= or[member[x_, QUASIGPS], not[equal[x_, rotate[x_]]], not[member[x_, BINOPS]]] := True
```

Corollary obtained by eliminating the variable **x**.

```
In[38]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, u], member[x, v]],
           {u → intersection[BINOPS, fix[IMAGE[ROT]]], v -> QUASIGPS}]]

Out[38]= subclass[intersection[BINOPS, fix[IMAGE[ROT]]], QUASIGPS] == True
```

```
In[39]:= subclass[intersection[BINOPS, fix[IMAGE[ROT]]], QUASIGPS] := True

In[40]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[BINOPS, fix[IMAGE[ROT]]],
           v → intersection[fix[IMAGE[ROT]], QUASIGPS]}]

Out[40]= equal[intersection[BINOPS, fix[IMAGE[ROT]]],
           intersection[QUASIGPS, fix[IMAGE[ROT]]]] == True

In[41]:= intersection[BINOPS, fix[IMAGE[ROT]]] := intersection[QUASIGPS, fix[IMAGE[ROT]]]
```

Corollary.

```
In[42]:= AssInt[BINOPS, P[cart[cart[V, V], V]], invar[ROT]] // Reverse

Out[42]= intersection[BINOPS, invar[ROT]] == intersection[QUASIGPS, fix[IMAGE[ROT]]]

In[43]:= intersection[BINOPS, invar[ROT]] := intersection[QUASIGPS, fix[IMAGE[ROT]]]
```

Corollary.

```
In[44]:= AssInt[invar[ROT], BINOPS, QUASIGPS]

Out[44]= intersection[QUASIGPS, invar[ROT]] == intersection[QUASIGPS, fix[IMAGE[ROT]]]

In[45]:= intersection[QUASIGPS, invar[ROT]] := intersection[QUASIGPS, fix[IMAGE[ROT]]]
```

Comment. Rotation-invariant quasigroups are also called **semisymmetric quasigroups**. See, for example, Smith's book, page 7.

```
In[46]:= "Jonathan D. H. Smith, An Introduction to
          Quasigroups and Their Representations, Chapman and Hall/CRC,
          Boca Raton, Florida, 2007. QA 181.5 .S65. ISBN 1-58488-537-8";
```