duality in group theory

Johan G. F. Belinfante
2009 February 10

```
In[1]:= SetDirectory["l:"]; << goedel.09feb09a;<< tools.m
```

```
:Package Title: goedel.09feb09a            2009 February 9 at 4:00 p.m.

It is now:  2009 Feb 10 at 18:1

Loading Simplification Rules

TOOLS.M                                    Revised 2009 February 9

weightlimit = 40
```

summary

If **x** is a group, so is **flip[x]**, called the **opposite group**. For every theorem in group theory a **dual theorem** is obtained by replacing the group with its opposite. When **gp** wrappers are used, some special rewrite rules are useful to facilitate the application of dualty. A simple example illustrating the principle of duality is presented in the final section.

simplification rules for gp[x]

In this section, some general simplification rules for the **gp** wrapper are derived by using the fact that groups are binary operations.

Theorem.

```
In[2]:= SubstTest[composite, binop[t], id[cart[V, V]], t → gp[x]] // Reverse

Out[2]= composite[gp[x], id[cart[V, V]]] == gp[x]

In[3]:= composite[gp[x_], id[cart[V, V]]] := gp[x]
```

Corollary.

```
In[4]:= ImageComp[gp[x], id[cart[V, V]], V] // Reverse

Out[4]= image[gp[x], cart[V, V]] == range[gp[x]]

In[5]:= image[gp[x], cart[V, V]] := range[gp[x]]
```

Corollary. Inverse images of **gp[x]** are relations.

```
In[6]:= IminComp[gp[x], id[cart[V, V]], y] // Reverse
Out[6]= composite[Id, image[inverse[gp[x]], y]] == image[inverse[gp[x]], y]
```

```
In[7]:= composite[Id, image[inverse[gp[x_]], y_]] := image[inverse[gp[x]], y]
```

The special case that **y** is **V** requires a separate rule.

Corollary. The domain of **gp[x]** is a relation.

```
In[8]:= IminComp[gp[x], id[cart[V, V]], V] // Reverse
Out[8]= composite[Id, domain[gp[x]]] == domain[gp[x]]
```

```
In[9]:= composite[Id, domain[gp[x_]]] := domain[gp[x]]
```

---

quasigroup rules

By definition, a group is a nonempty associative quasigroup. Accordingly, one can obtain many rewrite rules for groups by specializing known facts about quasigroups.

Theorem.

```
In[10]:= SubstTest[implies, member[t, GROUPS], member[t, QUASIGPS], t → gp[x]] // Reverse
Out[10]= member[gp[x], QUASIGPS] == True
```

```
In[11]:= member[gp[x_], QUASIGPS] := True
```

The final corollary of the preceding section can be sharpened:

Theorem. The relation **domain[gp[x]]** is the cartesian square of **range[gp[x]]**.

```
In[13]:= SubstTest[cartsq, range[quasigp[t]], t → gp[x]] // Reverse
Out[13]= cart[range[gp[x]], range[gp[x]]] == domain[gp[x]]
```

```
In[14]:= cart[range[gp[x_]], range[gp[x_]]] := domain[gp[x]]
```

Corollary. The domain of the domain of **gp[x]** is its range.

```
In[15]:= SubstTest[domain, domain[quasigp[t]], t → gp[x]] // Reverse
Out[15]= domain[domain[gp[x]]] == range[gp[x]]
```

```
In[16]:= domain[domain[gp[x_]]] := range[gp[x]]
```

Corollary. The fixed-point class of the domain of **gp[x]** is its range.

In[17]:= **SubstTest[fix, domain[quasigp[t]], t → gp[x]] // Reverse**

Out[17]= fix[domain[gp[x]]] == range[gp[x]]

In[18]:= **fix[domain[gp[x_]]] := range[gp[x]]**

Corollary. The range of the domain of **gp[x]** is its range.

In[19]:= **SubstTest[range, domain[quasigp[t]], t → gp[x]] // Reverse**

Out[19]= range[domain[gp[x]]] == range[gp[x]]

In[20]:= **range[domain[gp[x_]]] := range[gp[x]]**

---

empty gp[x] rules

Theorem. If **x** is not a group, then **gp[x]** is empty.

In[21]:= **Map[implies[#, empty[gp[x]]] &,
SubstTest[empty, intersection[x, image[V, intersection[set[x], t]]], t -> GROUPS]]**

Out[21]= or[equal[0, gp[x]], member[x, GROUPS]] == True

In[22]:= **or[equal[0, gp[x_]], member[x_, GROUPS]] := True**

Theorem. A function is empty if its domain is empty.

In[23]:= **SubstTest[empty, domain[funpart[t]], t → gp[x]] // Reverse**

Out[23]= equal[0, domain[gp[x]]] == equal[0, gp[x]]

In[24]:= **equal[0, domain[gp[x_]]] := equal[0, gp[x]]**

Observation. A similar rule holds for the domain of the domain, but no new rewrite rules are needed, thanks to the results derived in the preceding section:

In[25]:= **equal[0, domain[domain[gp[x]]]]**

Out[25]= equal[0, gp[x]]

---

duality rules for gp[x]

Theorem. Groups are sets.

In[26]:= **SubstTest[member, binop[t], V, t → gp[x]] // Reverse**

Out[26]= member[gp[x], V] == True

In[27]:= **member[gp[x_], V] := True**

Theorem. The **flip[gp[x]]** is a group if and only if **gp[x]** is a group.

```
In[28]:= member[composite[gp[x], SWAP], GROUPS] // AssertTest

Out[28]= member[composite[gp[x], SWAP], GROUPS] == not[equal[0, gp[x]]]

In[29]:= member[composite[gp[x_], SWAP], GROUPS] := not[equal[0, gp[x]]]
```

special wrapper removal rules

Special wrapper removal rules for the **gp** wrapper are derived in this section.

Theorem. Iterated wrapper rule.

```
In[30]:= equal[gp[gp[x]], gp[x]]

Out[30]= True

In[31]:= gp[gp[x_]] := gp[x]
```

Theorem. A similar special wrapper-removal rule for opposite groups.

```
In[32]:= equal[gp[flip[gp[x]]], flip[gp[x]]]

Out[32]= True

In[33]:= gp[composite[gp[x_], SWAP]] := composite[gp[x], SWAP]
```

dual associative laws

The associative law for groups can be formulated without variables for group elements. Two rewrite rules are derived in this section, related by duality.

Theorem. A rewrite rule for the associative law.

```
In[34]:= SubstTest[composite, cat[t], cross[Id, cat[t]], ASSOC, t → gp[x]] // Reverse

Out[34]= composite[gp[x], cross[Id, gp[x]], ASSOC] == composite[gp[x], cross[gp[x], Id]]

In[35]:= composite[gp[x_], cross[Id, gp[x_]], ASSOC] := composite[gp[x], cross[gp[x], Id]]
```

One can use duality to derive a similar result. To obtain a clean rewrite rule, the following lemma is needed.

Lemma.

```
In[47]:= Assoc[cross[x, y], cross[SWAP, Id],
           composite[inverse[ROT], SWAP, cross[Id, SWAP]]] // Reverse

Out[47]= composite[cross[composite[x, SWAP], y], inverse[ROT], SWAP, cross[Id, SWAP]] ==
          composite[cross[x, y], inverse[ASSOC]]

In[48]:= composite[cross[composite[x_, SWAP], y_], inverse[ROT], SWAP, cross[Id, SWAP]] :=
          composite[cross[x, y], inverse[ASSOC]]
```

Application of duality now yields the following:

```
In[49]:= Map[composite[#, cross[SWAP, Id], SWAP] &,
           SubstTest[composite, gp[t], cross[Id, gp[t]], ASSOC, t → flip[gp[x]]]] // Reverse

Out[49]= composite[gp[x], cross[gp[x], Id], inverse[ASSOC]] == composite[gp[x], cross[Id, gp[x]]]
```

The same result can also obtained more directly without explicitly using duality.

Theorem.   A dual rewrite rule for the associative law. (Here the same result is obtained from the general theory of associative relations.)

```
In[50]:= SubstTest[composite, assoc[t], cross[assoc[t], Id], inverse[ASSOC], t → gp[x]] // Reverse

Out[50]= composite[gp[x], cross[gp[x], Id], inverse[ASSOC]] == composite[gp[x], cross[Id, gp[x]]]

In[51]:= composite[gp[x_], cross[gp[x_], Id], inverse[ASSOC]] :=
          composite[gp[x], cross[Id, gp[x]]]
```