two equivalences associated with a subgroup

Johan G. F. Belinfante
2011 June 28

```
In[1]:= SetDirectory["l:"]; << goedel.11jun24a

:Package Title: goedel.11jun24a                2011 June 24 at 3:50 p.m.

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

It is now:  2011 Jun 28 at 18:40

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Jun 28 at 18:51
```

summary

If **x** is a subgroup of a group **y**, then **y ∘ id[V × range[x]] ∘ inverse[FIRST]** and **y ∘ id[V × range[x]] ∘ inverse[SECOND]** are equivalence relations. (Comment. The equivalence classes for these two equivalence relations are called left and right cosets of the subgroup. Two elements of the range of the group are equivalent if they belong to the same coset of the subgroup.)

derivation

It has already been shown previously, as a corollary of a theorem about semigroups, that the relations in question are transitive.

```
In[2]:= implies[subclass[gp[x], gp[y]],
          TRANSITIVE[composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]]]]

Out[2]= True
```

It therefore remains only to show that these relations are symmetric. It suffices to consider just one of these relations, since one can derive the corresponding result for the other using duality. The proof of symmetry is usually done by considering elements of the ranges of the group, but the derivation to be presented here avoids considering elements. It will briefly be explained why it was decided to avoid elements. If one denotes the inverse of an element **u ∈ range[gp[y]]** by **u^**, then the proof anounts to showing that **u^ · v ∈ range[gp[x]]** implies **v^ · u ∈ range[gp[x]]**. This involves both the composition law for the group as well as the group inversion operation, both of which would involve **APPLY**. The combination of the two applications amounts to application of the function **rotate[gp[x]]**.

```
In[3]:= APPLY[rotate[gp[y]], PAIR[u, v]]

Out[3]= APPLY[gp[y], PAIR[APPLY[inv[gp[y]], v], u]]
```

The idea pursued here is to take advantage of various rewrite rules involving **rotate[gp[x]]** instead. Since variables for group elements would later have to be eliminated anyway, it is more efficient not to introduce them in the first place.

Lemma. Simplification rule.

```
In[4]:= composite[gp[x], id[cart[range[gp[x]], y]]] // FastReifTriNormality

Out[4]= composite[gp[x], id[cart[range[gp[x]], y]]] == composite[gp[x], id[cart[V, y]]]

In[5]:= composite[gp[x_], id[cart[range[gp[x_]], y_]]] := composite[gp[x], id[cart[V, y]]]
```

Lemma.

```
In[6]:= SubstTest[implies, and[member[u, GROUPS], subclass[u, v], member[v, GROUPS]],
          equal[image[inv[v], range[u]], range[u]], {u → gp[x], v → gp[y]}] // Reverse

Out[6]= or[equal[0, gp[x]], equal[0, gp[y]],
          equal[image[inv[gp[y]], range[gp[x]]], range[gp[x]]],
          not[subclass[gp[x], gp[y]]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The preceding lemma has redundant literals which will now be removed.

Theorem. If **gp[x] ⊂ gp[y]**, then **range[gp[x]]** is fixed by imaging with **inv[gp[y]]**.

```
In[8]:= SubstTest[and, implies[p, q], or[p, q], {p -> or[equal[0, gp[x]], equal[0, gp[y]]],
           q → or[equal[image[inv[gp[y]], range[gp[x]]], range[gp[x]]],
             not[subclass[gp[x], gp[y]]]]}] // MapNotNot

Out[8]= or[equal[image[inv[gp[y]], range[gp[x]]], range[gp[x]]],
          not[subclass[gp[x], gp[y]]]] == True

In[9]:= or[equal[image[inv[gp[y_]], range[gp[x_]]], range[gp[x_]]],
          not[subclass[gp[x_], gp[y_]]]] := True
```

The key step of the proof is the following result, which is here derived painlessly using **FastReifNormality**. It amounts to an element-free consequence of the identity **(u^ · v)^ = v^ · u** used in the more traditional proof.

Theorem. A simplification rule.

```
In[10]:= composite[gp[x], id[cart[V, y]], inverse[FIRST]] // inverse //
           FastReifNormality // Reverse

Out[10]= composite[gp[x], id[cart[V, image[inv[gp[x]], y]]], inverse[FIRST]] ==
           composite[FIRST, id[cart[V, y]], inverse[gp[x]]]
```

```
In[11]:= composite[gp[x_], id[cart[V, image[inv[gp[x_]], y_]]], inverse[FIRST]] :=
          composite[FIRST, id[cart[V, y]], inverse[gp[x]]]
```

Theorem. A simplification rule.

```
In[12]:= SubstTest[image, inverse[flip[rotate[t]]], y, t → gp[x]] // Reverse

Out[12]= composite[image[inverse[gp[x]], y], inv[gp[x]]] ==
          composite[gp[x], id[cart[V, y]], inverse[FIRST]]

In[13]:= composite[image[inverse[gp[x_]], y_], inv[gp[x_]]] :=
          composite[gp[x], id[cart[V, y]], inverse[FIRST]]
```

Lemma.

```
In[14]:= SubstTest[implies, equal[u, v],
           equal[image[t, u], image[t, v]], {t -> inverse[flip[rotate[gp[y]]]],
            u -> image[inv[gp[y]], range[gp[x]]], v -> range[gp[x]]}] // Reverse

Out[14]= or[equal[composite[FIRST, id[cart[V, range[gp[x]]]], inverse[gp[y]]],
            composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]]],
           not[equal[image[inv[gp[y]], range[gp[x]]], range[gp[x]]]]] == True

In[15]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If **gp[x] ⊂ gp[y]**, then **gp[y] ∘ id[V × range[gp[x]]] ∘ inverse[FIRST]** is symmetric.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], not[implies[p1, p3]], {p1 → subclass[gp[x], gp[y]],
             p2 -> equal[image[inv[gp[y]], range[gp[x]]], range[gp[x]]],
             p3 -> equal[composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]],
               composite[FIRST, id[cart[V, range[gp[x]]]], inverse[gp[y]]]]}]] // Reverse

Out[16]= or[equal[composite[FIRST, id[cart[V, range[gp[x]]]], inverse[gp[y]]],
            composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]]],
           not[subclass[gp[x], gp[y]]]] == True

In[17]:= or[equal[composite[FIRST, id[cart[V, range[gp[x_]]]], inverse[gp[y_]]],
            composite[gp[y_], id[cart[V, range[gp[x_]]]], inverse[FIRST]]],
           not[subclass[gp[x_], gp[y_]]]] := True
```

Corollary. If **gp[x] ⊂ gp[y]**, then **gp[y] ∘ id[V × range[gp[x]]] ∘ inverse[FIRST]** is an equivalence relation.

```
In[18]:= SubstTest[not, not[implies[and[p0, p1], and[p2, p3]]],
           {p0 → equal[t, composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]]],
            p1 → subclass[gp[x], gp[y]], p2 → SYMMETRIC[t], p3 → TRANSITIVE[t]}] /.
          t → composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]]

Out[18]= or[EQUIVALENCE[composite[gp[y], id[cart[V, range[gp[x]]]], inverse[FIRST]]],
           not[subclass[gp[x], gp[y]]]] == True

In[19]:= or[EQUIVALENCE[composite[gp[y_], id[cart[V, range[gp[x_]]]], inverse[FIRST]]],
           not[subclass[gp[x_], gp[y_]]]] := True
```

Corollary. A dual result.

```
In[20]:= SubstTest[implies, subclass[gp[u], gp[v]],
           EQUIVALENCE[composite[gp[v], id[cart[V, range[gp[u]]]], inverse[FIRST]]],
           {u → flip[gp[x]], v → flip[gp[y]]}] // Reverse

Out[20]= or[EQUIVALENCE[composite[gp[y], id[cart[range[gp[x]], V]], inverse[SECOND]]],
           not[subclass[gp[x], gp[y]]]] == True

In[21]:= or[EQUIVALENCE[composite[gp[y_], id[cart[range[gp[x_]], V]], inverse[SECOND]]],
           not[subclass[gp[x_], gp[y_]]]] := True
```

Theorem. If **x** is a subgroup of a group **y**, then **y ∘ id[V × range[x]] ∘ inverse[FIRST]** is an equivalence relation.

```
In[22]:= SubstTest[implies, and[equal[x, gp[u]], equal[y, gp[v]]],
            or[EQUIVALENCE[composite[y, id[cart[V, range[x]]], inverse[FIRST]]],
             not[subclass[x, y]]], {u → x, v → y}] // Reverse // MapNotNot

Out[22]= or[EQUIVALENCE[composite[y, id[cart[V, range[x]]], inverse[FIRST]]],
           not[member[x, GROUPS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[23]:= or[EQUIVALENCE[composite[y_, id[cart[V, range[x_]]], inverse[FIRST]]],
           not[member[x_, GROUPS]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Dual Theorem. If **x** is a subgroup of a group **y**, then **y ∘ id[range[x] × V] ∘ inverse[SECOND]** is an equivalence relation.

```
In[24]:= SubstTest[implies, and[equal[x, gp[u]], equal[y, gp[v]]],
            or[EQUIVALENCE[composite[y, id[cart[range[x], V]], inverse[SECOND]]],
             not[subclass[x, y]]], {u → x, v → y}] // Reverse // MapNotNot

Out[24]= or[EQUIVALENCE[composite[y, id[cart[range[x], V]], inverse[SECOND]]],
           not[member[x, GROUPS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[25]:= or[EQUIVALENCE[composite[y_, id[cart[range[x_], V]], inverse[SECOND]]],
           not[member[x_, GROUPS]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```