idempotents in groups

Johan G. F. Belinfante
2009 February 15

```
In[1]:= SetDirectory["l:"]; << goedel.09feb14a; << tools.m

        :Package Title: goedel.09feb14a         2009 February 14 at 7:40 p.m.

        It is now:  2009 Feb 15 at 5:7

        Loading Simplification Rules

        TOOLS.M                                      Revised 2009 February 9

        weightlimit = 40
```

summary

The only idempotent element in a group is the identity element. If one group is contained in another, then they share the same identity element.

comment about gp wrappers

The group wrapper **gp[x]** represents a group unless **gp[x] = 0**. This case has to be excluded explicitly in some (but not all) rewrite rules. Such explicit literals generally can be eliminated by removing the **gp** wrappers. The use of **MapNotNot** helps eliminate redundant literals of the form **equal[0, x]** when **gp** wrappers are renoved.

a simplification rule

Theorem. Automatically replace **pair** with **PAIR** in applications of **gp[x]**.

```
In[2]:= SubstTest[APPLY, composite[t, id[cart[V, V]]], pair[u, v], t → gp[x]] // Reverse
Out[2]= APPLY[gp[x], pair[u, v]] == APPLY[gp[x], PAIR[u, v]]

In[3]:= APPLY[gp[x_], pair[u_, v_]] := APPLY[gp[x], PAIR[u, v]]
```

Comment. There is a similar rewrite rule for the **cat** wrapper.

idempotents

Theorem. The identity element **e[gp[x]]** is idempotent if **gp[x]** is a group.

```
In[4]:= SubstTest[member, pair[u, APPLY[funpart[t], u]],
          funpart[t], {u → pair[e[gp[x]], e[gp[x]]], t → gp[x]}] // Reverse

Out[4]= member[pair[pair[e[gp[x]], e[gp[x]]], e[gp[x]]], gp[x]] == not[equal[0, gp[x]]]

In[5]:= member[pair[pair[e[gp[x_]], e[gp[x_]]], e[gp[x_]]], gp[x_]] := not[equal[0, gp[x]]]
```

Lemma. There are no idempotent elements **u ∈ range[gp[x]]** other than the identity element **e[gp[x]]**.

```
In[6]:= SubstTest[implies, equal[u, v],
          equal[APPLY[gp[x], PAIR[u, w]], APPLY[gp[x], PAIR[v, w]]],
          {v → APPLY[gp[x], PAIR[u, u]], w → APPLY[inv[gp[x]], u]}] // Reverse

Out[6]= or[equal[u, e[gp[x]]],
          not[equal[u, APPLY[gp[x], PAIR[u, u]]]], not[member[u, range[gp[x]]]]] == True

In[7]:= or[equal[u_, e[gp[x_]]], not[equal[u_, APPLY[gp[x_], PAIR[u_, u_]]]],
          not[member[u_, range[gp[x_]]]]] := True
```

Corollary. (Obtained by eliminating the variable **u**.)

```
In[8]:= Map[equal[#, V] &, SubstTest[class, u,
          or[equal[u, v], not[equal[u, APPLY[funpart[w], PAIR[u, u]]]], not[member[u, z]]],
          {v → e[gp[x]], w → gp[x], z → range[gp[x]]}]]

Out[8]= subclass[fix[composite[gp[x], DUP]], set[e[gp[x]]]] == True

In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The identity element is the only idempotent element of a group. (Note that if **gp[x] = 0**, this reduces to **0 = 0**. So the rewrite rule remains valid whether or not **gp[x]** is a group.)

```
In[10]:= equal[fix[composite[gp[x], DUP]], set[e[gp[x]]]] // AssertTest

Out[10]= equal[fix[composite[gp[x], DUP]], set[e[gp[x]]]] == True

In[11]:= fix[composite[gp[x_], DUP]] := set[e[gp[x]]]
```

Observation. This will be needed in the next section.

```
In[12]:= abstract[x, fix[composite[x, DUP]]]

Out[12]= composite[SECOND, id[inverse[DUP]]]
```

subgroups

Theorem. If one group is a subset of another, then they share the same identity element. (Comment. The literal **equal[0, gp[x]]** is needed here because the empty set is a subset of **gp[y]**, but is not a group. In that case **e[gp[x]] = e[0] = V**.)

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → composite[SECOND, id[inverse[DUP]]], u → gp[x], v → gp[y]}] // Reverse

Out[13]= or[equal[0, gp[x]], equal[e[gp[x]], e[gp[y]]], not[subclass[gp[x], gp[y]]]] == True

In[14]:= or[equal[0, gp[x_]], equal[e[gp[x_]], e[gp[y_]]], not[subclass[gp[x_], gp[y_]]]] := True
```

Corollary. When the **gp** wrapper on **x** is removed, the literal **equal[0, x]** can be eliminated using **MapNotNot**.

```
In[15]:= SubstTest[implies, equal[x, gp[t]], or[equal[0, x],
             equal[e[x], e[gp[y]]], not[subclass[x, gp[y]]]], t → x] // Reverse // MapNotNot

Out[15]= or[equal[e[x], e[gp[y]]], not[member[x, GROUPS]], not[subclass[x, gp[y]]]] == True

In[16]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If one group is a subclass of another, they have the same identity element. (Comment. Here again an application of **MapNotNot** eliminates an unneeded literal **equal[0, y]**.)

```
In[17]:= SubstTest[implies, equal[y, gp[t]], or[equal[e[x], e[y]],
             not[member[x, GROUPS]], not[subclass[x, y]]], t → y] // Reverse // MapNotNot

Out[17]= or[equal[e[x], e[y]], not[member[x, GROUPS]],
           not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[18]:= or[equal[e[x_], e[y_]], not[member[x_, GROUPS]],
           not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```