subtractive laws of exponents for natural powers of a group element

Johan G. F. Belinfante
2013 July 19

```
In[1]:= SetDirectory["l:"]; << goedel.13jul17a

        :Package Title: goedel.13jul17a                    2013 July 17 at 6:30 a.m.

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

        It is now:  2013 Jul 19 at 16:15

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2013 Jul 19 at 16:32
```

summary

The list of natural powers of a group element **y ∈ range[gp[x]]** is given by **iterate[gp[x] ∘ LEFT[y], {e[gp[x]]}]**. The list of powers of the inverse of the element **y** is given by the composite **inv[gp[x]] ∘ iterate[gp[x] ∘ LEFT[y], {e[gp[x]]}]**. There are four cases to be considered for products of powers of a group element and powers of its inverse. One of the subtractive laws of exponents says that if **m ≥ n**, then the product of the **m**-th power of **y** and the **n**-th power of the inverse of **y** is equal to the **(m – n)**-th power of **y**. A second case says that the same holds when the order of the two factors is reversed. The remaining two cases, with **m ≤ n** are obtained by replacing the element **y** with its inverse. In this notebook it is shown how the rewrite rules for these four cases can be reduced to two rewrite rules by combining cases with **m ≥ n** and **m ≤ n**.

derivation

In the subtractive laws of exponents, the hypothesis **y ∈ range[gp[x]]** can not be left out. The only case currently available can be restated as follows.

```
In[2]:= implies[and[member[y, range[gp[x]]],
          equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]], equal[
          composite[t, rotate[NATADD]], composite[gp[x], cross[t, composite[inv[gp[x]], t]],
           id[composite[id[omega], inverse[S], id[omega]]]]]]

Out[2]= True
```

A second case can be derived from this by applying **flip** to the entire equation and replacing the element **y** with its inverse.

Theorem. A second case.

```
In[3]:= (Map[implies[member[t, range[gp[x]]], #] &, SubstTest[equal, flip[u], flip[v],
             {u -> composite[iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]], rotate[NATADD]],
              v -> composite[gp[x], cross[iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]],
                 composite[inv[gp[x]], iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]]],
                id[composite[id[omega], inverse[S], id[omega]]]]}]] //
           Reverse) /. t → APPLY[inv[gp[x]], y]

Out[3]= or[equal[composite[gp[x], cross[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
            composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
           id[composite[id[omega], S, id[omega]]], composite[inv[gp[x]],
           iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]], rotate[NATADD], SWAP]],
          not[member[y, range[gp[x]]]] == True
```

```
In[4]:= or[equal[composite[gp[x_], cross[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
            composite[inv[gp[x_]], iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]],
           id[composite[id[omega], S, id[omega]]], composite[inv[gp[x_]],
           iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]], rotate[NATADD], SWAP]],
          not[member[y_, range[gp[x_]]]] := True
```

Restatement.

```
In[5]:= implies[and[member[y, range[gp[x]]],
          equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]],
         equal[composite[gp[x], cross[t, composite[inv[gp[x]], t]],
          id[composite[id[omega], S, id[omega]]]],
          composite[inv[gp[x]], t, rotate[NATADD], SWAP]]]

Out[5]= True
```

The above two cases can be combined into a single rewrite rule as follows.

Theorem. Combine two cases.

```
In[6]:= (Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p1, p2, p3], p4],
            not[implies[p1, p4]], {p1 → and[member[y, range[gp[x]]],
               equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]],
             p2 → equal[composite[t, rotate[NATADD]], composite[gp[x], cross[t,
                composite[inv[gp[x]], t]], id[composite[id[omega], inverse[S], id[omega]]]]],
             p3 → equal[composite[gp[x], cross[t, composite[inv[gp[x]], t]], id[composite[
                 id[omega], S, id[omega]]]], composite[inv[gp[x]], t, rotate[NATADD], SWAP]],
             p4 → equal[composite[gp[x], cross[t, composite[inv[gp[x]], t]]], union[composite[
                t, rotate[NATADD]], composite[inv[gp[x]], t, rotate[NATADD], SWAP]]]}]] //
          Reverse) /. t -> iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]

Out[6]= or[equal[composite[gp[x], cross[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
            composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]],
          union[composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD]],
           composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
            rotate[NATADD], SWAP]]], not[member[y, range[gp[x]]]]] == True

In[8]:= or[equal[composite[gp[x_], cross[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
            composite[inv[gp[x_]], iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]]]],
          union[composite[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
            rotate[NATADD]], composite[inv[gp[x_]],
            iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], rotate[NATADD], SWAP]]],
         not[member[y_, range[gp[x_]]]]] := True
```

Restatement.

```
In[9]:= implies[and[member[y, range[gp[x]]],
          equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]],
         equal[composite[gp[x], cross[t, composite[inv[gp[x]], t]]],
          union[composite[t, rotate[NATADD]], composite[inv[gp[x]], t, rotate[NATADD], SWAP]]]]

Out[9]= True
```

A similar combination rule can be derived by replacing the group element with its inverse.

Corollary. (Replace **y** with its inverse.)

```
In[10]:= SubstTest[or,
           equal[composite[gp[x], cross[iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]],
              composite[inv[gp[x]], iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]]]]],
            union[composite[iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]], rotate[NATADD]],
             composite[inv[gp[x]], iterate[composite[gp[x], LEFT[t]], set[e[gp[x]]]],
              rotate[NATADD], SWAP]]],
           not[member[t, range[gp[x]]]], t → APPLY[inv[gp[x]], y]] // Reverse

Out[10]= or[equal[composite[gp[x],
            cross[composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
             iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]],
           union[composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
            rotate[NATADD]], composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
            rotate[NATADD], SWAP]]], not[member[y, range[gp[x]]]]] == True
```

```
In[12]:= or[equal[composite[gp[x_], cross[
              composite[inv[gp[x_]], iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]],
              iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]],
            union[composite[inv[gp[x_]], iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
              rotate[NATADD]], composite[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
              rotate[NATADD], SWAP]]], not[member[y_, range[gp[x_]]]] := True
```

Restatement.

```
In[13]:= implies[and[member[y, range[gp[x]]],
           equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
          equal[composite[gp[x], cross[composite[inv[gp[x]], t], t]],
           union[composite[inv[gp[x]], t, rotate[NATADD]], composite[t, rotate[NATADD], SWAP]]]]

Out[13]= True
```