subtractive law of exponents for powers of a group element

Johan G. F. Belinfante
2013 May 27

```
In[1]:= SetDirectory["l:"]; << goedel.13may25a

        :Package Title: goedel.13may25a                  2013 May 25 at 8:45 p.m.

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

        It is now:  2013 May 27 at 11:26

        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 May 27 at 11:42
```

summary

A subtractive law of exponents for powers of a group element **y ∈ range[gp[x]]** is derived. The list **iterate[gp[x] ∘ LEFT[y], {e[gp[x]]}]** of powers of **y** is a function with domain ω. The value of the power list at any natural number **n** is the **n**-th power of **y**. The subtractive law of exponents says that if **m ≤ n**, then the **(n − m)**-th power of **y** is the product of the **n**-th power and the inverse of the **m**-th power. Various equivalent formulations of this law are also derived.

derivation

The main part of the derivation is done without introducing natural number variables.

Lemma. Simplification rule.

```
In[2]:= equal[composite[id[range[gp[x]]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
          iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]] // assert

Out[2]= True

In[3]:= composite[id[range[gp[x_]]], iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]] :=
         iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]
```

Lemma. Temporary simplification rule.

```
In[4]:= (composite[gp[x],
           intersection[composite[inverse[FIRST], gp[x], cross[funpart[s], funpart[t]]],
            composite[inverse[SECOND], inv[gp[x]], funpart[t], SECOND]],
           id[SECOND], inverse[FIRST]] // FastReifTriNormality) /.
         {s → composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD]],
          t -> iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]}

Out[4]= composite[gp[x], intersection[composite[inverse[FIRST], gp[x],
           cross[composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD]],
            iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]], composite[inverse[SECOND],
           inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]], SECOND]],
          id[SECOND], inverse[FIRST]] == composite[
          iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]], rotate[NATADD], id[
           composite[FIRST, id[cart[domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
              domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]], inverse[NATADD]]]]

In[5]:= composite[gp[x_], intersection[composite[inverse[FIRST], gp[x_],
           cross[composite[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
             rotate[NATADD]], iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]]],
           composite[inverse[SECOND], inv[gp[x_]], iterate[composite[gp[x_], LEFT[y_]],
             set[e[gp[x_]]]], SECOND]], id[SECOND], inverse[FIRST]] :=
         composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD], id[
           composite[FIRST, id[cart[domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
             domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]], inverse[NATADD]]]]
```

Lemma. Simplification rule to eliminate **ROT**.

```
In[6]:= Map[inverse,
          composite[intersection[composite[inverse[FIRST], x, ROT], composite[inverse[SECOND],
              y, SECOND]], id[SECOND], inverse[FIRST]] // inverse // FastReifTriNormality]

Out[6]= composite[intersection[composite[inverse[FIRST], x, ROT],
           composite[inverse[SECOND], y, SECOND]], id[SECOND], inverse[FIRST]] ==
         composite[intersection[composite[inverse[SECOND], y, FIRST],
           composite[inverse[FIRST], x, id[FIRST], inverse[FIRST]]], SWAP]

In[7]:= composite[intersection[composite[inverse[FIRST], x_, ROT],
           composite[inverse[SECOND], y_, SECOND]], id[SECOND], inverse[FIRST]] :=
         composite[intersection[composite[inverse[SECOND], y, FIRST],
           composite[inverse[FIRST], x, id[FIRST], inverse[FIRST]]], SWAP]
```

Lemma. A temporary result.

```
In[8]:= Map[composite[gp[x], intersection[composite[inverse[FIRST], #],
            composite[inverse[SECOND], inv[gp[x]], iterate[composite[gp[x], LEFT[y]],
              set[e[gp[x]]]], SECOND]], id[SECOND], inverse[FIRST]] &,
          Assoc[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], NATADD,
           cross[rotate[NATADD], Id]]] // Reverse

Out[8]= composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD], id[
          composite[FIRST, id[cart[domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
            domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]]], inverse[NATADD]]]] ==
        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], inverse[S], id[omega]]]]

In[9]:= composite[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]],
          rotate[NATADD], id[composite[FIRST,
            id[cart[domain[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]], domain[
              iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]]]], inverse[NATADD]]]] :=
        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], inverse[S], id[omega]]]]
```

Main Theorem. Subtractive law of exponents.

```
In[10]:= SubstTest[implies, equal[w, domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
           equal[composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
             rotate[NATADD], id[composite[FIRST, id[cart[w, w]], inverse[NATADD]]]],
            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], inverse[S], id[omega]]]]], w → omega] // Reverse

Out[10]= or[equal[composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD]],
            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], inverse[S], id[omega]]]]],
           not[member[y, range[gp[x]]]]] == True

In[11]:= or[equal[
            composite[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], rotate[NATADD]],
            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], inverse[S], id[omega]]]]],
           not[member[y_, range[gp[x_]]]]] := True
```

eliminating iterate

The explicit occurrence of **iterate** in the subtractive law of exponents can be eliminated by using the fact that the power list is characterized by its property of being a binary homomorphism.

Lemma.

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[t, binhom[NATADD, gp[x]]], p2 → member[t, map[omega, range[gp[x]]]],
            p3 → member[APPLY[t, set[0]], range[gp[x]]]}]] // Reverse

Out[12]= or[member[APPLY[t, set[0]], range[gp[x]]], not[member[t, binhom[NATADD, gp[x]]]]] == True

In[13]:= or[member[APPLY[t_, set[0]], range[gp[x_]]],
           not[member[t_, binhom[NATADD, gp[x_]]]]] := True
```

Theorem.

```
In[14]:= Map[not, SubstTest[and, implies[and[p0, p1], p2], implies[and[p0, p1], p3],
           implies[and[p2, p3], p4], not[implies[and[p0, p1], p4]],
           {p0 → equal[y, APPLY[t, set[0]]], p1 → member[t, binhom[NATADD, gp[x]]],
            p2 → equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
            p3 → member[y, range[gp[x]]], p4 -> equal[composite[t, rotate[NATADD]],
              composite[gp[x], cross[t, composite[inv[gp[x]], t]], id[composite[
                 id[omega], inverse[S], id[omega]]]]]}] /. y → APPLY[t, set[0]]] // Reverse

Out[14]= or[equal[composite[t, rotate[NATADD]],
           composite[gp[x], cross[t, composite[inv[gp[x]], t]],
            id[composite[id[omega], inverse[S], id[omega]]]]],
          not[member[t, binhom[NATADD, gp[x]]]]] == True

In[15]:= or[equal[composite[t_, rotate[NATADD]],
           composite[gp[x_], cross[t_, composite[inv[gp[x_]], t_]],
            id[composite[id[omega], inverse[S], id[omega]]]]],
          not[member[t_, binhom[NATADD, gp[x_]]]]] := True
```

Corollary. Eliminating the **gp** wrapper.

```
In[16]:= SubstTest[implies, equal[x, gp[w]],
           or[equal[composite[t, rotate[NATADD]], composite[x, cross[t, composite[inv[x], t]],
              id[composite[id[omega], inverse[S], id[omega]]]]],
            not[member[t, binhom[NATADD, x]]]]], w → x] // Reverse // MapNotNot

Out[16]= or[equal[composite[t, rotate[NATADD]], composite[x,
            cross[t, composite[inv[x], t]], id[composite[id[omega], inverse[S], id[omega]]]]],
          not[member[t, binhom[NATADD, x]]], not[member[x, GROUPS]]] == True

In[17]:= or[equal[composite[t_, rotate[NATADD]],
           composite[x_, cross[t_, composite[inv[x_], t_]],
            id[composite[id[omega], inverse[S], id[omega]]]]],
          not[member[t_, binhom[NATADD, x_]]], not[member[x_, GROUPS]]] := True
```

Lemma. The group **x** can be replaced with its **flip**.

```
In[18]:= SubstTest[or, equal[composite[t, rotate[NATADD]],
            composite[gp[w], cross[t, composite[inv[gp[w]], t]],
             id[composite[id[omega], inverse[S], id[omega]]]]],
           not[member[t, binhom[NATADD, gp[w]]]], w → flip[gp[x]]] // Reverse

Out[18]= or[equal[composite[t, rotate[NATADD]],
            composite[gp[x], SWAP, cross[t, composite[inv[gp[x]], t]],
             id[composite[id[omega], inverse[S], id[omega]]]]],
           not[member[t, binhom[NATADD, gp[x]]]]] == True

In[19]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Theorem. An alternate elimination of the **gp** wrapper, which eliminates **inv[x]** by introducing **rotate[x]**.

```
In[20]:= SubstTest[implies, equal[x, gp[w]],
            or[equal[composite[t, rotate[NATADD]], composite[rotate[x],
               cross[t, t], id[composite[id[omega], inverse[S], id[omega]]]]],
             not[member[t, binhom[NATADD, x]]]], w → x] // Reverse // MapNotNot

Out[20]= or[equal[composite[t, rotate[NATADD]],
            composite[rotate[x], cross[t, t], id[composite[id[omega], inverse[S], id[omega]]]]],
           not[member[t, binhom[NATADD, x]]], not[member[x, GROUPS]]] == True

In[21]:= or[equal[composite[t_, rotate[NATADD]], composite[rotate[x_],
             cross[t_, t_], id[composite[id[omega], inverse[S], id[omega]]]]],
           not[member[t_, binhom[NATADD, x_]]], not[member[x_, GROUPS]]] := True
```

introducing natural number variables

One can introduce natural number variables to obtain the following more transparent statement of the subtractive law of exponents.

Corollary.

```
In[22]:= SubstTest[implies, implies[p, equal[s, t]], implies[p,
             equal[APPLY[s, PAIR[u, v]], APPLY[t, PAIR[u, v]]]], {p → member[y, range[gp[x]]],
             s -> composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], rotate[NATADD]],
             t -> 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], inverse[S], id[omega]]]]}] // Reverse // MapNotNot

Out[22]= or[equal[APPLY[gp[x], PAIR[APPLY[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], u],
              APPLY[inv[gp[x]], APPLY[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], v]]]],
            APPLY[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], natsub[u, v]]],
           member[u, v], not[member[y, range[gp[x]]]]] == True
```

```
In[23]:= or[equal[APPLY[gp[x_],
             PAIR[APPLY[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], u_], APPLY[
               inv[gp[x_]], APPLY[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], v_]]]],
            APPLY[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], natsub[u_, v_]]],
          member[u_, v_], not[member[y_, range[gp[x_]]]]] := True
```

Corollary. Eliminating the **gp** wrapper.

```
In[24]:= SubstTest[implies, equal[x, gp[t]],
           or[equal[APPLY[x, PAIR[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], u],
               APPLY[inv[x], APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], v]]]],
             APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], natsub[u, v]]],
            member[u, v], not[member[y, range[x]]]], t → x] // Reverse // MapNotNot

Out[24]= or[equal[APPLY[x, PAIR[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], u],
             APPLY[inv[x], APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], v]]]],
           APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], natsub[u, v]]],
          member[u, v], not[member[x, GROUPS]], not[member[y, range[x]]]] == True
```

```
In[25]:= or[equal[APPLY[iterate[composite[x_, LEFT[y_]], set[e[x_]]], natsub[u_, v_]],
           APPLY[x_, PAIR[APPLY[iterate[composite[x_, LEFT[y_]], set[e[x_]]], u_],
             APPLY[inv[x_], APPLY[iterate[composite[x_, LEFT[y_]], set[e[x_]]], v_]]]],
          member[u_, v_], not[member[x_, GROUPS]], not[member[y_, range[x_]]]] := True
```

The **iterate** construction can be eliminated here, too.

Theorem.

```
In[26]:= Map[not, SubstTest[and, implies[and[p0, p1], p2], implies[and[p0, p1], p3],
            implies[and[p2, p3], p4], not[implies[and[p0, p1], p4]],
            {p0 → equal[y, APPLY[t, set[0]]], p1 → member[t, binhom[NATADD, gp[x]]],
             p2 → equal[t, iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]],
             p3 → member[y, range[gp[x]]],
             p4 → or[equal[APPLY[gp[x], PAIR[APPLY[t, u], APPLY[inv[gp[x]], APPLY[t, v]]]],
                APPLY[t, natsub[u, v]]], member[u, v]]}] /. y → APPLY[t, set[0]]] // Reverse

Out[26]= or[equal[APPLY[t, natsub[u, v]],
           APPLY[gp[x], PAIR[APPLY[t, u], APPLY[inv[gp[x]], APPLY[t, v]]]],
          member[u, v], not[member[t, binhom[NATADD, gp[x]]]]] == True
```

```
In[27]:= or[equal[APPLY[gp[x_], PAIR[APPLY[t_, u_], APPLY[inv[gp[x_]], APPLY[t_, v_]]]],
           APPLY[t_, natsub[u_, v_]]], member[u_, v_],
          not[member[t_, binhom[NATADD, gp[x_]]]]] := True
```

Corollary. (Eliminating the **gp** wrapper.)

```
In[28]:= SubstTest[implies, equal[x, gp[w]], or[equal[APPLY[t, natsub[u, v]],
             APPLY[x, PAIR[APPLY[t, u], APPLY[inv[x], APPLY[t, v]]]]], member[u, v],
           not[member[t, binhom[NATADD, x]]]], w → x] // Reverse // MapNotNot

Out[28]= or[equal[APPLY[t, natsub[u, v]],
           APPLY[x, PAIR[APPLY[t, u], APPLY[inv[x], APPLY[t, v]]]]], member[u, v],
         not[member[t, binhom[NATADD, x]]], not[member[x, GROUPS]]] == True

In[29]:= or[equal[APPLY[t_, natsub[u_, v_]],
           APPLY[x_, PAIR[APPLY[t_, u_], APPLY[inv[x_], APPLY[t_, v_]]]]], member[u_, v_],
         not[member[t_, binhom[NATADD, x_]]], not[member[x_, GROUPS]]] := True
```