law of exponents for integer powers of a group element

Johan G. F. Belinfante
2013 July 20

```
In[1]:= SetDirectory["l:"]; << goedel.13jul19a

        :Package Title: goedel.13jul19a                    2013 July 19 at 5:00 p.m.

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

        It is now:  2013 Jul 20 at 11:50

        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 20 at 12:7
```

summary

A general law of exponents is derived in this second of a series of notebooks about integer powers of group elements. It is shown that if **y ∈ range[gp[x]]**, then the integer power function **w = intpow[gp[x], y]** satisfies the equation **w ∘ INTADD = gp[x] ∘ (w ⊗ w)**. Since this is an equation involving functions, it suffices to show that the right side of the equation is a subset of the left side of the equation. As a matter of fact, this inclusion also holds when **y** is not an element of **range[gp[x]]**.

domain of intpow[gp[x], y]

The following equation holds for the domain of the list of natural powers of a group element.

```
In[2]:= equal[omega, domain[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]]]]

Out[2]= member[y, range[gp[x]]]
```

A similar result will be derived for the integer power function.

Lemma. An implication in one direction.

```
In[4]:= SubstTest[implies, member[t, map[u, v]], equal[domain[t], u],
          {t -> intpow[gp[x], y], u → Z, v → range[gp[x]]}] // Reverse

Out[4]= or[equal[Z, domain[intpow[gp[x], y]]], not[member[y, range[gp[x]]]]] == True
```

```
In[5]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= Map[not, SubstTest[implies, member[v, image[u, x]],
           member[v, range[u]], {u → PLUS, v → inverse[plus[set[0]]]}]] // Reverse

Out[6]= member[composite[inverse[SUCC], id[omega]], image[PLUS, x]] == False

In[7]:= member[composite[inverse[SUCC], id[omega]], image[PLUS, x_]] := False
```

Lemma. The converse implication.

```
In[8]:= Map[implies[equal[Z, domain[intpow[gp[x], y]]], equal[omega, #]] &,
         IminComp[intpow[gp[x], y], PLUS, V]]

Out[8]= or[member[y, range[gp[x]]], not[equal[Z, domain[intpow[gp[x], y]]]]] == True

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The domain of the integer power function is the set **Z** of all integers if and only if **y ∈ range[gp[x]]**.

```
In[10]:= equiv[equal[Z, domain[intpow[gp[x], y]]], member[y, range[gp[x]]]]

Out[10]= True

In[11]:= equal[Z, domain[intpow[gp[x_], y_]]] := member[y, range[gp[x]]]
```

degenerate integer powers

In this section an inclusion is derived for the integer power function **intpow[gp[x], y]** for the less interesting case that **y** is not a member of **range[gp[x]]**. Only the zero-th power of **y** is defined in this case and the integer power function consists of the single point **PAIR[id[ω], e[gp[x]]]**.

Theorem.

```
In[12]:= SubstTest[implies, equal[z, composite[gp[x], LEFT[y]]], equal[intpow[gp[x], y],
           union[composite[iterate[z, set[e[gp[x]]]], inverse[PLUS]], composite[inv[gp[x]],
             iterate[z, set[e[gp[x]]]], inverse[PLUS], INVERSE]]], z → 0] // Reverse

Out[12]= or[equal[cart[set[id[omega]], set[e[gp[x]]]], intpow[gp[x], y]],
           member[y, range[gp[x]]]] == True

In[13]:= or[equal[cart[set[id[omega]], set[e[gp[x_]]]], intpow[gp[x_], y_]],
           member[y_, range[gp[x_]]]] := True
```

Theorem. An inclusion for the law of exponents when **y** is not an element of the group **gp[x]**.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → not[member[y, range[gp[x]]]],
            p2 -> equal[cart[set[id[omega]], set[e[gp[x]]]], intpow[gp[x], y]],
            p3 → subclass[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
              composite[intpow[gp[x], y], INTADD]]}]] // Reverse

Out[14]= or[member[y, range[gp[x]]],
           subclass[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
            composite[intpow[gp[x], y], INTADD]]] == True

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

additive laws of exponents

When **y ∈ range[gp[x]]**, the law of exponents for integer powers must be derived from the additive and subtractive laws of exponents for the list of natural powers. In this section the additive laws are considered.

Lemma.

```
In[16]:= Map[subclass[composite[#, inverse[cross[PLUS, PLUS]]],
           composite[intpow[gp[x], y], INTADD]] &,
         Assoc[intpow[gp[x], y], PLUS, NATADD]] // Reverse

Out[16]= subclass[composite[gp[x],
           cross[composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], inverse[PLUS]],
            composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], inverse[PLUS]]]],
         composite[intpow[gp[x], y], INTADD]] == True

In[17]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. An inclusion for the case of integer powers involving positive exponents.

```
In[18]:= Map[subclass[composite[gp[x], cross[#, #]], composite[intpow[gp[x], y], INTADD]] &,
          Assoc[intpow[gp[x], y], PLUS, inverse[PLUS]]]

Out[18]= subclass[composite[gp[x], cross[composite[intpow[gp[x], y], id[range[PLUS]]],
           composite[intpow[gp[x], y], id[range[PLUS]]]]],
         composite[intpow[gp[x], y], INTADD]] == True

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

The case that both factors involve negative exponents is completely analogous.

Lemma.

```
In[20]:= Map[subclass[composite[#, inverse[cross[PLUS, PLUS]], cross[INVERSE, INVERSE]],
           composite[intpow[gp[x], y], INTADD]] &,
         Map[flip, Assoc[intpow[gp[x], y], composite[INVERSE, PLUS], NATADD]]] // Reverse

Out[20]= subclass[composite[gp[x],
          cross[composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
           inverse[PLUS], INVERSE], composite[inv[gp[x]],
           iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], inverse[PLUS], INVERSE]]],
         composite[intpow[gp[x], y], INTADD]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. An inclusion for the case of integer powers involving negative exponents.

```
In[22]:= Map[subclass[composite[gp[x], cross[#, #]], composite[intpow[gp[x], y], INTADD]] &,
          Assoc[intpow[gp[x], y], composite[INVERSE, PLUS], inverse[composite[INVERSE, PLUS]]]]

Out[22]= subclass[
          composite[gp[x], cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
           composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]]],
         composite[intpow[gp[x], y], INTADD]] == True

In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

subtractive laws of exponents

For the case that one of the factors has a positive exponent, and the other has a negative exponent, the subtractive laws of exponents for natural powers must be used. The following lemma helps to make the transition from natural powers to integer powers.

Lemma. A temporary rewrite rule.

```
In[24]:= Assoc[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
          cross[composite[INVERSE, PLUS], PLUS],
          inverse[cross[composite[INVERSE, PLUS], PLUS]]] // Reverse

Out[24]= composite[gp[x], cross[composite[inv[gp[x]],
           iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], inverse[PLUS], INVERSE],
          composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]], inverse[PLUS]]]] ==
         composite[gp[x], cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
          composite[intpow[gp[x], y], id[range[PLUS]]]]]

In[25]:= composite[gp[x_], cross[composite[inv[gp[x_]],
           iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], inverse[PLUS], INVERSE],
          composite[iterate[composite[gp[x_], LEFT[y_]], set[e[gp[x_]]]], inverse[PLUS]]]] :=
         composite[gp[x], cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
          composite[intpow[gp[x], y], id[range[PLUS]]]]]
```

Lemma.

```
In[26]:= Map[subclass[composite[inv[gp[x]], #, inverse[cross[composite[INVERSE, PLUS], PLUS]]],
           composite[intpow[gp[x], y], INTADD]] &,
         Assoc[intpow[gp[x], y], PLUS, rotate[NATADD]]] // Reverse

Out[26]= subclass[composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
          rotate[NATADD], cross[composite[inverse[PLUS], INVERSE], inverse[PLUS]]],
         composite[intpow[gp[x], y], INTADD]] == True

In[27]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. Temporary rewrite rule.

```
In[28]:= Assoc[intpow[gp[x], y], INVERSE, INTADD]

Out[28]= composite[intpow[gp[x], y], INTADD, cross[INVERSE, INVERSE]] ==
          composite[inv[gp[x]], intpow[gp[x], y], INTADD]

In[29]:= composite[intpow[gp[x_], y_], INTADD, cross[INVERSE, INVERSE]] :=
          composite[inv[gp[x]], intpow[gp[x], y], INTADD]
```

Corollary.

```
In[30]:= SubstTest[implies, subclass[u, v],
           subclass[composite[t, u, w], composite[t, v, w]], {t → inv[gp[x]],
            u -> composite[inv[gp[x]], iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
              rotate[NATADD], cross[composite[inverse[PLUS], INVERSE], inverse[PLUS]]],
            v -> composite[intpow[gp[x], y], INTADD],
            w → composite[SWAP, cross[INVERSE, INVERSE]]}] // Reverse

Out[30]= subclass[composite[iterate[composite[gp[x], LEFT[y]], set[e[gp[x]]]],
          rotate[NATADD], SWAP, cross[composite[inverse[PLUS], INVERSE], inverse[PLUS]]],
         composite[intpow[gp[x], y], INTADD]] == True

In[31]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. An inclusion.

```
In[32]:= SubstTest[implies,
           and[equal[u, v], subclass[composite[v, w], z]], subclass[composite[u, w], z],
           {u -> 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]]]]]],
            v -> 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]],
            w → cross[composite[inverse[PLUS], INVERSE], inverse[PLUS]],
            z -> composite[intpow[gp[x], y], INTADD]}] // Reverse

Out[32]= or[not[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]]]],
           subclass[composite[gp[x], cross[composite[intpow[gp[x], y],
              id[image[INVERSE, range[PLUS]]]], composite[intpow[gp[x], y], id[range[PLUS]]]]],
            composite[intpow[gp[x], y], INTADD]]] == True

In[33]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. An inclusion for the case that the exponents have opposite signs.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → member[y, range[gp[x]]], p2 -> 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]]], p3 -> subclass[composite[gp[x],
              cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
               composite[intpow[gp[x], y], id[range[PLUS]]]]],
             composite[intpow[gp[x], y], INTADD]]}]] // Reverse

Out[34]= or[not[member[y, range[gp[x]]]], subclass[
           composite[gp[x], cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
             composite[intpow[gp[x], y], id[range[PLUS]]]]],
           composite[intpow[gp[x], y], INTADD]]] == True

In[35]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A similar result will now be derived for the case that the signs are switched.

Lemma.

```
In[36]:= SubstTest[implies, subclass[u, v],
           subclass[composite[t, u, w], composite[t, v, w]], {t → inv[gp[x]], u -> composite[
              gp[x], cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
               composite[intpow[gp[x], y], id[range[PLUS]]]]],
             v -> composite[intpow[gp[x], y], INTADD], w → SWAP}] // Reverse

Out[36]= or[not[subclass[composite[gp[x],
              cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
               composite[intpow[gp[x], y], id[range[PLUS]]]]],
             composite[intpow[gp[x], y], INTADD]]], subclass[
            composite[gp[x], cross[composite[inv[gp[x]], intpow[gp[x], y], id[range[PLUS]]],
              composite[inv[gp[x]], intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]]],
            composite[inv[gp[x]], intpow[gp[x], y], INTADD]]] == True

In[37]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[38]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], not[implies[p1, p3]], {p1 → member[y, range[gp[x]]],
             p2 → subclass[composite[gp[x], cross[composite[intpow[gp[x], y], id[image[INVERSE,
                    range[PLUS]]]], composite[intpow[gp[x], y], id[range[PLUS]]]]],
               composite[intpow[gp[x], y], INTADD]], p3 -> subclass[composite[gp[x],
                cross[composite[inv[gp[x]], intpow[gp[x], y], id[range[PLUS]]],
                 composite[inv[gp[x]], intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]]],
               composite[inv[gp[x]], intpow[gp[x], y], INTADD]]}]] // Reverse

Out[38]= or[not[member[y, range[gp[x]]]], subclass[
            composite[gp[x], cross[composite[inv[gp[x]], intpow[gp[x], y], id[range[PLUS]]],
              composite[inv[gp[x]], intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]]],
            composite[inv[gp[x]], intpow[gp[x], y], INTADD]]] == True

In[39]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

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

```
In[40]:= SubstTest[implies, member[t, range[gp[x]]], subclass[
            composite[gp[x], cross[composite[inv[gp[x]], intpow[gp[x], t], id[range[PLUS]]],
              composite[inv[gp[x]], intpow[gp[x], t], id[image[INVERSE, range[PLUS]]]]]],
            composite[inv[gp[x]], intpow[gp[x], t], INTADD]], t → APPLY[inv[gp[x]], y]] // Reverse

Out[40]= or[not[member[y, range[gp[x]]]],
           subclass[composite[gp[x], cross[composite[intpow[gp[x], y], id[range[PLUS]]],
              composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]]],
            composite[intpow[gp[x], y], INTADD]]] == True

In[41]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

general law of exponents for integer powers

The general law of exponents for integer powers is obtained by combining the four separate cases considered above.

Lemma. An inclusion for the case **y ∈ range[gp[x]]**.

```
In[42]:= Map[implies[member[y, range[gp[x]]],
              subclass[#, composite[intpow[gp[x], y], INTADD]]] &, SubstTest[composite, gp[x],
             union[p, q, r, s], {p -> cross[composite[intpow[gp[x], y], id[range[PLUS]]],
                composite[intpow[gp[x], y], id[range[PLUS]]]],
              q -> cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]],
                composite[intpow[gp[x], y], id[range[PLUS]]]],
              r -> cross[composite[intpow[gp[x], y], id[range[PLUS]]],
                composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]],
              s -> cross[composite[intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]], composite[
                 intpow[gp[x], y], id[image[INVERSE, range[PLUS]]]]}]] // MapNotNot // Reverse

Out[42]= or[not[member[y, range[gp[x]]]],
           subclass[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
            composite[intpow[gp[x], y], INTADD]]] == True

In[43]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

This result can now be combined with the inclusion for the case that **y** is not an element of the group **gp[x]**.

Theorem. General inclusion for the law of exponents.

```
In[44]:= SubstTest[and, implies[p, q], or[p, q], {p -> member[y, range[gp[x]]],
           q -> subclass[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
             composite[intpow[gp[x], y], INTADD]]}]

Out[44]= subclass[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
           composite[intpow[gp[x], y], INTADD]] == True

In[45]:= subclass[composite[gp[x_], cross[intpow[gp[x_], y_], intpow[gp[x_], y_]]],
           composite[intpow[gp[x_], y_], INTADD]] := True
```

Inclusions for functions can always be replaced with equations. In the present case, one obtains the following.

Lemma.

```
In[46]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u -> composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
            v -> composite[intpow[gp[x], y], INTADD]}] // Reverse

Out[46]= equal[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
           composite[intpow[gp[x], y], INTADD,
            id[cart[domain[intpow[gp[x], y]], domain[intpow[gp[x], y]]]]] == True

In[47]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. General law of exponents for integer powers of a group element.

```
In[49]:= SubstTest[implies, equal[z, domain[intpow[gp[x], y]]],
           equal[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
            composite[intpow[gp[x], y], INTADD, id[cart[z, z]]], z → Z] // Reverse

Out[49]= or[equal[composite[gp[x], cross[intpow[gp[x], y], intpow[gp[x], y]]],
            composite[intpow[gp[x], y], INTADD]], not[member[y, range[gp[x]]]]] == True

In[51]:= or[equal[composite[gp[x_], cross[intpow[gp[x_], y_], intpow[gp[x_], y_]]],
            composite[intpow[gp[x_], y_], INTADD]], not[member[y_, range[gp[x_]]]]] := True
```