multiplicative law of exponents for monoid elements

Johan G. F. Belinfante
2012 December 22

```
In[1]:= SetDirectory["l:"]; << goedel.12dec20a

        :Package Title: goedel.12dec20a            2012 December 20 at 11:40 a.m.

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

        It is now:  2012 Dec 22 at 9:21

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2012 Dec 22 at 9:37
```

summary

A multiplicative law of exponents is derived for powers of an element of the range of a monoid.

derivation

Any unital binary homomorphism of monoids is a functor. A restatement of this fact using **monoid** wrappers will now be derived.

Lemma.

```
In[7]:= SubstTest[implies, and[member[t, binhom[u, v]],
           member[u, MONOIDS], member[v, MONOIDS], equal[APPLY[t, e[u]], e[v]]],
         functor[t, u, v], {u → monoid[x], v → monoid[y]}] // Reverse

Out[7]= or[equal[0, monoid[x]], equal[0, monoid[y]],
          functor[t, monoid[x], monoid[y]], not[equal[APPLY[t, e[monoid[x]]], e[monoid[y]]]],
          not[member[t, binhom[monoid[x], monoid[y]]]]] == True

In[8]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Two redundant literals can be removed.

Theorem.

```
In[10]:= SubstTest[and, implies[p, q], or[p, q],
           {p -> or[equal[0, monoid[x]], equal[0, monoid[y]]],
            q → or[functor[t, monoid[x], monoid[y]], not[equal[APPLY[t, e[monoid[x]]],
                e[monoid[y]]]], not[member[t, binhom[monoid[x], monoid[y]]]]]}] // MapNotNot

Out[10]= or[functor[t, monoid[x], monoid[y]], not[equal[APPLY[t, e[monoid[x]]], e[monoid[y]]]],
           not[member[t, binhom[monoid[x], monoid[y]]]]] == True

In[12]:= or[functor[t_, monoid[x_], monoid[y_]],
           not[equal[APPLY[t_, e[monoid[x_]]], e[monoid[y_]]]],
           not[member[t_, binhom[monoid[x_], monoid[y_]]]]] := True
```

For any natural number **nat[y]**, the function **times[nat[y]]** is a functor from **NATADD** to itself. The composite of this with any functor **x** from **NATADD** to **z** is thus another functor from **NATADD** to **z**.

Lemma.

```
In[17]:= SubstTest[implies, and[functor[x, NATADD, z], functor[t, NATADD, NATADD]],
           functor[composite[x, t], NATADD, z], t → times[nat[y]]] // Reverse

Out[17]= or[functor[composite[x, times[nat[y]]], NATADD, z], not[functor[x, NATADD, z]]] == True

In[18]:= or[functor[composite[x_, times[nat[y_]]], NATADD, z_],
           not[functor[x_, NATADD, z_]]] := True
```

The list of powers of an element **y** of the range of a monoid **x** is a functor from **NATADD** to **x**.

```
In[32]:= implies[and[member[x, MONOIDS], member[y, range[x]]],
          functor[iterate[composite[x, LEFT[y]], set[e[x]]], NATADD, x]]

Out[32]= True
```

Theorem. The composite of the list of a powers of a monoid element with **times[nat[z]]** is a functor from **NATADD** to the monoid.

```
In[33]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
             not[implies[p1, p3]], {p1 → and[member[x, MONOIDS], member[y, range[x]]],
              p2 → functor[iterate[composite[x, LEFT[y]], set[e[x]]], NATADD, x],
              p3 → functor[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]],
                NATADD, x]}]] // Reverse

Out[33]= or[functor[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]],
             NATADD, x], not[member[x, MONOIDS]], not[member[y, range[x]]]] == True

In[36]:= or[functor[composite[iterate[composite[x_, LEFT[y_]], set[e[x_]]], times[nat[z_]]],
             NATADD, x_], not[member[x_, MONOIDS]], not[member[y_, range[x_]]]] := True
```

Lemma. (Automatic removal of a redundant literal.)

```
In[39]:= equiv[or[equal[0, x], not[member[y, range[x]]]], not[member[y, range[x]]]]

Out[39]= True
```

```
In[41]:= or[equal[0, x_], not[member[y_, range[x_]]]] := not[member[y, range[x]]]
```

Corollary.

```
In[42]:= SubstTest[or, functor[
           composite[iterate[composite[t, LEFT[y]], set[e[t]]], times[nat[z]]], NATADD, t],
         not[member[t, MONOIDS]], not[member[y, range[t]]], t → monoid[x]] // Reverse
```

```
Out[42]= or[functor[composite[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]],
           times[nat[z]]], NATADD, monoid[x]], not[member[y, range[monoid[x]]]]] == True
```

```
In[44]:= or[functor[composite[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]],
           times[nat[z_]]], NATADD, monoid[x_]], not[member[y_, range[monoid[x_]]]]] := True
```

Every functor from **NATADD** to a monoid **x** is the list of powers of its value at **1 = {0}**.

Lemma.

```
In[47]:= SubstTest[implies, and[functor[t, NATADD, x], member[x, MONOIDS]],
         equal[t, iterate[composite[x, LEFT[APPLY[t, set[0]]]], set[e[x]]]],
         t → composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]]] // Reverse
```

```
Out[47]= or[equal[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]],
           iterate[composite[x,
             LEFT[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], nat[z]]]], set[e[x]]]],
         not[functor[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]],
           NATADD, x]], not[member[x, MONOIDS]]] == True
```

```
In[48]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. A multiplicative law of exponents for the list of powers of a monoid element.

```
In[52]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → and[member[x, MONOIDS], member[y, range[x]]], p2 -> functor[
              composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]], NATADD, x],
            p3 -> equal[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]],
              iterate[composite[x, LEFT[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]],
                  nat[z]]]], set[e[x]]]]}]] // Reverse
```

```
Out[52]= or[equal[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[z]]], iterate[
           composite[x, LEFT[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], nat[z]]]],
           set[e[x]]]], not[member[x, MONOIDS]], not[member[y, range[x]]]] == True
```

```
In[54]:= or[equal[
           composite[iterate[composite[x_, LEFT[y_]], set[e[x_]]], times[nat[z_]]], iterate[
            composite[x_, LEFT[APPLY[iterate[composite[x_, LEFT[y_]], set[e[x_]]], nat[z_]]]],
            set[e[x_]]]], not[member[x_, MONOIDS]], not[member[y_, range[x_]]]] := True
```

Corollary.

```
In[56]:= SubstTest[or,
           equal[composite[iterate[composite[t, LEFT[y]], set[e[t]]], times[nat[z]]], iterate[
             composite[t, LEFT[APPLY[iterate[composite[t, LEFT[y]], set[e[t]]], nat[z]]]],
             set[e[t]]]], not[member[t, MONOIDS]],
           not[member[y, range[t]]], t → monoid[x]] // Reverse

Out[56]= or[equal[composite[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]],
             times[nat[z]]], iterate[composite[monoid[x],
              LEFT[APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]], nat[z]]]],
             set[e[monoid[x]]]]], not[member[y, range[monoid[x]]]]] == True

Out[57]= or[equal[composite[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]],
             times[nat[z_]]], iterate[composite[monoid[x_], LEFT[
               APPLY[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]], nat[z_]]]],
             set[e[monoid[x_]]]]], not[member[y_, range[monoid[x_]]]]] == True

In[58]:= or[equal[composite[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]],
             times[nat[z_]]], iterate[composite[monoid[x_], LEFT[
               APPLY[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]], nat[z_]]]],
             set[e[monoid[x_]]]]], not[member[y_, range[monoid[x_]]]]] := True
```

Theorem. Multiplicative law of powers for monoid elements.

```
In[63]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            not[implies[p1, p3]], {p1 -> and[member[x, MONOIDS], member[y, range[x]]],
             p2 -> equal[composite[iterate[composite[x, LEFT[y]], set[e[x]]], times[nat[u]]],
               iterate[composite[x, LEFT[
                  APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], nat[u]]]], set[e[x]]], p3 ->
              equal[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], natmul[nat[u], nat[v]]],
               APPLY[iterate[composite[x, LEFT[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]],
                    nat[u]]]], set[e[x]]], nat[v]]]}]] // Reverse

Out[63]= or[equal[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], natmul[nat[u], nat[v]]],
            APPLY[iterate[composite[x,
               LEFT[APPLY[iterate[composite[x, LEFT[y]], set[e[x]]], nat[u]]]], set[e[x]]],
             nat[v]]], not[member[x, MONOIDS]], not[member[y, range[x]]]] == True

In[67]:= or[equal[APPLY[iterate[composite[x_,
               LEFT[APPLY[iterate[composite[x_, LEFT[y_]], set[e[x_]]], nat[u_]]]],
             set[e[x_]]], nat[v_]], APPLY[iterate[composite[x_, LEFT[y_]],
             set[e[x_]]], natmul[nat[u_], nat[v_]]]],
           not[member[x_, MONOIDS]], not[member[y_, range[x_]]]] := True
```

Corollary.

```
In[68]:= SubstTest[or,
           equal[APPLY[iterate[composite[t, LEFT[y]], set[e[t]]], natmul[nat[u], nat[v]]],
            APPLY[iterate[composite[t, LEFT[APPLY[
                 iterate[composite[t, LEFT[y]], set[e[t]]], nat[u]]]], set[e[t]]], nat[v]]],
           not[member[t, MONOIDS]], not[member[y, range[t]]], t → monoid[x]] // Reverse

Out[68]= or[equal[APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]],
            natmul[nat[u], nat[v]]], APPLY[iterate[composite[monoid[x],
              LEFT[APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]], nat[u]]]],
             set[e[monoid[x]]]], nat[v]]], not[member[y, range[monoid[x]]]]] == True

In[70]:= or[equal[APPLY[
            iterate[composite[monoid[x_], LEFT[APPLY[iterate[composite[monoid[x_], LEFT[y_]],
                 set[e[monoid[x_]]]], nat[u_]]]], set[e[monoid[x_]]]], nat[v_]],
           APPLY[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]],
            natmul[nat[u_], nat[v_]]]], not[member[y_, range[monoid[x_]]]]] := True
```