functors and powers of monoid elements

Johan G. F. Belinfante
2013 January 28

```
In[1]:= SetDirectory["l:"]; << goedel.13jan26a

        :Package Title: goedel.13jan26a             2013 January 26 at 11:15 a.m.

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

        It is now:  2013 Jan 28 at 8:25

        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 Jan 28 at 8:41
```

summary

The list of powers of an element of the range of a monoid is a functor from **NATADD** to the monoid. The composite of a functor from one monoid to another with the power list of an element of the range of a monoid is therefore a monoid from **NATADD** to the second monoid, and is therefore the list of powers of a transformed element of the range of the second monoid.

an improved rewrite rule

In December 2012 the following rewrite rule was derived.

```
In[2]:= or[functor[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]],
          NATADD, monoid[x]], not[member[y, range[monoid[x]]]]]

Out[2]= True
```

This rule can be replaced with a better one because, as will now be shown, the converse also holds.

Lemma.

```
In[19]:= (Map[implies[equal[omega, domain[iterate[u, v]]], not[#]] &,
            SubstTest[empty, image[w, set[t]], {w → iterate[u, v], t → set[0]}]] // Reverse) /.
          {u -> composite[monoid[x], LEFT[y]], v -> set[e[monoid[x]]]}

Out[19]= or[member[y, range[monoid[x]]], not[equal[omega,
            domain[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]]]]] == True

In[20]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[22]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 -> functor[iterate[
               composite[monoid[x], LEFT[y]], set[e[monoid[x]]]], NATADD, monoid[x]], p2 ->
             equal[omega, domain[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]]],
            p3 -> member[y, range[monoid[x]]]}]] // Reverse

Out[22]= or[member[y, range[monoid[x]]],
           not[functor[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]],
             NATADD, monoid[x]]]] == True

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

Theorem. A replacement rewrite rule.

```
In[24]:= equiv[functor[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]],
           NATADD, monoid[x]], member[y, range[monoid[x]]]]

Out[24]= True

In[26]:= functor[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]],
           NATADD, monoid[x_]] := member[y, range[monoid[x]]]
```

---

main theorem

Lemma.

```
In[30]:= SubstTest[implies, and[functor[t, monoid[x], monoid[y]],
            functor[s, NATADD, monoid[x]]], functor[composite[t, s], NATADD, monoid[y]],
           s → iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]] // Reverse

Out[30]= or[functor[composite[t, iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
            NATADD, monoid[y]], not[functor[t, monoid[x], monoid[y]]],
           not[member[z, range[monoid[x]]]]] == True

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

Lemma.

```
In[49]:= Map[implies[member[z, range[monoid[x]]], equal[z, A[#]]] &,
           SubstTest[image, iterate[u, v], set[set[0]],
            {u -> composite[monoid[x], LEFT[z]], v -> set[e[monoid[x]]]}]] // Reverse

Out[49]= or[equal[z, APPLY[iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]], set[0]]],
           not[member[z, range[monoid[x]]]]] == True

In[50]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[52]:= SubstTest[implies, functor[w, NATADD, monoid[y]], equal[w,
            iterate[composite[monoid[y], LEFT[APPLY[w, set[0]]]], set[e[monoid[y]]]]], w ->
            composite[t, iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]] // Reverse

Out[52]= or[equal[
            composite[t, iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]], iterate[
             composite[monoid[y], LEFT[APPLY[t, APPLY[iterate[composite[monoid[x], LEFT[z]],
                  set[e[monoid[x]]]], set[0]]]]], set[e[monoid[y]]]]],
           not[functor[composite[t, iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
            NATADD, monoid[y]]]] == True

In[53]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main Theorem. The composite of a functor between monoids and a list of powers of an element of the range of a monoid is a list of powers of a transformed element.

```
In[56]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[p3, p4], implies[p2, p5], implies[and[p4, p5], p6],
            not[implies[and[p1, p2], p6]], {p1 -> functor[t, monoid[x], monoid[y]],
             p2 -> member[z, range[monoid[x]]], p3 -> functor[composite[t, iterate[
                 composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]], NATADD, monoid[y]], p4 ->
              equal[composite[t, iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
               iterate[composite[monoid[y], LEFT[APPLY[t, APPLY[iterate[composite[monoid[x],
                     LEFT[z]], set[e[monoid[x]]]], set[0]]]]], set[e[monoid[y]]]]],
             p5 -> equal[z, APPLY[iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]],
                set[0]]], p6 -> equal[composite[t,
                iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]], iterate[
                composite[monoid[y], LEFT[APPLY[t, z]]], set[e[monoid[y]]]]]}]] // Reverse

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

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

Corollary. An example.

```
In[74]:= (SubstTest[or, equal[
              composite[t, iterate[composite[monoid[u], LEFT[int[y]]], set[e[monoid[u]]]]],
              iterate[composite[monoid[v], LEFT[APPLY[t, int[y]]]], set[e[monoid[v]]]],
             not[functor[t, monoid[u], monoid[v]]], not[member[int[y], range[monoid[u]]]],
             {u → INTADD, v → INTADD}] /. t → inttimes[int[x]]) // Reverse
```

```
Out[74]= equal[composite[MIXMUL, LEFT[intmul[int[x], int[y]]]],
           composite[inttimes[int[x]], MIXMUL, LEFT[int[y]]]] == True
```

```
In[76]:= composite[inttimes[int[x_]], MIXMUL, LEFT[int[y_]]] :=
          composite[MIXMUL, LEFT[intmul[int[x], int[y]]]]
```