power sequence of a group element

Johan G. F. Belinfante
2012 November 21

```
In[1]:= SetDirectory["l:"]; << goedel.12nov17a
```

```
:Package Title: goedel.12nov17a              2012 November 17 at 9:40 a.m.

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

It is now:  2012 Nov 21 at 6:4

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 Nov 21 at 6:20
```

summary

For any element **g ∈ range[x]** of a group **x**, the **power sequence {e[x], g, g · g, ⋯ }** is given by **iterate[x ∘ LEFT[g], {e[x]}]**. Any binary homomorphism from **NATADD** to the group **x** is the power sequence of its value at **1 = {0}**.

derivation

Theorem. Binary homomorphisms from **NATADD** to a group **x** are unital.

```
In[2]:= SubstTest[or, equal[APPLY[t, e[w]], e[x]], not[member[t, binhom[w, x]]],
          not[member[w, MONOIDS]], not[member[x, GROUPS]], w → NATADD] // Reverse
```

```
Out[2]= or[equal[APPLY[t, 0], e[x]],
          not[member[t, binhom[NATADD, x]]], not[member[x, GROUPS]]] == True
```

```
In[3]:= or[equal[APPLY[t_, 0], e[x_]],
          not[member[t_, binhom[NATADD, x_]]], not[member[x_, GROUPS]]] := True
```

Lemma.

```
In[4]:= SubstTest[implies, equal[u, v],
          equal[composite[u, w], composite[v, w]], {u -> composite[funpart[t], NATADD],
           v -> composite[x, cross[funpart[t], funpart[t]]], w → LEFT[set[0]]}] // Reverse

Out[4]= or[equal[composite[x, LEFT[APPLY[funpart[t], set[0]]], funpart[t]],
           composite[funpart[t], id[omega], SUCC]], not[equal[
            composite[x, cross[funpart[t], funpart[t]]], composite[funpart[t], NATADD]]]] == True

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

Theorem. (Eliminate the **funpart** wrapper.)

```
In[6]:= SubstTest[implies, equal[t, funpart[w]],
          or[equal[composite[x, LEFT[APPLY[t, set[0]]], t], composite[t, id[omega], SUCC]],
           not[equal[composite[x, cross[t, t]], composite[t, NATADD]]], w → t] // Reverse

Out[6]= or[equal[composite[t, id[omega], SUCC], composite[x, LEFT[APPLY[t, set[0]]], t]],
          not[equal[composite[t, NATADD], composite[x, cross[t, t]]]], not[FUNCTION[t]]] == True

In[7]:= (% /. {x → x_, t → t_}) /. Equal→ SetDelayed
```

Lemma. Iteration rule for binary homomorphisms from **NATADD** to a group.

```
In[8]:= Map[not, SubstTest[and, (*implies[p1,p2],implies[p1,p3],*)
           implies[and[p2, p3], p4], implies[p1, p5], (*implies[and[p2,p4,p5],p6],*)
           not[implies[p1, p6]], {p1 → member[t, binhom[NATADD, x]], p2 → FUNCTION[t],
            p3 -> equal[composite[t, NATADD], composite[x, cross[t, t]]],
            p4 -> equal[composite[t, id[omega], SUCC], composite[x, LEFT[APPLY[t, set[0]]], t]],
            p5 → equal[domain[t], omega],
            p6 -> equal[composite[t, SUCC], composite[x, LEFT[APPLY[t, set[0]]], t]]}]] // Reverse

Out[8]= or[equal[composite[t, SUCC], composite[x, LEFT[APPLY[t, set[0]]], t]],
          not[member[t, binhom[NATADD, x]]]] == True

In[9]:= or[equal[composite[t_, SUCC], composite[x_, LEFT[APPLY[t_, set[0]]], t_]],
          not[member[t_, binhom[NATADD, x_]]]] := True
```

Lemma. The vertical section of a binary homomorphism at **0**.

```
In[10]:= Map[not,
           SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4], implies[and[p3, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 → member[t, binhom[NATADD, x]],
             p2 → member[x, GROUPS], p3 → equal[APPLY[t, 0], e[x]], p4 → FUNCTION[t],
             p5 → equal[image[t, set[0]], set[e[x]]]}]] // Reverse

Out[10]= or[equal[image[t, set[0]], set[e[x]]],
           not[member[t, binhom[NATADD, x]]], not[member[x, GROUPS]]] == True

In[11]:= (% /. {t → t_, x → x_}) /. Equal→ SetDelayed
```

The proof of the main theorem uses the uniqueness theorem for **iterate**. Many steps of the proof can be omitted, as indicated by **(* ⋯ *)**.

Main Theorem. A binary homomorphism from **NATADD** to a group **x** is the power sequence of the element **APPLY[t, {0}] ∈ range[x]**.

```
In[12]:= Map[not, SubstTest[and, (*implies[and[p1,p2],p3],implies[p1,p4],*) implies[p1, p5],
           implies[p1, p6], implies[and[p3, p4], p7], (*implies[and[p5,p6,p7],p8],*)
           not[implies[and[p1, p2], p8]], {p1 → member[t, binhom[NATADD, x]],
            p2 → member[x, GROUPS], p3 → equal[image[t, set[0]], set[e[x]]],
            p4 -> equal[composite[t, SUCC], composite[x, LEFT[APPLY[t, set[0]]], t]],
            p5 → equal[domain[t], omega], p6 → FUNCTION[t], p7 → equal[composite[t, id[omega]],
              iterate[composite[x, LEFT[APPLY[t, set[0]]]], set[e[x]]]], p8 ->
             equal[t, iterate[composite[x, LEFT[APPLY[t, set[0]]]], set[e[x]]]]}] // Reverse

Out[12]= or[equal[t, iterate[composite[x, LEFT[APPLY[t, set[0]]]], set[e[x]]]],
           not[member[t, binhom[NATADD, x]]], not[member[x, GROUPS]]] == True

In[13]:= or[equal[t_, iterate[composite[x_, LEFT[APPLY[t_, set[0]]]], set[e[x_]]]],
           not[member[t_, binhom[NATADD, x_]]], not[member[x_, GROUPS]]] := True
```