powers of commuting elements of a monoid

Johan G. F. Belinfante
2013 June 11

In[1]:= SetDirectory["l:"]; << goedel.13jun09a

:Package Title: goedel.13jun09a 2013 June 9 at 8:05 a.m.

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

It is now: 2013 Jun 11 at 14:4

Loading Simplification Rules

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

weightlimit = 40

Loading completed.

It is now: 2013 Jun 11 at 14:21

summary

If two elements of a monoid commute, then all powers of the one commute with all powers of the other.

derivation

First it will be shown that if two elements of a monoid commute, then each one commutes with all powers of the other.

Lemma. (Application of an intertwine rule for iterate.)

In[2]:= Map[implies[and[member[y, range[monoid[x]]], member[z, range[monoid[x]]]], #] &, SubstTest[implies, commute[u, v], equal[iterate[u, image[v, w]], composite[v, iterate[u, w]]], {u -> composite[monoid[x], LEFT[y]], v -> composite[monoid[x], LEFT[z]], w -> set[e[monoid[x]]]}]] // Reverse

Out[2]= or[equal[composite[monoid[x], LEFT[z]], iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]], iterate[composite[monoid[x], LEFT[y]], intersection[range[monoid[x]], set[z]]]], not[equal[APPLY[monoid[x], PAIR[y, z]], APPLY[monoid[x], PAIR[z, y]]]], not[member[y, range[monoid[x]]]], not[member[z, range[monoid[x]]]]] = True

In[3]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed

The above result can be cleaned up. The first step is to eliminate an unnecessary intersection.
Lemma. (Eliminate an intersection.)

\[\text{Lemma. (Eliminate an intersection.)}\]

\[
\begin{align*}
\text{In[4]} & := \text{SubstTest[implies, equal[t, intersection[range[monoid[x]], set[z]]], } \\
& \quad \text{or[equal[composite[monoid[x]], LEFT[z], iterate[composite[monoid[x]], LEFT[y]], } \\
& \quad \quad \text{set[e[monoid[x]]]]], iterate[composite[monoid[x]], LEFT[y], t]], } \\
& \quad \text{not[equal[APPLY[monoid[x]], PAIR[y, z], APPLY[monoid[x], PAIR[z, y]]]], } \\
& \quad \text{not[member[y, range[monoid[x]]]], } \\
& \quad \text{not[member[z, range[monoid[x]]]], t \rightarrow set[z] // Reverse} \\
\text{Out[4]} & := \text{or[equal[composite[monoid[x]], LEFT[z], } \\
& \quad \text{iterate[composite[monoid[x]], LEFT[y], set[e[monoid[x]]]]], } \\
& \quad \text{iterate[composite[monoid[x]], LEFT[y], set[z]]], } \\
& \quad \text{not[equal[APPLY[monoid[x]], PAIR[y, z], APPLY[monoid[x], PAIR[z, y]]]], } \\
& \quad \text{not[member[y, range[monoid[x]]]], } \\
& \quad \text{not[member[z, range[monoid[x]]]], = True} \\
\end{align*}
\]

\[
\begin{align*}
\text{In[5]} & := (\% / . \{x \rightarrow x_, y \rightarrow y_, z \rightarrow z_\}) / . \text{Equal} \rightarrow \text{SetDelayed} \\
\end{align*}
\]

The next step is to rewrite the expression \(\text{iterate[monoid[x]} \circ \text{LEFT}[y], {z}]\) in terms of the power list \(\text{iterate[monoid[x]} \circ \text{LEFT}[y], {e[monoid[x]]}]\).

Lemma.

\[
\begin{align*}
\text{In[8]} & := \text{Map[not, SubstTest[and, implies[p1, p2], } \\
& \quad \text{implies[p1, p3], implies[and[p2, p3, p4], not[implies[p1, p4]]], } \\
& \quad \text{\{p1 \rightarrow and[member[y, range[monoid[x]]], member[z, range[monoid[x]]]], } \\
& \quad \text{equal[APPLY[monoid[x]], PAIR[y, z], APPLY[monoid[x], PAIR[z, y]]]], } \\
& \quad \text{p2 \rightarrow equal[composite[monoid[x]], LEFT[z], iterate[composite[monoid[x]], LEFT[y]], } \\
& \quad \text{set[e[monoid[x]]]]], iterate[composite[monoid[x]], LEFT[y], set[z]]], } \\
& \quad \text{p3 \rightarrow equal[composite[monoid[x]], RIGHT[z], iterate[composite[monoid[x]], LEFT[y]], } \\
& \quad \text{set[e[monoid[x]]]]], iterate[composite[monoid[x]], LEFT[y], set[z]]], } \\
& \quad \text{p4 \rightarrow equal[composite[monoid[x]], LEFT[z], iterate[composite[monoid[x]], LEFT[y]], } \\
& \quad \text{set[e[monoid[x]]]]], composite[monoid[x]], \text{RIGHT[z], } \\
& \quad \text{iterate[composite[monoid[x]], LEFT[y], set[e[monoid[x]]]]]] // Reverse} \\
\text{Out[8]} & := \text{or[equal[composite[monoid[x]], LEFT[z], } \\
& \quad \text{iterate[composite[monoid[x]], LEFT[y], set[e[monoid[x]]]]], composite[monoid[x]}, \text{RIGHT[z], iterate[composite[monoid[x]], LEFT[y], set[e[monoid[x]]]]]], } \\
& \quad \text{not[equal[APPLY[monoid[x]], PAIR[y, z], APPLY[monoid[x], PAIR[z, y]]]], } \\
& \quad \text{not[member[y, range[monoid[x]]]], } \\
& \quad \text{not[member[z, range[monoid[x]]]], = True} \\
\end{align*}
\]

\[
(\% / . \{x \rightarrow x_, y \rightarrow y_, z \rightarrow z_\}) / . \text{Equal} \rightarrow \text{SetDelayed}
\]

There are two redundant literals here.

Lemma. The membership literal for \(z\) is redundant.
The result of the preceding section can be made more explicit by introducing an additional variable.
Lemma. Introducing a new variable $w$.

\begin{verbatim}
In[22]:= SubstTest[implies, equal[u, v], equal[image[u, set[w]], image[v, set[w]]],
   {u -> composite[monoid[x], LEFT[z]], iterate[composite[monoid[x], LEFT[y]],
      set[e[monoid[x]]]]}, v -> composite[monoid[x], RIGHT[z]],
   iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]]]] // Reverse
\end{verbatim}

\begin{verbatim}
Out[22]= or[equal[APPLY[monoid[x]],
   PAIR[z, APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]], w]],
   APPLY[monoid[x]],
   PAIR[APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]], w, z]],
   not[equal[composite[monoid[x], LEFT[z]],
      iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]], composite[monoid[x],
      RIGHT[z], iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]]]],
   composite[monoid[x],
      iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]],
   composite[monoid[x],
      iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]]]]]] = True
\end{verbatim}

\begin{verbatim}
In[23]:= (% /. \{w -> w_, x -> x_, y -> y_, z -> z\}) /. Equal \to SetDelayed
\end{verbatim}

Theorem.

\begin{verbatim}
In[24]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]]],
   {p1 -> equal[APPLY[monoid[x]], PAIR[y, z]], APPLY[monoid[x], PAIR[z, y]]],
   p2 -> equal[APPLY[monoid[x]], LEFT[z]], iterate[composite[monoid[x], LEFT[y]],
      set[e[monoid[x]]]], composite[monoid[x],
      RIGHT[z], iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]],
   p3 -> equal[APPLY[monoid[x]], PAIR[z, APPLY[iterate[composite[monoid[x], LEFT[y]],
      set[e[monoid[x]]]]], u]], APPLY[monoid[x], PAIR[APPLY[iterate[composite[monoid[x],
      LEFT[y]], set[e[monoid[x]]]], u, z]]]]]] // Reverse
\end{verbatim}

\begin{verbatim}
Out[24]= or[equal[APPLY[monoid[x]],
   PAIR[z, APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]], u]],
   APPLY[monoid[x]],
   PAIR[APPLY[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]], u, z]]],
   not[equal[APPLY[monoid[x], PAIR[y, z]], APPLY[monoid[x], PAIR[z, y]]]]] = True
\end{verbatim}

\begin{verbatim}
In[35]:= or[equal[APPLY[monoid[x_]],
   PAIR[APPLY[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]]], u_],
   z_], APPLY[monoid[x_], PAIR[z_,
   APPLY[iterate[composite[monoid[x_], LEFT[y_]], set[e[monoid[x_]]]]], u_]]],
   not[equal[APPLY[monoid[x_], PAIR[y_, z_]], APPLY[monoid[x_], PAIR[z_, y_]]]]] := True
\end{verbatim}

Corollary. If $y$ commute with $z$, then any power of $y$ commutes with any power of $z$. 
Lemma. A simplification rule.

In[41]:= \text{composite[}\text{cross}[x, y], id[\text{composite[}Id, z]]]\] // \text{TripleRotate}

Out[41]= composite[\text{cross}[x, y], id[\text{composite[}Id, z]]] = composite[\text{cross}[x, y], id[z]]

In[42]:= composite[\text{cross}[x, y], id[\text{composite[}Id, z]]] := composite[\text{cross}[x, y], id[z]]

Lemma.

\underline{eliminating the variables u and v}

The following special rule helps with eliminating the two variables $u$ and $v$ introduced in the preceding section.

In[37]:= \text{Map[}not, \text{SubstTest[}\text{and, implies[}p_1, p_2], implies[p_2, p_3], not[\text{implies[}p_1, p_3]]],

\{p_1 \rightarrow \text{equal[}\text{APPLY[}monoid[x], \text{PAIR}[y, z]], \text{APPLY[}monoid[x], \text{PAIR}[z, y]]],

p_2 \rightarrow \text{equal[}\text{APPLY[}monoid[x],

\text{PAIR}[z, \text{APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u]]],

\text{APPLY[}monoid[x], \text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]],

\text{set[}e[\text{monoid[}x]]], u], z]]], p_3 \rightarrow \text{equal[}\text{APPLY[}monoid[x],

\text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u],

\text{APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[z]], \text{set[}e[\text{monoid[}x]]], v]]],

\text{APPLY[}monoid[x], \text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[z]],

\text{set[}e[\text{monoid[}x]]], v], \text{APPLY[}\text{iterate[}\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u]])]] \] // \text{Reverse}

Out[37]= \text{or[}\text{equal[}\text{APPLY[}monoid[x],

\text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u],

\text{APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[z]], \text{set[}e[\text{monoid[}x]]], v]]], \text{APPLY[}monoid[x], \text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[z]], \text{set[}e[\text{monoid[}x]]], v], \text{APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u]]]]],

\text{not[}\text{equal[}\text{APPLY[}monoid[x], \text{PAIR}[y, z]], \text{APPLY[}monoid[x], \text{PAIR}[z, y]]]]\] = \text{True}

In[39]:= \text{or[}\text{equal[}\text{APPLY[}monoid[x],

\text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u],

\text{APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[z]], \text{set[}e[\text{monoid[}x]]], v]]], \text{APPLY[}monoid[x],

\text{PAIR[APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[z]], \text{set[}e[\text{monoid[}x]]], v], \text{APPLY[}iterate[\text{composite[}monoid[x], \text{LEFT}[y]], \text{set[}e[\text{monoid[}x]]], u]]]]],

\text{not[}\text{equal[}\text{APPLY[}monoid[x], \text{PAIR}[y, z]], \text{APPLY[}monoid[x], \text{PAIR}[z, y]]]]\] = \text{True}
In[43]:= intersection[case[equal[APPLY[monoid[x], PAIR[y, z]], APPLY[monoid[x], PAIR[z, y]]]],
   dif[composite[monoid[x]],
      cross[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]],
      iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
   composite[monoid[x],
      SWAP, cross[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]],
      iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]]]

Out[43]= composite[intersection[composite[complement[monoid[x]], SWAP], monoid[x]],
   cross[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]],
   iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
   id[intersection[complement[image[V, intersection[APPLY[monoid[x], PAIR[y, z]]],
      complement[APPLY[monoid[x], PAIR[z, y]]]]],
      complement[image[V, intersection[APPLY[monoid[x], PAIR[z, y]]]]]]] = 0

In[44]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

Lemma.

In[46]:= SubstTest[empty, intersection[case[p], dif[u, v]],
   {p → equal[APPLY[monoid[x], PAIR[y, z]], APPLY[monoid[x], PAIR[z, y]]],
   u → composite[monoid[x], cross[iterate[composite[monoid[x], LEFT[y]],
      set[e[monoid[x]]]]],
      iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
   v → composite[monoid[x], SWAP, cross[iterate[composite[monoid[x], LEFT[y]],
      set[e[monoid[x]]]]],
      iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]]]]

Out[46]= or[not[equal[APPLY[monoid[x], PAIR[y, z]], APPLY[monoid[x], PAIR[z, y]]]],
   subclass[composite[monoid[x]],
      cross[iterate[composite[monoid[x], LEFT[y]], set[e[monoid[x]]]]],
      iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]],
   true[composite[monoid[x], SWAP, cross[iterate[composite[monoid[x], LEFT[y]],
      set[e[monoid[x]]]]],
      iterate[composite[monoid[x], LEFT[z]], set[e[monoid[x]]]]]]] = True

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

Lemma.

In[48]:= SubstTest[implies, subclass[u, v],
   subclass[composite[u, w], composite[v, w]], {u → composite[x, cross[y, z]],
   v → composite[x, SWAP, cross[y, z]], w → SWAP}] // Reverse

Out[48]= or[not[subclass[composite[x, cross[y, z]], composite[x, SWAP, cross[y, z]]]],
   subclass[composite[x, SWAP, cross[z, y]], composite[x, cross[z, y]]] = True

In[49]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

Lemma.
Corollary. Any power of an element of a group commutes with any power of its inverse.
In[55]:= SubstTest[implies, 
equal[APPLY[monoid[t], PAIR[y, z]], APPLY[monoid[t], PAIR[z, y]]], 
equal[composite[monoid[t], cross[iterate[composite[monoid[t], LEFT[y]], set[e[monoid[t]]]]], 
iterate[composite[monoid[t], LEFT[z]], set[e[monoid[t]]]]], 
composite[monoid[t], SWAP, 
cross[iterate[composite[monoid[t], LEFT[y]], set[e[monoid[t]]]]], 
iterate[composite[monoid[t], LEFT[z]], set[e[monoid[t]]]]], 
{t \rightarrow \text{gp}[x], z \rightarrow \text{APPLY[inv[\text{gp}[x]], y]}}] // Reverse

Out[55]= equal[composite[\text{gp}[x], cross[iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]], 
composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]]], 
composite[\text{gp}[x], SWAP, cross[iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]], 
\text{composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]]]}, = True

In[57]:= composite[\text{gp}[x], SWAP, 
cross[iterate[composite[\text{gp}[x], LEFT[t]], set[e[\text{gp}[x]]]]], 
\text{composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[t]], set[e[\text{gp}[x]]]]]}], 
\text{t} \rightarrow \text{APPLY[inv[\text{gp}[x]], y]}}] // Reverse

Out[59]= composite[\text{gp}[x], SWAP, 
cross[composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]], 
iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]] = composite[\text{gp}[x], 
cross[composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]], 
iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]]}

In[60]:= composite[\text{gp}[x], SWAP, 
cross[composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]], 
iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]] = composite[\text{gp}[x], 
cross[composite[\text{inv[\text{gp}[x]]}, iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]], 
iterate[composite[\text{gp}[x], LEFT[y]], set[e[\text{gp}[x]]]]]]

Corollary. (An analogous result with \text{y} replaced by its inverse.)