homomorphic image of a binary operation

Johan G. F. Belinfante and Ming Li
2006 September 21

summary

The image of a binary operation under a binary homomorphism is a binary operation. This fact is derived in this notebook by a series of lemmas. The derivations of some lemmas went much faster when some steps of the proof were omitted, relying on rewrite rules to supply the needed facts.

terminology

A binary operation is defined to be a mapping from \( \text{cart}[u,u] \) to \( u \) for some set \( u \). If \( x \) is a binary operation, then \( u = \text{fix}[	ext{domain}[x]] \). A binary homomorphism from \( x \) to \( y \) is defined to be a mapping \( w \) from \( \text{fix}[	ext{domain}[x]] \) to \( \text{fix}[	ext{domain}[y]] \) such that \( \text{composite}[w, x] \) is equal to \( \text{composite}[y, \text{cross}[w, w]] \). The restriction \( \text{composite}[y, \text{id}[\text{cart}[\text{range}[w], \text{range}[w]]]] \) of \( y \) is called the homomorphic image of \( x \) under \( w \).

FUNCTION lemma

Lemma.

\[
\text{In}[2]:= \quad \text{SubstTest}[\text{implies, and}[	ext{FUNCTION}[\text{composite}[x, y]], \text{FUNCTION}[y]], \\
\quad \text{FUNCTION}[\text{composite}[x, \text{id}[\text{range}[y]]]], y \rightarrow \text{cross}[w, w]]
\]

\[
\text{Out}[2]:= \quad \text{or}[\text{FUNCTION}[\text{composite}[x, \text{id}[\text{cart}[\text{range}[w], \text{range}[w]]]]], \\
\quad \text{not}[\text{FUNCTION}[\text{composite}[	ext{Id}, w]]], \text{not}[\text{FUNCTION}[\text{composite}[x, \text{cross}[w, w]]]]] = \text{True}
\]

\[
\text{In}[3]:= \quad (\% / (x \rightarrow x_-, w \rightarrow w_-)) / \text{. Equal} \rightarrow \text{SetDelayed}
\]
The following theorem establishes that the homomorphic image is a function. (Comment. There is a speedup by a factor 20 in the following derivation by deliberately omitting two steps of the proof: `implies[and[p3,p4,p5],p6] and implies[p4,p7].`)

```plaintext
In[4]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
implies[p2, p5], implies[and[p6, p7], p8], not[implies[and[p1, p2], p8]],
{p1 \to member[x, BINOPS], p2 \to member[w, binhom[x, y]], p3 \to FUNCTION[x],
p4 \to FUNCTION[w], p5 \to equal[composite[w, x], composite[y, cross[w, w]]],
p6 \to FUNCTION[composite[y, cross[w, w]]], p7 \to FUNCTION[cart[range[w]], range[w]]],
p8 \to FUNCTION[composite[y, id[cart[range[w]], range[w]]]]])
Out[4]= or[FUNCTION[composite[y, id[cart[range[w]], range[w]]]]],
not[member[w, binhom[x, y]]], not[member[x, BINOPS]]] = True

In[5]:= (% /. \{x \to x_, y \to y_, w \to w_\}) \/. Equal \to \text{SetDelayed}
```

---

domain lemma

One needs to show that the domain of the homomorphic image is a cartesian square.

Lemma.

```plaintext
In[6]:= SubstTest[implies, equal[\{u, v\}, equal[domain[\{u\}], domain[\{v\}]]],
\{u \to composite[w, x], v \to composite[y, cross[w, w]]\}]
Out[6]= or[equal[composite[\text{inverse}[w], domain[y], w], image[\text{inverse}[x], domain[w]]],
not[\text{equal}[\text{composite}[w, x], \text{composite}[y, \text{cross}[w, w]]]]] = True

In[7]:= (% /. \{w \to w_, x \to x_, y \to y_\}) \/. Equal \to \text{SetDelayed}
```

The transform of square by a function is a square:

```plaintext
In[8]:= SubstTest[implies, equal[\{u, v\}, equal[image[\{w, u\}], image[\{w, v\}]]],
\{u \to \text{composite[\text{inverse}[funpart[z]], t, funpart[z]],
v \to \text{cartsq[s], w \to cross[funpart[z], funpart[z]]}\}]
Out[8]= or[\text{equal}[\text{cart}[\text{image}[\text{funpart[z]}, s]], \text{image}[\text{funpart[z]}, s]],
\text{composite[\text{id}[\text{range}[\text{funpart[z]]]], t, \text{id}[\text{range}[\text{funpart[z]]]]]},
not[\text{equal}[\text{cart}[s, s], \text{composite[\text{inverse}[\text{funpart[z]], t, \text{funpart[z]]]}}]]] = True

In[9]:= (% /. \{s \to s_, t \to t_, z \to z_\}) \/. Equal \to \text{SetDelayed}
```

Remove the `funpart` wrapper:

```plaintext
In[10]:= SubstTest[implies, and[\text{equal}[\{w, \text{funpart[z]}\}], \text{equal}[\text{composite[\text{inverse}[w], t, w], \text{cartsq[u]}]]],
\text{equal}[\text{cart}[\text{image}[\{w, u\}], \text{image}[\{w, u\}], \text{composite[\text{id}[\text{range}[w]], t, \text{id}[\text{range}[w]]]]], z \to w]
Out[10]= or[\text{equal}[\text{cart}[\text{image}[\{w, u\}], \text{image}[\{w, u\}], \text{composite[\text{id}[\text{range}[w]], t, \text{id}[\text{range}[w]]]]],
not[\text{equal}[\text{cart}[u, u], \text{composite[\text{inverse}[w], t, w]]], not[\text{FUNCTION}[w]]]] = True

In[11]:= (% /. \{t \to t_, u \to u_, w \to w_\}) \/. Equal \to \text{SetDelayed}
```
Lemma. (Comment. The following derivation leaves out one step of the proof: \texttt{implies[and[p3, p4, p6], p7].})

```mathematica
In[12]:= Map[not, SubstTest[and, \texttt{implies[p1, p3], implies[p2, p4], implies[p2, p5], implies[p5, p6], not[implies[and[p1, p2], p7]]}],
{p1 -> member[x, BINOPS], p2 -> member[w, binhom[x, y]], p3 -> subclass[range[x], fix[domain[x]]], p4 -> equal[domain[w], fix[domain[x]]], p5 -> equal[composite[w, x], composite[y, cross[w, w]]], p6 -> equal[composite[inverse[w], domain[y], w], image[inverse[x], domain[w]]], p7 -> equal[composite[inverse[w], domain[y], w], domain[x]]}]
```  

```
Out[12]= \texttt{or[equal[composite[inverse[w], domain[y], w], domain[x]], not[member[w, binhom[x, y]]], not[member[x, BINOPS]] = True}
```  

```
In[13]:= \texttt{or[member[w, \texttt{\_}, binhom[x, \_]], domain[\_]], not[member[\_ \_ \_, binhom[\_ \_, \_]], not[member[\_ \_, BINOPS]]] := True}
```  

Theorem. The domain of the homomorphic image is a cartesian square.

```
In[14]:= Map[not, SubstTest[and, \texttt{implies[and[p1, p2], p3], implies[p1, p4], implies[p2, p5], not[implies[and[p1, p2], p6]]},
{p1 -> member[x, BINOPS], p2 -> member[w, binhom[x, y]], p3 -> equal[composite[inverse[w], domain[y], w], domain[x]], p4 -> equal[domain[x], cartsq[fix[domain[x]]]], p5 -> FUNCTION[w], p6 -> equal[cart[\texttt{image[w, fix[domain[x]]]}, image[w, fix[domain[x]]]],
\texttt{composite[id[range[w]], domain[y], id[range[w]]]}]}]
```  

```
Out[14]= \texttt{or[equal[cart[\texttt{image[w, fix[domain[x]]]}, image[w, \texttt{fix[domain[x]]]}], \texttt{composite[id[range[w]], domain[y], id[range[w]]]}]], not[member[w, binhom[x, y]]], not[member[x, BINOPS]] = True}
```  

```
In[15]:= (% /. {\texttt{w \_\_, x \_\_, y \_\_, y \_\_}) / \texttt{Equal \\rightarrow SetDelayed}
```  

---

**range lemmas**

Lemma 1.

```
In[16]:= SubstTest[\texttt{implies, equal[u, v], equal[range[u], range[v]]},
{u -> composite[w, x], v -> composite[y, cross[w, w]]}]
```  

```
Out[16]= \texttt{or[equal[\texttt{image[w, range[x]]}, \texttt{image[y, cart[range[w], range[w]]]}],
not[equal[\texttt{composite[w, x], composite[y, cross[w, w]]}]] = True}
```  

```
In[17]:= (% /. {\texttt{w \_\_, x \_\_, y \_\_, y \_\_}) / \texttt{Equal \\rightarrow SetDelayed}
```

Theorem. If \texttt{w} is a binary homomorphism from \texttt{x} to \texttt{y}, then \texttt{range[w]} is closed under \texttt{y}. 
Restatement:

In[20]:=  
Out[20]= True

Corollary.

In[21]:=  
Out[21]= True

In[22]:=  

main theorem

Observation: One does not need to be all that specific about the assertion that some class is a cartesian square.

In[23]:=  
Out[23]= True

To show that a function \( x \) is a set, it suffices to prove that its domain is a set.

In[24]:=  
Out[24]= True

If it is also known that the domain is a square, all one needs is that \( \text{fix[domain[x]]} \) is a set. This motivates the following lemma, which provides a criterion for \( x \) to be a binary operation.
Main theorem. The homomorphic image of a binary operation is a binary operation.

The variable \( w \) that appears in one of the range lemmas is easily obtained.

serendipity

The variable \( w \) that appears in one of the range lemmas is easily obtained.