binop[x] ⊆ COMPOSE

Johan G. F. Belinfante
2011 October 7

\[\text{In[1]}:= \text{SetDirectory["l:"]; \text{\textless\textgreater goedel.11oct04b}}\]

:Package Title: goedel.11oct04b 2011 October 4 at 3:20 p.m.

Loading takes about thirteen minutes, half that time due to built-in pauses.

It is now: 2011 Oct 7 at 16:6

Loading Simplification Rules

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

\[\text{weightlimit = 40}\]

Loading completed.

It is now: 2011 Oct 7 at 16:18

summary

A subset of \text{COMPOSE} is a binary operation if and only if the restriction of \text{COMPOSE} to the cartesian square of a set which is closed under composition. All such binary operations are semigroups. An example is the addition of non-negative integers.

derivation

Theorem. The restriction of \text{COMPOSE} to \(x \times x\) is a binary operation if and only if \(x \in \text{binclosed[COMPOSE]}\).

\[\text{In[2]}:= \text{SubstTest[and, FUNCTION[t], equal[domain[t], cartsq[fix[domain[t]]],
subclass[range[t], fix[domain[t]]], member[t, V],
t -> composite[COMPOSE, id[cart[x, x]]]]]]\]

\[\text{Out[2]}= \text{\text{member[composite[COMPOSE, id[cart[x, x]]], BINOPS] =
and[member[x, V], subclass[\text{image[COMPOSE, cart[x, x]], x]]]}}\]

\[\text{In[3]}:= \text{member[composite[COMPOSE, id[cart[x_, x_]], BINOPS] =
and[member[x, V], subclass[\text{image[COMPOSE, cart[x, x]], x]]]}}\]

Lemma.
In[4]:= Map[empty, dif{id[binclosed[COMPOSE]], image[inverse[CART], image[
inverse[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]], BINOPS]]} // Normality
Out[4]= subclass[binclosed[COMPOSE], fix[image[inverse[CART],
image[inverse[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]], BINOPS]]] = True

In[5]:= % /. Equal -> SetDelayed

Theorem. A variable-free restatement.

In[6]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
{t -> composite[IMAGE[composite[id[COMPOSE], inverse[FIRST]]], CART],
u -> id[binclosed[COMPOSE]], v -> image[inverse[CART],
image[inverse[IMAGE[composite[id[COMPOSE], inverse[FIRST]]], BINOPS]]}] // Reverse
image[CART, id[binclosed[COMPOSE]]], BINOPS] = True

In[7]:= subclass[image[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]],
image[CART, id[binclosed[COMPOSE]]], BINOPS] := True

Observation. The following inclusion follows from this.

In[8]:= subclass[image[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]],
image[CART, id[binclosed[COMPOSE]]], intersection[BINOPS, P[COMPOSE]]]
Out[8]= True

The reverse inclusion will be derived in the next section and is combined with the above inclusion to obtain a variable-free equation.

---

the reverse inclusion

The binop wrapper will be used to derive the reverse inclusion.

Lemma.

In[9]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
{t -> composite[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]]},
u -> set[domain[binop[x]]], v -> image[CART, Id]] // Reverse
Out[9]= member[composite[COMPOSE, id[domain[binop[x]]]],
image[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]], image[CART, Id]] = True

In[10]:= (% /. x -> x_ ) /. Equal -> SetDelayed

Theorem. If binop[x] ⊆ COMPOSE, then binop[x] is the restriction of COMPOSE to domain[binop[x]].
Reverse, SubstTest
Equal
Theorem.

or[equal(binop[x], composite[COMPOSE, id[domain[binop[x]]]]),
not[subclass[binop[x], COMPOSE]]] = True

Theorem.

or[member[binop[x],
image[IMAGE[composite[id[COMPOSE], inverse[FIRST]]],
image[CART, Id]]],
not[subclass[binop[x], COMPOSE]]] = True

Corollary.

subclass[intersection[BINOPS, P[COMPOSE]],
image[IMAGE[composite[id[COMPOSE], inverse[FIRST]]], image[CART, Id]]] = True

Theorem.

or[member[composite[COMPOSE, id[domain[binop[x]]]], BINOPS],
not[subclass[binop[x], COMPOSE]]] = True

Lemma.
Theorem. If \( \text{binop}[x] \subseteq \text{COMPOSE} \), then \( \text{fix}[	ext{domain}[\text{binop}[x]]] \) is closed under composition.

Lemma.

Theorem. The promised reverse inclusion.

Main Theorem. A variable-free equation.
In[28]:= image[IMAGE[composite[id[COMPOSE], inverse[FIRST]]]],
    image[CART, id[binclosed[COMPOSE]]] := intersection[BINOPS, P[COMPOSE]]

---

semigroup result

Lemma.

In[29]:= SubstTest[implies, and[associative[t], subclass[image[t, cart[x, x]], x]],
    associative[composite[t, id[cart[x, x]]]], t -> COMPOSE] // Reverse

Out[29]= or[associative[composite[COMPOSE, id[cart[x, x]]]],
    not[subclass[image[COMPOSE, cart[x, x]], x]]] = True

In[30]:= or[associative[composite[COMPOSE, id[cart[x_, x_]]]],
    not[subclass[image[COMPOSE, cart[x_, x_]], x_]]] = True

Lemma.

In[31]:= Map[implies[and[member[x, V], subclass[image[COMPOSE, cart[x, x]], x]], #] &,
    SubstTest[and, associative[t], member[t, BINOPS], t -> composite[COMPOSE, id[cart[x, x]]]]]

Out[31]= or[member[composite[COMPOSE, id[cart[x, x]]], SEMIGPS],
    not[member[x, V], not[subclass[image[COMPOSE, cart[x, x]], x]]] = True

In[32]:= (% /. x -> x_) /. Equal -> SetDelayed

Lemma. (Converse result.)

In[33]:= SubstTest[implies, member[t, SEMIGPS],
    member[t, BINOPS], t -> composite[COMPOSE, id[cart[x, x]]]] // Reverse

Out[33]= or[and[member[x, V], subclass[image[COMPOSE, cart[x, x]], x]],
    not[member[composite[COMPOSE, id[cart[x, x]]], SEMIGPS]]] = True

In[34]:= (% /. x -> x_) /. Equal -> SetDelayed

Theorem. The restriction of \( \text{COMPOSE} \) to \( x \times x \) is a semigroup if and only if \( x \in \text{binclosed[COMPOSE]} \).

In[35]:= equiv[member[composite[COMPOSE, id[cart[x, x]]], SEMIGPS],
    and[member[x, V], subclass[image[COMPOSE, cart[x, x]], x]]]

Out[35]= True

In[36]:= member[composite[COMPOSE, id[cart[x_, x_]]], SEMIGPS] :=
    and[member[x, V], subclass[image[COMPOSE, cart[x, x]], x]]

Lemma.
Theorem. Any binary operation which is a subset of COMPOSE is a semigroup.

Corollary. An equation.

definition of non-negative integers

The non-negative integers form a semigroup under addition.
Theorem. The integer zero is a neutral element for addition.

In[49]:= Map[member[id[omega]], range[#]] &,
    SubstTest[fix, composite[inverse[FIRST], Di, COMPOSE, SWAP, id[cart[x, x]]],
        x -> range[PLUS]]] // Reverse

Out[49]= member[id[omega]],
    image[fix[composite[inverse[FIRST], Di, COMPOSE]], range[PLUS]]] = False

In[50]:= % /. Equal -> SetDelayed

Theorem. The integer zero is a neutral element for addition.

In[51]:= Map[member[id[omega]], #] &,
    SubstTest[intersection, fix[domain[w]],
        complement[domain[fix[composite[inverse[SECOND], Di, w]]]],
        complement[range[fix[composite[inverse[FIRST], Di, w]]]],
        w -> composite[COMPOSE, id[cartsq[range[PLUS]]]]]

Out[51]= member[id[omega]], ids[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]]] = True

In[52]:= % /. Equal -> SetDelayed

Theorem.

In[53]:= Map[member[id[omega]], #] &,
    SubstTest[ids, binop[t], t -> composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]]

Out[53]= equal[e[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]], id[omega]] = True

In[54]:= e[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] := id[omega]

Theorem.

In[55]:= SubstTest[ids, binop[t],
    t -> composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] // Reverse

Out[55]= ids[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] = set[id[omega]]

In[56]:= ids[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] := set[id[omega]]

Lemma. Addition is associative.

In[57]:= SubstTest[associative, semig[t],
    t -> composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] // Reverse

Out[57]= associative[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] = True

In[58]:= associative[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]]] := True

Theorem. The non-negative integers form a monoid under addition.

In[59]:= member[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]], MONOIDS] // AssertTest

Out[59]= member[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]], MONOIDS] = True

In[60]:= member[composite[COMPOSE, id[cart[range[PLUS], range[PLUS]]]], MONOIDS] := True