subsemigroups

Johan G. F. Belinfante
2009 July 15

```
In[1]:= SetDirectory["l:"]; << goedel.09jul14a;<< tools.m

        :Package Title: goedel.09jul14a              2009 July 14 at 3:55 p.m.

        It is now:  2009 Jul 15 at 14:35

        Loading Simplification Rules

        TOOLS.M                                      Revised 2009 July 2

        weightlimit = 40
```

summary

The restriction of a given semigroup operation **x** to the cartesian square of any set binary-closed under **x** is a again a semigroup operation, and conversely, all semigroup operations contained in a given one are of this form. Because each semigroup operation contained in a given one is determined by the fixed-point class of its domain, one can set up an explicit correspondence between the semigroup operations contained in a given semigroup operation and the sets binary-closed under it. The class **intersection[SEMIGPS, P[x]]** of all semigroup operations contained in a given semigroup operation **x** is a set. An explicit formula relating this set to the class **binclosed[x]** of sets closed under **x** is derived in this notebook. Similar results for binary operations in general are already available and can be specialized to semigroups by taking results about **binop[t]** and making the replacement **t → semigp[x]**.

derivation

Theorem. Every subclass of a semigroup operation is a restriction.

```
In[2]:= SubstTest[implies, and[subclass[w, t], FUNCTION[t]],
          equal[w, composite[t, id[domain[w]]]]], t → semigp[x]] // Reverse

Out[2]= or[equal[w, composite[semigp[x], id[domain[w]]]], not[subclass[w, semigp[x]]]] == True

In[3]:= or[equal[composite[semigp[x_], id[domain[w_]]], w_],
          not[subclass[w_, semigp[x_]]]] := True
```

Theorem. If a binary operation **y** is a subclass of **semigp[x]**, then **y** is the restriction of **semigp[x]** to the cartesian square of **fix[domain[y]]**.

```
In[4]:= Map[not, SubstTest[and, implies[p2, p3], implies[p1, p4],
          implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]], {p1 → member[y, BINOPS],
           p2 → subclass[y, semigp[x]], p3 → equal[y, composite[semigp[x], id[domain[y]]]],
           p4 → equal[cart[fix[domain[y]], fix[domain[y]]], domain[y]], p5 → equal[y,
             composite[semigp[x], id[cart[fix[domain[y]], fix[domain[y]]]]]]}]] // Reverse

Out[4]= or[equal[y, composite[semigp[x], id[cart[fix[domain[y]], fix[domain[y]]]]]],
         not[member[y, BINOPS]], not[subclass[y, semigp[x]]]] == True

In[5]:= or[equal[y_, composite[semigp[x_], id[cart[fix[domain[y_]], fix[domain[y_]]]]]],
         not[member[y_, BINOPS]], not[subclass[y_, semigp[x_]]]] := True
```

Lemma. (An explicit formula for the range of a restriction of **semigp[x]**.)

```
In[6]:= SubstTest[implies, equal[y, t], equal[range[y], range[t]],
         t -> composite[semigp[x], id[cart[fix[domain[y]], fix[domain[y]]]]]] // Reverse

Out[6]= or[equal[image[semigp[x], cart[fix[domain[y]], fix[domain[y]]]], range[y]],
         not[equal[y, composite[semigp[x], id[cart[fix[domain[y]], fix[domain[y]]]]]]]] == True

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

Theorem. If a binary operation **y** is contained in **semigp[x]**, then **fix[domain[y]]** is binary-closed under **semigp[x]**.

```
In[8]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p3, p4],
          implies[p1, p5], implies[and[p4, p5], p6], not[implies[and[p1, p2], p6]],
          {p1 → member[y, BINOPS], p2 → subclass[y, semigp[x]],
           p3 → equal[y, composite[semigp[x], id[cart[fix[domain[y]], fix[domain[y]]]]]],
           p4 → equal[range[y], image[semigp[x], cartsq[fix[domain[y]]]]],
           p5 → subclass[range[y], fix[domain[y]]], p6 -> subclass[image[semigp[x],
              cart[fix[domain[y]], fix[domain[y]]]], fix[domain[y]]]}]] // Reverse

Out[8]= or[not[member[y, BINOPS]], not[subclass[y, semigp[x]]], subclass[
          image[semigp[x], cart[fix[domain[y]], fix[domain[y]]]], fix[domain[y]]]] == True

In[9]:= or[not[member[y_, BINOPS]], not[subclass[y_, semigp[x_]]], subclass[
          image[semigp[x_], cart[fix[domain[y_]], fix[domain[y_]]]], fix[domain[y_]]]] := True
```

Converse Theorem. The restriction of a semigroup operation to the cartesian square of any binary-closed class is a binary operation.

```
In[10]:= SubstTest[implies, subclass[image[binop[t], cart[y, y]], y],
          member[composite[binop[t], id[cart[y, y]]], BINOPS], t → semigp[x]] // Reverse

Out[10]= or[member[composite[semigp[x], id[cart[y, y]]], BINOPS],
          not[subclass[image[semigp[x], cart[y, y]], y]]] == True

In[11]:= or[member[composite[semigp[x_], id[cart[y_, y_]]], BINOPS],
          not[subclass[image[semigp[x_], cart[y_, y_]], y_]]] := True
```

Lemma. The restriction of a semigroup operation to the cartesian square of any binary-closed class is associative.

```
In[12]:= SubstTest[implies, subclass[image[assoc[t], cart[y, y]], y],
           associative[composite[assoc[t], id[cart[y, y]]]], t → semigp[x]] // Reverse

Out[12]= or[associative[composite[semigp[x], id[cart[y, y]]]],
           not[subclass[image[semigp[x], cart[y, y]], y]]] == True

In[13]:= or[associative[composite[semigp[x_], id[cart[y_, y_]]]],
           not[subclass[image[semigp[x_], cart[y_, y_]], y_]]] := True
```

Combining these lemmas yields the following theorem.

Theorem. The restriction of a semigroup operation to the cartesian square of any binary-closed class is a semigroup operation.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[p1, p4]], {p1 -> subclass[image[semigp[x], cart[y, y]], y],
             p2 → member[composite[semigp[x], id[cart[y, y]]], BINOPS],
             p3 -> associative[composite[semigp[x], id[cart[y, y]]]],
             p4 -> member[composite[semigp[x], id[cart[y, y]]], SEMIGPS]}]] // Reverse

Out[14]= or[member[composite[semigp[x], id[cart[y, y]]], SEMIGPS],
           not[subclass[image[semigp[x], cart[y, y]], y]]] == True

In[15]:= or[member[composite[semigp[x_], id[cart[y_, y_]]], SEMIGPS],
           not[subclass[image[semigp[x_], cart[y_, y_]], y_]]] := True
```

an explicit formula for the set of subsemigroups of a given semigroup

Lemma. (Eliminating the variable **y**.)

```
In[16]:= Map[equal[V, #] &, SubstTest[class, y, implies[member[y, u], member[y, v]],
           {u → binclosed[semigp[x]], v → fix[image[inverse[CART], image[inverse[IMAGE[DUP]],
                image[inverse[IMAGE[cross[Id, semigp[x]]]], SEMIGPS]]]]}]]

Out[16]= subclass[binclosed[semigp[x]], fix[image[inverse[CART], image[inverse[IMAGE[DUP]],
              image[inverse[IMAGE[cross[Id, semigp[x]]]], SEMIGPS]]]]] == True

In[17]:= (% /. x → x_) /. Equal → SetDelayed
```

An improved statement of this lemma can be derived by applying a suitable image to both sides of the above inclusion.

Theorem. The class of restrictions of a given semigroup operation **semigp[x]** to cartesian squares of binary-closed subsets is contained in the class **SEMIGPS** of all semigroup operations. (This inclusion will later be replaced with an equation.)

```
In[18]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]], {t → IMAGE[DUP],
            u → image[CART, id[binclosed[semigp[x]]]], v → image[inverse[IMAGE[DUP]],
              image[inverse[IMAGE[cross[Id, semigp[x]]]], SEMIGPS]]}] // Reverse

Out[18]= subclass[image[IMAGE[composite[id[semigp[x]], inverse[FIRST]]],
            image[CART, id[binclosed[semigp[x]]]]], SEMIGPS] == True
```

```
In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary. Any binary operation that is a subclass of a semigroup operation is itself a semigroup operation.

```
In[20]:= Map[subclass[#, SEMIGPS] &,
          SubstTest[image, IMAGE[composite[id[binop[t]], inverse[FIRST]]],
           image[CART, id[binclosed[binop[t]]]], t → semigp[x]]]

Out[20]= subclass[intersection[BINOPS, P[semigp[x]]], SEMIGPS] == True

In[21]:= subclass[intersection[BINOPS, P[semigp[x_]]], SEMIGPS] := True
```

Corollary. A temporary rewrite rule.

```
In[22]:= equal[intersection[BINOPS, P[semigp[x]]],
           intersection[SEMIGPS, P[semigp[x]]]] // AssertTest

Out[22]= equal[intersection[BINOPS, P[semigp[x]]], intersection[SEMIGPS, P[semigp[x]]]] == True

In[23]:= intersection[BINOPS, P[semigp[x_]]] := intersection[SEMIGPS, P[semigp[x]]]
```

Corollary. An explicit formula for the set of subsemigroups of a given semigroup.

```
In[24]:= SubstTest[image, IMAGE[composite[id[binop[t]], inverse[FIRST]]],
           image[CART, id[binclosed[binop[t]]]], t → semigp[x]] // Reverse

Out[24]= image[IMAGE[composite[id[semigp[x]], inverse[FIRST]]],
           image[CART, id[binclosed[semigp[x]]]]] == intersection[SEMIGPS, P[semigp[x]]]

In[25]:= image[IMAGE[composite[id[semigp[x_]], inverse[FIRST]]],
           image[CART, id[binclosed[semigp[x_]]]]] := intersection[SEMIGPS, P[semigp[x]]]
```

Corollary. (Restatement without the **semigp** wrapper.)

```
In[26]:= SubstTest[implies, equal[x, semigp[t]],
           equal[intersection[SEMIGPS, P[x]], image[IMAGE[composite[id[x], inverse[FIRST]]],
             image[CART, id[binclosed[x]]]]]], t → x] // Reverse

Out[26]= or[
           equal[image[IMAGE[composite[id[x], inverse[FIRST]]], image[CART, id[binclosed[x]]]],
            intersection[SEMIGPS, P[x]]], not[member[x, SEMIGPS]]] == True

In[27]:= or[equal[
            image[IMAGE[composite[id[x_], inverse[FIRST]]], image[CART, id[binclosed[x_]]]],
            intersection[SEMIGPS, P[x_]]], not[member[x_, SEMIGPS]]] := True
```