semigroup wrapper

Johan G. F. Belinfante
2008 October 19

```
In[1]:= SetDirectory["l:"]; << goedel.08oct18b;<< tools.m

        :Package Title: goedel.08oct18b          2008 October 18 at 8:00 p.m.

        It is now:  2008 Oct 19 at 21:28

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 17

        weightlimit = 40
```

summary

A new semigroup wrapper **semigp[x]** is defined by analogy with the **binop[x]** wrapper for binary operations, and many of its basic properties are derived as corollaries of results for **binop[x]**. Aside from translating various previously known facts, one minor new result emerged as an application: a flip rule for semigroups.

definition

The following rewrite rule serves to define the **semigp[x]** wrapper:

```
In[3]:= image[V, intersection[semigp[x_], set[y_]]] :=
         intersection[image[V, intersection[ASSOCIATIVE, set[x]]],
          image[V, intersection[BINOPS, set[x]]], image[V, intersection[x, set[y]]]]
```

normalization

Theorem.

```
In[7]:= semigp[x] // Renormality// Reverse

Out[7]= intersection[binop[x], image[V, intersection[ASSOCIATIVE, set[x]]]] == semigp[x]

In[8]:= intersection[binop[x_], image[V, intersection[ASSOCIATIVE, set[x_]]]] := semigp[x]
```

wrapper introduction and removal rules

Lemma. (Simplification rule.)

```
In[10]:= equiv[or[equal[0, x], member[x, SEMIGPS]], member[x, SEMIGPS]]

Out[10]= True

In[12]:= or[equal[0, x_], member[x_, SEMIGPS]] := member[x, SEMIGPS]
```

Theorem. (Wrapper removal rule.)

```
In[13]:= SubstTest[equal, x,
           intersection[x, image[V, intersection[t, set[x]]]], t → SEMIGPS] // Reverse

Out[13]= equal[x, semigp[x]] == member[x, SEMIGPS]

In[14]:= equal[x_, semigp[x_]] := member[x, SEMIGPS]
```

Theorem. (Automatic removal.)

```
In[15]:= implies[member[x, SEMIGPS], equal[semigp[x], x]]

Out[15]= True

In[16]:= semigp[x_] := x /; member[x, SEMIGPS]
```

Theorem. (Wrapper introduction rule.)

```
In[17]:= SubstTest[member,
           intersection[x, image[V, intersection[t, set[x]]]], t, t → SEMIGPS] // Reverse

Out[17]= member[semigp[x], SEMIGPS] == True

In[18]:= member[semigp[x_], SEMIGPS] := True
```

reify rule

```
In[19]:= SubstTest[reify, x,
           intersection[f[x], image[V, intersection[t, set[f[x]]]]], t → SEMIGPS] // Reverse

Out[19]= reify[x, semigp[f[x]]] ==
          composite[reify[x, f[x]], id[image[inverse[VERTSECT[reify[x, f[x]]], SEMIGPS]]]

In[20]:= reify[x_, semigp[y_]] :=
          composite[reify[x, y], id[image[inverse[VERTSECT[reify[x, y]]], SEMIGPS]]]
```

properties of the semigroup wrapper

Theorem.

```
In[21]:= SubstTest[implies, member[t, SEMIGPS], member[t, BINOPS], t → semigp[x]] // Reverse
Out[21]= member[semigp[x], BINOPS] == True
```

```
In[22]:= member[semigp[x_], BINOPS] := True
```

Corollary.

```
In[23]:= SubstTest[subclass, binop[t], cart[cart[V, V], V], t → semigp[x]] // Reverse
Out[23]= subclass[semigp[x], cart[cart[V, V], V]] == True
```

```
In[24]:= subclass[semigp[x_], cart[cart[V, V], V]] := True
```

Corollary.

```
In[25]:= SubstTest[composite, binop[t], id[cart[V, V]], t → semigp[x]] // Reverse
Out[25]= composite[semigp[x], id[cart[V, V]]] == semigp[x]
```

```
In[26]:= composite[semigp[x_], id[cart[V, V]]] := semigp[x]
```

Theorem.

```
In[27]:= SubstTest[member, binop[t], V, t → semigp[x]] // Reverse
Out[27]= member[semigp[x], V] == True
```

```
In[28]:= member[semigp[x_], V] := True
```

Theorem

```
In[29]:= SubstTest[FUNCTION, binop[t], t → semigp[x]] // Reverse
Out[29]= FUNCTION[semigp[x]] == True
```

```
In[30]:= FUNCTION[semigp[x_]] := True
```

Corollary.

```
In[31]:= SubstTest[implies, equal[x, semigp[t]], FUNCTION[x], t → x] // Reverse
Out[31]= or[FUNCTION[x], not[member[x, SEMIGPS]]] == True
```

```
In[32]:= or[FUNCTION[x_], not[member[x_, SEMIGPS]]] := True
```

flip rule

Theorem.

```
In[41]:= (or[member[composite[t, SWAP], SEMIGPS], not[member[t, SEMIGPS]]] // AssertTest//
           MapNotNot) /. t → semigp[x]

Out[41]= member[composite[semigp[x], SWAP], SEMIGPS] == True

In[42]:= member[composite[semigp[x_], SWAP], SEMIGPS] := True
```

Corollary.

```
In[45]:= SubstTest[implies, equal[x, semigp[t]], member[flip[x], SEMIGPS], t → x] // Reverse

Out[45]= or[member[composite[x, SWAP], SEMIGPS], not[member[x, SEMIGPS]]] == True

In[46]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[48]:= SubstTest[implies, equal[x, semigp[t]], member[flip[flip[x]], SEMIGPS], t → x] // Reverse

Out[48]= or[member[composite[x, id[cart[V, V]]], SEMIGPS], not[member[x, SEMIGPS]]] == True

In[49]:= or[member[composite[x_, id[cart[V, V]]], SEMIGPS], not[member[x_, SEMIGPS]]] := True
```

Lemma.

```
In[52]:= SubstTest[implies, member[t, SEMIGPS],
           member[flip[t], SEMIGPS], t → flip[flip[x]]] // Reverse

Out[52]= or[member[composite[x, SWAP], SEMIGPS],
           not[member[composite[x, id[cart[V, V]]], SEMIGPS]]] == True

In[53]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[50]:= SubstTest[implies, member[t, SEMIGPS], member[flip[t], SEMIGPS], t → flip[x]] // Reverse

Out[50]= or[member[composite[x, id[cart[V, V]]], SEMIGPS],
           not[member[composite[x, SWAP], SEMIGPS]]] == True

In[51]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[54]:= equiv[member[composite[x, SWAP], SEMIGPS],
           member[composite[x, id[cart[V, V]]], SEMIGPS]]

Out[54]= True
```

```
In[56]:= member[composite[x_, SWAP], SEMIGPS] := member[composite[x, id[cart[V, V]]], SEMIGPS]
```

domain and range rules

Theorem.

```
In[57]:= SubstTest[empty, domain[funpart[t]], t → semigp[x]] // Reverse
Out[57]= equal[0, domain[semigp[x]]] == equal[0, semigp[x]]

In[58]:= equal[0, domain[semigp[x_]]] := equal[0, semigp[x]]
```

Lemma.

```
In[59]:= ImageComp[semigp[x], id[cart[V, V]], V] // Reverse
Out[59]= image[semigp[x], cart[V, V]] == range[semigp[x]]

In[60]:= image[semigp[x_], cart[V, V]] := range[semigp[x]]
```

Theorem.

```
In[65]:= SubstTest[cartsq, domain[domain[binop[t]]], t → semigp[x]] // Reverse
Out[65]= cart[fix[domain[semigp[x]]], fix[domain[semigp[x]]]] == domain[semigp[x]]

In[66]:= cart[fix[domain[semigp[x_]]], fix[domain[semigp[x_]]]] := domain[semigp[x]]
```

Theorem.

```
In[68]:= SubstTest[domain, domain[binop[t]], t → semigp[x]] // Reverse
Out[68]= domain[domain[semigp[x]]] == fix[domain[semigp[x]]]

In[69]:= domain[domain[semigp[x_]]] := fix[domain[semigp[x]]]
```

Theorem.

```
In[70]:= SubstTest[range, cart[t, t], t → fix[domain[semigp[x]]]] // Reverse
Out[70]= range[domain[semigp[x]]] == fix[domain[semigp[x]]]

In[71]:= range[domain[semigp[x_]]] := fix[domain[semigp[x]]]
```

Corollary.

```
In[72]:= SubstTest[inverse, cartsq[t], t → fix[domain[semigp[x]]]] // Reverse
Out[72]= inverse[domain[semigp[x]]] == domain[semigp[x]]

In[73]:= inverse[domain[semigp[x_]]] := domain[semigp[x]]
```

Corollary.

```
In[74]:= SubstTest[inverse, inverse[t], t → domain[semigp[x]]]

Out[74]= composite[Id, domain[semigp[x]]] == domain[semigp[x]]

In[75]:= composite[Id, domain[semigp[x_]]] := domain[semigp[x]]
```

associativity

Lemma.

```
In[77]:= SubstTest[implies, member[t, SEMIGPS], associative[t], t → semigp[x]] // Reverse

Out[77]= associative[semigp[x]] == True

In[78]:= associative[semigp[x_]] := True
```

Theorem.

```
In[80]:= SubstTest[implies, associative[t], equal[composite[t, cross[Id, t], ASSOC],
           composite[t, cross[t, Id]]], t → semigp[x]] // Reverse

Out[80]= equal[composite[semigp[x], cross[semigp[x], Id]],
           composite[semigp[x], cross[Id, semigp[x]], ASSOC]] == True

In[82]:= composite[semigp[x_], cross[Id, semigp[x_]], ASSOC] :=
          composite[semigp[x], cross[semigp[x], Id]]
```

Corollary.

```
In[88]:= Map[equal[#, composite[semigp[x], RIGHT[z], semigp[x], LEFT[y]]] &, Assoc[
           composite[semigp[x], cross[Id, semigp[x]]], ASSOC, composite[RIGHT[z], LEFT[y]]]]

Out[88]= equal[composite[semigp[x], LEFT[y], semigp[x], RIGHT[z]],
           composite[semigp[x], RIGHT[z], semigp[x], LEFT[y]]] == True

In[89]:= equal[composite[semigp[x_], LEFT[y_], semigp[x_], RIGHT[z_]],
           composite[semigp[x_], RIGHT[z_], semigp[x_], LEFT[y_]]] := True
```

Corollary.

```
In[97]:= SubstTest[implies, associative[t],
           TRANSITIVE[composite[t, inverse[FIRST]]], t → semigp[x]] // Reverse

Out[97]= TRANSITIVE[composite[semigp[x], inverse[FIRST]]] == True

In[98]:= TRANSITIVE[composite[semigp[x_], inverse[FIRST]]] := True
```

Corollary.

```
In[99]:= SubstTest[implies, associative[t],
           TRANSITIVE[composite[t, inverse[SECOND]]], t → semigp[x]] // Reverse

Out[99]= TRANSITIVE[composite[semigp[x], inverse[SECOND]]] == True

In[100]:=
         TRANSITIVE[composite[semigp[x_], inverse[SECOND]]] := True
```