commuting elements of a semigroup

Johan G. F. Belinfante
2013 July 5

```
In[1]:= SetDirectory["l:"]; << goedel.13jul04a

        :Package Title: goedel.13jul04a                    2013 July 4 at 5:25 p.m.

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

        It is now:  2013 Jul 5 at 9:36

        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 Jul 5 at 9:53
```

summary

The commutativity relation for a semigroup is related to the relation **COMMUTE** via the regular representation **APPLY[-CURRY, semigp[x]]**.

derivation

One key to reasoning about the regular representation of a semigroup is to use unwrapped variables, which prevents rewrite rules from introducing **image[V, –]** expressions. The latter expressions are particularly hard to deal with in membership rules involving ordered pairs. The following removes the **semigp** wrapper from the application rule for the regular representation.

Theorem. Unwrapped application rule for the regular representation of a semigroup.

```
In[2]:= SubstTest[implies, and[equal[x, semigp[t]], member[y, fix[domain[x]]]],
          equal[APPLY[APPLY[CURRY, x], y], composite[x, LEFT[y]]], t → x] // Reverse

Out[2]= or[equal[APPLY[APPLY[CURRY, x], y], composite[x, LEFT[y]]],
          not[member[x, SEMIGPS]], not[member[y, fix[domain[x]]]]] == True

In[3]:= or[equal[APPLY[APPLY[CURRY, x_], y_], composite[x_, LEFT[y_]]],
          not[member[x_, SEMIGPS]], not[member[y_, fix[domain[x_]]]]] := True
```

the COMMUTE theorem

In this section a theorem is derived that relates the commutativity relation for a semigroup to the relation **COMMUTE**.

Lemma. A temporary rewrite rule involving **funpart**.

```
In[4]:= member[pair[u, v], composite[inverse[funpart[t]], s, funpart[t]]] // AssertTest

Out[4]= member[pair[u, v], composite[inverse[funpart[t]], s, funpart[t]]] ==
         and[member[u, domain[funpart[t]]], member[v, domain[funpart[t]]],
          member[pair[APPLY[funpart[t], u], APPLY[funpart[t], v]], s]]

In[5]:= member[pair[u_, v_], composite[inverse[funpart[t_]], s_, funpart[t_]]] :=
         and[member[u, domain[funpart[t]]], member[v, domain[funpart[t]]],
          member[pair[APPLY[funpart[t], u], APPLY[funpart[t], v]], s]]
```

Theorem. A temporary rewrite rule for introducing membership in a composite involving **COMMUTE**.

```
In[6]:= SubstTest[implies, and[equal[x, funpart[t]], member[u, domain[x]],
           member[v, domain[x]], commute[APPLY[x, u], APPLY[x, v]]],
          member[pair[u, v], composite[inverse[x], COMMUTE, x]], t → x] // Reverse

Out[6]= or[member[pair[u, v], composite[inverse[x], COMMUTE, x]],
          not[equal[composite[APPLY[x, u], APPLY[x, v]], composite[APPLY[x, v], APPLY[x, u]]]],
          not[FUNCTION[x]], not[member[u, domain[x]]], not[member[v, domain[x]]]] == True

In[7]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Lemma. Specialization to the case of the function **APPLY[CURRY, x]**.

```
In[10]:= Map[implies[member[x, y], #] &,
           SubstTest[or, member[pair[u, v], composite[inverse[t], COMMUTE, t]],
            not[equal[composite[APPLY[t, u], APPLY[t, v]], composite[APPLY[t, v], APPLY[t, u]]]],
            not[FUNCTION[t]], not[member[u, domain[t]]],
            not[member[v, domain[t]]], t → APPLY[CURRY, x]]] // Reverse

Out[10]= or[member[pair[u, v], composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]],
           not[equal[composite[APPLY[APPLY[CURRY, x], u], APPLY[APPLY[CURRY, x], v]],
             composite[APPLY[APPLY[CURRY, x], v], APPLY[APPLY[CURRY, x], u]]]],
           not[member[u, domain[domain[x]]]], not[member[v, domain[domain[x]]]],
           not[member[x, y]], not[subclass[x, cart[cart[V, V], V]]] == True

In[11]:= (% /. {x → x_, y → y_, u → u_, v → v_}) /. Equal → SetDelayed
```

Theorem. Specialization to the case that **x** is a semigroup.

```
In[12]:= Map[not, SubstTest[and, (*implies[p2,p5],implies[p3,p6],implies[p1,p7],*)
           implies[and[p1, p4, p5, p6, p7], p8],
           not[implies[and[p1, p2, p3, p4], p8]], {p1 → member[x, SEMIGPS],
            p2 → member[u, fix[domain[x]]], p3 → member[v, fix[domain[x]]],
            p4 → commute[APPLY[APPLY[CURRY, x], u], APPLY[APPLY[CURRY, x], v]],
            p5 -> member[u, domain[domain[x]]], p6 -> member[v, domain[domain[x]]],
            p7 -> subclass[x, cart[cart[V, V], V]], p8 -> member[pair[u, v],
              composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]]}]] // Reverse

Out[12]= or[member[pair[u, v], composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]],
           not[equal[composite[APPLY[APPLY[CURRY, x], u], APPLY[APPLY[CURRY, x], v]],
             composite[APPLY[APPLY[CURRY, x], v], APPLY[APPLY[CURRY, x], u]]]],
           not[member[u, fix[domain[x]]]], not[member[v, fix[domain[x]]]],
           not[member[x, SEMIGPS]]] == True

In[13]:= (% /. {x → x_, y → y_, u → u_, v → v_}) /. Equal → SetDelayed
```

Lemma. Main result using ordered pairs. (This takes quite a while.)

```
In[14]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           (* implies[and[p1,p2],p4],implies[and[p1,p2],p5],*)
           implies[and[p1, p4], p6], implies[and[p1, p5], p7],
           (*implies[and[p3,p6,p7],p8],*) implies[and[p1, p4, p5, p8], p9],
           not[implies[and[p1, p2], p9]], {p1 → member[x, SEMIGPS],
            p2 -> member[pair[u, v], fix[composite[SWAP, inverse[x], x]]],
            p3 -> commute[composite[x, LEFT[u]], composite[x, LEFT[v]]],
            p4 → member[u, fix[domain[x]]], p5 → member[v, fix[domain[x]]],
            p6 → equal[composite[x, LEFT[u]], APPLY[APPLY[CURRY, x], u]],
            p7 → equal[composite[x, LEFT[v]], APPLY[APPLY[CURRY, x], v]],
            p8 → commute[APPLY[APPLY[CURRY, x], u], APPLY[APPLY[CURRY, x], v]],
            p9 → member[pair[u, v],
              composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]]}]] // Reverse

Out[14]= or[member[pair[u, v], composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]],
           not[member[x, SEMIGPS]],
           not[member[pair[u, v], fix[composite[SWAP, inverse[x], x]]]]] == True

In[15]:= (% /. {x → x_, u → u_, v → v_}) /. Equal → SetDelayed
```

The following theorem eliminates the pair of variables **u** and **v**.

Theorem. An inclusion relating the commutativity relation of a semigroup to the relation **COMMUTE**, using the regular representation.

```
In[16]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[u, v],
           implies[and[member[x, w], member[pair[u, v], y]], member[pair[u, v], z]],
           {w → SEMIGPS, y -> fix[composite[SWAP, inverse[x], x]],
            z -> composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]}]] // MapNotNot

Out[16]= or[not[member[x, SEMIGPS]], subclass[fix[composite[SWAP, inverse[x], x]],
           composite[inverse[APPLY[CURRY, x]], COMMUTE, APPLY[CURRY, x]]]] == True
```

```
In[17]:= or[not[member[x_, SEMIGPS]], subclass[fix[composite[SWAP, inverse[x_], x_]],
           composite[inverse[APPLY[CURRY, x_]], COMMUTE, APPLY[CURRY, x_]]]] := True
```

Corollary. Reintroduce the **semigp** wrapper.

```
In[18]:= SubstTest[implies, member[t, SEMIGPS],
           subclass[fix[composite[SWAP, inverse[t], t]], composite[
             inverse[APPLY[CURRY, t]], COMMUTE, APPLY[CURRY, t]]], t → semigp[x]] // Reverse

Out[18]= subclass[fix[composite[SWAP, inverse[semigp[x]], semigp[x]]],
           composite[inverse[APPLY[CURRY, semigp[x]]], COMMUTE, APPLY[CURRY, semigp[x]]]] == True

In[19]:= subclass[fix[composite[SWAP, inverse[semigp[x_]], semigp[x_]]], composite[
           inverse[APPLY[CURRY, semigp[x_]]], COMMUTE, APPLY[CURRY, semigp[x_]]]] := True
```

Corollary.

```
In[20]:= SubstTest[implies, subclass[y, z], subclass[composite[u, y, v], composite[u, z, v]],
           {u → APPLY[CURRY, semigp[x]], v → inverse[APPLY[CURRY, semigp[x]]],
            y -> fix[composite[SWAP, inverse[semigp[x]], semigp[x]]], z -> composite[
              inverse[APPLY[CURRY, semigp[x]]], COMMUTE, APPLY[CURRY, semigp[x]]]}] // Reverse

Out[20]= subclass[composite[APPLY[CURRY, semigp[x]],
           fix[composite[SWAP, inverse[semigp[x]], semigp[x]]],
           inverse[APPLY[CURRY, semigp[x]]]], COMMUTE] == True

In[21]:= subclass[composite[APPLY[CURRY, semigp[x_]],
           fix[composite[SWAP, inverse[semigp[x_]], semigp[x_]]],
           inverse[APPLY[CURRY, semigp[x_]]]], COMMUTE] := True
```

---

special cases

In this section the main theorem is specialized to some examples involving commutative groups.

Theorem.

```
In[22]:= SubstTest[subclass, composite[APPLY[CURRY, semigp[x]],
           fix[composite[SWAP, inverse[semigp[x]], semigp[x]]],
           inverse[APPLY[CURRY, semigp[x]]]], COMMUTE, x → INTADD] // Reverse

Out[22]= subclass[cart[range[INTPLUS], range[INTPLUS]], COMMUTE] == True

In[23]:= subclass[cart[range[INTPLUS], range[INTPLUS]], COMMUTE] := True
```

Theorem.

```
In[24]:= SubstTest[subclass, composite[APPLY[CURRY, semigp[x]],
           fix[composite[SWAP, inverse[semigp[x]], semigp[x]]],
           inverse[APPLY[CURRY, semigp[x]]]], COMMUTE, x → RATADD] // Reverse

Out[24]= subclass[cart[range[RATPLUS], range[RATPLUS]], COMMUTE] == True
```

```
In[25]:= subclass[cart[range[RATPLUS], range[RATPLUS]], COMMUTE] := True
```

Theorem.

```
In[26]:= SubstTest[subclass, composite[APPLY[CURRY, semigp[x]],
            fix[composite[SWAP, inverse[semigp[x]], semigp[x]]],
            inverse[APPLY[CURRY, semigp[x]]]], COMMUTE, x → RATMUL] // Reverse
Out[26]= subclass[cart[binhom[RATADD, RATADD], binhom[RATADD, RATADD]], COMMUTE] == True

In[27]:= subclass[cart[binhom[RATADD, RATADD], binhom[RATADD, RATADD]], COMMUTE] := True
```