commuting powers

Johan G. F. Belinfante
2013 May 29

```
In[1]:= SetDirectory["l:"]; << goedel.13may27a

        :Package Title: goedel.13may27a                 2013 May 27 at 12:00 noon

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

        It is now:  2013 May 29 at 12:7

        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 May 29 at 12:23
```

summary

If **x** and **y** commute, then any power of **x** commutes with any power of **y**. A variable-free statement of this fact is derived.

derivation

Theorem. If **x** and **y** commute, then any power of **x** commutes with any power of **y**.

```
In[2]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → commute[x, y], p2 → commute[x, image[power[y], v]],
            p3 -> commute[image[power[x], u], image[power[y], v]]}]] // Reverse

Out[2]= or[equal[composite[image[power[x], u], image[power[y], v]],
           composite[image[power[y], v], image[power[x], u]]],
         not[equal[composite[x, y], composite[y, x]]]] == True

In[3]:= or[equal[composite[image[power[x_], u_], image[power[y_], v_]],
           composite[image[power[y_], v_], image[power[x_], u_]]],
         not[equal[composite[x_, y_], composite[y_, x_]]]] := True
```

The following observation is the key to eliminating the variables **u** and **v**.

```
In[4]:= reify[w, cart[set[first[w]], set[second[w]]]]

Out[4]= id[cart[V, V]]
```

Lemma. A simplification rule.

```
In[5]:= subclass[composite[SWAP, x], composite[SWAP, y]] // AssertTest

Out[5]= subclass[composite[SWAP, x], composite[SWAP, y]] ==
         subclass[composite[id[cart[V, V]], x], y]

In[6]:= subclass[composite[SWAP, x_], composite[SWAP, y_]] :=
         subclass[composite[id[cart[V, V]], x], y]
```

In addition, some temporary lemmas are needed to simplify some expressions that occur in the derivation.

---

new

Lemma..

```
In[9]:= SubstTest[subclass, coflip[s], coflip[t],
         {s -> composite[SWAP, RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
          t -> composite[RIF, cross[power[x], power[y]]]}]

Out[9]= subclass[
          composite[SWAP, RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
          composite[RIF, cross[power[x], power[y]]]] ==
         subclass[composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
          composite[SWAP, RIF, cross[power[x], power[y]]]]

In[10]:= subclass[
           composite[SWAP, RIF, cross[composite[SWAP, power[x_]], composite[SWAP, power[y_]]]],
           composite[RIF, cross[power[x_], power[y_]]]] :=
          subclass[composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
           composite[SWAP, RIF, cross[power[x], power[y]]]]
```

Lemma.

```
In[11]:= SubstTest[subclass, inverse[s], inverse[t],
          {s -> composite[cross[composite[inverse[power[x]], SWAP],
              composite[inverse[power[y]], SWAP]], inverse[RIF], SWAP],
           t -> composite[cross[inverse[power[x]], inverse[power[y]]], inverse[RIF]]}]

Out[11]= subclass[composite[cross[composite[inverse[power[x]], SWAP],
             composite[inverse[power[y]], SWAP]], inverse[RIF], SWAP],
           composite[cross[inverse[power[x]], inverse[power[y]]], inverse[RIF]]] ==
          subclass[composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
           composite[SWAP, RIF, cross[power[x], power[y]]]]
```

```
In[12]:= subclass[composite[cross[composite[inverse[power[x_]], SWAP],
             composite[inverse[power[y_]], SWAP]], inverse[RIF], SWAP],
           composite[cross[inverse[power[x_]], inverse[power[y_]]], inverse[RIF]]] :=
          subclass[composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
           composite[SWAP, RIF, cross[power[x], power[y]]]]
```

Lemma.

```
In[14]:= SubstTest[subclass, inverse[s], inverse[t],
          {s -> composite[cross[inverse[power[x]], inverse[power[y]]], inverse[RIF], SWAP],
           t -> composite[cross[composite[inverse[power[x]], SWAP],
              composite[inverse[power[y]], SWAP]], inverse[RIF]]}]
```

```
Out[14]= subclass[composite[cross[inverse[power[x]], inverse[power[y]]], inverse[RIF], SWAP],
           composite[cross[composite[inverse[power[x]], SWAP],
             composite[inverse[power[y]], SWAP]], inverse[RIF]]] ==
          subclass[composite[SWAP, RIF, cross[power[x], power[y]]],
           composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]]]
```

```
In[15]:= subclass[composite[cross[inverse[power[x_]], inverse[power[y_]]], inverse[RIF], SWAP],
           composite[cross[composite[inverse[power[x_]], SWAP],
             composite[inverse[power[y_]], SWAP]], inverse[RIF]]] :=
          subclass[composite[SWAP, RIF, cross[power[x], power[y]]],
           composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]]]
```

Main Theorem.

```
In[16]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, w, case[implies[p, equal[image[s, f[w]], image[t, f[w]]]]],
           {p → commute[x, y], f[w] → cart[set[first[w]], set[second[w]]],
            s -> composite[SWAP, RIF, cross[power[x], power[y]]],
            t -> composite[RIF, cross[coflip[power[x]], coflip[power[y]]]]}]]
```

```
Out[16]= or[equal[composite[RIF, cross[composite[SWAP, power[x]], composite[SWAP, power[y]]]],
            composite[SWAP, RIF, cross[power[x], power[y]]]],
           not[equal[composite[x, y], composite[y, x]]]] == True
```

```
In[17]:= or[
           equal[composite[RIF, cross[composite[SWAP, power[x_]], composite[SWAP, power[y_]]]],
            composite[SWAP, RIF, cross[power[x_], power[y_]]]],
           not[equal[composite[x_, y_], composite[y_, x_]]]] := True
```