permutation groups

Johan G. F. Belinfante
2012 January 18

```
In[1]:= SetDirectory["l:"]; << goedel.12jan16a

        :Package Title: goedel.12jan16a               2012 January 16 at 8:30 a.m.

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

        It is now:  2012 Jan 18 at 14:1

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2012 Jan 18 at 14:14
```

summary

The set of all permutations of a set is a group under composition.

binary closure

In this section it is shown that **bij[x, x]** is binary closed under **COMPOSE**.

Lemma.

```
In[2]:= SubstTest[implies, member[t, MONOIDS], equal[domain[t], cartsq[range[t]]],
          t → composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]] // Reverse

Out[2]= or[equal[bij[x, x], image[COMPOSE, cart[bij[x, x], bij[x, x]]]], not[member[x, V]]] == True

In[3]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[4]:= SubstTest[implies, empty[t],
          equal[t, image[COMPOSE, cartsq[t]]], t → bij[x, x]] // Reverse

Out[4]= or[equal[bij[x, x], image[COMPOSE, cart[bij[x, x], bij[x, x]]]], member[x, V]] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The set of all permutations of a class **x** is binary closed under **COMPOSE**.

```
In[6]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[x, V], q -> equal[bij[x, x], image[COMPOSE, cart[bij[x, x], bij[x, x]]]]}]

Out[6]= equal[bij[x, x], image[COMPOSE, cart[bij[x, x], bij[x, x]]]] == True

In[7]:= image[COMPOSE, cart[bij[x_, x_], bij[x_, x_]]] := bij[x, x]
```

---

ids results

Lemma.

```
In[8]:= SubstTest[implies, equal[t, 0],
           equal[0, ids[composite[COMPOSE, id[cart[t, t]]]], t → bij[x, x]] // Reverse

Out[8]= or[equal[0, ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]], member[x, V]] ==
          True

In[9]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma..

```
In[10]:= SubstTest[implies, member[u, v], not[empty[v]],
            {u → id[x], v -> ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]]}] // Reverse

Out[10]= or[not[equal[0, ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]]],
            not[member[x, V]]] == True

In[11]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem.

```
In[12]:= equiv[equal[0, ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]],
           not[member[x, V]]]

Out[12]= True

In[13]:= equal[0, ids[composite[COMPOSE, id[cart[bij[x_, x_], bij[x_, x_]]]]] :=
           not[member[x, V]]
```

Lemma.

```
In[14]:= SubstTest[implies, and[empty[u], empty[v]], equal[u, v],
            {u -> ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]],
             v -> set[id[x]]}] // Reverse

Out[14]= or[equal[ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]], set[id[x]]],
            member[x, V]] == True

In[15]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[18]:= Map[member[id[setpart[x]], #] &, SubstTest[ids, semigp[t], t -> composite[COMPOSE,
           id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]]]
```

```
Out[18]= equal[e[composite[COMPOSE,
           id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]],
         id[setpart[x]]] == True
```

```
In[19]:= e[composite[COMPOSE, id[cart[bij[setpart[x_], setpart[x_]],
            bij[setpart[x_], setpart[x_]]]]] := id[setpart[x]]
```

Lemma.

```
In[20]:= SubstTest[ids, binop[t], t -> composite[COMPOSE,
           id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] // Reverse
```

```
Out[20]= ids[composite[COMPOSE,
           id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] ==
         set[id[setpart[x]]]
```

```
In[21]:= ids[composite[COMPOSE, id[cart[bij[setpart[x_], setpart[x_]],
            bij[setpart[x_], setpart[x_]]]]] := set[id[setpart[x]]]
```

Lemma.

```
In[22]:= SubstTest[implies, equal[x, setpart[t]],
           equal[ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]], set[id[x]]],
           t → x] // Reverse
```

```
Out[22]= or[equal[ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]], set[id[x]]],
           not[member[x, V]]] == True
```

```
In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. A formula for the class of all identity elements for the restriction of **COMPOSE** to the cartesian square of **bij[x, x]**.

```
In[24]:= SubstTest[and, implies[p, q], or[p, q], {p → member[x, V],
           q → equal[ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]], set[id[x]]]}]
```

```
Out[24]= equal[ids[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]], set[id[x]]] == True
```

```
In[25]:= ids[composite[COMPOSE, id[cart[bij[x_, x_], bij[x_, x_]]]]] := set[id[x]]
```

inv

Lemma.

```
In[26]:= member[oopart[setpart[t]], bij[x, x]] // AssertTest

Out[26]= member[oopart[setpart[t]], bij[x, x]] ==
          and[equal[x, domain[oopart[setpart[t]]]], equal[x, range[oopart[setpart[t]]]]]

In[27]:= member[oopart[setpart[t_]], bij[x_, x_]] :=
          and[equal[x, domain[oopart[setpart[t]]]], equal[x, range[oopart[setpart[t]]]]]
```

Lemma.

```
In[28]:= Map[member[pair[oopart[setpart[t]], inverse[oopart[setpart[t]]]], #] &, SubstTest[
           intersection, image[inverse[t], ids[t]], inverse[image[inverse[t], ids[t]]],
           t -> composite[COMPOSE, id[cartsq[bij[setpart[x], setpart[x]]]]]]]

Out[28]= member[pair[oopart[setpart[t]], inverse[oopart[setpart[t]]]], inv[composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]] ==
          and[equal[domain[oopart[setpart[t]]], setpart[x]],
           equal[range[oopart[setpart[t]]], setpart[x]]]

In[29]:= member[pair[oopart[setpart[t_]], inverse[oopart[setpart[t_]]]], inv[composite[COMPOSE,
             id[cart[bij[setpart[x_], setpart[x_]], bij[setpart[x_], setpart[x_]]]]]] :=
          and[equal[domain[oopart[setpart[t]]], setpart[x]],
           equal[range[oopart[setpart[t]]], setpart[x]]]
```

Lemma.

```
In[31]:= SubstTest[implies, equal[t, oopart[setpart[w]]],
           implies[member[t, bij[setpart[x], setpart[x]]], member[pair[t, inverse[t]],
             inv[composite[COMPOSE, id[cart[bij[setpart[x], setpart[x]],
                 bij[setpart[x], setpart[x]]]]]]], w → t] // Reverse

Out[31]= or[member[pair[t, inverse[t]], inv[composite[COMPOSE,
              id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]],
           not[FUNCTION[t]], not[FUNCTION[inverse[t]]],
           not[member[t, bij[setpart[x], setpart[x]]]]] == True

In[32]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[33]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p1, p2, p3], p4],
            not[implies[p1, p4]], {p1 -> member[t, bij[setpart[x], setpart[x]]],
             p2 → FUNCTION[t], p3 → FUNCTION[inverse[t]],
             p4 -> member[pair[t, inverse[t]], inv[composite[COMPOSE, id[cart[
                   bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]]]}]] // Reverse

Out[33]= or[member[pair[t, inverse[t]], inv[composite[COMPOSE,
              id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]],
           not[member[t, bij[setpart[x], setpart[x]]]]] == True

In[34]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[35]:= Map[composite[Id, complement[#]] &,
          dif[composite[INVERSE, id[bij[setpart[x], setpart[x]]]],
             inv[composite[COMPOSE, id[cart[bij[setpart[x], setpart[x]],
                 bij[setpart[x], setpart[x]]]]]] // complement // RelnNormality]

Out[35]= composite[intersection[INVERSE, complement[inv[composite[COMPOSE,
           id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]]],
          id[bij[setpart[x], setpart[x]]] == 0

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[37]:= SubstTest[empty, dif[u, v],
          {u -> composite[INVERSE, id[bij[setpart[x], setpart[x]]]], v -> inv[composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]]}]

Out[37]= subclass[composite[INVERSE, id[bij[setpart[x], setpart[x]]]], inv[composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]] == True

In[38]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. The restriction of **COMPOSE** to the cartesian square of **bij[x, x]** is a category.

```
In[39]:= SubstTest[implies, member[t, MONOIDS], category[t], t -> composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] // Reverse

Out[39]= category[composite[COMPOSE,
            id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] == True

In[40]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[41]:= SubstTest[FUNCTION, inv[cat[t]], t -> composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] // Reverse

Out[41]= FUNCTION[inv[composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] == True

In[42]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[43]:= SubstTest[subclass, domain[inv[cat[t]]], range[cat[t]], t -> composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]] // Reverse

Out[43]= subclass[domain[inv[composite[COMPOSE,
              id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]],
           bij[setpart[x], setpart[x]]] == True

In[44]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. Simplification rule.

```
In[45]:= equal[composite[inv[composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]],
           id[bij[setpart[x], setpart[x]]]], inv[composite[COMPOSE,
            id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]]
```

```
Out[45]= True
```

```
In[46]:= composite[inv[composite[COMPOSE,
             id[cart[bij[setpart[x_], setpart[x_]], bij[setpart[x_], setpart[x_]]]]]],
           id[bij[setpart[x_], setpart[x_]]]] := inv[composite[COMPOSE,
            id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]]
```

Lemma.

```
In[47]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u -> composite[INVERSE, id[bij[setpart[x], setpart[x]]], v -> inv[composite[COMPOSE,
               id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]}] // Reverse
```

```
Out[47]= equal[composite[INVERSE, id[bij[setpart[x], setpart[x]]], inv[composite[COMPOSE,
             id[cart[bij[setpart[x], setpart[x]], bij[setpart[x], setpart[x]]]]]] == True
```

```
In[48]:= inv[composite[COMPOSE,
             id[cart[bij[setpart[x_], setpart[x_]], bij[setpart[x_], setpart[x_]]]]] :=
           composite[INVERSE, id[bij[setpart[x], setpart[x]]]]
```

A better result can be derived, sans the **setpart** wrappers.

Lemma.

```
In[49]:= SubstTest[implies, equal[x, setpart[w]],
           equal[inv[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]],
            composite[INVERSE, id[bij[x, x]]], w → x] // Reverse
```

```
Out[49]= or[equal[composite[INVERSE, id[bij[x, x]]],
             inv[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]]], not[member[x, V]]] == True
```

```
In[50]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[51]:= SubstTest[implies, empty[t], equal[composite[INVERSE, id[t]],
             inv[composite[COMPOSE, id[cartsq[t]]]]], t → bij[x, x]] // Reverse
```

```
Out[51]= or[equal[composite[INVERSE, id[bij[x, x]]],
             inv[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]]], member[x, V]] == True
```

```
In[52]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Formula for the inversion function for the restriction of **COMPOSE** to the cartesian square of **bij[x, x]**.

```
In[53]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[x, V], q -> equal[composite[INVERSE, id[bij[x, x]]],
             inv[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]]}]

Out[53]= equal[composite[INVERSE, id[bij[x, x]]],
           inv[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]] == True

In[54]:= inv[composite[COMPOSE, id[cart[bij[x_, x_], bij[x_, x_]]]]] :=
          composite[INVERSE, id[bij[x, x]]]
```

Main Theorem. The restriction of **COMPOSE** to the cartesian square of the set of all permutations of a set **x** is a group.

```
In[55]:= SubstTest[and, member[t, MONOIDS], equal[range[t], domain[inv[t]]],
          t -> composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]]]

Out[55]= member[composite[COMPOSE, id[cart[bij[x, x], bij[x, x]]]], GROUPS] == member[x, V]

In[56]:= member[composite[COMPOSE, id[cart[bij[x_, x_], bij[x_, x_]]]], GROUPS] := member[x, V]
```