the regular representation of a group

Johan G. F. Belinfante
2013 July 1

```
In[1]:= SetDirectory["l:"]; << goedel.13jun30a

        :Package Title: goedel.13jun30a                 2013 June 30 at 2:05 p.m.

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

        It is now:  2014 Jun 30 at 11:51

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2014 Jun 30 at 12:8
```

summary

This notebook is concerned with the regular representation of a group. Some of the simpler results about regular representations also hold for monoids.

If **x** is a monoid, then so is **COMPOSE ∘ id[cartsq[range[APPLY[CURRY, x]]]]**.

```
In[2]:= or[member[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
          MONOIDS], not[member[x, MONOIDS]]]

Out[2]= True
```

The regular representation **APPLY[CURRY, x]** is a functor from **x** to this monoid.

```
In[3]:= or[functor[APPLY[CURRY, x], x, composite[COMPOSE,
           id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]], not[member[x, MONOIDS]]]

Out[3]= True
```

The neutral element of this monoid is **id[range[x]]**. When **x** is a group, the monoid **COMPOSE ∘ id[cartsq[range[-APPLY[CURRY, x]]]]** is a group. A formula for its inversion function is derived.

derivation

Lemma.

```
In[4]:= member[id[range[monoid[x]]],
          ids[composite[COMPOSE, id[cart[range[APPLY[CURRY, monoid[x]]],
              range[APPLY[CURRY, monoid[x]]]]]]]] // AssertTest

Out[4]= member[id[range[monoid[x]]],
          ids[composite[COMPOSE, id[cart[range[APPLY[CURRY, monoid[x]]],
              range[APPLY[CURRY, monoid[x]]]]]]] == not[equal[0, monoid[x]]]

In[5]:= member[id[range[monoid[x_]]],
          ids[composite[COMPOSE, id[cart[range[APPLY[CURRY, monoid[x_]]],
              range[APPLY[CURRY, monoid[x_]]]]]]] := not[equal[0, monoid[x]]]
```

Theorem. (Eliminate the **monoid** wrapper.)

```
In[6]:= SubstTest[implies, and[equal[x, monoid[t]], not[empty[x]]],
          member[id[range[x]], ids[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]], t → x] // Reverse

Out[6]= or[member[id[range[x]],
           ids[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
          not[member[x, MONOIDS]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The neutral element for the regular representation.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 → member[x, MONOIDS], p2 → member[
              composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
              MONOIDS], p3 -> member[id[range[x]], ids[
               composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
            p4 → equal[id[range[x]], e[composite[COMPOSE,
                id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]]}]] // Reverse

Out[8]= or[equal[
           e[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
           id[range[x]]], not[member[x, MONOIDS]]] == True

In[9]:= or[equal[
           e[composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]]],
           id[range[x_]]], not[member[x_, MONOIDS]]] := True
```

Theorem. The set of neutral elements for the regular representation.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 → member[x, MONOIDS], p2 → member[
             composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
             MONOIDS], p3 -> member[id[range[x]], ids[
              composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
            p4 → equal[set[id[range[x]]], ids[composite[COMPOSE,
                id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]]}]] // Reverse

Out[10]= or[equal[
           ids[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
           set[id[range[x]]], not[member[x, MONOIDS]]] == True

In[11]:= or[equal[ids[
            composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]]],
           set[id[range[x_]]], not[member[x_, MONOIDS]]] := True
```

---

the case of a group

Everything true for monoids is also true for groups.

Lemma.

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[x, GROUPS], p2 → member[x, MONOIDS], p3 -> member[composite[COMPOSE, id[
                cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]], MONOIDS]}]] // Reverse

Out[12]= or[
           member[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
            MONOIDS], not[member[x, GROUPS]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → member[x, GROUPS],
            p2 → member[x, MONOIDS], p3 -> functor[APPLY[CURRY, x], x, composite[COMPOSE,
               id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]]}]] // Reverse

Out[14]= or[functor[APPLY[CURRY, x], x,
           composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
           not[member[x, GROUPS]]] == True

In[15]:= or[functor[APPLY[CURRY, x_], x_,
           composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]]],
           not[member[x_, GROUPS]]] := True
```

Lemma.

```
In[16]:= SubstTest[or, equal[composite[inv[y], t], composite[t, inv[x]]],
           not[category[x]], not[category[y]], not[equal[domain[inv[x]], range[x]]],
           not[functor[t, x, y]], {x → gp[u], y → monoid[v]}] // Reverse

Out[16]= or[equal[composite[t, inv[gp[u]]], composite[inv[monoid[v]], t]],
           not[functor[t, gp[u], monoid[v]]]] == True

In[17]:= (% /. {t → t_, u → u_, v → v_}) /. Equal → SetDelayed
```

Theorem.

```
In[18]:= SubstTest[implies, and[equal[x, gp[u]], equal[y, monoid[v]]],
            or[equal[composite[t, inv[x]], composite[inv[y], t]], not[functor[t, x, y]]],
            {u → x, v → y}] // Reverse // MapNotNot

Out[18]= or[equal[composite[t, inv[x]], composite[inv[y], t]],
           not[functor[t, x, y]], not[member[x, GROUPS]], not[member[y, MONOIDS]]] == True

In[19]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[20]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[and[p1, p2, p3], p4], not[implies[p1, p4]], {p1 → member[x, GROUPS],
             p2 → member[composite[COMPOSE, id[cartsq[range[APPLY[CURRY, x]]]]], MONOIDS],
             p3 → functor[APPLY[CURRY, x], x,
               composite[COMPOSE, id[cartsq[range[APPLY[CURRY, x]]]]]],
             p4 -> equal[composite[APPLY[CURRY, x], inv[x]], composite[inv[composite[
                  COMPOSE, id[cartsq[range[APPLY[CURRY, x]]]]], APPLY[CURRY, x]]]}]] // Reverse

Out[20]= or[equal[composite[APPLY[CURRY, x], inv[x]], composite[
            inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
            APPLY[CURRY, x]]], not[member[x, GROUPS]]] == True

In[21]:= or[equal[composite[APPLY[CURRY, x_], inv[x_]], composite[inv[
             composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]]],
            APPLY[CURRY, x_]]], not[member[x_, GROUPS]]] := True
```

Lemma.

```
In[22]:= SubstTest[implies, equal[x, gp[t]],
            FUNCTION[inverse[APPLY[CURRY, x]]], t → x] // Reverse // MapNotNot

Out[22]= or[FUNCTION[inverse[APPLY[CURRY, x]]], not[member[x, GROUPS]]] == True

In[23]:= or[FUNCTION[inverse[APPLY[CURRY, x_]]], not[member[x_, GROUPS]]] := True
```

Lemma.

```
In[24]:= SubstTest[or, equal[composite[APPLY[CURRY, t], inv[t]], composite[
             inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, t]], range[APPLY[CURRY, t]]]]]],
             APPLY[CURRY, t]]], not[member[t, GROUPS]], t → gp[x]] // Reverse

Out[24]= or[equal[0, gp[x]], equal[composite[APPLY[CURRY, gp[x]], inv[gp[x]]],
           composite[inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, gp[x]]],
                range[APPLY[CURRY, gp[x]]]]]]], APPLY[CURRY, gp[x]]]]] == True

In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[26]:= SubstTest[and, implies[p, q], or[p, q],
          {p -> equal[0, gp[x]], q -> equal[composite[APPLY[CURRY, gp[x]], inv[gp[x]]],
             composite[inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, gp[x]]],
                  range[APPLY[CURRY, gp[x]]]]]]], APPLY[CURRY, gp[x]]]]}]

Out[26]= equal[composite[APPLY[CURRY, gp[x]], inv[gp[x]]],
           composite[inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, gp[x]]],
                range[APPLY[CURRY, gp[x]]]]]]], APPLY[CURRY, gp[x]]]] == True

In[27]:= composite[inv[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]],
           APPLY[CURRY, gp[x_]]] := composite[APPLY[CURRY, gp[x]], inv[gp[x]]]

In[28]:= SubstTest[member, composite[COMPOSE,
             id[cart[range[APPLY[CURRY, monoid[t]]], range[APPLY[CURRY, monoid[t]]]]],
           MONOIDS, t → gp[x]] // Reverse

Out[28]= member[
           composite[COMPOSE, id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]],
           MONOIDS] == not[equal[0, gp[x]]]

In[29]:= member[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]],
           MONOIDS] := not[equal[0, gp[x]]]
```

Lemma.

```
In[30]:= (implies[or[member[t, MONOIDS], empty[t]], category[t]] // NotNotTest) /. t ->
           composite[COMPOSE, id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]

Out[30]= category[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]] == True

In[31]:= category[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]] := True
```

Lemma.

```
In[32]:= SubstTest[subclass, domain[inv[cat[t]]], range[cat[t]], t -> composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]] // Reverse

Out[32]= subclass[domain[inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]],
         range[APPLY[CURRY, gp[x]]]] == True

In[33]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[34]:= equal[composite[inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]],
         id[range[APPLY[CURRY, gp[x]]]]], inv[composite[COMPOSE,
          id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]]

Out[34]= True

In[35]:= composite[inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]],
         id[range[APPLY[CURRY, gp[x_]]]] := inv[composite[COMPOSE,
          id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]
```

Theorem.

```
In[36]:= Assoc[inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]],
         APPLY[CURRY, gp[x]], inverse[APPLY[CURRY, gp[x]]]]

Out[36]= inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]] ==
         composite[APPLY[CURRY, gp[x]], inv[gp[x]], inverse[APPLY[CURRY, gp[x]]]]

In[37]:= inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]] :=
         composite[APPLY[CURRY, gp[x]], inv[gp[x]], inverse[APPLY[CURRY, gp[x]]]]
```

Theorem.

```
In[38]:= SubstTest[and, member[t, MONOIDS],
          equal[domain[inv[t]], range[t]], t -> composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]

Out[38]= member[
           composite[COMPOSE, id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]],
           GROUPS] == not[equal[0, gp[x]]]

In[39]:= member[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]],
          GROUPS] := not[equal[0, gp[x]]]
```

Corollary.

```
In[40]:= SubstTest[implies, and[equal[x, gp[t]], not[empty[x]]],
           member[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
            GROUPS], t → x] // Reverse

Out[40]= or[
           member[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
            GROUPS], not[member[x, GROUPS]]] == True

In[41]:= or[member[
            composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]],
            GROUPS], not[member[x_, GROUPS]]] := True
```

Theorem.

```
In[42]:= SubstTest[implies, equal[x, gp[t]], equal[
             inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
             composite[APPLY[CURRY, x], inv[x], inverse[APPLY[CURRY, x]]]],
            t → x] // Reverse // MapNotNot

Out[42]= or[equal[composite[APPLY[CURRY, x], inv[x], inverse[APPLY[CURRY, x]]],
            inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
           not[member[x, GROUPS]]] == True

In[43]:= or[equal[composite[APPLY[CURRY, x_], inv[x_], inverse[APPLY[CURRY, x_]]], inv[
             composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]]],
           not[member[x_, GROUPS]]] := True
```