inversion for the regular representation of a group

Johan G. F. Belinfante
2013 July 2

```
In[1]:= SetDirectory["l:"]; << goedel.13jul01a

        :Package Title: goedel.13jul01a                 2013 July 1 at 10:45 a.m.

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

        It is now:  2013 Jul 2 at 11:49

        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 2 at 12:5
```

summary

This notebook is concerned with the inversion function for the regular representation of a group. The following rewrite rule is currently available, but will be replaced.

```
In[2]:= inv[composite[COMPOSE,
          id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]]

Out[2]= composite[APPLY[CURRY, gp[x]], inv[gp[x]], inverse[APPLY[CURRY, gp[x]]]]
```

It is shown in this notebook that this inversion function is a restriction of **INVERSE**. A replacement rewrite rule is derived as well as related results.

derivation

Theorem.

```
In[3]:= SubstTest[APPLY, APPLY[CURRY, binop[t]], y, t → gp[x]] // Reverse

Out[3]= APPLY[APPLY[CURRY, gp[x]], y] == union[
          complement[image[V, intersection[range[gp[x]], set[y]]]], composite[gp[x], LEFT[y]]]

In[4]:= APPLY[APPLY[CURRY, gp[x_]], y_] := union[
          complement[image[V, intersection[range[gp[x]], set[y]]]], composite[gp[x], LEFT[y]]]
```

Corollary.

```
In[5]:= SubstTest[image, funpart[t], set[y], t → APPLY[CURRY, gp[x]]] // Reverse

Out[5]= image[APPLY[CURRY, gp[x]], set[y]] == intersection[
          image[V, intersection[range[gp[x]], set[y]]], set[composite[gp[x], LEFT[y]]]]

In[6]:= image[APPLY[CURRY, gp[x_]], set[y_]] := intersection[
          image[V, intersection[range[gp[x]], set[y]]], set[composite[gp[x], LEFT[y]]]]
```

Lemma. (Temporary rewrite rule.)

```
In[7]:= APPLY[CURRY, gp[x]] // FastReifNormality // Reverse

Out[7]= composite[INVERSE, IMAGE[cross[Id, inv[gp[x]]]],
          VERTSECT[inverse[gp[x]]], id[range[gp[x]]]] == APPLY[CURRY, gp[x]]

In[8]:= composite[INVERSE, IMAGE[cross[Id, inv[gp[x_]]]],
          VERTSECT[inverse[gp[x_]]], id[range[gp[x_]]]] := APPLY[CURRY, gp[x]]
```

Theorem. An intertwine rule for inversion functions.

```
In[9]:= Map[composite[INVERSE, #] &,
         composite[INVERSE, APPLY[CURRY, gp[x]], inv[gp[x]]] // FastReifNormality]

Out[9]= composite[APPLY[CURRY, gp[x]], inv[gp[x]]] == composite[INVERSE, APPLY[CURRY, gp[x]]]

In[10]:= composite[APPLY[CURRY, gp[x_]], inv[gp[x_]]] := composite[INVERSE, APPLY[CURRY, gp[x]]]
```

Corollary. Removing the **gp** wrapper.

```
In[11]:= SubstTest[implies, equal[x, gp[t]], equal[
             inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
             composite[INVERSE, id[range[APPLY[CURRY, x]]]], t → x] // Reverse // MapNotNot

Out[11]= or[equal[composite[INVERSE, id[range[APPLY[CURRY, x]]]],
            inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]]],
           not[member[x, GROUPS]]] == True

In[12]:= or[equal[composite[INVERSE, id[range[APPLY[CURRY, x_]]]], inv[
             composite[COMPOSE, id[cart[range[APPLY[CURRY, x_]], range[APPLY[CURRY, x_]]]]]]],
           not[member[x_, GROUPS]]] := True
```

replacement rule

The old rewrite rule for the inversion function for the regular representation will now be removed, preparatory to deriving a replacement rule.

```
In[13]:= inv[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]]] =.
```

Lemma. Reintroduce the **gp** wrapper.

```
In[14]:= SubstTest[or, equal[composite[INVERSE, id[range[APPLY[CURRY, t]]]],
           inv[composite[COMPOSE, id[cart[range[APPLY[CURRY, t]], range[APPLY[CURRY, t]]]]]],
          not[member[t, GROUPS]], t → gp[x]] // Reverse

Out[14]= or[equal[0, gp[x]],
          equal[composite[INVERSE, id[range[APPLY[CURRY, gp[x]]]], inv[composite[COMPOSE,
            id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]]]] == True

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

Theorem. Replacement rule.

```
In[16]:= SubstTest[and, or[p, q], implies[p, q], {p -> equal[0, gp[x]],
           q -> equal[composite[INVERSE, id[range[APPLY[CURRY, gp[x]]]], inv[composite[COMPOSE,
             id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]]]}]

Out[16]= equal[composite[INVERSE, id[range[APPLY[CURRY, gp[x]]]], inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]]]] == True

In[17]:= inv[composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x_]]], range[APPLY[CURRY, gp[x_]]]]]] :=
          composite[INVERSE, id[range[APPLY[CURRY, gp[x]]]]]
```

---

related results of interest

Corollary. The function **INVERSE** commutes with **id[range[APPLY[CURRY, gp[x]]]]**.

```
In[18]:= SubstTest[inverse, inv[t], t -> composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]] // Reverse

Out[18]= composite[id[range[APPLY[CURRY, gp[x]]]], INVERSE] ==
          composite[INVERSE, id[range[APPLY[CURRY, gp[x]]]]]

In[19]:= composite[id[range[APPLY[CURRY, gp[x_]]]], INVERSE] :=
          composite[INVERSE, id[range[APPLY[CURRY, gp[x]]]]]
```

Theorem.

```
In[20]:= SubstTest[U, range[APPLY[CURRY, binop[t]]], t → gp[x]] // Reverse

Out[20]= U[range[APPLY[CURRY, gp[x]]]] == domain[gp[x]]

In[21]:= U[range[APPLY[CURRY, gp[x_]]]] := domain[gp[x]]
```

Theorem.

```
In[22]:= ImageComp[id[range[APPLY[CURRY, gp[x]]]], INVERSE, V]

Out[22]= image[INVERSE, range[APPLY[CURRY, gp[x]]]] == range[APPLY[CURRY, gp[x]]]
```

```
In[23]:= image[INVERSE, range[APPLY[CURRY, gp[x_]]]] := range[APPLY[CURRY, gp[x]]]
```

special case of INTADD

In this section the special case of the group of integers under addition is considered.

Theorem.

```
In[24]:= SubstTest[member, composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]],
         GROUPS, x → INTADD] // Reverse

Out[24]= member[composite[COMPOSE, id[cart[range[INTPLUS], range[INTPLUS]]]], GROUPS] == True

In[25]:= member[composite[COMPOSE, id[cart[range[INTPLUS], range[INTPLUS]]]], GROUPS] := True
```

Theorem.

```
In[26]:= SubstTest[implies, member[x, MONOIDS], equal[
           e[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]]],
           id[range[x]]], x → INTADD] // Reverse

Out[26]= equal[e[composite[COMPOSE, id[cart[range[INTPLUS], range[INTPLUS]]]]], id[Z]] == True

In[27]:= e[composite[COMPOSE, id[cart[range[INTPLUS], range[INTPLUS]]]]] := id[Z]
```

Theorem.

```
In[28]:= SubstTest[inv, composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]], x →
           INTADD] // Reverse

Out[28]= inv[composite[COMPOSE, id[cart[range[INTPLUS], range[INTPLUS]]]]] ==
          composite[INVERSE, id[range[INTPLUS]]]

In[29]:= inv[composite[COMPOSE, id[cart[range[INTPLUS], range[INTPLUS]]]]] :=
          composite[INVERSE, id[range[INTPLUS]]]
```

Theorem.

```
In[30]:= SubstTest[composite, id[range[APPLY[CURRY, gp[x]]]], INVERSE, x → INTADD] // Reverse

Out[30]= composite[id[range[INTPLUS]], INVERSE] == composite[INVERSE, id[range[INTPLUS]]]

In[31]:= composite[id[range[INTPLUS]], INVERSE] := composite[INVERSE, id[range[INTPLUS]]]
```

Theorem.

```
In[32]:= SubstTest[U, range[APPLY[CURRY, gp[x]]], x → INTADD] // Reverse

Out[32]= U[range[INTPLUS]] == cart[Z, Z]
```

```
In[33]:= U[range[INTPLUS]] := cart[Z, Z]
```

Theorem.

```
In[34]:= SubstTest[image, INVERSE, range[APPLY[CURRY, gp[x]]], x → INTADD] // Reverse
Out[34]= image[INVERSE, range[INTPLUS]] == range[INTPLUS]

In[35]:= image[INVERSE, range[INTPLUS]] := range[INTPLUS]
```

Theorem. Intertwine rule for inversions.

```
In[37]:= SubstTest[composite, APPLY[CURRY, gp[x]], inv[gp[x]], x → INTADD] // Reverse
Out[37]= composite[INTPLUS, INVERSE] == composite[INVERSE, INTPLUS]

In[38]:= composite[INTPLUS, INVERSE] := composite[INVERSE, INTPLUS]
```

special case of RATADD

In this section the special case of the group of rational numbers under addition is considered.

Lemma.

```
In[39]:= Map[not, SubstTest[member, gp[x], GROUPS, x → RATADD]]
Out[39]= equal[0, RATADD] == False

In[40]:= equal[0, RATADD] := False
```

Theorem.

```
In[41]:= SubstTest[member, composite[COMPOSE,
            id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]]],
           GROUPS, x → RATADD] // Reverse
Out[41]= member[composite[COMPOSE, id[cart[range[RATPLUS], range[RATPLUS]]]], GROUPS] == True

In[42]:= member[composite[COMPOSE, id[cart[range[RATPLUS], range[RATPLUS]]]], GROUPS] := True
```

Theorem.

```
In[43]:= SubstTest[implies, member[x, MONOIDS], equal[
            e[composite[COMPOSE, id[cart[range[APPLY[CURRY, x]], range[APPLY[CURRY, x]]]]],
            id[range[x]]], x → RATADD] // Reverse
Out[43]= equal[e[composite[COMPOSE, id[cart[range[RATPLUS], range[RATPLUS]]]]], id[RATS]] == True

In[44]:= e[composite[COMPOSE, id[cart[range[RATPLUS], range[RATPLUS]]]]] := id[RATS]
```

Theorem.

```
In[45]:= SubstTest[inv, composite[COMPOSE,
           id[cart[range[APPLY[CURRY, gp[x]]], range[APPLY[CURRY, gp[x]]]]], x →
           RATADD] // Reverse

Out[45]= inv[composite[COMPOSE, id[cart[range[RATPLUS], range[RATPLUS]]]]] ==
          composite[INVERSE, id[range[RATPLUS]]]

In[46]:= inv[composite[COMPOSE, id[cart[range[RATPLUS], range[RATPLUS]]]]] :=
          composite[INVERSE, id[range[RATPLUS]]]
```

Theorem.

```
In[47]:= SubstTest[composite, id[range[APPLY[CURRY, gp[x]]], INVERSE, x → RATADD] // Reverse

Out[47]= composite[id[range[RATPLUS]], INVERSE] == composite[INVERSE, id[range[RATPLUS]]]

In[48]:= composite[id[range[RATPLUS]], INVERSE] := composite[INVERSE, id[range[RATPLUS]]]
```

Theorem.

```
In[49]:= SubstTest[U, range[APPLY[CURRY, gp[x]]], x → RATADD] // Reverse

Out[49]= U[range[RATPLUS]] == cart[RATS, RATS]

In[50]:= U[range[RATPLUS]] := cart[RATS, RATS]
```

Theorem.

```
In[51]:= SubstTest[image, INVERSE, range[APPLY[CURRY, gp[x]]], x → RATADD] // Reverse

Out[51]= image[INVERSE, range[RATPLUS]] == range[RATPLUS]

In[52]:= image[INVERSE, range[RATPLUS]] := range[RATPLUS]
```

Theorem. Intertwine rule for inversions.

```
In[53]:= SubstTest[composite, APPLY[CURRY, gp[x]], inv[gp[x]], x → RATADD] // Reverse

Out[53]= composite[RATPLUS, inv[RATADD]] == composite[INVERSE, RATPLUS]

In[54]:= composite[RATPLUS, inv[RATADD]] := composite[INVERSE, RATPLUS]
```