inverse elements in a group, part 1

Johan G. F. Belinfante
2009 February 1

```
In[1]:= << l:goedel.09jan31a;<< l:tools.m

        :Package Title: goedel.09jan31a            2009 January 31 at 3:50 p.m.

        It is now:  2009 Feb 1 at 3:21

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 January 29

        weightlimit = 40
```

summary

Following Kurosh, a group is defined in the **GOEDEL** program to be a nonempty associative quasigroup operation. Because of this, the usual axioms for a group become theorems that require proof. In this notebook, the axioms about inverse elements are derived, as well as a few basic theorems about inverses. The terminology used in this notebook is slightly nonstandard. Strictly speaking, the statement **x ∈ GROUPS** means that **x** is a **group multiplication law**, that is, a particular type of binary operation on a set **range[x]**.

```
In[2]:= implies[member[x, GROUPS], member[x, map[cart[range[x], range[x]], range[x]]]]

Out[2]= True
```

For short, it will be stated that **x** is a **group**, albeit in common parlance that term applies instead to the underlying set **range[x]**. The only real advantage of putting the focus squarely on the mapping **x** rather than the set **range[x]** is to avoid the need for two separate variables when talking about a group. In the informal comments accompanying the theorems, the composite of two elements **u** and **v** for a binary operation **x** will be denoted **u · v**, which must be translated in the **GOEDEL** program as **APPLY[x, PAIR[u, v]]** .

domain rules

The domain of a group is the cartesian square of its range. Two corollaries of this fact are derived in this section. The first of these is used in a later section, but the second one is included only for the sake of completeness.

Theorem. The domain of the domain of a group is the same as its range.

```
In[3]:= Map[not, SubstTest[and, implies[p2, p3], not[implies[p1, p3]],
           {p1 -> member[x, GROUPS], p2 -> equal[domain[x], cartsq[range[x]]],
            p3 -> equal[domain[domain[x]], range[x]]}]] // Reverse

Out[3]= or[equal[domain[domain[x]], range[x]], not[member[x, GROUPS]]] == True

In[4]:= or[equal[domain[domain[x_]], range[x_]], not[member[x_, GROUPS]]] := True
```

Similarly, the following holds:

```
In[5]:= Map[not, SubstTest[and, implies[p2, p3], not[implies[p1, p3]],
           {p1 -> member[x, GROUPS], p2 -> equal[domain[x], cartsq[range[x]]],
            p3 -> equal[range[domain[x]], range[x]]}]] // Reverse

Out[5]= or[equal[range[x], range[domain[x]]], not[member[x, GROUPS]]] == True

In[6]:= or[equal[range[x_], range[domain[x_]]], not[member[x_, GROUPS]]] := True
```

---

groups and categories

The **inverse pair relation** is a constructor **inv[x]** that was introduced in the study of category theory. (Actually **inv[x]** was defined for an arbitrary class **x**, not just for categories.) Morphisms **u** and **v** form an inverse pair if their composites in both orders are identity elements: $\mathbf{u \cdot v \in ids[x]}$ and $\mathbf{v \cdot u \in ids[x]}$. Of course, all theorems about inverse pairs of morphisms in categories also apply to groups since a group is after all a very special type of category:

```
In[7]:= Map[not, SubstTest[and, implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[x, GROUPS], p2 → member[x, MONOIDS], p3 → category[x]}]] // Reverse

Out[7]= or[category[x], not[member[x, GROUPS]]] == True

In[8]:= or[category[x_], not[member[x_, GROUPS]]] := True
```

For example, the fact that **inv[x]** is a function for groups can be deduced as a corollary of a corresponding theorem in category theory, which amounts to the statement that isomorphisms in a category have unique two-sided inverses.

```
In[9]:= Map[not, SubstTest[and, implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[x, GROUPS], p2 → category[x], p3 → FUNCTION[inv[x]]}]] // Reverse

Out[9]= or[FUNCTION[inv[x]], not[member[x, GROUPS]]] == True

In[10]:= or[FUNCTION[inv[x_]], not[member[x_, GROUPS]]] := True
```

Categories may have many identity morphisms, and categories need not be sets. A small category with a single identity morphism is called a **monoid**. In particular, groups are monoids, and it is often more expeditious to obtain theorems about groups by specializing results about monoids rather than going all the way back to category theory. If the needed monoid facts are not already available, it may sometimes pay in the long run to first make them available for monoids by deducing them from results in category theory, and then further specializing them to obtain theorems about groups.

Lemma. If **x** is a category and if **pair[u, v] ∈ inv[x]**, then **u · v ∈ ids[x]**. Here **ids[x]** is the set of identity morphisms for the category **x**. The derivation just amounts to eliminating the **cat** wrapper from a known theorem about categories.

```
In[11]:= SubstTest[implies, and[equal[x, cat[t]], member[pair[u, v], inv[x]]],
           member[APPLY[x, PAIR[u, v]], ids[x]], t → x] // Reverse

Out[11]= or[member[APPLY[x, PAIR[u, v]], ids[x]],
           not[category[x]], not[member[pair[u, v], inv[x]]]] == True

In[12]:= or[member[APPLY[x_, PAIR[u_, v_]], ids[x_]],
           not[category[x_]], not[member[pair[u_, v_], inv[x_]]]] := True
```

Theorem. If **x** is a monoid and if **pair[u,v] ∈ inv[x]**, then **u · v = e[x]**.

```
In[13]:= Map[not, SubstTest[and, implies[and[p2, p3], p5], not[implies[and[p1, p2], p6]],
            {p1 → member[x, MONOIDS], p2 → member[pair[u, v], inv[x]], p3 → category[x],
             p4 → equal[ids[x], set[e[x]]], p5 → member[APPLY[x, PAIR[u, v]], ids[x]],
             p6 → equal[e[x], APPLY[x, PAIR[u, v]]]}]] // Reverse

Out[13]= or[equal[APPLY[x, PAIR[u, v]], e[x]],
           not[member[x, MONOIDS]], not[member[pair[u, v], inv[x]]]] == True

In[14]:= or[equal[APPLY[x_, PAIR[u_, v_]], e[x_]],
           not[member[x_, MONOIDS]], not[member[pair[u_, v_], inv[x_]]]] := True
```

Theorem. Two-sided invertibility follows from left and right invertibility for monoids, and hence in particular for groups. This result is improved upon in the final section of this notebook, but for now this will suffice.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            not[implies[p1, p3]], {p1 → member[x, GROUPS], p2 → member[x, MONOIDS],
             p3 -> equal[intersection[domain[image[inverse[x], set[e[x]]]],
                range[image[inverse[x], set[e[x]]]], domain[inv[x]]]}]] // Reverse

Out[15]= or[equal[domain[inv[x]], intersection[domain[image[inverse[x], set[e[x]]]],
             range[image[inverse[x], set[e[x]]]]], not[member[x, GROUPS]]] == True

In[16]:= (% /. x → x_) /. Equal → SetDelayed
```

quasigp results

Groups by definition are quasigroups, and hence one can also obtain facts about groups by specializing theorems about quasigroups. Since the axioms of quasigroups are rotation-invariant, one can do ‘proofs by rotation’. Here is a simple example:

Theorem. A result about quasigroups deduced by using rotation.

```
In[17]:= SubstTest[member, APPLY[quasigp[t], PAIR[u, v]],
           range[quasigp[t]], t → rotate[quasigp[x]]] // Reverse

Out[17]= member[APPLY[rotate[quasigp[x]], PAIR[u, v]], range[quasigp[x]]] ==
          and[member[u, range[quasigp[x]]], member[v, range[quasigp[x]]]]

In[18]:= member[APPLY[rotate[quasigp[x_]], PAIR[u_, v_]], range[quasigp[x_]]] :=
          and[member[u, range[quasigp[x]]], member[v, range[quasigp[x]]]]
```

Corollary. Removing the **quasigp** wrapper yields this restatement of the theorem.

```
In[19]:= SubstTest[implies,
           and[equal[x, quasigp[t]], and[member[u, range[x]], member[v, range[x]]]],
           member[APPLY[rotate[x], PAIR[u, v]], range[x]], t → x] // Reverse

Out[19]= or[member[APPLY[rotate[x], PAIR[u, v]], range[x]], not[member[u, range[x]]],
           not[member[v, range[x]]], not[member[x, QUASIGPS]]] == True

In[20]:= or[member[APPLY[rotate[x_], PAIR[u_, v_]], range[x_]], not[member[u_, range[x_]]],
           not[member[v_, range[x_]]], not[member[x_, QUASIGPS]]] := True
```

---

invertibility in a group

Lemma. A lower bound for the set **domain[inv[x]]** of invertible elements in a group **x** is obtained by using the fact that groups are quasigroups. Comment. In this derivation, as in many other ones in this notebook, execution time is minimized by supplying only a very brief sketch of the complete proof. Rewrite rules supply the omitted steps.

```
In[21]:= Map[not, SubstTest[and, implies[and[p2, p3, p4, p5], p6], p0, p1, p2, not[p6],
             {p0 -> equal[v, APPLY[into[x], PAIR[u, e[x]]]], p1 → member[x, GROUPS],
              p2 → member[u, range[x]], p3 → equal[x, quasigp[t]], p4 -> member[e[x], range[x]],
              p5 -> equal[domain[inv[x]], intersection[domain[image[inverse[x], set[e[x]]]],
                 range[image[inverse[x], set[e[x]]]]]],
              p6 → member[u, domain[inv[x]]]}] // Reverse] /.
           {t → x, v -> APPLY[into[x], PAIR[u, e[x]]]}

Out[21]= or[member[u, domain[inv[x]]], not[member[u, range[x]]], not[member[x, GROUPS]]] == True

In[22]:= or[member[u_, domain[inv[x_]]],
           not[member[u_, range[x_]]], not[member[x_, GROUPS]]] := True
```

Lemma. The variable **u** is eliminated using **class** rules.

```
In[23]:= Map[equal[V, #] &,
          SubstTest[class, u, or[member[u, v], not[member[u, range[x]]], not[member[x, w]]],
           {v → domain[inv[x]], w → GROUPS}]]

Out[23]= or[not[member[x, GROUPS]], subclass[range[x], domain[inv[x]]]] == True

In[24]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. The reverse inclusion also holds.

```
In[25]:= SubstTest[implies, and[subclass[u, v], implies[p, equal[v, w]]],
           implies[p, subclass[u, w]], {p → member[x, GROUPS],
            u -> domain[inv[x]], v -> domain[domain[x]], w -> range[x]}] // Reverse

Out[25]= or[not[member[x, GROUPS]], subclass[domain[inv[x]], range[x]]] == True

In[26]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Since every element of a group is invertible, the set of invertible elements of a group coincides with the range of the group.

```
In[27]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
          {p → member[x, GROUPS], u -> domain[inv[x]], v -> range[x]}]

Out[27]= or[equal[domain[inv[x]], range[x]], not[member[x, GROUPS]]] == True

In[28]:= or[equal[domain[inv[x_]], range[x_]], not[member[x_, GROUPS]]] := True
```

APPLY rules

The axioms for a group given in most text books include two formulas about multiplying an element with its inverse. These formulas are derived in this section.

Lemma. Applying a function to an element of its domain yields an element of its range. Since **inv[x]** is a symmetric relation, its domain and range are equal. This lemma helps to cope with a rewrite rule that automatically transforms the expression **range[inv[x]]** to **domain[inv[x]]**.

```
In[29]:= SubstTest[implies, and[FUNCTION[t], member[u, domain[t]]],
           member[APPLY[t, u], range[t]], t → inv[x]] // Reverse

Out[29]= or[member[APPLY[inv[x], u], domain[inv[x]]],
           not[FUNCTION[inv[x]]], not[member[u, domain[inv[x]]]]] == True

In[30]:= or[member[APPLY[inv[x_], u_], domain[inv[x_]]],
           not[FUNCTION[inv[x_]]], not[member[u_, domain[inv[x_]]]]] := True
```

Theorem. If **x** is a group and $\mathbf{u \in range[x]}$, then $\mathbf{u^{-1} \in range[x]}$.

```
In[31]:= Map[not, SubstTest[and, implies[p1, p4],
            implies[and[p1, p2], p3], implies[p1, p6], not[implies[and[p1, p2], p7]],
            {p1 → member[x, GROUPS], p2 → member[u, range[x]], p3 → member[u, domain[inv[x]]],
             p4 → FUNCTION[inv[x]], p5 → member[pair[u, APPLY[inv[x], u]], inv[x]],
             p6 → equal[domain[inv[x]], range[x]],
             p7 → member[APPLY[inv[x], u], range[x]]}]] // Reverse

Out[31]= or[member[APPLY[inv[x], u], range[x]],
           not[member[u, range[x]]], not[member[x, GROUPS]]] == True
```

```
In[32]:= or[member[APPLY[inv[x_], u_], range[x_]],
            not[member[u_, range[x_]]], not[member[x_, GROUPS]]] := True
```

Theorem. If **x** is a group and **u ∈ range[x]**, then $\mathbf{u \cdot u^{-1} = e[x]}$. (This is usually taken as an axiom in group theory.)

```
In[33]:= Map[not, SubstTest[and, implies[p1, p4], implies[and[p1, p2], p3],
              implies[and[p5, p8], p9], not[implies[and[p1, p2], p9]], {p1 → member[x, GROUPS],
               p2 → member[u, range[x]], p3 → member[u, domain[inv[x]]], p4 → FUNCTION[inv[x]],
               p5 → member[pair[u, APPLY[inv[x], u]], inv[x]], p8 → member[x, MONOIDS],
               p9 → equal[e[x], APPLY[x, PAIR[u, APPLY[inv[x], u]]]]}]] // Reverse

Out[33]= or[equal[APPLY[x, PAIR[u, APPLY[inv[x], u]]], e[x]],
            not[member[u, range[x]]], not[member[x, GROUPS]]] == True

In[34]:= or[equal[APPLY[x_, PAIR[u_, APPLY[inv[x_], u_]]], e[x_]],
            not[member[u_, range[x_]]], not[member[x_, GROUPS]]] := True
```

Lemma. The inverse pair relation **inv[x]** is symmetric for any class **x**.

```
In[35]:= Map[implies[member[pair[v, u], inv[x]], #] &,
            SubstTest[member, pair[v, u], inverse[t], t → inv[x]]] // MapNotNot

Out[35]= or[member[pair[u, v], inv[x]], not[member[pair[v, u], inv[x]]]] == True

In[36]:= or[member[pair[u_, v_], inv[x_]], not[member[pair[v_, u_], inv[x_]]]] := True
```

Theorem. If **x** is a group and **u ∈ range[x]**, then $\mathbf{u^{-1} \cdot u = e[x]}$.

```
In[37]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4], implies[p5, p6],
              implies[p1, p7], implies[and[p6, p7], p8], not[implies[and[p1, p2], p8]],
              {p1 → member[x, GROUPS], p2 → member[u, range[x]], p3 → member[u, domain[inv[x]]],
               p4 → FUNCTION[inv[x]], p5 → member[pair[u, APPLY[inv[x], u]], inv[x]],
               p6 → member[pair[APPLY[inv[x], u], u], inv[x]], p7 → member[x, MONOIDS],
               p8 → equal[e[x], APPLY[x, PAIR[APPLY[inv[x], u], u]]]}]] // Reverse

Out[37]= or[equal[APPLY[x, PAIR[APPLY[inv[x], u], u]], e[x]],
            not[member[u, range[x]]], not[member[x, GROUPS]]] == True

In[38]:= or[equal[APPLY[x_, PAIR[APPLY[inv[x_], u_], u_]], e[x_]],
            not[member[u_, range[x_]]], not[member[x_, GROUPS]]] := True
```

involution property

Lemma.

```
In[39]:= (SubstTest[implies, and[equal[t, oopart[w]], member[u, domain[t]],
              equal[v, APPLY[t, u]]], equal[u, APPLY[inverse[t], v]], w → t] /.
            {t → inv[x], v -> APPLY[inv[x], u]}) // Reverse

Out[39]= or[equal[u, APPLY[inv[x], APPLY[inv[x], u]]],
            not[FUNCTION[inv[x]]], not[member[u, domain[inv[x]]]]] == True
```

```
In[40]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Corollary. The statement of the lemma simplifies when **x** is a category.

```
In[41]:= SubstTest[implies, and[member[u, domain[inv[t]]], FUNCTION[inv[t]]],
           equal[u, APPLY[inv[t], APPLY[inv[t], u]]], t → cat[x]] // Reverse

Out[41]= or[equal[u, APPLY[inv[cat[x]], APPLY[inv[cat[x]], u]]],
           not[member[u, domain[inv[cat[x]]]]]] == True

In[42]:= or[equal[u_, APPLY[inv[cat[x_]], APPLY[inv[cat[x_]], u_]]],
           not[member[u_, domain[inv[cat[x_]]]]]] := True
```

Theorem. If **x** is a group and $\mathbf{u} \in \mathbf{range[x]}$, then $(\mathbf{u}^{-1})^{-1} = \mathbf{u}$.

```
In[43]:= Map[not, SubstTest[and, implies[and[p2, p3, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 → member[x, GROUPS], p2 → member[u, range[x]],
             p3 → equal[range[x], domain[inv[x]]], p4 → FUNCTION[inv[x]],
             p5 -> equal[u, APPLY[inv[x], APPLY[inv[x], u]]]}]] // Reverse

Out[43]= or[equal[u, APPLY[inv[x], APPLY[inv[x], u]]],
           not[member[u, range[x]]], not[member[x, GROUPS]]] == True

In[44]:= or[equal[u_, APPLY[inv[x_], APPLY[inv[x_], u_]]],
           not[member[u_, range[x_]]], not[member[x_, GROUPS]]] := True
```

---

the inverse of e[x]

Theorem. If **x** is a group, then the identity element **e[x]** is its own inverse.

```
In[45]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[and[p4, p5], p6], not[implies[p1, p6]], {p1 → member[x, GROUPS],
             p2 → member[e[x], range[x]], p3 → member[APPLY[inv[x], e[x]], range[x]],
             p4 → equal[APPLY[x, PAIR[e[x], APPLY[inv[x], e[x]]]], e[x]],
             p5 -> equal[APPLY[x, PAIR[e[x], APPLY[inv[x], e[x]]]], APPLY[inv[x], e[x]]],
             p6 -> equal[e[x], APPLY[inv[x], e[x]]]}]] // Reverse

Out[45]= or[equal[APPLY[inv[x], e[x]], e[x]], not[member[x, GROUPS]]] == True

In[46]:= or[equal[APPLY[inv[x_], e[x_]], e[x_]], not[member[x_, GROUPS]]] := True
```

Corollary. The identity element **e[x]** in a group **x** is a fixed point of the function **inv[x]**.

```
In[47]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4], p5],
            not[implies[p1, p5]], {p1 -> member[x, GROUPS],
             p2 -> equal[APPLY[inv[x], e[x]], e[x]], p3 → FUNCTION[inv[x]],
             p4 → member[e[x], range[x]], p5 → member[e[x], fix[inv[x]]]}]] // Reverse

Out[47]= or[member[e[x], fix[inv[x]]], not[member[x, GROUPS]]] == True

In[48]:= or[member[e[x_], fix[inv[x_]]], not[member[x_, GROUPS]]] := True
```

an explicit formula for inv[x]

In this section it is shown that if **x** is a group, then any one-sided inverse of an element **u ∈ x** is a two-sided inverse. As a consequence, a simple formula can be derived for **inv[x]** which just amounts to the statement that elements **u** and **v** form an inverse pair in a group if **u · v = e[x]**. Here again, as in the rest of this notebook, execution time can be drastically reduced, often by a factor of ten or more, by simply omitting proof steps and by bundling literals. The derivations that are presented here are the best versions found after extensive experimentation. (All execution times quoted are for a particular computer with a 3.4 GHz Pentium 4 processor running Microsoft Windows XP.)

Theorem. If **x** is a group and **u · v = e[x]**, then **v = u^{-1}**. (This derivation still takes 30 seconds even after extensive use of bundling and judicious omission of steps.)

```
In[49]:= Map[not, SubstTest[and, p1, implies[and[p1, p3, p6], p7],
            implies[and[p1, p4], p8], not[and[p1, p4, p7, p8]],
            {p1 → and[member[x, GROUPS], equal[e[x], APPLY[x, PAIR[u, v]]], not[
                equal[APPLY[inv[x], u], v]]], p3 → equal[APPLY[x, PAIR[APPLY[inv[x], u], e[x]]],
              APPLY[x, PAIR[APPLY[x, PAIR[APPLY[inv[x], u], u]], v]]],
             p4 → and[member[u, range[x]], member[v, range[x]]],
             p6 → member[APPLY[inv[x], u], range[x]],
             p7 →
              equal[APPLY[inv[x], u], APPLY[x, PAIR[APPLY[x, PAIR[APPLY[inv[x], u], u]], v]]],
             p8 → equal[e[x], APPLY[x, PAIR[APPLY[inv[x], u], u]]]}] // Reverse

Out[49]= or[equal[v, APPLY[inv[x], u]],
           not[equal[APPLY[x, PAIR[u, v]], e[x]]], not[member[x, GROUPS]]] == True

In[50]:= or[equal[v_, APPLY[inv[x_], u_]],
           not[equal[APPLY[x_, PAIR[u_, v_]], e[x_]]], not[member[x_, GROUPS]]] := True
```

Theorem. If **x** is a group, and if **u · v = e[x]**, then **v · u = e[x]**.

```
In[51]:= Map[not, SubstTest[and, implies[and[p1, p3, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 → member[x, GROUPS],
             p2 -> equal[APPLY[x, PAIR[u, v]], e[x]], p3 -> equal[v, APPLY[inv[x], u]],
             p4 → member[u, range[x]], p5 → equal[APPLY[x, PAIR[v, u]], e[x]]}] // Reverse

Out[51]= or[equal[APPLY[x, PAIR[v, u]], e[x]],
           not[equal[APPLY[x, PAIR[u, v]], e[x]]], not[member[x, GROUPS]]] == True

In[52]:= or[equal[APPLY[x_, PAIR[v_, u_]], e[x_]],
           not[equal[APPLY[x_, PAIR[u_, v_]], e[x_]]], not[member[x, GROUPS]]] := True
```

Lemma. If **x** is a category and if **u · v ∈ ids[x]** and **v · u ∈ ids[x]**, then **pair[u, v] ∈ inv[x]**. This fact is obtained here by simply removing the **cat** wrapper from a known theorem about categories.

```
In[53]:= SubstTest[implies, and[equal[x, cat[t]], member[APPLY[x, PAIR[u, v]], ids[x]],
           member[APPLY[x, PAIR[v, u]], ids[x]]], member[pair[u, v], inv[x]], t → x] // Reverse

Out[53]= or[member[pair[u, v], inv[x]],
           not[category[x]], not[member[APPLY[x, PAIR[u, v]], ids[x]]],
           not[member[APPLY[x, PAIR[v, u]], ids[x]]]] == True

In[54]:= or[member[pair[u_, v_], inv[x_]],
           not[category[x_]], not[member[APPLY[x_, PAIR[u_, v_]], ids[x_]]],
           not[member[APPLY[x_, PAIR[v_, u_]], ids[x_]]]] := True
```

Theorem. If **x** is a group and if **u · v = e[x]**, then **pair[u, v] ∈ inv[x]**.

```
In[55]:= Map[not, SubstTest[and, implies[p2, p4], implies[and[p3, p4], p5], not[implies[p1, p5]],
           {p1 → and[member[x, GROUPS], equal[APPLY[x, PAIR[u, v]], e[x]]],
            p2 → and[member[e[x], ids[x]], equal[APPLY[x, PAIR[v, u]], e[x]]],
            p3 → and[category[x], member[APPLY[x, PAIR[u, v]], ids[x]]], p4 ->
             member[APPLY[x, PAIR[v, u]], ids[x]], p5 → member[pair[u, v], inv[x]]}]] // Reverse

Out[55]= or[member[pair[u, v], inv[x]],
           not[equal[APPLY[x, PAIR[u, v]], e[x]]], not[member[x, GROUPS]]] == True

In[56]:= or[member[pair[u_, v_], inv[x_]],
           not[equal[APPLY[x_, PAIR[u_, v_]], e[x_]]], not[member[x_, GROUPS]]] := True
```

Corollary. The **class** rules are used to eliminate the variables **u** and **v**. Technical notes: A **funpart** wrapper was introduced to help the **class** rules to deal with the **APPLY** constructor, and then this wrapper was traded off for a **cat** wrapper to achieve some further simplifications.

```
In[57]:= Map[or[subclass[image[inverse[cat[x]], set[e[cat[x]]]], inv[cat[x]]], #] &,
          Map[empty[composite[Id, complement[#]]] &,
            SubstTest[class, pair[u, v], or[member[pair[u, v], t],
              not[equal[APPLY[funpart[y], PAIR[u, v]], w]], not[member[funpart[y], z]]],
             {t → inv[funpart[y]], w → e[funpart[y]], z -> GROUPS}]] /. y → cat[x]]

Out[57]= or[not[member[cat[x], GROUPS]],
           subclass[image[inverse[cat[x]], set[e[cat[x]]]], inv[cat[x]]]] == True

In[58]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. An explicit formula for **inv[x]** for the case that **x** is a group. Technical notes: This derivation still takes about 18 seconds, even after judicious bundling of literals and omission of proof steps that can be supplied by rewrite rules. To help control the action of rewrite rules, two auxiliary variables **t** and **u** are introduced, which are eliminated from the final statement of the theorem.

In[59]:=
```
Map[not,
  SubstTest[and, p1, implies[and[p1, p2], p3], implies[p1, p5], not[and[p1, p4, p6]],
    {p1 → and[equal[t, x], equal[u, image[inverse[x], set[e[x]]]], member[x, GROUPS],
       not[equal[u, inv[x]]]], p2 → category[x], p3 → equal[cat[t], x], p4 ->
      subclass[u, inv[x]], p5 → equal[ids[x], set[e[x]]], p6 → subclass[inv[x], u]}] /.
   {t → x, u -> image[inverse[x], set[e[x]]]}] // Reverse
```

Out[59]=
```
or[equal[image[inverse[x], set[e[x]]], inv[x]], not[member[x, GROUPS]]] == True
```

In[60]:=
```
or[equal[image[inverse[x_], set[e[x_]]], inv[x_]], not[member[x_, GROUPS]]] := True
```