inverse pair relation for submonoids

Johan G. F. Belinfante
2011 May 23

```
In[1]:= SetDirectory["l:"]; << goedel.11may22a

        :Package Title: goedel.11may22a              2011 May 23 at 6:20 a.m.

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

        It is now:  2011 May 23 at 18:21

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2011 May 23 at 18:32
```

summary

If **x** is a submonoid of a group **y**, then the inverse pair relation for **x** is the intersection of that for **y** with the domain of **x**. If the range of a submonoid **x** of a group **y** is subvariant under the inverse pair relation for the group, then the submonoid is a subgroup.

derivation

Theorem.

```
In[2]:= Map[not, SubstTest[and, implies[p1, p4], implies[and[p1, p2, p3], p5],
           implies[and[p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
           {p1 -> member[x, MONOIDS], p2 -> member[y, GROUPS], p3 -> subclass[x, y],
            p4 → equal[domain[x], cartsq[range[x]]], p5 -> equal[x, composite[y, id[domain[x]]]],
            p6 → equal[x, composite[y, id[cartsq[range[x]]]]]}]] // Reverse

Out[2]= or[equal[x, composite[y, id[cart[range[x], range[x]]]]],
          not[member[x, MONOIDS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[3]:= or[equal[x_, composite[y_, id[cart[range[x_], range[x_]]]]],
          not[member[x_, MONOIDS]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Lemma.

```
In[4]:= SubstTest[implies, and[equal[s, t], equal[u, v]],
          equal[image[inverse[s], u], image[inverse[t], v]],
          {s → x, t -> composite[y, id[cartsq[range[x]]]], u → ids[x], v -> ids[y]}] // Reverse

Out[4]= or[equal[composite[id[range[x]], image[inverse[y], ids[y]], id[range[x]]],
           image[inverse[x], ids[x]]],
          not[equal[x, composite[y, id[cart[range[x], range[x]]]]]],
          not[equal[ids[x], ids[y]]]] == True

In[5]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= Map[not,
          SubstTest[and, (* implies[and[p1,p2,p3],p4],implies[p1,p5],implies[p2,p6], *)
           implies[and[p4, p5, p6], p7], not[implies[and[p1, p2, p3], p7]],
           {p1 -> member[x, MONOIDS], p2 -> member[y, GROUPS], p3 -> subclass[x, y],
            p4 → equal[e[x], e[y]], p5 → equal[ids[x], set[e[x]]],
            p6 → equal[ids[y], set[e[y]]], p7 -> equal[ids[x], ids[y]]}]] // Reverse

Out[6]= or[equal[ids[x], ids[y]], not[member[x, MONOIDS]],
          not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[7]:= or[equal[ids[x_], ids[y_]], not[member[x_, MONOIDS]],
          not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Lemma.

```
In[8]:= SubstTest[implies, equal[u, v],
          equal[intersection[u, inverse[u]], intersection[v, inverse[v]]],
          {u -> composite[id[range[x]], image[inverse[y], ids[y]], id[range[x]]],
           v -> image[inverse[x], ids[x]]}] // Reverse

Out[8]= or[equal[composite[id[range[x]], inv[y], id[range[x]]], inv[x]],
          not[equal[composite[id[range[x]], image[inverse[y], ids[y]], id[range[x]]],
            image[inverse[x], ids[x]]]]] == True

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[10]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4], implies[and[p1, p2, p3], p5],
            implies[and[p4, p5], p6], implies[p6, p7], not[implies[and[p1, p2, p3], p7]],
            {p1 -> member[x, MONOIDS], p2 -> member[y, GROUPS], p3 -> subclass[x, y],
             p4 → equal[ids[x], ids[y]], p5 -> equal[x, composite[y, id[cartsq[range[x]]]]],
             p6 → equal[composite[id[range[x]], image[inverse[y], ids[y]], id[range[x]]],
               image[inverse[x], ids[x]]],
             p7 -> equal[composite[id[range[x]], inv[y], id[range[x]]], inv[x]]}]] // Reverse

Out[10]= or[equal[composite[id[range[x]], inv[y], id[range[x]]], inv[x]],
           not[member[x, MONOIDS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True
```

```
In[11]:= or[equal[composite[id[range[x_]], inv[y_], id[range[x_]]], inv[x_]],
            not[member[x_, MONOIDS]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Corollary.

```
In[12]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4], implies[p1, p5],
             implies[and[p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
             {p1 -> member[x, MONOIDS], p2 -> member[y, GROUPS], p3 -> subclass[x, y],
              p4 → equal[composite[id[range[x]], inv[y], id[range[x]]], inv[x]],
              p5 → equal[domain[x], cart[range[x], range[x]]],
              p6 → equal[inv[x], intersection[domain[x], inv[y]]]}]] // Reverse

Out[12]= or[equal[intersection[domain[x], inv[y]], inv[x]],
            not[member[x, MONOIDS]], not[member[y, GROUPS]], not[subclass[x, y]]] == True

In[13]:= or[equal[intersection[domain[x_], inv[y_]], inv[x_]],
            not[member[x_, MONOIDS]], not[member[y_, GROUPS]], not[subclass[x_, y_]]] := True
```

Theorem. A sufficient criterion for a submonoid to be a subgroup, is that every element of the submonoid be the inverse of an element of the submonoid. That is, a submonoid **x** of a group **y** is a subgroup if **range[x]** is subvariant under **inv[y]**.

```
In[30]:= Map[not,
           SubstTest[and, implies[and[p1, p2, p3], p4], implies[p4, p5], implies[and[p5, p6], p7],
            implies[and[p1, p7], p8], not[implies[and[p1, p2, p3, p6], p8]],
            {p1 -> member[x, MONOIDS], p2 -> member[y, GROUPS], p3 -> subclass[x, y],
             p4 → equal[composite[id[range[x]], inv[y], id[range[x]]], inv[x]],
             p5 → equal[intersection[image[inv[y], range[x]], range[x]], domain[inv[x]]],
             p6 → subvariant[inv[y], range[x]], p7 → equal[range[x], domain[inv[x]]],
             p8 → member[x, GROUPS]}]] // Reverse

Out[30]= or[member[x, GROUPS], not[member[x, MONOIDS]], not[member[y, GROUPS]],
           not[subclass[x, y]], not[subclass[range[x], image[inv[y], range[x]]]]] == True

In[31]:= or[member[x_, GROUPS], not[member[x_, MONOIDS]], not[member[y_, GROUPS]],
           not[subclass[x_, y_]], not[subclass[range[x_], image[inv[y_], range[x_]]]]] := True
```