image[V,-] rules for group elements

Johan G. F. Belinfante
2009 February 12

In[1]:=
```
SetDirectory["l:"]; << goedel.09feb11b; << tools.m
```

```
:Package Title: goedel.09feb11b            2009 February 11 at 2:45 p.m.

It is now:  2009 Feb 12 at 12:55

Loading Simplification Rules

TOOLS.M                                    Revised 2009 February 9

weightlimit = 40
```

summary

Knuth and Bendix long ago proposed an algorithm that attempts to transform a given set of equations into a convergent rewrite system. An application of this algorithm yields a set of ten rewrite rules for group application.

In[2]:=
```
"D. E. Knuth and P. B. Bendix, Simple word problems in
 universal algebras, in Computational Problems in Abstract Algebra,
 edited by J. Leech, pp. 263-297, Oxford, Pergamon Press, 1970.";
```

These ten rules are also discussed on page 19 in the following reference:

In[4]:=
```
"Leo Bachmair, Canonical Equational Proofs, Birkhäuser, Boston, 1991.";
```

The Knuth-Bendix rewrite rules for group application are approximated by ten rewrite rules in the **GOEDEL** program. The fact that these rules are about a group **gp[x]** is made known to the computer via presence of the **gp** wrapper , but since the computer still has no way of knowing that the elements to which the group application is being applied are members of **range[gp[x]]**, the rules also contain **image[V, ---]** expressions for which additional rewrite rules must be supplied. One such rule is derived in this notebook.

the ten Knuth Bendix rules for group application

The associative law for groups is represented by a rewrite rule that puts no conditions on the three variables **u**, **v** and **w**. No restriction on the variables is needed because **APPLY** occurs on both sides of the equation **u · (v · w) = (u · v) · w** If one or more of these is not a member of **range[gp[x]]**, then this equation reduces to **V = V**.

In[5]:=
```
APPLY[gp[x], PAIR[u, APPLY[gp[x], PAIR[v, w]]]]
```

Out[5]=
```
APPLY[gp[x], PAIR[APPLY[gp[x], PAIR[u, v]], w]]
```

The dual pair of rules that express the two axioms about the group identity element **e[gp[x]]** do introduce an **image[V, -]** term, however:

```
In[6]:=  APPLY[gp[x], PAIR[e[gp[x]], u]]

Out[6]=  union[u, complement[image[V, intersection[range[gp[x]], set[u]]]]]

In[7]:=  APPLY[gp[x], PAIR[u, e[gp[x]]]]

Out[7]=  union[u, complement[image[V, intersection[range[gp[x]], set[u]]]]]
```

The same holds for the two rewrite rules that represent the two group axioms about inverses:

```
In[8]:=  APPLY[gp[x], PAIR[APPLY[inv[gp[x]], u], u]]

Out[8]=  union[complement[image[V, intersection[range[gp[x]], set[u]]]], e[gp[x]]]

In[9]:=  APPLY[gp[x], PAIR[u, APPLY[inv[gp[x]], u]]]

Out[9]=  union[complement[image[V, intersection[range[gp[x]], set[u]]]], e[gp[x]]]
```

The theorem that the identity element is its own inverse involves no variables for elements, and is therefore clean:

```
In[10]:=  APPLY[inv[gp[x]], e[gp[x]]]

Out[10]=  e[gp[x]]
```

The theorem about the inverse of a product involves variables for group elements, but it is nonetheless clean because, like the associative law, an **APPLY** occurs on both sides of the equation.

```
In[11]:=  APPLY[inv[gp[x]], APPLY[gp[x], PAIR[u, v]]]

Out[11]=  APPLY[gp[x], PAIR[APPLY[inv[gp[x]], v], APPLY[inv[gp[x]], u]]]
```

The theorem about the inverse of an inverse involves one variable for a group element, and needs a corresponding **image[V, -]** modification.

```
In[12]:=  APPLY[inv[gp[x]], APPLY[inv[gp[x]], u]]

Out[12]=  union[u, complement[image[V, intersection[range[gp[x]], set[u]]]]]
```

Finally there is a dual pair of rewrite rules that follow from a combination of the laws of inverses and the associative law. These rules both involve two variables representing group elements, and both need **image[V, -]** modifications.

```
In[13]:=  APPLY[gp[x], PAIR[APPLY[gp[x], PAIR[u, v]], APPLY[inv[gp[x]], v]]]

Out[13]=  union[u, complement[image[V, intersection[range[gp[x]], set[u]]]],
           complement[image[V, intersection[range[gp[x]], set[v]]]]]

In[14]:=  APPLY[gp[x], PAIR[APPLY[gp[x], PAIR[u, APPLY[inv[gp[x]], v]]], v]]

Out[14]=  union[u, complement[image[V, intersection[range[gp[x]], set[u]]]],
           complement[image[V, intersection[range[gp[x]], set[v]]]]]
```

image[V, -] rules

Only the inverse of an element of a group can be an element of the group.

```
In[15]:= member[APPLY[inv[gp[x]], u], range[gp[x]]]

Out[15]= member[u, range[gp[x]]]
```

The corresponding **image[V, -]** rule for this fact is already available:

```
In[16]:= image[V, intersection[range[gp[x]], set[APPLY[inv[gp[x]], u]]]]

Out[16]= image[V, intersection[range[gp[x]], set[u]]]
```

A similar result holds for the product of two group elements.

Theorem. The product **u · v** belongs to **range[gp[x]]** if and only if both **u** and **v** belong to **range[gp[x]]**.

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

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

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

Corollary. The corresponding **image[V, -]** rule is needed for rewriting.

```
In[19]:= image[V, intersection[range[gp[x]], set[APPLY[gp[x], PAIR[u, v]]]]] // Normality

Out[19]= image[V, intersection[range[gp[x]], set[APPLY[gp[x], PAIR[u, v]]]]] ==
          intersection[image[V, intersection[range[gp[x]], set[u]]],
           image[V, intersection[range[gp[x]], set[v]]]]

In[20]:= image[V, intersection[range[gp[x_]], set[APPLY[gp[x_], PAIR[u_, v_]]]]] :=
          intersection[image[V, intersection[range[gp[x]], set[u]]],
           image[V, intersection[range[gp[x]], set[v]]]]
```