Automated Proofs in Group Theory with OTTER

José A. Alonso, Juan J. Arrabal, Delia Balbontín y Javier Herrera
Departamento de Álgebra, Computación, Geometría y Topología
Universidad de Sevilla

March 26, 1992

1 Introduction

Experimentation is a component vital of research in automated reasoning. Some problems for experimentation in Group Theory have been proposed by Wos in this book *Automated Reasoning: 33 Basic Research Problems* , the number 3 (p. 198) is the kernel problem and consist of automatic prove that the kernel of a group G in group H homomorphism is a normal subgroup of the group G. In Yu [90] is resolved the kernel problem for finite groups using the Boyer-Moore Theorem proving. In this work we show a resolution of kernel theorem using OTTER.

2 Problem representation

In this section we will see how represent through clauses that K is the kernel of homomorphism f of group G in group H.

2.1 G is a group

We begin with the representation that $(G, .)$ is a group; that is, . is a binary operation in set G that:

1. for all $x, y \in G$, $x.y \in G$ (closure);
2. there exists one element $e \in G$ so that $x.e = e.x = x$ for every element $x \in G$ (identity existence);
3. for all $x \in G$ exists one element $x^{-1} \in G$, so that $x.x^{-1} = x^{-1}.x = e$ (inverse existence);
4. for all $x, y, z \in G$, $(x.y).z = x.(y.z)$ (associative law).

The representation $(G, .)$ is a group is found in file `G_GROUP.IN` whose content is

```
% Description : "G is a group" axioms.
% Representation
% G(x)       : x is an element of G
% prod(x,y)  : product of x and y in G
% e          : identity element of G
% inv(x)     : inverse of x in G

list(usable).
% closure:
-G(x) | -G(y) | G(prod(x,y)).

% identity existence:
G(e).
% -G(x) | (prod(e,x) = x).          % demodulator
% -G(x) | (prod(x,e) = x).          % demodulator

% inverse existence:
-G(x) | G(inv(x)).
% -G(x) | (prod(x,inv(x)) = e).     % demodulator
% -G(x) | (prod(inv(x),x) = e).     % demodulator

% associative law:
-G(x) | -G(y) | -G(z) |
   (prod(prod(x,y),z) = prod(x,prod(y,z))).
end_of_list.
```

```
list(demodulators).
(prod(e,x) = x).
(prod(x,e) = x).
(prod(x,inv(x)) = e).
(prod(inv(x),x) = e).
end_of_list.
```

More over, is need the reflexive axiom written in the file `AX_REFLE.IN`:

```
list(usable).
(x = x).
end_of_list.
```

2.2 H is a group

Some way, the $(H, .)$ representation is a group is found in file `H_GROUP.IN` whose content is

```
% Description : "H is a group" axioms.
% Representation
% H(x)         : x is an element of H
% prod1(x,y)   : product of x and y in H
% e1           : identity element of H
% inv1(x)      : inverse of x in H

list(usable).
% closure:
-H(x) | -H(y) | H(prod1(x,y)).

% identity existence:
H(e1).
% -H(x) | (prod1(e1,x) = x).        % demodulator
% -H(x) | (prod1(x,e1) = x).        % demodulator

% inverse existence:
-H(x) | H(inv1(x)).
% -H(x) | (prod1(x,inv1(x)) = e1).  % demodulator
```

```
% -H(x) | (prod1(inv1(x),x) = e1).  % demodulator

% associative law:
-H(x) | -H(y) | -H(z) |
   (prod1(prod1(x,y),z) = prod1(x,prod1(y,z))).
end_of_list.

list(demodulators).
(prod1(e1,x) = x).
(prod1(x,e1) = x).
(prod1(x,inv1(x)) = e1).
(prod1(inv1(x),x) = e1).
end_of_list.
```

2.3 f is a homomorphism from G to H

A homomorphism f from a group $(G, .)$ to a group $(H, *)$ is a mapping from G to H so that for all $x, y \in G$, $f(x.y) = f(x) * f(y)$. Your representation is found in file `F_HOMOM.IN` whose content is

```
% Representation:
% f(x)       : image of x by f

list(usable).
-G(x) | H(f(x)).
-G(x) | -G(y) | (x != y) | (f(x) = f(y)).
-G(x) | -G(y) |
   (f(prod(x,y)) = prod1(f(x),f(y))).
end_of_list.
```

2.4 K is the kernel of f

The kernel of the homomorphism f is the set

$$K = \{x \in G : f(x) = e1\}$$

Your representation is found in file `K_NUCLEO.IN` whose content is

```
list(usable).
-K(x) | G(x).
-K(x) | (f(x) = e1).
(f(x) != e1) | K(x).
end_of_list.
```

3 The kernel is a subgroup

A subset A of a group G is a subgroup if

1. The identity of G is member of A.
2. If $x, y \in A$, then $x.y \in A$.
3. If $x \in A$, then $x^{-1} \in A$.

In this section we present a way to get a OTTER's proof of following theorem.

Theorem 1 *Let f be a group homomorphism from G to H. Then, the kernel of f is a subgroup of G.*

To proof we will use the following lemmas

Lemma 1 *If f is a group homomorphism from G to H, then the identity element of G is in the kernel of f.*

Proof: We assume the identity element of G don't lie in f kernel (i.e. `-K(e).`) We must verify this clause in union with the sets `ax_refle.in`, `g_group.in`, `h_group.in`, `f_homom.in`, and `k_kern.in` are an inconsistent clauses set. This is possible with these options: (1) hyper–resolution as deduction rule; (2) the support set consist in the denial (`-K(e)`) of goal and the clause `G(e)`, and (3) back demodulation for strategy. This is the content of file `L1.IN`:

```
set(hyper_res).
set(back_demod).

list(sos).
G(e).
-K(e).
end_of_list.
```

To get the proof we must merge the files `l1.in`, `ax_refle.in`, `g_group.in`, `h_group.in`, `f_homom.in` y `k_kern.in` in another file and we must apply OTTER to him. By MSDOS we have automated this process by the file `DEM.BAT`:

```
@echo off
copy %2.in+%3.in+%4.in+%5.in+%6.in+%1.in %1.aux
otter <%1.aux >%1.out
del %1.aux
```

Using this file, with the command

```
C> dem l1 ax_refle g_group h_group f_homom k_kern
```

we get the file `L1.OUT` which content the following proof of Lemma 1:

```
7 [] (prod(x,e) = x).
12 [] -H(x) | H(inv1(x)).
13 [] -H(x) | -H(y) | -H(z) |
      (prod1(prod1(x,y),z) =
      prod1(x,prod1(y,z))).
15 [] (prod1(x,e1) = x).
16 [] (prod1(x,inv1(x)) = e1).
18 [] -G(x) | H(f(x)).
20 [] -G(x) | -G(y) |
      (f(prod(x,y)) = prod1(f(x),f(y))).
23 [] (f(x) != e1) | K(x).
24 [] G(e).
25 [] -K(e).
27,26 [hyper,24,20,24,demod,7] (prod1(f(e),f(e))
      = f(e)).
28 [hyper,24,18] H(f(e)).
30 [hyper,28,12] H(inv1(f(e))).
46 [hyper,30,13,28,28,demod,27,16,16,15] (f(e) =
   e1).
64 [hyper,46,23] K(e).
65 [binary,64,25] .
```

Lemma 2 *If f is a group homomorphism from G to H, then for all $x, y \in ker(f)$, $x.y \in ker(f)$.*

Proof: Analogous at lemma 1, with exception of: (1) we include in support set they are two elements of kernel those product don't lie in the kernel, and (2) we add weight use strategy to direct and restrict the search. So we made the file `L2.IN`:

```
set(hyper_res).
set(back_demod).
assign(max_weight,50).

list(sos).
K(a).
K(b).
-K(prod(a,b)).
end_of_list.

weight_list(pick_and_purge).
weight(e1,-100).
end_of_list.
```

With the command

```
C> dem l2 ax_refle g_group h_group f_homom k_kern
```

we get the file `L2.OUT` which content the following proof of Lemma 2:

```
15 [] (prod1(x,e1) = x).
20 [] -G(x) | -G(y) | (f(prod(x,y)) =
   prod1(f(x),f(y))).
21 [] -K(x) | G(x).
22 [] -K(x) | (f(x) = e1).
23 [] (f(x) != e1) | K(x).
24 [] K(a).
25 [] K(b).
26 [] -K(prod(a,b)).
28,27 [hyper,24,22] (f(a) = e1).
29 [hyper,24,21] G(a).
31,30 [hyper,25,22] (f(b) = e1).
32 [hyper,25,21] G(b).
44 [hyper,32,20,29,demod,28,31,15] (f(prod(a,b))
```

```
   = e1).
60 [hyper,44,23] K(prod(a,b)).
61 [binary,60,26] .
```

Lemma 3 *If f is a group homomorphism from G to H, then for all $x \in ker(f)$, $x^{-1} \in ker(f)$.*

Proof: Similar at lemma 2, but we include in support set the clause they are a kernel element those inverse don't lie in kernel. So we made the file `L3.IN`:

```
set(hyper_res).
set(back_demod).
assign(max_weight,50).

list(sos).
K(a).
-K(inv(a)).
end_of_list.

weight_list(pick_and_purge).
weight(e1,-100).
end_of_list.
```

With the command

```
C> dem l3 ax_refle g_group h_group f_homom k_kern
```

we get the file `L3.OUT` which content the following proof of Lemma 3:

```
3 [] G(e).
4 [] -G(x) | G(inv(x)).
6 [] (prod(e,x) = x).
8 [] (prod(x,inv(x)) = e).
14 [] (prod1(e1,x) = x).
15 [] (prod1(x,e1) = x).
20 [] -G(x) | -G(y) | (f(prod(x,y)) =
      prod1(f(x),f(y))).
21 [] -K(x) | G(x).
22 [] -K(x) | (f(x) = e1).
```

```
23 [] (f(x) != e1) | K(x).
24 [] K(a).
25 [] -K(inv(a)).
27,26 [hyper,24,22] (f(a) = e1).
28 [hyper,24,21] G(a).
32,31 [hyper,28,20,3,demod,6,27,27,15] (f(e) =
      e1).
34 [hyper,28,4] G(inv(a)).
39 [hyper,34,20,28,demod,8,32,27,14] (f(inv(a)) =
   e1).
54 [hyper,39,23] K(inv(a)).
55 [binary,54,25] .
```

Proof of Theorem 1: We use the denial of goal formula as support set and the clauses of above lemmas as usable clauses. So we get the file `T1.IN` for Theorem 1:

```
set(hyper_res).
set(back_demod).
assign(max_weight,1).

list(usable).
K(e).                              % Lemma 1
-K(x) | -K(y) | K(prod(x,y)).      % Lemma 2
-K(x) | K(inv(x)).                 % Lemma 3
end_of_list.

formula_list(sos).
(-K(e) |
 (K(a) & K(b) & -K(prod(a,b))) |
 (K(c) & -K(inv(c)))).
end_of_list.

weight_list(pick_and_purge).
weight(K(e),-1).
weight(K(a),-1).
weight(K(b),-1).
weight(K(c),-1).
```

```
weight(K(prod(a,b)),-1).
weight(K(inv(c)),-1).
weight(G(*1),10).
end_of_list.
```

With the command

```
C> dem t1 ax_refle g_group h_group f_homom k_kern
```

we get the file `T1.OUT` which content the following proof of Theorem 1

```
23 [] K(e).
24 [] -K(x) | -K(y) | K(prod(x,y)).
25 [] -K(x) | K(inv(x)).
26 [] -K(e) | K(a) | K(c).
27 [] -K(e) | K(a) | -K(inv(c)).
28 [] -K(e) | K(b) | K(c).
29 [] -K(e) | K(b) | -K(inv(c)).
30 [] -K(e) | -K(prod(a,b)) | K(c).
31 [] -K(e) | -K(prod(a,b)) | -K(inv(c)).
32 [hyper,26,23] K(a) | K(c).
33 [hyper,28,23] K(b) | K(c).
34 [hyper,32,25] K(a) | K(inv(c)).
35 [hyper,33,24,32] K(c) | K(prod(a,b)).
36 [hyper,33,25] K(b) | K(inv(c)).
38 [hyper,34,27,23] K(a).
40 [hyper,35,30,23] K(c).
41 [hyper,36,29,23] K(b).
42 [hyper,40,25] K(inv(c)).
43 [hyper,41,24,38] K(prod(a,b)).
44 [hyper,43,31,23,42] .
```

4 The kernel is normal

A subgroup A of group G is normal if for all $x \in G$ and all $y \in A$, $x.y.x^{-1} \in A$. In this section we show a OTTER's proof of next theorem

Theorem 2 *If f is a group homomorphism from G to H, then the kernel of f is a normal subgroup of G.*

We use to prove it the following lemma:

Lemma 4 *If f is a group homomorphism from G to H, then for all $x, y, z \in G$*

$$f((x.y).z) = (f(x).f(y)).f(z)$$

Proof:

We are made these elections: (1) include the denial of goal (there exists $a, b, c \in G$ so that

$$f((a.b).c) \neq (f(a).f(b)).f(c))$$

in support set; (2) choose hyper–resolution and paramodulation as inference rules, and (3) choose as strategy dynamic demodulation, the weight use to direct and restrict the search, and use queue support set. So we made the file `L4.IN`:

```
set(hyper_res).
set(para_from).
set(dynamic_demod).
set(sos_queue).
assign(max_weight,1).

list(sos).
G(a).
G(b).
G(c).
(f(prod(prod(a,b),c)) !=
  prod1(prod1(f(a),f(b)),f(c))).
end_of_list.

weight_list(pick_and_purge).
weight(G(*1),100).
weight(H(*1),500).
weight(prod(a,b),-200).
weight(prod(*1,*1),500).
weight(prod1(f(x),f(y)),-100).
weight(inv(*1),500).
weight(inv1(*1),500).
```

```
weight(f(*1),-100).
weight(e,500).
weight(e1,500).
end_of_list.
```

With the command

```
C> dem l4 ax_refle g_group h_group f_homom k_kern
```

we get the file `L4.OUT` which content the following proof of Lemma 4:

```
2 [] -g(x) | -g(y) | g(prod(x,y)).
20 [] -g(x) | -g(y) | (f(prod(x,y)) =
   prod1(f(x),f(y))).
21 [] g(a).
22 [] g(b).
23 [] g(c).
24 [] (f(prod(prod(a,b),c)) !=
   prod1(prod1(f(a),f(b)),f(c))).
25 [hyper,22,20,21] (prod1(f(a),f(b)) =
   f(prod(a,b))).
26 [hyper,22,2,21] g(prod(a,b)).
27 [para_from,25,24] (prod1(f(prod(a,b)),f(c)) !=
   f(prod(prod(a,b),c))).
37 [hyper,26,20,23] (prod1(f(prod(a,b)),f(c)) =
   f(prod(prod(a,b),c))).
38 [binary,37,27] .
```

Proof of Theorem 2:

We are made the following options: (1) include the Lemma 4 in usable clauses list; (2) include in support set the denial of goal (they are elements a of kernel and b of G so that $b.a.b^{-1}$ don't lie in kernel); (3) choose hyper–resolution as inference rule; (4) choose back demodulation, the weight use to direct and restrict the search, and use queue support set. So we made the file `T2.IN`:

```
set(hyper_res).
set(back_demod).
```

```
set(sos_queue).
assign(max_weight,700).

list(usable).
-g(x) | -g(y) | -g(z) |
   (f(prod(prod(x,y),z)) =
   prod1(prod1(f(x),f(y)),f(z))).
end_of_list.

list(sos).
k(a).
g(b).
-k(prod(prod(b,a),inv(b))).
end_of_list.

weight_list(pick_and_purge).
weight(inv(a),1000).
weight(prod(a,a),1000).
weight(prod(a,b),1000).
weight(prod(b,b),1000).
weight(g(prod(*1,*1)),1000).
weight(g(*1),300).
weight(h(*1),1000).
weight(inv(*1),300).
weight(inv1(*1),500).
end_of_list.
```

With the command

```
C> dem t2 ax_refle g_group h_group f_homom k_kern
```

we get the file `T2.OUT` which content the following proof

```
3 [] G(e).
4 [] -G(x) | G(inv(x)).
6 [] (prod(e,x) = x).
7 [] (prod(x,e) = x).
8 [] (prod(x,inv(x)) = e).
```

```
15 [] (prod1(x,e1) = x).
20 [] -G(x) | -G(y) | (f(prod(x,y)) =
   prod1(f(x),f(y))).
21 [] -K(x) | G(x).
22 [] -K(x) | (f(x) = e1).
23 [] (f(x) != e1) | K(x).
24 [] -G(x) | -G(y) | -G(z) |
   (f(prod(prod(x,y),z)) =
   prod1(prod1(f(x),f(y)),f(z))).
25 [] K(a).
26 [] G(b).
27 [] -K(prod(prod(b,a),inv(b))).
29,28 [hyper,25,22] (f(a) = e1).
30 [hyper,25,21] G(a).
41 [hyper,26,4] G(inv(b)).
48,47 [hyper,30,20,3,demod,6,29,29,15] (f(e) =
      e1).
50 [hyper,41,24,26,30,demod,29,15]
   (f(prod(prod(b,a),inv(b))) =
   prod1(f(b),f(inv(b)))).
52,51 [hyper,41,24,26,3,demod,7,8,48,48,15]
      (prod1(f(b),f(inv(b))) = e1).
69 [back_demod,50,demod,52]
   (f(prod(prod(b,a),inv(b))) = e1).
73 [hyper,69,23] K(prod(prod(b,a),inv(b))).
74 [binary,73,27] .
```

5 Conclusions

This paper describes an experiment with OTTER in Group Theory. In our opinion, this paper provides some basic techniques to attack theorems in Group Theory using OTTER. The next target would be Lagrange's and Sylow's Theorems.

It must be emphasized that to prove the two theorems of this paper, only 4 intermediate lemmas had to be suggested to the prover.

Finalmente, deseamos destacar la importancia de la elección de las es-

trategias (fundamentalmente, la elección de pesos) en el éxito del experimento.

6 References

1. McCune, W.W., *OTTER 2.0 User's Guide* . Tech. Report ANL–90/9, Argonne National Laboratory, Argonne, Ill., March 1990.

2. McCune, W.W., *What's New in Otter 2.2* . Tech. Report ANL/MCS–TM–153, Argonne National Laboratory, Argonne, Ill., July 1991.

3. Wos, L., *Automated Reasoning: 33 Basic Research Problems* , Prentice–Hall, 1987.

4. Yu, Y., 'Computer proofs in Group Theory', *J. Automated Reasoning* **6** (1990) 251–286.