

PGRelFind

A GAP4 program found in the ACE package

by

Greg Gamble and George Havas

Centre for Discrete Mathematics and Computing
Department of Information Technology and Electrical Engineering
The University of Queensland, St. Lucia 4072, Australia

and

Alexander Hulpke

Department of Mathematics, Colorado State University
Weber Building, Fort Collins, CO 80523, USA

written for a 2002 paper

by

Colin M. Campbell, George Havas, Alexander Hulpke and Edmund F. Robertson

in which

$L_3(5)$ is shown to be efficient

February 2006

Contents

1	PGRelFind: A GAP Example using ACE	3
1.1	Introduction	3
1.2	Defining the functions we use	4
1.3	The doXXX.g files	7
1.4	Examples	9
1.5	Authors	11
	Bibliography	12
	Index	13

1

PGRelFind: A GAP Example using ACE

PGRelFind is a GAP 4 program found in the ACE package written by Alexander Hulpke, Greg Gamble and George Havas for a 2002 paper by Colin M. Campbell, George Havas, Alexander Hulpke and Edmund F. Robertson [CHHR02] in which the simple group $L_3(5)$ is shown to be efficient.

1.1 Introduction

The purpose of this example is to show how GAP together with the ACE package may be used to determine the deficiency of some perfect groups. The GAP code necessary is provided in the ACE package via the function `ACEReadResearchExample`. The functions used are found in the file "pgrrelfind.g" (which is well-commented) in the `ace/res-examples` directory formed when one installs the ACE package.

Some background

The **deficiency** of a finite presentation $\{X \mid R\}$ of a finite group G is $|R| - |X|$, and the **deficiency** of the group G is the minimum of the deficiencies of all finite presentations of G .

The following result (with a constructive proof) is Lemma 2.2 in [CHHR02].

Lemma Let G be a simple group with trivial Schur multiplier. Suppose G has a presentation of the form

$$P = \{a, b \mid a^2 = 1, b^p = 1, w(a, b) = 1\}$$

with p prime. Then G has deficiency zero.

Prior to the paper [CHHR02] there remained just two simple groups with trivial Schur multiplier of order less than one million, namely $L_3(5)$ and $PSp_4(4)$, that had not been shown to have deficiency zero. Using `TranslatePresentation` and `PGRelFind` the question for $L_3(5)$ is now settled, but whether $PSp_4(4)$ has deficiency 0 is still open.

We now sketch the method used in [CHHR02] to show that a simple group G with trivial Schur multiplier (in particular, $L_3(5)$) has deficiency 0. We start with a known two-generator presentation of the group G , on say x and y . Then an invocation of `TranslatePresentation` (which uses Tietze transformations) translates the presentation into a presentation on two new generators a and b say, having several relators but for which the first two relators are of form a^2 and b^p for some prime p . Finally, `PGRelFind` is invoked to find a presentation on the generators a and b found by `TranslatePresentation` that contains a^2 and b^p and just one more relator. If the search by `PGRelFind` is successful then the Lemma shows the group has deficiency zero. If the initial presentation is already in the form required by `PGRelFind`, the `TranslatePresentation` step can be omitted.

1.2 Defining the functions we use

After starting up GAP (on most systems that is done by typing `gap` or `gap4` at the system prompt), one needs to load up the ACE package which, if installed, is done as follows:

```
gap> LoadPackage("ace");
-----
Loading    ACE (Advanced Coset Enumerator) 5.0
GAP code by Greg Gamble <gregg@itee.uq.edu.au> (address for correspondence)
    Alexander Hulpke (http://www.math.colostate.edu/~hulpke)
    [uses ACE binary (C code program) version: 3.001]
C code by  George Havas (http://www.itee.uq.edu.au/~havas)
    Colin Ramsay (http://www.itee.uq.edu.au/~cram)

                For help, type: ?ACE
-----
true
```

The ACE package provides a function `ACEReadResearchExample`, which, without an argument, reads the file `"pgrrelfind.g"` in the `ace/res-examples` directory formed when one installs the ACE package, and in so doing defines functions such as `PGRelFind` that were used in [CHHR02] to show that the group $L_3(5)$ has deficiency 0:

```
gap> ACEReadResearchExample();
#I The following are now defined:
#I
#I Functions:
#I   PGRelFind, ClassesGenPairs, TranslatePresentation,
#I   IsACEResExampleOK
#I Variables:
#I   ACEResExample, ALLRELS, newrels, F, a, b, newF, x, y,
#I   L2_8, L2_16, L3_3s, U3_3s, M11, L2_32,
#I   U3_4s, J1s, L3_5s, PSp4_4s, presentations
#I
#I Also:
#I
#I TCENUM = ACETCENUM (Todd-Coxeter Enumerator is now ACE)
#I
#I For an example of their application, you might like to try:
#I gap> ACEReadResearchExample("doL28.g" : optex := [1,2,4,5,8]);
#I (the output is 65 lines followed by a 'gap>' prompt)
#I
#I For information type: ?Using ACEReadResearchExample
gap>
```

The output (Info-ed at `InfoACELevel 1`) states that a number of functions and variables have been defined, which we will now describe.

1 ► `PGRelFind(fgens, rels, sgens)`

F

For the perfect **simple** group $S = \langle fgens \mid rels \rangle$ with subgroup $\langle sgens \rangle$ where each element of the list `sgens` is a word in the generators `fgens`, `PGRelFind` tries to find a third relator such that together with the two order-defining relators of `fgens` a presentation on three relators is determined, and if successful returns a record with fields `fgens := fgens`, `rels` set to the new 3-relator list found, and `sgens := sgens`. `fgens`

should be a list of two generators of a free group, say $[a, b]$; *rels* should contain words in the generators *fgens*, the first of which should determine that a is an involution, and the second should determine that b is of order p for some prime p . The subgroup $\langle sgens \rangle$ should ideally be a maximal subgroup of $\langle fgens \mid rels \rangle$. It turns out that one only needs to test for relators that are the products of an odd number of bi-syllables of form ab^k , for some integer k in the range $1, \dots, p-1$. We will use the term **bi-syllabic length** to mean the number of bi-syllables of form ab^k in such a relator. Furthermore, each relator we test can be assumed to have the prefix (which we subsequently call the *head*) $ababab^{-1}$ if p is 3, or ab otherwise.

Each relator tested is of form *head* * *middle* * *tail*, where each of *head*, *middle*, *tail* is a word of integral bi-syllabic length. Moreover,

- *head* is fixed (as mentioned above);
- all *tails* of the opposite parity bi-syllabic length up to a maximum prescribed length are pre-computed (since *head* is generally of odd bi-syllabic length, *tails* are generally of even bi-syllabic length); and
- the bi-syllabic length of the *middles* is always even, starting from an initial minimum *middle* length and increasing in steps of *granularity* + 2, where *granularity* is defined to be the difference of the maximum and minimum bi-syllabic lengths of a *tail* (see below).

Since the *tails* in general are not all of the same length, it is possible when a search is successful that the first relator found is not the shortest. Hence searching continues until a relator of shortest length can be guaranteed.

To provide some user control of the algorithm, PGRelFind has a number of options (entered after the arguments and after a colon in the usual way):

head := *head*

Redefines *head* which by default is $a*b*a*b*a/b$ if the order of b is 3, or $a*b$ otherwise, where a and b are as defined above; *head* must be a word consisting of $a*b^k$ bi-syllables for integers k .

Nrandom := *nOrFunc*

Sets the number of *middles* to be generated for each length of a *middle*; *nOrFunc* may be a non-negative integer or a single-argument function that returns a positive integer.

Each *middle* may be generated sequentially or randomly. If *nOrFunc* = 0 then *middles* are generated sequentially (and hence exhaustively), as they are by default. If *nOrFunc* is a positive integer then, for each bi-syllabic length of a *middle*, *nOrFunc* “random” *middles* are generated. If *nOrFunc* is a single-argument function then for each bi-syllabic length *len* of a *middle*, *nOrFunc*(*len*) “random” *middles* are generated.

ACEworkspace := *workspace*

Sets the option **workspace** used by ACE to *workspace* when running the index check of the large subgroup. If the index is right, ACE is set to use $2*workspace$ to check the index of the cyclic subgroup $\langle b \rangle$. By default, *workspace* is set to 10^6 , which is too small for many of the *doXXX.g* files (see below). Of course, groups for which ACE overflows the workspace are indistinguishable from infinite groups, and one can't easily be sure whether a relator wasn't found because the workspace was too small or because there wasn't one to be found.

Ntails := *Ntails*

Sets the approximate number of *tails* generated to *Ntails*; it is used to set **maxTailLength** (described next); *Ntails* must be a positive integer. The default behaviour is given by *Ntails* = 2048.

maxTailLength := *len*

Sets the **intended** maximum bi-syllabic length of a *tail* to *len*; *len* must be a positive integer.

The default behaviour is given by $len = \text{LogInt}(Ntails, \text{order}b - 1)$, where *orderb* is the order of b . The **actual** maximum bi-syllabic length of a *tail* may be set 1 less, in order that it has the same parity as the minimum bi-syllabic length of a *tail*.

`minMiddleLength := len`

Sets the minimum bi-syllabic length of a *middle* (which, by default is 0) to *len*; *len* should be a non-negative even integer.

`maxMiddleLength := len`

Sets the **intended** maximum bi-syllabic length of a *middle* to *len*; *len* should be a positive integer. The default behaviour is given by *len* = 30. The **actual** maximum bi-syllabic length of a *middle* is adjusted downwards by the least amount necessary to ensure that the difference of the minimum and maximum bi-syllabic lengths of a *middle* is divisible by *granularity* + 2. (Recall that *granularity* is the difference of the maximum and minimum bi-syllabic lengths of a *tail*.)

Please note that the relators tested are saved in the lists `ALLRELS` and `newrels` (see 1.2.6).

2 ▶ `ClassesGenPairs(G, orderx, ordery)` F

finds, and returns as a list of 2-element lists, generator pairs of elements for the group *G* of orders *orderx* and *ordery*, where *orderx* and *ordery* are positive integers.

3 ▶ `TranslatePresentation(fgens, rels, sgens, newgens)` F

finds a new presentation, in terms of the generators *newgens*, for the **simple** group $S = \langle fgens \mid rels \rangle$, with subgroup generated by *sgens*; *fgens* should be a **pair** of free group generators; *rels*, *sgens* and *newgens* should be lists of words in *fgens*. Furthermore, *newgens* should be a list of two words, the first of which should be of even order and the second of odd prime order in the group *S*. The idea is that, by using Tietze transformations, the variables *x* and *y* (see 1.2.8) are assigned to the words of *newgens*, which make up the new *fgens*, and *rels* and *sgens* are written in terms of the new *fgens*, `TranslatePresentation` returns a record with fields `fgens`, `rels` and `sgens` that are the new values of *fgens*, and *rels* and *sgens*, respectively.

4 ▶ `IsACEResExampleOK()` F

does a number of integrity checks and tries to give an accurate diagnosis if something is wrong, returning `true` if everything is “OK” and `false` otherwise. It is called whenever `ACEReadResearchExample` is executed and is not intended for direct usage by users.

5 ▶ `ACEResExample` V

is initially set to a record with one field `filename` whenever `ACEReadResearchExample` is executed and the value set to that field is the name of the file read by `ACEReadResearchExample`; other fields are set as the need arises. Essentially, it is used as a temporary variable store when `ACEReadResearchExample` reads a `doXXX.g` file.

6 ▶ `ALLRELS` V

▶ `newrels` V

are initially set to null lists. Each time `PGRelFind` is executed these each “third” relator tested is appended to `ALLRELS` and each relator that is satisfied by the group defined by the *fgens* and *rels* arguments of `PGRelFind` (see 1.2.1) is appended to `newrels`. The user should normally reset these lists back to null lists between invocations of `PGRelFind`.

7 ▶ `F` V

▶ `a` V

▶ `b` V

are the free group `F` and its two generators `a` and `b`, used in the presentations described below.

- 8 ▶ `newF` V
- ▶ `x` V
- ▶ `y` V

are the free group `newF` and its two generators `x` and `y`; the record field `fgens` returned by `TranslatePresentation` is the list `[x, y]` (see 1.2.3).

- 9 ▶ `L2_8` V
- ▶ `L2_16` V
- ▶ `L3_3s` V
- ▶ `U3_3s` V
- ▶ `M11` V
- ▶ `L2_32` V
- ▶ `U3_4s` V
- ▶ `J1s` V
- ▶ `L3_5s` V
- ▶ `PSp4_4s` V

contain presentations for the perfect simple groups $L_2(8)$, $L_2(16)$, $L_3(3)$, $U_3(3)$, M_{11} , $L_2(32)$, $U_3(4)$, J_1 , $L_3(5)$ and $PSp_4(4)$, respectively. Those with names ending in `s` contain lists of records (the `s` is meant to suggest “plural”), and those with no `s` are just single records, where each record has the following fields:

`source`

a citation giving the source from which the presentation was obtained, where `CCN85`, `CMY79` and `CR84` indicate `[CCN+85]`, `[CMY79]` and `[CR84]`, respectively;

`rels`

the relators of the group in terms of the generators `a` and `b` defined above (see 1.2.7); and

`sgens`

the generators of a large (usually maximal) subgroup of the group, in terms of the generators `a` and `b`.

- 10 ▶ `presentations` V

is a record whose fields are the names of the presentation variables of 1.2.9, and whose values are the variables, e.g. `presentations.L2_16` is the same object as `L2_16`.

1.3 The *doXXX.g* files

Corresponding to each group for which presentations are supplied there is a `doXXX.g` file in the ACE package’s `res-examples` directory which can be read and executed via the command `ACEReadResearchExample`. There is also `ACEPrintResearchExample` if you only wish to see the essential commands executed by the corresponding `ACEReadResearchExample` command. Since these two commands are important for us we repeat their descriptions from the ACE package manual here:

- 1 ▶ `ACEReadResearchExample(filename)` F
- ▶ `ACEReadResearchExample()` F

perform `Read` (see Section 9.7.1 in the GAP Reference Manual) on `filename` or, with no argument, the file with filename `"pgrelfind.g"` in the `res-examples` directory; e.g. the effect of running

```
gap> ACEReadResearchExample("pgrelfind.g");
```

is equivalent to executing:

```
gap> Read( Filename(DirectoriesPackageLibrary("ace", "res-examples"),
>                 "pgrelfind.g" ) );
```

- 2 ▶ ACEPrintResearchExample(*example-filename*) F
▶ ACEPrintResearchExample(*example-filename*, *output-filename*) F

print the “essential” contents of the file *example-filename* in the `res-examples` directory to the terminal, or with two arguments to the file *output-filename*; *example-filename* and *output-filename* should be strings. ACEPrintResearchExample is provided to make it easy for users to copy and edit the examples for their own purposes.

Each `doXXX.g` file (other than the generic `doGrp.g`) has a name that is formed by taking one of the fields of `presentations`, removing any underscore or `s`, prepending “do” and appending “.g”; also, each requires that you run

```
gap> ACEReadResearchExample();
```

first to define the functions described above. If you forget, never mind ... you will be reminded.

In essence, each `doXXX.g` file (other than the generic `doGrp.g`) provides a valid `TranslatePresentation` and `PGRelFind` command combination using one of the presentations in the `presentations` record, for a particular group.

The `doGrp.g` file provides a way of choosing **any** presentation in the `presentations` record, not just the particular ones used by the other `doXXX.g` files.

Many of the outputs are fairly lengthy and you may wish to use `LogTo` (see 9.7.5) in order to peruse the output at leisure, later. Also, for many of the larger groups one needs to be able to set `ACEworkspace` larger than 10^6 , but how large? The investigation showing that $L_3(5)$ has deficiency 0 was run on an SGI Origin 2000 with 6.4 Gb of RAM and with `ACEworkspace` set to 10^8 , to find a “third” relator of 23 bisyllables. It turns out however that one need only set `ACEworkspace := 5 * 10^6` to find that relator ... one can be very knowledgeable in hindsight! In the list below we give values for `ACEworkspace` that will succeed in finding the shortest “third” relator that we found, when the method succeeded. (Of course, `PGRelFind` runs a lot faster if the workspace given to ACE is kept small.)

Recall `ACEReadResearchExample` expects a filename (i.e. a string), or no argument; here are the **filenames** of the `doXXX.g` provided:

`"doGrp.g"`

This allows the user to experiment with any of the groups that have data in the `presentations` record. The user **must** supply a value for the following option:

`grp := presField`

presField must be the name (i.e. a string) of a field in the `presentations` record (this selects the group to be investigated); and, possibly one or both of the following options:

`n := n`

If *presField*, above, ends in `s` then the user **must** supply a positive integer *n* for `n`, which indicates that the *n*th record of `presentations`. (*presField*) is desired.

`newgens := wordlist`

wordlist must be a list of two words in the generators `a` and `b` which are to be the words used for the *newgens* argument of `TranslatePresentation` (see 1.2.3 for the conditions they must satisfy). If `newgens` is not supplied, it is assumed that `TranslatePresentation` is not needed, i.e. that the generators `a` and `b` are already an involution and an element of odd prime order, respectively, and that `PGRelFind` may be immediately applied.

The user may also supply `PGRelFind` options, though unlike `grp` and `newgens`, they will not be directly observable in the output. An example using both sets of options is given below.

"doL28.g"

This provides a nice small example. You may supply any `PGRelFind` options you wish, and/or you may use the following option which has been specifically designed to help you get acquainted with the `PGRelFind` options (only the equivalent of the `optex` options will be explicitly observable in the output).

`optex := val`

val may be an integer or a list of integers in the range $1, \dots, 8$. The output of `ACEReadResearchExample()`; suggests a combination that is 65 lines long. Most of the possible choices for `optex` give an output of the order of 100 to 200 lines, which isn't too bad if you have say an `xterm` window with the capacity to scroll back.

"doL216.g"

As is the case with the remaining `doXXX.g` files, there are no options other than those for `PGRelFind`. It is sufficient to set `ACEworkspace := 2 * 10^4`.

"doL33.g"

It is sufficient to set `ACEworkspace := 2 * 10^4`.

"doU33.g"

Though $U_3(3)$ is known to have deficiency 0, the `TranslatePresentation-PGRelFind` method does not appear to be able to demonstrate it.

"doM11.g"

`ACEworkspace` needs to be set to at least $6 * 10^6$.

"doL232.g"

It is sufficient to set `ACEworkspace := 5 * 10^4`.

"doU34.g"

It is sufficient to set `ACEworkspace := 5 * 10^4`.

"doJ1.g"

`ACEworkspace` needs to be set to at least $5 * 10^6$.

"doL35.g"

`ACEworkspace` needs to be set to at least $5 * 10^6$.

"doPSp44.g"

We were unable to determine whether $PSp_4(4)$ has deficiency 0, via the `TranslatePresentation-PGRelFind` method.

1.4 Examples

The following example shows that you can use both the options peculiar to the file `doGrp.g` and the options of `PGRelFind` when `ACEReadResearchExample` is called with argument (filename) `"doGrp.g"`.

```
gap> ACEReadResearchExample("doGrp.g")
>           : grp := "L2_8", newgens := [a^3*b, a^2*b],
>           head := x*y*x*y*x*y^-1*x*y*x*y);
# IsACEResExampleOK() sets ACEResExample.grp    from options grp, n
#           ACEResExample.newgens from option newgens
gap> ACEResExample.G := TranslatePresentation([a, b],
>           ACEResExample.grp.rels,
>           ACEResExample.grp.sgens,
>           ACEResExample.newgens);
#I there are 4 generators and 6 relators of total length 45
#I there are 2 generators and 4 relators of total length 55
rec(
```

```

fgens := [ x, y ],
rels := [ x^2, y^3, y^-1*x*y^-1*x*y^-1*x*y^-1*x*y^-1*x*y^-1*x*y^-1*x*y^-1*x,
          x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1*x*y*x*y^-1 ],
sgens := [ x*y^-1 ] )
gap> ACEResExample.Gn := PGRelFind(ACEResExample.G.fgens,
>                                ACEResExample.G.rels,
>                                ACEResExample.G.sgens);
GroupOrder=504 SubgroupIndex=72
#bisyllables in head = 5 head: x*y*x*y*x*y^-1*x*y*x*y
Max #bisyllables in tail = 10 (granularity = 10)
#bisyllables in middle = 0
Candidate relator: x*y*x*y*x*y^-1*x*y*x*y*x*y^-1*x*y^-1*x*y*x*y*x*y*x*y^-1
-1*x*y^-1*x*y^-1*x*y*x*y^-1
#bisyllables = 15 (#bisyllables in tail = 10) #words tested: 1
ACEStats:
  index=72 cputime=10 ms maxcosets=98 totcosets=121
Large subgroup index OK
ACEStats for cyclic subgroup:
  index=168 cputime=10 ms maxcosets=170 totcosets=232
Cyclic subgroup index OK
Success! ... new relator:
  x*y*x*y*x*y^-1*x*y*x*y*x*y^-1*x*y^-1*x*y*x*y*x*y*x*y^-1*x*y^-1*x*y^-1
-1*x*y*x*y^-1
... continuing (there may be a shorter relator).
Candidate relator: x*y*x*y*x*y^-1*x*y*x*y*x*y^-1*x*y*x*y^-1*x*y^-1*x*y^-1
-1*x*y*x*y*x*y*x*y^-1*x*y^-1
#bisyllables = 15 (#bisyllables in tail = 10) #words tested: 1
ACEStats:
  index=72 cputime=10 ms maxcosets=102 totcosets=123
Large subgroup index OK
ACEStats for cyclic subgroup:
  index=168 cputime=0 ms maxcosets=170 totcosets=236
Cyclic subgroup index OK
Success! ... new relator:
  x*y*x*y*x*y^-1*x*y*x*y*x*y^-1*x*y*x*y^-1*x*y^-1*x*y^-1*x*y*x*y*x*y*x*y^-1
-1*x*y^-1
... continuing (there may be a shorter relator).
Middles of length 0 exhausted.
Relator found of length 15, is shortest.
rec(
  gens := [ x, y ],
  rels :=
    [ x^2, y^3, x*y*x*y*x*y^-1*x*y*x*y*x*y^-1*x*y*x*y^-1*x*y^-1*x*y^-1*x*y*x*y*\
x*y*x*y^-1*x*y^-1 ],
  sgens := [ x*y^-1 ] )
gap>

```

The following

```
gap> ACEReadResearchExample("doGrp.g" : grp := "L3_3s", n := 1);
```

is essentially equivalent to

```
gap> ACEReadResearchExample("doL33.g");
```

Observe that the option `n` is needed because `grp` selects a list of records (as indicated by the trailing `s` of `"L3_3s"`). Also, observe that the `newgens` option was not used since we already have a presentation where the generators `a` and `b` are an involution and of odd prime order, respectively.

1.5 Authors

Alexander Hulpke
Department of Mathematics, Colorado State University
Weber Building, Fort Collins, CO 80523, USA

Greg Gamble and George Havas
Department of Information Technology and Electrical Engineering
The University of Queensland, St. Lucia 4072, Australia

For email addresses refer to the ACE package banner in the section: 1.2.

Bibliography

- [CCN+85] J[ohn] H. Conway, R[obert] T. Curtis, S[imon] P. Norton, R[ichard] A. Parker, and R[obert] A. Wilson. *Atlas of finite groups*. Oxford University Press, 1985.
- [CHHR02] Colin M. Campbell, George Havas, Alexander Hulpke, and Edmund F. Robertson. Efficient simple groups. *Communications in Algebra*, 30(9):4613–4619, 2002.
- [CMY79] John J. Cannon, John McKay, and Kiang Chuen Young. The non-abelian simple groups G , $|G| < 10^5$ — presentations. *Communications in Algebra*, 7(13):1397–1406, 1979.
- [CR84] Colin M. Campbell and Edmund F. Robertson. Presentations for the simple groups G , $10^5 < |G| < 10^6$. *Communications in Algebra*, 12(21):2643–2663, 1984.

Index

This index covers only this manual. A page number in *italics* refers to a whole section which is devoted to the indexed subject. Keywords are sorted with case and spaces ignored, e.g., “PermutationCharacter” comes before “permutation group”.

A

a, 6

ACEPrintResearchExample, 8

ACEReadResearchExample, 7

ACEResExample, 6

ALLRELS, 6

Authors, 11

B

b, 6

C

ClassesGenPairs, 6

D

Defining the functions we use, 4

E

Examples, 9

F

F, 6

I

Introduction, 3

IsACEResExampleOK, 6

J

J1s, 7

L

L2_16, 7

L2_32, 7

L2_8, 7

L3_3s, 7

L3_5s, 7

M

M11, 7

N

newF, 7

newrels, 6

P

PGRelFind, 4

presentations, 7

PSp4_4s, 7

T

The doXXX.g files, 7

TranslatePresentation, 6

U

U3_3s, 7

U3_4s, 7

X

x, 7

Y

y, 7