TransGrp

Library of Transitive Groups

by

Alexander Hulpke

Department of Mathematics
Colorado State University

Including Data by

John Cannon
Derek Holt
Gordon Royle
and
Gareth Tracy

Contents

1 The Library of Transitive Groups 3

1.1 Transitive Permutation Groups . . . 3

1.2 Selection Functions 4

1.3 Interfacing with the library 5

1.4 Supplemental Downloads 5

Bibliography 6

Index 7

1 The Library of Transitive Groups

The Transitive Groups Library is created by Alexander Hulpke. You are free to distribute it, provided that you make no modifications.

The actual groups in the library and their generating sets are due to a number of authors: Gregory Butler, John McKay, Gordon Royle, Alexander Hulpke, John Cannon, Gareth Tracy, and Derek Holt.

The list of transitive groups up to degree 11 was published in [BM83], the list of degree 12 was published in [Roy87], degree 14 and 15 were published in [But93] and degrees 16–30 were published in [Hul96] and [Hul05]. Degree 32 was published in [CH08], degrees 34-46 have been obtained by Derek Holt and Gordon Royle in [HR20], and degree 48 is due to Derek Holt, Gordon Royle, and Gareth Tracy in [HRT]. Groups of prime degree of course are primitive and were known long before.

The transitive groups library will contain representatives for all transitive permutation groups of degree at most 47. Not all degrees might be yet available in the current release. Due to the total number, degree 32 and 48 need to be downloaded separately, see section "Supplemental Downloads" below.

Two permutations groups of the same degree are considered to be equivalent, if there is a renumbering of points, which maps one group into the other one. In other words, if they lie in the same conjugacy class under operation of the full symmetric group by conjugation.

1.1 Transitive Permutation Groups

1 ▶ `TransitiveGroupsAvailable(` *deg* `)` F

returns whether the transitive groups groups of degree *deg* are available for use. This function should be used to test for the scope of the library available.

2 ▶ `TransitiveGroup(` *deg*`,` *nr* `)` F

returns the *nr*-th transitive group of degree *deg*. Both *deg* and *nr* must be positive integers. The transitive groups of equal degree are sorted with respect to their size, so for example `TransitiveGroup(` *deg*`, 1 )` is a transitive group of degree and size *deg*, e.g, the cyclic group of size *deg*, if *deg* is a prime.

3 ▶ `NrTransitiveGroups(` *deg* `)` F

returns the number of transitive groups of degree *deg* stored in the library of transitive groups. The function returns `fail` if *deg* is beyond the range of the library.

The arrangement and the names of the groups of degree up to 15 is the same as given in [CHM98]. With the exception of the symmetric and alternating group (which are represented as `SymmetricGroup` and `AlternatingGroup`) the generators for these groups also conform to this paper with the only difference that 0 (which is not permitted in GAP for permutations to act on) is always replaced by the degree.

The arrangement for all degrees is intended to be equal to the arangement within the system Magma, thus it should be safe to refer to particular (classes of) groups by their index numbers.

```
gap> TransitiveGroup(10,22);
S(5)[x]2
gap> l:=AllTransitiveGroups(NrMovedPoints,12,Size,1440,IsSolvable,false);
[ S(6)[x]2, M_10.2(12)=A_6.E_4(12)=[S_6[1/720]{M_10}S_6]2 ]
gap> List(l,IsSolvable);
[ false, false ]
```

4 ► `TransitiveIdentification( G )` A

Let G be a permutation group, acting transitively on a set of up to 30 points. Then `TransitiveIdentification` will return the position of this group in the transitive groups library. This means, if G acts on m points and `TransitiveIdentification` returns n, then G is permutation isomorphic to the group `TransitiveGroup(m,n)`.

Note: The points moved do **not** need to be $[1..n]$, the group $\langle (2,3,4),(2,3)\rangle$ is considered to be transitive on 3 points. If the group has several orbits on the points moved by it the result of `TransitiveIdentification` is undefined.

```
gap> TransitiveIdentification(Group((1,2),(1,2,3)));
2
```

5 ► `MinimalTransitiveIndices( deg )` F

returns a list of indices of the transitive groups of degree *deg*, which are minimally trransitive, i.e. for which every proper subgroup is not transitive.

```
gap> MinimalTransitiveIndices(12);
[ 1, 2, 3, 4, 5, 7, 9, 17, 31, 34, 40, 46, 47, 57, 162, 166, 246 ]
```

1.2 Selection Functions

1 ► `AllTransitiveGroups( fun1, val1, ... )` F
► `AllLibraryTransitiveGroups( fun1, val1, ... )` F
► `OneTransitiveGroup( fun1, val1, ... )` F

These functions take an arbitrary number of pairs (but at least one pair) of arguments. The first argument in such a pair is a function that can be applied to the groups in the library, and the second argument is either a single value that this function must return in order to have this group included in the selection, or a list of such values. It returns all (ore one) group satisfying the parameters:

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,         [1..100],
>                        IsAbelian,    false    );
```

returns a list of all transitive groups with degree between 10 and 15 and size less than 100 that are not abelian.

Thus the `AllTransitiveGroups` behaves as if it was implemented by a function similar to the one defined below, where `TransitiveGroupsList` is a list of all transitive groups. (Note that in the definition below we assume for simplicity that `AllTransitiveGroups` accepts exactly 4 arguments. It is of course obvious how to change this definition so that the function would accept a variable number of arguments.)

```
AllTransitiveGroups := function( fun1, val1, fun2, val2 )
local    groups, g, i;
  groups := [];
  for i  in [ 1 .. Length( TransitiveGroupsList ) ] do
    g := TransitiveGroupsList[i];
    if      fun1(g) = val1  or IsList(val1) and fun1(g) in val1
        and fun2(g) = val2  or IsList(val2) and fun2(g) in val2
     then
      Add( groups, g );
```

```
      fi;
    od;
    return groups;
  end;
```

Note that the real selection functions are considerably more difficult, to improve the efficiency. Most important, each recognizes a certain set of properties which are precomputed for the library without having to compute them anew for each group. This will substantially speed up the selection process.

The selection functions for the transitive groups library are `AllTransitiveGroups` and `OneTransitiveGroup`. They obtain the following properties from the database without having to compute them anew:

`NrMovedPoints`, `Size`, `Transitivity`, and `IsPrimitive`.

The function `AllLibraryTransitiveGroups` works the same way as `AllTransitiveGroups` but does not warn if no degree is specified.

1.3 Interfacing with the library

Only access using the functions described in this manual is promised to remain stable. Code that wants to use the transitive groups library should use `TransitiveGroupsAvailable` to establish the data being installed for the desired degree. (This function has a dummy equivalent in the main GAP library so that it is always available to return `false`.) Then `NrTransitiveGroups` should be used to determine the range of valid indices for the given degree. Routines should not try to access data structures of the library directly.

1.4 Supplemental Downloads

There are almost 3 million groups of degree 32 and these groups require over a GB of disk space when uncompressed. These groups are therefore not part of the package distribution, but are made available as a supplemental download at

`https://www.math.colostate.edu/~hulpke/transgrp/trans32.tgz`

Simply unpack this archive (which is a tar archive compressed with gzip – consult your computer administrator on the correct way of unpacking) in the top folder of the `transgrp` package (this folder will contain the `PackageInfo.g` file, it will create a folder dat32 containing the groups of degree 32. Once you restart GAP it will automatically recognize the existence of these groups.

The 195826352 groups of degree 48 take over 30GB of disk space in compressed form and thus require a particular download:

Create a folder `dat48` in the top folder of the `transgrp` package. Then download the 11 files `Trans48Part....tar` from

`https://zenodo.org/record/5935751`

and unpack them into this folder. **Do not uncompress the resulting `.gz` files** (You can delete the `.tar` archive files afterwards.) Once GAP is restarted these groups will be available, but `TransitiveIdentification` (for groups not created from the library) or the `AllTransitiveGroups` selector will not work.

Conversion of these groups to GAP is due in part to Jesse Lansdown, whose help is greatly appreciated

Due to the compressed storage, it is perceivable that the groups of degree 48 do not work under Windows. This has not been tested and no promise for this particular functionality is made.

Bibliography

[BM83] G. Butler and J. McKay. The transitive groups of degree up to eleven. *Comm. Algebra*, 11(8):863–911, 1983.

[But93] G. Butler. The transitive groups of degree fourteen and fifteen. *J. Symbolic Comput.*, 16(5):413–422, 1993.

[CH08] John J. Cannon and Derek F. Holt. The transitive permutation groups of degree 32. *Experiment. Math.*, 17(3):307–314, 2008.

[CHM98] J. H. Conway, A. Hulpke, and J. McKay. On transitive permutation groups. *LMS J. Comput. Math.*, 1:1–8 (electronic), 1998.

[HR20] Derek Holt and Gordon Royle. A census of small transitive groups and vertex-transitive graphs. *J. Symbolic Comput.*, 101:51–60, 2020.

[HRT] Derek Holt, Gordon Royle, and Gareth Tracy. The transitive groups of degree 48 and some applications. *J. Algebra*.

[Hul96] A. Hulpke. *Konstruktion transitiver Permutationsgruppen*. Dissertation, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1996.

[Hul05] A. Hulpke. Constructing transitive permutation groups. *J. Symbolic Comput.*, 39(1):1–30, 2005.

[Roy87] G. F. Royle. The transitive groups of degree twelve. *J. Symbolic Comput.*, 4(2):255–268, 1987.

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

`AllLibraryTransitiveGroups`, 4
`AllTransitiveGroups`, 4

I

Interfacing with the library, *5*

M

`MinimalTransitiveIndices`, 4

N

`NrTransitiveGroups`, 3

O

`OneTransitiveGroup`, 4

S

Selection Functions, *4*
Supplemental Downloads, *5*

T

`TransitiveGroup`, 3
`TransitiveGroupsAvailable`, 3
`TransitiveIdentification`, 4
Transitive Permutation Groups, *3*