transgrp

Library of transitive Groups

by

Alexander Hulpke

Department of Mathematics
Colorado State University

Contents

1 The Library of Transitive Groups

1.1 Transitive Permutation Groups

The transitive groups library currently contains representatives for all transitive permutation groups of degree at most 30. 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 ▶ `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.

2 ▶ `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.

This library was computed by Gregory Butler, John McKay, Gordon Royle and Alexander Hulpke. 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 [Butler93] and degrees 16–30 were published in [Hulpke96] and [HulpkeTG]. (Groups of prime degree of course are primitive and were known long before.)

The arrangement and the names of the groups of degree up to 15 is the same as given in [ConwayHulpkeMcKay98]. 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.

```
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 ]
```

3 ▶ `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
```

1.2 Selection Functions

1 ► `AllTransitiveGroups(` *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`.

Bibliography

[BM83] Gregory Butler and John McKay. The transitive groups of degree up to 11. *Comm.Alg.*, 11:863–911, 1983.

[But93] Gregory Butler. The transitive groups of degree fourteen and fifteen. *J.Symb.Comp.*, pages 413–422, 1993.

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

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

[Hul05] Alexander Hulpke. Constructing transitive permutation groups. *J.Symb.Comp.*, 39:1–30, 2005.

[Roy87] Gordon F. Royle. The transitive groups of degree twelve. *J.Symb.Comp.*, pages 255–268, 1987.