GrSyLi

GrSyLi Graphs with Symmetries library

Version 0.1

1 January 2024

Rhys J. Evans
Antonio Montero
Primož Potočnik

Rhys J. Evans Email: rhysjevans00@gmail.com
Homepage: https://rhysje00.github.io/

Antonio Montero Email: antonio.montero@fmf.uni-lj.si
Homepage: https://anteromontonio.github.io/

Primož Potočnik Email: primoz.potocnik@fmf.uni-lj.si
Homepage: https://users.fmf.uni-lj.si/potocnik/work.htm

Copyright

Acknowledgements

Acknowledgements TODO

Contents

Chapter 1

The GrSyLi package

This is the manual for the GrSyLi package version 0.1.

The GrSyLi package contains various collections of graphs with interesting symmetry properties. Each collection of graphs are attained from complete or partial enumerations published in international journals. The papers containing these enumerations are referenced throughout this document, and their authors are included as contributors (unless they are an author of this package). The GrSyLi package provides functionality enabling easy access to these graphs, along with several precomputed properties related to many of the graphs stored within.",

Currently, all of the functions in this package deal with finite graphs in Digraphs format [DBJM+19], and much of the functionality provided is based on the very nice code found in (**Digraphs: Visualising and IO**).

1.1 Installing GrSyLi

To install the GrSyLi package, you first need to download or clone the package, found at https://rhysje00.github.io/grsyli/. Once downloaded, there is no need for any extra compiling. If possible, it is recommended that you place the GrSyLi directory into the package directory of your GAP distribution. For guidance on how to load GrSyLi package if you cannot move the directory directly into your GAP package directory, see Section 1.3.

The GrSyLi package requires the following GAP packages:

- GAPDoc [LN19], version 1.6.6 or higher;
- Digraphs [DBJM+19], version 1.6.1 or higher.

Each of the above packages are part of the standard GAP distribution.

1.2 Building the documentation for GrSyLi

Once downloaded, you can build all package documentation by running `gap path/to/grsyli/makedoc.g` from the command line. This will build a pdf version of this manual and save it in the directory `path/to/grsyli/doc/`, as well as provide the GAP help viewer with the content of the manual.

1.3 Loading GrSyLi

If the GrSyLi package has been downloaded and placed in the GAP package directory, it can by loaded at the GAP prompt by typing the following.

```
gap> LoadPackage("grsyli");
true
```

Otherwise, you can follow the methods found in the GAP reference manual, chapter 76. For example, if `path/to/grsyli` is the path to the GrSyLi package directory relative to your GAP session, it can be loaded at the GAP prompt by typing the following.

```
gap> SetPackagePath("grsyli","path/to/grsyli");
gap> LoadPackage("grsyli");
true
```

1.4 Citing GrSyLi

If you use the GrSyLi package in your research, please tell us about it by emailing `rhysjevans00@gmail.com`. We are interested in any research involving the use of the GrSyLi package and might refer to your work in the future. If you wish to refer to the GrSyLi package in a published work, please cite GrSyLi like a journal article. The following is a BibTeX entry for the current GrSyLi version:

```
@Manual{cvt,
        author = {Evans, Rhys J. and Montero, Antonio and
                  Poto\v{c}nik, Primo\v{z}},
        key = {cvt},
        title = {{GrSyLi -- GRaphs with SYmmetries LIbrary for GAP,
                  Version 0.1}},
        url = {\verb+(https://rhysje00.github.io/grsyli/)+},
        year = {2024}
```

Chapter 2

The cubic vertex-transitive graph library

In this Chapter we give functions for accessing the cubic vertex-transitive graphs stored in this package, and related properties. Currently, this package contains all connected cubic vertex-transitive graphs on up to 1280 vertices. For information and references on these graphs, see [PSV13b].

Let Γ be a simple graph (undirected, loopless, without multiple edges). Then Γ is *cubic* (or *trivalent*, or 3*-valent*) if each vertex of the graph has exactly 3 neighbours.

The graph Γ is *vertex-transitive* if the automorphism group of Γ acts transitively on the vertices of Γ.

2.1 Accessing the cubic vertex-transitive graphs

In this Section we introduce functions for the access to the cubic vertex-transitive graphs stored in the GrSyLi package.

2.1.1 CubicVTGraph

▷ `CubicVTGraph(n, i[, data])` (function)

Returns: A digraph.

Given positive integers `n,i`, this function returns the `i`th cubic vertex-transitive graph with `n` vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graph returned by this function will have been assigned the precomputed properties and attributes from `SetCubicVTGraphProps` (2.2.4). If `data=false` or not specified, none of these properties or attributes are given to the resulting graph.

Example

```
gap> CubicVTGraph(50,4);
<immutable symmetric digraph with 50 vertices, 150 edges>
gap> CubicVTGraph(50,10);
fail
```

2.1.2 AllCubicVTGraphs

▷ `AllCubicVTGraphs(n[, data])` (function)

Returns: A list

Given a positive integer `n`, this function returns a list containing all cubic vertex-transitive graphs with `n` vertices available in this package. If there are no such graphs, the function returns `fail`.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetCubicVTGraphProps` (2.2.4). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> gammas:=AllCubicVTGraphs(50,true);;
gap> List(gammas,IsSPXDigraphCVT);
[ false, false, false, false, false, false, false, false, false ]
gap> gammas:=AllCubicVTGraphs(100,true);;
gap> ForAny(gammas,IsSPXDigraphCVT);
true
```

2.1.3 IteratorOfCubicVTGraphs

▷ `IteratorOfCubicVTGraphs(n[, data])` (function)

Returns: A list

Given a positive integer `n`, this function returns an iterator over all cubic vertex-transitive graphs with `n` vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetCubicVTGraphProps` (2.2.4). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> cnt:=0;; iter:=IteratorOfCubicVTGraphs(1152,true);
<iterator>
gap> for gamma in iter do
> if IsSPXDigraphCVT(gamma) then cnt:=cnt+1; fi;
> od;
gap> cnt;
6
```

2.2 Properties of the cubic vertex-transitive graphs and library

In this Section we give the functions which give information abuot the cubic vertex-transitive graphs library, and the properties and attributes of the graphs it contains.

2.2.1 Precomputed attributes of the cubic vertex-transitive graphs

For a given cubic vertex-transitive graph Γ stored in this package, there are several precomputed attributes stored in the GrSyLi package. These include the following:

- The diameter of Γ (`DigraphDiameter` (**Digraphs: DigraphDiameter**)).
- The girth of the Γ (`DigraphUndirectedGirth` (**Digraphs: DigraphUndirectedGirth**)).
- The bipartiteness of Γ (`IsBipartiteDigraph` (**Digraphs: IsBipartiteDigraph**)).
- The Cayleyness of Γ (`IsCayleyDigraphCVT` (2.2.7)).
- The arc-transitiveness of Γ (`IsArcTransitiveDigraph` (2.2.8)).
- The property of Γ being isomorphic to a a split Praeger-Xu graph (`IsSPXDigraphCVT` (2.2.9)).
- The solvability of the automorphism group of Γ (`HasSolvableAutGroup` (2.2.10)).

Now we introduce functions which are used to find information about the library and each of the graphs it stores.

2.2.2 NrCubicVTGraphs

▷ `NrCubicVTGraphs(`*n*`)` (function)
▷ `NumberCubicVTGraphs(`*n*`)` (function)

Returns: An integer

Given a positive integer *n*, this function returns the number of cubic vertex-transitive graphs with *n* vertices stored in this package.

For any positive integers n up to 1280, the current package stores all cubic vertex-transitive graphs with n vertices.

Example

```
gap> NrCubicVTGraphs(50);
9
```

2.2.3 IdOfCubicVTGraph

▷ `IdOfCubicVTGraph(`*gamma*`)` (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllCubicVTGraphs(n)` and the `i`th graph found when iterating through `IteratorOfCubicVTGraphs(n)`.

Example

```
gap> gamma:=CubicVTGraph(8,2);;
gap> IdOfCubicVTGraph(gamma);
2
gap> gamma:=CycleDigraph(8);;
gap> IdOfCubicVTGraph(gamma);
fail
```

2.2.4 SetCubicVTGraphProps

▷ `SetCubicVTGraphProps(`*gamma*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package. This includes

- All properties and attributes found in Subsection 2.2.1.
- `IsCubicDigraph` (2.2.6).
- `IsVertexTransitive` (**Digraphs: IsVertexTransitive**).

Example

```
gap> gamma:=CompleteDigraph(4);;
gap> SetCubicVTGraphProps(gamma);
gap> IsArcTransitiveDigraph(gamma);
true
```

2.2.5 SmallCubicVTGraphsInfo

▷ `SmallCubicVTGraphsInfo(`*n*`)` (function)

Returns:

Given a positive integer *n*, this function prints information on the cubic vertex-transitive graphs on *n* vertices currently stored in this package. This includes the total number of graphs, the enumeration status of these graphs, and the number of graphs with several properties precomputed and stored in this library.

Example

```
gap> SmallCubicVTGraphsInfo(50);
CVT: There are 9 cubic vertex-transitive graphs on 50 vertices.

Of these 9 graphs, there are;
   - 6 bipartite graphs,
   - 8 Cayley graphs,
   - 1 arc-transitive graphs,
   - 0 split Praeger-Xu graphs,
    - 9 graphs with solvable automorphism group.
```

2.2.6 IsCubicDigraph

▷ IsCubicDigraph(*gamma*) (property)

Returns: `true` or `false`.

Given a digraph *gamma*, this function returns `true` if *gamma* is a cubic digraph, and otherwise it returns `false`.

Example
```
gap> IsCubicDigraph(CycleDigraph(5));
false
gap> IsCubicDigraph(CompleteDigraph(4));
true
```

2.2.7 IsCayleyDigraphCVT

▷ IsCayleyDigraphCVT(*gamma*) (property)

Returns: `true` or `false`.

Given a cubic vertex-transitive graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns `true` if *gamma* is a Cayley (di)graph and `false` otherwise.

The properties and attributes of a cubic vertex-transitive graph that can be found in the GrSyLi package can be assigned using the function `SetCubicVTGraphProps` (2.2.4), or loaded automatically by the functions `CubicVTGraph` (2.1.1), `AllCubicVTGraphs` (2.1.2) or `IteratorOfCubicVTGraphs` (2.1.3). This property is not equivalent to `IsCayleyDigraph` (**Digraphs: IsCayleyDigraph**), as a digraph with this property has associated group and generators.

Example
```
gap> gamma:=CubicVTGraph(50,2,true);;
gap> IsCayleyDigraphCVT(gamma);
true
gap> gamma:=CubicVTGraph(50,9,true);;
gap> IsCayleyDigraphCVT(gamma);
false
```

2.2.8 IsArcTransitiveDigraph

▷ IsArcTransitiveDigraph(*gamma*) (property)

Returns: `true` or `false`.

Given a digraph *gamma*, this function returns `true` if *gamma* is arc-transitive and `false` otherwise. This is a synonym for the function `IsEdgeTransitive` (**Digraphs: IsEdgeTransitive**).

Example

```
gap> gamma:=CubicVTGraph(50,4,true);;
gap> IsArcTransitiveDigraph(gamma);
false
gap> gamma:=CubicVTGraph(50,8,true);;
gap> IsArcTransitiveDigraph(gamma);
true
```

2.2.9 IsSPXDigraphCVT

▷ `IsSPXDigraphCVT(`*gamma*`)` (property)

Returns: `true` or `false`.

Given a cubic vertex-transitive graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns `true` if *gamma* is a split Praeger-Xu graph and `false` otherwise.

The properties and attributes of a cubic vertex-transitive graph that can be found in the GrSyLi package can be assigned using the function `SetCubicVTGraphProps` (2.2.4), or loaded automatically by the functions `CubicVTGraph` (2.1.1), `AllCubicVTGraphs` (2.1.2) or `IteratorOfCubicVTGraphs` (2.1.3).

Example

```
gap> gamma:=CubicVTGraph(48,6,true);;
gap> IsSPXDigraphCVT(gamma);
false
gap> gamma:=CubicVTGraph(48,7,true);;
gap> IsSPXDigraphCVT(gamma);
true
```

2.2.10 HasSolvableAutGroup

▷ `HasSolvableAutGroup(`*gamma*`)` (property)

Returns: `true` or `false`.

Given a digraph *gamma*, this function returns `true` if *gamma* has a solvable automorphism group, and `false` otherwise.

Example

```
gap> gamma:=CubicVTGraph(102,15,true);;
gap> HasSolvableAutGroup(gamma);
true
gap> gamma:=CubicVTGraph(102,16,true);;
gap> HasSolvableAutGroup(gamma);
false
```

Chapter 3

The arc-transitive 2-valent digraph library

In this Chapter we give functions for accessing the arc-transitive 2-valent asymmetric digraphs stored in this package, and related properties. Currently, this package contains all arc-transitive 2-valent connected asymmetric digraphs on up to 1000 vertices. For information and references about these digraphs, see [PSV13a].

Let Γ be a digraph. Then, Γ is *asymmetric* if it has no arcs for which the opposite arc is also in Γ, and is *connected* if the underlying graph of Γ is connected. The digraph Γ is *arc-transitive* if the automorphism group of Γ acts transitively on the arcs of Γ. Further, the digraph Γ is 2*-valent* if each vertex of Γ has 2 in-neighbours and 2 out-neighbours.

3.1 Global variables for the arc-transitive 2-valent digraph library

In this Section we introduce functions and variables which give information about the arc-transitive 2-valent digraph library and the graphs stored within.

3.1.1 ATD_2VALENT_ORDER_MAX

▷ `ATD_2VALENT_ORDER_MAX` (global variable)

This variable stores the largest value n for which the current package contains all arc-transitive 2-valent digraphs with at most n vertices. The number of arc-transitive 2-valent digraphs stored for each n is stored in the list `ATD_2VALENT_NUMBERS` (3.1.2). This variable is currently set to 1000.

Example
```
gap> ATD_2VALENT_ORDER_MAX;
1000
```

3.1.2 ATD_2VALENT_NUMBERS

▷ `ATD_2VALENT_NUMBERS` (global variable)

This variable stores the number of arc-transitive 2-valent digraphs stored in this package

For a positive integer n, ATD_2VALENT_NUMBERS[n] is the number of arc-transitive 2-valent graphs available in this package.

Example
```
gap> ATD_2VALENT_NUMBERS[1000];
182
```

3.1.3 ATD_2VALENT_INFO

▷ ATD_2VALENT_INFO (global variable)

This variable stores various properties of the arc-transitive 2-valent digraphs stored in this package.

For each digraph Γ stored in this package on n vertices and with index i (the position in which Γ is stored relative to its number of vertices), ATD_2VALENT_INFO[n,i] contains a list with the following entries:

1. The self-oppositeness of Γ (see IsSelfOppositeDigraph (3.3.1)).
2. The index of the opposite graph of Γ (see IdOfOppositeDigraph (3.3.5)).
3. The arc-transitiveness of the underlying graph of Γ (see HasATUnderlyingGraph (3.3.2)).
4. The underlying graph of Γ. This is a string giving the position of the underlying graph in the list of GWG graphs GHAT graphs or HAT graphs (see NameOfUnderlyingGraph (3.3.6)).
5. The maximum s such that Γ is s-arc-transitive (see MaximumArcTransitiveness (3.3.7)).
6. The abelian-ness of a vertex stabiliser of Γ (see HasAbelianVertexStabilizer (3.3.3)).
7. $|T_v : G_v|$ of a vertex of Γ (see StabIndexMinATUnd (3.3.8)).
8. $|A_v : G_v|$ of a vertex of Γ (see StabIndexUnd (3.3.9)).
9. The solvability of the automorphism group of Γ (see HasSolvableAutGroup (2.2.10)).
10. The radius of Γ (see AlterCycleRadius (3.3.10)).
11. The attachment number of Γ (see AlterCycleAttachmentNo (3.3.11)).
12. The attachment type of Γ (see AlterCycleAttachmentType (3.3.12)).
13. The number of alternating cycles in Γ (see NrAlterCycles (3.3.13)).
14. The alter-exponent of Γ (see AlterExponent (3.3.14)).
15. The alter-perimeter of Γ (see AlterPerimeter (3.3.15)).
16. The alter-sequence of Γ (see AlterSequence (3.3.16)).

17. The generalised wreath digraph-ness of Γ (see `IsGeneralizedWreathDigraph` (3.3.4)).

Example
```
gap> ATD_2VALENT_INFO[8,1];
[ true, 1, true, "GWG(4;1)", 1, true, 2, 72, true, 4, 8, "---", 2, 1, 2,
[ 4 ], false ]
```

3.1.4 NrAT2ValentDigraphs

▷ `NrAT2ValentDigraphs(n)` (function)
▷ `NumberAT2ValentDigraphs(n)` (function)

Returns: An integer

Given a positive integer *n*, this function returns the number of arc-transitive 2-valent digraphs with *n* vertices stored in this package.

For any positive integers n up to 1000, the current package stores all arc-transitive 2-valent digraphs with n vertices.

Example
```
gap> NrAT2ValentDigraphs(500);
63
```

3.1.5 IdOfAT2ValentDigraph

▷ `IdOfAT2ValentDigraph(gamma)` (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllAT2ValentDigraphs(n)` and the `i`th graph found when iterating through `IteratorOfAT2ValentDigraphs(n)`.

Example
```
gap> gamma:=AT2ValentDigraph(100,5);;
gap> IdOfAT2ValentDigraph(gamma);
5
```

3.1.6 SetAT2ValentDigraphPropsNC

▷ `SetAT2ValentDigraphPropsNC(gamma, n, i)` (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma*. The properties and attribute set are the same as by the function `SetAT2ValentDigraphProps` (3.1.7).

Example

```
gap> gamma:=AT2ValentDigraph(150,5);;
gap> SetAT2ValentDigraphPropsNC(gamma,150,5);
gap> IsGeneralizedWreathDigraph(gamma);
false
```

3.1.7 SetAT2ValentDigraphProps

▷ `SetAT2ValentDigraphProps(`*gamma*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package. This includes

- All properties and attributes found in `ATD_2VALENT_INFO` (3.1.3).
- `IsArcTransitiveDigraph` (2.2.8).
- `IsVertexTransitive` (**Digraphs: IsVertexTransitive**).
- `DigraphReverse` (**Digraphs: DigraphReverse**), which stores the opposite graph of *gamma* with it's library id as defined in `IdOfAT2ValentDigraph` (3.1.5)).
- `DigraphSymmetricClosure` (**Digraphs: DigraphSymmetricClosure**), which stores the underlying graph of *gamma* as described in `NameOfUnderlyingGraph` (3.3.6).

Example

```
gap> gamma:=AT2ValentDigraph(150,5);;
gap> SetAT2ValentDigraphProps(gamma);
gap> IsGeneralizedWreathDigraph(gamma);
false
```

3.2 Accessing the arc-transitive 2-valent digraphs

In this Section we introduce functions for the access to the cubic vertex-transitive graphs stored in the GrSyLi package.

3.2.1 AT2ValentDigraph

▷ `AT2ValentDigraph(`*n*, *i*[, *data*]`)` (function)

Returns: A digraph.

Given positive integers *n*,*i*, this function returns the *i*th arc-transitive 2-valent digraph with *n* vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*=`true`, the graph returned by this function will have been assigned the precomputed properties and attributes from SetAT2ValentDigraphProps (3.1.7). If *data*=false or not specified, none of these properties or attributes are given to the resulting graph.

Example
```
gap> AT2ValentDigraph(300,40);
<immutable digraph with 300 vertices, 600 edges>
gap> AT2ValentDigraph(300,100);
fail
```

3.2.2 AllAT2ValentDigraphs

▷ AllAT2ValentDigraphs(*n*[, *data*]) (function)

Returns: A list

Given a positive integer *n*, this function returns a list containing all arc-transitive digraphs with *n* vertices available in this package. If there are no such graphs, the function returns fail.

When the optional argument *data* is specified, it must have value true or false. If *data*=true, the graphs returned by this function will have been assigned the precomputed properties and attributes from SetAT2ValentDigraphProps (3.1.7). If *data*=false or not specified, none of these properties or attributes are given to the resulting graphs.

Example
```
gap> gammas:=AllAT2ValentDigraphs(300,true);;
gap> ForAny(gammas,HasAbelianVertexStabilizer);
true
```

3.2.3 IteratorOfAT2ValentDigraphs

▷ IteratorOfAT2ValentDigraphs(*n*) (function)

Returns: A list

Given a positive integer *n*, this function returns an iterator over all arc-transitive 2-valent digraphs with *n* vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument *data* is specified, it must have value true or false. If *data*=true, the graphs returned by this function will have been assigned the precomputed properties and attributes from SetAT2ValentDigraphProps (3.1.7). If *data*=false or not specified, none of these properties or attributes are given to the resulting graphs.

Example
```
gap> cnt:=0;; iter:=IteratorOfAT2ValentDigraphs(100,true);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
```

```
15
```

3.3 Properties and attributes of the arc-transitive 2-valent graphs

In this Section we give the properties and attributes of the arc-transitive 2-valent graphs which are stored in this library.

3.3.1 IsSelfOppositeDigraph

▷ `IsSelfOppositeDigraph(`*gamma*`)` (property)

Returns: `true` or `false`.

Given a digraph *gamma*, this function returns `true` if *gamma* is isomorphic to the opposite (or reverse) of *gamma*, and otherwise it returns `false` (see `DigraphReverse` (**Digraphs: DigraphReverse**)).

Example

```
gap> gamma:=AT2ValentDigraph(300,1);;
gap> IsSelfOppositeDigraph(gamma);
false
gap> gamma:=AT2ValentDigraph(300,20);;
gap> IsSelfOppositeDigraph(gamma);
true
```

3.3.2 HasATUnderlyingGraph

▷ `HasATUnderlyingGraph(`*gamma*`)` (property)

Returns: `true` or `false`.

Given a digraph *gamma*, this function returns `true` if the underlying graph of *gamma* is arc-transitive, and otherwise it returns `false` (see `DigraphSymmetricClosure` (**Digraphs: DigraphSymmetricClosure**), `IsEdgeTransitive` (**Digraphs: IsEdgeTransitive**) and `IsArcTransitiveDigraph` (2.2.8)).

Example

```
gap> gamma:=AT2ValentDigraph(300,10);;
gap> HasATUnderlyingGraph(gamma);
false
gap> gamma:=AT2ValentDigraph(300,30,true);;
gap> HasATUnderlyingGraph(gamma);
true
```

3.3.3 HasAbelianVertexStabilizer

▷ `HasAbelianVertexStabilizer(`*gamma*`)` (property)

Returns: `true` or `false`.

Given a digraph *gamma*, this function returns `true` if the stabilizer of a vertex in the automorphism group of *gamma* is abelian, and otherwise it returns `false`.

Example
```
gap> gammas:=AllAT2ValentDigraphs(700,true);;
gap> ForAll(gammas,HasAbelianVertexStabilizer);
true
```

3.3.4 IsGeneralizedWreathDigraph

▷ `IsGeneralizedWreathDigraph(`*gamma*`)` (property)

Returns: `true` or `false`.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns `true` if it is isomorphic to a generalized wreath digraph, and otherwise it returns `false`.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

Let n be an integer such that $n > 2$. The *generalized wreath digraph*, $\overrightarrow{W}_n$, has vertex-set $\mathbb{Z}_n \times \mathbb{Z}_2$. Then, two distinct vertices $(i,a),(j,b)$ are adjacent if and only if $j-i \equiv 1 \pmod{n}$.

For more information on the generalized wreath digraphs, see [PSV15].

Example
```
gap> gamma:=AT2ValentDigraph(700,30,true);;
gap> IsGeneralizedWreathDigraph(gamma);
false
gap> gamma:=AT2ValentDigraph(700,43,true);;
gap> IsGeneralizedWreathDigraph(gamma);
true
```

3.3.5 IdOfOppositeDigraph

▷ `IdOfOppositeDigraph(`*gamma*`)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the position, `i`, at which the opposite (or reverse) of *gamma* is stored (see `DigraphReverse` (**Digraphs: DigraphReverse**)). In particular, if *gamma* has order n, then `AT2ValentDigraph(`n`,i)` is the opposite graph of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

Example
```
gap> gamma:=AT2ValentDigraph(700,10,true);;
gap> IdOfOppositeDigraph(gamma);
9
```

3.3.6 NameOfUnderlyingGraph

▷ `NameOfUnderlyingGraph(`*gamma*`)` (attribute)

Returns: A string.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the name of the underlying graph of *gamma* (see `DigraphSymmetricClosure` (**Digraphs: DigraphSymmetricClosure**)).

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

The string returned will start with 3 or 4 letter, and then contain 2 numbers enclosed in the "[" and "]" characters, separated from eachother by the character ";". In the following we give the 3 possible strings of letters found in these names, and their meaning:

`"GHAT"`
: This means the underlying graph is G-half-arc-transitive for some subgroup of its automorphism group.

`"HAT"`
: This means the underlying graph is half-arc-transitive (i.e. it is G-half-arc-transitive where G is its automorphism group.

`"GWD"`
: This means the underlying graph is a generalized wreath graph.

The underlying graph corresponding to this string is then the graph in the list found in [PSV15] with the same name as the string, and indexed by the remaining two integers in the string.

Example
```
gap> gamma:=AT2ValentDigraph(700,40,true);;
gap> NameOfUnderlyingGraph(gamma);
"GHAT[700;12]"
```

3.3.7 MaximumArcTransitiveness

▷ `MaximumArcTransitiveness(`*gamma*`)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the maximum integer s such that *gamma* is s-arc-transitive.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

An *s-arc* of a digraph Γ is an $(s+1)$-tuple $(v_0, v_1, \ldots, v_s)$ of vertices of Γ, such that (v_{i-1}, v_i) is an arc of Γ for every $i \in \{1, \ldots, s\}$ and $v_{i-1} \neq v_{i+1}$ for every $i \in \{1, \ldots, s-1\}$. A digraph Γ is *s-arc-transitive* if the automorphism group of Γ acts transitively on the set of s-arcs of Γ.

For more information on s-arc-transitive graphs, see TODO REF

Example

```
gap> gamma:=AT2ValentDigraph(748,18,true);;
gap> MaximumArcTransitiveness(gamma);
373
```

3.3.8 StabIndexMinATUnd

▷ `StabIndexMinATUnd(gamma)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the index of a vertex stabilizer in the smallest arc-transitive group of automorphisms of the underlying graph of *gamma*, if defined (see `NameOfUnderlyingGraph` (3.3.6)). Otherwise, this function returns `0`

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

Let Γ be an arc-transitive 2-valent digraph stored in this package, with automorphism group G. Further, let T be the smallest subgroup of the automorphism group of the underlying graph of Γ such that the group is arc-transitive. If $G < T$, then this function returns the index $|T_v : G_v|$ for a vertex v in Γ. Otherwise, it returns 0.

For more information on this attribute, see [PSV15]. TODO add ref?

Example

```
gap> gammas:=AllAT2ValentDigraphs(420,true);;
gap> StabIndexMinATUnd(gammas[20]);
0
gap> StabIndexMinATUnd(gammas[40]);
2
gap> StabIndexMinATUnd(gammas[76]);
8
```

3.3.9 StabIndexUnd

▷ `StabIndexUnd(`*`gamma`*`)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the index of a vertex stabilizer in the automorphism group of the underlying graph of *gamma*. (see `NameOfUnderlyingGraph` (3.3.6)).

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

Let Γ be an arc-transitive 2-valent digraph stored in this package, with automorphism group G. Further, let A be the automorphism group of the underlying graph of Γ. Then this function returns the index $|A_v : G_v|$ for a vertex v in Γ. TODO add ref?

Example
```
gap> gamma:=CubicVTGraph(102,16,true);;
gap> StabIndexUnd(gamma);
3
```

3.3.10 AlterCycleRadius

▷ `AlterCycleRadius(`*`gamma`*`)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the radius of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

For the definition of radius and more information about this parameter, see [PSV15]. TODO add definitions?

Example
```
gap> gamma:=AT2ValentDigraph(48,4,true);;
gap> AlterCycleRadius(gamma);
3
```

3.3.11 AlterCycleAttachmentNo

▷ `AlterCycleAttachmentNo(`*`gamma`*`)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the attachment number of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

For the definition of attachment number and more information about this parameter, see [PSV15]. TODO add definitions?

Example

```
gap> gamma:=AT2ValentDigraph(48,4,true);;
gap> AlterCycleAttachmentNo(gamma);
3
```

3.3.12 AlterCycleAttachmentType

▷ `AlterCycleAttachmentType(`*gamma*`)` (attribute)

Returns: An string.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the attachment number of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

For the definition of attachment type and more information about this parameter, see [PSV15]. TODO add definitions?

Example

```
gap> gamma:=CubicVTGraph(102,16,true);;
gap> AlterCycleAttachmentType(gamma);
3
```

3.3.13 NrAlterCycles

▷ `NrAlterCycles(`*gamma*`)` (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the number of alternating cycles in *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

For the definition of alternating cycles and more information about this parameter, see [PSV15]. TODO add definitions?

Example

```
gap> gamma:=AT2ValentDigraph(100,4,true);;
gap> NrAlterCycles(gamma);
10
```

3.3.14 AlterExponent

▷ AlterExponent(*gamma*) (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the alter-exponent of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function SetAT2ValentDigraphProps (3.1.7), or loaded automatically by the functions AT2ValentDigraph (3.2.1), AllAT2ValentDigraphs (3.2.2) or IteratorOfAT2ValentDigraphs (3.2.3).

For the definition of alter-exponent and more information about this parameter, see [PSV15]. TODO add definitions?

Example
```
gap> gamma:=AT2ValentDigraph(100,4,true);;
gap> AlterExponent(gamma);
1
```

3.3.15 AlterPerimeter

▷ AlterPerimeter(*gamma*) (attribute)

Returns: An integer.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the alter-perimeter of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function SetAT2ValentDigraphProps (3.1.7), or loaded automatically by the functions AT2ValentDigraph (3.2.1), AllAT2ValentDigraphs (3.2.2) or IteratorOfAT2ValentDigraphs (3.2.3).

For the definition of alter-perimeter and more information about this parameter, see [PSV15]. TODO add definitions?

Example
```
gap> gamma:=AT2ValentDigraph(100,4,true);;
gap> AlterPerimeter(gamma);
10
```

3.3.16 AlterSequence

▷ AlterSequence(*gamma*) (attribute)

Returns: A list.

Given an arc-transitive 2-valent digraph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the alter-sequence of *gamma*.

The properties and attributes of an arc-transitive 2-valent digraph that can be found in the GrSyLi package can be assigned using the function `SetAT2ValentDigraphProps` (3.1.7), or loaded automatically by the functions `AT2ValentDigraph` (3.2.1), `AllAT2ValentDigraphs` (3.2.2) or `IteratorOfAT2ValentDigraphs` (3.2.3).

For the definition of alter-sequence and more information about this parameter, see [PSV15]. TODO add definitions?

Example

```
gap> gamma:=AT2ValentDigraph(100,4,true);;
gap> AlterSequence(gamma);
[ 10 ]
```

Chapter 4

The G-half-arc-transitive graph library

In this Chapter we give functions for accessing the G-half-arc-transitive 4 valent graphs stored in this package, and related properties. Currently, this package contains all G-half-arc-transitive 4 valent on up to 1000 vertices. For information and references on these graphs, see [PSV13a].

Let Γ be a simple graph (undirected, loopless, without multiple edges). Then Γ is *4-valent* (or *tetravalent*) if each vertex of the graph has exactly 4 neighbours.

The graph Γ is *G-half-arc-transitive* if G is a group of automorphisms of Γ which acts transitively on the vertices and edges of Γ, but not on the arcs of Γ.

4.1 Global variables for the G-half-arc-transitive 4-valent graph library

In this Section we introduce functions and variables which give information about the G-half-arc-transitive 4-valent graph library and the graphs stored within.

4.1.1 GHAT_4VALENT_ORDER_MAX

▷ `GHAT_4VALENT_ORDER_MAX` (global variable)

This variable stores the largest value n for which the current package contains all G-half-arc-transitive 4-valent graphs with at most n vertices. The number of G-half-arc-transitive 4-valent graphs stored for each n is stored in the list `GHAT_4VALENT_NUMBERS` (4.1.2). This variable is currently set to 1000.

Example

```
gap> GHAT_4VALENT_ORDER_MAX;
1000
```

4.1.2 GHAT_4VALENT_NUMBERS

▷ `GHAT_4VALENT_NUMBERS` (global variable)

This variable stores the number of G-half-arc-transitive 4-valent graphs stored in this package

For a positive integer n, `GHAT_4VALENT_NUMBERS[n]` is the number of G-half-arc-transitive 4-valent graphs available in this package.

Example
```
gap> GHAT_4VALENT_NUMBERS[40];
5
```

4.1.3 GHAT_4VALENT_GRAPH_INFO

▷ `GHAT_4VALENT_GRAPH_INFO` (global variable)

This variable stores various properties of the graphs stored in this package.

For each graph Γ stored in this package on n vertices and with index i (the position in which Γ is stored relative to its number of vertices), `GHAT_4VALENT_GRAPH_INFO[n,i]` contains a list with the following entries:

1. The girth of the Γ (`DigraphUndirectedGirth` (**Digraphs: DigraphUndirectedGirth**)).
2. The bipartiteness of Γ (`IsBipartiteDigraph` (**Digraphs: IsBipartiteDigraph**)).
3. The Cayley type of Γ (`CayleyType` (4.3.2)).
4. The size of the stabilizer of a vertex in the automorphism group of Γ (`SizeStabAut` (4.3.3)).
5. The size of the stabilizer of a vertex in the half-arc-transitive group of automorphisms of Γ (`SizeStabGHATGroups` (4.3.4)).
6. The solvability of the automorphism group of Γ (`HasSolvableAutGroup` (2.2.10)).
7. The consistent cycle types of Γ (`ConsistentCycleTypes` (4.3.5)).

Example
```
gap> GHAT_4VALENT_GRAPH_INFO[24];
[ [ 4, true, "Circ", 4, [ 2, 2, 2 ], true, [ "6s", "8s", "24s" ] ],
[ 4, true, "Circ", 4, [ 2, 2, 2 ], true, [ "8s", "12s", "24s" ] ],
[ 3, false, "Cay", 4, [ 2, 2 ], true, [ "3s", "8s", "12s" ] ],
[ 5, false, "Cay", 4, [ 2, 2 ], true, [ "6s", "8s", "12s" ] ] ]
```

4.1.4 NrGHAT4ValentGraphs

▷ `NrGHAT4ValentGraphs(n)` (function)

▷ `NumberGHAT4ValentGraphs(n)` (function)

Returns: An integer

Given a positive integer `n`, this function returns the number of G-half-arc-transitive 4-valent graphs with `n` vertices stored in this package.

For any positive integers n up to 1000, the current package stores all G-half-arc-transitive 4-valent graphs with n vertices.

Example

```
gap> NrGHAT4ValentGraphs(600);
80
```

4.1.5 IdOfGHAT4ValentGraph

▷ `IdOfGHAT4ValentGraph(`*`gamma`*`)` (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllGHAT4ValentGraphs(n)` and the `i`th graph found when iterating through `IteratorOfGHAT4ValentGraphs(n)`.

Example

```
gap> gamma:=GHAT4ValentGraph(768,20);;
gap> IdOfGHAT4ValentGraph(gamma);
20
```

4.1.6 SetGHAT4ValentGraphPropsNC

▷ `SetGHAT4ValentGraphPropsNC(`*`gamma, n, i`*`)` (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma* with respect to the components of the entry of `GHAT_4VALENT_GRAPH_INFO` (4.1.3) with order *n* and index *i*. The properties and attribute set are the same as by the function `SetGHAT4ValentGraphProps` (4.1.7).

Example

```
gap> gamma:=GHAT4ValentGraph(768,20);;
gap> SetGHAT4ValentGraphPropsNC(gamma,768,20);
gap> SizeStabAut(gamma);
4
```

4.1.7 SetGHAT4ValentGraphProps

▷ `SetGHAT4ValentGraphProps(`*`gamma`*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package. This includes

- All properties and attributes found in `GHAT_4VALENT_GRAPH_INFO` (4.1.3).

- (IsCayleyDigraphCVT (2.2.7)).
- IsVertexTransitive (**Digraphs: IsVertexTransitive**).

Example

```
gap> gamma:=GHAT4ValentGraph(768,20);;
gap> SetGHAT4ValentGraphProps(gamma);
gap> SizeStabAut(gamma);
4
```

4.1.8 SmallGHAT4ValentGraphsInfo

▷ SmallGHAT4ValentGraphsInfo(*n*) (function)

Returns:

Given a positive integer *n*, this function prints information on the G-half-arc-transitive 4-valent graphs on *n* vertices currently stored in this package. This includes the total number of graphs, the enumeration status of these graphs, and the number of graphs with several properties precomputed and stored in this library.

Example

```
gap> SmallGHAT4ValentGraphsInfo(50);
CVT: There are 9 <M>G</M>-half-arc-transitive 4-valent graphs on 50 vertices.

Of these 9 graphs, there are;
   - 6 bipartite graphs,
   - 8 Cayley graphs,
   - 1 arc-transitive graphs,
   - 0 split Praeger-Xu graphs,
   - 9 graphs with solvable automorphism group.
```

4.2 Accessing the G-half-arc-transitive 4-valent graphs

In this Section we introduce functions for the access to the G-half-arc-transitive 4-valent graphs stored in the GrSyLi package.

4.2.1 GHAT4ValentGraph

▷ GHAT4ValentGraph(*n*, *i*[, *data*]) (function)

Returns: A digraph.

Given positive integers *n*,*i*, this function returns the *i*th G-half-arc-transitive 4-valent graph with *n* vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*=`true`, the graph returned by this function will have been assigned the precomputed properties and attributes from SetGHAT4ValentGraphProps (4.1.7). If *data*=false or not specified, none of these properties or attributes are given to the resulting graph.

Example

```
gap> GHAT4ValentGraph(600,20);
<immutable symmetric digraph with 600 vertices, 2400 edges>
gap> GHAT4ValentGraph(600,81);
fail
```

4.2.2 AllGHAT4ValentGraphs

▷ AllGHAT4ValentGraphs(*n*[, *data*]) (function)

Returns: A list

Given a positive integer *n*, this function returns a list containing all G-half-arc-transitive 4-valent graphs with *n* vertices available in this package. If there are no such graphs, the function returns fail.

When the optional argument *data* is specified, it must have value true or false. If *data*=true, the graphs returned by this function will have been assigned the precomputed properties and attributes from SetGHAT4ValentGraphProps (4.1.7). If *data*=false or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> gammas:=AllGHAT4ValentGraphs(600);;
gap> Length(gammas);
80
```

4.2.3 IteratorOfGHAT4ValentGraphs

▷ IteratorOfGHAT4ValentGraphs(*n*) (function)

Returns: A list

Given a positive integer *n*, this function returns an iterator over all G-half-arc-transitive 4-valent graphs with *n* vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument *data* is specified, it must have value true or false. If *data*=true, the graphs returned by this function will have been assigned the precomputed properties and attributes from SetGHAT4ValentGraphProps (4.1.7). If *data*=false or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> cnt:=0;; iter:=IteratorOfGHAT4ValentGraphs(600);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
```

```
57
```

4.3 Properties and attributes of the G-half-arc-transitive 4-valent graphs

In this Section we give the properties and attributes of the G-half-arc-transitive 4-valent graphs which are stored in this library.

4.3.1 Properties from the Digraphs package

A few of the properties we have precomputed for the graphs of this library are already defined in the Digraphs package. These are;

- Diameter, in the attribute `DigraphDiameter` (**Digraphs: DigraphDiameter**).
- Girth, in the attribute `DigraphUndirectedGirth` (**Digraphs: DigraphUndirectedGirth**).
- Bipartiteness, in the property `IsBipartiteDigraph` (**Digraphs: IsBipartiteDigraph**).
- Arc-transitivity, in the property `IsEdgeTransitive` (**Digraphs: IsEdgeTransitive**).

Whenever we wish to store these as GAP properties or attributes, we will use the aforementioned Digraphs functionality. We also give a synonym for arc-transitivity, `IsArcTransitiveDigraph` (2.2.8).

Now we introduce properties and attributes which have been precomputed and stored in the GrSyLi package, and are not defined in the functionality of the Digraphs package.

4.3.2 CayleyType

▷ `CayleyType(`*gamma*`)` (attribute)

Returns: A string.

Given a G-half-arc-transitive 4-valent graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the Cayley type of *gamma*.

Let Γ be a graph. Then `CayleyType(`Γ`)` takes value as one of the following strings:

Circ if Γ is isomorphic to a circulant graph (a Cayley graph defined on a cyclic group).

AbCay
if Γ is isomorphic to a Cayley graph defined on an abelian group, but not isomorphic to a circulant graph.

Cay if Γ is isomorphic to a Cayley graph defined on a nonabelian group, but not isomorphic to a Cayley graph defined on an abelian group.

n-Cay
if Γ is not isomorphic to a Cayley graph.

Example

```
gap> gamma:=GHAT4ValentGraph(768,20,true);;
gap> CayleyType(gamma);
"Cay"
```

4.3.3 SizeStabAut

▷ `SizeStabAut(`*gamma*`)` (attribute)

Returns: An integer.

Given a G-half-arc-transitive 4-valent graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the size of the stabilizer of a vertex in the automorphism group of *gamma*.

The properties and attributes of a G-half-arc-transitive 4-valent graph that can be found in the GrSyLi package can be assigned using the function `SetGHAT4ValentGraphProps` (4.1.7), or loaded automatically by the functions `GHAT4ValentGraph` (4.2.1), `AllGHAT4ValentGraphs` (4.2.2) or `IteratorOfGHAT4ValentGraphs` (4.2.3).

Example

```
gap> gamma:=GHAT4ValentGraph(768,20,true);;
gap> SizeStabAut(gamma);
4
```

4.3.4 SizeStabGHATGroups

▷ `SizeStabGHATGroups(`*gamma*`)` (attribute)

Returns: A list.

Given a G-half-arc-transitive 4-valent graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns a list containing the size of the stabilizer of a vertex in each maximal group H for which *gamma* is H-half-arc-transitive.

The properties and attributes of a G-half-arc-transitive 4-valent graph that can be found in the GrSyLi package can be assigned using the function `SetGHAT4ValentGraphProps` (4.1.7), or loaded automatically by the functions `GHAT4ValentGraph` (4.2.1), `AllGHAT4ValentGraphs` (4.2.2) or `IteratorOfGHAT4ValentGraphs` (4.2.3).

Example

```
gap> gamma:=GHAT4ValentGraph(768,20,true);;
gap> SizeStabGHATGroups(gamma);
[ 2, 2, 2 ]
```

4.3.5 ConsistentCycleTypes

▷ `ConsistentCycleTypes(`*gamma*`)` (attribute)

Returns: A list.

Given a G-half-arc-transitive 4-valent graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the list of consistent cycle types of the graph *gamma*. For a G-consistent cycle `C` in *gamma*, the consistent cycle type of `C` is denoted as a string, starting with length of `C`, and with last character `s` if `C` is G-symmetric, or `c` if `C` is G-chiral.

The properties and attributes of a G-half-arc-transitive 4-valent graph that can be found in the GrSyLi package can be assigned using the function `SetGHAT4ValentGraphProps` (4.1.7), or loaded automatically by the functions `GHAT4ValentGraph` (4.2.1), `AllGHAT4ValentGraphs` (4.2.2) or `IteratorOfGHAT4ValentGraphs` (4.2.3).

Let Γ be a graph with group of automorphisms G, and cycle C. The cycle C is G*-consistent* if there is an element of G which induces a 1-step rotation on C. A G-consistent cycle C is G*-symmetric* if there is an element of G which reverses the orientation of C. A G-consistent cycle C is G*-chiral* if there is no element of G which reverses the orientation of C. For information and references on consistent cycles, see [PSV13a].

Example

```
gap> gamma:=GHAT4ValentGraph(768,20,true);;
gap> ConsistentCycleTypes(gamma);
[ "4s", "6s", "12s" ]
```

Chapter 5

The half-arc-transitive graph library

In this Chapter we give functions for accessing the half-arc-transitive 4-valent graphs stored in this package, and related properties. Currently, this package contains all half-arc-transitive 4 valent on up to 1000 vertices. For information and references on these graphs, see [PSV13a].

Let Γ be a simple graph (undirected, loopless, without multiple edges). Then Γ is *4-valent* (or *tetravalent*) if each vertex of the graph has exactly 4 neighbours.

The graph Γ is *half-arc-transitive* if the automorphism group of Γ acts transitively on the vertices and edges of Γ, but not on the arcs of Γ.

5.1 Global variables for the half-arc-transitive 4-valent graph library

In this Section we introduce functions and variables which give information about the half-arc-transitive 4-valent graph library and the graphs stored within.

5.1.1 HAT_4VALENT_ORDER_MAX

▷ `HAT_4VALENT_ORDER_MAX` (global variable)

This variable stores the largest value n for which the current package contains all half-arc-transitive 4-valent graphs with at most n vertices. The number of half-arc-transitive 4-valent graphs stored for each n is stored in the list `HAT_4VALENT_NUMBERS` (5.1.2). This variable is currently set to 1000.

Example

```
gap> HAT_4VALENT_ORDER_MAX;
1000
```

5.1.2 HAT_4VALENT_NUMBERS

▷ `HAT_4VALENT_NUMBERS` (global variable)

This variable stores the number of half-arc-transitive 4-valent graphs stored in this package

For a positive integer n, `HAT_4VALENT_NUMBERS[n]` is the number of half-arc-transitive 4-valent graphs available in this package.

Example
```
gap> Maximum(HAT_4VALENT_NUMBERS);
185
```

5.1.3 HAT_4VALENT_GRAPH_INFO

▷ `HAT_4VALENT_GRAPH_INFO` (global variable)

This variable stores various properties of the graphs stored in this package.

For each graph Γ stored in this package on n vertices and with index i (the position in which Γ is stored relative to its number of vertices), `HAT_4VALENT_GRAPH_INFO[n,i]` contains a list with the following entries:

1. The girth of the Γ (`DigraphUndirectedGirth` (**Digraphs: DigraphUndirectedGirth**)).
2. The bipartiteness of Γ (`IsBipartiteDigraph` (**Digraphs: IsBipartiteDigraph**)).
3. The Cayleyness of Γ (`IsCayleyDigraphCVT` (2.2.7)).
4. The size of the stabilizer of a vertex in the automorphism group of Γ (`SizeStabAut` (4.3.3)).
5. The solvability of the automorphism group of Γ (`HasSolvableAutGroup` (2.2.10)).
6. The radius of Γ (see `AlterCycleRadius` (3.3.10)).
7. The attachment number of Γ (see `AlterCycleAttachmentNo` (3.3.11)).
8. The attachment type of Γ (see `AlterCycleAttachmentType` (3.3.12)).
9. The alter-exponent of Γ (see `AlterExponent` (3.3.14)).
10. The alter-perimeter of Γ (see `AlterPerimeter` (3.3.15)).
11. The alter-sequence of Γ (see `AlterSequence` (3.3.16)).
12. The length of the shortest consistent cycle in Γ (see `ConsistentCycleTypes` (4.3.5)).
13. The length of the longest consistent cycle in Γ (see `ConsistentCycleTypes` (4.3.5)).
14. The meta-cirulant types of Γ (see `HAT4ValentMetaCirculantClasses` (5.3.2)).

Example
```
gap> HAT_4VALENT_GRAPH_INFO[27];
[ [ 5, false, true, 2, true, 9, 9, "tight", 1, 3, [ 9 ], 6, 9, [ 1, 2 ] ] ]
```

5.1.4 NrHAT4ValentGraphs

▷ `NrHAT4ValentGraphs(n)` (function)
▷ `NumberHAT4ValentGraphs(n)` (function)

Returns: An integer

Given a positive integer *n*, this function returns the number of half-arc-transitive 4-valent graphs with *n* vertices stored in this package.

For any positive integers *n* up to 1000, the current package stores all half-arc-transitive 4-valent graphs with *n* vertices.

Example
```
gap> NrHAT4ValentGraphs(600);
27
```

5.1.5 IdOfHAT4ValentGraph

▷ `IdOfHAT4ValentGraph(gamma)` (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllHAT4ValentGraphs(n)` and the `i`th graph found when iterating through `IteratorOfHAT4ValentGraphs(n)`.

Example
```
gap> gamma:=HAT4ValentGraph(768,20);;
gap> IdOfHAT4ValentGraph(gamma);
20
```

5.1.6 SetHAT4ValentGraphPropsNC

▷ `SetHAT4ValentGraphPropsNC(gamma, n, i)` (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma* with respect to the components of the entry of `HAT_4VALENT_GRAPH_INFO` (5.1.3) with order *n* and index *i*. The properties and attribute set are the same as by the function `SetHAT4ValentGraphProps` (5.1.7).

Example
```
gap> gamma:=HAT4ValentGraph(768,20);;
gap> SetHAT4ValentGraphPropsNC(gamma,768,20);
gap> SizeStabAut(gamma);
8
```

5.1.7 SetHAT4ValentGraphProps

▷ `SetHAT4ValentGraphProps(`*gamma*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package. This includes

- All properties and attributes found in `HAT_4VALENT_GRAPH_INFO` (5.1.3).
- (`IsCayleyDigraphCVT` (2.2.7)).
- `IsVertexTransitive` (**Digraphs: IsVertexTransitive**).

Example

```
gap> gamma:=HAT4ValentGraph(768,20);;
gap> SetHAT4ValentGraphProps(gamma);
gap> SizeStabAut(gamma);
8
```

5.1.8 SmallHAT4ValentGraphsInfo

▷ `SmallHAT4ValentGraphsInfo(`*n*`)` (function)

Returns:

Given a positive integer *n*, this function prints information on the half-arc-transitive 4-valent graphs on *n* vertices currently stored in this package. This includes the total number of graphs, the enumeration status of these graphs, and the number of graphs with several properties precomputed and stored in this library.

Example

```
gap> SmallHAT4ValentGraphsInfo(50);
CVT: There are 9 half-arc-transitive 4-valent graphs on 50 vertices.

Of these 9 graphs, there are;
   - 6 bipartite graphs,
   - 8 Cayley graphs,
   - 1 arc-transitive graphs,
   - 0 split Praeger-Xu graphs,
   - 9 graphs with solvable automorphism group.
```

5.2 Accessing the half-arc-transitive 4-valent graphs

In this Section we introduce functions for the access to the half-arc-transitive 4-valent graphs stored in the GrSyLi package.

5.2.1 HAT4ValentGraph

▷ `HAT4ValentGraph(n, i[, data])` (function)

Returns: A digraph.

Given positive integers `n,i`, this function returns the `i`th half-arc-transitive 4-valent graph with `n` vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graph returned by this function will have been assigned the precomputed properties and attributes from `SetHAT4ValentGraphProps` (5.1.7). If `data=false` or not specified, none of these properties or attributes are given to the resulting graph.

Example

```
gap> HAT4ValentGraph(600,20);
<immutable symmetric digraph with 600 vertices, 2400 edges>
gap> HAT4ValentGraph(600,28);
fail
```

5.2.2 AllHAT4ValentGraphs

▷ `AllHAT4ValentGraphs(n[, data])` (function)

Returns: A list

Given a positive integer `n`, this function returns a list containing all half-arc-transitive 4-valent graphs with `n` vertices available in this package. If there are no such graphs, the function returns `fail`.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetHAT4ValentGraphProps` (5.1.7). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> gammas:=AllHAT4ValentGraphs(600);;
gap> Length(gammas);
27
```

5.2.3 IteratorOfHAT4ValentGraphs

▷ `IteratorOfHAT4ValentGraphs(n)` (function)

Returns: A list

Given a positive integer `n`, this function returns an iterator over all half-arc-transitive 4-valent graphs with `n` vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetHAT4ValentGraphProps` (5.1.7). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> cnt:=0;; iter:=IteratorOfHAT4ValentGraphs(600);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
25
```

5.3 Properties and attributes of the half-arc-transitive 4-valent graphs

In this Section we give the properties and attributes of the half-arc-transitive 4-valent graphs which are stored in this library.

5.3.1 Properties from the Digraphs package

A few of the properties we have precomputed for the graphs of this library are already defined in the Digraphs package. These are;

- Diameter, in the attribute `DigraphDiameter` (**Digraphs: DigraphDiameter**).
- Girth, in the attribute `DigraphUndirectedGirth` (**Digraphs: DigraphUndirectedGirth**).
- Bipartiteness, in the property `IsBipartiteDigraph` (**Digraphs: IsBipartiteDigraph**).
- Arc-transitivity, in the property `IsEdgeTransitive` (**Digraphs: IsEdgeTransitive**).

Whenever we wish to store these as GAP properties or attributes, we will use the aforementioned Digraphs functionality. We also give a synonym for arc-transitivity, `IsArcTransitiveDigraph` (2.2.8).

Now we introduce properties and attributes which have been precomputed and stored in the GrSyLi package, and are not defined in the functionality of the Digraphs package.

5.3.2 HAT4ValentMetaCirculantClasses

▷ `HAT4ValentMetaCirculantClasses(`*gamma*`)` (attribute)

Returns: A list.

Given a half-arc-transitive 4-valent graph *gamma* from in the GrSyLi package such that its properties and attributes have been assigned, this function returns the meta-circulant classes of *gamma*.

Let Γ be a graph. Then the graph Γ is *meta-circulant* if the automorphism group of Γ has metacycli subgroup generated by ρ, σ such that $\langle \rho \rangle$ acts semiregularly on the vertices of Γ. Any half-arc-transitive 4-valent graph could fall into one of four known classes.

For information and references on these classes, see [PSV13a].

Example

```
gap> gamma:=HAT4ValentGraph(768,104,true);;
gap> HAT4ValentMetaCirculantClasses(gamma);
[ 2, 4 ]
```

Chapter 6

The arc-transitive 4-valent graph library

In this Chapter we give functions for accessing the arc-transitive 4-valent graphs stored in this package, and related properties. Currently, this package contains all arc-transitive 4-valent graphs on up to 640 vertices. For information and references about these graphs, see [PSV15] and [PSV13b].

Let Γ be a simple graph (undirected, loopless, without multiple edges). Then Γ is *4-valent* (or *tetravalent*) if each vertex of the graph has exactly 4 neighbours.

The graph Γ is *arc-transitive* if the automorphism group of Γ acts transitively on the arcs of Γ.

6.1 Global variables for the arc-transitive 4-valent graph library

In this Section we introduce functions and variables which give information about the arc-transitive 4-valent graph library and the graphs stored within.

6.1.1 AT_4VALENT_ORDER_MAX

▷ AT_4VALENT_ORDER_MAX (global variable)

This variable stores the largest value n for which the current package contains all arc-transitive 2-valent digraphs with at most n vertices. The number of arc-transitive 2-valent digraphs stored for each n is stored in the list AT_4VALENT_NUMBERS (6.1.2). This variable is currently set to 640.

Example

```
gap> AT_4VALENT_ORDER_MAX;
640
```

6.1.2 AT_4VALENT_NUMBERS

▷ AT_4VALENT_NUMBERS (global variable)

This variable stores the number of arc-transitive 2-valent digraphs stored in this package

For a positive integer n, AT_4VALENT_NUMBERS[n] is the number of arc-transitive 2-valent graphs available in this package.

```
gap> AT_4VALENT_NUMBERS[200];
40
```

6.1.3 NrAT4ValentGraphs

▷ `NrAT4ValentGraphs(n)` (function)
▷ `NumberAT4ValentGraphs(n)` (function)

Returns: An integer

Given a positive integer *n*, this function returns the number of arc-transitive 2-valent digraphs with *n* vertices stored in this package.

For any positive integers n up to 1000, the current package stores all arc-transitive 2-valent digraphs with n vertices.

```
gap> NrAT4ValentGraphs(640);
158
```

6.1.4 IdOfAT4ValentGraph

▷ `IdOfAT4ValentGraph(gamma)` (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllAT4ValentGraphs(n)` and the `i`th graph found when iterating through `IteratorOfAT4ValentGraphs(n)`.

```
gap> gamma:=CompleteDigraph(5);;
gap> IdOfAT4ValentGraph(gamma);
1
gap> gamma:=AT4ValentGraph(50,2);;
gap> IdOfAT4ValentGraph(gamma);
2
```

6.1.5 SetAT4ValentGraphPropsNC

▷ `SetAT4ValentGraphPropsNC(gamma, n, i)` (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma*. The properties and attribute set are the same as by the function `SetAT4ValentGraphProps` (6.1.6).

Example
```
gap> gamma:=AT4ValentGraph(50,2);;
gap> SetAT4ValentGraphPropsNC(gamma,50,2);
gap> IdOfAT4ValentGraph(gamma);
2
```

6.1.6 SetAT4ValentGraphProps

▷ `SetAT4ValentGraphProps(`*gamma*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package.

Example
```
gap> gamma:=CompleteDigraph(5);;
gap> SetAT4ValentGraphProps(gamma);
gap> IdOfAT4ValentGraph(gamma);
1
```

6.2 Accessing the arc-transitive 4-valent graphs

In this Section we introduce functions for the access to the arc-transitive 4-valent graphs stored in the GrSyLi package.

6.2.1 AT4ValentGraph

▷ `AT4ValentGraph(`*n*`, `*i*`[, `*data*`])` (function)

Returns: A digraph.

Given positive integers *n*,*i*, this function returns the *i*th arc-transitive 2-valent digraph with *n* vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*`=true`, the graph returned by this function will have been assigned the precomputed properties and attributes from `SetAT4ValentGraphProps` (6.1.6). If *data*`=false` or not specified, none of these properties or attributes are given to the resulting graph.

Example
```
gap> AT4ValentGraph(640,20);
<immutable symmetric digraph with 640 vertices, 2560 edges>
gap> AT4ValentGraph(640,159);
fail
```

6.2.2 AllAT4ValentGraphs

▷ `AllAT4ValentGraphs(n[, data])` (function)

Returns: A list

Given a positive integer *n*, this function returns a list containing all arc-transitive digraphs with *n* vertices available in this package. If there are no such graphs, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*`=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetAT4ValentGraphProps` (6.1.6). If *data*`=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example
```
gap> gammas:=AllAT4ValentGraphs(640);;
gap> Length(gammas);
158
```

6.2.3 IteratorOfAT4ValentGraphs

▷ `IteratorOfAT4ValentGraphs(n)` (function)

Returns: A list

Given a positive integer *n*, this function returns an iterator over all arc-transitive 2-valent digraphs with *n* vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*`=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetAT4ValentGraphProps` (6.1.6). If *data*`=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example
```
gap> cnt:=0;; iter:=IteratorOfAT4ValentGraphs(640);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
155
```

Chapter 7

The 2-arc-transitive 4-valent graph library

In this Chapter we give functions for accessing the 2-arc-transitive 4-valent graphs stored in this package, and related properties. Currently, this package contains all 2-arc-transitive 4-valent graphs on up to 727 vertices, and many more on up to 2000 vertices. For information and references about these graphs, see [Pot09].

Let Γ be a simple graph (undirected, loopless, without multiple edges). Then Γ is *4-valent* (or *tetravalent*) if each vertex of the graph has exactly 4 neighbours.

The graph Γ is *2-arc-transitive* if the automorphism group of Γ acts transitively on the 2-arcs of Γ.

7.1 Global variables for the 2-arc-transitive 4-valent graph library

In this Section we introduce functions and variables which give information about the 2-arc-transitive 4-valent graph library and the graphs stored within.

7.1.1 2AT_4VALENT_ORDER_MAX

▷ 2AT_4VALENT_ORDER_MAX (global variable)

This variable stores the largest value n for which the current package contains all arc-transitive 2-valent digraphs with at most n vertices. The number of arc-transitive 2-valent digraphs stored for each n is stored in the list 2AT_4VALENT_NUMBERS (7.1.2). This variable is currently set to 2000.

Example
```
gap> 2AT_4VALENT_ORDER_MAX;
2000
```

7.1.2 2AT_4VALENT_NUMBERS

▷ 2AT_4VALENT_NUMBERS (global variable)

This variable stores the number of arc-transitive 2-valent digraphs stored in this package

For a positive integer *n*, 2AT_4VALENT_NUMBERS[*n*] is the number of arc-transitive 2-valent graphs available in this package.

Example

```
gap> Maximum(2AT_4VALENT_NUMBERS);
11
```

7.1.3 Nr2AT4ValentGraphs

▷ Nr2AT4ValentGraphs(*n*) (function)

▷ Number2AT4ValentGraphs(*n*) (function)

Returns: An integer

Given a positive integer *n*, this function returns the number of arc-transitive 2-valent digraphs with *n* vertices stored in this package.

For any positive integers *n* up to 1000, the current package stores all arc-transitive 2-valent digraphs with *n* vertices.

Example

```
gap> Nr2AT4ValentGraphs(1920);
11
```

7.1.4 IdOf2AT4ValentGraph

▷ IdOf2AT4ValentGraph(*gamma*) (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns fail.

The index i of a graph gamma in this library is the position at which the graph is stored relative to its number of vertices. In particular, if gamma has n vertices, then gamma will be the ith entry of All2AT4ValentGraphs(n) and the ith graph found when iterating through IteratorOf2AT4ValentGraphs(n).

Example

```
gap> gamma:=CompleteDigraph(5);;
gap> IdOf2AT4ValentGraph(gamma);
1
gap> gamma:=2AT4ValentGraph(1920,10);;
gap> IdOf2AT4ValentGraph(gamma);
10
```

7.1.5 Set2AT4ValentGraphPropsNC

▷ Set2AT4ValentGraphPropsNC(*gamma*, *n*, *i*) (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma*. The properties and attribute set are the same as by the function Set2AT4ValentGraphProps (7.1.6).

Example

```
gap> gamma:=2AT4ValentGraph(1920,10);;
gap> Set2AT4ValentGraphPropsNC(gamma,1920,10);
gap> IdOf2AT4ValentGraph(gamma);
10
```

7.1.6 Set2AT4ValentGraphProps

▷ Set2AT4ValentGraphProps(*gamma*) (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package.

Example

```
gap> gamma:=2AT4ValentGraph(1920,5);;
gap> Set2AT4ValentGraphProps(gamma);
gap> IdOf2AT4ValentGraph(gamma);
5
```

7.2 Accessing the 2-arc-transitive 4-valent graphs

In this Section we introduce functions for the access to the 2-arc-transitive 4-valent graphs stored in the GrSyLi package.

7.2.1 2AT4ValentGraph

▷ 2AT4ValentGraph(*n*, *i*[, *data*]) (function)

Returns: A digraph.

Given positive integers *n*,*i*, this function returns the *i*th arc-transitive 2-valent digraph with *n* vertices available in this package. If there is no such graph, the function returns fail.

When the optional argument *data* is specified, it must have value true or false. If *data*=true, the graph returned by this function will have been assigned the precomputed properties and attributes from Set2AT4ValentGraphProps (7.1.6). If *data*=false or not specified, none of these properties or attributes are given to the resulting graph.

Example

```
gap> 2AT4ValentGraph(1920,3);
<immutable symmetric digraph with 1920 vertices, 7680 edges>
```

```
gap> 2AT4ValentGraph(1920,12);
fail
```

7.2.2 All2AT4ValentGraphs

▷ All2AT4ValentGraphs(*n*[, *data*]) (function)

Returns: A list

Given a positive integer *n*, this function returns a list containing all arc-transitive digraphs with *n* vertices available in this package. If there are no such graphs, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*`=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `Set2AT4ValentGraphProps` (7.1.6). If *data*`=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> gammas:=All2AT4ValentGraphs(1920);;
gap> Length(gammas);
11
```

7.2.3 IteratorOf2AT4ValentGraphs

▷ IteratorOf2AT4ValentGraphs(*n*) (function)

Returns: A list

Given a positive integer *n*, this function returns an iterator over all arc-transitive 2-valent digraphs with *n* vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*`=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `Set2AT4ValentGraphProps` (7.1.6). If *data*`=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> cnt:=0;; iter:=IteratorOf2AT4ValentGraphs(1920);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
0
```

Chapter 8

The edge-transitive 4-valent graph library

In this chapter we give functions for accessing the edge-transitive 4-valent graphs stored in this package, and related properties. Currently, this package contains an incomplete list of edge-transitive 4-valent graphs on up to 512 vertices. For information and references about these graphs, see [PW16].

Let Γ be a simple graph (undirected, loopless, without multiple edges). Then Γ is *4-valent* (or *tetravalent*) if each vertex of the graph has exactly 4 neighbours.

The graph Γ is *edge-transitive* if the automorphism group of Γ acts transitively on the edges of Γ.

8.1 Global variables for the edge-transitive 4-valent graph library

In this Section we introduce functions and variables which give information about the edge-transitive 4-valent graph library and the graphs stored within.

8.1.1 ET_4VALENT_ORDER_MAX

▷ ET_4VALENT_ORDER_MAX (global variable)

This variable stores the largest value *n* for which the current package contains all arc-transitive 2-valent digraphs with at most *n* vertices. The number of arc-transitive 2-valent digraphs stored for each *n* is stored in the list ET_4VALENT_NUMBERS (8.1.2). This variable is currently set to 512.

Example
```
gap> ET_4VALENT_ORDER_MAX;
640
```

8.1.2 ET_4VALENT_NUMBERS

▷ ET_4VALENT_NUMBERS (global variable)

This variable stores the number of arc-transitive 2-valent digraphs stored in this package

For a positive integer *n*, `ET_4VALENT_NUMBERS[n]` is the number of arc-transitive 2-valent graphs available in this package.

Example
```
gap> Maximum(ET_4VALENT_NUMBERS);
664
```

8.1.3 NrET4ValentGraphs

▷ `NrET4ValentGraphs(n)` (function)
▷ `NumberET4ValentGraphs(n)` (function)

Returns: An integer

Given a positive integer `n`, this function returns the number of arc-transitive 2-valent digraphs with `n` vertices stored in this package.

For any positive integers *n* up to 1000, the current package stores all arc-transitive 2-valent digraphs with *n* vertices.

Example
```
gap> NrET4ValentGraphs(400);
107
```

8.1.4 IdOfET4ValentGraph

▷ `IdOfET4ValentGraph(gamma)` (attribute)

Returns: An integer

Given a digraph `gamma`, if `gamma` is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to `gamma`. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllET4ValentGraphs(n)` and the `i`th graph found when iterating through `IteratorOfET4ValentGraphs(n)`.

Example
```
gap> gamma:=CompleteDigraph(5);;
gap> IdOfET4ValentGraph(gamma);
1
gap> gamma:=ET4ValentGraph(512,40);;
gap> IdOfET4ValentGraph(gamma);
40
```

8.1.5 SetET4ValentGraphPropsNC

▷ `SetET4ValentGraphPropsNC(`*`gamma, n, i`*`)` (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma*. The properties and attribute set are the same as by the function `SetET4ValentGraphProps` (8.1.6).

Example

```
gap> gamma:=ET4ValentGraph(512,40);;
gap> SetET4ValentGraphPropsNC(gamma,512,40);
gap> IdOfET4ValentGraph(gamma);
40
```

8.1.6 SetET4ValentGraphProps

▷ `SetET4ValentGraphProps(`*`gamma`*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package.

Example

```
gap> gamma:=ET4ValentGraph(512,40);;
gap> SetET4ValentGraphProps(gamma);
gap> IdOfET4ValentGraph(gamma);
40
```

8.2 Accessing the edge-transitive 4-valent graphs

In this Section we introduce functions for the access to the edge-transitive 4-valent graphs stored in the GrSyLi package.

8.2.1 ET4ValentGraph

▷ `ET4ValentGraph(`*`n, i[, data]`*`)` (function)

Returns: A digraph.

Given positive integers *n*,*i*, this function returns the *i*th arc-transitive 2-valent digraph with *n* vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*=`true`, the graph returned by this function will have been assigned the precomputed properties and attributes from `SetET4ValentGraphProps` (8.1.6). If *data*=`false` or not specified, none of these properties or attributes are given to the resulting graph.

Example

```
gap> ET4ValentGraph(512,100);
<immutable symmetric digraph with 512 vertices, 2048 edges>
```

```
gap> ET4ValentGraph(512,700);
fail
```

8.2.2 AllET4ValentGraphs

▷ `AllET4ValentGraphs(n[, data])` (function)

Returns: A list

Given a positive integer `n`, this function returns a list containing all arc-transitive digraphs with `n` vertices available in this package. If there are no such graphs, the function returns `fail`.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetET4ValentGraphProps` (8.1.6). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example
```
gap> gammas:=AllET4ValentGraphs(400);;
gap> Length(gammas);
107
```

8.2.3 IteratorOfET4ValentGraphs

▷ `IteratorOfET4ValentGraphs(n)` (function)

Returns: A list

Given a positive integer `n`, this function returns an iterator over all arc-transitive 2-valent digraphs with `n` vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetET4ValentGraphProps` (8.1.6). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example
```
gap> cnt:=0;; iter:=IteratorOfET4ValentGraphs(400);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
106
```

Chapter 9

The locally arc-transitive $\{3,4\}$-valent graph library

In this chapter we give functions for accessing the locally arc-transitive $\{3,4\}$-valent graphs stored in this package, and related properties.

In this package, we collect bipartite biregular graphs of valence $\{3,4\}$ that are locally G-arc-transitive for some group of automorphisms G with the edge stabiliser acting faithfully on the union of the neighbourhoods of the vertices of the edge. Currently, this package stores all such graphs on up to 350 vertices, and many more on up to 1050. For more information and references about these graphs, see [Pot12].

9.1 Global variables for the locally arc-transitive $\{3,4\}$-valent graph library

In this Section we introduce functions and variables which give information about the locally arc-transitive $\{3,4\}$-valent graph library and the graphs stored within.

9.1.1 LAT_34VALENT_ORDER_MAX

▷ LAT_34VALENT_ORDER_MAX (global variable)

This variable stores the largest value n for which the current package contains all arc-transitive 2-valent digraphs with at most n vertices. The number of arc-transitive 2-valent digraphs stored for each n is stored in the list LAT_34VALENT_NUMBERS (9.1.2). This variable is currently set to 1050.

Example
```
gap> LAT_34VALENT_ORDER_MAX;
1050
```

9.1.2 LAT_34VALENT_NUMBERS

▷ LAT_34VALENT_NUMBERS (global variable)

This variable stores the number of arc-transitive 2-valent digraphs stored in this package

For a positive integer *n*, `LAT_34VALENT_NUMBERS[`*n*`]` is the number of arc-transitive 2-valent graphs available in this package.

Example

```
gap> Maximum(LAT_34VALENT_NUMBERS);
249
```

9.1.3 NrLAT34ValentGraphs

▷ `NrLAT34ValentGraphs(`*n*`)` (function)
▷ `NumberLAT34ValentGraphs(`*n*`)` (function)

Returns: An integer

Given a positive integer *n*, this function returns the number of arc-transitive 2-valent digraphs with *n* vertices stored in this package.

For any positive integers n up to 1000, the current package stores all arc-transitive 2-valent digraphs with n vertices.

Example

```
gap> NrLAT34ValentGraphs(896);
249
```

9.1.4 IdOfLAT34ValentGraph

▷ `IdOfLAT34ValentGraph(`*gamma*`)` (attribute)

Returns: An integer

Given a digraph *gamma*, if *gamma* is isomorphic to a graph stored in this package, this function returns the index of the graph isomorphic to *gamma*. Otherwise, this function returns `fail`.

The index `i` of a graph `gamma` in this library is the position at which the graph is stored relative to its number of vertices. In particular, if `gamma` has `n` vertices, then `gamma` will be the `i`th entry of `AllLAT34ValentGraphs(n)` and the `i`th graph found when iterating through `IteratorOfLAT34ValentGraphs(n)`.

Example

```
gap> gamma:=LAT34ValentGraph(896,20);;
gap> IdOfLAT34ValentGraph(gamma);
20
```

9.1.5 SetLAT34ValentGraphPropsNC

▷ `SetLAT34ValentGraphPropsNC(`*gamma, n, i*`)` (function)

Returns:

Given a digraph *gamma*, this function sets the properties and attributes of *gamma*. The properties and attribute set are the same as by the function `SetLAT34ValentGraphProps` (9.1.6).

Example
```
gap> gamma:=LAT34ValentGraph(896,20);;
gap> SetLAT34ValentGraphPropsNC(gamma,896,20);
gap> IdOfLAT34ValentGraph(gamma);
20
```

9.1.6 SetLAT34ValentGraphProps

▷ `SetLAT34ValentGraphProps(`*gamma*`)` (function)

Returns:

Given a digraph *gamma*, if this graph is isomorphic to a graph stored in this library, this function sets the properties and attributes of *gamma* precomputed in this package.

Example
```
gap> gamma:=LAT34ValentGraph(896,20);;
gap> SetLAT34ValentGraphProps(gamma);
gap> IdOfLAT34ValentGraph(gamma);
20
```

9.2 Accessing the locally arc-transitive $\{3,4\}$-valent graphs

In this Section we introduce functions for the access to the locally arc-transitive $\{3,4\}$-valent graphs stored in the GrSyLi package.

9.2.1 LAT34ValentGraph

▷ `LAT34ValentGraph(`*n*`,` *i*`[,` *data*`])` (function)

Returns: A digraph.

Given positive integers *n*,*i*, this function returns the *i*th arc-transitive 2-valent digraph with *n* vertices available in this package. If there is no such graph, the function returns `fail`.

When the optional argument *data* is specified, it must have value `true` or `false`. If *data*`=true`, the graph returned by this function will have been assigned the precomputed properties and attributes from `SetLAT34ValentGraphProps` (9.1.6). If *data*`=false` or not specified, none of these properties or attributes are given to the resulting graph.

Example
```
gap> LAT34ValentGraph(896,200);
<immutable symmetric digraph with 896 vertices, 3072 edges>
gap> LAT34ValentGraph(896,300);
fail
```

9.2.2 AllLAT34ValentGraphs

▷ `AllLAT34ValentGraphs(n[, data])` (function)

Returns: A list

Given a positive integer `n`, this function returns a list containing all arc-transitive digraphs with `n` vertices available in this package. If there are no such graphs, the function returns `fail`.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetLAT34ValentGraphProps` (9.1.6). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> gammas:=AllLAT34ValentGraphs(224);;
gap> Length(gammas);
22
```

9.2.3 IteratorOfLAT34ValentGraphs

▷ `IteratorOfLAT34ValentGraphs(n)` (function)

Returns: A list

Given a positive integer `n`, this function returns an iterator over all arc-transitive 2-valent digraphs with `n` vertices available in this package. If there are such no graphs, the function returns an empty iterator.

When the optional argument `data` is specified, it must have value `true` or `false`. If `data=true`, the graphs returned by this function will have been assigned the precomputed properties and attributes from `SetLAT34ValentGraphProps` (9.1.6). If `data=false` or not specified, none of these properties or attributes are given to the resulting graphs.

Example

```
gap> cnt:=0;; iter:=IteratorOfLAT34ValentGraphs(224);;
gap> for gamma in iter do
> if HasSolvableAutGroup(gamma) then
> cnt:=cnt+1;
> fi;
> od;
gap> cnt;
22
```

References

[DBJM$^+$19] J. De Beule, J. Jonušas, J. D. Mitchell, M. C. Torpey, and W. A. Wilson. Digraphs – a GAP package, Version 0.15.3, 2019. Refereed GAP package, available at https://gap-packages.github.io/Digraphs/. 5

[LN19] F. Lübeck and M. Neunhöffer. GAPDoc – a GAP package, Version 1.6.1, 2019. Refereed GAP package, available at http://www.math.rwth-aachen.de/ Frank.Luebeck/GAPDoc/index.html. 5

[Pot09] Primož Potočnik. A list of 4-valent 2-arc-transitive graphs and finite faithful amalgams of index (4, 2). *European Journal of Combinatorics*, 30(5):1323–1336, 2009. Part Special Issue on Metric Graph Theory. 45

[Pot12] Primož Potočnik. Locally arc-transitive graphs of valence $\{3,4\}$ with trivial edge kernel. *Journal of Algebraic Combinatorics*, 38, 2012. 53

[PSV13a] Primož Potočnik, Pablo Spiga, and Gabriel Verret. A census of 4-valent half-arc-transitive graphs and arc-transitive digraphs of valence two. *Ars Mathematica Contemporanea*, 8, 2013. 13, 26, 33, 34, 39

[PSV13b] Primož Potočnik, Pablo Spiga, and Gabriel Verret. Cubic vertex-transitive graphs on up to 1280 vertices. *Journal of Symbolic Computation*, 50:465–477, 2013. 7, 41

[PSV15] Primož Potočnik, Pablo Spiga, and Gabriel Verret. Bounding the order of the vertex-stabiliser in 3-valent vertex-transitive and 4-valent arc-transitive graphs. *Journal of Combinatorial Theory, Series B*, 111:148–180, 2015. 19, 20, 21, 22, 23, 24, 25, 41

[PW16] Primož Potočnik and Steve Wilson. Recipes for Edge-Transitive Tetravalent Graphs. *The Art of Discrete and Applied Mathematics*, 3, 2016. 49

Index