Kevin I. Piterman and Leandro Vendramin

Computer algebra with GAP

– Monograph –

March 6, 2023

Preface

Implementing new algorithms and the ever-growing power of modern computers have a revolutionary impact on science. Mathematics, whose artistic and creative nature could have been thought to be a barrier to the influence of computers, does not escape the reach of this revolution. Much of the mathematics published nowadays makes extensive use of machine computations. Remarkably, a substantial part of the results could not be obtained without it or would take an unreasonable amount of time. This includes many outstanding theoretical results that depend on computers to verify a hypothesis or perform particular calculations while proving a theorem. The use of computers is also growing in teaching mathematics at all levels. Regarding abstract algebra, one computer algebra system that is hard to overlook is GAP ("Groups, Algorithms and Programming"). It is extensively used when teaching elementary algebra and in advanced scientific literature, testing and disproving conjectures. This is one motivation that led us to write this monograph.

The first version of GAP was released in December 1988, named Version 2.4, and it was a project started by Joachim Beubüser in 1986 at the Lehrstuhl D für Mathematik, RWTH-Aachen University. Indeed, the system was officially presented during an Oberwolfach meeting in May 1988. The GAP system has a kernel written in C, and its interpreter for the programming language belongs to the family of PASCAL languages. It may be regarded as an object-oriented and imperative programming language. It is important to remark that GAP is an open-source program and is available for download at https://www.gap-system.org/. On this webpage, users can access a development repository hosted on GitHub, contribute to the improvement of the software, and ask questions about its functionality.

This monograph includes the following topics: a basic introduction to the language, basic arithmetic, permutations, matrices, polynomial rings, finite fields, finite and finitely presented groups, small groups, group representations and character theory, and simple groups. Advanced topics include testing several open conjectures and theorems. In addition, each chapter ends with an extensive list of problems. We hope the reader will find some problems challenging and exciting as they are based on outstanding research papers. Selected solutions can be found at the end of the book.

The book is the result of a series of courses delivered in the last few years at Universidad de Buenos Aires, Universidad de Santiago de Chile, Reunión Anual de la Unión Matemática Argentina (Bahía Blanca, Argentina), Dalhousie University (Halifax, Canada), Vrije Universiteit Brussel (Belgium), and the Cimpa School "Crossroads of geometry, representation theory and higher structures".

We have also extensively used GAP in our research, particularly emphasizing the structure of simple groups, conjugacy classes and representations of finite groups, computation of non-commutative Gröbner bases, and classification of discrete algebraic structures. This experience gives us a substantial share of insight to write a comprehensive and user-friendly presentation of the capabilities of this computer algebra system. We aimed to build on the material of several mini-courses we have delivered, extending it to a book format and updating it with recent developments in the area. We hope that a book presenting the foundations from scratch and reaching a high level of mathematical concepts will rapidly become a valuable resource for the mathematical community. On the one hand, this belief is supported by the opinion of many colleagues who have encouraged us to write this particular book to respond to their need for a consistent reference for this powerful tool. On the other hand, by starting from the basics, the book will likely be helpful for a variety of graduate and undergraduate courses in mathematics. Also, instructors can use this book to design new exercises.

Acknowledgments

We thank the *Centre International de Rencontres Mathématiques (CIRM)* in Marseille, France, and the *Mathematisches Forschungsinstitut Oberwolfach (MFO)* for their invitations to the "Research in Pairs" programs. We express our infinite gratitude to the staff of these institutions.

Several colleagues have helped with their suggestions by providing examples, questions, exercises, solutions, and references. Among them, we especially thank Bill Allombert, Thomas Brauer, Jan De Beule, Ilaria Colazzo, Andrew Darlington, Bettina Eick, Fernando Fantino, Marco Farinati, César Galindo, Agustín García Iglesias, Gastón García, István Heckenberger, Max Horn, Alexander Hulpke, Mikhail Kotchetov, Nicolás Libedinsky, Mitja Mastnak, Andrés Navas, Iván Sadofschi Costa, Peter Selinger, Chris Wensley.

Oberwolfach, February 2023 *Kevin I. Piterman, Leandro Vendramin*

Contents

Part I Basic theory

Part I
Basic theory

Chapter 1
First steps

In this chapter, we begin with basic commands and arithmetic operations. Then we continue with the basics of the GAP programming language, such as structure controls and fundamental objects and variable types. We will also discuss some important algebraic structures, such as integers and rational numbers, finite fields, sets and permutations, matrices, polynomials and vector spaces.

1.1 The very first steps

Immediately after running GAP, we will see some information about the distribution that we have installed. This information includes the software version number and the packages loaded in memory. We will also see that GAP is ready:

```
gap>
```

To close GAP one uses **quit**:

```
gap> quit;
```

Every command should end with the symbol ; (semicolon). The symbol ;; (double semicolon) is also used to end a command, but it means that no screen output will be produced.

```
gap> 2+5;;
gap> 2+5;
7
```

To see information related to commands, functions, tutorials, and manuals, one uses the symbol ? (question mark). Here we have some examples:

```
gap> ?tutorial
gap> ?sets
gap> ?help
gap> ?permutations
gap> ?Eigenvalues
gap> ?CyclicGroup
gap> ?FreeGroup
gap> ?SylowSubgroup
```

To write comments, one uses the symbol # (number sign or hash) as the following example shows:

```
gap> # This is just a comment!
```

To make the command line more readable, one could use the symbol \ (backslash), as we see in the following example:

```
gap> # Let us compute 1+2+3
gap> 1\
> +2\
> +3;
6
```

The function `LogTo` saves the output in a given file. Everything you see on your terminal will also appear in this file. When the function `LogTo` is called with no parameters, GAP will stop writing a log file.

```
gap> # Save the output to the file mylog
gap> LogTo("mylog");
gap> # Stop saving the output
gap> LogTo();
```

`NamesUserGVars` returns the names of the global variables created by the user; note that loading a package may create new global variables. On the other hand, `MemoryUsage` returns the bytes used by a variable:

```
gap> p := 2^30;;
gap> NamesUserGVars();
[ "ProcessInitFiles", "p" ]
gap> s := "a string";;
gap> NamesUserGVars();
[ "ProcessInitFiles", "p", "s" ]
gap> MemoryUsage(p);
8
gap> MemoryUsage(s);
41
```

In the UNIX-based operating system, to run a GAP script from the terminal, we simply write

```
$ gap myfile.g
```

If we choose to limit the memory used, say for example, to 2GB, we call GAP as follows:

```
$ gap -o 2G myfile.g
```

The function `SaveWorkspace` saves a “snapshot" image of the current workspace in a file. Then, if we load this file when opening GAP, we will recover the full session that we had when this function was called. This means, for example, that we will recover all the variables (and hence functions) that were defined in our workspace at the moment we called `SaveWorkspace`.

In the following example we see how it works:

```
gap> n := 100;
gap> SaveWorkspace("myfile.data");
true
gap> quit;
```

A file called `myfile.data` was produced. Now we call GAP as follows:

```
$ gap -L myfile.data
```

We asked GAP to read the workspace stored in the file `myfile.data`. This means we will recover the GAP session precisely where `SaveWorkspace` was called. In particular, our new GAP session based on the file `myfile.data` will know the value of the variable `n`.

1.2 Basic arithmetic

One can do basic arithmetic operations with rational numbers:

```
gap> 1+1;
2
gap> 2*3;
6
gap> 8/2;
4
gap> (1/3)+(2/5);
11/15
gap> 2^(-4);
1/16
gap> 2*(-6)+4;
-8
gap> (1-5^2)^2-2*(2+4*2)^2;
376
gap> NumeratorRat(3/5);
3
gap> DenominatorRat(3/5);
5
```

One uses `a mod m` to obtain the remainder modulo *m* of *a*:

```
gap> 6 mod 4;
2
gap> -6 mod 5;
4
```

There are several built-in functions that one can use for specific purposes. For example, `Factors` returns the factorization of an integer and `IsPrime` detects whether an integer is prime or not:

```
gap> Factors(10);
[ 2, 5 ]
gap> Factors(18);
[ 2, 3, 3 ]
gap> Factors(1800);
[ 2, 2, 2, 3, 3, 5, 5 ]
gap> IsPrime(1800);
false
gap> Factors(37);
[ 37 ]
gap> IsPrime(37);
true
```

To obtain more information about integers, one can check the documentation:

```
gap> ?Integers
```

Other useful functions: `Sqrt` computes square roots, `Factorial` computes the factorial of a positive integer, `Gcd` computes the greatest common divisor of a finite list of integers, and `Lcm` computes the least common multiple:

```
gap> Sqrt(25);
5
gap> Factorial(15);
1307674368000
gap> Gcd(10, 4);
2
gap> Lcm(10, 4, 2, 6);
60
```

Remark 1.1. What happens if one tries to compute `Sqrt(7)`? We will get a result, but it will not be a rational number. See §1.4.1 for more details.

The function `Inverse` (resp. `AdditiveInverse`) returns the multiplicative (resp. additive) inverse of an element:

```
gap> AdditiveInverse(2/3);
-2/3
gap> Inverse(2/3);
3/2
```

We can work with very large numbers. Let us see some examples.

Example 1.1. One can easily prove that $n = 164$ is the largest integer such that 7^n divides 1000!:

```
gap> Factorial(1000) mod 7^164;
0
gap> Factorial(1000) mod 7^165 = 0;
false
```

Example 1.2. Let us compute $999^{179} \bmod 1793$:

```
gap> 999^179 mod 1763;
1219
```

The following example appears in `Mathoverflow`, question #282035. Can you prove the result without using computer software?

Example 1.3. The sum of digits of 3^{1000} is divisible by 7:

```
gap> Sum(ListOfDigits(3^1000)) mod 7;
0
```

1.3 Basic programming

1.3.1 Objects and variables

An object is something that we can assign to a variable, such as a number, a function, a string, a group, a field, an element of a group, a group homomorphism, a ring, a matrix, or a vector space. To assign an object to a variable, one uses the operator := as the following example shows:

```
gap> p := 32;;
gap> p;
32
gap> p = 32;
true
gap> p := p+1;;
gap> p;
33
gap> p = 32;
false
```

Remark 1.2. The symbols = (conditional) and := (assignment operator) are different!

Remark 1.3. What happens if we forget to save the result of a previous calculation in a variable? We can do the following:

```
gap> 2*(5+1)-6;
6
gap> n := last;
6
```

Note that `last` returns the very last output. One also has `last2` and `last3` to obtain the second-to-last and third-to-last outputs, respectively.

The global variable `time` stores the number of milliseconds the last command took. In the following example, we create some random matrices and see the number of milliseconds the calculations take:

```
gap> m := RandomInvertibleMat(20);;
gap> time;
11
gap> m := RandomInvertibleMat(50);;
gap> time;
430
```

The function `StringTime` converts a given number of milliseconds to a readable string:

```
gap> StringTime(430);
" 0:00:00.430"
gap> StringTime(2000);
" 0:00:02.000"
```

On page 23, we discuss other ways of measuring run times.

The function **`IsBound`** checks if a variable points to a value (i.e., if the variable is already defined). The function **`Unbind`** deletes an identifier. Thus **`Unbind`** can be used, for example, to get rid of unwanted large objects:

```
gap> IsBound(my_variable);
false
gap> my_variable := 100;;
gap> IsBound(my_variable);
true
gap> NamesUserGVars();
[ "ProcessInitFiles", "my_variable" ]
gap> Unbind(my_variable);
gap> IsBound(my_variable);
false
gap> NamesUserGVars();
[ "ProcessInitFiles" ]
```

The internal variable `ProcessInitFiles` appears in newer versions of GAP.

1.3.2 Conditionals

There are three important logical operators: **`not`**, **`and`**, **`or`**. We also have comparison operators; for example, the expression `x<>y` returns **`true`** if `x` and `y` are different, and **`false`** otherwise:

```
gap> x := 20;; y := 10;;
gap> x <> y;
true
gap> x > y;
true
```

```
gap> (x > 0) or (x < y);
true
gap> (x > 0) and (x < y);
false
gap> (2*y < x);
false
gap> (2*y <= x);
true
gap> not (x < y);
true
```

Example 1.4. Let us check that $100^{300} > 300^{100}$:

```
gap> 100^300 > 300^100;
true
```

The construction **`if ... then ... fi`** is easy to understand if we look at particular examples. To illustrate the use of the if–then statement, we refer to Example 1.5 below, where the function

$$f : n \mapsto \begin{cases} n^3 & \text{if } n \equiv 0 \bmod 3, \\ n^5 & \text{if } n \equiv 1 \bmod 3, \\ 0 & \text{otherwise,} \end{cases}$$

is constructed.

1.3.3 Functions

There are two equivalent ways of constructing simple functions. For example, to construct the map $x \mapsto x^2$ either we can use the *one-line definition*

```
gap> InlineSquare := x->x^2;
function( x ) ... end
```

or the more natural

```
gap> MySquare := function(x)
> return x^2;
> end;
function( x ) ... end
```

In both cases, we will obtain the same result:

```
gap> InlineSquare(4);
16
gap> MySquare(4);
16
gap> InlineSquare(-5);
```

```
25
gap> MySquare(-5);
25
```

Remark 1.4. The function `x->x^2` will return the square `x^2` of the object `x` whenever this makes sense. We do not need to specify the type of `x`.

One can also define functions with no arguments. The following is a classic example:

```
gap> SayHi := function()
> Display("Hello world");
> end;
function(  ) ... end
gap> SayHi();
Hello world
```

Example 1.5. Let us write a function to compute the map

$$f\colon n \mapsto \begin{cases} n^3 & \text{if } n \equiv 0 \bmod 3,\\ n^5 & \text{if } n \equiv 1 \bmod 3,\\ 0 & \text{otherwise.}\end{cases}$$

Here is the code and some experiments:

```
gap> f := function(n)
> if n mod 3 = 0 then
>   return n^3;
> elif n mod 3 = 1 then
>   return n^5;
> else
>   return 0;
> fi;
> end;
function( n ) ... end
gap> f(10);
100000
gap> f(5);
0
gap> f(4);
1024
```

Example 1.6. The Fibonacci sequence f_n is defined recursively as $f_1 = f_2 = 1$ and

$$f_{n+1} = f_n + f_{n-1}$$

for $n \geq 2$. The following function computes the Fibonacci numbers:

```
gap> MyFibonacci := function(n)
> if n = 1 or n = 2 then
>   return 1;
> else
>   return MyFibonacci(n-1)+MyFibonacci(n-2);
> fi;
> end;
function( n ) ... end
gap> MyFibonacci(10);
55
```

Note that with this method, it is impossible to compute higher terms of the sequence. So, for example, we cannot compute the value of f_{100}. Do you know why?

For a positive integer n, let

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ 3n+1 & \text{if } n \text{ is odd.} \end{cases}$$

Conjecture 1.1 (Collatz). For any positive integer n, there exists an integer $m \geq 1$ such that $f^m(n) = 1$, where $f^m = f \circ \cdots \circ f$ (m-times).

The conjecture, also known as the $3n + 1$ problem, the Syracuse problem, Kakutani's problem, Hasse's algorithm, and Ulam's conjecture, is still open; we refer to [30] for more information.

Example 1.7 (Testing Collatz conjecture). Let us test the conjecture for $n = 5$:

```
gap> f := function(n)
> if n mod 2 = 0 then
>   return n/2;
> else
>   return 3*n+1;
> fi;
> end;
function( n ) ... end
gap> f(f(f(f(f(5)))));
1
```

An interesting exercise is to write a function that for each n returns the smallest integer m such that $f^m(n) = 1$. A possible solution uses recursive functions:

```
gap> g := function(n)
> if f(n) = 1 then
>   return 1;
> else
>   return 1+g(f(n));
> fi;
> end;
function( n ) ... end
```

1.3.4 Loops

We will explain how loops work with the following straightforward problem. We want to check that

$$1 + 2 + 3 + \cdots + 100 = 5050.$$

We can use the function `Sum`:

```
gap> Sum([1..100]);
5050
```

An alternative way of performing this computation is by using the for loop structure **`for ... do ... od`**:

```
gap> s := 0;;
gap> for k in [1..100] do
> s := s+k;
> od;
gap> s;
5050
```

We can also use the while loop structure: **`while ... do ... od`**:

```
gap> s := 0;;
gap> k := 1;;
gap> while k <= 100 do
> s := s+k;
> k := k+1;
> od;
gap> s;
5050
```

Alternatively, we can also use the structure **`repeat ... until`**:

```
gap> s := 0;;
gap> k := 1;;
gap> repeat
> s := s+k;
> k := k+1;
> until k > 100;
gap> s;
5050
```

Example 1.8. Let us compute (again) some Fibonacci numbers, this time without using recursive functions. The following method is computationally more efficient than that of Example 1.6.

```
gap> MyFibonacci := function(n)
> local k, x, y, tmp;
> x := 1;
> y := 1;
> for k in [3..n] do
>   tmp := y;
>   y := x+y;
>   x := tmp;
> od;
> return y;
> end;
function( n ) ... end
gap> MyFibonacci(100);
354224848179261915075
gap> MyFibonacci(1000);
4346655768693745643568852767504062580256466051737178040248172908\
9536555417949051890403879840079255169295922593080322634775209689\
6232398733224711616429964409065331879382989696499285160037044761\
37795166849228875
```

In the previous function, we defined local variables. The local statement must be the first statement of a function definition. The nature of local variables in a function prevents the value of the variables from being overwritten outside this particular function.

Example 1.9. Divisors of a given integer can be obtained with `DivisorsInt`. In this example, we run over the divisors of 100 and print only those that are odd:

```
gap> Filtered(DivisorsInt(100), x->x mod 2 = 1);
[ 1, 5, 25 ]
```

Similarly:

```
gap> for d in DivisorsInt(100) do
> if d mod 2 = 1 then
>   Display(d);
> fi;
> od;
1
5
25
```

With **`continue`** one can skip iterations. An equivalent (but less elegant) approach to the problem of Example 1.9 is the following:

```
gap> for d in DivisorsInt(100) do
> if d mod 2 = 0 then
>   continue;
> fi;
> Display(d);
> od;
1
5
25
```

With **`break`** one breaks a loop. In the following example, we run over the numbers $1, 2, \ldots, 100$ and stop when a number whose square is divisible by 20 appears:

```
gap> First([1..100], x->x^2 mod 20 = 0);
10
```

Similarly:

```
gap> for k in [1..100] do
> if k^2 mod 20 = 0 then
>   Display(k);
>   break;
> fi;
> od;
10
```

1.3.5 Strings

A string (of characters) is an expression delimited by the symbol " (quotation mark):

```
gap> mystring := "hello world";
hello world
```

Each element of a string will have a unique position that identifies it. Such a position is called the index of the element. Indices start at position one. To extract one character, one uses the expression `mystring[position]`; to extract substrings `mystring{positions}`. Let us see some examples:

```
gap> mystring[1];
'h'
gap> mystring[3];
'l'
gap> mystring{[1,2,3,4,5]};
"hello"
gap> mystring{[7,8,9,10,11]};
"world"
gap> mystring{[11,10,9,8,7,6,5,4,3,2,1]};
"dlrow olleh"
```

Several functions allow us to work with strings. The function `String` converts anything into a string of characters:

```
gap> String(1234);
"1234"
gap> String(01234);
"1234"
gap> String([1,2,3]);
"[ 1, 2, 3 ]"
gap> String(true);
"true"
```

The function `ReplacedString` replaces substrings:

```
gap> ReplacedString("Hello world", "world", "all");
"Hello all"
```

The function `Print` allows us to print data (in this case, a string) on the screen:

```
gap> mystring := "Hello world";;
gap> Print(mystring);
Hello world
```

Let us see another example:

```
gap> n := 100;;
gap> m := 5;;
gap> Print(n, " times ", m, " is ", n*m);
100 times 5 is 500
```

The function `Print` can be used with some special characters. For example, `\n` means “new line”:

```
gap> Print("Hello\nworld");
Hello
world
gap> Print("To write \\...");
To write \...
```

The functions `PrintTo` and `AppendTo` work as `Print` but the output goes to a file. It is important to remark that `PrintTo` will overwrite an existing file!

1.3.6 Lists

A list is an ordered sequence of objects (maybe of different types), including empty places. Thus, for example, an array of numbers is just a list of numbers. Lists are written using square brackets:

```
gap> IsList([1, 2, 3]);
true
gap> IsList([1, 2, 3, "abc"]);
true
gap> IsList([1, 2,, "abc"]);
true
gap> 2 in [1, 2, 5, 4, 10];
true
gap> 3 in [0, 10, "abc"];
false
gap> ListWithIdenticalEntries(3, "a");
[ "a", "a", "a" ]
```

We can create lists in a clean way by using list comprehension. The following examples illustrate different ways to construct lists and need no further explanations:

```
gap> List([1..10], x->-x);
[ -1, -2, -3, -4, -5, -6, -7, -8, -9, -10 ]
gap> List([1, 2, 3, 4, 5], x->x^2);
[ 1, 4, 9, 16, 25 ]
gap> List([1, 2, 3, 4, 5], IsPrime);
[ false, true, true, false, true ]
gap> Filtered([1..20], IsPrime);
[ 2, 3, 5, 7, 11, 13, 17, 19 ]
```

As it happens with strings, each element of a list will have a unique position, the element index, that identifies it.

Example 1.10. Let us create a list with the first six prime numbers. Both `Size` and `Length` return the number of elements of the list:

```
gap> primes := [2, 3, 5, 7, 11, 13];
[ 2, 3, 5, 7, 11, 13 ]
gap> Size(primes);
6
gap> Length([1,2,3,,4,,,5]);
8
gap> Size([1,2,3,,4,,,5]);
8
```

Remark 1.5. `Size` for a list calls `Length` and the length of a list is defined as the index of the last bound entry. Let us see some examples:

```
gap> Size([,,,,]);
0
gap> [,,,,];
[  ]
gap> Size([1,,,,]);
1
gap> [1,,,,];
[ 1 ]
gap> Size([,,,,1]);
5
gap> [,,,,1];
[ ,,,, 1 ]
```

To access an element inside a list, one should refer to the position:

```
gap> primes[1];
2
gap> primes[2];
3
```

Let us obtain the sublist consisting of the elements in the second, third and fifth positions:

```
gap> primes;
[ 2, 3, 5, 7, 11, 13 ]
```

```
gap> primes{[2, 3, 5]};
[ 3, 5, 11 ]
```

Another example (to avoid confusion):

```
gap> mylist := ["a", "b", "c", "d", "e", "f"];
[ "a", "b", "c", "d", "e", "f" ]
gap> mylist{[1, 3, 5]};
[ "a", "c", "e" ]
```

One uses `Position` to find elements inside a list. If the element we are looking for does not belong to the list, `Position` will return `fail`; otherwise, it will return the first occurrence of the element. Let us look at some examples:

```
gap> Position([5, 4, 6, 3, 7, 3, 7], 5);
1
gap> Position([5, 4, 6, 3, 7, 3, 7], 1);
fail
gap> Position([5, 4, 6, 3, 7, 3, 7], 7);
5
```

The functions `Add` and `Append` are used to add elements at the end of a list. The following examples show how these functions work:

```
gap> primes;
[ 2, 3, 5, 7, 11, 13 ]
gap> # Add 19 at the end of the list
gap> Add(primes, 19);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 19 ]
gap> # Add the prime 17 at position 7
gap> Add(primes, 17, 7);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 17, 19 ]
gap> # Add 23 and 29 at the end
gap> Append(primes, [23, 29]);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ]
```

To remove elements from a list, one uses `Remove`:

```
gap> # Remove the first element of the list
gap> Remove(primes, 1);;
gap> primes;
[ 3, 5, 7, 11, 13, 17, 19, 23, 29 ]
```

For lists, **`IsBound`** can be used to check whether entries are bound:

```
gap> l := [1..5];;
gap> IsBound(l[5]);
true
gap> IsBound(l[6]);
false
```

The function `Concatenation` concatenates two or more lists. This function returns a new list consisting of the lists used in the argument:

```
gap> Concatenation([1, 2, 3], [4, 5, 6]);
[ 1, 2, 3, 4, 5, 6 ]
```

Remark 1.6. There is a difference between `Append` and `Concatenation`. The function `Append` modifies the lists used in the argument, while `Concatenation` does not.

The function `Collected` returns a new list where each element of the original list appears with multiplicity:

```
gap> Factors(720);
[ 2, 2, 2, 2, 3, 3, 5 ]
gap> Collected(last);
[ [ 2, 4 ], [ 3, 2 ], [ 5, 1 ] ]
```

To make a copy of a list, one should use the function `ShallowCopy`. The following example shows the difference between `ShallowCopy` and the assignment operator:

```
gap> a := [1, 2, 3, 4];;
gap> b := a;;
gap> c := ShallowCopy(a);;
gap> Add(a, 5);
gap> a;
[ 1, 2, 3, 4, 5 ]
gap> b;
[ 1, 2, 3, 4, 5 ]
gap> c;
[ 1, 2, 3, 4 ]
gap> Add(b, 10);
gap> a;
[ 1, 2, 3, 4, 5, 10 ]
gap> b;
[ 1, 2, 3, 4, 5, 10 ]
```

Remark 1.7. The function `ShallowCopy` takes a picture of the list at a particular moment. In our example, we have a list `a`, a mirror image of `a` called `b`, and a photograph (shallow copy) of `a` called `c`. If we modify the lists `a` or `b`, then both `a` and `b` change, but not the list `c`. Conversely, modifying `c` will not change `a` and `b`.

The function `Reversed` returns a list containing the elements of our list in reversed order. In the following example, the variable `list` will not be modified by the function `Reversed`:

```
gap> mylist := [2, 4, 7, 3];;
gap> Reversed(mylist);
[ 3, 7, 4, 2 ]
gap> mylist;
[ 2, 4, 7, 3 ]
```

The function `SortedList` returns a new list where the elements are sorted with respect to the operator `<=`. In the following example, one sees that `SortedList` will not modify the value of the variable `mylist`:

```
gap> mylist := [2, 4, 7, 3];;
gap> SortedList(mylist);
[ 2, 3, 4, 7 ]
gap> mylist;
[ 2, 4, 7, 3 ]
```

The function `Sort` sorts a list in increasing order. Can you recognize the difference between `Sort` and `SortedList`?

```
gap> mylist := [2, 4, 7, 3];;
gap> Sort(mylist);
gap> mylist;
[ 2, 3, 4, 7 ]
```

Remark 1.8. If we want to apply `SortedList` or `Sort` to a given list, then all the elements of the list must be of the same type and comparable for the operator `<=`.

One can also use particular ordering functions:

Example 1.11. Let $X = \{1, \dots, 10\}$. We let $y \preceq x$ if and only if x has at least the same number of prime factors as y. Then

$$1 \preceq 9 \preceq 8 \preceq 7 \preceq 5 \preceq 4 \preceq 3 \preceq 2 \preceq 10 \preceq 6.$$

Note that, for example, $3 \preceq 2$ and $2 \preceq 3$. Here is the code:

```
gap> MyOrder := function(x, y)
> return Size(PrimeDivisors(x)) <= Size(PrimeDivisors(y));
> end;
function( x, y ) ... end
gap> l := [1..10];;
gap> Sort(l, MyOrder);
gap> l;
[ 1, 9, 8, 7, 5, 4, 3, 2, 10, 6 ]
```

The function `Filtered` allows us to obtain the elements of a list that satisfy a particular given property; if we just want to count such elements, we use `Number`. In the same vein, `First` returns the first element of a list that satisfies a given property. Here we have some examples:

```
gap> mylist := [1, 2, 3, 4, 5];;
gap> Filtered(mylist, x->x mod 2 = 0);
[ 2, 4 ]
gap> Number(mylist, x->x mod 2 = 0);
2
gap> Filtered(mylist, x->x mod 2 = 1);
[ 1, 3, 5 ]
```

```
gap> Filtered(mylist, IsOddInt);
[ 1, 3, 5 ]
gap> First(mylist, x->x mod 2 = 0);
2
```

One can also perform arithmetic operations on lists:

```
gap> Sum([1..5], x->2*x);
30
gap> Sum(ListOfDigits(342));
9
gap> Product([1..5], x->x^2);
14400
```

Example 1.12. Let us compute how many powers of 2 divide 18000. This number is four, as the following code shows:

```
gap> Factors(18000);
[ 2, 2, 2, 2, 3, 3, 5, 5, 5 ]
gap> Collected(Factors(18000));
[ [ 2, 4 ], [ 3, 2 ], [ 5, 3 ] ]
gap> Number(Factors(18000), x->x = 2);
4
```

The function `ForAny` returns **`true`** if there is an element in the list satisfying the required condition and **`false`** otherwise. Similarly, `ForAll` returns **`true`** if all the elements of the list satisfy the required condition and **`false`** otherwise. The following examples illustrate how these functions work:

```
gap> ForAny([2, 4, 6, 8, 10], x->x mod 2 = 0);
true
gap> ForAll([2, 4, 6, 8, 10], x->x > 0);
true
gap> ForAny([2, 3, 4, 5], IsPrime);
true
gap> ForAll([2, 3, 4, 5], IsPrime);
false
gap> ForAny([false, false], x->x = true);
false
gap> ForAny([true, false], x->x = true);
true
gap> ForAll([true, false], x->x = true);
false
gap> ForAll([true, true], x->x = true);
true
```

1.3.7 Ranges

Ranges are lists of integers where the difference between any two consecutive elements is constant:

```
gap> Elements([1..5]);
[ 1, 2, 3, 4, 5 ]
gap> Elements([1, 3..11]);
[ 1, 3, 5, 7, 9, 11 ]
gap> Elements([0, -2..-8]);
[ -8, -6, -4, -2, 0 ]
gap> IsRange([1..100]);
true
gap> IsRange([1, 3, 5, 6]);
false
```

We can use `Elements` to list all the elements in a given range. Conversely, `ConvertToRangeRep` converts (if possible) a list into a range:

```
gap> l := [1, 2, 3, 4, 5 ];;
gap> ConvertToRangeRep(l);;
gap> l;
[ 1 .. 5 ]
gap> l := [7, 11, 15, 19, 23 ];
gap> IsRange(l);
true
gap> ConvertToRangeRep(l);
gap> l;
[ 7, 11 .. 23 ]
```

1.3.8 Sets

A set is a particular type of ordered list with no repetitions nor gaps. To create sets (or to convert a list into a set) one uses `Set`:

```
gap> mylist := [1, 2, 3, 1, 5, 6, 2];;
gap> IsSet(mylist);
false
gap> Set(mylist);
[ 1, 2, 3, 5, 6 ]
gap> Set([1,2,,,3,2,,1]);
[ 1, 2, 3 ]
gap> Set([1..10], x->x mod 3);
[ 0, 1, 2 ]
```

To add elements use `AddSet` and `UniteSet`. To remove an element, use the function `RemoveSet`. These commands modify the set given as a parameter. Here we have some examples:

```
gap> myset := Set([1, 2, 4, 5]);;
gap> # Let us add the number 10
gap> AddSet(myset, 10);
gap> myset;
[ 1, 2, 4, 5, 10 ]
```

```
gap> # Let us remove the number 4
gap> RemoveSet(myset, 4);
gap> myset;
[ 1, 2, 5, 10 ]
gap> UniteSet(myset, [1, 1, 5, 6]);
gap> myset;
[ 1, 2, 5, 6, 10 ]
```

One uses `Union`, `Intersection`, `Difference` and `Cartesian` to perform basic set operations. Examples:

```
gap> S := Set([1, 2, 8, 11]);;
gap> T := Set([2, 5, 7, 8]);;
gap> Intersection(S, T);
[ 2, 8 ]
gap> Union(S, T);
[ 1, 2, 5, 7, 8, 11 ]
gap> Difference(S, T);
[ 1, 11 ]
gap> Difference(T, S);
[ 5, 7 ]
gap> Difference(S, S);
[  ]
gap> Cartesian(S, T);
[ [ 1, 2 ], [ 1, 5 ], [ 1, 7 ], [ 1, 8 ], [ 2, 2 ],
  [ 2, 5 ], [ 2, 7 ], [ 2, 8 ], [ 8, 2 ], [ 8, 5 ],
  [ 8, 7 ], [ 8, 8 ], [ 11, 2 ], [ 11, 5 ],
  [ 11, 7 ], [ 11, 8 ] ]
```

Note that the functions `Union`, `Intersection` and `Difference` create new sets, so they do not modify the sets given as parameters.

1.3.9 Records

Records allow us to put several objects in the same structure. Let us say that we want to create a structure for the point $(1,2)$ of the plane:

```
gap> point := rec(x := 1, y := 2);;
gap> point.x;
1
gap> point.y;
2
gap> RecNames( point );
[ "x", "y" ]
```

The function **`IsBound`** can be used to test whether our structure contains a given component:

```
gap> point := rec( x := 1, y := 2 );;
gap> IsBound(point.z);
false
```

```
gap> point.z := 3;;
gap> IsBound(point.z);
true
gap> point;
rec( x := 1, y := 2, z := 3 )
gap> Unbind(point.z);
gap> point;
rec( x := 1, y := 2 )
```

1.4 Other numbers

1.4.1 Floating–point numbers

GAP supports floating-point numbers in machine format. Note that floating-point numbers are not rationals. For example, the number `3` is an integer and `3.0` is a float. Every floating-point number must contain a decimal digit. To get a rational approximation for a floating-point number, one uses the function `Rat`. Conversely, the function `Float` returns a floating-point number from a rational one. Let us show some examples:

```
gap> # An approximation of pi
gap> a := 3.1416;
gap> Rat(a);
3927/1250
gap> # A better approximation of pi
gap> b := 4*Atan(1.0);
3.14159
gap> Rat(b);
817696623/260280919
gap> Float(last);
3.14159
gap> Float(1/4);
0.25
gap> Float(1/3);
0.333333
```

One possible application of floating-point numbers is related to run times.

Example 1.13. Let us assume that we have a list of strings representing run-times (in milliseconds). Say that the numbers are 1.0324, 2.4102, 0.2112, and 0.4324. We need to sum up all these numbers:

```
gap> times := ["1.0324", "2.4102", "0.2112", "0.4324"];;
gap> List(times, Float);
[ 1.0324, 2.4102, 0.2112, 0.4324 ]
gap> Sum(last);
4.0862
```

The function `NanosecondsSinceEpoch` returns the time (in nanoseconds) that has passed since some fixed (and unknown) time in the past. This function is appropriate for doing time measurements. In the following code, we compute the number of milliseconds we need to create a big random matrix and compute its determinant:

```
gap> t0 := NanosecondsSinceEpoch();;
gap> m := RandomMat(123, 123);;
gap> Determinant(m);;
gap> t1 := NanosecondsSinceEpoch();;
gap> # The need the difference between t1 and t0 in milliseconds
gap> mytime := Int(Float((t1-t0)/10^6));;
gap> StringTime(mytime);
" 0:00:15.752"
```

1.4.2 Finite fields

To create the finite field of p^n elements (here p is a prime number) we use the function `GF` (Galois Field). The characteristic of a field can be obtained with `Characteristic`:

```
gap> GF(2);
GF(2)
gap> GF(4);
GF(2^2)
gap> GF(9);
GF(3^2)
gap> Characteristic(GF(2));
2
gap> Characteristic(GF(9));
3
```

Let p be a prime number and let F denote the field with $q = p^n$ elements for some positive integer n. The subset

$$\{x \in F : x \neq 0\}$$

is a cyclic group of size $q-1$; generated, say, by ζ. Then $F = \{0, 1, \zeta, \zeta^2, \dots, \zeta^{q-2}\}$, so each non-zero element of F is a power of ζ.

We assume that $q \leq 2^{16}$. Each non-zero element of the finite field `GF(q)` will be a power of the generator `Z(q)`. The zero of `GF(q)` will be `0*Z(q)` or, equivalently, `Zero(GF(q))`. Similarly, `One(GF(q))` will be the multiplicative neutral element of `GF(q)`:

```
gap> Size(GF(4));
4
gap> Elements(GF(4));
[ 0*Z(2), Z(2)^0, Z(2^2), Z(2^2)^2 ]
gap> Z(4);
```

```
Z(2^2)
gap> Inverse(Z(4));
Z(2^2)^2
gap> Zero(GF(4));
0*Z(2)
gap> 0 in GF(4);
false
gap> Zero(Rationals);
0
gap> One(GF(4));
Z(2)^0
gap> 1 in GF(4);
false
gap> One(Rationals);
1
```

If p is a prime number, it is natural to identify the field $\mathbb{F}_p$ with the ring $\mathbb{Z}/p$ of integers modulo p. This identification can be made with the function `Int`:

```
gap> Elements(GF(5));
[ 0*Z(5), Z(5)^0, Z(5), Z(5)^2, Z(5)^3 ]
gap> Int(Z(5)^0);
1
gap> Int(Z(5)^1);
2
gap> Int(Z(5)^2);
4
gap> Int(Z(5)^3);
3
```

Remark 1.9. The representation for elements of finite fields mentioned before works for fields of size $\leq 2^{16}$. Elements of bigger finite fields will use a different representation:

```
gap> Random(GF(2^16));
Z(2^16)^233
gap> x := Random(GF(2^17));
z+z4+z5+z6+z7+z8+z9+z10+z11+z13+z15+z16
gap> y := Random(GF(3^11));
2+z2+z4+z6+2z7+2z8+z9+z10
```

The random element of the finite field $\mathbb{F}_{2^{17}}$ of size 2^{17} is

$$x = z + z^4 + z^5 + z^6 + z^7 + z^8 + z^9 + z^{10} + z^{11} + z^{13} + z^{15} + z^{16},$$

where z is a primitive root that is a generator of the multiplicative group of $\mathbb{F}_{2^{17}}$. Similarly, the random element of the field $\mathbb{F}_{3^{11}}$ is

$$y = 2 + z^2 + z^4 + z^6 + 2z^7 + 2z^8 + z^9 + z^{10},$$

where z is a primitive root.

1.4.3 Cyclotomic numbers

We can also work with cyclotomic fields. The function `CF` creates a cyclotomic field. To create primitive roots of 1, one uses the function `E`. More precisely: `E(n)` returns $e^{2\pi i/n}$. Typically, cyclotomic numbers will be represented as rational linear combinations of primitive roots of 1. Let us see some examples:

```
gap> Characteristic(CF(3));
0
gap> Characteristic(CF(4));
0
gap> E(6) in Rationals;
false
gap> E(6) in Cyclotomics;
true
gap> E(3) in CF(3);
true
gap> E(3) in CF(4);
false
gap> E(6);
-E(3)^2
```

In general, most of the basic arithmetic operations discussed at the beginning of this chapter can be performed with cyclotomic numbers:

```
gap> E(3)^2+E(3);
-1
gap> E(5)^5-E(5);
-2*E(5)-E(5)^2-E(5)^3-E(5)^4
```

The functions `Inverse` and `AdditiveInverse` behave well with cyclotomic numbers:

```
gap> AdditiveInverse(E(7));
-E(7)
gap> Inverse(E(7));
E(7)^6
```

With cyclotomic numbers, we can also compute square roots:

```
gap> Sqrt(-1);
E(4)
gap> Sqrt(2);
E(8)-E(8)^3
gap> Sqrt(7);
E(28)^3-E(28)^11-E(28)^15+E(28)^19-E(28)^23+E(28)^27
```

1.5 Permutations

Let n be a positive integer. A *permutation* in n letters (or symbols) is a bijective map

$$\sigma\colon \{1,\dots,n\}\to\{1,\dots,n\}.$$

We write $\mathbb{S}_n$ to denote the set of permutations $\sigma\colon \{1,\dots,n\}\to\{1,\dots,n\}$. For example, the permutation

$$\sigma = \begin{pmatrix} 1\,2\,3\,4 \\ 3\,1\,2\,4 \end{pmatrix}$$

is the bijective map

$$1\mapsto 3,\quad 2\mapsto 1,\quad 3\mapsto 2,\quad 4\mapsto 4.$$

We will use the exponential notation, so for example

$$1^\sigma = 3,\quad 2^\sigma = 1,\quad 3^\sigma = 2,\quad 4^\sigma = 3.$$

Usually, one writes a permutation as a product of disjoint cycles. For example:

$$\begin{pmatrix} 1\,2\,3\,4 \\ 2\,4\,1\,3 \end{pmatrix} = (1\,2\,4\,3), \qquad \begin{pmatrix} 1\,2\,3\,4\,5 \\ 2\,1\,4\,3\,5 \end{pmatrix} = (1\,2)(3\,4)(5) = (1\,2)(3\,4).$$

The permutation $\left(\begin{smallmatrix} 1\,2\,3\,4\,5 \\ 2\,1\,4\,3\,5 \end{smallmatrix}\right) = (1\,2)(3\,4)$ in GAP is `(1,2)(3,4)`.

The function `IsPerm` checks whether some object is a permutation. Let us see some examples:

```
gap> IsPerm((1,2)(3,4));
true
gap> (1,2)(3,4)(5) = (1,2)(3,4);
true
gap> (1,2)(3,4) = (3,4)(2,1);
true
gap> IsPerm(25);
false
gap> IsPerm([1, 2, 3, 4]);
false
```

The image of an element `i` under the natural right action of a permutation `p` is `i^p`. The preimage of the element `i` under `p` can be obtained with `i/p`. In the following example, we compute the image of 1 and the preimage of 3 by the permutation (123):

```
gap> 2^(1,2,3);
3
gap> 2/(1,2,3);
1
```

The composition of permutations will be performed from left to right. For example,

$$(1\,2\,3)(2\,3\,4) = (1\,3)(2\,4),$$

as the following code shows:

```
gap> (1,2,3)*(2,3,4);
(1,3)(2,4)
```

The exponential notation is quite convenient here: If σ and τ are permutations of $\{1,\ldots,n\}$, then

$$i^{(\sigma\tau)} = (i^{\sigma})^{\tau}$$

for all $i \in \{1,\ldots,n\}$.

There are two ways of computing inverses of permutations:

```
gap> Inverse((1,2,3));
(1,3,2)
gap> (1,2,3)^(-1);
(1,3,2)
```

Example 1.14. The function `Permuted` returns a new list that contains the elements of our list permuted according to the given permutation perm:

```
gap> Permuted([10..15], (1,2));
[ 11, 10, 12, 13, 14, 15 ]
gap> p := (1,2,3);;
gap> a := [3, 2, 10, 5];;
gap> b := Permuted(a, (1,2,3));
[ 10, 3, 2, 5 ]
gap> # We check that b[x^p] = a[x] for all x
gap> List([1..4], x->b[x^p]) = a;
true
```

Sometimes it is useful to recover the cycle structure of a given permutation. The following code needs no further explanation:

```
gap> p := (1,2,3)(4,5)(8,9);;
gap> Cycles(p, MovedPoints(p));
[ [ 1, 2, 3 ], [ 4, 5 ], [ 8, 9 ] ]
gap> Cycles(p, [1..LargestMovedPoint(p)]);
[ [ 1, 2, 3 ], [ 4, 5 ], [ 6 ], [ 7 ], [ 8, 9 ] ]
```

Let σ be a permutation, written as a product of disjoint cycles. The function `ListPerm` returns a list containing i^{σ} at position i. Conversely, any list representing a permutation (i.e., of size n containing the integers from 1 to n) can be transformed into a permutation with the function `PermList`. Let us see some examples:

```
gap> # The permutation (12) in two letters
gap> ListPerm((1,2));
[ 2, 1 ]
gap> # The permutation (12) in four letters
gap> ListPerm((1,2), 4);
[ 2, 1, 3, 4 ]
gap> ListPerm((1,2,3)(4,5));
[ 2, 3, 1, 5, 4 ]
gap> ListPerm((1,3));
[ 3, 2, 1 ]
gap> PermList([1, 2, 3]);
()
gap> PermList([2, 1]);
```

```
(1,2)
gap> PermList([3, 4]);
fail
gap> n := 9;;
gap> PermList(Concatenation([2..n], [1]));
(1,2,3,4,5,6,7,8,9)
```

The sign of a permutation σ is the number $(-1)^k$, where $\sigma = \tau_1 \cdots \tau_k$ is some factorization of σ as a product of transpositions. To compute the sign of a permutation, one uses the function `SignPerm`:

```
gap> SignPerm(());
1
gap> SignPerm((1,2));
-1
gap> SignPerm((1,2,3,4,5));
1
gap> SignPerm((1,2)(3,4,5));
-1
gap> SignPerm((1,2)(3,4));
1
```

Example 1.15. For a given n we will construct the permutation $\sigma \in \mathbb{S}_n$ given by $j \mapsto n - j + 1$. We will write σ as a product of disjoint cycles and compute its sign:

```
gap> n := 5;;
gap> p := PermList(List([1..n], j->n-j+1));
(1,5)(2,4)
gap> SignPerm(p);
1
```

1.6 Matrices

A matrix is just a rectangular array of elements. The size of a matrix can be obtained with `DimensionsMat`. Sometimes (for example if one has an integer matrix) the function `Display` shows matrices in a nice way. `LaTeX` returns the LaTeX command[1] needed to write a matrix:

```
gap> m := [[1, 2, 3], [4, 5, 6]];;
gap> Display(m);
[ [  1,  2,  3 ],
  [  4,  5,  6 ] ]
gap> LaTeX(m);
"\\left(\\begin{array}{rrr}%\n1&2&3\\
\\%\n4&5&6\\\\%\n\\end{array}\\right)%\n"
gap> m[1][1];
1
```

[1] The function `LaTeX` works for other objects as well.

```
gap> m[1][2];
2
gap> m[2][1];
4
gap> DimensionsMat(m);
[ 2, 3 ]
```

Example 1.16. Let $v = (1,2,3)$ and $w = (0,5,-7)$ be row vectors of $\mathbb{Q}^3$. Let us check that $-5v = (-5,-10,-15)$ and $2v - w = (2,-1,13)$. We also check that the inner product between v and w is $v \cdot w = -11$:

```
gap> v := [1, 2, 3];;
gap> w := [0, 5, -7];;
gap> IsRowVector(v);
true
gap> IsRowVector(w);
true
gap> -5*v;
[ -5, -10, -15 ]
gap> 2*v-w;
[ 2, -1, 13 ]
gap> v*w;
-11
```

Example 1.17. With `PermutationMat`, we can construct a permutation matrix P_σ of a given dimension. The matrix P_σ represents the permutation σ acting on the right by permuting the basis vectors. Here we have an illustrative example:

```
gap> v := [5, 6, 7, 8];;
gap> p := PermutationMat((1,2,3), 4);;
gap> Display(p);
[ [  0,  1,  0,  0 ],
  [  0,  0,  1,  0 ],
  [  1,  0,  0,  0 ],
  [  0,  0,  0,  1 ] ]
gap> Permuted(v, (1,2,3));
[ 7, 5, 6, 8 ]
gap> v*p;
[ 7, 5, 6, 8 ]
```

Example 1.18. Let

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 2 \\ 6 & 1 \\ 0 & 2 \end{pmatrix}.$$

```
gap> A := [[1, 1, 1], [0, 1, 1], [0, 0, 1]];;
gap> B := [[1, 4, 7], [2, 5, 8]];;
gap> C := [[1, 2], [6, 1], [0, 2]];;
```

We note that the function `Display` displays matrices in a very nice way (if possible):

```
gap> A;
[ [ 1, 1, 1 ], [ 0, 1, 1 ], [ 0, 0, 1 ] ]
gap> Print(A);
[ [ 1, 1, 1 ], [ 0, 1, 1 ], [ 0, 0, 1 ] ]
gap> Display(A);
[ [  1,  1,  1 ],
  [  0,  1,  1 ],
  [  0,  0,  1 ] ]
```

Let us compute A^3, BC, CB, $A+CB$ and $2A-5CB$:

```
gap> Display(A^3);
[ [  1,  3,  6 ],
  [  0,  1,  3 ],
  [  0,  0,  1 ] ]
gap> Display(B*C);
[ [  25,  20 ],
  [  32,  25 ] ]
gap> Display(C*B);
[ [   5,  14,  23 ],
  [   8,  29,  50 ],
  [   4,  10,  16 ] ]
gap> Display(A+C*B);
[ [   6,  15,  24 ],
  [   8,  30,  51 ],
  [   4,  10,  17 ] ]
gap> Display(2*A-5*C*B);
[ [   -23,   -68,  -113 ],
  [   -40,  -143,  -248 ],
  [   -20,   -50,   -78 ] ]
```

To construct a null matrix, one uses the function `NullMat`. The identity is constructed with the function `IdentityMat`. To construct diagonal matrices, one uses `DiagonalMat`. Let us see some examples:

```
gap> Display(NullMat(2,3));
[ [  0,  0,  0 ],
  [  0,  0,  0 ] ]
gap> Display(IdentityMat(3));
[ [  1,  0,  0 ],
  [  0,  1,  0 ],
  [  0,  0,  1 ] ]
gap> Display(DiagonalMat([1,2]));
[ [  1,  0 ],
  [  0,  2 ] ]
```

For a given matrix `m`, we know that `m[i][j]` returns the (i,j)-th element of our matrix. To extract submatrices from a matrix, one uses `m{rows}{columns}` as the following example shows:

```
gap> m := [\
> [1, 2, 3, 4, 5],\
> [6, 7, 8, 9, 3],\
> [3, 2, 1, 2, 4],\
> [7, 5, 3, 0, 0],\
> [0, 0, 0, 0, 1]];
gap> m{[2, 4, 5]}{[1, 3]};
[ [ 6, 8 ], [ 7, 3 ], [ 0, 0 ] ]
```

It is possible to work with matrices with coefficients in arbitrary rings. We start by working with matrices over the finite field $\mathbb{F}_5$ of five elements:

```
gap> m := [[1, 2, 3], [3, 2, 1], [0, 0, 2]]*One(GF(5));
[ [ Z(5)^0, Z(5), Z(5)^3 ],
  [ Z(5)^3, Z(5), Z(5)^0 ],
  [ 0*Z(5), 0*Z(5), Z(5) ] ]
gap> Display(m);
 1 2 3
 3 2 1
 . . 2
```

Now let us work with 3×3 matrices with coefficients in the ring $\mathbb{Z}/4$ of integers modulo 4. Let us compute the identity of $M_3(\mathbb{Z}/4)$:

```
gap> m := IdentityMat(3, ZmodnZ(4));;
gap> Display(m);
matrix over Integers mod 4:
[ [  1,  0,  0 ],
  [  0,  1,  0 ],
  [  0,  0,  1 ] ]
```

One uses the function `Inverse` to compute the inverse of an invertible (square) matrix. This function returns `fail` if the matrix is not invertible. `IsIdentityMat` returns either **`true`** if the argument is the identity matrix or **`false`** otherwise. We also use `TransposedMat` to compute the transpose of a matrix:

```
gap> m := [[1, -2, -1], [0, 1, 0], [1, -1, 0]];;
gap> Display(Inverse(m));
[ [   0,   1,   1 ],
  [   0,   1,   0 ],
  [  -1,  -1,   1 ] ]
gap> Inverse([[1, 0], [2, 0]]);
fail
gap> IsIdentityMat(m*Inverse(m));
true
gap> Display(TransposedMat(m)*m);
[ [   2,  -3,  -1 ],
  [  -3,   6,   2 ],
  [  -1,   2,   1 ] ]
```

Creating a random matrix is easy: `RandomMat` returns a random rectangular matrix over a given ring, which defaults to $\mathbb{Z}$. With `RandomInvertibleMat` (resp. with `RandomUnimodularMat`) one creates a random square matrix with integer entries invertible (resp. over the integers).

```
gap> RandomMat(2, 2);
[ [ 2, 0 ], [ 4, -1 ] ]
gap> RandomInvertibleMat(2);
[ [ 1, -2 ], [ -1, 0 ] ]
gap> Inverse(last);
[ [ 0, -1 ], [ -1/2, -1/2 ] ]
gap> RandomUnimodularMat(3);
[ [ 5, -15, 28 ], [ -2, 6, -11 ], [ -11, 32, -60 ] ]
gap> Inverse(last);
[ [ -8, -4, -3 ], [ 1, 8, -1 ], [ 2, 5, 0 ] ]
```

Example 1.19. An easy induction exercise shows the Fibonacci sequence (f_n) can be computed using

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n = \begin{pmatrix} f_{n-1} & f_n \\ f_n & f_{n+1} \end{pmatrix}, \qquad n \geq 1.$$

We use this trick to compute (very efficiently) Fibonacci numbers:

```
gap> MyFibonacci := function(n)
> local m;
> m := [[0, 1], [1, 1]]^n;;
> return m[1][2];
> end;
function( n ) ... end
gap> MyFibonacci(10);
55
gap> MyFibonacci(100000);
<integer 259...875 (20899 digits)>
```

One can easily compute the characteristic and minimal polynomial of a matrix with `CharacteristicPolynomial` and `MinimalPolynomial`, respectively:

```
gap> a := [[1, 0], [0, 1]];;
gap> b := [[1, 1], [0, 1]];;
gap> CharacteristicPolynomial(Rationals, Rationals, a);
x_1^2-2*x_1+1
gap> MinimalPolynomial(Rationals, a);
x_1-1
gap> CharacteristicPolynomial(Rationals, Rationals, b);
x_1^2-2*x_1+1
gap> MinimalPolynomial(Rationals, b);
x_1^2-2*x_1+1
```

Let A be an $m \times n$ and let b be an $1 \times n$ matrix (say, a row vector of length n). `SolutionMat` returns (if possible) some x of size $1 \times m$ such that $xA = b$. If there is no x such that $xA = b$, the function returns `fail`. Let us see some examples:

```
gap> # Only one solution
gap> SolutionMat([[1, 2], [3, 4]], [13, 2]);
[ -23, 12 ]
gap> # Infinite solutions
gap> SolutionMat([[1, 2], [2, 4]], [1, 2]);
```

```
[ 1, 0 ]
gap> # No solutions
gap> SolutionMat([[1, 2], [2, 4]], [1, 3]);
fail
```

The trace of a square matrix is computed with `Trace`, the determinant with `Determinant`, and the rank with `Rank`. Examples:

```
gap> m := [[1, 2, 3], [5, 4, 3], [0, 0, 2]];;
gap> Determinant(m);
-12
gap> Trace(m);
7
gap> Rank(m);
3
gap> Rank(NullMat(2, 2));
0
gap> Rank([[1, 2, 3], [4, 5, 6], [6, 7, 8]]);
2
gap> Rank([[1, 2, 3], [4, 5, 6], [6, 7, 9]]);
3
```

The function `NullspaceMat` computes the vector space generated by the solutions of $xA = 0$, where x is a matrix of size $1 \times m$ and A is a matrix of size $m \times n$.

Remark 1.10. It is very important to remember that GAP always acts on the right. This means that vectors in general will be row vectors. This is crucial to understand how functions such as `NullspaceMat`, `Eigenvectors`, `SolutionMat` work.

1.7 Polynomials

One can define polynomial rings with `PolynomialRing`. Indeterminates are defined with `IndeterminatesOfPolynomialRing`. As an example, we define the polynomial $\mathbb{Q}[x]$ in one variable x over the ring $\mathbb{Q}$:

```
gap> A := PolynomialRing(Rationals, ["x"]);;
gap> x := IndeterminatesOfPolynomialRing(A)[1];
x
```

Let us see some operations on polynomials. One uses the function `Value` to evaluate polynomials. With `Factors`, one obtains the factorization of a polynomial over the default ring where the polynomial is defined. `RootsOfPolynomial` returns all the roots of the polynomial. Here we have some examples:

```
gap> x := Indeterminate(Rationals);;
gap> DefaultRing(x);
Rationals[x]
gap> IsPolynomial(x^2+x-2);
true
gap> IsPolynomial((x^2+x-2)/(x-1));
```

```
true
gap> AsPolynomial((x^2+x-2)/(x-1));
x+2
gap> Value(x^2+x-2, 0);
-2
gap> Value(x^2+x-2, 1);
0
gap> Value(x^2+x-2, -1);
-2
gap> Factors(x^2+x-2);
[ x-1, x+2 ]
gap> RootsOfPolynomial(x^2+x-2);
[ 1, -2 ]
gap> Derivative(x^2+2*x+1);
2*x_1+2
```

The degree of a polynomial can be obtained with `Degree`, and the coefficients with the function `CoefficientsOfUnivariatePolynomial`. The example below needs no further explanation:

```
gap> x := Indeterminate(Rationals);;
gap> f := x^5+2*x^3+3*x^2+4;;
gap> Degree(f);
5
gap> CoefficientsOfUnivariatePolynomial(f);
[ 4, 0, 3, 2, 0, 1 ]
gap> LeadingCoefficient(f);
1
```

Example 1.20. Let $f = 21x^2 + 9$ and $g = 20x^4 + 10x$ be polynomials in $(\mathbb{Z}/30)[x]$. Let us check that $fg = 0$:

```
gap> x := Indeterminate(Integers mod 30);;
gap> f := 21*x^2+9;;
gap> g := 20*x^4+10*x;;
gap> f*g;
ZmodnZObj(0,30)
gap> IsZero(f*g);
true
```

Example 1.21. Let us check that $2x^2 + 1$ divides $6x^3 + 10x^2 + 3x + 5$ in $\mathbb{Q}[x]$:

```
gap> x := Indeterminate(Rationals);;
gap> (6*x^3+10*x^2+3*x+5) mod (2*x^2+1);
0
gap> (6*x^3+10*x^2+3*x+5)/(2*x^2+1);
3*x+5
```

Example 1.22. Let us factorize in $\mathbb{Q}[x]$ the polynomial $f = 2x^5 + 3x^4 - x^2 - 2x + 1$ and prove that

$$2x^5 + 3x^4 - x^2 - 2x + 1 = (2x - 1)(x^2 + x - 1)(x^2 + x + 1).$$

Here is the code:

```
gap> x := Indeterminate(Rationals);;
gap> Factors(2*x^5+3*x^4-x^2-2*x+1);
[ 2*x-1, x^2+x-1, x^2+x+1 ]
```

Now we see that a cubic root of one is a root of our f. Let us try to factorize f over $\mathbb{Q}(\omega)$, where ω is a cubic root of one:

```
gap> Factors(PolynomialRing(Field(E(3)), "x"),\
> 2*x^5+3*x^4-x^2-2*x+1);
[ 2*x-1, x+(-E(3)), x+(-E(3)^2), x^2+x-1 ]
```

Example 1.23. Let $f = x^2+5x+2$ and $g = x^4+1$. We check that $3f-2g$ is irreducible (over the integers):

```
gap> x := Indeterminate(Integers, "x");;
gap> y := Indeterminate(Integers, "y");;
gap> f := x^2+5*x+2;;
gap> g := x^4+1;;
gap> 3*f-2*g;
-2*x^4+3*x^2+15*x+4
gap> IsIrreducible(last);
true
```

Example 1.24. The polynomial $x^2 - x + 41$ gives a prime for $x \in \{0, \dots, 40\}$. Let us check this:

```
gap> x := Indeterminate(Rationals);;
gap> p := x^2-x+41;;
gap> List([0..40], j->Value(p, j));
[ 41, 41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151,
  173, 197, 223, 251, 281, 313, 347, 383, 421, 461,
  503, 547, 593, 641, 691, 743, 797, 853, 911, 971,
  1033, 1097, 1163, 1231, 1301, 1373, 1447, 1523,
  1601 ]
gap> Filtered(last, j->not IsPrime(j));
[  ]
```

1.8 Vector spaces

GAP can handle finite-dimensional vector spaces over different fields. For example, we can play with the two-dimensional rational vector space $\mathbb{Q}^2$:

```
gap> V := Rationals^2;
( Rationals^2 )
gap> Basis(V);
CanonicalBasis( ( Rationals^2 ) )
gap> Elements(last);
```

```
[ [ 0, 1 ], [ 1, 0 ] ]
gap> Dimension(V);
2
```

We can perform simple calculations with elements of the vector space:

```
gap> [1, 2, 3] in V;
false
gap> v := [1, 2];;
gap> v in V;
true
gap> w := [5, -2];;
gap> v-3*w;
[ -14, 8 ]
gap> last in V;
true
```

We now check, for example, that the set $B = \{(1,0),(2,1)\}$ is a basis of $\mathbb{Q}^2$ and that $\{(1,0),(2,0)\}$ is not:

```
gap> Basis(V, [[1, 0], [2, 0]]);
fail
gap> B := Basis(V, [[1, 0], [2, 1]]);
Basis( ( Rationals^2 ), [ [ 1, 0 ], [ 2, 1 ] ] )
gap> Elements(last);
[ [ 1, 0 ], [ 2, 1 ] ]
```

Now we write a given vector, say $(1,1)$, in the basis B:

```
gap> Coefficients(B, [2, 1]);
[ 0, 1 ]
gap> Coefficients(B, [1, 1]);
[ -1, 1 ]
```

Conversely, there is an easy way to write linear combinations of basis elements:

```
gap> LinearCombination(B, [5, 3]);
[ 11, 3 ]
```

We can do several interesting things with finite-dimensional vector spaces. We can, for example, compute the set of all $\mathbb{Q}$-linear maps $\mathbb{Q}^2 \to \mathbb{Q}^2$. Here is the code:

```
gap> hom := Hom(Rationals, V, V);;
gap> Dimension(hom);
4
gap> f := Random(hom);
<linear mapping by matrix, ( Rationals^2 ) -> ( Rationals^2 )>
gap> Display(f);
LeftModuleHomomorphismByMatrix( CanonicalBasis( ( Rationals\
^2 ) ), [ [ 0, 0 ], [ -3/2, 0 ]
 ], CanonicalBasis( ( Rationals^2 ) ) )
gap> [1, 0]^f;
[ 0, 0 ]
gap> [0, 1]^f;
[ -3/2, 0 ]
```

Now we play with the randomly chosen linear map:

```
gap> K := Kernel(f);;
gap> Dimension(K);
1
gap> Elements(Basis(K));
[ [ 1, 0 ] ]
gap> V/K;
( Rationals^1 )
gap> Range(f);
( Rationals^2 )
gap> Image(f);
<vector space over Rationals, with 2 generators>
gap> IsInjective(f);
false
gap> IsSurjective(f);
false
gap> Dimension(Range(f));
2
gap> Dimension(Image(f));
1
```

Let us verify the dimension theorem for vector spaces:

```
gap> S := Subspace(V, [[1, 2]]);;
gap> T := Subspace(V, [[5, -2]]);;
gap> Dimension(S);
1
gap> Dimension(T);
1
gap> Dimension(Intersection(S, T));
0
gap> Dimension(S+T);
2
```

1.9 Problems

1.1. Use `ChineseRem` to find (if possible) the smallest solution of

$$\begin{cases} x \equiv 3 \bmod 10, \\ x \equiv 8 \bmod 15, \\ x \equiv 5 \bmod 84. \end{cases}$$

1.2. Find (if possible) the smallest solution of

$$\begin{cases} x \equiv 29 \bmod 52, \\ x \equiv 19 \bmod 72. \end{cases}$$

1.3. Compute the $\gcd$ of 42823 and 6409.

1.4. Find $x, y \in \mathbb{Z}$ such that

$$\gcd(5033464705, 3138740337) = 5033464705x + 3138740337y.$$

1.5. Describe the following sequence: $a_1 = 3$, $a_{n+1} = 3^{a_n} \bmod 100$.

1.6. Compute $2 \cdot 4 \cdot 6 \cdots 200$.

1.7. Prove that

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{1}{9999} - \frac{1}{10000} = \frac{1}{5001} + \frac{1}{5002} + \cdots + \frac{1}{10000}.$$

1.8. Find the last two digits of 3^{400}.

1.9. Find the roots of $x^2 + x + 7 \equiv 0 \bmod m$ for $m \in \{15, 189\}$.

1.10. Write a function that returns the binary expansion of an integer. Could you do this for other bases?

1.11. Use `MinimalPolynomial` to compute the minimal polynomial of $3 + \sqrt{5}$ over the rational numbers.

1.12. For a positive integer k let a_n be the sequence given by $a_1 = \cdots = a_{k+1} = 1$ and $a_n = a_{n-k} + a_{n-k-1}$ for all $n > k+1$. Write a function depending on k that constructs the sequence a_n. For more information see `http://oeis.org/A103379`.

1.13 (Somos sequence). Write a function that returns the n-th term of a_n, where $a_0 = a_1 = a_2 = a_3 = 1$ and

$$a_n = \frac{a_{n-1}a_{n-3} + a_{n-2}^2}{a_{n-4}}$$

for all $n \geq 4$. For more information see `http://oeis.org/A006720`.

1.14. Write a function that returns $\pi(n)$, the number of prime numbers $\leq n$.

1.15. Use the function `Permuted` to write a function that shows all the anagrams of a given word.

1.16. Given a list of non-negative numbers, write a function that displays the histogram associated with this list. For example, if the argument is the list `[1,4,2]`, the function should display

```
X
XXXX
XX
```

1.17. Write a function that, given a list of words, returns the longest one.

1.18. Write a function that returns the average value of a given list of numbers.

1.19. Write a function that, given a letter, returns **`true`** if the letter is a vowel and **`false`** otherwise.

1.20. Use `CharacteristicPolynomial` to compute the characteristic polynomial of the matrix $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 2 & -1 \\ 1 & 1 & 0 \end{pmatrix}$. Can you compute the minimal polynomial of A?

1.21. Use the function `QuotientRemainder` to compute the quotient and the remainder of $f = 2x^4 + 3x^3 + 2x + 4$ and $g = 3x^2 + x + 2$ in the ring $(\mathbb{Z}/5)[x]$.

1.22. Compute $3x^{101} - 15x^{16} - 2x^7 - 5x^4 + 3x^3 + 2x^2 + 1 \bmod x^3 + 1$.

1.23. Prove that $x = 2$ is the only root in $\mathbb{Z}/5$ of $x^{1000} + 4x + 1 \in (\mathbb{Z}/5)[x]$.

1.24. Factorize $x^4 - 1$ in $(\mathbb{Z}/5)[x]$ and in $(\mathbb{Z}/7)[x]$.

1.25. Prove that $x^2 - 79x + 1601$ gives a prime number for $x \in \{0, 1, \dots, 79\}$.

1.26. Use `Value` to evaluate the polynomial $xy + x^5y^3 - 2x + y$ in $(x, y) = (123, 567)$.

1.27. Write the first 50 twin primes.

1.28. FRACTRAN is a programming language invented by J. Conway. A FRACTRAN program is simply an ordered list of positive rational numbers together with an initial positive integer input n. The program is run by updating the integer n as follows:

- For the first rational f in the list for which $nf \in \mathbb{Z}$, replace n by nf.
- Repeat this rule until no rational number in the list produces an integer when multiplied by n, then stop.

Write an implementation of the FRACTRAN language.

Starting with $n = 2$, the program

$$\frac{17}{91}, \frac{78}{85}, \frac{19}{51}, \frac{23}{38}, \frac{29}{33}, \frac{77}{29}, \frac{95}{23}, \frac{77}{19}, \frac{1}{17}, \frac{11}{13}, \frac{13}{11}, \frac{15}{2}, \frac{1}{7}, 55$$

produces the sequence

$$2, 15, 825, 725, 1925, 2275, 425, 390, 330, 290, 770 \dots$$

In 1987, J. Conway proved that this sequence contains the set $\{2^p : p \text{ prime}\}$. See `https://oeis.org/A007542` for more information.

1.29. The first terms of Conway's "look and say" sequence are the following:

```
1
11
21
1211
111221
312211
```

After guessing how each term is computed, write a script to create the first terms of the sequence.

1.30. Write

$$\begin{pmatrix} 1\,2\,3\,4\,5\,6 \\ 2\,5\,3\,4\,6\,1 \end{pmatrix}, \qquad \begin{pmatrix} 1\,2\,3\,4\,5\,6\,7\,8\,9 \\ 2\,3\,4\,5\,1\,7\,8\,9\,6 \end{pmatrix}, \qquad \begin{pmatrix} 1\,2\,3\,4\,5 \\ 3\,2\,4\,5\,1 \end{pmatrix},$$

as a product of disjoint cycles.

1.31. Write the permutations $(1\,2\,3)(4\,5)(1\,6\,2\,5)(3\,4\,1)$ and $(1\,2)(2\,4\,5)(1\,2)$ as a product of disjoint cycles.

1.32. Find a permutation τ such that

(a) $\tau^{-1}(1\,2)(3\,4)\tau = (5\,6)(1\,3)$.
(b) $\tau^{-1}(1\,2\,3)(7\,8)\tau = (2\,5\,7)(1\,3)$.
(c) $\tau^{-1}(1\,2)(3\,4)(5\,6\,7)\tau = (1\,8)(2\,3)(4\,5\,6)$.

1.33. Compute $\tau^{-1}\sigma\tau$ in the following cases:

(a) $\sigma = (1\,2\,3)$ and $\tau = (3\,4)$.
(b) $\sigma = (5\,6\,7)$ and $\tau = (1\,2)(3\,4)$.

1.34. Let $\sigma\colon \{1,\dots,9\} \longrightarrow \{1,\dots,9\}$ be given by $i \mapsto 10-i$. Write the permutation σ as a product of disjoint cycles.

1.35. For $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 2 & 3 \end{pmatrix}$ compute

$$I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \frac{1}{4!}A^4.$$

1.36. Write the function

$$(n, A) \mapsto I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \cdots + \frac{1}{n!}A^n.$$

1.37. For a positive integer n the *Hilbert matrix* H_n is defined as

$$(H_n)_{ij} = \frac{1}{i+j-1}, \quad i, j \in \{1,\dots,n\}.$$

Write the function $n \mapsto H_n$.

1.38. Use the function `KroneckerProduct` to compute

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 & 7 \\ 2 & 1 & 0 \\ 0 & 1 & 9 \end{pmatrix}.$$

1.39. Let S and T be the vector spaces (over the rationals) generated respectively by the vectors $\{(0,1,0),(0,0,1)\}$ and $\{(1,2,0),(3,1,2)\}$.

(a) Use `VectorSpace` to create S and T.
(b) Compute $\dim(S)$, $\dim(T)$, $\dim(S \cap T)$ and $\dim(S+T)$.
(c) Compute $\dim(S \otimes T)$ and find a basis of $S \otimes T$.

1.40. Let $i = \sqrt{-1}$. Write the coordinates of the vector $(1,0,1)$ in the basis given by $(2i,1,0)$, $(2,-i,1)$, $(0,1+i,1-i)$.

1.41. Use `Subspaces` to count all possible subspaces of the four-dimensional vector space over the field $\mathbb{F}_3$. What about all subspaces of dimension two?

1.42. Count the number of subspaces of $\mathbb{F}_3^4$ such that every element (x_1,x_2,x_3,x_4) of the subspace is such that $x_1^2+x_2^2+x_3^2+x_4^2=0$. Hint: Use the function `IsZero`.

1.43. *Walsh matrices* $H(2^k)$, $k \geq 1$, are defined as follows:

$$H(2) = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H(2^k) = H(2) \otimes H(2^{k-1}), \quad k \geq 1,$$

Construct the function $n \mapsto H(2^n)$.

1.44. Use the functions `Eigenvalues` and `Eigenvectors` to compute the eigenvalues and eigenvectors, respectively, of the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{pmatrix} \in \mathbb{Q}^{3\times 3}.$$

The function `Eigenvectors` returns generators of the eigenspaces, where $v \neq 0$ is an eigenvector of A with eigenvalue λ if and only if $vA = \lambda v$.

1.45. Use the function `NullspaceMat` to compute the null space of the matrix A from Problem 1.44. The null space of A is defined as the set of vectors v such that $vA = 0$.

1.46. Let $i = \sqrt{-1}$. This exercise shows that one can do arithmetic in the ring $\mathbb{Z}[i]$ of Gaussian integers.

(a) Prove that $1+3i \nmid 4+i$.
(b) Apply the division algorithm to $\alpha = 2+7i$ and $\beta = 1+2i$.
(c) Compute $\gcd(7+17i, 8-14i)$.

(d) Are $2 + 3i$, 2 and 3 prime elements in $\mathbb{Z}[i]$?
(e) Factorize $8 - 14i$ and $-39 + 48i$.

1.47. Write a function that checks whether a polynomial is monic, i.e., the leading coefficient is one. You may want to use the functions `One`, `CoefficientsRing`, `DefaultRing`, `LeadingCoefficient`.

1.48. Let K be the field of three elements. Find the number of monic irreducible polynomials in $K[X]$ of degree up to six.

1.49. Let $a_0 = a_1 = 1$ and $a_{n+1} = 3a_n + 4a_{n-1}$ for $n \geq 1$. Compute a_{100} and a_{1000}.

1.50. Let $\omega \in \mathbb{C}$ be a primitive 7-th root of one. Use the functions `CF`, `Subfields` and `GaloisGroup` to prove that the lattice of subfields of $\mathbb{Q}(\omega)$ is given by the following diagram:

$$\begin{tikzcd}
 & \mathbb{Q}(\omega) \arrow[dl, no head] \arrow[dr, no head] & \\
\mathbb{Q}(\omega + \omega^2 + \omega^4) \arrow[dr, no head] & & \mathbb{Q}(\omega + \omega^{-1}) \arrow[dl, no head] \\
 & \mathbb{Q} &
\end{tikzcd}$$

1.51. Let $\omega \in \mathbb{C}$ be a 20-th root of one. Can you construct the lattice of subfields of $\mathbb{Q}(\omega)$?

Chapter 2
Basic group theory

One of the most remarkable implementations in GAP concerns groups. We can construct groups from generators as permutations or even matrices with coefficients in some ring. Moreover, we can define finitely presented groups and make abstract computations on them. We quickly see that many basic operations can be performed, such as asking for the order of a group, taking generators, subgroups, indexes, normalizers, centralizers, centers, etc. We can also investigate group theoretical properties such as nilpotency, solvability, simplicity, etc.

In this chapter, we will see how to work with some traditional groups such as symmetric and alternating groups, cyclic groups, dihedral groups, abelian groups and even linear groups. Indeed, we can represent these groups in, roughly, two ways: as matrices groups and as permutation groups. Usually, the latter representation has a better performance during computations. We explore some standard group properties and construct group homomorphisms, quotients, products, semidirect products, and conjugacy classes of elements and subgroups. For solvable groups, we perform more specific computations such as the calculation of Hall subgroups and representatives of conjugacy classes of subgroups via functions whose algorithms are designed for this particular class of groups. We will also study group actions on finite sets and other groups, and compute and analyze properties of orbits and stabilizers in several examples. In particular, we explore some actions on cosets. Finally, we close the chapter by introducing group presentations and explaining some examples. We will show that, although we may get infinite finitely presented groups, we can still perform computations and examine certain properties.

2.1 Basic constructions

A *matrix group* is a subgroup of $\mathbf{GL}_n(R)$ for some positive integer n and some ring R. A *permutation group* is a subgroup of some $\mathbb{S}_n$. One can construct groups from a list of generators with the function `Group`.

Example 2.1. With `Order`, we compute the order of the following groups.

(a) The group generated by the transposition $(1\,2)$.
(b) The group generated by the 5-cycle $(1\,2\,3\,4\,5)$.
(c) The group generated by the permutations $\{(1\,2),(1\,2\,3\,4\,5)\}$.

Here is the code:

```
gap> Order(Group([(1,2)]));
2
gap> Order(Group([(1,2,3,4,5)]));
5
gap> Order(Group([(1,2), (1,2,3,4,5)]));
120
```

For a positive integer n, let C_n be the (multiplicative) cyclic group of order n. One can construct cyclic groups with `CyclicGroup`. With no extra arguments, this function returns an abstract representation of a cyclic group.

Example 2.2. Let us construct the cyclic group C_2 of size two as an abstract group, as a matrix group and as a permutation group:

```
gap> CyclicGroup(2);
<pc group of size 2 with 1 generators>
gap> CyclicGroup(IsMatrixGroup, 2);
Group([ [ [ 0, 1 ], [ 1, 0 ] ] ])
gap> FieldOfMatrixGroup(last);
Rationals
gap> CyclicGroup(IsPermGroup, 2);
Group([ (1,2) ])
```

We can also specify over which field we construct matrix groups:

```
gap> C5 := CyclicGroup(IsMatrixGroup, GF(2), 5);;
gap> Display(Random(C5));
 1 . . . .
 . 1 . . .
 . . 1 . .
 . . . 1 .
 . . . . 1
gap> Display(Random(C5));
 . . 1 . .
 . . . 1 .
 . . . . 1
 1 . . . .
 . 1 . . .
gap> FieldOfMatrixGroup(C5);
GF(2)
```

With `TrivialGroup`, one constructs the trivial group:

```
gap> TrivialGroup(IsPermGroup);
Group(())
gap> TrivialGroup();
<pc group of size 1 with 0 generators>
```

Do not use the equal sign (=) to check whether a group is trivial or not:

```
gap> TrivialGroup(IsPermGroup) = TrivialGroup();
false
gap> IsTrivial(TrivialGroup(IsPermGroup));
true
gap> IsTrivial(TrivialGroup());
true
```

When using GAP in teaching, it is often desirable to have a friendly output. This can be achieved by turning on the teaching mode:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
gap> Elements(CyclicGroup(5));
[ <identity ...>, a, a^2, a^3, a^4 ]
```

We can also construct finite (elementary) abelian groups. Note that here our groups are multiplicative.

Example 2.3. We construct the multiplicative group C_4 using the function to construct abelian group:

```
gap> TeachingMode(false);
#I  Teaching mode is turned OFF
gap> AbelianGroup([4]);
<pc group of size 4 with 1 generators>
gap> Elements(last);
[ <identity> of ..., f1, f2, f1*f2 ]
```

Again we can use the teaching mode:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
gap> AbelianGroup([4]);
<fp group of size 4 on the generators [ f1 ]>
gap> Elements(last);
[ <identity ...>, f1, f1^2, f1^3 ]
```

More generally, we can construct any finite abelian group by specifying the orders of the cyclic factors.

Example 2.4. We construct a big finite abelian group:

```
gap> AbelianGroup([1234, 567890]);;
gap> StructureDescription(last);
"C350388130 x C2"
```

Example 2.5. We construct the group $C_2^2 \times C_3$:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
gap> A := AbelianGroup([2, 2, 3]);
<fp group of size 12 on the generators [ f1, f2, f3 ]>
gap> Elements(A);
[ <identity ...>, f1, f2, f3, f1*f2, f1*f3, f2*f3, f3^2,
  f1*f2*f3, f1*f3^2, f2*f3^2, f1*f2*f3^2 ]
gap> StructureDescription(A);
"C6 x C2"
```

For concrete calculations, we may use a permutation representation of our abelian group:

```
gap> A := AbelianGroup(IsPermGroup, [2, 2, 3]);
Group([ (1,2), (3,4), (5,6,7) ])
gap> for a in A do
> Display(a);
> od;
()
(5,7,6)
(5,6,7)
(3,4)
(3,4)(5,7,6)
(3,4)(5,6,7)
(1,2)
(1,2)(5,7,6)
(1,2)(5,6,7)
(1,2)(3,4)
(1,2)(3,4)(5,7,6)
(1,2)(3,4)(5,6,7)
```

As before, we use `IsPermGroup` to construct the abelian group $C_2^2 \times C_3$ as a permutation group.

Why permutation groups? Their elements may be easier to manipulate, and several algorithms are particularly good for permutation groups.

Example 2.6. We construct the elementary abelian group C_2^3 and verify that the square of every element of the group is the identity:

```
gap> ElementaryAbelianGroup(IsPermGroup, 8);
Group([ (1,2), (3,4), (5,6) ])
gap> Elements(last);
[ (), (5,6), (3,4), (3,4)(5,6), (1,2), (1,2)(5,6), (1,2)(3,4),
  (1,2)(3,4)(5,6) ]
gap> Number(last, x->IsOne(x^2));
8
```

For a positive integer n, the *dihedral group* of order $2n$ is the group

$$\mathbb{D}_{2n} = \langle r, s \mid srs = r^{-1},\ s^2 = r^n = 1 \rangle.$$

To construct dihedral groups, we use `DihedralGroup`. By default, the function returns an abstract representation of a dihedral group. As we did before in the case of abelian groups, we can construct dihedral groups as permutation groups.

Example 2.7. Let us construct $\mathbb{D}_6$, compute its order and check that it is not an abelian group:

```
gap> D6 := DihedralGroup(6);;
gap> Order(D6);
6
gap> IsAbelian(D6);
false
```

We display the elements of the group we have constructed and contrast this with the representation of the elements when we construct $\mathbb{D}_6$ as a permutation group:

```
gap> Elements(DihedralGroup(6));
[ <identity> of ..., f1, f2, f1*f2, f2^2, f1*f2^2 ]
gap> Elements(DihedralGroup(IsPermGroup, 6));
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
```

When the teaching mode is active, we get a more familiar representation of the dihedral groups:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
gap> D6 := DihedralGroup(6);
<fp group of size 6 on the generators [ r, s ]>
gap> Elements(D6);
[ <identity ...>, r^-1, r, s, r*s, s*r ]
gap> for x in D6 do
> Print("The element ", x, " has order ", Order(x), "\n");
> od;
The element <identity ...> has order 1
The element r has order 3
The element r^-1 has order 3
The element s has order 2
The element r*s has order 2
The element s*r has order 2
```

One can construct the symmetric group $\mathbb{S}_n$ of bijective functions of the set $\{1,\dots,n\}$ with `SymmetricGroup`. To construct the alternating group $\mathbb{A}_n$ we use the command `AlternatingGroup`.

Example 2.8. Let us construct $\mathbb{S}_4$ and $\mathbb{A}_4$ and display their elements:

```
gap> S4 := SymmetricGroup(4);;
gap> A4 := AlternatingGroup(4);;
gap> Elements(A4);
[ (), (2,3,4), (2,4,3), (1,2)(3,4), (1,2,3), (1,2,4),
  (1,3,2), (1,3,4), (1,3)(2,4), (1,4,2), (1,4,3),
  (1,4)(2,3) ]
gap> Elements(S4);
```

```
[ (), (3,4), (2,3), (2,3,4), (2,4,3), (2,4), (1,2),
  (1,2)(3,4), (1,2,3), (1,2,3,4), (1,2,4,3), (1,2,4),
  (1,3,2), (1,3,4,2), (1,3), (1,3,4), (1,3)(2,4),
  (1,3,2,4), (1,4,3,2), (1,4,2), (1,4,3), (1,4),
  (1,4,2,3), (1,4)(2,3) ]
```

Now we check if some specific permutations belong to these groups:

```
gap> (1,2,3) in A4;
true
gap> (1,2) in A4;
false
gap> (1,2,3)(4,5) in S4;
false
```

Example 2.9. Let us compute the order of the elements of the group $\mathbb{S}_5$:

```
gap> S5 := SymmetricGroup(5);;
gap> Collected(List(S5, Order));
[ [ 1, 1 ], [ 2, 25 ], [ 3, 20 ], [ 4, 30 ], [ 5, 24 ],
  [ 6, 20 ] ]
```

Example 2.10. Let us show that

$$G = \left\langle \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \right\rangle$$

is a non-abelian group of order eight not isomorphic to a dihedral group. Recall that the imaginary unit $i = \sqrt{-1}$ is represented as `E(4)`. To check that G is not isomorphic to $\mathbb{D}_8$ we show that G contains a unique element of order two and that $\mathbb{D}_8$ has five elements of order two:

```
gap> a := [[0, E(4)], [E(4), 0]];;
gap> b := [[0, 1], [-1, 0]];;
gap> G := Group([a, b]);;
gap> Order(G);
8
gap> IsAbelian(G);
false
gap> Number(G, x->Order(x) = 2);
1
gap> Number(DihedralGroup(8), x->Order(x) = 2);
5
```

We can factorize elements of a group in terms of the generating set used.

Example 2.11. It is known that

$$G = \langle (1\,2), (1\,3) \rangle \simeq \mathbb{S}_3.$$

We decompose every element of our group in terms of the generators $x_1 = (1\,2)$ and $x_2 = (2\,3)$. For example,

$$(1\,2\,3) = (2\,3)(1\,2) = x_2x_1,$$

as the following code shows:

```
gap> G := Group([(1,2), (2,3)]);;
gap> Factorization(G, (1,2));
x1
gap> Factorization(G, (2,3));
x2
gap> Factorization(G, (1,2,3));
x2*x1
```

Note that the function `GeneratorsOfGroup` returns, by default, a different set of generators:

```
gap> S3 := SymmetricGroup(3);;
gap> GeneratorsOfGroup(S3);
[ (1,2,3), (1,2) ]
gap> Factorization(S3, (1,2,3));
x1
gap> Factorization(S3, (1,2));
x2
gap> Factorization(S3, (2,3));
x1*x2
```

Example 2.12. The Mathieu group M_{11} is a simple group of order 7920. It can be defined as the subgroup of $\mathbb{S}_{11}$ generated by the permutations

$$(1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11), \quad (3\,7\,11\,8)(4\,10\,5\,6).$$

Let us construct M_{11} and check with `IsSimple` that it is simple:

```
gap> a := (1,2,3,4,5,6,7,8,9,10,11);;
gap> b := (3,7,11,8)(4,10,5,6);;
gap> M11 := Group([a, b]);;
gap> Order(M11);
7920
gap> IsSimple(M11);
true
```

The command `MathieuGroup` can also be used to construct Mathieu groups:

```
gap> MathieuGroup(11);
Group([ (1,2,3,4,5,6,7,8,9,10,11), (3,7,11,8)(4,10,5,6) ])
```

Example 2.13. The function `Group` also constructs infinite groups. For example, let us consider two matrices with finite order and such that their product has infinite order:

```
gap> a := [[0, -1], [1, 0]];;
gap> b:= [[0, 1], [-1, -1]];;
gap> Order(a);
```

```
4
gap> Order(b);
3
gap> Order(a*b);
infinity
gap> Order(Group([a, b]));
infinity
```

Remark 2.1. Not always the computer will be able to determine whether an element has finite or infinite order.

With `Subgroup`, we construct the subgroup of a group generated by a list of elements. The index of a subgroup can be computed with `Index`.

Example 2.14. Let us check that the subgroup of $\mathbb{S}_3$ generated by $(1\,2)$ is $\{\mathrm{id}, (1\,2)\}$ and has index three, and the subgroup generated by $(1\,2\,3)$ is $\{\mathrm{id}, (1\,2\,3), (1\,3\,2)\}$ and has index two:

```
gap> S3 := SymmetricGroup(3);;
gap> Elements(Subgroup(S3, [(1,2)]));
[ (), (1,2) ]
gap> Index(S3, Subgroup(S3, [(1,2)]));
3
gap> Elements(Subgroup(S3, [(1,2,3)]));
[ (), (1,2,3), (1,3,2) ]
gap> Index(S3, Subgroup(S3, [(1,2,3)]));
2
```

The function `AllSubgroups` returns the list of subgroups of a given group:

```
gap> AllSubgroups(S3);
[ Group(()), Group([ (2,3) ]), Group([ (1,2) ]),
  Group([ (1,3) ]), Group([ (1,2,3) ]),
  Group([ (1,2,3), (2,3) ]) ]
```

Recall that a subset K of G is said to be *normal* if K is invariant under conjugation by elements of G, that is $gKg^{-1} \subseteq K$ for all $g \in G$. If K is a normal subgroup of G, then G/K is a group.

Example 2.15. With `IsSubgroup` we check that $\mathbb{A}_{10}$ is a subgroup of $\mathbb{S}_{10}$. With `IsNormal` we see that $\mathbb{A}_{10}$ is a subset of $\mathbb{S}_{10}$ invariant under conjugation:

```
gap> S10 := SymmetricGroup(10);;
gap> A10 := AlternatingGroup(10);;
gap> IsSubgroup(S10, A10);
true
gap> IsNormal(S10, A10);
true
gap> Index(S10, A10);
2
```

Since $\mathbb{A}_{10}$ is a normal subgroup of $\mathbb{S}_{10}$, it is possible to construct the quotient group $\mathbb{S}_{10}/\mathbb{A}_{10}$:

```
gap> Q := S10/A10;;
gap> StructureDescription(Q);
"C2"
```

We see that $\mathbb{S}_9$ is a subgroup of $\mathbb{S}_{10}$ that is not normal:

```
gap> S9 := SymmetricGroup(9);;
gap> IsSubgroup(S10, S9);
true
gap> IsNormal(S10, S9);
false
```

Example 2.16. Let us show that in $\mathbb{D}_8$ there are subgroups H and K such that K is normal in H, H is normal in $\mathbb{D}_8$, but K is not normal in $\mathbb{D}_8$:

```
gap> D8 := DihedralGroup(IsPermGroup, 8);;
gap> Elements(D8);
[ (), (2,4), (1,2)(3,4), (1,2,3,4),
  (1,3), (1,3)(2,4), (1,4,3,2),
  (1,4)(2,3) ]
gap> K := Subgroup(D8, [(2,4)]);;
gap> Elements(K);
[ (), (2,4) ]
gap> H := Subgroup(D8, [(1,2,3,4)^2, (2,4)]);;
gap> Elements(H);
[ (), (2,4), (1,3), (1,3)(2,4) ]
gap> IsNormal(D8, K);
false
gap> IsNormal(D8, H);
true
gap> IsNormal(H, K);
true
```

Example 2.17. Let us compute quotient groups of the cyclic group C_4. Since every subgroup of C_4 is normal, we can use `AllSubgroups` to check that C_4 contains a unique non-trivial proper subgroup K. The quotient C_4/K has two elements:

```
gap> C4 := CyclicGroup(IsPermGroup, 4);;
gap> AllSubgroups(C4);
[ Group(()), Group([ (1,3)(2,4) ]),
  Group([ (1,2,3,4) ]) ]
gap> K := last[2];;
gap> Order(C4/K);
2
```

Recall that for a positive integer n, the *generalized quaternion group* is the group

$$Q_{4n} = \langle x, y \mid x^{2n} = y^4 = 1, x^n = y^2, y^{-1}xy = x^{-1} \rangle.$$

We use `QuaternionGroup` to construct generalized quaternion groups. As we did before, we can use the filter `IsPermGroup` (resp. `IsMatrixGroup`) to obtain generalized quaternion groups as permutation (resp. matrix) groups.

Example 2.18. Let us check that each subgroup of the quaternion group Q_8 of order eight is normal and that Q_8 is non-abelian:

```
gap> Q8 := QuaternionGroup(IsMatrixGroup, 8);;
gap> Display(Random(Q8));
[ [   0,  -1,   0,   0 ],
  [   1,   0,   0,   0 ],
  [   0,   0,   0,   1 ],
  [   0,   0,  -1,   0 ] ]
gap> IsAbelian(Q8);
false
gap> ForAll(AllSubgroups(Q8), x->IsNormal(Q8, x));
true
```

If G is a group, its *center* is the subgroup

$$Z(G) = \{x \in G : xy = yx \text{ for all } y \in G\}.$$

The *commutator* of two elements $x, y \in G$ is defined as $[x, y] = x^{-1}y^{-1}xy$. The *commutator subgroup*, or *derived subgroup* of G, is the subgroup $[G, G]$ generated by all the commutators of G.

Example 2.19. We compute the commutator of two randomly chosen elements of the dihedral group $\mathbb{D}_{26}$ of size 26, verify that the center of $\mathbb{D}_{26}$ is trivial and compute $[\mathbb{D}_{26}, \mathbb{D}_{26}]$ with the function `DerivedSubgroup`:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
gap> D26 := DihedralGroup(26);;
gap> x := Random(D26);
r^4*s
gap> y := Random(D26);
r^5
gap> Comm(x,y); # This computes [x,y]
r^-3
gap> IsTrivial(Center(D26));
true
gap> der := DerivedSubgroup(D26);;
gap> StructureDescription(der);
"C13"
```

Example 2.20. Recall that a group G is said to be *perfect* if $[G, G] = G$. We present some examples of perfect groups:

```
ap> IsPerfect(AlternatingGroup(5));
true
gap> IsPerfect(SymmetricGroup(5));
```

```
false
gap> IsPerfect(SL(2,7));
true
gap> IsPerfect(GL(2,7));
false
```

With `CommutatorSubgroup` we can compute the *commutator subgroup*

$$[H,K] = \langle [h,k] : h \in H,\ k \in K \rangle$$

of the subgroups H and K of a given group G.

Example 2.21. We compute the commutator subgroup of two randomly chosen subgroups of $\mathbb{D}_{16}$:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
gap> D16 := DihedralGroup(16);;
gap> all := AllSubgroups(D16);
[ Group([  ]), Group([ r^4 ]), Group([ s ]), Group([ r^2*s ]),
  Group([ s*r^2 ]), Group([ s*r^4 ]), Group([ r*s ]),
  Group([ s*r ]), Group([ r^3*s ]), Group([ s*r^3 ]),
  Group([ r^-2 ]), Group([ s, r^4 ]), Group([ s*r^2, r^4 ]),
  Group([ r*s, r^4 ]), Group([ s*r, r^4 ]), Group([ r^-2, s ]),
  Group([ r^-2, r*s ]), Group([ r ]), Group([ r, s ]) ]
gap> H := Random(all);
Group([ s*r^3 ])
gap> K := Random(all);
Group([ s*r^4 ])
gap> CommutatorSubgroup(H, K);
Group([ r^-2 ])
```

Recall that if G is a group and $g \in G$, the *conjugacy class* of g in G is the subset $g^G = \{x^{-1}gx : x \in G\}$. The *centralizer* of g in G is the subgroup

$$C_G(g) = \{x \in G : xg = gx\}.$$

To compute conjugacy classes, we have the functions `ConjugacyClasses` and `ConjugacyClass`; the centralizer can be computed with `Centralizer`.

Example 2.22. Let us check that $\mathbb{S}_3$ contains three conjugacy classes with representatives id, $(1\,2)$ and $(1\,2\,3)$, so that

$$(1\,2)^{\mathbb{S}_3} = \{(1\,2),(1\,3),(2\,3)\}, \qquad (1\,2\,3)^{\mathbb{S}_3} = \{(1\,2\,3),(1\,3\,2)\}.$$

```
gap> S3 := SymmetricGroup(3);;
gap> ConjugacyClasses(S3);
[ ()^G, (1,2)^G, (1,2,3)^G ]
gap> Elements(ConjugacyClass(S3, (1,2)));
[ (2,3), (1,2), (1,3) ]
```

```
gap> Elements(ConjugacyClass(S3, (1,2,3)));
[ (1,2,3), (1,3,2) ]
```

Let us check that $C_{\mathbb{S}_3}((1\,2\,3)) = \{\mathrm{id}, (1\,2\,3), (1\,3\,2)\}$:

```
gap> Elements(Centralizer(S3, (1,2,3)));
[ (), (1,2,3), (1,3,2) ]
```

Example 2.23. In this example, we use the function `Representative` to construct a list of representatives of conjugacy classes of $\mathbb{A}_4$:

```
gap> A4 := AlternatingGroup(4);;
gap> List(ConjugacyClasses(A4), Representative);
[ (), (1,2)(3,4), (1,2,3), (1,2,4) ]
```

With the function `IsConjugate` we can check whether two elements (or two subgroups) are conjugate. If two elements g and h are conjugate, we want to find an element x such that $g = x^{-1}hx$. For that purpose, we use `RepresentativeAction`.

Example 2.24. Let us check that $(1\,2\,3)$ and $(1\,3\,2) = (1\,2\,3)^2$ are not conjugate in the alternating group $\mathbb{A}_4$:

```
gap> A4 := AlternatingGroup(4);;
gap> g := (1,2,3);;
gap> IsConjugate(A4, g, g^2);
false
```

Note that $(1\,2\,3)$ and $(1\,3\,2)$ are conjugate both in $\mathbb{S}_4$ and $\mathbb{A}_5$:

```
gap> S4 := SymmetricGroup(4);;
gap> A5 := AlternatingGroup(5);;
gap> IsConjugate(S4, g, g^2);
true
gap> IsConjugate(A5, g, g^2);
true
```

Now we find elements $x \in \mathbb{S}_4$ and $y \in \mathbb{A}_5$ such that

$$(1\,2\,3)^x = x^{-1}(1\,2\,3)x = (1\,3\,2), \quad (1\,2\,3)^y = y^{-1}(1\,2\,3)y = (1\,3\,2).$$

For that purpose, we use `RepresentativeAction`:

```
gap> x := RepresentativeAction(S4, g, g^2);
(2,3)
gap> x^(-1)*g*x = g^2;
true
gap> g^x = g^2;
true
gap> y := RepresentativeAction(A5, g, g^2);
(2,3)(4,5)
gap> g^y = g^2;
true
```

Note that there might be several possible elements $x \in \mathbb{S}_4$ such that $(1\,2\,3)^x = (1\,3\,2)$. The function `RepresentativeAction` returns *one* of such possible x.

Example 2.25. We can use `RepresentativeAction` to find an element of the group that conjugates a certain tuple $(x_1, \dots, x_n)$ to $(y_1, \dots, y_n)$, as we show below:

```
gap> G := SymmetricGroup(4);;
gap> x1 := (1,2,3);;
gap> x2 := (1,3);;
gap> y1 := (2,3,4);;
gap> y2 := (3,4);;
gap> RepresentativeAction(G, [x1, x2], [y1, y2], OnTuples);
(1,4)
gap> RepresentativeAction(G, [x1, x2], [x2, x2], OnTuples);
fail
```

Example 2.26. It is well-known that the converse of Lagrange's theorem does not hold. The following example is based on [5]. We show that $\mathbb{A}_4$ has no subgroups of order six with several different methods.

A naive idea to prove that $\mathbb{A}_4$ has no subgroups of order six is to study all the $\binom{12}{6} = 924$ subsets of $\mathbb{A}_4$ of size six and check that none of these subsets is a group:

```
gap> A4 := AlternatingGroup(4);;
gap> k := 0;;
gap> for x in Combinations(Elements(A4), 6) do
> if Size(Subgroup(A4, x)) = Size(x) then
>   k := k+1;
> fi;
> od;
gap> k;
0
```

The following code shows an equivalent way of performing the previous computation:

```
gap> ForAny(Combinations(Elements(A4), 6),\
> x->Size(Subgroup(A4, x)) = Size(x));
false
```

Now we use a similar idea. Every subgroup of order six contains exactly five non-identity elements. So we see that none of the $\binom{11}{5} = 462$ subsets of $\mathbb{A}_4$ with five elements can generate a subgroup of order six. In the subsequent code, we do not use `Combinations`. Instead, combinations will be generated by using an iterator.

```
gap> k := 0;;
gap> for t in IteratorOfCombinations(\
> Filtered(A4, x->not x = ()), 5) do
> if Size(Subgroup(A4, t)) = Size(t)+1 then
>   k := k+1;
> fi;
> od;
gap> k;
0
```

In the previous code, we used an *iterator*. Iterators provide the possibility to loop through the elements of a collection without needing to store them.

Here we have another idea: if $\mathbb{A}_4$ has a subgroup of order six, then the index of this subgroup in $\mathbb{A}_4$ is two. The function `SubgroupsOfIndexTwo` returns a list with all index-two subgroups of a given group. The function belongs to the package `LAGUNA`:

```
gap> LoadPackage("LAGUNA");;
```

This package was written by V. Bovdi, O. Konovalov, R. Rossmanith and C. Schneider, and provides functions for group algebras and their associated Lie algebras.

We now check that $\mathbb{A}_4$ has no subgroups of index two:

```
gap> SubgroupsOfIndexTwo(A4);
[  ]
```

Of course, we can simply construct all subgroups and check that there are no subgroups of order six:

```
gap> List(AllSubgroups(A4), Order);
[ 1, 2, 2, 2, 3, 3, 3, 3, 4, 12 ]
gap> 6 in last;
false
```

In fact, it is enough to construct all conjugacy classes of subgroups:

```
gap> c := ConjugacyClassesSubgroups(A4);;
gap> List(c, x->Order(Representative(x)));
[ 1, 2, 3, 4, 12 ]
gap> 6 in last;
false
```

Moreover, since a subgroup of index two is necessarily maximal, it is enough to look for conjugacy classes of maximal subgroups:

```
gap> cm := ConjugacyClassesMaximalSubgroups(A4);;
gap> List(cm, x->Order(Representative(x)));
[ 4, 3 ]
gap> 6 in last;
false
```

Another approach is to use conjugacy classes of elements in $\mathbb{A}_4$. Indeed, the conjugacy classes of $\mathbb{A}_4$ are:

$$\begin{aligned}
&\{\mathrm{id}\}, && \{(2\,4\,3),(1\,2\,3),(1\,3\,4),(1\,4\,2)\},\\
&\{(1\,2)(3\,4),(1\,3)(2\,4),(1\,4)(2\,3)\}, && \{(2\,3\,4),(1\,2\,4),(1\,3\,2),(1\,4\,3)\}.
\end{aligned}$$

This is how we construct the conjugacy classes of $\mathbb{A}_4$:

```
gap> ConjugacyClasses(A4);
[ ()^G, (1,2)(3,4)^G, (1,2,3)^G, (1,2,4)^G ]
gap> Elements(ConjugacyClass(A4, ()));
[ () ]
```

```
gap> Elements(ConjugacyClass(A4, (1,2)(3,4)));
[ (1,2)(3,4), (1,3)(2,4), (1,4)(2,3) ]
gap> Elements(ConjugacyClass(A4, (1,2,3)));
[ (2,4,3), (1,2,3), (1,3,4), (1,4,2) ]
gap> Elements(ConjugacyClass(A4, (1,2,4)));
[ (2,3,4), (1,2,4), (1,3,2), (1,4,3) ]
```

Assume that $\mathbb{A}_4$ has a subgroup K of order six. Then K has index two in $\mathbb{A}_4$ and hence it is normal in $\mathbb{A}_4$. This means that K is a union of conjugacy classes of $\mathbb{A}_4$ and that $\{1\} \subseteq K$. This is a contradiction!

Let us now use the commutator to prove that $\mathbb{A}_4$ has no subgroups of order six. If there exists a subgroup K of order six, then K is normal in $\mathbb{A}_4$ and the quotient $\mathbb{A}_4/K$ is cyclic of order two. This implies that

$$[\mathbb{A}_4, \mathbb{A}_4] = \{\mathrm{id}, (1\,2)(3\,4), (1\,3)(2\,4), (1\,4)(2\,3)\},$$

is contained in K, a contradiction since 4 does not divide 6:

```
gap> DerivedSubgroup(A4);
Group([ (1,4)(2,3), (1,3)(2,4) ])
gap> Elements(last);
[ (), (1,2)(3,4), (1,3)(2,4), (1,4)(2,3) ]
```

One more variation. If K is a subgroup of $\mathbb{A}_4$ of order six, then there are two possibilities: either $K \simeq \mathbb{S}_3$ or $K \simeq C_6$. The group $\mathbb{A}_4$ has no elements of order six:

```
gap> Filtered(A4, x->Order(x) = 6);
[  ]
```

Then $K \simeq \mathbb{S}_3$ and hence K contains three elements of order two. Thus

$$\{\mathrm{id}, (1\,2)(3\,4), (1\,3)(2\,4), (1\,4)(2\,3)\}$$

is a subgroup of K of order four, which contradicts Lagrange's theorem.

To study isomorphisms between finite groups one uses `IsomorphismGroups`. This function returns `fail` if the groups are not isomorphic, or some isomorphism otherwise. The following trick works as an isomorphism test:

```
gap> S3 := SymmetricGroup(3);;
gap> C6 := CyclicGroup(6);;
gap> C2xC3 := AbelianGroup([2, 3]);;
gap> not IsomorphismGroups(S3, C6) = fail;
false
gap> not IsomorphismGroups(C2xC3, C6) = fail;
true
```

Since we use this verification several times, we can turn the trick into a function:

```
gap> AreIsomorphic := function(G, H)
> return not IsomorphismGroups(G, H) = fail;
> end;
function( G, H ) ... end
```

```
gap> AreIsomorphic(C2xC3, C6);
true
gap> AreIsomorphic(S3, C6);
false
```

Example 2.27. Let us construct a group G such that $G = \langle a\rangle \times \langle b\rangle$ with $C_4 \simeq \langle a\rangle$ and $C_2 \simeq \langle b\rangle$. We also prove that

$$\langle a^2\rangle \simeq \langle b\rangle, \qquad G/\langle a^2\rangle \not\simeq G/\langle b\rangle.$$

Here is the code that shows our claims:

```
gap> G := AbelianGroup(IsPermGroup, [4, 2]);
Group([ (1,2,3,4), (5,6) ])
gap> K := Subgroup(G, [(5,6)]);;
gap> L := Subgroup(G, [(1,2,3,4)^2]);;
gap> IsomorphismGroups(K, L);
[ (5,6) ] -> [ (1,3)(2,4) ]
gap> IsomorphismGroups(G/K, G/L);
fail
```

One can show that:

$$\langle a^2\rangle \simeq \langle b\rangle \simeq C_2, \qquad G/\langle a^2\rangle \simeq C_4, \qquad G/\langle b\rangle \simeq C_2 \times C_2.$$

We can also work with classical groups. Use

```
gap> ?classical groups
```

to get more information.

Example 2.28. One can use the function `GL` to construct some general linear groups such as $\mathbf{GL}_n(\mathbb{Z})$, $\mathbf{GL}_n(\mathbb{Z}/m)$ and $\mathbf{GL}_n(\mathbb{F}_q)$:

```
gap> Order(GL(2,Integers));
infinity
gap> Order(GL(2,ZmodnZ(4)));
96
gap> Order(GL(2,GF(4)));
180
gap> Order(GL(3,GF(4)));
181440
```

Similarly, with `SL` one constructs $\mathbf{SL}_n(\mathbb{Z})$, $\mathbf{SL}_n(\mathbb{Z}/m)$ and $\mathbf{SL}_n(\mathbb{F}_q)$:

```
gap> Order(SL(2,GF(3)));
24
gap> Order(SL(2,Integers));
infinity
gap> Order(SL(2,ZmodnZ(4)));
48
gap> Order(SL(2,GF(4)));
60
```

We write $\mathbf{GL}_n(q)$ to denote $\mathbf{GL}_n(\mathbb{F}_q)$. We use the same notation for other classical groups. Following this convenient notation for constructing linear groups over finite fields, we can specify the size of the field:

```
gap> G := GL(3,4);;
gap> Size(G);
181440
gap> Center(G);
<group of 3x3 matrices over GF(2^2)>
gap> StructureDescription(last);
"C3"
gap> IsSubgroup(G, SL(3,4));
true
gap> DerivedSubgroup(G) = SL(3,4);
true
```

We now compute the quotient of $\mathbf{GL}_3(4)$ by its center. Note that GAP will return a permutation representation of this quotient. Also here `StructureDescription` shows something that might be different from what we expect:

```
gap> Q := GL(3,4)/Center(GL(3,4));;
gap> Order(Q);
60480
gap> StructureDescription(Q);
"PSL(3,4) : C3"
gap> not IsomorphismGroups(Q, PGL(3,4)) = fail;
true
```

Since the function `IsomorphismGroups` does not return `fail`, our groups are indeed isomorphic.

It is known that the commutator of a finite group is not always equal to the set of commutators, as we show in the following example based on [8]:

Example 2.29. Let G be the subgroup of $\mathbb{S}_{16}$ generated by the permutations

$$\begin{aligned} a &= (1\,3)(2\,4), & b &= (5\,7)(6\,8),\\ c &= (9\,11)(10\,12), & d &= (13\,15)(14\,16),\\ e &= (1\,3)(5\,7)(9\,11), & f &= (1\,2)(3\,4)(13\,15),\\ g &= (5\,6)(7\,8)(13\,14)(15\,16), & h &= (9\,10)(11\,12). \end{aligned}$$

We show that $[G,G]$ has order 16:

```
gap> a := (1,3)(2,4);;
gap> b := (5,7)(6,8);;
gap> c := (9,11)(10,12);;
gap> d := (13,15)(14,16);;
gap> e := (1,3)(5,7)(9,11);;
gap> f := (1,2)(3,4)(13,15);;
gap> g := (5,6)(7,8)(13,14)(15,16);;
gap> h := (9,10)(11,12);;
gap> G := Group([a,b,c,d,e,f,g,h]);;
gap> D := DerivedSubgroup(G);;
gap> Size(D);
16
```

We now show that the set of commutators has 15 elements. In particular, we show that $cd \in [G, G]$ and that cd is not a commutator:

```
gap> Size(Set(Cartesian(G, G), Comm));
15
gap> c*d in Difference(D, Set(Cartesian(G, G), Comm));
true
```

To conclude this section, we show how to use `SylowSubgroup` to compute with Sylow subgroups.

Example 2.30. We compute a Sylow subgroup of the linear group $\mathbf{SL}_4(16)$:

```
gap> G := SL(4,16);;
gap> Order(G);
1148120010326016000
gap> SylowSubgroup(G, 2);
<group of 4x4 matrices of size 16777216 over GF(2^4)>
```

The calculation of Sylow subgroups of $\mathbf{SL}_4(16)$ is about 30% faster when using a permutation representation:

```
gap> G := SL(4,16);;
gap> n := Order(G);;
gap> List(PrimeDivisors(n), p->SylowSubgroup(G, p));;time;
9605
gap> G := SL(IsPermGroup, 4, 16);;
gap> List(PrimeDivisors(n), p->SylowSubgroup(G, p));;time;
6192
```

Example 2.31. Let $G = \mathbf{SL}_2(3)$, $P \in \mathrm{Syl}_2(G)$ and $Q \in \mathrm{Syl}_3(G)$. The following code shows that $G = PQ$:

```
gap> G := SL(2,3);;
gap> P := SylowSubgroup(G, 2);;
gap> Q := SylowSubgroup(G, 3);;
gap> PQ := List(Cartesian(P,Q), x->x[1]*x[2]);;
gap> Size(PQ) = Size(G);
true
```

We now compute all Sylow 3-subgroups of G:

```
gap> sylows3 := ConjugacyClassSubgroups(G, Q);;
gap> Size(sylows3);
4
gap> for x in sylows3 do
> Display(x);
> od;
Group([ [ [ Z(3)^0, Z(3)^0 ], [ 0*Z(3), Z(3)^0 ] ] ])
Group([ [ [ Z(3), Z(3)^0 ], [ Z(3), 0*Z(3) ] ] ])
Group([ [ [ 0*Z(3), Z(3)^0 ], [ Z(3), Z(3) ] ] ])
Group([ [ [ Z(3)^0, 0*Z(3) ], [ Z(3), Z(3)^0 ] ] ])
```

The function `Normalizer` returns the normalizer of a subgroup:

```
gap> N := Normalizer(G, Q);;
gap> Index(G, N);
4
```

Indeed, N is a Borel subgroup of G, and we can check that it consists of the upper-triangular matrices of determinant one:

```
gap> for x in N do
> Display(x);
> Print("--\n");
> od;
 1 .
 . 1
--
 1 2
 . 1
--
 1 1
 . 1
--
 2 .
 . 2
--
 2 1
 . 2
--
 2 2
 . 2
--
gap> N = Filtered(Elements(G), x->x[2][1] = 0*Z(3));
true
```

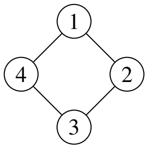
2.2 Group actions

In this section, we present several examples of group actions. In different well-known situations, we compute orbits, stabilizers, cores, and permutation representations.

Example 2.32. We now explore the case of $\mathbb{S}_3$ acting by right multiplication on the set of right cosets of $\mathbb{A}_3$ in $\mathbb{S}_3$:

```
gap> S3 := SymmetricGroup(3);;
gap> A3 := AlternatingGroup(3);;
gap> omega := RightCosets(S3, A3);
[ RightCoset(Alt( [ 1 .. 3 ] ),()),
  RightCoset(Alt( [ 1 .. 3 ] ),(2,3)) ]
gap> for x in omega do
> Display(Elements(x));
> od;
[ (), (1,2,3), (1,3,2) ]
[ (2,3), (1,2), (1,3) ]
gap> Size(Orbits(S3, omega, OnRight));
1
gap> Stabilizer(S3, Random(omega), OnRight);
Group([ (1,2,3) ])
```

Example 2.33. We now study the natural action of the dihedral group $\mathbb{D}_8$ of order eight on the vertex set $\{1,2,3,4\}$ of the square:



We show, for example, that the action is transitive:

```
gap> D8 := DihedralGroup(IsPermGroup, 8);;
gap> GeneratorsOfGroup(D8);
[ (1,2,3,4), (2,4) ]
gap> # Here r=(1,2,3,4) and s=(2,4)
gap> Orbits(D8, [1..4]);
[ [ 1, 2, 3, 4 ] ]
gap> IsTransitive(D8, [1..4]);
true
gap> Stabilizer(D8, 1);
Group([ (2,4) ])
```

We now compute the *core* of the action, i.e. the set of elements of $\mathbb{D}_8$ acting trivially on $\{1,2,3,4\}$:

```
gap> Filtered(D8, g->ForAll([1..4], x->x^g = x));
[ () ]
```

Example 2.34. Let $G = \mathbf{GL}_2(5)$ and $V = \mathbb{F}_5^2$. We study the natural action (by right multiplication) of G on V:

```
gap> G := GL(2,5);;
gap> omega := AsList(GF(5)^2);;
gap> Size(Orbits(G, omega));
2
gap> v := Random(omega);
[ Z(5)^3, Z(5)^2 ]
gap> Orbit(G, v);
[ [ Z(5)^3, Z(5)^2 ], [ Z(5)^0, Z(5)^2 ],
  [ Z(5)^3, Z(5)^3 ], [ Z(5), Z(5)^2 ],
  [ 0*Z(5), Z(5)^0 ], [ Z(5)^0, Z(5)^3 ],
  [ Z(5)^2, Z(5)^3 ], [ Z(5)^2, Z(5)^2 ],
  [ Z(5)^2, Z(5) ], [ Z(5)^2, 0*Z(5) ],
  [ Z(5), Z(5)^3 ], [ Z(5)^0, Z(5)^0 ],
  [ Z(5)^3, Z(5) ], [ Z(5)^3, 0*Z(5) ],
  [ 0*Z(5), Z(5) ], [ Z(5), Z(5)^0 ],
  [ Z(5)^3, Z(5)^0 ], [ Z(5)^0, Z(5) ],
  [ 0*Z(5), Z(5)^3 ], [ Z(5)^0, 0*Z(5) ],
  [ Z(5)^2, Z(5)^0 ], [ Z(5), Z(5) ],
  [ Z(5), 0*Z(5) ], [ 0*Z(5), Z(5)^2 ] ]
gap> Stabilizer(G, v);
Group([ [ [ Z(5)^0, Z(5) ], [ 0*Z(5), Z(5) ] ],
  [ [ Z(5)^2, Z(5)^0 ], [ Z(5)^2, Z(5)^2 ] ] ])
gap> StructureDescription(last);
"C5 : C4"
```

We now construct a permutation representation corresponding to this action, that is, the image of the group homomorphism induced by the action:

```
gap> gens := GeneratorsOfGroup(G);
[ [ [ Z(5), 0*Z(5) ], [ 0*Z(5), Z(5)^0 ] ],
  [ [ Z(5)^2, Z(5)^0 ], [ Z(5)^2, 0*Z(5) ] ] ]
gap> a := gens[1];;
gap> Display(a);
 2 .
 . 1
gap> b := gens[2];;
gap> Display(b);
 4 1
 4 .
gap> alpha := PermList(List([1..Size(omega)], \
> x->Position(omega, omega[x]*a)));;
gap> alpha;
(6,11,16,21)(7,12,17,22)(8,13,18,23)(9,14,19,24)(10,15,20,25)
gap> beta := PermList(List([1..Size(omega)], \
> x->Position(omega, omega[x]*b)));;
gap> beta;
(2,16,9)(3,21,15)(4,6,17)(5,11,23)(7,22,10)(8,12,13)
(14,18,19)(20,24,25)
gap> StructureDescription(Group([alpha, beta]));
"GL(2,5)"
```

Alternatively, we can use the function `Action`:

```
gap> Action(G, GF(5)^2, OnPoints);
Group([ (6,11,16,21)(7,12,17,22)(8,13,18,23)
  (9,14,19,24)(10,15,20,25), (2,16,9)(3,21,15)
  (4,6,17)(5,11,23)(7,22,10)(8,12,13)(14,18,19)(20,24,25) ])
```

Example 2.35. We now consider the action of $G = \mathbf{GL}_2(5)$ on one-dimensional subspaces of $V = \mathbb{F}_5^2$. We prove that the core of the action is $Z(G)$ and that the quotient $G/Z(G) = \mathbf{PGL}_2(5)$ is isomorphic to $\mathbb{S}_5$:

```
gap> G := GL(2,5);;
gap> V := GF(5)^2;;
gap> Size(Subspaces(V, 1));
6
gap> core := Filtered(G, g->ForAll(Subspaces(V, 1), x->x^g = x));
[ [ [ Z(5)^0, 0*Z(5) ], [ 0*Z(5), Z(5)^0 ] ],
  [ [ Z(5)^3, 0*Z(5) ], [ 0*Z(5), Z(5)^3 ] ],
  [ [ Z(5), 0*Z(5) ], [ 0*Z(5), Z(5) ] ],
  [ [ Z(5)^2, 0*Z(5) ], [ 0*Z(5), Z(5)^2 ] ] ]
gap> center := Center(G);;
gap> Group(core) = center;
true
gap> StructureDescription(G/center);
"S5"
gap> Size(Orbits(G, Subspaces(V, 1)));
1
gap> Transitivity(G, Subspaces(V, 1));
3
gap> W := Random(Subspaces(V, 1));;
gap> Random(W);
[ Z(5)^3, Z(5) ]
gap> StructureDescription(Stabilizer(G, W));
"C4 x (C5 : C4)"
```

We now construct the permutation representation of this action:

```
gap> P := Action(G, Subspaces(V, 1), OnPoints);
Group([ (3,6,5,4), (1,2,5)(3,4,6) ])
gap> StructureDescription(P);
"S5"
gap> NrMovedPoints(P);
6
gap> Transitivity(P, [1..6]);
3
gap> StructureDescription(Stabilizer(P, 1));
"C5 : C4"
```

Example 2.36. Let $G = \mathbf{GL}_3(3)$ act on the set $\Omega = \{x \in G : |x| = 2\}$ by conjugation. We check that the action has a unique fixed point, namely the unique central matrix of order two:

```
gap> G := GL(3,3);;
gap> omega := Filtered(G, x->Order(x) = 2);;
gap> fix := Intersection(omega, Center(G));;
gap> Size(fix);
1
gap> Display(fix[1]);
 2 . .
 . 2 .
 . . 2
gap> # There is an alternative way to compute fixed points
gap> fix = FixedPoints(omega, G, OnPoints);
true
```

Computing the orbits and their representatives, we conclude that there is a unique fixed point:

```
gap> List(Orbits(G, omega, OnPoints), Size);
[ 117, 117, 1 ]
gap> # Orbit representatives
gap> for x in Orbits(G, omega, OnPoints) do
> Display(x[1]);
> Print("--\n");
> od;
 1 1 .
 . 2 .
 . . 1
--
 . . 1
 . 2 .
 1 . .
--
 2 . .
 . 2 .
 . . 2
--
gap> x := Random(omega);;
gap> StructureDescription(Stabilizer(G, x));
"C2 x GL(2,3)"
```

We now construct a permutation representation of the action. In this case, we obtain two orbits and not three, as the permutation representation overlooks fixed points. Here is the code:

```
gap> P := Action(G, omega, OnPoints);;
gap> StructureDescription(P);
"PSL(3,3)"
gap> NrMovedPoints(P);
234
gap> List(Orbits(P), Size);
[ 117, 117 ]
gap> y := Random([1..NrMovedPoints(P)]);;
gap> StructureDescription(Stabilizer(P, y));
"GL(2,3)"
```

Example 2.37. Let $G = \mathbf{SL}_2(8)$. We construct the projective action of G on the set $\mathbb{F}_8 \cup \{\infty\}$ of size nine. We use `OnLines`, which is the function of the action of G on the set Ω of vectors $(x, y) \in \mathbb{F}_8^2 \setminus \{(0, 0)\}$ with the first non-zero coordinate equal to one:

```
gap> G := SL(2,8);;
gap> w := [0, 1]*One(GF(8));
[ 0*Z(2), Z(2)^0 ]
gap> omega := Concatenation(List(GF(8), \
> x->[1, x]*One(GF(8))), [w]);;
gap> Size(omega);
9
```

```
gap> g := Random(G);;
gap> Display(g);
z = Z(8)
 z^3 z^4
 z^1 z^1
gap> v := [1, 0]*One(GF(8));
[ Z(2)^0, 0*Z(2) ]
gap> OnLines(v, g);
[ Z(2)^0, Z(2^3) ]
gap> OnLines(w, g);
[ Z(2)^0, Z(2)^0 ]
```

Note that `OnLines` and `OnRight` are different actions:

```
gap> OnLines(w, g) in omega;
true
gap> OnRight(w, g) in omega;
false
```

We now perform some calculations:

```
gap> Size(Orbits(G, omega, OnLines));
1
gap> v := Random(omega);
[ Z(2)^0, Z(2^3)^4 ]
gap> StructureDescription(Stabilizer(G, v, OnLines));
"(C2 x C2 x C2) : C7"
gap> Transitivity(G, omega, OnLines);
3
gap> StructureDescription(Action(G, omega, OnLines));
"PSL(2,8)"
```

We construct an explicit permutation representation as a subgroup of $\mathbb{S}_9$. We need a point at infinity, so let $\Sigma = \mathbb{F}_8 \cup \{\infty\}$ and write $\mathbb{F}_8^\times = \langle \zeta \rangle$ for some ζ. Let Q be the group generated by the permutations of Σ given by

$$\alpha(x) = x + 1, \quad \beta(x) = \zeta x, \quad \gamma(x) = 1/x.$$

By convention, $\alpha(\infty) = \beta(\infty) = \infty$, $\gamma(0) = \infty$ and $\gamma(\infty) = 0$. Let us construct these permutations:

```
gap> alpha := function(x)
> if x = infinity then
>   return infinity;
> else
>   return x+One(GF(8));
> fi;
> end;
function( x ) ... end
gap> beta := function(x)
> if x = infinity then
>   return infinity;
> else
>   return x*Z(8);
> fi;
> end;
function( x ) ... end
gap> gamma := function(x)
> if IsZero(x) then
>   return infinity;
> elif x = infinity then
>   return Zero(GF(8));
> else
>   return 1/x;
> fi;
> end;
function( x ) ... end
gap> sigma := Concatenation(AsList(GF(8)), [infinity]);;
gap> Size(sigma);
9
gap> a := PermList(List(sigma, x->Position(sigma, alpha(x))));
(1,2)(3,5)(4,8)(6,7)
gap> b := PermList(List(sigma, x->Position(sigma, beta(x))));
(2,3,4,5,6,7,8)
gap> c := PermList(List(sigma, x->Position(sigma, gamma(x))));
(1,9)(3,8)(4,7)(5,6)
```

After making the identifications

$$\alpha \equiv (1\,2)(3\,5)(4\,8)(6\,7), \quad \beta \equiv (2\,3\,4\,5\,6\,7\,8), \quad \gamma \equiv (1\,9)(3\,8)(4\,7)(5\,6),$$

we conclude that $\langle \alpha, \beta, \gamma \rangle \simeq \mathbf{PSL}_2(8)$:

```
gap> Q := Group([a, b, c]);;
gap> StructureDescription(Q);
"PSL(2,8)"
```

One can use the function `ActionHomomorphism` to construct the natural homomorphism induced by an action:

```
gap> S3 := SymmetricGroup(3);;
gap> f := ActionHomomorphism(S3, [1..3], OnPoints);;
gap> Image(f) = Action(S3, [1..3], OnPoints);
true
```

With this function, we will be able to construct all sorts of actions. We will delve into this in the next section.

2.3 Homomorphisms

We have several ways to construct group homomorphisms. For instance, the function `GroupHomomorphismByImages` returns the group homomorphism constructed by specifying the values of the map on a given set of generators of the domain. If these values are not consistent (in the sense that no such homomorphism can exist), then the function will return `fail`.

Properties of group homomorphisms can be studied with `Image`, `IsInjective`, `IsSurjective`, `Kernel`, `PreImage`, `PreImages`, etc.

Example 2.38. The map $\mathbb{S}_4 \longrightarrow \mathbb{S}_3$ that maps each transposition of $\mathbb{S}_4$ into $(1\,2)$ extends to a group homomorphism f. This homomorphism f is not injective ($\ker f$ has twelve elements) and it is not surjective (for example $(1\,2\,3) \notin f(\mathbb{S}_4)$):

```
gap> S4 := SymmetricGroup(4);;
gap> S3 := SymmetricGroup(3);;
gap> f := GroupHomomorphismByImages(S4, S3,\
> [(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)],\
> [(1,2),(1,2),(1,2),(1,2),(1,2),(1,2)]);
[ (1,2), (1,3), (1,4), (2,3), (2,4), (3,4) ] ->
[ (1,2), (1,2), (1,2), (1,2), (1,2), (1,2) ]
gap> Size(Kernel(f));
12
gap> IsInjective(f);
false
gap> Size(Image(f));
2
gap> IsSurjective(f);
false
gap> (1,2,3) in Image(f);
false
gap> (1,3)^f;
(1,2)
gap> (1,2,3)^f;
()
gap> (1,2,3,4)^f;
(1,2)
```

If K is a normal subgroup of G, the canonical map $G \longrightarrow G/K$ can be constructed with the function `NaturalHomomorphismByNormalSubgroup`.

Example 2.39. Let us construct the cyclic group C_{12} with generator g as a group of permutations, the subgroup $K = \langle g^6 \rangle$ and the quotient C_{12}/K. We also construct the canonical (surjective) map $C_{12} \longrightarrow C_{12}/K$:

```
gap> g := (1,2,3,4,5,6,7,8,9,10,11,12);;
gap> C12 := Group(g);;
gap> K := Subgroup(C12, [g^6]);;
gap> f := NaturalHomomorphismByNormalSubgroup(C12, K);
[ (1,2,3,4,5,6,7,8,9,10,11,12) ] -> [ f1 ]
gap> Image(f, g^6);
<identity> of ...
gap> (g^6)^f = Image(f, g^6);
true
```

Example 2.40. With `GQuotients` we can determine when a given group is an epimorphic image of another one. For example, $\mathbb{S}_3$ is an epimorphic image of $\mathbb{S}_4$, but $C_2 \times C_2$ is not:

```
gap> S4 := SymmetricGroup(4);;
gap> GQuotients(S4, SymmetricGroup(3));
[ [ (2,4), (1,2,3) ] -> [ (2,3), (1,2,3) ] ]
gap> GQuotients(S4, AbelianGroup([2, 2]));
[  ]
```

Example 2.41. The group $\mathbb{S}_3$ is an epimorphic image of $\mathbb{S}_4$. In fact, if

$$K = \{\mathrm{id}, (1\,2)(3\,4), (1\,3)(2\,4), (1\,4)(2\,3)\},$$

then K is a normal subgroup of $\mathbb{S}_4$ and $\mathbb{S}_4/K \simeq \mathbb{S}_3$. Let us construct the canonical map $\mathbb{S}_4 \to \mathbb{S}_4/K$:

```
gap> S4 := SymmetricGroup(4);;
gap> K := Subgroup(S4, [(1,2)(3,4), (1,3)(2,4)]);;
gap> p := NaturalHomomorphismByNormalSubgroup(S4, K);;
gap> Elements(Kernel(p));
[ (), (1,2)(3,4), (1,3)(2,4), (1,4)(2,3) ]
gap> IsSurjective(p);
true
```

The function `AutomorphismGroup` computes the automorphism group of a finite group. If G is a group, the automorphisms of G of the form $x \mapsto g^{-1}xg$, where $g \in G$, are the *inner automorphisms* of G. The function `IsInnerAutomorphism` checks whether a given automorphism is inner.

Example 2.42. Let us check that $\mathrm{Aut}(\mathbb{S}_3)$ is a non-abelian group of six elements:

```
gap> aut := AutomorphismGroup(SymmetricGroup(3));
<group of size 6 with 2 generators>
gap> IsAbelian(aut);
false
```

Example 2.43. Let us prove that for $n \in \{2, 3, 4, 5\}$ each automorphism of $\mathbb{S}_n$ is inner. Here is the code that shows our claim:

```
gap> for n in [2..5] do
> G := SymmetricGroup(n);;
> if ForAll(AutomorphismGroup(G), IsInnerAutomorphism) then
>   Print("Each automorphism of S", n, " is inner.\n");
> fi;
> od;
Each automorphism of S2 is inner.
Each automorphism of S3 is inner.
Each automorphism of S4 is inner.
Each automorphism of S5 is inner.
```

It is known that in $\mathbb{S}_6$ there are non-inner automorphisms:

```
gap> S6 := SymmetricGroup(6);;
gap> f := First(AutomorphismGroup(S6),\
> x->not IsInnerAutomorphism(x));
[ (1,2,3,4,5,6), (1,2) ] -> [ (2,3)(4,6,5), (1,2)(3,5)(4,6) ]
```

The automorphism of $\mathbb{S}_6$ such that

$$(1\,2\,3\,4\,5\,6) \mapsto (2\,3)(4\,6\,5) \quad \text{and} \quad (1\,2) \mapsto (1\,2)(3\,5)(4\,6)$$

is not inner. Let us compute the values of this homomorphism in the transpositions:

```
gap> for t in ConjugacyClass(S6, (1,2)) do
> Print("f(", t, ")=", Image(f,t), "\n");
> od;
f((1,2))=(1,2)(3,5)(4,6)
f((1,3))=(1,6)(2,5)(3,4)
f((1,4))=(1,4)(2,3)(5,6)
f((1,5))=(1,5)(2,4)(3,6)
f((1,6))=(1,3)(2,6)(4,5)
f((2,3))=(1,3)(2,4)(5,6)
f((2,4))=(1,5)(2,6)(3,4)
f((2,5))=(1,6)(2,3)(4,5)
f((2,6))=(1,4)(2,5)(3,6)
f((3,4))=(1,2)(3,6)(4,5)
f((3,5))=(1,4)(2,6)(3,5)
f((3,6))=(1,5)(2,3)(4,6)
f((4,5))=(1,3)(2,5)(4,6)
f((4,6))=(1,6)(2,4)(3,5)
f((5,6))=(1,2)(3,4)(5,6)
```

With `InnerAutomorphismsAutomorphismGroup`, one constructs the inner automorphism group of a given group.

Example 2.44. Let us check that $\mathrm{Aut}(\mathbb{S}_6)/\mathrm{Inn}(\mathbb{S}_6) \simeq C_2$:

```
gap> S6 := SymmetricGroup(6);;
gap> A := AutomorphismGroup(S6);;
gap> Size(A);
1440
gap> I := InnerAutomorphismsAutomorphismGroup(A);;
gap> Order(A/I);
2
```

The following conjecture, proposed by O. Schreier in 1926, is now a theorem thanks to the classification of finite simple groups.

Conjecture 2.1 (Schreier). If G is a finite simple group, then the outer automorphism group $\mathrm{Out}(G) = \mathrm{Aut}(G)/\mathrm{Inn}(G)$ is solvable.

So far, there is no known proof of the Schreier conjecture that is independent of the classification of simple groups.

Example 2.45 (Testing Schreier's conjecture). We first write a function to compute the outer automorphism group:

```
gap> OuterAutomorphismGroup := function(G)
> local aut;
> aut := AutomorphismGroup(G);
> return aut/InnerAutomorphismsAutomorphismGroup(aut);
> end;
function( G ) ... end
```

We write a function that checks if the conjecture is true (recall that the conjecture is about outer automorphisms of simple groups):

```
gap> Schreier := function(G)
> if not IsSimple(G) then
>   return fail;
> fi;
> return IsSolvable(OuterAutomorphismGroup(G));
> end;
function( G ) ... end
```

Now we check the conjecture in some small examples:

```
gap> Schreier(MathieuGroup(11));
true
gap> Schreier(MathieuGroup(12));
true
gap> Schreier(AlternatingGroup(5));
true
gap> Schreier(PSL(2,7));
true
```

Finally, with `AllHomomorphisms`, one constructs the set of all group homomorphisms between two given groups. Similarly, one uses `AllEndomorphisms` to compute the set of endomorphisms of a group.

Example 2.46. We prove that there are ten endomorphisms of $\mathbb{S}_3$:

```
gap> S3 := SymmetricGroup(3);;
gap> Size(AllEndomorphisms(S3));
10
```

2.4 Semidirect products

We begin this section by using the function `DirectProduct` to construct direct products of groups; we can obtain the associated canonical maps with `Embedding` and `Projection`.

Example 2.47. We construct the direct product $\mathbb{S}_3 \times \mathbb{D}_8$ and show that the canonical embedding $\mathbb{S}_3 \to \mathbb{S}_3 \times \mathbb{D}_8$, $x \mapsto (x, 1)$, is injective but not surjective, and the natural projection $\mathbb{S}_3 \times \mathbb{D}_8 \to \mathbb{S}_3$, $(x, y) \mapsto x$, is surjective but not injective:

```
gap> S3 := SymmetricGroup(3);;
gap> D8 := DihedralGroup(8);;
gap> S3xD8 := DirectProduct(S3, D8);;
gap> i_S3 := Embedding(S3xD8, 1);;
gap> p_S3 := Projection(S3xD8, 1);;
gap> IsInjective(i_S3);
true
gap> IsSurjective(i_S3);
false
gap> IsInjective(p_S3);
false
gap> IsSurjective(p_S3);
true
```

Example 2.48. Let us check that $C_4 \times C_4$ and $C_2 \times Q_8$ have three elements of order two and twelve elements of order four:

```
gap> C4 := CyclicGroup(IsPermGroup, 4);;
gap> C2 := CyclicGroup(IsPermGroup, 2);;
gap> Q8 := QuaternionGroup(8);;
gap> C4xC4 := DirectProduct(C4, C4);;
gap> C2xQ8 := DirectProduct(C2, Q8);;
gap> Collected(List(C4xC4, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 4, 12 ] ]
gap> Collected(List(C2xQ8, Order));
[ [ 1, 1 ], [ 2, 3 ], [ 4, 12 ] ]
```

The groups $C_4 \times C_4$ and $C_2 \times Q_8$ are not isomorphic. An easy way to see this is to notice that $C_4 \times C_4$ is abelian and $C_2 \times Q_8$ is not:

```
gap> IsAbelian(C4xC4);
true
gap> IsAbelian(C2xQ8);
false
```

Example 2.49. The center of $C_2 \times \mathbb{S}_3$ is not stable under endomorphisms of $C_2 \times \mathbb{S}_3$. We see that

$$Z(C_2 \times \mathbb{S}_3) = \{\mathrm{id}, (1\,2)\}$$

and that there exists at least one endomorphism of $C_2 \times \mathbb{S}_3$ that permutes the non-trivial element of the center:

```
gap> C2 := CyclicGroup(IsPermGroup, 2);;
gap> S3 := SymmetricGroup(3);;
gap> C2xS3 := DirectProduct(C2, S3);;
gap> center := Center(C2xS3);
Group([ (1,2) ])
gap> ForAll(AllEndomorphisms(C2xS3),\
> f->Image(f, (1,2)) in center);
false
```

With `DirectFactorsOfGroup` we can get the decomposition of a given group as a direct product:

```
gap> D12 := DihedralGroup(12);;
gap> List(DirectFactorsOfGroup(D12), StructureDescription);
[ "S3", "C2" ]
gap> List(DirectFactorsOfGroup(GL(2,4)), StructureDescription);
[ "C3", "A5" ]
```

Now we use the command `SemidirectProduct(H, phi, K)` to construct the semidirect product $K \rtimes_{\phi} H$, where $\phi\colon H \to \operatorname{Aut}(K)$ is a group homomorphism.

Example 2.50. Let $\langle g \rangle \simeq C_2$ be the cyclic group of order two. We construct the semidirect product $C_7 \rtimes_{\phi} C_2$, where $\phi\colon C_2 \to \operatorname{Aut}(C_7)$, $g \mapsto (x \mapsto x^{-1})$, and show that $C_7 \rtimes_{\phi} C_2 \simeq \mathbb{D}_{14}$:

```
gap> C7 := CyclicGroup(7);;
gap> C2 := CyclicGroup(2);;
gap> f := GroupHomomorphismByFunction(C7, C7, x->Inverse(x));;
gap> f in AutomorphismGroup(C7);
true
gap> Order(f);
2
gap> # Take a generator of C2
gap> g := C2.1;;
gap> phi := GroupHomomorphismByImages(C2, \
> AutomorphismGroup(C7), [g], [f]);;
gap> G := SemidirectProduct(C2, phi, C7);;
gap> StructureDescription(G);
"D14"
```

We construct the canonical inclusion maps $C_2 \to C_7 \rtimes_{\phi} C_2$, $C_7 \to C_7 \rtimes_{\phi} C_2$ and the surjective map $C_7 \rtimes_{\phi} C_2 \to C_2$:

```
gap> i_C2 := Embedding(G, 1);;
gap> StructureDescription(Image(i_C2));
"C2"
gap> i_C7 := Embedding(G, 2);;
gap> StructureDescription(Image(i_C7));
"C7"
gap> p := Projection(G);;
gap> StructureDescription(Image(p));
"C2"
```

Example 2.51. Let us construct all possible semidirect products (up to isomorphism) of the form $C_4^2 \rtimes C_2$:

```
gap> C2 := CyclicGroup(2);;
gap> C4xC4 := AbelianGroup([4, 4]);;
gap> hom := AllHomomorphisms(C2, AutomorphismGroup(C4xC4));;
gap> list := [];;
gap> for phi in hom do
> Add(list, SemidirectProduct(C2, phi, C4xC4));
> od;
gap> Set(list, IdGroup);
[ [ 32, 11 ], [ 32, 21 ], [ 32, 24 ],
  [ 32, 25 ], [ 32, 31 ], [ 32, 33 ],
  [ 32, 34 ] ]
```

Example 2.52. Let

$$Q = \left\langle \begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 3 \\ 7 & 10 \end{pmatrix} \right\rangle \subseteq \mathbf{SL}_2(11).$$

We construct the semidirect product $G = \mathbb{F}_{11}^2 \rtimes Q$ with the natural action:

```
gap> a := [[4, 1], [0, 3]]*One(GF(11));;
gap> b := [[0, 3], [7, 10]]*One(GF(11));;
gap> Q := Group([a, b]);;
gap> StructureDescription(Q);
"SL(2,5)"
gap> G := SemidirectProduct(Q, GF(11)^2);
<matrix group of size 14520 with 3 generators>
gap> StructureDescription(G);
"(C11 x C11) : SL(2,5)"
```

We now prove that G is a Frobenius group. We need, for example, to check that the centralizers $C_Q(k)$ are trivial for all $k \in K \setminus \{1\}$:

```
gap> i_K := Embedding(G, 2);;
gap> i_Q := Embedding(G, 1);;
gap> K := Image(i_K);;
gap> ForAll(K, k->IsOne(k) or \
> IsTrivial(Centralizer(Image(i_Q), k)));
true
```

Example 2.53. Let us perform some calculations with the canonical homomorphism $\mathbf{GL}_2(\mathbb{Z}/4) \to \mathbf{GL}_2(\mathbb{Z}/2)$. We first define the groups:

```
gap> gl4 := GL(2, Integers mod 4);;
gap> gl2 := GL(2, Integers mod 2);;
gap> Order(gl4);
96
gap> Order(gl2);
6
```

We check that the group $\mathbf{GL}_2(\mathbb{Z}/4)$ is generated by $\left\{ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \right\}$. Here is the code that proves our claim:

```
gap> gens := GeneratorsOfGroup(gl4);;
gap> for x in gens do
```

```
> Display(x);
> od;
matrix over Integers mod 4:
[ [  0,  1 ],
  [  1,  0 ] ]
matrix over Integers mod 4:
[ [  1,  1 ],
  [  0,  1 ] ]
matrix over Integers mod 4:
[ [  3,  0 ],
  [  0,  1 ] ]
```

We construct the homomorphism from $\mathbf{GL}_2(\mathbb{Z}/4)$ onto $\mathbf{GL}_2(\mathbb{Z}/2)$ given by

$$\begin{pmatrix}0&1\\1&0\end{pmatrix} \mapsto \begin{pmatrix}0&1\\1&0\end{pmatrix}, \quad \begin{pmatrix}1&1\\0&1\end{pmatrix} \mapsto \begin{pmatrix}1&1\\0&1\end{pmatrix}, \quad \begin{pmatrix}3&0\\0&1\end{pmatrix} \mapsto \mathrm{id}\,.$$

```
gap> a := [[0, 1], [1, 0]]*One(gl2);;
gap> b := [[1, 1], [0, 1]]*One(gl2);;
gap> f := GroupHomomorphismByImages(gl4, gl2, gens, \
> [a, b, One(gl2)]);;
gap> IsSurjective(f);
true
gap> Size(Kernel(f));
16
```

A particular type of group homomorphism is given by group actions.

Example 2.54. Let us construct the action of $C_2 = \langle g \rangle$ on abelian groups by inversion. We first need to construct the function corresponding to the action:

```
gap> inversion := function(a, g)
> if IsOne(g) then
>   return a;
> else
>   return Inverse(a);
> fi;
> end;
function( a, g ) ... end
```

Now we perform concrete calculations on the abelian group $A = C_2 \times C_4$:

```
gap> C2 := CyclicGroup(IsPermGroup, 2);;
gap> A := AbelianGroup(IsPermGroup, [2, 4]);
Group([ (1,2), (3,4,5,6) ])
gap> Orbits(C2, A, inversion);
[ [ () ], [ (3,4,5,6), (3,6,5,4) ], [ (3,5)(4,6) ], [ (1,2) ],
  [ (1,2)(3,4,5,6), (1,2)(3,6,5,4) ], [ (1,2)(3,5)(4,6) ] ]
gap> rho := ActionHomomorphism(C2, A, inversion);;
gap> Image(rho);
Group([ (2,4)(6,8) ])
gap> Kernel(rho);
Group(())
```

```
gap> # By definition, the range is the codomain of the map
gap> Range(rho);
Sym( [ 1 .. 8 ] )
```

In this case, it is an action by group automorphisms. To construct such homomorphism, we could use `GroupHomomorphismByFunction`. However, this function does not check that the resulting map is a group homomorphism. For that reason, we rather prefer to use `GroupHomomorphismByImages`:

```
gap> s := AsSet(A);;
gap> f := GroupHomomorphismByImages(A, A, s, List(s, Inverse));;
gap> IsInjective(f) and IsSurjective(f);
true
```

If we replace A by a non-abelian group, for example, $\mathbb{S}_3$, we have an action of C_2 that is not an action by group automorphisms:

```
gap> S3 := SymmetricGroup(3);;
gap> s := AsSet(S3);;
gap> f := GroupHomomorphismByImages(S3, S3, s, List(s, Inverse));
fail
```

Note that the assignment $(1\,2\,3) \mapsto (1\,2\,3)^{-1} = (1\,3\,2)$, $(1\,2) \mapsto (1\,2)^{-1} = (1\,2)$, does extend to a unique endomorphism of $\mathbb{S}_3$. That is, it inverts this set of generators of $\mathbb{S}_3$ but is not the inversion map $x \mapsto x^{-1}$:

```
gap> gens := [(1,2,3), (1,2)];;
gap> f := GroupHomomorphismByImages(S3, S3, gens, \
> List(gens, Inverse));
[ (1,2,3), (1,2) ] -> [ (1,3,2), (1,2) ]
gap> for x in S3 do
> Print(x, "->", x^f, "\n");
> od;
()->()
(2,3)->(1,3)
(1,3)->(2,3)
(1,3,2)->(1,2,3)
(1,2,3)->(1,3,2)
(1,2)->(1,2)
```

Example 2.55. Let

$$\gamma\colon \mathbf{GL}_n(q) \to \mathbf{GL}_n(q), \quad \gamma(x) = (x^T)^{-1},$$

be the map that takes a matrix to its transpose-inverse. Clearly, γ is an automorphism of $\mathbf{GL}_n(q)$ such that $\gamma^2 = \mathrm{id}$. Moreover, γ leaves $\mathbf{SL}_n(q)$ and the center $Z(\mathbf{GL}_n(q))$ invariant; in particular γ induces automorphisms on $\mathbf{PSL}_n(q)$ and $\mathbf{PGL}_n(q)$. We implement the map γ and its action on invertible matrices:

```
gap> gammaFunction := x->TransposedMat(Inverse(x));;
gap> gammaAction := function(x, g)
> if IsOne(g) then
>   return x;
> else
>   return gammaFunction(x);
> fi;
> end;
function( x, g ) ... end
```

We consider the action of the cyclic group $\langle\gamma\rangle$ of order 2 on $G = \mathbf{GL}_3(3)$:

```
gap> G := GL(3,3);;
gap> Size(G);
11232
gap> s := AsSet(G);;
gap> gamma := GroupHomomorphismByFunction(G, G, gammaFunction);;
gap> Order(gamma);
2
gap> gamma in AutomorphismGroup(G);
true
```

We now compute the set $\{x \in G : x^\gamma = x\}$ of fixed points. Then we quickly obtain the number of orbits:

```
gap> fix := Group(Filtered(s, x->x^gamma = x));;
gap> StructureDescription(fix);
"C2 x S4"
gap> # This is Burnside's lemma
gap> (Size(G) + Size(fix))/2;
5640
```

Indeed, the subgroup of fixed points is the group of orthogonal matrices over the field of three elements:

```
gap> IsomorphismGroups(GO(3,3), fix) <> fail;
true
```

Since γ acts on G by group automorphisms, we now construct the semidirect product $S = G \rtimes \langle\gamma\rangle$; we use `DisplayCompositionSeries` to display the composition factors:

```
gap> C2 := Subgroup(AutomorphismGroup(G), [gamma]);;
gap> S := SemidirectProduct(C2, G);;
gap> Order(S);
22464
gap> DisplayCompositionSeries(S);
G (3 gens, size 22464)
 | C2
S (5 gens, size 11232)
 | L3(3)
S (1 gens, size 2)
 | C2
1 (0 gens, size 1)
```

Note that `DisplayCompositionSeries` prints `G` for our semidirect product (of size 22464) and `S` for the intermediate subgroups.

We now verify that the set of fixed points is equal to the centralizer $C_G(\gamma)$:

```
gap> i_C2 := Embedding(S, 1);;
gap> i_G := Embedding(S, 2);;
gap> C := Centralizer(Image(i_G), Image(i_C2));;
gap> Image(i_G, fix) = C;
true
gap> StructureDescription(C);
"C2 x S4"
```

Example 2.56. We continue with Example 2.55. Let $S = \mathbf{GL}_3(3) \rtimes \langle\gamma\rangle$ be the semidirect product of the linear group $\mathbf{GL}_3(3)$ by the transpose-inverse map γ. We construct the quotient $Q = S/Z(\mathbf{GL}_3(3))$:

```
gap> Q := S/Center(Image(i_G));;
gap> StructureDescription(Q);
"PSL(3,3) : C2"
```

Note that $\mathbf{GL}_3(3)$ and Q have the same composition factors but they are not isomorphic since, for example, Q has trivial center and $Z(\mathbf{GL}_3(3)) = C_2$:

```
gap> DisplayCompositionSeries(Q);
G (3 gens, size 11232)
 | C2
S (12 gens, size 5616)
 | L3(3)
1 (0 gens, size 1)
gap> DisplayCompositionSeries(GL(3,3));
G (size 11232)
 | L3(3)
S (1 gens, size 2)
 | C2
1 (size 1)
gap> IsTrivial(Center(Q));
true
gap> StructureDescription(Center(GL(3,3)));
"C2"
```

Example 2.57. We now repeat what we did in Examples 2.55 and 2.56 with the group $G = \mathbf{GL}_2(7)$:

```
gap> G := GL(2,7);;
gap> s := AsSet(G);;
gap> gamma := GroupHomomorphismByFunction(G, G, gammaFunction);;
gap> C2 := Subgroup(AutomorphismGroup(G), [gamma]);;
gap> Size(Orbits(C2, G, gammaAction));
1016
gap> S := SemidirectProduct(C2, G);;
gap> StructureDescription(S);
"GL(2,7) : C2
```

We check the structure of the centralizer $C_G(\gamma)$:

```
gap> i_C2 := Embedding(S, 1);;
gap> i_G := Embedding(S, 2);;
gap> C := Centralizer(Image(i_G), Image(i_C2));
<permutation group with 2 generators>
gap> StructureDescription(C);
"D16"
```

In this case, $S/Z(G) \simeq \mathbf{PGL}_2(7) \rtimes C_2$ is indeed isomorphic to $\mathbf{PGL}_2(7) \times C_2$ since the induced action of γ on $\mathbf{PGL}_2(7)$ is inner:

$$\gamma \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{1}{ad-bc} \begin{pmatrix} d & -c \\ -b & a \end{pmatrix} = \frac{1}{ad-bc} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Here is the code:

```
gap> N := Center(Image(i_G));;
gap> Q := S/N;;
gap> DirectFactorsOfGroup(Q);
[ PSL(3,2) : C2, C2 ]
gap> not IsomorphismGroups(Q, DirectProduct(PGL(2,7),C2)) = fail;
true
gap> not IsomorphismGroups(Q/Center(Q), PGL(2,7)) = fail;
true
```

Example 2.58. Let us see how the alternating group $\mathbb{A}_5$ acts on a coset space by right multiplication. First, we compute the list of conjugacy classes of subgroups of $\mathbb{A}_5$. There are nine conjugacy classes of subgroups:

```
gap> A5 := AlternatingGroup(5);;
gap> classes := ConjugacyClassesSubgroups(A5);;
gap> Size(classes);
9
```

For example, the fourth element in our list is the conjugacy class of subgroups with representative $\langle (2\,3)(4\,5), (2\,4)(3\,5) \rangle$. The conjugacy class has five subgroups:

```
gap> class := classes[4];
Group( [ (2,3)(4,5), (2,4)(3,5) ] )^G
gap> Size(class);
5
gap> for x in class do
> Display(x);
> od;
Group( [ (2,3)(4,5), (2,4)(3,5) ] )
Group( [ (1,2)(4,5), (1,4)(2,5) ] )
Group( [ (1,2)(3,4), (1,3)(2,4) ] )
Group( [ (1,5)(2,3), (1,3)(2,5) ] )
Group( [ (1,5)(3,4), (1,4)(3,5) ] )
```

We can obtain some information on representatives of conjugacy classes of subgroups of $\mathbb{A}_5$:

```
gap> List(classes, x->Order(Representative(x)));
[ 1, 2, 3, 4, 5, 6, 10, 12, 60 ]
gap> List(classes, x->Index(A5, Representative(x)));
[ 60, 30, 20, 15, 12, 10, 6, 5, 1 ]
gap> List(classes, x->StructureDescription(Representative(x)));
[ "1", "C2", "C3", "C2 x C2", "C5", "S3", "D10", "A4", "A5" ]
```

Let H be the subgroup of $\mathbb{A}_5$ isomorphic to the cyclic group C_5 of order five. We now construct the action of $\mathbb{A}_5$ on the set $\mathbb{A}_5/H$ by right multiplication:

```
gap> H := Representative(classes[5]);;
gap> Elements(H);
[ (), (1,2,3,4,5), (1,3,5,2,4), (1,4,2,5,3), (1,5,4,3,2) ]
gap> f := ActionHomomorphism(A5, RightCosets(A5, H), OnRight);;
gap> Kernel(f);
1
gap> IsInjective(f);
true
gap> IsSurjective(f);
false
```

2.5 Solvable groups

We now describe some algorithms and concepts related to solvable groups.

We know that `AllSubgroups` returns the list of all subgroups of a given group, but it is intended primarily for small examples, and may become inefficient for larger groups. In general, it is better to use `ConjugacyClassesSubgroups`. In the case of solvable groups, there are better algorithms to compute conjugacy classes of subgroups. The function `SubgroupsSolvableGroup` returns a list of representatives of subgroups up to conjugation.

Example 2.59. Let P be the Sylow 2-subgroup of the Mathieu group M_{24}. We want to compute the conjugacy classes of subgroups of P of order 2^5. We can proceed as follows:

```
gap> P := SylowSubgroup(MathieuGroup(24), 2);;
gap> Number(ConjugacyClassesSubgroups(P), \
> x->Order(Representative(x)) = 2^5);
1403
```

Thus there are 1403 conjugacy classes of subgroups of order 2^5.

Since P is solvable, we can try something different. If we use the function `SubgroupsSolvableGroup` in combination with `ExactSizeConsiderFunction` the calculation is about much faster (the reader should compare the different methods):

```
gap> Size(SubgroupsSolvableGroup(P, \
> rec(consider := ExactSizeConsiderFunction(2^5))));
1728
```

The numbers are different! This happens because `SubgroupsSolvableGroup` returned a list of subgroups of P that includes all subgroups of order 2^5 and (maybe) other subgroups of P. To obtain the right number of subgroups we need to remove the unwanted subgroups (this is super fast, no extra time is needed):

```
gap> Number(SubgroupsSolvableGroup(P, rec(consider := \
> ExactSizeConsiderFunction(2^5))) ,x->Order(x) = 2^5);
1403
```

This does not mean that it is always more convenient to use this function when dealing with solvable groups. For example, if we want to compute the list of normal subgroups, `NormalSubgroups` is much faster than `SubgroupsSolvableGroup` (again, the reader should compare the run-times of these methods):

```
gap> Size(NormalSubgroups(P));
157
gap> Number(SubgroupsSolvableGroup(P, \
> rec(normal := true)), x->IsNormal(P, x));
157
```

Note that, again, `SubgroupsSolvableGroup` produced a list of subgroups that includes all normal subgroups of P and (maybe) some other non-normal subgroups of P. This means that an extra filter is needed.

Hall's theorem states that if G is a finite solvable group of order ab with $\gcd(a, b) = 1$, then there exists a unique (non-empty) conjugacy class of subgroups of G order a. Such subgroups of order a are called Hall π-subgroups of G, where π is the set of prime divisors of a.

Example 2.60. We now perform some calculations with the function `HallSubgroup`:

```
gap> D60 := DihedralGroup(IsPermGroup, 60);;
gap> IsSolvable(D60);
true
gap> StructureDescription(HallSubgroup(D60, [2, 3]));
"D12"
gap> StructureDescription(HallSubgroup(D60, [2, 5]));
"D20"
gap> StructureDescription(HallSubgroup(D60, [3, 5]));
"C15"
```

Now we show that the alternating group $\mathbb{A}_5$, a simple group of order 60, does not contain Hall subgroups of order 15 and 20:

```
gap> A5 := AlternatingGroup(5);;
gap> HallSubgroup(A5, [3, 5]);
fail
gap> HallSubgroup(A5, [2, 5]);
fail
```

Hall $\{2,3\}$-subgroups of the simple group $\mathbf{PSL}_2(7)$ of order 168 are not conjugate (but isomorphic):

```
gap> G := PSL(2,7);;
gap> hall := HallSubgroup(G, [2, 3]);;
gap> Size(hall);
2
gap> IsConjugate(G, hall[1], hall[2]);
false
gap> not IsomorphismGroups(hall[1], hall[2]) = fail;
true
```

We finally show that Hall $\{2,3\}$-subgroups of the simple group $\mathbf{PSL}_2(11)$ of order 660 are not isomorphic (and hence not conjugate):

```
gap> G := PSL(2,11);;
gap> Order(PSL(2,11));
660
gap> hall := HallSubgroup(G, [2, 3]);;
gap> Size(hall);
2
gap> List(hall, StructureDescription);
[ "D12", "A4" ]
gap> IsomorphismGroups(hall[1], hall[2]);
fail
```

Example 2.61. Schottenfels' theorem [48, Theorem 8.24] states that $\mathbf{PSL}_3(4)$ and $\mathbb{A}_8$ are simple groups of the same order, but they are not isomorphic:

```
gap> G := PSL(3,4);;
gap> Order(G);
20160
gap> A8 := AlternatingGroup(8);;
gap> Order(A8);
20160
gap> IsomorphismGroups(G, A8);
fail
```

The fact that these groups are not isomorphic also follows from the structure of Hall's subgroups:

```
gap> HallSubgroup(G, [2, 3]);
fail
gap> StructureDescription(HallSubgroup(A8, [2, 3]));
"(C2 x C2 x C2 x C2) : (S3 x S3)"
```

2.6 Finitely presented groups

Let us start working with free groups. The function `FreeGroup` constructs the free group in a finite number of generators.

Example 2.62. We create the free group F_2 in two generators and we create some random elements with the function `Random`:

```
gap> f := FreeGroup(2);
<free group on the generators [ f1, f2 ]>
gap> f.1^2;
f1^2
gap> f.1^2*f.1;
f1^3
gap> f.1*f.1^(-1);
<identity ...>
gap> Random(f);
f1^-3
```

Example 2.63. The function `Length` can be used to compute the length of a word in a free group. In this example, we create 10000 random elements in F_2 and compute their lengths.

```
gap> f := FreeGroup(2);;
gap> Collected(List(List([1..10000], x->Random(f)), Length));
[ [ 0, 2270 ], [ 1, 1044 ], [ 2, 1113 ],
  [ 3, 986 ], [ 4, 874 ], [ 5, 737 ],
  [ 6, 642 ], [ 7, 500 ], [ 8, 432 ],
  [ 9, 329 ], [ 10, 248 ], [ 11, 189 ],
  [ 12, 152 ], [ 13, 119 ], [ 14, 93 ],
  [ 15, 68 ], [ 16, 57 ], [ 17, 34 ],
  [ 18, 30 ], [ 19, 23 ], [ 20, 19 ],
  [ 21, 16 ], [ 22, 8 ], [ 23, 3 ], [ 24, 4 ],
  [ 25, 4 ], [ 26, 2 ], [ 27, 2 ], [ 28, 1 ],
  [ 31, 1 ] ]
```

Some of the functions we have used before can also be used in free groups. Examples of these functions are `Normalizer`, `RepresentativeAction`, `IsConjugate`, `Intersection`, `IsSubgroup`, `Subgroup`.

Example 2.64. Here we perform some elementary calculations in F_2, the free group with generators a and b. We also compute the automorphism group of F_2.

```
gap> f := FreeGroup("a", "b");;
```

We now need to assign the generators to variables. We could do, for example, the following sequence of commands:

```
gap> a := GeneratorsOfGroup(f)[1];;
gap> b := GeneratorsOfGroup(f)[2];;
```

However, there is an easier way to do this:

```
gap> AssignGeneratorVariables(f);
#I  Assigned the global variables [ a, b ]
```

Now we perform some calculations in the free group:

```
gap> Random(f);
b^-1*a^-5
gap> Centralizer(f, a);
Group([ a ])
gap> Index(f, Centralizer(f, a));
infinity
gap> Subgroup(f, [a, b]);
Group([ a, b ])
gap> Order(Subgroup(f, [a, b]));
infinity
gap> AutomorphismGroup(f);
<group of size infinity with 3 generators>
gap> GeneratorsOfGroup(AutomorphismGroup(f));
[ [ a, b ] -> [ a^-1, b ],
  [ a, b ] -> [ b, a ],
  [ a, b ] -> [ a*b, b ] ]
```

We now check that the subgroup S generated by a^2, b and aba^{-1} has index two in F_2. We compute $\mathrm{Aut}(S)$ and check that this is not a free group:

```
gap> S := Subgroup(f, [a^2, b, a*b*a^(-1)]);
Group([ a^2, b, a*b*a^-1 ])
gap> Index(f, S);
2
gap> A := AutomorphismGroup(S);
<group of size infinity with 3 generators>
gap> IsFreeGroup(A);
false
```

Example 2.65. Let $n \geq 3$ and $p \geq 2$ be integers. An astonishing result of Coxeter [12] states that the group with generators $\sigma_1, \dots, \sigma_{n-1}$ and relations

$$\begin{aligned} &\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} && \text{if } i \in \{1, \dots, n-2\},\\ &\sigma_i\sigma_j = \sigma_j\sigma_i && \text{if } |i-j| \geq 2,\\ &\sigma_i^p = 1 && \text{if } i \in \{1, \dots, n-1\}\rangle, \end{aligned}$$

is finite if and only if $(p-2)(n-2) < 4$.

We study the case $n = 3$. Let

$$G = \langle a, b \mid aba = bab,\ a^p = b^p = 1\rangle.$$

We claim that

$$G \simeq \begin{cases} \mathbb{S}_3 & \text{if } p = 2,\\ \mathbf{SL}_2(3) & \text{if } p = 3,\\ \mathbf{SL}_2(3) \rtimes C_4 & \text{if } p = 4,\\ \mathbf{SL}_2(3) \times C_5 & \text{if } p = 5. \end{cases}$$

We use the function `ParseRelators` to translate the list of relations (in a human-readable form) into relators. Here is the proof of the above claim:

```
gap> f := FreeGroup("a", "b");;
gap> AssignGeneratorVariables(f);
#I  Assigned the global variables [ a, b ]
gap> p := 2;;
gap> while p-2 < 4 do
> G := f/ParseRelators(f, "a*b*a=b*a*b, a^p=1, b^p=1");
> Display(StructureDescription(G));
> p := p+1;
> od;
S3
SL(2,3)
SL(2,3) : C4
C5 x SL(2,5)
```

Example 2.66. For positive integers l, m, n, we define the *von Dyck group* (or triangular group) of type (l, m, n) as the group

$$G(l,m,n) = \langle a, b \mid a^l = b^m = (ab)^n = 1 \rangle.$$

It is known that $G(l, m, n)$ is finite if and only if

$$\frac{1}{l} + \frac{1}{m} + \frac{1}{n} > 1.$$

We claim that

$$G(2,3,3) \simeq \mathbb{A}_4, \quad G(2,3,4) \simeq \mathbb{S}_4, \quad G(2,3,5) \simeq \mathbb{A}_5.$$

We construct these groups using relators, and not the function `ParseRelators` as we did in the previous example:

```
gap> f := FreeGroup("a", "b");;
gap> AssignGeneratorVariables(f);;
#I  Assigned the global variables [ a, b ]
gap> StructureDescription(f/[a^2, b^3, (a*b)^3]);
"A4"
gap> StructureDescription(f/[a^2, b^3, (a*b)^4]);
"S4"
gap> StructureDescription(f/[a^2, b^3, (a*b)^5]);
"A5"
```

Example 2.67. This example is taken from [14]. Let us check that the group

$$\langle a, b, c \mid a^3 = b^3 = c^4 = 1,\ ac = ca^{-1},\ aba^{-1} = bcb^{-1} \rangle$$

is trivial. For that purpose, we use `IsTrivial`:

```
gap> f := FreeGroup("a", "b", "c");;
gap> AssignGeneratorVariables(f);;
#I  Assigned the global variables [ a, b, c ]
```

```
gap> G := f/[a^3, b^3, c^4, c^(-1)*a*c*a,
> a*b*a^(-1)*b*c^(-1)*b^(-1)];;
gap> IsTrivial(G);
true
```

Example 2.68. In [39] it is proved that for a positive integer n,

$$\langle a, b \mid a^{-1}b^n a = b^{n+1},\ a = a^{i_1}b^{j_1}a^{i_2}b^{j_2}\cdots a^{i_k}b^{j_k}\rangle,$$

is trivial if $i_1 + i_2 + \cdots i_k = 0$. As an example, we show that

$$\langle a, b \mid a^{-1}b^2 a = b^3,\ a = a^{-1}ba\rangle$$

is the trivial group:

```
gap> f := FreeGroup("a", "b");;
gap> AssignGeneratorVariables(f);;
#I  Assigned the global variables [ a, b ]
gap> G := f/[a^(-1)*b^2*a*b^(-3), a^(-1)*(a^(-1)*b*a)];;
gap> IsTrivial(G);
true
```

Remark 2.2. The *word problem* asks for an algorithm that decides whether two given expressions are equivalent with respect to a set of rewriting identities. In 1955, P. Novikov proved that there exists a finitely presented group such that the word problem for this group is undecidable (see [44]). There are families of groups with a solvable word problem (e.g., finite groups, free (abelian) groups, (bi)automatic groups, Garside groups, hyperbolic groups); see for example [51].

For each $n \geq 2$, the *Burnside group* $B(2, n)$ is defined as the group

$$B(2, n) = \langle a, b \mid w^n = 1 \text{ for all word } w \text{ in the letters } a \text{ and } b\rangle.$$

Burnside's problem can be stated as follows: For which positive integer n is the free Burnside group $B(2, n)$ finite? Clearly, $B(2, 2) \simeq C_2 \times C_2$. Moreover, the following additional results are known: the groups $B(2, 3)$, $B(2, 4)$, and $B(2, 6)$ are finite. The cases $B(2, 5)$, $B(2, 7)$ and $B(2, 8)$ remain open. We refer [9, Chapter 6] for more information.

Example 2.69. We prove that the group $B(2, 3)$ is a finite group of order 27. Let F be the free group of rank two. Here we do not need to assign variables to the generators. We divide F by the normal subgroup generated by $\{w_1^3, \dots, w_{10000}^3\}$, where $w_1, \dots, w_{10000}$ are some randomly chosen words of F. The following code shows that $B(2, 3)$ is finite:

```
gap> f := FreeGroup(2);;
gap> rels := Set([1..10000], x->Random(f)^3);;
gap> B23 := f/rels;;
gap> Order(B23);
27
gap> Number(B23, x->IsOne(x^3));
27
```

Note that our group `B23` is exactly the group $B(2,3)$, as every non-trivial element has order three. The group $B(2,3)$ is isomorphic to the Heisenberg group

$$H_3 = \left\{ \begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix} : a, b, c \in \mathbb{F}_3 \right\} = \left\langle \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \right\rangle .$$

Here is the code:

```
gap> a := [[1,1,0],[0,1,0],[0,0,1]]*One(GF(3));;
gap> b := [[1,0,0],[0,1,1],[0,0,1]]*One(GF(3));;
gap> Display(a);
 1 1 .
 . 1 .
 . . 1
gap> Display(b);
 1 . .
 . 1 1
 . . 1
gap> H3 := Group([a, b]);;
gap> Order(H3);
27
gap> not IsomorphismGroups(B23, H3) = fail;
true
```

Example 2.70. It is known that $B(2,4)$ is a finite group of order 4096. Here we present a computational proof. We use the same trick as before:

```
gap> f := FreeGroup(2);;
gap> rels := Set([1..10000], x->Random(f)^4);;
gap> B24 := f/rels;;
gap> Order(B24);
4096
gap> Number(B24, x->IsOne(x^4));
4096
```

We conclude this chapter with examples of finitely presented groups that satisfy certain finiteness conditions. We will combine computer computations with different theorems to recover the structure of the center, the commutator subgroup, and the respective quotients.

Example 2.71. We will study the structure of the following finitely presented group:

$$G = \langle a, b, c \mid ab = ca, ac = ba, bc = ab \rangle .$$

We first construct G and check that it is not abelian. We will do this using the functions `AssignGeneratorVariables` and `ParseRelators`:

```
gap> f := FreeGroup("x", "y", "z");;
gap> AssignGeneratorVariables(f);;
#I  Assigned the global variables [ x, y, z ]
gap> G := f/ParseRelators(f, "xy=zx, xz=yx, yz=xy");
<fp group on the generators [ x, y, z ]>
gap> a := G.1;;
gap> b := G.2;;
gap> c := G.3;;
gap> IsAbelian(G);
false
```

We note that G is infinite, as $G/[G,G] \simeq \mathbb{Z}$:

```
gap> Order(G);
infinity
gap> AbelianInvariants(G/DerivedSubgroup(G));
[ 0 ]
gap> StructureDescription(G/DerivedSubgroup(G));
"Z"
```

Note that $a^2 = b^2 = c^2$ and $Z(G) = \langle b^2 \rangle \simeq \mathbb{Z}$ is a finite-index subgroup. In fact, $G/Z(G) \simeq \mathbb{S}_3$:

```
gap> a^2 = b^2 and b^2 = c^2;
true
gap> Center(G);
Group([ b^2 ])
gap> Order(Center(G));
infinity
gap> Index(G,Center(G));
6
gap> StructureDescription(G/Center(G));
"S3"
```

Note that the group homomorphism $G \to \mathbb{S}_3$,

$$a \mapsto (1\,2), \quad b \mapsto (1\,3), \quad c \mapsto (2\,3),$$

has kernel $Z(G)$. Since $Z(G)$ has finite-index in G, it follows that every conjugacy class of G is finite. For example:

```
gap> Size(ConjugacyClass(G, a));
3
gap> Size(ConjugacyClass(G, a*b));
2
gap> Size(ConjugacyClass(G, a*b*a^5));
3
```

Let $p\colon G \to C_2$ be the group homomorphism that sends the generators a, b and c to the permutation $(1\,2)$, i.e.,

$$a \mapsto (1\,2), \quad b \mapsto (1\,2), \quad c \mapsto (1\,2).$$

Then $\ker p \simeq \mathbb{Z} \times C_3$:

```
gap> C2 := Group([(1,2)]);;
gap> p := GroupHomomorphismByImages(G, C2, [a, b, c],\
> [(1,2), (1,2), (1,2)]);;
gap> IsInjective(p);
false
gap> IsSurjective(p);
true
gap> K := Kernel(p);;
gap> IsAbelian(K);
true
gap> AbelianInvariants(K);
[ 0, 3 ]
```

We claim that $[G,G] \simeq C_3$. On the one hand, since G is non-abelian, $[G,G]$ is non-trivial. Moreover, by a theorem of Schur [27, Theorem 5.7], $[G,G]$ is finite, as $Z(G)$ has finite-index in G. On the other hand, since $G/\ker p \simeq C_2$ is abelian, $[G,G] \subseteq \ker p$. The group $\ker p \simeq \mathbb{Z} \times C_3$ contains a unique non-trivial torsion subgroup. Hence $[G,G] \simeq C_3$.

Example 2.72. Let

$$G = \langle a, b, c, d \mid a^b = d, a^c = b, a^d = c, b^a = c \rangle.$$

We show that $\langle a^3 \rangle$ is a central subgroup of G and that $G/\langle a^3 \rangle \simeq \mathbf{SL}_2(3)$:

```
gap> f := FreeGroup("x", "y", "z", "w");;
gap> AssignGeneratorVariables(f);
#I  Assigned the global variables [ x, y, z, w ]
gap> G := f/ParseRelators(f, "x^y=w, x^z=y, x^w=z, y^x=z");;
gap> StructureDescription(G/Center(G));
"A4"
gap> a := GeneratorsOfGroup(G)[1];
gap> b := GeneratorsOfGroup(G)[2];
gap> c := GeneratorsOfGroup(G)[3];
gap> d := GeneratorsOfGroup(G)[4];
gap> a^3 = b^3 and b^3 = c^3 and c^3 = d^3;
true
gap> a^3 in Center(G);
true
gap> N := Subgroup(G, [a^3]);;
gap> StructureDescription(G/N);
"SL(2,3)"
gap> Size(ConjugacyClass(G, a));
4
```

This example is a particular case of a theorem of Dietzmann [27, Theorem 5.10]. This theorem states that if G is a group and X is a normal subset of G such that every element of X has finite order, then the subgroup generated by X is finite.

2.7 Problems

2.1. Compute the order of the subgroup of $\mathbf{GL}_2(\mathbb{Z})$ generated by

$$\begin{pmatrix}1&0\\0&1\end{pmatrix}, \qquad \begin{pmatrix}-1&0\\0&-1\end{pmatrix}, \qquad \begin{pmatrix}-1&0\\0&1\end{pmatrix}, \qquad \begin{pmatrix}1&0\\0&-1\end{pmatrix}.$$

Can you recognize this group?

2.2. Construct the Heisenberg group

$$H_5 = \left\{ \begin{pmatrix}1&a&b\\0&1&c\\0&0&1\end{pmatrix} : a,b,c \in \mathbb{F}_5 \right\}$$

as a permutation group.

2.3. Prove that all subgroups of $C_4 \times Q_8$ are normal.

2.4. Let G be the set of matrices of the form $\begin{pmatrix}1&b\\0&d\end{pmatrix}$, where $b \in \mathbb{F}_4$ and $d \in \mathbb{F}_4 \setminus \{0\}$. Prove that G is a group and compute its order.

2.5. Find all groups of order 12 that are non-trivial semidirect products.

2.6. Use the function `IsomorphicSubgroups` to prove that $\mathbb{A}_6$ does not contain a subgroup isomorphic to $\mathbb{S}_5$ and that $\mathbb{A}_7$ contains a subgroup isomorphic to $\mathbb{S}_5$.

2.7. Prove that $\mathbb{A}_6$ does not contain subgroups of prime-index.

2.8. Prove that $\mathbf{SL}_2(3)$ has a unique normal subgroup of order eight.

2.9. Find a subgroup of $\mathbf{SL}_2(5)$ isomorphic to $\mathbf{SL}_2(3)$.

2.10. Use the functions `SylowSubgroup` and `ConjugacyClassSubgroups` to construct all Sylow subgroups of $\mathbb{A}_4$ and $\mathbb{S}_4$.

2.11. Prove that $\mathbb{S}_5$ has a Sylow 2-subgroup isomorphic to the dihedral group of eight elements.

2.12. Can you recognize the structure of Sylow 2-subgroups of $\mathbb{S}_6$?

2.13. Use the function `Normalizer` to compute the number of conjugates of Sylow 2-subgroups of $\mathbb{A}_5$.

2.14. Find all Sylow subgroups of C_{27}, $\mathbf{SL}_2(5)$, $\mathbb{S}_7$, $\mathbb{S}_3 \times \mathbb{A}_4$ and $\mathbb{S}_3 \times C_{20}$.

2.15. Let $p \in \{2,3\}$. Compute the conjugacy classes of subgroups of $\mathbb{S}_3 \times \mathbb{S}_3$ and find (if possible) three Sylow p-subgroups, say A, B, C, such that $A \cap B = \{1\}$ and $A \cap C \neq \{1\}$.

2.16. Prove that the group

$$\left\{\begin{pmatrix}1 & b\\ 0 & d\end{pmatrix} : b, d \in \mathbb{F}_{19},\ d \neq 0\right\}$$

is not simple.

2.17. Let G be the group generated by the permutations $(1\,2)(6\,11)(8\,12)(9\,13)$, $(5\,13\,9)(6\,10\,11)(7\,8\,12)$ and $(2\,4\,3)(5\,8\,9)(6\,10\,13)(7\,11\,12)$. How many elements of G are commutators?

2.18. Prove that $\mathbb{A}_4 \times C_7$ does not contain subgroups of index two.

2.19. Prove that $\mathbb{A}_5$ does not contain subgroups of order $8, 15, 20, 24, 30, 40$.

2.20. It is known that an abelian subgroup of $\mathbb{S}_n$ has order $\leq 3^{\lfloor n/3 \rfloor}$. How good is this bound? For $n \in \{5, 6, 7, 8\}$ find (if possible) an abelian subgroup of $\mathbb{S}_n$ of order $3^{\lfloor n/3 \rfloor}$.

2.21. Prove that $\mathbf{SL}_2(5)$ does not contain subgroups isomorphic to $\mathbb{A}_5$, but contains a proper quotient isomorphic to $\mathbb{A}_5$.

2.22. Prove that for each positive divisor d of 24, there exists a subgroup of $\mathbb{S}_4$ of order d.

2.23. Let $G = \mathbf{SL}_2(13)$. Prove the following statements:

(a) G contains a unique element of order two.
(b) G does not have subgroups of order 42.
(c) $\mathbf{PSL}_2(13)$ is the only non-abelian simple quotient of G.

2.24. Can you recognize the group $\mathbf{SL}_3(2)/Z(\mathbf{SL}_3(2))$?

2.25. A *covering* of a group H is a group G such that $H = G/Z$ for some subgroup $Z \leq Z(G) \cap G'$.

(a) For each $z \in \{1, 2, 3\}$, prove that there exists a covering group G of $\mathbb{A}_6$ such that $|G|/|\mathbb{A}_6| = z$.
(b) Can you construct a covering group G such that $|G|/|\mathbb{A}_6| = 6$?

Hint: There is a largest perfect group G with $Z(G) = C_6$ and $G/Z(G) \simeq \mathbb{A}_6$; the remaining groups are just quotients of G.

2.26. Compute all the finite groups of order ≤ 240 that have $\mathbb{A}_5$ as a composition factor.

2.27. Let $\langle r, s \mid r^8 = s^2 = 1,\ srs = r^{-1}\rangle$ be the dihedral group of sixteen elements. Find all subgroups containing r^2.

2.28. Find all the group homomorphisms $\mathbb{S}_3 \longrightarrow \mathbf{SL}_2(3)$.

2.29. Are there any surjective homomorphisms $\mathbb{D}_{16} \to \mathbb{D}_8$? What about $\mathbb{D}_{16} \to C_2$?

2.30. Prove that $\mathrm{Aut}(\mathbb{A}_4) \simeq \mathbb{S}_4$.

2.31. Prove that $\mathrm{Aut}(\mathbb{D}_8) \simeq \mathbb{D}_8$ and that $\mathrm{Aut}(\mathbb{D}_{16}) \not\simeq \mathbb{D}_{16}$.

2.32. Compute the order of the group $\mathrm{Aut}(C_{11} \times C_2 \times C_3)$.

2.33. Prove that $\mathbb{D}_{12} \simeq \mathbb{S}_3 \times C_2$.

2.34. Let G be a group of order 12 such that $G \not\simeq \mathbb{A}_4$. Prove that G contains an element of order six.

2.35. Compute the list of normal subgroups of $\mathbf{GL}_2(3)$.

2.36. Compute the list of minimal subgroups of $\mathbf{PGL}_2(7)$.

2.37. Recall that a field automorphism for $L = \mathbf{PSL}_n(p^a)$ is an automorphism of L which is $\mathrm{Aut}(L)$-conjugate to the automorphism induced by some power of the Frobenius map $x \mapsto x^p$. Compute the list of subgroups of $\mathrm{Aut}(\mathbf{PSL}_2(9))$ containing a field automorphism of order two and $\mathbf{PSL}_2(9)$.

2.38. Use the function `CharacteristicSubgroups` to compute all characteristic subgroups of $\mathbf{GL}_3(7)$. Can you describe these subgroups?

2.39. Compute the socle and the list of minimal normal subgroups of $\mathbf{GL}_2(9)$.

2.40. Compute the Fitting and the Frattini subgroup of $\mathbf{SL}_2(3)$.

2.41. Compute the list of all maximal normal subgroups of $\mathbf{SL}_2(5)$.

2.42. Find all characteristic subgroups of the Heisenberg group H_5.

2.43. Prove that $\mathbf{PSL}_2(7)$ has a maximal subgroup of order 16.

2.44. Given a set of prime numbers π and a group G, we say that G is π-separable if there exists a chain of subgroups

$$\{1\} = N_0 \subseteq N_1 \subseteq \ldots \subseteq N_s = G,$$

where each N_i is normal in N_{i+1} and each N_{i+1}/N_i is either a π-group or a π'-group. Write a function that detects π-separable groups.

2.45. Let G be a finite group and H be a subgroup. The *Chermak–Delgado* measure of H is the number $m_G(H) = |H||C_G(H)|$.

(a) Write a function to compute the Chermak–Delgado measure.
(b) Compute $m_G(H)$ for $G \in \{\mathbb{S}_3, \mathbb{D}_8\}$ and H a subgroup of G.

2.46. Recall that the *holomorph* of a group G is the semidirect product $G \rtimes \mathrm{Aut}(G)$.

(a) Write a function that returns the holomorph of given a group G.

(b) Compute the holomorph of $\mathbb{A}_4$.
(c) Find a permutation representation of small degree of the holomorph of $\mathbb{A}_4$ and find a minimal normal subgroup of order four.

This is an exercise of [40].

2.47. Let G be a finite group and $\mathcal{P}$ be the set of subsets of G containing the identity of G and

$$f\colon \mathcal{P} \to \mathcal{P}, \quad S \mapsto \{xy^{-1} : x, y \in S\}.$$

(a) Write a function for the map f.
(b) Write a function that, given an element $S \in \mathcal{P}$, returns the smallest non-negative integer n such that $f^n(S) = f^{n+1}(S)$. Which subsets have $n = 0$?
(c) Write a function that computes

$$\mu(G) = \min\{n \geq 0 : f^n(S) = f^{n+1}(S) \text{ for all } S \in \mathcal{P}\}.$$

(d) Compute $\mu(\mathbb{S}_3)$, $\mu(\mathbb{D}_8)$ and $\mu(\mathbb{A}_4)$.

2.48. Prove that the group

$$\langle (1\,2\,3\,\cdots\,7), (2\,6)(3\,4)\rangle$$

is simple, has order 168 and acts transitively on $\{1, \dots, 7\}$. Can you recognize this group?

2.49. Compute the order of the group $\langle a, b \mid a^2 = b^2 = (bab^{-1})^3 = 1\rangle$.

2.50. Prove that $\langle a, b \mid a^2 = aba^{-1}b = 1\rangle$ is an infinite group.

2.51. Compute the order of the group $\langle a, b \mid a^8 = b^2a^4 = ab^{-1}ab = 1\rangle$.

2.52. Can you recognize the group $\langle a, b \mid a^5 = 1, b^2 = (ab)^3, (a^3ba^4b)^2 = 1\rangle$?

2.53. Prove that the group $\langle a, b \mid a^2 = b^3 = a^{-1}b^{-1}ab = 1\rangle$ is finite and cyclic.

2.54. Prove that the group $\langle a, b \mid a^2 = b^3 = 1\rangle$ is non-abelian.

2.55. Compute the order of the group

$$\langle a, b, c \mid a^3 = b^3 = c^3 = 1,\ aba = bab,\ cbc = bcb,\ ac = ca\rangle.$$

2.56. Prove that the group $\langle a, b, c \mid bab^{-1} = a^2,\ cbc^{-1} = b,\ aca^{-1} = c^2\rangle$ is trivial. This is an exercise of Serre's book [50, §1].

2.57. Find a permutation representation of the group $B(2, 3)$.

2.58. Prove that $B(3, 3) = \langle a, b, c \mid w^3 = 1 \text{ for all words } w \text{ in } a, b, c\rangle$ is a finite group. Can you compute the order of $B(3, 3)$?

2.59. Let G be a finite group with k conjugacy classes. It is known that the probability that two elements of G commute is equal to $\mathrm{prob}(G) = k/|G|$. Compute this probability for $\mathbf{SL}_2(3)$, $\mathbb{A}_4$, $\mathbb{A}_5$, $\mathbb{S}_4$ and Q_8.

2.60. Use the function `SubnormalSeries` to find a subnormal series for the subgroup of $\mathbf{SL}_2(3)$ generated by $\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$.

2.61. Use `DerivedSeriesOfGroup` to compute the derived series of $\mathbf{SL}_2(3)$. Can you compute the length of the derived series?

Chapter 3
Advanced group theory

Throughout this chapter, we work with more particular problems in group theory and use GAP to perform computations and verify certain conditions. We begin with a brief introduction to the group databases contained in the system, with particular emphasis on the `SmallGroup` library. Then we revisit group representations and character theory and take a closer look at the packages `repsn`, `AtlasRep`, and `CTblLib`. Since the latter already contains the character tables of many finite groups (for example, the sporadic simple groups), we use this information to significantly improve the performance of our calculations. Next, we employ these databases to analyze and test selected conjectures on group theory, some of them still open. For example, we investigate the celebrated McKay conjecture, Thompson's conjecture on products of conjugacy classes, and a conjecture raised by Harada in 2018 for which little is known We also review Ore's conjecture and Szep's conjecture, which were proved using the classification of finite simple groups.

By the end of the chapter, we give a brief introduction to group rings and provide examples including the computation of the Wedderburn decompositions with the package `Wedderga`. Then we finish by working with the Promislow group, which is an infinite group defined by generators and relations, study a matrix representation proposed in Passman's book, and present a script that checks Gardam's proof of the counterexample of the Kaplansky unit conjecture (namely, there are non-trivial units in the group ring of the Promislow group over the field of two elements).

3.1 Group databases

GAP contains several useful group databases. The groups are sorted by their orders and listed up to isomorphism. This database is part of a library named `SmallGroups`. It contains the following families of groups:

(a) Of order ≤ 2000 except order 1024.
(b) Of cube-free order ≤ 50000.
(c) Of order p^7 for $p \in \{3, 5, 7, 11\}$.

(d) Of order p^n for $n \leq 6$ and all primes p.
(e) Of order $q^n p$ for q^n dividing 2^8, 3^6, 5^5 or 7^4 and all primes p with $p \neq q$.
(f) Of square-free order.
(g) Of order that factorizes into at most three primes.

The library was written by H. Besche, B. Eick, and E. O'Brien.

As one can imagine, this library is a handy tool when one needs to look for examples and counterexamples.

Example 3.1. Let us see what `SmallGroups` knows about groups of order twelve:

```
gap> SmallGroupsInformation(12);

  There are 5 groups of order 12.
    1 is of type 6.2.
    2 is of type c12.
    3 is of type A4.
    4 is of type D12.
    5 is of type 2^2x3.

  The groups whose order factorises in at most 3 primes
  have been classified by O. Hoelder. This classification is
  used in the SmallGroups library.

  This size belongs to layer 1 of the SmallGroups library.
  IdSmallGroup is available for this size.
```

Some of the examples of this section were extracted from [35].

Example 3.2. Let us check that there exist non-abelian groups of odd order and that the smallest one has order 21:

```
gap> First(AllSmallGroups(Size, [1, 3..21]),\
> x->not IsAbelian(x));;
gap> Size(last);
21
```

Example 3.3. In one line, we can check that there are no simple groups of order 84. We use the filter `IsSimple` with the function `AllSmallGroups`:

```
gap> AllSmallGroups(Size, 84, IsSimple, true);
[  ]
```

With the function `StructureDescription`, one explores the structure of a given group. The function returns a short string, giving insight into the group's structure.

Example 3.4. Let us see what the groups of order twelve look like:

```
gap> List(AllSmallGroups(Size, 12), StructureDescription);
[ "C3 : C4", "C12", "A4", "D12", "C6 x C2" ]
```

The group `C3 : C4` denotes a semidirect product $C_3 \rtimes C_4$ of C_3 by C_4.

Example 3.5. Let us explore more group homomorphisms. We know that the symmetric group $\mathbb{S}_4$ is generated by the transpositions $(1\,2)$, $(2\,3)$ and $(3\,4)$. We let f be the group homomorphism $\mathbb{S}_4 \to \mathbb{S}_3$ given by $(1\,2) \mapsto (1\,2)$, $(2\,3) \mapsto (2\,3)$ and $(3\,4) \mapsto (1\,2)$. Let us perform some calculations related to this group homomorphism:

```
gap> S4 := SymmetricGroup(4);;
gap> S3 := SymmetricGroup(3);;
gap> f := GroupHomomorphismByImages(S4, S3, \
[(1,2), (2,3), (3,4)], [(1,2), (2,3), (1,2)]);;
gap> K := Kernel(f);;
gap> StructureDescription(K);
"C2 x C2"
gap> IsInjective(f);
false
gap> StructureDescription(S4/K);
"S3"
gap> StructureDescription(Image(f));
"S3"
gap> IsSurjective(f);
true
```

It is important to remark that the string returned by `StructureDescription` is not an isomorphism invariant: non-isomorphic groups can have the same string value and two isomorphic groups in different representations can produce different strings.

Example 3.6. Two groups of order 20 that can be written as a semidirect product $C_5 \rtimes C_4$. We see that `StructureDescription` does not distinguish such groups:

```
gap> List(AllSmallGroups(Size, 20), StructureDescription);
[ "C5 : C4", "C20", "C5 : C4", "D20", "C10 x C2" ]
```

To identify groups in the database `SmallGroups`, one uses the function `IdGroup`. Here we have some examples:

```
gap> IdGroup(SymmetricGroup(3));
[ 6, 1 ]
gap> IdGroup(SymmetricGroup(4));
[ 24, 12 ]
gap> IdGroup(AlternatingGroup(4));
[ 12, 3 ]
gap> IdGroup(DihedralGroup(8));
[ 8, 3 ]
gap> IdGroup(QuaternionGroup(8));
[ 8, 4 ]
```

Example 3.7. In [31], T. Lam and D. Leep proved that each index-two subgroup of $\mathrm{Aut}(\mathbb{S}_6)$ is isomorphic either to $\mathbb{S}_6$, $\mathbf{PGL}_2(9)$ or to the Mathieu group M_{10}. Let's check this claim using the function `IdGroup`:

```
gap> autS6 := AutomorphismGroup(SymmetricGroup(6));;
gap> List(SubgroupsOfIndexTwo(autS6), IdGroup);
[ [ 720, 764 ], [ 720, 763 ], [ 720, 765 ] ]
gap> IdGroup(PGL(2,9));
[ 720, 764 ]
gap> IdGroup(MathieuGroup(10));
[ 720, 765 ]
gap> IdGroup(SymmetricGroup(6));
[ 720, 763 ]
```

Example 3.8. Now we prove a theorem of R. Guralnick [19]. The theorem states that the smallest finite group G such that $\{[x, y] : x, y \in G\} \neq [G, G]$ has order 96.

```
gap> G := First(AllSmallGroups(Size, [1..100]),\
> x->Order(DerivedSubgroup(x)) <> Size(\
> Set(Cartesian(x, x), Comm)));;
gap> Order(G);
96
gap> IdGroup(G);
[ 96, 3 ]
```

With `IdGroup` (or with `IsomorphismGroups`) we check that

$$G \simeq \langle (1\,2\,5)(3\,6\,11)(4\,7\,9)(8\,10\,12), (1\,8\,4\,3)(5\,12)(6\,10)(9\,11)\rangle.$$

How do we find this isomorphism? We use `IsomorphismPermGroup` to construct a faithful representation of G as a permutation group. What is the minimal possible degree of a faithful permutation representation? We can find this number with `MinimalFaithfulPermutationDegree`:

```
gap> MinimalFaithfulPermutationDegree(G);
12
```

With the function, `MinimalFaithfulPermutationRepresentation`, we construct an isomorphic permutation group of minimal degree:

```
gap> f := MinimalFaithfulPermutationRepresentation(G);
[ f1, f2, f3, f4, f5, f6 ] ->
[ (1,2,5)(3,6,11)(4,7,9)(8,10,12),
  (1,3)(2,10,7,6)(4,8)(11,12),
  (2,6)(3,8)(5,12,9,11)(7,10), (5,9)(11,12),
  (2,7)(5,9)(6,10)(11,12),
  (1,4)(2,7)(3,8)(5,9)(6,10)(11,12) ]
```

We obtain an isomorphic copy of G inside $\mathbb{S}_{12}$. To construct a minimal set of generators, we then use `MinimalGeneratingSet`:

```
gap> P := Image(f);
Group([ (1,2,5)(3,6,11)(4,7,9)(8,10,12),
  (1,3)(2,10,7,6)(4,8)(11,12),
  (2,6)(3,8)(5,12,9,11)(7,10),
  (5,9)(11,12), (2,7)(5,9)(6,10)(11,12),
  (1,4)(2,7)(3,8)(5,9)(6,10)(11,12) ])
gap> MinimalGeneratingSet(P);
[ (1,2,5)(3,6,11)(4,7,9)(8,10,12),
  (1,8,4,3)(5,12)(6,10)(9,11) ]
```

In Example 3.8 we could have tried to find a faithful representation of small degree with the function `SmallerDegreePermutationRepresentation`. Note that independent calls to this function may return representations of different and non-minimal degrees. In the same vein, `SmallGeneratingSet` returns a “small" set of generators that may not be of minimal size. In general, the functions mentioned in this remark run faster than those used in Example 3.8.

For a finite group G, let $\operatorname{cs}(G)$ denote the set of sizes of the conjugacy classes of G, that is

$$\operatorname{cs}(G) \coloneqq \{|g^G| : g \in G\}.$$

For example,

$$\operatorname{cs}(\mathbb{S}_3) = \{1,2,3\}, \quad \operatorname{cs}(\mathbb{A}_5) = \{1,12,15,20\}, \quad \operatorname{cs}(\mathbf{SL}_2(3)) = \{1,4,6\}.$$

Here is the code that verifies our claim:

```
gap> cs := function(G)
> return Set(ConjugacyClasses(G), Size);
> end;
function( G ) ... end
gap> cs(SymmetricGroup(3));
[ 1, 2, 3 ]
gap> cs(AlternatingGroup(5));
[ 1, 12, 15, 20 ]
gap> cs(SL(2,3));
[ 1, 4, 6 ]
```

We will write $G_{n,k}$ to denote the k-th group of size n in the database, thus $G_{n,k}$ is a group with `IdGroup` equal to `[ n, k ]`.

Example 3.9. This example is taken from [41, Theorem A] and answers a question posed by R. Brauer [7, Question 2(ii)]. G. Navarro proved that there exist finite groups G and H such that G is solvable, H is not solvable and $\operatorname{cs}(G) = \operatorname{cs}(H)$.

Let

$$G = G_{240,13} \times G_{960,1019} \quad \text{and} \quad H = G_{960,239} \times G_{480,959}.$$

Then G is solvable and H is not. Moreover, $\operatorname{cs}(G) = \operatorname{cs}(H)$, as the following code shows:

```
gap> U := SmallGroup(960, 239);;
gap> V := SmallGroup(480, 959);;
gap> L := SmallGroup(960, 1019);;
gap> K := SmallGroup(240, 13);;
gap> UxV := DirectProduct(U, V);;
gap> KxL := DirectProduct(K, L);;
gap> IsSolvable(UxV);
false
gap> IsSolvable(KxL);
true
```

One could try to directly compute $\mathrm{cs}(U \times V)$. However, this calculation seems to be hard. The trick is to use that

$$\mathrm{cs}(U \times V) = \{nm : n \in \mathrm{cs}(U), m \in \mathrm{cs}(V)\}.$$

Here is the code:

```
gap> cs(KxL) = Set(Cartesian(cs(U), cs(V)), x->x[1]*x[2]);
true
```

Example 3.10. This example appeared in [41]. It answers another question raised by R. Brauer [7, Question 4(ii)]. G. Navarro proved that there exist finite groups G and H such that G is nilpotent, $Z(H) = 1$ and $\mathrm{cs}(G) = \mathrm{cs}(H)$. The groups are $G = \mathbb{D}_8 \times G_{243,26}$ and $H = G_{486,36}$. Here is the code:

```
gap> K := DihedralGroup(8);;
gap> L := SmallGroup(243, 26);;
gap> H := SmallGroup(486, 36);;
gap> IsTrivial(Center(H));
true
gap> G := DirectProduct(K, L);;
gap> cs(G) = cs(H);
true
gap> IsNilpotent(G);
true
```

Example 3.11. Let p be a prime number and G be a finite p-group. We denote by $s_m(G)$ the number of subgroups of G of order p^m. The following function computes $s_m(G)$ when G is a p-group:

```
gap> s := function(m, G)
> local p, f;
> if not IsPGroup(G) then
>   return fail;
> fi;
> p := PrimeDivisors(Order(G))[1];
> f := Filtered(ConjugacyClassesSubgroups(G), \
  x->Order(Representative(x)) = p^m);
> return Sum(f, Size);
> end;
function( m, G ) ... end
```

For example,

$$s_m(\mathbb{D}_{32}) = \begin{cases} 1 & \text{if } m \in \{0,5\}, \\ 17 & \text{if } m = 1, \\ 9 & \text{if } m = 2, \\ 5 & \text{if } m = 3, \\ 3 & \text{if } m = 4. \end{cases}$$

The previous result was obtained with the following code:

```
gap> G := DihedralGroup(2^5);;
gap> List([0..5], m->s(m,G));
[ 1, 17, 9, 5, 3, 1 ]
```

In [2], Berkovich proved that if $p > 2$ is a prime number and G is a finite p-group of order p^n and exponent p, then

$$s_m(G) \equiv 1 + p + 2p^2 \bmod p^3$$

for all $m \in \{2,\dots,n-2\}$.

The following function tests Berkovich's theorem:

```
gap> Berkovich := function(G)
> local p, n;
> p := PrimeDivisors(Order(G))[1];
> n := LogInt(Order(G), p);
> return ForAll([2..n-2], m->RemInt(1+p+2*p^2-s(m,G), p^3) = 0);
> end;
function( G ) ... end
```

With this, we now quickly check the theorem for some particular groups:

```
gap> G := Random(AllGroups(Size, 3^7, Exponent, 3));;
gap> Berkovich(G);
true
gap> G := Random(AllGroups(Size, 5^4, Exponent, 5));;
gap> Berkovich(G);
true
gap> G := Random(AllGroups(Size, 11^4, Exponent, 11));;
gap> Berkovich(G);
true
```

A group G is said to be *quasisimple* if G is perfect and $G/Z(G)$ is (non-abelian) simple.

Example 3.12. The group $\mathbf{SL}_2(5)$ is quasisimple:

```
gap> G := SL(2,5);;
gap> IsPerfect(G);
true
gap> IsSimple(G/Center(G));
true
gap> IsQuasiSimple(G);
true
```

However, $\mathbf{GL}_2(5)$ is not quasisimple since it is not perfect:

```
gap> G := GL(2,5);;
gap> IsPerfect(G);
false
gap> IsQuasiSimple(G);
false
```

Example 3.13. We prove that the smallest finite perfect group that is not quasisimple is a group of the form $C_2^4 \rtimes \mathbb{A}_5$.

With `SizeNumbersPerfectGroups` we list the orders of perfect groups in the database and find the group needed:

```
gap> for x in SizeNumbersPerfectGroups() do
> G := PerfectGroup(x);
> if not IsQuasiSimple(G) then
>   Display(x);
>   break;
> fi;
> od;
[ 960, 1 ]
gap> StructureDescription(G);
"(C2 x C2 x C2 x C2) : A5"
```

There are two perfect groups of order 960. Both groups are not quasisimple:

```
gap> NrPerfectGroups(960);
2
gap> IsQuasiSimple(PerfectGroup(960, 2));
false
```

Example 3.14. In [3], H. Blau proved proved that there exist (finitely many) quasisimple groups that contain central elements that are non-commutators. The smallest of such groups is a Schur covering of $\mathbb{A}_6$.

We first write an efficient function to compute the set of commutators of a given group:

```
gap> SetOfCommutators := function(G)
> local T;
> T := Elements(RightTransversal(G, Center(G)));
> return Set(Cartesian(T, T), x->Comm(x[1], x[2]));
> end;
function( G ) ... end
```

Note that this function is more efficient than the one we presented on page 61.

In the following code it is crucial to construct perfect groups as permutation groups, as calculations will be much faster:

```
gap> for x in SizeNumbersPerfectGroups() do
> G := PerfectGroup(IsPermGroup, x);
> dif := Difference(Center(G), SetOfCommutators(G));
> if Size(dif) > 0 then
>   Display(x);
>   break;
> fi;
> od;
[ 2160, 1 ]
gap> A6 := AlternatingGroup(6);;
gap> not IsomorphismGroups(SchurCover(A6), G) = fail;
true
```

A group G is *n-transitive* on a set X if for any two sequences of distinct points $x_1, \dots, x_n$ and $y_1, \dots, y_n$ there exists $g \in G$ such that $x_i^g = y_i$ for all $i \in \{1, \dots, n\}$. If in addition such g is always unique, we say that G is *sharply n-transitive* on X.

Example 3.15. Let us prove that there is no sharply 4-transitive group of degree seven or nine. This solves [6, Exercise 1.18].

We first create a list of all transitive groups of degrees seven and nine and we keep only those groups that are at least 4-transitive:

```
gap> l := AllTransitiveGroups(NrMovedPoints, [7, 9], \
> Transitivity, [4..9]);
[ A7, S7, A9, S9 ]
```

Finally, we check that none of these groups are sharply 4-transitive. For that purpose, we see that the action on tuples is never regular:

```
gap> ForAny(l, x->IsRegular(x, \\
> Arrangements([1..NrMovedPoints(x)], 4), OnTuples));
false
```

It is known that a finite permutation group that does not contain the alternating group is, at most, 5-transitive. Except for the alternating and symmetric groups, the only finite groups which are 4- or 5-transitive are the Mathieu groups M_{11}, M_{12}, M_{23} and M_{24}. The proof of this statement depends on the classification of finite simple groups.

Let G be a group that acts transitively on a set X. A *block* of the action is a non-empty subset Δ of X such that either $\Delta^g = \Delta$ or $\Delta^g \cap \Delta = \emptyset$ for all $g \in G$. We say that G is *primitive* on X if the only blocks are the one-element subsets of X and X.

Example 3.16. Let us prove that primitive groups of degree eight are double transitive. First, we note that there are seven primitive groups of degree eight:

```
gap> l := AllPrimitiveGroups(NrMovedPoints, 8);
[ AGL(1, 8), AGammaL(1, 8), ASL(3, 2),
  PSL(2, 7), PGL(2, 7), A(8), S(8) ]
```

To check that all these groups are indeed at least 2-transitive, we use the function `Transitivity`:

```
gap> List(l, Transitivity);
[ 2, 2, 3, 2, 3, 6, 8 ]
gap> ForAll(last, x->x > 1);
true
```

The *commuting probability* $p(G)$ of a finite group G is defined as the probability that a randomly chosen pair of elements of G commute. Then $p(G) = k(G)/|G|$, where $k(G)$ is the number of conjugacy classes of G. Problem 2.59 asks to compute the commuting probability of a finite group. We first present a solution to this problem:

```
gap> CommutingProbability := G->NrConjugacyClasses(G)/Order(G);
function( G ) ... end
```

In 1970, J. C. Dixon observed that the commuting probability of a finite non-abelian simple group is $\leq 1/12$. This bound is attained for the alternating simple group $\mathbb{A}_5$:

```
gap> CommutingProbability(AlternatingGroup(5));
1/12
```

One can find Dixon's proof in the 16-th volume of the *Canadian Mathematical Bulletin* of 1973. The proof we present here is based on a proof due to I. Sadofschi Costa. We first assume that the commuting probability of G is $> 1/12$. Since G is a non-abelian simple group, the identity is the only central element. Let us assume first that there is a conjugacy class of G of size m, where m is such that $1 < m \leq 12$. Then G is a transitive subgroup of $\mathbb{S}_m$. For these groups the problem is easy: we show that there are no non-abelian simple groups that act transitively on sets of size $m \in \{2, \dots, 12\}$ with commuting probability $> 1/12$. To do this, we list these transitive groups and their commuting probabilities and verify that all commuting probabilities are $\leq 1/12$:

```
gap> l := AllTransitiveGroups(NrMovedPoints, [2..12], \\
> IsAbelian, false, IsSimple, true);;
[ A5, L(6) = PSL(2,5) = A_5(6), A6,
  L(7) = L(3,2), A7, L(8)=PSL(2,7), A8,
  L(9)=PSL(2,8), A9, A_5(10), L(10)=PSL(2,9),
  A10, L(11)=PSL(2,11)(11), M(11), A11, A_5(12),
  L(2,11), M_11(12), M(12), A12 ]
gap> List(l, CommutingProbability);
[ 1/12, 1/12, 7/360, 1/28, 1/280, 1/28, 1/1440,
  1/56, 1/10080, 1/12, 7/360, 1/75600, 2/165,
  1/792, 31/19958400, 1/12, 2/165, 1/792, 1/6336,
  43/239500800 ]
gap> ForAny(l, x->CommutingProbability(x) > 1/12);
false
```

Now assume that all non-trivial conjugacy classes of G have at least 13 elements. Then the class equation implies that

$$|G| \geq \frac{13}{12}|G| - 12,$$

and therefore $|G| \leq 144$. Thus one needs to check what happens with groups of order ≤ 144. But we know that the only non-abelian simple group of size ≤ 144 is the alternating simple group $\mathbb{A}_5$.

```
gap> AllGroups(Size, [2..144], IsAbelian, false, IsSimple, true);
[ Alt( [ 1 .. 5 ] ) ]
```

The following example implements a remarkable trick that goes back to R. Baer. We refer to [27, Lemma 4.4.37] for more information.

Example 3.17 (Baer trick). Let G be a finite nilpotent group of odd order and nilpotency class at most two. For example, the small group $G_{27,3}$ has nilpotency class two:

```
gap> G := SmallGroup(27, 3);;
gap> NilpotencyClassOfGroup(G);
2
```

Since G has odd order, the map $G \to G, x \mapsto x^2$, is bijective. To construct this map, we use `MappingByFunction`:

```
gap> f := MappingByFunction(G, G, x->x^2);;
gap> IsBijective(f);
true
```

In particular, every element x of G admits a unique square root $\sqrt{x}$:

```
gap> sqrt := Inverse(f);;
```

Baer proved that the operation

$$x + y = xy\sqrt{[y, x]}$$

turns G into an abelian group. To construct this group, we use Cayley's theorem:

```
gap> BaerTrick := function(G)
> local m, i, j, x, y, l, sqrt;
> n := Order(G);
> l := AsList(G);
> m := NullMat(n,n);
> sqrt := Inverse(MappingByFunction(G, G, x->x^2));
> for i in [1..n] do
>   x := l[i];
>   for j in [1..n] do
>     y := l[j];
>     m[i][j] := Position(l, x*y*sqrt(Comm(y,x)));
>   od;
> od;
> return Group(List(m, PermList));
> end;
function( G ) ... end
```

Group	Baer trick
$G_{27,3} \simeq (C_3 \times C_3) \rtimes C_3$	$C_3 \times C_3 \times C_3$
$G_{27,4} \simeq C_9 \rtimes C_3$	$C_9 \times C_3$
$G_{81,3} \simeq (C_9 \rtimes C_3) \rtimes C_3$	$C_9 \times C_3 \times C_3$
$G_{81,4} \simeq C_9 \rtimes C_9$	$C_9 \times C_9$
$G_{81,6} \simeq C_{27} \rtimes C_3$	$C_{27} \times C_3$
$G_{81,12} \simeq C_3 \times ((C_3 \times C_3) \rtimes C_3)$	$C_3 \times C_3 \times C_3 \times C_3$
$G_{81,13} \simeq C_3 \times (C_9 \rtimes C_3)$	$C_9 \times C_3 \times C_3$
$G_{81,14} \simeq (C_9 \times C_3) \rtimes C_3$	$C_9 \times C_3 \times C_3$

For example, the table
shows abelian groups obtained by Baer's construction. The code to produce such table is the following:

```
gap> for G in AllGroups(Size, [1,3..99], \
> IsNilpotent, true, NilpotencyClassOfGroup, 2) do
> A := BaerTrick(G);
> Print(IdGroup(G), ", ", StructureDescription(G), ",  ", \
> StructureDescription(A), "\n");
> od;
[ 27, 3 ], (C3 x C3) : C3,  C3 x C3 x C3
[ 27, 4 ], C9 : C3,  C9 x C3
[ 81, 3 ], (C9 x C3) : C3,  C9 x C3 x C3
[ 81, 4 ], C9 : C9,  C9 x C9
[ 81, 6 ], C27 : C3,  C27 x C3
[ 81, 12 ], C3 x ((C3 x C3) : C3),  C3 x C3 x C3 x C3
[ 81, 13 ], C3 x (C9 : C3),  C9 x C3 x C3
[ 81, 14 ], (C9 x C3) : C3,  C9 x C3 x C3
```

3.2 Representations

A *(matrix) representation* of a group G is a group homomorphism

$$\rho\colon G \to \mathbf{GL}_n(K), \quad g \mapsto \rho_g,$$

where $n \geq 1$ and K is a field. The representation ρ is said to be *faithful* if ρ is injective. The *character* of the representation ρ is the function $\chi\colon G \to K$ where $\chi(g)$ is trace of the matrix ρ_g. Recall that, when G is finite and $|G|$ is invertible in K, the representation ρ is irreducible if and only if

$$\langle \chi, \chi \rangle = \frac{1}{|G|} \sum_{g \in G} \chi(g)\chi(g^{-1}) = 1.$$

We refer to [26] and [49] for an introduction to the representation theory of finite groups.

Example 3.18. Let us construct the representation ρ of $\mathbb{A}_4$ given by

$$(1\,2)(3\,4) \mapsto \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & -1 \\ 0 & 0 & -1 \end{pmatrix}, \quad (1\,2\,3) \mapsto \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{pmatrix}.$$

We use the function `GroupHomomorphismByImages`:

```
gap> A4 := AlternatingGroup(4);;
gap> a := [[0, 1, -1], [1, 0, -1], [0, 0, -1]];;
gap> b := [[0, 0, -1], [0, 1, -1], [1, 0, -1]];;
gap> rho := GroupHomomorphismByImages(A4,\
> [ (1,2)(3,4), (1,2,3) ], [ a, b ]);;
gap> IsGroupHomomorphism(rho);
true
```

This is indeed a faithful representation of $\mathbb{A}_4$:

```
gap> IsTrivial(Kernel(rho));
true
```

Just to see how it works, let us compute $\rho_{(1\,3\,2)}$, the image of $(1\,3\,2) \in \mathbb{A}_4$ under ρ. We are working with 3×3 matrices so it is better to use the function `Display`:

```
gap> Display(Image(rho, (1,3,2)));
[ [  -1,   0,   1 ],
  [  -1,   1,   0 ],
  [  -1,   0,   0 ] ]
```

Now we construct the character χ of ρ. We also check that ρ is irreducible since

$$\langle \chi, \chi \rangle = \frac{1}{|\mathbb{A}_4|} \sum_{x \in \mathbb{A}_4} \chi(x)\chi(x^{-1}) = 1.$$

```
gap> chi := x->TraceMat(x^rho);;
gap> 1/Order(A4)*Sum(List(A4, x->chi(x)*chi(x^(-1))));
1
```

We now construct irreducible representations of a given group. This can be done with the package `Repsn`, written by V. Dabbaghian.

Example 3.19. Let us construct the irreducible representations of $\mathbb{S}_3$. The irreducible characters of a finite group can be constructed with `Irr`:

```
gap> S3 := SymmetricGroup(3);;
gap> irr := Irr(S3);
[ Character( CharacterTable( Sym( [ 1 .. 3 ] ) ), [ 1, -1, 1 ] ),
  Character( CharacterTable( Sym( [ 1 .. 3 ] ) ), [ 2, 0, -1 ] ),
  Character( CharacterTable( Sym( [ 1 .. 3 ] ) ), [ 1, 1, 1 ] ) ]
```

To construct irreducible representations we need to load the package `repsn`:

```
gap> LoadPackage("repsn");
```

The package contains `IrreducibleAffordingRepresentation`. This function produces irreducible representations from irreducible characters. Since we are working with $\mathbb{S}_3$, we will only need to consider the character of degree two. We will produce the faithful representation $\mathbb{S}_3 \to \mathbf{GL}_2(\mathbb{C})$ given by

$$(1\,2\,3) \mapsto \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega \end{pmatrix}, \quad (1\,2) \mapsto \begin{pmatrix} 0 & \omega \\ \omega^2 & 0 \end{pmatrix},$$

where ω is a primitive cubic root of one. Here is the code:

```
gap> f := IrreducibleAffordingRepresentation(irr[2]);
[ (1,2,3), (1,2) ] -> [ [ [ E(3)^2, 0 ], [ 0, E(3) ] ],
  [ [ 0, E(3) ], [ E(3)^2, 0 ] ] ]
gap> Image(f, (1,2,3));
[ [ E(3)^2, 0 ], [ 0, E(3) ] ]
gap> Display(Image(f, (1,2,3)));
[ [  E(3)^2,       0 ],
  [       0,    E(3) ] ]
gap> Display(Image(f, (1,2)));
[ [       0,    E(3) ],
  [  E(3)^2,       0 ] ]
```

In [4, Problem 1], R. Brauer asked which algebras are group algebras. This question might be very hard to answer in a general situation. Here we play with some particular examples, combining algebra and computer calculations.

Example 3.20. Is

$$\mathbb{C} \times M_2(\mathbb{C}) \times M_5(\mathbb{C})$$

a (complex) group algebra?

The answer is no. By Wedderburn's theorem, if $\mathbb{C} \times M_2(\mathbb{C}) \times M_5(\mathbb{C})$ is the group algebra of some group G, then G has order 30 with three irreducible characters of degrees one, two and five, respectively. We will prove that there are no such groups.

We list the degrees of irreducible characters of groups of order 30. We see that there are four groups of order 30 and none of them has exactly three irreducible characters:

```
gap> n := 30;;
gap> for G in AllGroups(Size, n) do
> Display(Size(Irr(G)));
> od;
15
12
9
30
```

We now compute the degrees of the irreducible characters of groups of order 30:

```
gap> for G in AllGroups(Size, n) do
> Print(CharacterDegrees(G), "\n");
> od;
```

```
[ [ 1, 10 ], [ 2, 5 ] ]
[ [ 1, 6 ], [ 2, 6 ] ]
[ [ 1, 2 ], [ 2, 7 ] ]
[ [ 1, 30 ] ]
```

In fact, this shows that the groups algebras of groups of order 30 are

$$\mathbb{C}^{10} \times M_2(\mathbb{C})^5, \qquad \mathbb{C}^6 \times M_2(\mathbb{C})^6, \qquad \mathbb{C}^2 \times M_2(\mathbb{C})^7, \qquad \mathbb{C}^{30}.$$

The package `CTblLib`, written by T. Breuer, contains character tables of some groups. For example, it includes the character tables of all the sporadic simple groups, and several of their matrices (over the rationals and finite fields) and permutation representations.

Example 3.21. The first Janko group J_1 is a simple group of order 175560. It has 15 conjugacy classes (and hence 15 irreducible representations). We can get all this information only from the character table:

```
gap> t := CharacterTable("J1");;
gap> Size(t);
175560
gap> IsSimple(t);
true
gap> NrConjugacyClasses(t);
15
gap> Size(Irr(t));
15
gap> LinearCharacters(t);
[ Character( CharacterTable( "J1" ),
  [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ) ]
gap> CharacterDegrees(t);
[ [ 1, 1 ], [ 56, 2 ], [ 76, 2 ], [ 77, 3 ],
  [ 120, 3 ], [ 133, 3 ], [ 209, 1 ] ]
gap> SizesConjugacyClasses(t);
[ 1, 1463, 5852, 5852, 5852, 29260, 25080, 17556,
  17556, 15960, 11704, 11704, 9240, 9240, 9240 ]
gap> SizesCentralizers(t);
[ 175560, 120, 30, 30, 30, 6, 7, 10, 10, 11, 15, 15,
  19, 19, 19 ]
```

The package `AtlasRep` provides an interface to the *Atlas of Finite Group Representations* [11]. It comprises representations of some (almost) simple groups and information about their maximal subgroups and conjugacy classes. It was prepared by R. Abbott, J. Bray, S. Linton, S. Nickerson, S. Norton, R. Parker, I. Suleiman, J. Tripp, P. Walsh and R. Wilson. To load the package we proceed as follows:

```
gap> LoadPackage("atlasrep");
```

We refer to the `atlasrep` manual for information related to the different ways of obtaining the database from the Internet.

Example 3.22. We compute the structure of the Sylow subgroups of the first Janko group J_1. By default, the package uses a permutation representation (inside $\mathbb{S}_{266}$) for the group J_1:

```
gap> J1 := AtlasGroup("J1");
<permutation group of size 175560 with 2 generators>
gap> Collected(Factors(Order(J1)));
[ [ 2, 3 ], [ 3, 1 ], [ 5, 1 ], [ 7, 1 ], [ 11, 1 ], [ 19, 1 ] ]
gap> NrMovedPoints(J1);
266
gap> for p in PrimeDivisors(Order(J1)) do
> Display(StructureDescription(SylowSubgroup(J1, p)));
> od;
C2 x C2 x C2
C3
C5
C7
C11
C19
```

The function `DisplayAtlasInfo` displays all the information available on a given group. For example, regarding the Janko group J_1, the first lines of the output are:

```
gap> DisplayAtlasInfo("J1");
Representations for G = J1:    (all refer to std. generators 1)
---------------------------
 1: G <= Sym(266)     rank 5, on cosets of L2(11) (1st max.)
 2: G <= Sym(1045)    rank 11, on cosets of 2^3:7:3 (2nd max.)
 3: G <= Sym(1463)    rank 22, on cosets of 2xA5 (3rd max.)
 4: G <= Sym(1540)    rank 21, on cosets of 19:6 (4th max.)
 5: G <= Sym(1596)    rank 19, on cosets of 11:10 (5th max.)
 6: G <= Sym(2926)    rank 67, on cosets of D6xD10 (6th max.)
 7: G <= Sym(4180)    rank 107, on cosets of 7:6 (7th max.)
 8: G <= GL(20,2)     character 20a
 ...
```

The full output shows that the package contains 91 (faithful) representations for J_1.

The function `AtlasGenerators` returns a record with information on a particular representation. For example, the 47-th one is a representation of dimension seven over the field $\mathbb{F}_{11}$:

```
gap> recJ1 := AtlasGenerators("J1", 47);;
gap> recJ1!.ring;
GF(11)
gap> gens := recJ1!.generators;
[ < immutable compressed matrix 7x7 over GF(11) >,
  < immutable compressed matrix 7x7 over GF(11) > ]
gap> Display(gens[1]);
  .  7  9  9 10  7  9
  7  .  1  4  3  3  4
  9  1  2  8  3  6  2
  9  4  8 10  1  6  .
 10  3  3  1  8  9  1
  7  3  6  6  9  1  3
  9  4  2  .  1  3  .
gap> Display(gens[2]);
  5  7  1  8  5  8  2
 10  4 10  .  3  3  8
  5 10  8  9  1  2  1
  2  3  3  6  1  2  9
  7  4  7  2 10  7  3
  3  9  .  4  2  3  5
  3  4  3  3  9  4  9
gap> Order(Group(gens));
175560
gap> IsSimple(Group(gens));
true
```

Example 3.23. We now construct the first Conway group Co_1 and check that each Sylow 2-subgroup is self-normalizing:

```
gap> Co1 := AtlasGroup("Co1");
<permutation group of size 4157776806543360000 with 2 generators>
gap> Collected(Factors(Order(Co1)));
[ [ 2, 21 ], [ 3, 9 ], [ 5, 4 ], [ 7, 2 ], [ 11, 1 ],
  [ 13, 1 ], [ 23, 1 ] ]
gap>
gap> IsSimple(Co1);
true
gap> P := SylowSubgroup(Co1, 2);
<permutation group of size 2097152 with 16 generators>
gap> N := Normalizer(Co1, P);
<permutation group with 16 generators>
gap> Order(N) = Order(P);
true
```

The package `AtlasRep` contains representative of conjugacy classes of (some) maximal subgroups of (some) finite sporadic simple groups.

Example 3.24. Let Th be the Thompson sporadic simple group. Using maximal subgroups we will easily describe the structure of the Sylow 5-subgroups: these are semidirect products of the form $C_5^2 \rtimes C_5$. To this aim, we first get the generators of the 10-th maximal subgroup of Th which has order 12000 and hence contains a Sylow 5-subgroup of Th. This maximal subgroup is represented using matrices over $\mathbb{F}_2$. As group calculations involving big matrices are, in general, hard to perform, we use a permutation group representation:

```
gap> gens := AtlasGenerators("Th", 1, 10).generators;
[ <an immutable 248x248 matrix over GF2>,
  <an immutable 248x248 matrix over GF2> ]
gap> m10 := Image(IsomorphismPermGroup(Group(gens)));;
gap> Order(m10);
12000
gap> StructureDescription(SylowSubgroup(m10, 5));
"(C5 x C5) : C5"
```

Can you try to compute the structure of the Sylow 5-subgroups doing the calculations directly in Th?

We say that two finite groups *have the same character table* if their character tables differ from permutations of rows and columns. There are non-isomorphic groups which have the same character table.

Example 3.25. It is well-known that the groups $\mathbb{D}_8$ and Q_8 are non-isomorphic and yet have the same character table. This can be easily seen as follows:

```
gap> D8 := DihedralGroup(8);;
gap> t_D8 := CharacterTable(D8);;
gap> Display(t_D8);
CT1

     2  3  2  2  3  2

       1a 2a 4a 2b 2c

X.1     1  1  1  1  1
X.2     1 -1  1  1 -1
X.3     1  1 -1  1 -1
X.4     1 -1 -1  1  1
X.5     2  .  . -2  .
gap> Q8 := QuaternionGroup(8);;
gap> t_Q8 := CharacterTable(Q8);;
gap> Display(t_Q8);
CT2

     2  3  2  2  3  2

       1a 4a 4b 2a 4c
    2P 1a 2a 2a 1a 2a
    3P 1a 4a 4b 2a 4c

X.1     1  1  1  1  1
X.2     1 -1 -1  1  1
X.3     1 -1  1  1 -1
X.4     1  1 -1  1 -1
X.5     2  .  . -2  .
```

It can be proved that $\mathbb{D}_8$ and Q_8 are not isomorphic in many different ways. For example, this follows from counting the number of elements of order two:

```
gap> OrdersClassRepresentatives(t_D8);
[ 1, 2, 4, 2, 2 ]
gap> OrdersClassRepresentatives(t_Q8);
[ 1, 4, 4, 2, 4 ]
```

With the function `TransformingPermutations` we verify that the matrices of characters are equivalent:

```
gap> Display(Irr(t_D8));
[ [    1,    1,    1,    1,    1 ],
  [   1,  -1,   1,   1,  -1 ],
  [   1,   1,  -1,   1,  -1 ],
  [   1,  -1,  -1,   1,   1 ],
  [   2,   0,   0,  -2,   0 ] ]
gap> Display(Irr(t_Q8));
[ [   1,   1,   1,   1,   1 ],
  [   1,  -1,  -1,   1,   1 ],
  [   1,  -1,   1,   1,  -1 ],
  [   1,   1,  -1,   1,  -1 ],
  [   2,   0,   0,  -2,   0 ] ]
gap> TransformingPermutations(Irr(t_D8), Irr(t_Q8));
rec( columns := (), group := Group([ (3,5), (2,3) ]),
  rows := (2,3,4) )
```

A *Brauer pair* is a pair of non-isomorphic finite groups with the same character tables and power maps (up to permutation); we refer to [34, Definition 2.6.1] for a more precise definition. The notion is based on a question posed by R. Brauer in [4], where he asked whether such pairs exist. The first example of a Brauer pair goes back to E. Dade [13].

Example 3.26 (Dade's solution to Brauer's problem). Dade's paper shows the existence of Brauer pairs of groups of order p^7, for $p \geq 5$ a prime number. These groups have nilpotency class three and exponent p (with this, the condition on power maps is automatically satisfied once we know the groups have the same character table).

To check character tables up to permutations of rows and columns one uses the function `TransformingPermutations`. However, as the use of this function is quite expensive in our groups, we will try to find two non-isomorphic groups of order 5^7 of nilpotency class three and exponent five with *exactly* the same character table:

```
gap> p := 5;;
gap> m := 7;;
gap> l :=  AllSmallGroups(Size, p^m, Exponent, \
> p, NilpotencyClassOfGroup, 3);;
> tables := List(l, CharacterTable);;
> for c in Combinations([1..Size(l)], 2) do
>   x := tables[c[1]];
>   y := tables[c[2]];
>   if Irr(x) = Irr(y) then
>     Display(IdGroup(UnderlyingGroup(x)));
>     Display(IdGroup(UnderlyingGroup(y)));
>     break;
>   fi;
> od;
[ 78125, 19912 ]
[ 78125, 19913 ]
```

Example 3.27. In [15], B. Eick and J. Müller mention that E. Skrzipczyk found pairs of non-isomorphic groups of order 2^8 with the same character table and power maps. The following code uses Skrzipczyk's ideas in the case of groups of order 3^6:

```
gap> p := 3;
gap> m := 6;
gap> l := AllSmallGroups(Size, p^m, Exponent, p^2);;
gap> tables := List(l, CharacterTable);;
gap> for c in Combinations([1..Size(l)], 2) do
> x := tables[c[1]];
> y := tables[c[2]];
> if not NrConjugacyClasses(x) = NrConjugacyClasses(y) then
>   continue;
> fi;
> if not Set(CharacterDegrees(x)) = Set(CharacterDegrees(y)) then
>   continue;
> fi;
> m_y := Irr(y);
> PowerMap(y, p^m);
> if not \
> TransformingPermutationsCharacterTables(x, y) = fail then
>   Display([IdGroup(UnderlyingGroup(x)), \
>   IdGroup(UnderlyingGroup(y))]);
>   break;
> fi;
> od;
[ [  729,   44 ],
  [  729,   45 ] ]
```

3.3 Some conjectures

The theory of groups and their representations was developed around 200 years ago. Still, amazingly enough, many fundamental and easily formulated problems remain wide open. In this section, we will explore some of these questions using the computational tools discussed so far.

3.3.1 McKay's conjecture

For a finite group G, we write $\operatorname{Irr}(G)$ to denote the complete set of complex irreducible characters of G. For a prime number p dividing $|G|$, one defines

$$\operatorname{Irr}_{p'}(G) = \{\chi \in \operatorname{Irr}(G) : p \nmid \chi(1)\}.$$

J. McKay posed the following conjecture for simple groups and $p = 2$ in [38], and it was later generalized by M. Isaacs [25]:

Conjecture 3.1 (McKay). Let G be a finite group. If $P \in \operatorname{Syl}_p(G)$, then

$$|\operatorname{Irr}_{p'}(G)| = |\operatorname{Irr}_{p'}(N_G(P))|.$$

For $p = 2$, the conjecture was proved by G. Malle and B. Späth in [37]. For other primes, the conjecture is still open. We refer to [42] for the state-of-the-art.

Example 3.28 (Testing McKay's conjecture). Let us check Conjecture 3.1 in some small examples. We first write a naive function that checks the conjecture:

```
gap> McKay1 := function(G, p)
> local N, n, m;
> N := Normalizer(G, SylowSubgroup(G, p));
> n := Number(Irr(G), x->Degree(x) mod p <> 0);
> m := Number(Irr(N), x->Degree(x) mod p <> 0);
> return n = m;
> end;
function( G, p ) ... end
```

With this function, we can easily check the conjecture in several small examples, such as $\mathbf{SL}_2(3)$. This group has order 24, so we need to consider $p \in \{2, 3\}$:

```
gap> McKay1(SL(2,3), 2);
true
gap> McKay1(SL(2,3), 3);
true
```

The package `CTblLib` contains character tables of some normalizers of Sylow subgroups of sporadic simple groups. For example, to obtain the character table of the normalizer of the Sylow 2-subgroup of the Thompson sporadic group Th we proceed as follows:

```
gap> CharacterTable("ThN2");
CharacterTable( "ThN2" )
```

We can use this database of normalizers of Sylow subgroups to study some sporadic simple groups more efficiently:

```
gap> McKay2 := function(name)
> local t, t_N, p;
> t := CharacterTable(name);
> for p in PrimeDivisors(Size(t)) do
>   t_N := CharacterTable(Concatenation(name, "N", String(p)));
>   if t_N = fail then
>     return fail;
>   else
>     if not Number(Irr(t), x->Degree(x) mod p <> 0) = \
>     Number(Irr(t_N), x->Degree(x) mod p <> 0) then
>       return false;
>     fi;
>   fi;
> od;
> return true;
> end;
function( name ) ... end
```

We use this function to verify the conjecture for some sporadic groups:

```
gap> McKay2("Fi23");
true
gap> McKay2("Th");
true
```

However, not all character tables of normalizer of Sylow subgroups are stored:

```
gap> CharacterTable("Co1N2");
fail
```

To verify Conjecture 3.1 for the group Co_1 we proceed as follows. First, we consider $p = 2$ combining the character table of the group with the explicit calculation of the normalizer of the Sylow 2-subgroup:

```
gap> t := CharacterTable("Co1");;
gap> Number(Irr(t), x->Degree(x) mod 2 <> 0);
32
gap> Co1 := AtlasGroup("Co1");;
gap> P := SylowSubgroup(Co1, 2);;
gap> N := Normalizer(Co1, P);;
gap> Number(Irr(CharacterTable(N)), x->Degree(x) mod 2 <> 0);
32
```

For $p \in \{3, 5, 7, 11, 13, 23\}$, we use the information available in the library `CTblLib`:

```
gap> # McKay's conjecture for Co1 and odd primes
gap> t := CharacterTable("Co1");;
gap> for p in [3,5,7,11,13,23] do
> t_N := CharacterTable(Concatenation("Co1N", String(p)));
> n := Number(Irr(t_N), x->Degree(x) mod p <> 0);
> m := Number(Irr(t), x->Degree(x) mod p <> 0);
> Print("p = ", p, ": ", n = m, "\n");
> od;
p = 3: true
p = 5: true
p = 7: true
p = 11: true
p = 13: true
p = 23: true
```

In [55], R. Wilson proved that McKay's conjecture is true for the sporadic simple groups using a description of the normalizers of their Sylow subgroups.

3.3.2 Isaacs–Navarro conjecture

In [28], M. Isaacs and G. Navarro proposed a refinement of Conjecture 3.1. For a prime number p, an integer $k \in \mathbb{Z}$ coprime with p, and a finite group G, let

$$M_k(G) = |\{\chi \in \mathrm{Irr}_{p'}(G) : \chi(1) \equiv \pm k \bmod p\}|.$$

The following conjecture proposed by Isaacs and Navarro in [28] is still open and implies McKay's conjecture.

Conjecture 3.2 (Isaacs–Navarro). Let p be a prime number. Let G be a finite group and P be a Sylow p-subgroup of G. If $k \in \mathbb{Z}$ is coprime with p, then

$$M_k(G) = M_k(N_G(P)).$$

Example 3.29 (Testing Isaacs–Navarro conjecture). Here we have a naive function that checks Conjecture 3.2:

```
gap> IsaacsNavarro := function(G, k, p)
> local m, n, N;
> N := Normalizer(G, SylowSubgroup(G, p));
> m := Number(Filtered(Irr(G), x->Degree(x)\
> mod p <> 0), x->Degree(x) mod p in [-k, k] mod p);
> n := Number(Filtered(Irr(N), x->Degree(x)\
> mod p <> 0), x->Degree(x) mod p in [-k, k] mod p);
> return m = n;
> end;
function( G, k, p ) ... end
```

We consider the group $\mathbf{SL}_2(3)$. We need to verify the conjecture for $p \in \{2,3\}$ and $k \in \{1,2\}$.

```
gap> IsaacsNavarro(SL(2,3), 1, 2);
true
gap> IsaacsNavarro(SL(2,3), 1, 3);
true
gap> IsaacsNavarro(SL(2,3), 2, 3);
true
```

3.3.3 Ore's conjecture

In 1951, O. Ore and, independently, N. Ito proved that every element of an alternating simple group is a commutator. Ore also mentioned that "it is possible that a similar theorem holds for any simple group of finite order, but it seems that at present, we do not have the necessary methods to investigate the question". This gave rise to the following conjecture:

Conjecture 3.3 (Ore). Every element of a finite non-abelian simple group is a commutator.

The conjecture is now a theorem thanks to the work of M. Liebeck, E. O'Brien, A. Shalev and P. Tiep [32]. See [36] for a survey on Ore's conjecture.

Example 3.30 (Testing Ore's conjecture). We will prove Ore's conjecture for some sporadic simple groups. Let G be a finite simple group. It is known that $g \in G$ is a commutator if and only if

$$\sum_{\chi \in \mathrm{Irr}(G)} \frac{\chi(g)}{\chi(1)} \neq 0.$$

Let us write a computer script to check whether every element in a group is a commutator. Our function needs the character table of a group and returns **`true`** if every element of the group is a commutator and **`false`** otherwise:

```
gap> Ore := function(t)
> local s, f, k;
> for k in [1..NrConjugacyClasses(t)] do
>   s := 0;
>   for f in Irr(t) do
>     s := s + f[k]/Degree(f);
>   od;
>   if s <= 0 then
>     return false;
>   fi;
> od;
> return true;
> end;
function( t ) ... end
```

Now, for example, we check the conjecture for the first Janko group, the Mathieu simple groups and the Monster group:

```
gap> Ore(CharacterTable("J1"));
true
gap> Ore(CharacterTable("M11"));
true
gap> Ore(CharacterTable("M12"));
true
gap> Ore(CharacterTable("M22"));
true
gap> Ore(CharacterTable("M23"));
true
gap> Ore(CharacterTable("M24"));
true
gap> Ore(CharacterTable("M"));
true
```

3.3.4 Thompson's conjecture

In [1], Z. Arad and M. Herzog list several conjectures on products of conjugacy classes in finite non-abelian simple groups. The following conjecture is attributed to J. G. Thompson:

Conjecture 3.4 (Thompson). Let G be a finite non-abelian simple group. Then there exists a conjugacy class C of G such that $C^2 = G$, where $C^2 = \{xy : x, y \in C\}$.

The conjecture is still open.

Example 3.31 (Testing Thompson's conjecture). We first present a function that checks the conjecture in a very naive way:

```
gap> Thompson := function(G)
> local squareSet;
> # Return the square of a set
> squareSet := function(s)
>   return Set(Cartesian(s, s), x->x[1]*x[2]);
> end;
> return ForAny(ConjugacyClasses(G), \
> x->Size(squareSet(x)) = Order(G));
> end
function( G ) ... end
```

Now we test the conjecture in some small simple groups:

```
gap> Thompson(AlternatingGroup(5));
true
gap> Thompson(MathieuGroup(11));
true
gap> Thompson(PSL(2,7));
true
```

On the other hand, Thompson's conjecture is about (non-abelian) simple groups, and it may not hold for arbitrary groups:

```
gap> Thompson(SymmetricGroup(4));
false
gap> Thompson(SymmetricGroup(5));
false
gap> Thompson(SL(2,7));
false
```

Note that if C is a conjugacy class of G such that $C^2 = G$, then C is a *real conjugacy class*, which means that $C = C^{-1} = \{x^{-1} : x \in C\}$. Equivalently, C is real if and only if $\chi(x) \in \mathbb{R}$ for $x \in C$. To check if a given conjugacy class is real, we can use the following code:

```
gap> IsReal := function(C)
> return Inverse(Random(C)) in C;
> end;
function( C ) ... end
gap> A4 := AlternatingGroup(4);;
gap> cc := ConjugacyClasses(A4);
[ ()^G, (1,2)(3,4)^G, (1,2,3)^G, (1,2,4)^G ]
gap> List(cc, IsReal);
[ true, true, false, false ]
```

The use of real conjugacy classes could make our previous function slightly better. In [36], G. Malle mentions an even better technique to check Thompson's conjecture: If G is a finite group and C is a real conjugacy class, then $C^2 = G$ if and only if, for some fixed $x \in C$,

$$\sum_{\chi \in \mathrm{Irr}(G)} \frac{|\chi(x)|^2 \chi(g)}{\chi(1)} \neq 0$$

holds for all $g \in G$. See [34, Exercise 2.6.3].

Real conjugacy classes can be obtained from a character table using the function `RealClasses` of the `CTblLib` package:

```
gap> t := CharacterTable("A4");;
gap> NrConjugacyClasses(t);
4
gap> OrdersClassRepresentatives(t);
[ 1, 2, 3, 3 ]
gap> RealClasses(t);
[ 1, 2 ]
```

We now present another function that verifies Thompson's conjecture:

```
gap> Thompson := function(t)
> local k, C;
> k := NrConjugacyClasses(t);
> for C in RealClasses(t) do
>   if ForAll([1..k], D->Sum(Irr(t), \
>   f->(f[C]^2*f[D])/Degree(f)) <> 0) then
>     return true;
>   fi;
> od;
> return false;
> end;
function( t ) ... end
gap> Thompson(CharacterTable("M"));
true
gap> Thompson(CharacterTable("A10"));
true
gap> Thompson(CharacterTable("Th"));
true
gap> Thompson(CharacterTable("S10"));
false
```

3.3.5 Szep's conjecture

In [52, 53], J. Szep formulated the following conjecture:

Conjecture 3.5 (Szep). Let G be a finite non-abelian simple group and $x, y \in G\setminus\{1\}$. Then $G \neq C_G(x)C_G(y)$.

The conjecture, originally formulated is now a theorem. It was proved by E. Fisman and Z. Arad in 1987 using the classification of finite simple groups. We refer to [21] for more information.

```
gap> Szep := function(G)
> local x, y, C, D, CxD;
> for x in G do
>   if IsOne(x) then
>     continue;
>   fi;
>   for y in G do
>     if IsOne(y) or x = y then
>       continue;
>     fi;
>     C := Centralizer(G, x);
>     D := Centralizer(G, y);
>     CxD := Cartesian(C, D);
>     if Size(Set(CxD, z->z[1]*z[2])) = Order(G) then
>       return false;
>     fi;
>   od;
> od;
> return true;
> end;
function( G ) ... end
```

With this naive function, we can verify the conjecture in small examples:

```
gap> Szep(AlternatingGroup(5));
true
gap> Szep(PSL(2,5));
true
gap> Szep(PSL(2,7));
true
```

Conjecture 3.5 does not hold for non-simple groups:

```
gap> Szep(AlternatingGroup(4));
false
gap> Szep(DihedralGroup(34));
false
```

3.3.6 Arad–Herzog conjecture

In [1], Z. Arad and M. Herzog proposed the following conjecture:

Conjecture 3.6 (Arad–Herzog). If G is a non-abelian simple group, then the product of two non-trivial conjugacy classes of G is never a single conjugacy class.

The conjecture is still open; we refer the reader to [21] for more information.

Example 3.32 (Testing Arad–Herzog conjecture). We first write a naive function to test Conjecture 3.6:

```
gap> AradHerzog1 := function(G)
> local cc, c, d, g, s;
> # We only consider non-trivial conjugacy classes
> cc := Filtered(ConjugacyClasses(G), x->Size(x) > 1);
> for c in cc do
>   for d in cc do
>     s := Set(Cartesian(c, d), x->x[1]*x[2]);
>     g := Random(s);
>     if Size(s) = Size(ConjugacyClass(G, g)) then
>       return false;
>     fi;
>   od;
> od;
> return true;
> end;
function( G ) ... end
```

We now study some simple groups:

```
gap> AradHerzog1(AlternatingGroup(5));
true
gap> AradHerzog1(AlternatingGroup(6));
true
```

Conjecture 3.6 does not hold in general for non-simple groups:

```
gap> AradHerzog1(DihedralGroup(6));
false
gap> AradHerzog1(QuaternionGroup(32));
false
```

Note that conjugacy classes are trivial in abelian groups, so the conjecture vacuously holds for this class of groups:

```
gap> AradHerzog1(AbelianGroup([8, 4]));
true
```

We will now write a better version of the function to test if the conjecture holds:

```
gap> AradHerzog2 := function(G)
> local classes, c, g, h, s
> # We only consider non-trivial conjugacy classes
> classes := Filtered(ConjugacyClasses(G), x->Size(x)>1);
> for c in classes do
>   for g in List(classes, Representative) do
>     s := Set(c, x->x*g);
>     h := Random(s);
>     if ForAll(s, x->IsConjugate(G, x, h)) then
>       return false;
>     fi;
>   od;
> od;
> return true;
> end;
function( G ) ... end
```

Now we can prove the conjecture holds in other simple groups, such as $\mathrm{Sz}(8)$, $\mathbf{PSL}_2(8)$, $\mathbf{PSL}_3(5)$ and J_1:

```
gap> AradHerzog2(Sz(IsPermGroup, 8));
true
gap> AradHerzog2(PSL(2,8));
true
gap> AradHerzog2(PSL(3,5));
true
gap> AradHerzog2(AtlasGroup("J1"));
true
```

Finally, we present a different function to check the conjecture by only using character tables. The method is based on Lemma 2.2 of [21], which gives a sufficient condition to check whether the conjecture is true:

```
gap> AradHerzog3 := function(t)
> local c, i, k;
> k := NrConjugacyClasses(t);
> # We only consider non-trivial conjugacy classes
> c := Filtered([1..k], x->SizesConjugacyClasses(t)[x] > 1);
> for i in IteratorOfTuples(c, 3) do
>   if ForAll(Irr(t), \
>     x->x[i[1]]*x[i[2]] = x[i[3]]*Degree(x)) then
>     return;
>   fi;
> od;
> return true;
>end;
function( t ) ... end
```

This function returns `true` if the conjecture holds. However, when the function does not return `true`, one cannot conclude that the conjecture is false.

We prove the conjecture for some sporadic simple groups:

```
gap> AradHerzog3(CharacterTable("J1"));
true
gap> AradHerzog3(CharacterTable("Co1"));
true
gap> AradHerzog3(CharacterTable("B"));
true
gap> AradHerzog3(CharacterTable("M"));
true
```

3.3.7 Hughes' conjecture

We now discuss Hughes' conjecture. For a finite group G and a prime number p, let $H_p(G)$ be the subgroup of G generated by all elements of order $\neq p$. We first construct the Hughes subgroup:

```
gap> HughesSubgroup := function(p, G)
> return Subgroup(G, Filtered(G, g->not Order(g) = p));
> end;
function( p, G ) ... end
```

Hughes made the following conjecture on the order of $H_p(G)$:

Conjecture 3.7 (Hughes). If G is a finite group and p is a prime number, then

$$(G : H_p(G)) \in \{1, p, |G|\}.$$

Let us test some examples:

```
gap> StructureDescription(HughesSubgroup(2, SymmetricGroup(4)));
"S4"
gap> StructureDescription(HughesSubgroup(2, DihedralGroup(64)));
"C32"
gap> H := AlternatingGroup(5);;
gap> K := PGL(IsPermGroup, 2, 7);;
gap> G := DirectProduct(H, K);;
gap> Order(G);
20160
gap> Order(HughesSubgroup(3, G));
20160
```

The conjecture is known to be true if $p \in \{2, 3\}$ or if the group is not a p-group [24]. There are counterexamples for $p \in \{5, 7, 11, 13, 19\}$; see [23]. Experts suspect that there are counterexamples for every $p \geq 5$.

3.3.8 Harada's conjecture

In [22], K. Harada made the following conjecture:

Conjecture 3.8 (Harada). Let G be a finite group, $\chi_1, \dots, \chi_s$ be representatives of irreducible characters of G and $K_1, \dots, K_s$ be the conjugacy classes of G. Then

$$\prod_{j=1}^{s} \chi_j(1) \quad \text{divides} \quad \prod_{j=1}^{s} |K_j|.$$

The conjecture is wide open; see [43].

Example 3.33 (Testing Harada's conjecture). We first show a function that checks whether the conjecture is true:

```
gap> Harada1 := function(G)
> local m, n;
> m := Product(ConjugacyClasses(G), Size);
> n := Product(Irr(G), Degree);
> return IsInt(m/n);
> end;
function( G ) ... end
```

We can verify Harada's conjecture for all groups of order ≤ 60 in one line:

```
gap> AllGroups(Size, [1..60], IsAbelian, false, Harada1, false);
[  ]
```

As a different application, we will prove that Harada's conjecture is true for all sporadic simple groups:

```
gap> Harada2 := function(T)
> local m, n;
> m := Product(SizesConjugacyClasses(T));
> n := Product(Irr(T), Degree);
> return IsInt(m/n);
> end;
function( T ) ... end
```

We will verify the conjecture for the 2671 character tables included in the `CTblLib` library, which includes all the sporadic simple groups:

```
gap> tables := AllCharacterTableNames();;
gap> Size(tables);
2671
gap> ForAll(tables, T->Harada2(CharacterTable(T)));
true
```

3.3.9 Berkovich's conjecture

In 1973, Y. Berkovich posed the following conjecture:

Conjecture 3.9 (Berkovich). Let P be a finite non-abelian p-group. Then $\mathrm{Aut}(P)$ contains a non-inner automorphism of order p.

The existence of a non-inner automorphism of order p^s, for some $s \geq 1$, had been previously established by W. Gaschütz [18].

Example 3.34 (Testing Berkovich's Conjecture 3.9). The following function test whether a given p-group satisfies Berkovich's conjecture.

```
gap> Berkovich := function(P)
> local p, A, I, S;
> if Order(P) = 1 or not IsPGroup(P) then
>   return fail;
> fi;
> p := PrimeDivisors(Order(P))[1];
> A := AutomorphismGroup(P);
> I := InnerAutomorphismsAutomorphismGroup(A);
> S := SylowSubgroup(A, p);
> return ForAny(Set(S), x->Order(x) = p and not x in I);
> end;;
```

We test the conjecture in some cases:

```
gap> P := Random(AllGroups(Size, 2^3, IsAbelian, false));;
gap> Berkovich(P);
true
gap> P := Random(AllGroups(Size, 5^5, IsAbelian, false));;
gap> Berkovich(P);
true
gap> P := Random(AllGroups(Size, 7^3, IsAbelian, false));;
gap> Berkovich(P);
true
```

3.3.10 Wall's conjecture

In [54], G. E. Wall made the following conjecture:

Conjecture 3.10 (Wall). Let G be a finite group. Then G has at most $|G|$ maximal subgroups.

The conjecture is still open. Wall proved Conjecture 3.10 for solvable groups. M. Liebeck, L. Pyber and A. Shalev proved the conjecture for infinitely many simple groups; see [33].

Example 3.35 (Testing Wall's conjecture). The following function checks if Conjecture 3.10 holds for a given group:

```
gap> Wall1 := function(G)
> local m;
> m := ConjugacyClassesMaximalSubgroups(G);
> return Sum(m, Size) <= Order(G);
> end;
function( G ) ... end
```

Let us verify the conjecture for all groups of order ≤ 60:

```
gap> AllGroups(Size, [1..60], Wall1, false);
[  ]
```

Using the information on character tables, we can quickly verify Wall's conjecture for some sporadic simple groups:

```
gap> Wall2 := function(T)
> return Sum(Maxes(T), \
> x->Size(T)/Size(CharacterTable(x))) <= Size(T);
> end;
function( T ) ... end
```

For example, the conjecture holds for Conway groups, Mathieu groups and the Baby Monster:

```
gap> Wall2(CharacterTable("Co1"));
true
gap> Wall2(CharacterTable("Co2"));
true
gap> Wall2(CharacterTable("Co3"));
true
gap> Wall2(CharacterTable("M11"));
true
gap> Wall2(CharacterTable("M12"));
true
gap> Wall2(CharacterTable("M22"));
true
gap> Wall2(CharacterTable("M23"));
true
gap> Wall2(CharacterTable("M24"));
true
gap> Wall2(CharacterTable("B"));
true
```

3.3.11 Quillen's conjecture

If G is a finite group and p is a prime number, then the p-core of G, denoted by $O_p(G)$, is the largest normal p-subgroup of G.

Example 3.36. Let us start with some examples of the calculation of p-cores:

```
gap> G := DihedralGroup(100);;
gap> StructureDescription(PCore(G, 2));
"C2"
gap> StructureDescription(PCore(G, 3));
"1"
gap> StructureDescription(PCore(G, 5));
"C25"
```

Example 3.37. Let us verify that the 2-core of the symmetric group $\mathbb{S}_4$ is equal to the intersection

$$O_2(P) = \bigcap_{P\in\mathrm{Syl}_2(G)} P$$

of the Sylow 2-subgroups:

```
gap> G := SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> P := SylowSubgroup(G, 2);
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> C := ConjugacyClassSubgroups(G, P);
Group( [ (1,2), (3,4), (1,3)(2,4) ] )^G
gap> Intersection(AsList(C)) = PCore(G, 2);
true
```

The p-rank of a finite group G is the maximal dimension (as an $\mathbb{F}_p$-vector space) of an elementary abelian p-subgroup.

Example 3.38. We show that

$$m_2(\mathbf{PSL}_3(2)) = 2, \quad m_2(\mathbf{PSL}_3(4)) = 4, \quad m_{11}(\mathbf{PSL}_3(2)) = 0, \quad m_2(\mathbb{S}_{10}) = 5.$$

To compute the p-rank of a finite group G, it is enough to consider a Sylow p-subgroup P of G. Since P is solvable, we can use `SubgroupsSolvableGroup`; see Section 2.5. The following function computes the p-rank of a finite group:

```
gap> PRank := function(G, p)
> local P, f;
> P := SylowSubgroup(G, p);
> f := Filtered(SubgroupsSolvableGroup(P), IsElementaryAbelian);
> return Maximum(List(f, x->LogInt(Size(x), p)));
> end;
function( G, p ) ... end
```

Let's see some examples:

```
gap> PRank(PSL(3,2), 2);
2
gap> PRank(PSL(3,4), 2);
4
gap> PRank(PSL(3,2), 11);
0
gap> PRank(SymmetricGroup(10), 2);
5
gap> PRank(SymmetricGroup(13), 2);
6
```

A conjecture raised by D. Quillen in 1978 states that for a finite group G and a prime number p, if its complex of p-subgroups is contractible, then the p-core of G is trivial; see [47]. A stronger version of the conjecture can be formulated using the Euler characteristic as follows:

Conjecture 3.11 (Strong Quillen's conjecture). If G is a finite group, p is a prime number and $O_p(G) = 1$, then

$$\sum_{A \in C_p} (-1)^{m_p(A)} p^{\binom{m_p(A)}{2}} (G : N_G(A)) \neq 0,$$

where C_p is a set of representatives of conjugacy classes of elementary abelian p-subgroups of G.

While the original statement of the conjecture was proved for solvable groups by Quillen, the version mentioned in Conjecture 3.11 is still open even for solvable groups.

Example 3.39 (Testing Quillen's conjecture). We write a naive function to test Quillen's conjecture:

```
gap> Quillen := function(G, p)
> local P, A, f, C;
> if not IsTrivial(PCore(G, p)) then
>   return true;
> fi;
> P := SylowSubgroup(G, p);
> f := Filtered(SubgroupsSolvableGroup(P), IsElementaryAbelian);
> C := [];
> for A in f do
>   if ForAll(C, x->not IsConjugate(G, A, x)) then
>     Add(C, A);
>   fi;
> od;
> return Sum(C, x->(-1)^(PRank(x, p))\
> *p^(Binomial(PRank(x, p), 2))*Index(G, Normalizer(G, x))) <> 0;
> end;
```

Now we test Conjecture 3.11 on some groups:

```
gap> Quillen(PSL(3,4), 3);
true
gap> Quillen(PSL(3,4), 2);
true
gap> Quillen(SymmetricGroup(13), 5);
true
gap> Quillen(SymmetricGroup(13), 3);
true
gap> Quillen(MathieuGroup(24), 3);
true
gap> Quillen(MathieuGroup(22), 2);
true
```

Example 3.40. We verify Quillen's Conjecture 3.11 for all groups of order ≤ 60. To make our calculations faster, we use the fact the conjecture is trivially true in the case of p-groups:

```
gap> f := Filtered(AllGroups(Size, [1..60]), \
> x->not IsPrimePowerInt(Order(x)));;
gap> for G in f do
> for p in PrimeDivisors(Order(G)) do
>   if not Quillen(G, p) then
>     Print("False for the group ", IdGroup(G), "!\n");
>   fi;
> od;
> od;
```

This calculation produces no output, meaning that Conjecture 3.11 is true for groups of order ≤ 60.

3.4 Group rings

Let K be a field and G be a (not necessarily finite) group. The group ring $A = K[G]$ is the K-algebra with basis $\{e_g : g \in G\}$ and multiplication induced by

$$e_g e_h = e_{gh} \quad \text{for } g, h \in G.$$

Every element of A is a finite sum of the form

$$\alpha = \sum_{g \in G} \lambda_g e_g$$

for scalars $\lambda_g \in K$.

The *augmentation ideal* $I(G)$ of A is the kernel of the map $A \to K$, $e_g \mapsto 1$. Thus

$$I(A) = \left\{ \sum_{g \in G} \lambda_g e_g \in A : \sum_{g \in G} \lambda_g = 0 \right\}.$$

Example 3.41. Let K be the field of two elements, $G = \mathbb{S}_3$ and $A = K[G]$. For example, $e_{(2\,3)} + e_{(1\,2)} + e_{(1\,2\,3)} + e_{(1\,3)} \in A$. Moreover, A is a non-commutative algebra of dimension six:

```
gap> G := SymmetricGroup(3);;
gap> A := GroupRing(GF(2), G);;
gap> IsGroupAlgebra(A);
true
gap> Dimension(A);
6
gap> One(A);
(Z(2)^0)*()
gap> Zero(A);
<zero> of ...
gap> Random(A);
(Z(2)^0)*(2,3)+(Z(2)^0)*(1,2)+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3)
```

We compute the augmentation ideal:

```
gap> I := AugmentationIdeal(A);;
gap> Dimension(I);
5
gap> A/I;
<algebra of dimension 1 over GF(2)>
```

We now perform some easy calculations. Let

$$\alpha = e_{(1\,2)} + e_{(1\,3)} \in A, \quad \beta = e_{(1\,2)} + e_{(1\,2\,3)} + e_{(2\,3)} \in A.$$

We compute $\alpha + \beta$ and $\alpha\beta$:

$$\begin{aligned}\alpha+\beta &= e_{(1\,3)}+e_{(1\,2\,3)}+e_{(2\,3)},\\ \alpha\beta &= e_{\mathrm{id}}+e_{(2\,3)}+e_{(1\,2\,3)}+e_{(1\,3)}.\end{aligned}$$

To perform calculations, we need the canonical embedding $G \longrightarrow A$:

```
gap> f := Embedding(G, A);;
gap> a := (1,2)^f+(1,3)^f;
(Z(2)^0)*(1,2)+(Z(2)^0)*(1,3)
gap> b := (1,2)^f+(1,2,3)^f+(2,3)^f;;
gap> a+b;
(Z(2)^0)*(2,3)+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3)
gap> a*b;
(Z(2)^0)*()+(Z(2)^0)*(2,3)+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3)
```

Note that, for example, α is not invertible:

```
gap> Inverse(a);
fail
gap> IsUnit(a);
false
```

We compute the group of invertible elements. It is generated by

$$e_{(1\,3)},\quad e_{(1\,3\,2)},\quad e_{(2\,3)}+e_{(1\,2)}+e_{(1\,2\,3)}+e_{(1\,3\,2)}+e_{(1\,3)},$$

and it is isomorphic to the dihedral group of twelve elements:

```
gap> U := Units(A);
<group of size 12 with 3 generators>
gap> Order(U);
12
gap> StructureDescription(U);
"D12"
gap> GeneratorsOfGroup(U);
[ (Z(2)^0)*(1,3), (Z(2)^0)*(1,3,2),
  (Z(2)^0)*(2,3)+(Z(2)^0)*(1,2)+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2)
    +(Z(2)^0)*(1,3) ]
```

Example 3.42. The center of $\mathbb{Q}[\mathbb{S}_3]$ has basis

$$\{e_{\mathrm{id}}, e_{(1\,2)}+e_{(1\,3)}+e_{(2\,3)}, e_{(1\,2\,3)}+e_{(1\,3\,2)}\}.$$

We verify this with the following code:

```
gap> G := SymmetricGroup(3);;
gap> A := GroupRing(Rationals, G);;
gap> Center(A);
<algebra-with-one of dimension 3 over Rationals>
gap> Elements(Basis(Center(A)));
[ (1)*(), (1)*(2,3)+(1)*(1,2)+(1)*(1,3),
  (1)*(1,2,3)+(1)*(1,3,2) ]
```

An element α of a ring is said to be *idempotent* if $\alpha^2 = \alpha$.

Example 3.43. Let K be the field of two elements, and $G = \langle g \rangle$ be the cyclic group of order three with generator $g = (1\,2\,3)$. The idempotents of $K[G]$ are 0, e_1, e_g, e_{g^2} and $e_g + e_{g^2}$.

```
gap> G := Group([(1,2,3)]);;
gap> A := GroupRing(GF(2), G);;
gap> Idempotents(A);
[ <zero> of ..., (Z(2)^0)*(),
  (Z(2)^0)*()+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2),
  (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ]
```

Let R be a unitary ring. The *Jacobson radical* $J(R)$ of R is defined as the intersection of all left maximal ideals of R. One proves that $J(R)$ is indeed an ideal of R and that $x \in J(R)$ if and only if $1 + rx$ is invertible for all $r \in R$.

Example 3.44. Let K be the field of two elements, $G = \mathbb{D}_8$ be the dihedral group of order eight and $A = K[G]$. To have a nicer computer presentation of the elements of $\mathbb{D}_8$ (and hence of A), we turn on the teaching mode:

```
gap> TeachingMode(true);
#I  Teaching mode is turned ON
```

We now compute the Jacobson radical $J(A)$. It has dimension seven and a basis is given by

$$\{e_1 + e_{r^{-1}}, e_{r^{-1}} + e_r, e_r + e_s, e_s + e_{r^2}, e_{r^2} + e_{rs}, e_{rs} + e_{sr}, e_{sr} + e_{sr^2}\}.$$

Here is the code:

```
gap> G := DihedralGroup(8);;
gap> K := GF(2);;
gap> A := GroupRing(K, G);;
gap> J := RadicalOfAlgebra(A);;
gap> Dimension(J);
7
gap> Elements(Basis(J));
gap> Elements(Basis(J));
[ (Z(2)^0)*<identity ...>+(Z(2)^0)*r^-1,
  (Z(2)^0)*r^-1+(Z(2)^0)*r,
  (Z(2)^0)*r+(Z(2)^0)*s, (Z(2)^0)*s+(Z(2)^0)*r^2,
  (Z(2)^0)*r^2+(Z(2)^0)*r*s, (Z(2)^0)*r*s+(Z(2)^0)*s*r,
  (Z(2)^0)*s*r+(Z(2)^0)*s*r^2 ]
```

We also check that, for example, $e_{sr} + e_{sr^2}$ is a nilpotent element:

```
gap> f := Embedding(G, A);;
gap> GeneratorsOfGroup(G);
[ r, s ]
gap> AssignGeneratorVariables(G);
#I  Assigned the global variables [ r, s ]
```

```
gap> R := r^f;;
gap> S := s^f;;
gap> a := (Z(2)^0)*R^2+(Z(2)^0)*R*S;;
gap> IsNilpotentElement(A, a);
true
gap> a in J;
true
gap> IsZero(a^2);
true
```

Example 3.45. Let K be the field of three elements, $G = \mathbb{S}_4$ and $A = K[G]$. We compute the Jacobson radical $J(A)$ and check that the elements of the form $1 + x$ for $x \in J(A)$ are units:

```
gap> G := SymmetricGroup(4);;
gap> A := GroupRing(GF(3), G);;
gap> J := RadicalOfAlgebra(A);
<algebra of dimension 4 over GF(3)>
gap> Size(J);
81
gap> ForAll(J, x->IsUnit(One(A)+x));
true
```

Let K be a field, G be a group and $A = K[G]$. By definition, the *trivial units* of A are the elements of the form λe_g for $\lambda \in K \setminus \{0\}$ and $g \in G$.

Example 3.46. Let K be the field of two elements and G be the cyclic group of order two. The group ring $K[G]$ has only trivial units:

```
gap> G := CyclicGroup(IsPermGroup, 2);;
gap> A := GroupRing(GF(2), G);;
gap> Units(A);
<group with 1 generator>
gap> Elements(Units(A));
[ (Z(2)^0)*(), (Z(2)^0)*(1,2) ]
```

Recall that $\mathbb{Q}(i) = \{x + yi : x, y \in \mathbb{Q}\}$ is the field of *Gaussian rationals*. Some examples:

```
gap> E(4) in GaussianRationals;
true
gap> CF(4) = GaussianRationals;
true
gap> Random(GaussianRationals);
4/3+1/2*E(4)
```

In the following example, we show different Wedderburn decompositions. For that purpose, we use the package `Wedderga`:

```
gap> LoadPackage("Wedderga");
```

Example 3.47. Let G be the cyclic group of order four. Let us compute the Wedderburn decomposition of $\mathbb{Q}[G]$. We verify that $\mathbb{Q}[G] \simeq \mathbb{Q} \times \mathbb{Q} \times \mathbb{Q}(i)$.

```
gap> G := CyclicGroup(4);;
gap> A := GroupRing(Rationals, G);;
gap> WedderburnDecompositionInfo(A);
[ [ 1, Rationals ], [ 1, GaussianRationals ], [ 1, Rationals ] ]
```

The smallest algebraic number field for G that is also a splitting field is $\mathbb{Q}(i)$. In particular, the following code shows that, if $K = \mathbb{Q}(i)$, then $K[G] \simeq K^4$:

```
gap> B := GroupRing(CF(4), G);;
gap> WedderburnDecompositionInfo(B);
[ [ 1, GaussianRationals ],
  [ 1, GaussianRationals ],
  [ 1, GaussianRationals ],
  [ 1, GaussianRationals ] ]
```

Since the field $\mathbb{F}_5$ of five elements contains a primitive 4-th root of one, it is a splitting field for G. In particular, $\mathbb{F}_5[G] \simeq (\mathbb{F}_5)^4$:

```
gap> A := GroupRing(GF(5), G);;
gap> WedderburnDecompositionInfo(A);
[ [ 1, 5 ], [ 1, 5 ], [ 1, 5 ], [ 1, 5 ] ]
```

Example 3.48. Let $G = \mathbb{D}_{10}$ be the dihedral group of ten elements. We show that $\mathbb{Q}[G] \simeq \mathbb{Q} \times \mathbb{Q} \times M_2(\mathbb{Q}(\xi + \xi^{-1}))$, where ξ is a primitive 5-th root of one:

```
gap> G := DihedralGroup(10);;
gap> K := Rationals;;
gap> A := GroupRing(K, G);;
gap> WedderburnDecompositionInfo(A);
[ [ 1, Rationals ], [ 1, Rationals ], [ 2, NF(5,[ 1, 4 ]) ] ]
gap> Dimension(NF(5,[ 1, 4 ]));
2
```

Note that `NF(5,[ 1, 4 ])` is the smallest field extension of $\mathbb{Q}$ that contains the element $\xi + \xi^{-1}$:

```
gap> E(5)+E(5)^(-1) in NF(5,[ 1, 4 ]);
true
gap> Field(E(5)+E(5)^(-1));
NF(5,[ 1, 4 ])
```

A *Lie algebra* is a vector space L with a bilinear map $L \times L \to L$, $(x, y) \mapsto [x, y]$, such that

$$[x, x] = 0 \qquad \text{for all } x \in L, \tag{3.1}$$

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0 \qquad \text{for all } x, y, z \in L. \tag{3.2}$$

Equality (3.2) is known as the *Jacobi identity*.

A Lie algebra L is semisimple if and only if the only abelian ideal of L is zero. Equivalently, L is semisimple if and only if the solvable radical of L is zero. The following function detects semisimplicity:

```
gap> IsSemiSimple := function(L)
> return Dimension(LieSolvableRadical(L)) = 0;
> end;
function( L ) ... end
```

W. Plesken suggested the study of a certain Lie subalgebra of the group ring with the classical Lie bracket; see [10]. Let K be a field, G be a finite group and $A = K[G]$. For each $g \in G$, let $\eta_g = g - g^{-1}$. The *Plesken Lie algebra* of G is the vector space generated by $\{\eta_g : g \in G\}$ with Lie bracket given by

$$[\eta_g, \eta_h] = \eta_g\eta_h - \eta_h\eta_g, \quad g, h \in G.$$

We use the package `Laguna` to compute the Plesken Lie algebra of a finite group:

```
gap> Plesken := function(K, G)
> local L, e, f;
> L := LieAlgebra(GroupRing(K, G));
> f := Embedding(G, L);
> e := g->g^f-Inverse(g)^f;
> return Subalgebra(L, List(G, e));
> end;
function( K, G ) ... end
```

Example 3.49. The Plesken Lie algebra of the alternating group $\mathbb{A}_4$ over the field of two elements has basis

$$\{\eta_{(2\,3\,4)} + \eta_{(2\,4\,3)}, \eta_{(1\,3\,4)} + \eta_{(1\,4\,3)}, \eta_{(1\,2\,4)} + \eta_{(1\,4\,2)}, \eta_{(1\,2\,3)} + \eta_{(1\,3\,2)}\}.$$

and is not semisimple:

```
gap> G := AlternatingGroup(4);;
gap> L := Plesken(GF(2), G);;
#I  LAGUNA package: Constructing Lie algebra ...
gap> Dimension(L);
4
gap> BasisVectors(Basis(L));
[ LieObject( (Z(2)^0)*(2,3,4)+(Z(2)^0)*(2,4,3) ),
  LieObject( (Z(2)^0)*(1,3,4)+(Z(2)^0)*(1,4,3) ),
  LieObject( (Z(2)^0)*(1,2,4)+(Z(2)^0)*(1,4,2) ),
  LieObject( (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ) ]
gap> Dimension(LieSolvableRadical(L));
4
```

Example 3.50. For the symmetric group $\mathbb{S}_3$, the Plasken Lie algebra is not semisimple:

```
gap> L := Plesken(Rationals, SymmetricGroup(3));;
```

```
#I  LAGUNA package: Constructing Lie algebra ...
gap> Dimension(LieSolvableRadical(L));
1
```

Example 3.51. We now check that the Plesken Lie algebra of Q_8 is semisimple and isomorphic to $\mathfrak{sl}_2$:

```
gap> G := QuaternionGroup(8);;
gap> L := Plesken(Rationals, G);
#I  LAGUNA package: Constructing Lie algebra ...
<Lie algebra over Rationals, with 8 generators>
gap> Dimension(L);
3
gap> LieSolvableRadical(L);
<Lie algebra of dimension 0 over Rationals>
gap> SemiSimpleType(L);
"A1"
```

Example 3.52. We verify that the Plesken Lie algebra of the alternating group $\mathbb{A}_5$ is semisimple:

```
gap> G := AlternatingGroup(5);;
gap> L := Plesken(Rationals, G);;
#I  LAGUNA package: Constructing Lie algebra ...
gap> Dimension(L);
22
gap> IsSemisimple(L);
true
gap> SemiSimpleType(L);
"B2 A1 A1 A1 A1"
```

A theorem of A. M. Cohen and D. E. Taylor characterizes semisimplicity of complex Plesken Lie algebras [10].

3.5 Kaplansky’s unit conjecture

This section discusses the existence of a counterexample for the following well-known problem.

Conjecture 3.12 (Kaplansky). Let G be a torsion-free group and K a field. Then all units of $K[G]$ are trivial.

It is known that groups satisfying the unique product property satisfy Kaplanksy’s unit conjecture. In [17], it was shown by G. Gardam that the Promislow group, which fails the unique product property, also fails Kaplanksy’s unit conjecture over the finite field of two elements. Nevertheless, the unit problem is still open for fields of characteristic zero.

In this section, we work with the Promislow group P and show that it fails the unique product property. We also present Gardam's theorem as a computer calculation.

Recall that a group G has the *unique product property* if for all finite non-empty subsets A and B of G there exists $x \in G$ that can be written uniquely as $x = ab$ with $a \in A$ and $b \in B$.

Left-ordered groups satisfy the unique product property. However, the converse does not hold: unique product groups are not necessarily left-ordered. The first example exhibiting this phenomenon appears in [29].

In the following example, we show that the Promislow group P does not have the unique product property; see [46].

Example 3.53. The Promislow group

$$P = \langle a, b \mid a^{-1}b^2a = b^{-2},\ b^{-1}a^2b = a^{-2}\rangle$$

does not have the unique product property. Let

$$S = \{a^2b, b^2a, aba^{-1}, (b^2a)^{-1}, (ab)^{-2}, b, (ab)^2a, (ab)^2, (aba)^{-1}, \\ bab, b^{-1}, a, aba, a^{-1}\}. \quad (3.3)$$

We use the representation $G \to \mathbf{GL}_4(\mathbb{Q})$ given by

$$a \mapsto \begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad b \mapsto \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \end{pmatrix}$$

to check that G does not have unique product property, as each

$$s \in S^2 = \{s_1s_2 : s_1, s_2 \in S\}$$

admits at least two different decompositions of the form $s = xy = uv$ for $x, y, u, v \in S$. We first create the matrix representations of a and b.

```
gap> a := [[0,1,0,0], [2,0,0,0], [0,0,0,1/2], [0,0,1,0]];;
gap> b := [[0,0,1,0], [0,0,0,1], [2,0,0,0], [0,1/2,0,0]];;
```

Now we create a function that produces the set S.

```
gap> Promislow := function(x, y)
> return Set([
> x^2*y,
> y^2*x,
>  x*y*Inverse(x),
>  (y^2*x)^(-1),
>  (x*y)^(-2),
>  y,
>  (x*y)^2*x,
>  (x*y)^2,
>  (x*y*x)^(-1),
>  y*x*y,
>  y^(-1),
>  x,
>  x*y*x,
>  x^(-1)
]);
end;;
```

So the set S of (3.3) will be `Promislow(a, b)`. We now create a function that checks whether every element of the subset S admits more than one representation:

```
gap> is_UPP := function(S)
> local l, x, y;
> l := [];
> for x in S do
>   for y in S do
>     Add(l, x*y);
>   od;
> od;
> if ForAll(Collected(l), x->x[2] <> 1) then
>   return false;
> else
>   return fail;
> fi;
> end;;
```

Finally, we check whether every element of S admits more than one representation:

```
gap> S := Promislow(a, b);;
gap> is_UPP(S);
false
```

In the following example, we reproduce the proof of Gardam's theorem using computer calculations.

Example 3.54 (Gardam). Let K be the field of two elements. For $x = a^2$, $y = b^2$ and $z = (ab)^2$, let

$$p = (1+x)(1+y)(1+z^{-1}), \qquad q = x^{-1}y^{-1} + x + y^{-1}z + z,$$
$$r = 1 + x + y^{-1}z + xyz, \qquad s = 1 + (x + x^{-1} + y + y^{-1})z^{-1}.$$

Then $u = p + qa + rb + sab$ is a non-trivial unit in $K[P]$.

The subgroup $N = \langle x, y, z\rangle$ is a normal free abelian subgroup of rank 3 and $P/N \simeq C_2 \times C_2$ (see [45, Lemma 13.3.3]). Moreover, the set $\{1, a, b, ab\}$ is a right-transversal for the subgroup N of P. In particular, this implies that every element α of $K[P]$ can be written uniquely as $\alpha_0 + \alpha_1 a + \alpha_2 b + \alpha_3 ab$, where α_0, α_1, α_2, α_3 are elements in the Laurent polynomial ring $K[x^{\pm 1}, y^{\pm 1}, z^{\pm 1}]$. Therefore, an element $\alpha = \alpha_0 + \alpha_1 a + \alpha_2 b + \alpha_3 ab$ is a trivial unit if and only if there exists an index i such that $\alpha_j = 0$ for $j \neq i$ and α_i is a monomial of $K[x^{\pm 1}, y^{\pm 1}, z^{\pm 1}]$.

From the previous paragraph, we conclude that the element u is not a trivial unit in $K[P]$. Therefore it remains to show that u is indeed a unit. To that purpose, we explicitly construct its inverse v and check that $u^{-1} = v$.

Indeed, we claim that the inverse of u is the element $v = p_1 + q_1 a + r_1 b + s_1 ab$, where

$$p_1 = x^{-1}(a^{-1}pa), \qquad q_1 = -x^{-1}q, \qquad r_1 = -y^{-1}r, \qquad s_1 = z^{-1}(a^{-1}sa).$$

We show that $uv = vu = 1$. We first need to create the group $P = \langle a, b\rangle$. We use the matrix representation of P given in Example 3.53:

```
gap> a := [[0,1,0,0], [2,0,0,0], [0,0,0,1/2], [0,0,1,0]];;
gap> b := [[0,0,1,0], [0,0,0,1], [2,0,0,0], [0,1/2,0,0]];;
gap> P := Group([a, b]);
```

We create the group algebra $K[P]$ and the embedding $P \hookrightarrow K[P]$. The field K will be `GF(2)` and the embedding will be denoted by `f`.

```
gap> A := GroupRing(GF(2), P);;
gap> f := Embedding(P, A);;
```

Next we define the elements x, y and z:

```
gap> x := Image(f, a^2);;
gap> y := Image(f, b^2);;
gap> z := Image(f, (a*b)^2);;
```

Now we define the elements p, q, r and s:

```
gap> p := (One(A)+x)*(One(A)+y)*(One(A)+Inverse(z));;
gap> r := One(A)+x+Inverse(y)*z+x*y*z;;
gap> q := Inverse(x)*Inverse(y)+x+Inverse(y)*z+z;;
gap> s := One(A)+(x+Inverse(x)+y+Inverse(y))*Inverse(z);;
```

Finally, we define the coefficients p_1, q_1, r_1 and s_1 of the element v:

```
gap> p1 := Inverse(x)*p^Image(f, a);;
gap> q1 := -Inverse(x)*q;;
gap> r1 := -Inverse(y)*r;;
gap> s1 := Inverse(z)*s^Image(f, a);;
```

To conclude, we verify that $u^{-1} = v$:

```
gap> u := p+q*a+r*b+s*a*b;;
gap> v := p1+q1*a+r1*b+s1*a*b;;
gap> IsOne(u*v);
true
gap> IsOne(v*u);
true
```

3.6 Problems

3.1. Prove that every group of order < 60 is solvable.

3.2. Prove that a group of order 455 is cyclic.

3.3. Let G be a simple group of order 168. Compute the number of elements of order seven of G.

3.4. Use the function `AllSmallNonabelianSimpleGroups` to prove that there are no simple groups of order 2540 and 9075.

3.5. Find a group G of order 3^6 such that $\{[x,y] : x, y \in G\} \neq [G,G]$.

3.6. Find a group G of order 2^7 such that $\{[x,y] : x, y \in G\} \neq [G,G]$.

3.7. Prove that groups of orders 15, 35, and 77 are cyclic.

3.8. Prove that a simple group of order 60 is isomorphic to $\mathbb{A}_5$.

3.9. Prove that the only non-abelian simple group of order < 100 is $\mathbb{A}_5$.

3.10. Is the following true? For any finite group G, the set $\{x^2 : x \in G\}$ is a subgroup of G.

3.11. Let G be a finite group and p a prime number. Without using the functions `AllSubgroups` or `ConjugacyClassesSubgroups` on G, write a script that computes all the p-subgroups of G.

3.12. A Carter subgroup C of a group G is a nilpotent self-normalizing subgroup, that is, $C = N_G(C)$.

(a) Write a function that returns the Carter subgroups of a given group.
(b) If G is a solvable group, then there exists a unique (non-empty) conjugacy class of Carter subgroups. Find the smallest non-solvable group satisfying this property.

By the main result of [20], if $p \geq 5$ is a prime and G contains a self-normalizing Sylow p-subgroup, then G is solvable.

3.13. Prove the following theorem of Guralnick [19]. There exists a group G of order $n \leq 200$ such that $[G,G] \neq \{[x,y] : x, y \in G\}$ if and only if

$$n \in \{96, 128, 144, 162, 168, 192\}.$$

3.14. Prove the following extension of Guralnick's theorem (Problem 3.13). There exists a group G of order $n < 1024$ such that $[G,G] \neq \{[x,y] : x, y \in G\}$ if and only if n is one of the following numbers: 96, 128, 144, 162, 168, 192, 216, 240, 256, 270, 288, 312, 320, 324, 336, 360, 378, 384, 400, 432, 448, 450, 456, 480, 486, 504, 512, 528, 540, 560, 576, 594, 600, 624, 640, 648, 672, 702, 704, 720, 729, 744, 750, 756, 768, 784, 792, 800, 810, 816, 832, 840, 864, 880, 882, 888, 896, 900, 912, 918, 936, 960, 972, 1000, 1008.

3.15. Let G be a finite group. Create functions to compute:

(a) The set of subnormal subgroups of G.
(b) The set of perfect subgroups of G.
(c) The set of quasisimple subgroups of G.
(d) The layer of G, that is, the subgroup generated by the subnormal quasisimple subgroups of G.
(e) The generalized Fitting subgroup of G, that is, the subgroup generated by the Fitting subgroup and the layer of G.

3.16. Determine all finite groups of order < 2000 whose composition factors are isomorphic to C_2, C_3, C_5 or $\mathbb{A}_5$.

3.17. Let $\pi = \{2, 3, 5\}$. Determine all π-separable groups of order < 1586.

3.18. Prove that the smallest Brauer pair corresponds to groups of order 2^8. To speed up the calculations one can use the following fact: If (G, H) is a Brauer pair, then $G/[G, G] \simeq H/[H, H]$ and $Z(G) \simeq Z(H)$. See [34, Lemma 2.6.3].

3.19. Describe the group algebras $\mathbb{C}[G]$ for G a finite group of order 28.

3.20. Construct the irreducible representations of $\mathbb{D}_8$ and $\mathbf{SL}_2(3)$.

3.21. Construct the irreducible representations of $\mathbb{A}_4$, $\mathbb{S}_4$ and $\mathbb{A}_5$.

3.22. Verify McKay's Conjecture 3.1 for all sporadic simple groups.

3.23. Verify Isaacs–Navarro Conjecture 3.2 for Th and Co_1.

3.24. Verify Wall's Conjecture 3.10 for the Monster group.

3.25. A group G is said to be *rational* if all its characters are rational-valued.

(a) Write a function that returns true if a given group is rational or false otherwise.
(b) Find the rational groups of order ≤ 200.
(c) Prove that $\mathrm{Sp}_6(2)$ and $\Omega_8^+(2)$ are rational simple groups.
(d) Are there rational sporadic simple groups?

By a theorem of W. Feit and G. M. Seitz [16], $\mathrm{Sp}_6(2)$ and $\Omega_8^+(2)$ are the only non-abelian rational finite simple groups.

3.26. Let G be the group of Example 2.72.

(a) Prove that $a^3 \in Z(G)$ is an element of infinite order.
(b) Prove that $Z(G) \simeq \mathbb{Z} \times C_2$.
(c) Prove that $[G, G] \simeq Q_8$.

3.27. Find five quasisimple groups such that not every central element is a commutator.

3.28. Let q be a prime power and $r > 0$. Let $f(x) = x^q$ be the Frobenius map of $\mathbb{F}_{q^r}$. Then f is an automorphism of order r which acts on the vector space $V = \mathbb{F}_{q^r}^n$ coordinate-wise. Construct the semidirect product $V \rtimes \langle f \rangle$.

3.29. Let G be a Sylow 3-subgroup of $\mathbf{GL}_3(3)$ and H be a non-abelian semidirect product $C_3 \rtimes C_9$. Prove that $G \not\simeq H$ and $\mathbb{Q}[G] \simeq \mathbb{Q}[H]$ as $\mathbb{Q}$-algebras.

References

1. Z. Arad and M. Herzog, editors. *Products of conjugacy classes in groups*, volume 1112 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1985.
2. Y. G. Berkovich. On the number of subgroups of given order in a finite p-group of exponent p. *Proc. Amer. Math. Soc.*, 109(4):875–879, 1990.
3. H. I. Blau. A fixed-point theorem for central elements in quasisimple groups. *Proc. Amer. Math. Soc.*, 122(1):79–84, 1994.
4. R. Brauer. Representations of finite groups. In *Lectures on Modern Mathematics, Vol. I*, pages 133–175. Wiley, New York, 1963.
5. M. Brennan and D. Machale. Variations on a Theme: A_4 Definitely Has no Subgroup of Order Six! *Math. Mag.*, 73(1):36–40, 2000.
6. P. J. Cameron. *Permutation groups*, volume 45 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1999.
7. A. R. Camina and R. D. Camina. The influence of conjugacy class sizes on the structure of finite groups: a survey. *Asian-Eur. J. Math.*, 4(4):559–588, 2011.
8. R. D. Carmichael. *Introduction to the theory of groups of finite order*. Dover Publications, Inc., New York, 1956.
9. T. Ceccherini-Silberstein and M. D'Adderio. *Topics in groups and geometry—growth, amenability, and random walks*. Springer Monographs in Mathematics. Springer, Cham, [2021] ©2021. With a foreword by Efim Zelmanov.
10. A. M. Cohen and D. E. Taylor. On a certain Lie algebra defined by a finite group. *Amer. Math. Monthly*, 114(7):633–639, 2007.
11. J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. $\mathbb{ATLAS}$ *of finite groups*. Oxford University Press, Eynsham, 1985. Maximal subgroups and ordinary characters for simple groups, With computational assistance from J. G. Thackray.
12. H. S. M. Coxeter. *Kaleidoscopes*. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley & Sons, Inc., New York, 1995. Selected writings of H. S. M. Coxeter, Edited by F. Arthur Sherk, Peter McMullen, Anthony C. Thompson and Asia Ivić Weiss, A Wiley-Interscience Publication.
13. E. C. Dade. Answer to a question of R. Brauer. *J. Algebra*, 1:1–4, 1964.
14. P. de la Harpe. *Topics in geometric group theory*. Chicago Lectures in Mathematics. University of Chicago Press, Chicago, IL, 2000.
15. B. Eick and J. Müller. On p-groups forming Brauer pairs. *J. Algebra*, 304(1):286–303, 2006.
16. W. Feit and G. M. Seitz. On finite rational groups and related topics. *Illinois J. Math.*, 33(1):103–131, 1989.
17. G. Gardam. A counterexample to the unit conjecture for group rings. *Ann. of Math. (2)*, 194(3):967–979, 2021.
18. W. Gaschütz. Nichtabelsche p-Gruppen besitzen äussere p-Automorphismen. *J. Algebra*, 4:1–2, 1966.

19. R. M. Guralnick. Commutators and commutator subgroups. *Adv. in Math.*, 45(3):319–330, 1982.
20. R. M. Guralnick, G. Malle, and G. Navarro. Self-normalizing Sylow subgroups. *Proc. Amer. Math. Soc.*, 132(4):973–979, 2004.
21. R. M. Guralnick, G. Malle, and P. H. Tiep. Products of conjugacy classes in finite and algebraic simple groups. *Adv. Math.*, 234:618–652, 2013.
22. K. Harada. Revisiting character theory of finite groups. *Bull. Inst. Math. Acad. Sin. (N.S.)*, 13(4):383–395, 2018.
23. G. Havas and M. Vaughan-Lee. On counterexamples to the Hughes conjecture. *J. Algebra*, 322(3):791–801, 2009.
24. D. R. Hughes and J. G. Thompson. The H-problem and the structure of H-groups. *Pacific J. Math.*, 9:1097–1101, 1959.
25. I. M. Isaacs. Characters of solvable and symplectic groups. *Amer. J. Math.*, 95:594–635, 1973.
26. I. M. Isaacs. *Character theory of finite groups*. AMS Chelsea Publishing, Providence, RI, 2006. Corrected reprint of the 1976 original [Academic Press, New York; MR0460423].
27. I. M. Isaacs. *Finite group theory*, volume 92 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2008.
28. I. M. Isaacs and G. Navarro. New refinements of the McKay conjecture for arbitrary finite groups. *Ann. of Math. (2)*, 156(1):333–344, 2002.
29. S. Kionke and J. Raimbault. On geometric aspects of diffuse groups. *Doc. Math.*, 21:873–915, 2016. With an appendix by Nathan Dunfield.
30. J. C. Lagarias. The $3x + 1$ problem and its generalizations. *Amer. Math. Monthly*, 92(1):3–23, 1985.
31. T. Y. Lam and D. B. Leep. Combinatorial structure on the automorphism group of S_6. *Exposition. Math.*, 11(4):289–308, 1993.
32. M. W. Liebeck, E. A. O'Brien, A. Shalev, and P. H. Tiep. The Ore conjecture. *J. Eur. Math. Soc. (JEMS)*, 12(4):939–1008, 2010.
33. M. W. Liebeck, L. Pyber, and A. Shalev. On a conjecture of G. E. Wall. *J. Algebra*, 317(1):184–197, 2007.
34. K. Lux and H. Pahlings. *Representations of groups*, volume 124 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2010. A computational approach.
35. D. MacHale. Minimum counterexamples in group theory. *Math. Mag.*, 54(1):23–28, 1981.
36. G. Malle. The proof of Ore's conjecture (after Ellers-Gordeev and Liebeck-O'Brien-Shalev-Tiep). *Astérisque*, 361(361):Exp. No. 1069, ix, 325–348, 2014.
37. G. Malle and B. Späth. Characters of odd degree. *Ann. of Math. (2)*, 184(3):869–908, 2016.
38. J. McKay. A new invariant for finite simple groups. *Notices Amer. Math. Soc.*, 18(2):397, 1971.
39. C. F. Miller, III and P. E. Schupp. Some presentations of the trivial group. In *Groups, languages and geometry (South Hadley, MA, 1998)*, volume 250 of *Contemp. Math.*, pages 113–115. Amer. Math. Soc., Providence, RI, 1999.
40. R. F. Morse. On the application of computational group theory to the theory of groups. In *Computational group theory and the theory of groups*, volume 470 of *Contemp. Math.*, pages 1–19. Amer. Math. Soc., Providence, RI, 2008.
41. G. Navarro. The set of conjugacy class sizes of a finite group does not determine its solvability. *J. Algebra*, 411:47–49, 2014.
42. G. Navarro. *Character theory and the McKay conjecture*, volume 175 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2018.
43. G. Navarro. Problems on characters: solvable groups. *Publ. Mat.*, 67(1):173–198, 2023.
44. P. S. Novikov. *Ob algoritmičeskoĭ nerazrešimosti problemy toždestva slov v teorii grupp*. Izdat. Akad. Nauk SSSR, Moscow, 1955. Trudy Mat. Inst. Steklov. no. 44.
45. D. S. Passman. *The algebraic structure of group rings*. Robert E. Krieger Publishing Co., Inc., Melbourne, FL, 1985. Reprint of the 1977 original.
46. S. D. Promislow. A simple example of a torsion-free, nonunique product group. *Bull. London Math. Soc.*, 20(4):302–304, 1988.

47. D. Quillen. Homotopy properties of the poset of nontrivial p-subgroups of a group. *Adv. in Math.*, 28(2):101–128, 1978.
48. J. J. Rotman. *An introduction to the theory of groups*, volume 148 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, fourth edition, 1995.
49. J.-P. Serre. *Linear representations of finite groups*. Graduate Texts in Mathematics, Vol. 42. Springer-Verlag, New York-Heidelberg, 1977. Translated from the second French edition by Leonard L. Scott.
50. J.-P. Serre. *Trees*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. Translated from the French original by John Stillwell, Corrected 2nd printing of the 1980 English translation.
51. J. Stillwell. The word problem and the isomorphism problem for groups. *Bull. Amer. Math. Soc. (N.S.)*, 6(1):33–56, 1982.
52. J. Szeṕ. Sui gruppi fattorizzabili non semplici. *Rend. Mat. e Appl. (5)*, 22:245–252, 1963.
53. J. Szép. Sui gruppi fattorizzabili. *Rend. Sem. Mat. Fis. Milano*, 38:228–230, 1968.
54. G. E. Wall. Some applications of the Eulerian functions of a finite group. *J. Austral. Math. Soc.*, 2:35–59, 1961/1962.
55. R. A. Wilson. The McKay conjecture is true for the sporadic simple groups. *J. Algebra*, 207(1):294–305, 1998.

Index