

The OSCAR-GAP Interface

Behind the Scenes

Max Horn

RPTU University Kaiserslautern-Landau

6 May 2026



GAP in OSCAR, GAP in Julia

The GAP.jl Package for Julia

The GAP Prompt for Julia

Example: Using OSCAR in the Alnuth Package

Example: Wrapping GAP semigroups for OSCAR

GAP in OSCAR, GAP in Julia

GAP in OSCAR: The General Philosophy

- GAP is not just *used* by OSCAR – it is an *integral part*
- Users should not need to explicitly invoke GAP
- Users should not even need to *know* how to explicitly invoke GAP
- Instead use it automatically whenever necessary and sensible
- To this end, GAP experts work on OSCAR towards optimal integration
- However users still *can* call it explicitly \rightsquigarrow “escape hatch”
- To enable all that, we first worked on making GAP available in Julia, with minimal overhead and minimal “frills”
- (Similar for other cornerstones)

The GAP.jl Package for Julia

What is GAP.jl?

- The Julia package GAP.jl makes GAP objects accessible from Julia
- ...and GAP functions callable from Julia
- ...and vice-versa: it is *bidirectional*
- Bundles its own GAP, fully compiled, and with (almost) all GAP packages
 - This is essential: our GAP is compiled to use the Julia memory manager
- Each GAP.jl version bundles one specific GAP version
 - For example, GAP.jl 0.16.0 and 0.16.1 both bundle GAP 4.15.1
- GAP.jl is used by Oscar.jl to access GAP functionality, but can also be used on its own

Loading GAP in Julia

Not what you'd normally do when using OSCAR, but you *can* do it

```
julia> using GAP
GAP 4.15.1 of 2025-10-18
┌ GAP ───────────────────────────────────────────────────────────────────────────────────┐
│ https://www.gap-system.org                                                         │
│ Architecture: aarch64-apple-darwin20-julia1.12-64-kv10                            │
│ Configuration: gmp 6.3.0, Julia GC, Julia 1.12.2, readline                        │
│ Loading the library and packages ...                                               │
│ Packages: ACLib 1.3.3, Alnuth 3.2.1, AtlasRep 2.1.9, AutoDoc 2025.10.16,          │
│ AutPGrp 1.11.1, Browse 1.8.21, CaratInterface 2.3.7, CRISP 1.4.8,                 │
│ Cryst 4.1.30, CrystCat 1.1.10, CTblLib 1.3.11, curlInterface 2.4.2,               │
│ FactInt 1.6.3, FGA 1.5.0, Forms 1.2.13, GAPDoc 1.6.7, genss 1.6.9, IO 4.9.3,      │
│ IRREDSOL 1.4.4, JuliaInterface 0.16.1, LAGUNA 3.9.7, orb 5.0.1,                  │
│ PackageManager 1.6.3, Polenta 1.3.11, Polycyclic 2.17, PrimGrp 4.0.1,            │
│ RadiRoot 2.9, recog 1.4.4, ResClasses 4.7.4, SmallGrp 1.5.4, Sophus 1.27,        │
│ SpinSym 1.5.2, StandardFF 1.0, TomLib 1.2.11, TransGrp 3.6.5, utils 0.92         │
│ Try '??help' for help. See also '?copyright', '?cite' and '?authors'              │
julia>
```

Spot the JuliaInterface package in the list?

What does GAP.jl do?

- Goal: make GAP functionality available in Julia with minimal overhead

```
julia> G = GAP.Globals.SymmetricGroup(3)
GAP: Sym( [ 1 .. 3 ] )
```

- GAP modified to use Julia memory manager: GAP objects *are* Julia objects

```
julia> typeof(G)
GapObj
```

- Supports some operations for “basic” objects types (strings, lists, records, ...)

```
julia> l = GAP.Globals.AsSet(G)
GAP: [ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
```

```
julia> l[1]
GAP: ()
```

```
julia> length(l)
6
```

Conversion

- So `GAP.jl` makes GAP objects available in Julia
- Also allows converting “basic” object types (integers, rationals, strings, lists, records, ...) between both systems via *coercion* syntax:

```
julia> v = GapObj( [1, 2, 3] )  
GAP: [ 1, 2, 3 ]  
  
julia> Vector{Int}(v) == [1, 2, 3]  
true
```

- Conversions must be explicit (exception: booleans, small ints)
- Problem with automatic conversion:
 - performance loss \rightsquigarrow must be possible to disable anyway
 - “correct” conversion in general not clear (to what should a GAP group be translated)?
- Instead, OSCAR *wraps* GAP objects into high-level OSCAR objects, and performs conversions only if necessary

More Conversion

- GAP and Julia objects can be nested

```
julia> l = ["a", "b", "c"];  
  
julia> v = GapObj(l)  
GAP: [ <Julia: "a">, <Julia: "b">, <Julia: "c"> ]  
  
julia> Vector{String}(v) == l  
true
```

- It may be useful to perform “recursive” conversions

```
julia> w = GapObj(l; recursive=true)  
GAP: [ "a", "b", "c" ]  
  
julia> Vector{String}(w) == l  
true
```

Calling GAP from Julia

```
gap> Combinations([1..3], 2);  
[ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]
```

Now do it from Julia:

```
julia> x = GAP.Obj(1:3)  
GAP: [ 1 .. 3 ]  
  
julia> y = GAP.Globals.Combinations(x, 2)  
GAP: [ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]  
  
julia> typeof(y)  
GapObj  
  
julia> Vector{Vector{Int}}(y)  
3-element Vector{Vector{Int64}}:  
 [1, 2]  
 [1, 3]  
 [2, 3]
```

Calling GAP from Julia

```
gap> Combinations([1..3], 2);  
[ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]
```

Now do it from Julia:

```
julia> x = GAP.Obj(1:3)  
GAP: [ 1 .. 3 ]  
  
julia> y = GAP.Globals.Combinations(x, 2)  
GAP: [ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]  
  
julia> typeof(y)  
GapObj  
  
julia> Vector{Set{Int}}(y)  
3-element Vector{Set{Int64}}:  
 Set([2, 1])  
 Set([3, 1])  
 Set([2, 3])
```

Calling GAP from Julia

```
gap> Combinations([1..3], 2);  
[ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]
```

Now do it from Julia:

```
julia> x = GAP.Obj(1:3)  
GAP: [ 1 .. 3 ]  
  
julia> y = GAP.Globals.Combinations(x, 2)  
GAP: [ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]  
  
julia> typeof(y)  
GapObj  
  
julia> Set{Set{Int}}(y)  
Set{Set{Int64}} with 3 elements:  
  Set([3, 1])  
  Set([2, 3])  
  Set([2, 1])
```

Calling GAP from Julia (cont.)

```
gap> Combinations([1..3], 2);  
[ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ]
```

Can we wrap this into a Julia function?

```
julia> my_combinations(list, n::Int) =  
    Vector{Vector{Int}}(  
        GAP.Globals.Combinations(GAP.Obj(list), n)  
    );
```

```
julia> my_combinations(1:3, 2)  
3-element Vector{Vector{Int64}}:  
 [1, 2]  
 [1, 3]  
 [2, 3]
```

Calling Julia From GAP

```
 julia> clipboard() # get content of clipboard as a string
 "Hello, World"
```

```
 julia> using JSON ; JSON.parsefile("example.json")
 Dict{String, Any} with 2 entries:
  "b" => 2
  "a" => 1
```

Now do it from GAP:

```
 gap> s := Julia.clipboard(); Julia.GAP.Obj(s);
 <Julia: "Hello, World">
 "Hello, World"

 gap> d := Julia.JSON.parsefile(Julia.String("example.json"));
 <Julia: Dict{String, Any}{"b" => 2, "a" => 1}>

 gap> Julia.GAP.Obj(d);
 rec( a := 1, b := 2 )
```

Calling GAP from Julia from GAP from ...

```
julia> f = GAP.evalstr("""function(x) Print("f: In GAP, got ", x, "\\n"); end""")
GAP: function( x ) ... end

julia> g(y) = begin println("g: In Julia, got $y") ; f(y) end
g (generic function with 1 method)

julia> h = GAP.evalstr("""function(z) Print("h: In GAP, got ", z, "\\n"); Julia.g(z); end""")
GAP: function( z ) ... end

julia> h( [1,2,3] )
h: In GAP, got [1, 2, 3]
g: In Julia, got [1, 2, 3]
f: In GAP, got [1, 2, 3]
```

This allows us to incrementally reimplement GAP features in Julia

The GAP Prompt for Julia

How to Let GAP Run the Show

- So... how did I actually do this in a GAP prompt:

```
gap> s := Julia.clipboard(); Julia.GAP.Obj(s);  
<Julia: "Hello, World">  
"Hello, World"
```

- The secret sauce is `GAP.prompt()`
- Demo time!

Example: Using OSCAR in the Alnuth Package

Using OSCAR in the Alnuth Package

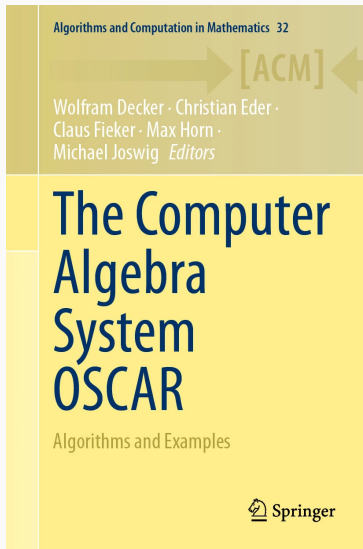
- The GAP package “Alnuth” provides “ALgebraic NUmber THeory” capabilities
- Originally by interfacing to KANT
- This was later replaced by a PARI/GP interface
- Today in addition it can interface with OSCAR
- Let's look at the code...

Example: Wrapping GAP semigroups for OSCAR

- GAP provides functionality for semigroups, and of course many packages (e.g. `semigroups`, `numericalsgps`) provide more
- OSCAR currently has no semigroups at all
- ...due to Demand Driven Development: there was no “demand” yet ...
- But at least this makes for a nice example:
- Showcase: wrapping GAP semigroups for OSCAR in less than 60 lines of code

Check out the OSCAR book to learn from many worked out examples, in 19 chapters on topics ranging from basics to current research.

 book.oscar-system.org



Subscribe to the OSCAR newsletter to stay informed about recent developments, upcoming events and more.

 oscar-system.org/newsletter

