Visualising GAP objects: Current status and the future

Lukas Schnelle

GAPDays Spring 2026

Background

Currently there are multiple packages, which provide functionality to visualise GAP objects.

Background

Currently there are multiple packages, which provide functionality to visualise GAP objects.

Motivation

Last GAPDays (Summer 2025) James Mitchell and I noticed that we both work on packages which can visualise graphs in tikz with some duplicate functionality.

Background

Currently there are multiple packages, which provide functionality to visualise GAP objects.

Motivation

Last GAPDays (Summer 2025) James Mitchell and I noticed that we both work on packages which can visualise graphs in tikz with some duplicate functionality.

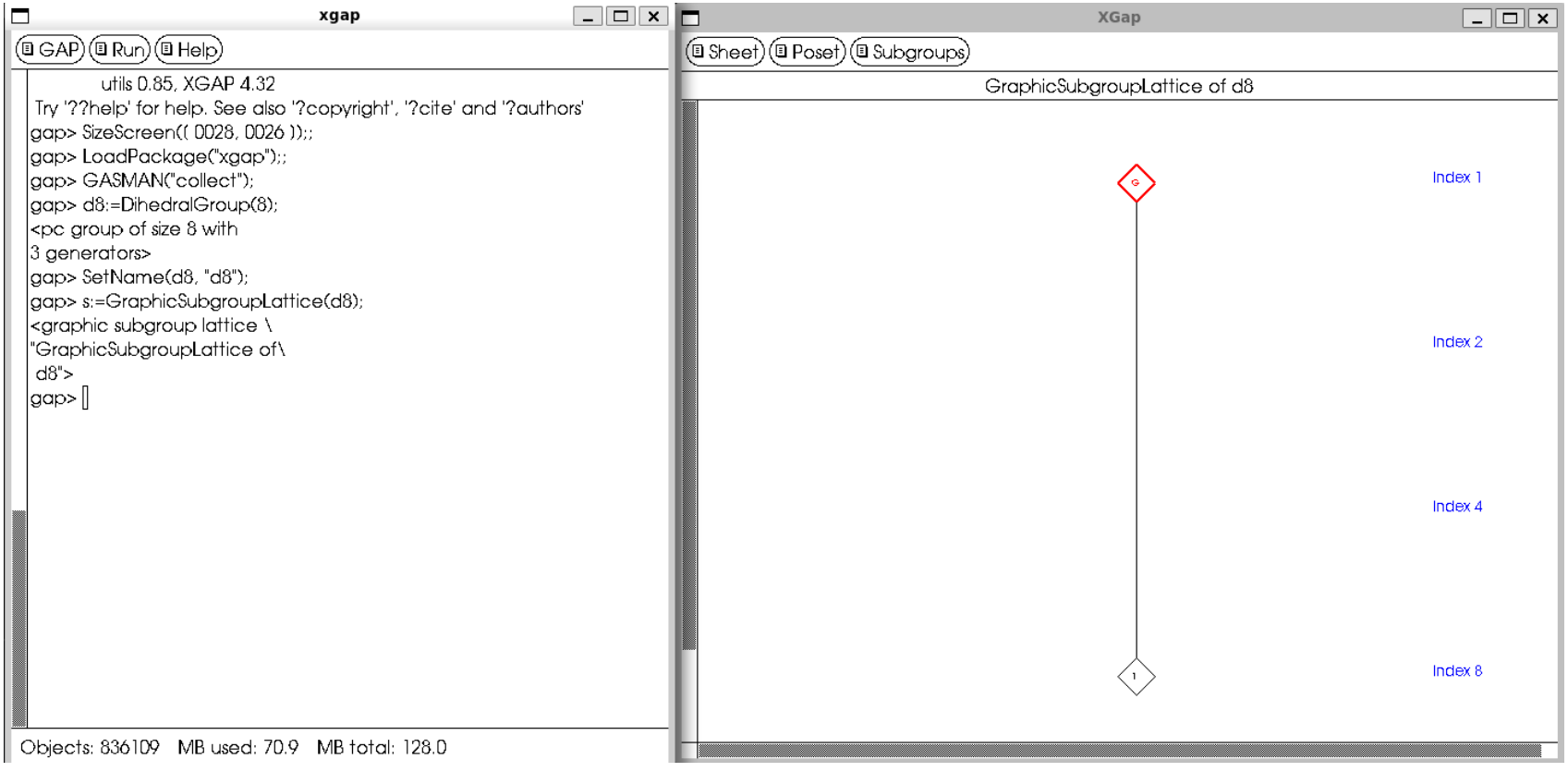
First we will give a (likely incomplete) overview of them with some examples.

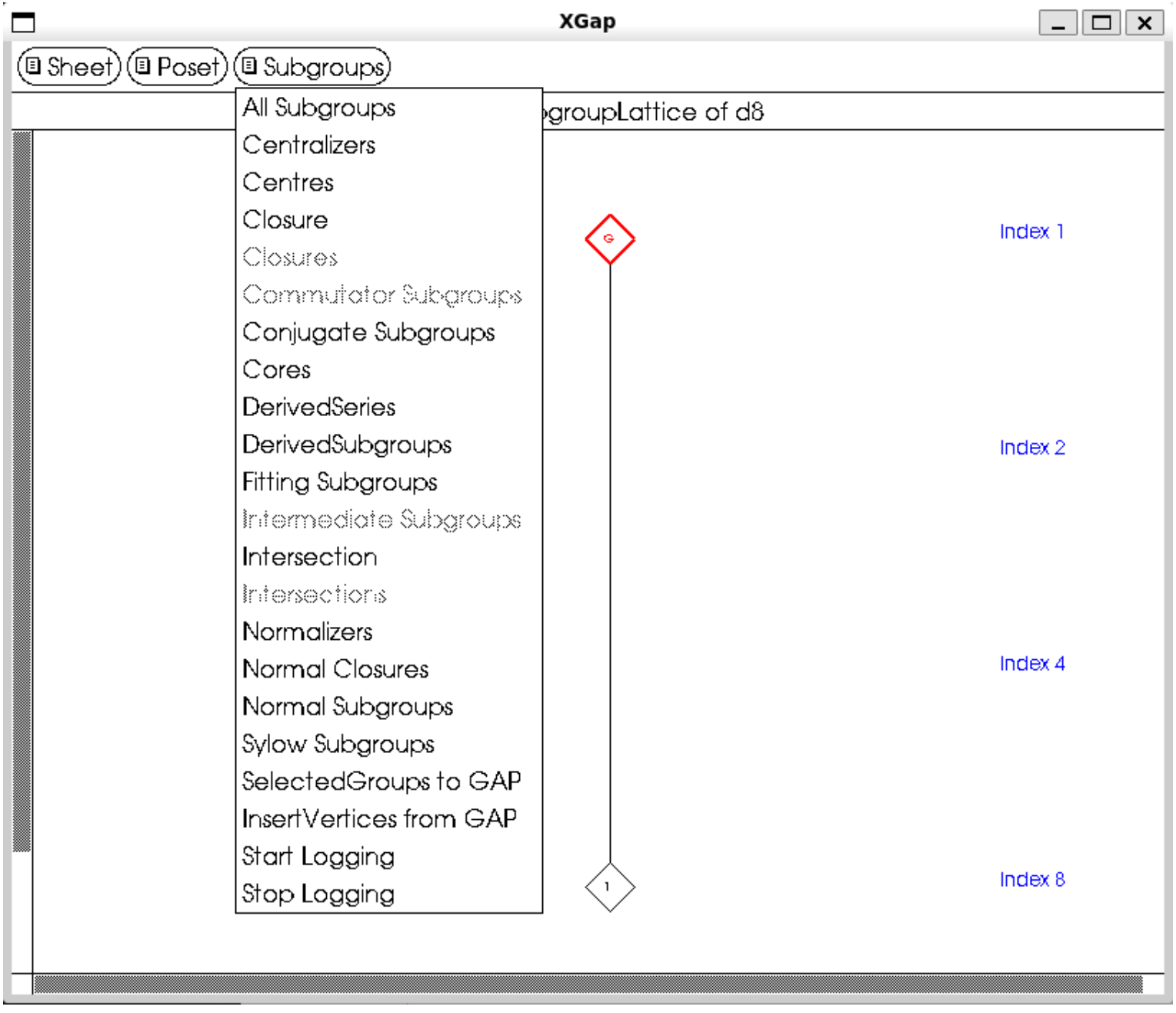
XGAP (By Frank Celler and Max Neunhöffer)

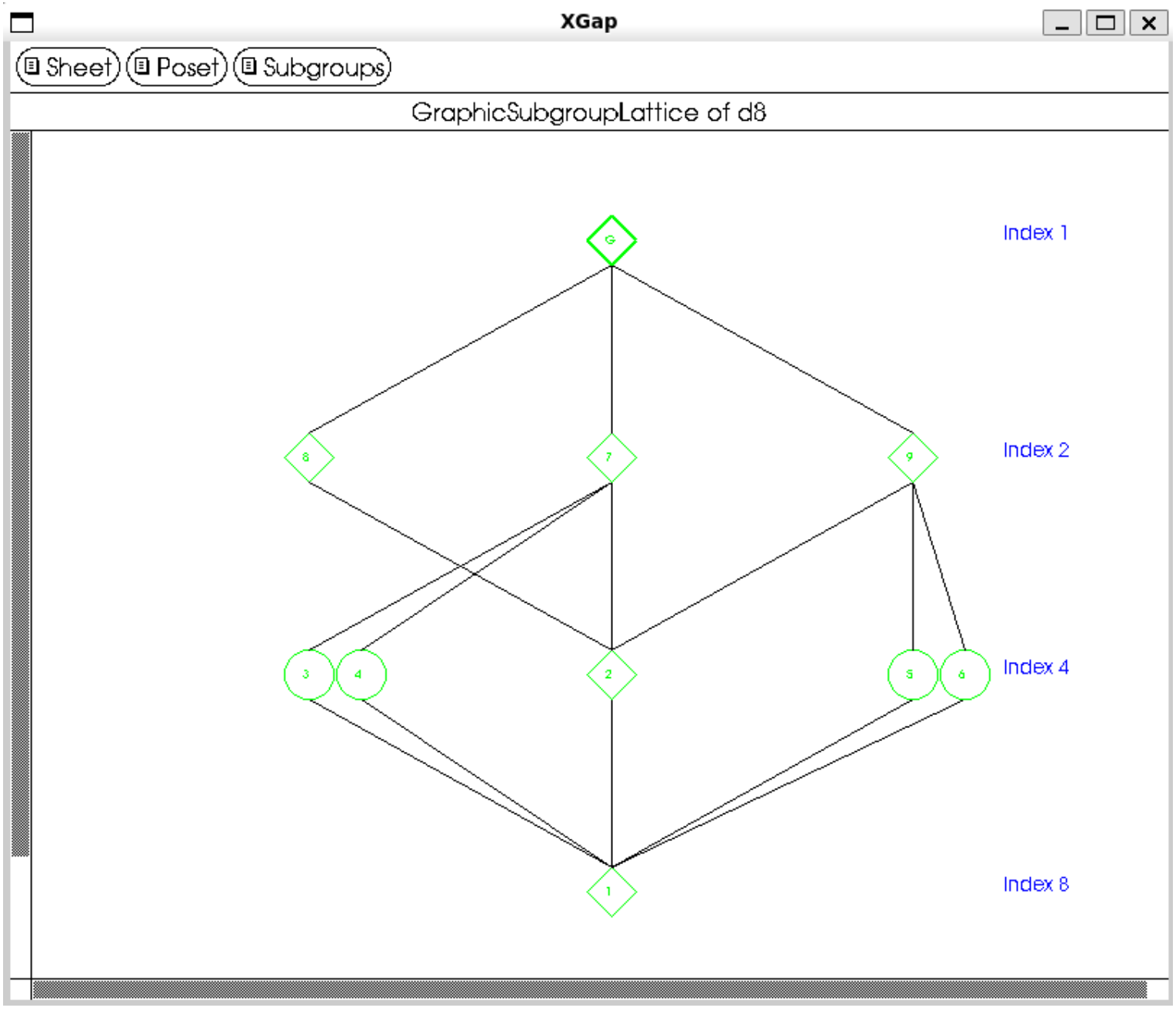
Provides a GUI in which e.g. subgroup lattices can be investigated.
Needs to be started separately.

```
schnellecom@CURIE:/opt/gap/pkg/xgap$ ./xgap.sh
```


xgap
GAP
Run
Help
GAP 4.14.0 of 2024-12-05
GAP https://www.gap-system.org
Architecture: x86_64-pc-linux-gnu-default64-kv9
Configuration: gmp 6.2.1, GASMAN
Loading the library and packages ...
Packages: AClib 1.3.2, Alnuth 3.2.1, AtlasRep 2.1.9, AutPGrp 1.11,
Browse 1.8.21, CaratInterface 2.3.7, CRISP 1.4.6, Cryst 4.1.29,
CrystCat 1.1.10, CTblLib 1.3.9, curlInterface 2.4.0,
FactInt 1.6.3, FGA 1.5.0, Forms 1.2.12, GAPDoc 1.6.7,
genss 1.6.9, IO 4.9.1, IRREDSOL 1.4.4, LAGUNA 3.9.7, orb 4.9.1,
Polenta 1.3.10, Polycyclic 2.16, PrimGrp 3.4.4, RadiRoot 2.9,
recog 1.4.3, ResClasses 4.7.3, SmallGrp 1.5.4, Sophus 1.27,
SpinSym 1.5.2, StandardFF 1.0, TomLib 1.2.11, TransGrp 3.6.5,
utils 0.85, XGAP 4.32
Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap> SizeScreen((0028, 0026));;
gap> LoadPackage("xgap");;
gap> GASMAN("collect");
gap>
Objects: 836109 MB used: 70.9 MB total: 128.0


xgap
GAP
Run
Help
utils 0.85, XGAP 4.32
Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap> SizeScreen((0028, 0026));;
gap> LoadPackage("xgap");;
gap> GASMAN("collect");
gap> d8:=DihedralGroup(8);
<pc group of size 8 with
3 generators>
gap> SetName(d8, "d8");
gap> s:=GraphicSubgroupLattice(d8);
<graphic subgroup lattice \
"GraphicSubgroupLattice of\
d8">
gap>
Objects: 836109 MB used: 70.9 MB total: 128.0
XGap
Sheet
Poset
Subgroups
GraphicSubgroupLattice of d8
Index 1
Index 2
Index 4
Index 8


XGap
Sheet
Poset
Subgroups
All Subgroups
Centralizers
Centres
Closure
Closures
Commutator Subgroups
Conjugate Subgroups
Cores
DerivedSeries
DerivedSubgroups
Fitting Subgroups
Intermediate Subgroups
Intersection
Intersections
Normalizers
Normal Closures
Normal Subgroups
Sylow Subgroups
SelectedGroups to GAP
InsertVertices from GAP
Start Logging
Stop Logging
groupLattice of d8
Index 1
Index 2
Index 4
Index 8


XGap
Sheet
Poset
Subgroups
GraphicSubgroupLattice of d8
Index 1
Index 2
Index 4
Index 8
G
8
7
9
3
4
2
5
6
1

Advantages

- Direct GUI to use
- Freely movable visualisation

Advantages

- Direct GUI to use
- Freely movable visualisation

Challenges

- Needs to be started standalone
- Needs to be adjusted to every OS

francy (by Manuel Martins)

Idea

Allow the XGAP system to run in a Jupyter Notebook.

Status

I was not able to get this running.

```
LoadPackage("digraphs"); LoadPackage("francy");

G := SymmetricGroup(3); as := AllSubgroups(G); nodes := [];
d := Digraph(as, {H, K} -> IsSubgroup(H, K));

vertices := DigraphVertices(d); edges := DigraphEdges(d);

canvas := Canvas(Concatenation("Subgroups Digraph of ",
  String(G)));
graph := Graph(GraphType.DIRECTED);
Add(canvas, graph);

customMessage := FrancyMessage(FrancyMessageType.INFO,
  "Simple Groups", "A group is simple if it is nontrivial
  and has no nontrivial normal subgroups.");

IsGroupSimple := function(i)
  Add(canvas, simpleGroupMessage);
  if IsSimpleGroup(as[i]) then
    Add(canvas, FrancyMessage("Simple",
      Concatenation("The vertex ", String(i),
        ", representing the subgroup ", String(as[i]),
        ", is simple.")));
  else
    Add(canvas, FrancyMessage("Not Simple",
      Concatenation("The vertex ", String(i),
        ", representing the subgroup ", String(as[i]),
        ", is not simple.")));
  fi;
  return Draw(canvas);
end;
```

```
for i in vertices do
  nodes[i] := Shape(ShapeType.CIRCLE, String(i));
  Add(nodes[i], Menu("Is this subgroup simple?",
      Callback(IsGroupSimple, [i])));
  Add(graph, nodes[i]);
od;

for i in edges do
  Add(graph, Link(nodes[i[1]], nodes[i[2]]));
od;

Draw(canvas);
```

(from [1])

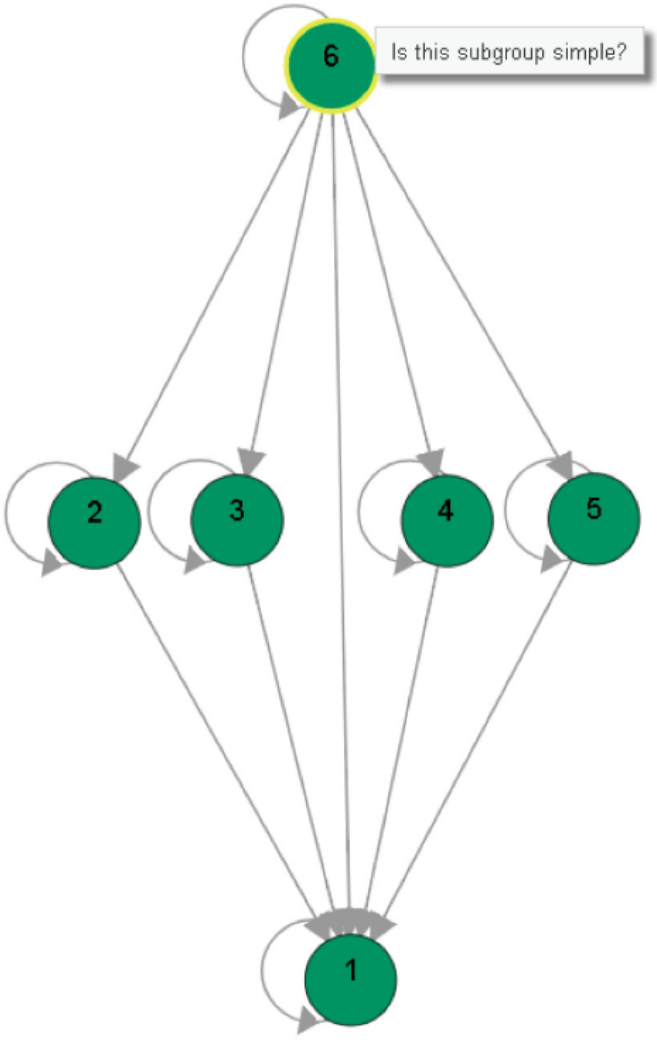
Francy

Subgroups Digraph of SymmetricGroup([1 .. 3])

Simple Groups A group is simple if it is nontrivial and has no nontrivial normal subgroups.

Simple The vertex 2, representing the subgroup Group([(2,3)]), is simple.

Not Simple The vertex 6, representing the subgroup SymmetricGroup([1 .. 3]), is not simple.



(from [1])

Advantages

- Runs in browser
- Highly customisable

Challenges

- Needs additional installation of Jupyter

Advantages

- Runs in browser
- Highly customisable

Challenges

- Needs additional installation of Jupyter

Related packages

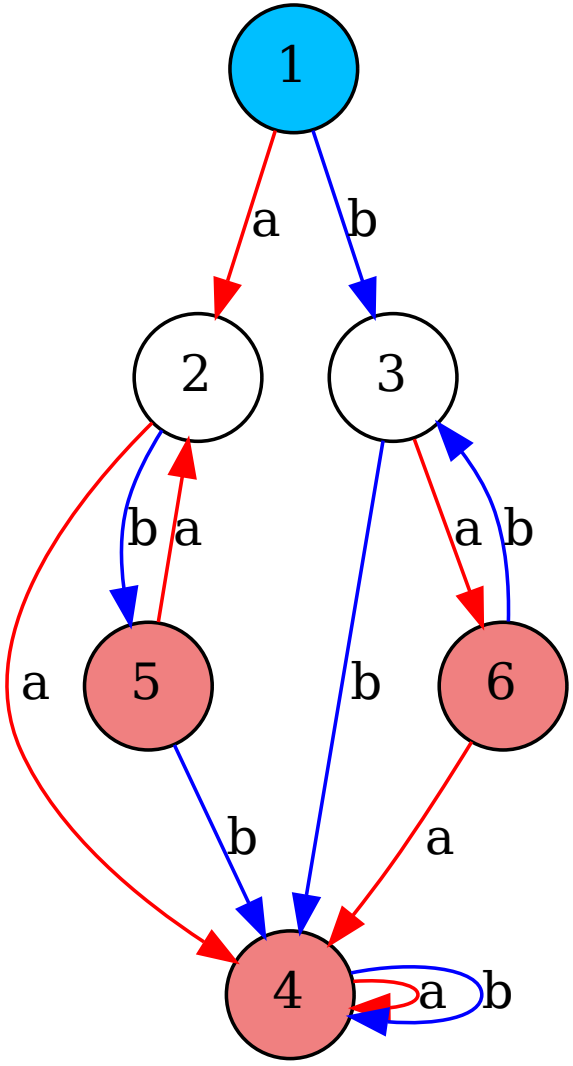
- jupyterviz
- semigroupviz

SgpViz (by Manuel Delgado and José João Morais)

Semigroup visualisation

- Can visualise e.g. Cayley graphs
- Using graphviz

```
gap> f := FreeMonoid("a","b");;
gap> a := GeneratorsOfMonoid( f )[ 1 ];;
gap> b := GeneratorsOfMonoid( f )[ 2 ];;
gap> r:=[[a^3,a^2], [a^2*b,a^2], [b*a^2,a^2], [b^2,a^2], [a*b*a,a], [b*a*b,b] ];;
gap> b21:= f/r;
<fp monoid on the generators [ a, b ]>
gap> DrawCayleyGraph(b21);
```


1
a
b
2
3
b
a
a
b
a
5
b
6
b
a
4
a
b

Advantages

- Uses graphviz standard
- Outputs PDFs

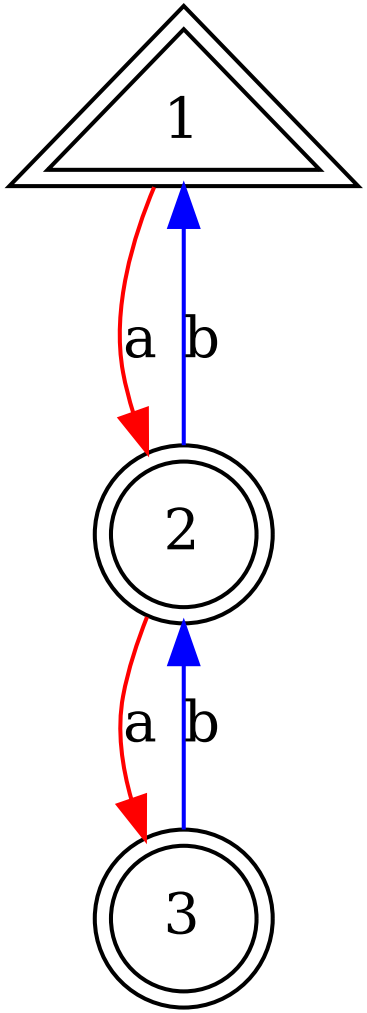
Challenges

- Needs additional installation of graphviz on system

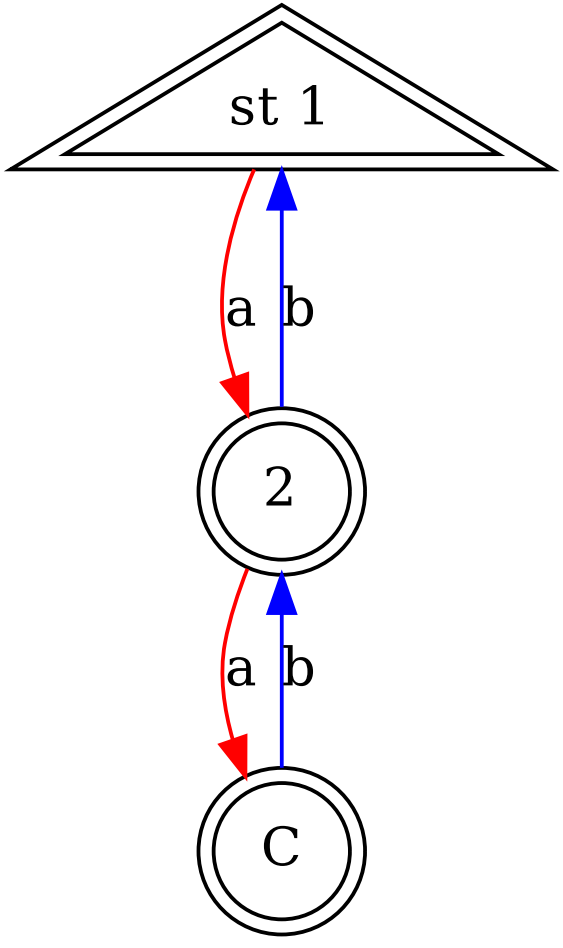
Automata (by Manuel Delgado, Steve Linton and José João Morais)

Automata (by Manuel Delgado, Steve Linton and José João Morais)

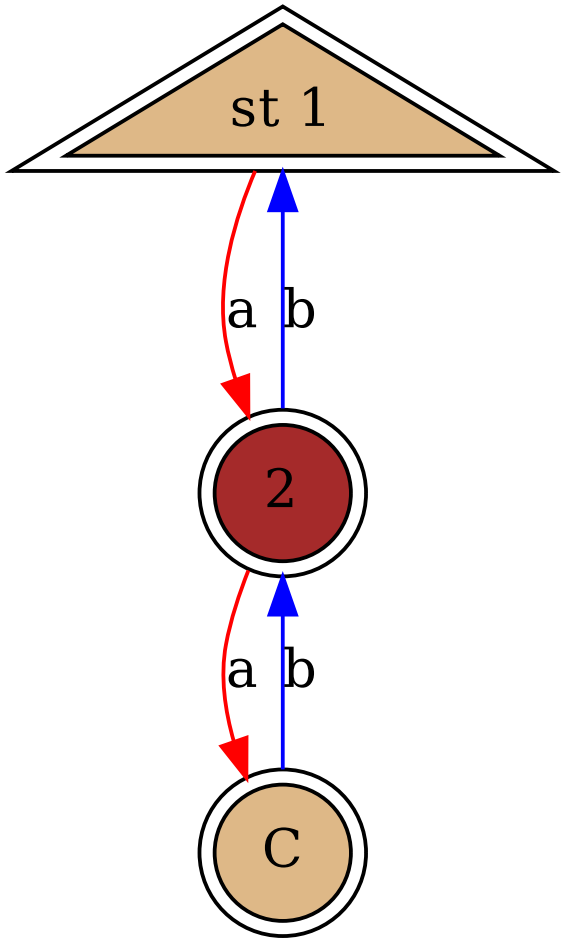
```
gap> x:=Automaton("det",3,2,[ [ 2, 3, 0 ], [ 0, 1, 2 ] ],[ 1 ],[ 1, 2, 3 ]);;
gap> DrawAutomaton(x);
```


1
a
b
2
a
b
3

```
gap> DrawAutomaton(x,["st 1", "2", "C"]);
```



```
gap> DrawAutomaton(x,["st 1", "2", "C"],[[2],[1,3]]);
```



Advantages

- Uses graphviz standard
- Outputs PDFs

Advantages

- Uses graphviz standard
- Outputs PDFs

Challenges

- Needs additional installation of graphviz on system

graphviz (by James D. Mitchell and Matthew Pancer)

Digraphs visualisation

- (some work on) having Digraphs package as extension
- Also uses graphviz standard

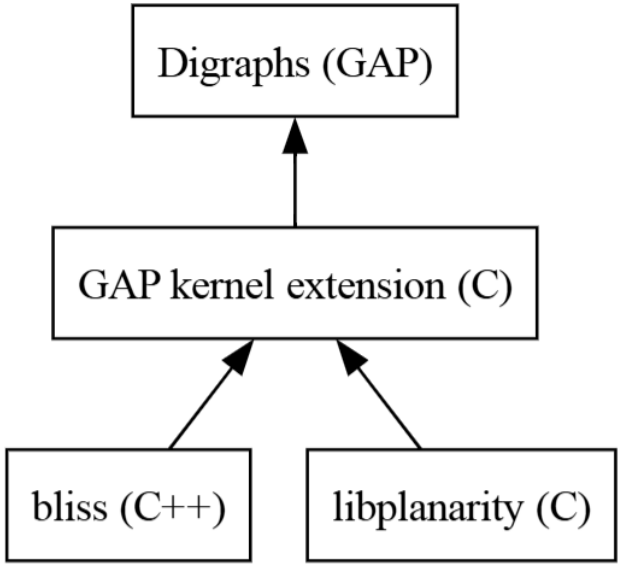
graphviz (by James D. Mitchell and Matthew Pancer)

Digraphs visualisation

- (some work on) having Digraphs package as extension
- Also uses graphviz standard

```
gap> D := Digraph([[], [1], [2], [2]]);
<immutable digraph with 4 vertices, 3 edges>
gap> SetDigraphVertexLabels(D,
> ["Digraphs (GAP)",
> "GAP kernel extension (C)",
> "bliss (C++)",
> "libplanarity (C)"]);
gap> gv := GraphvizVertexLabelledDigraph(D);
<graphviz digraph "hgn" with 4 nodes and 3 edges>
gap> GraphvizSetAttr(gv, "node [shape=box]");
<graphviz digraph "hgn" with 4 nodes and 3 edges>
gap> GraphvizSetAttr(gv, "rankdir", "BT");
<graphviz digraph "hgn" with 4 nodes and 3 edges>
```

```
gap> Print(AsString(gv), "\n");
//dot
digraph hgn {
        node [shape=box] rankdir=BT
        1 [label=""Digraphs (GAP)""]
        2 [label=""GAP kernel extension (C)""]
        3 [label=""bliss (C++)""]
        4 [label=""libplanarity (C)""]
        2 -> 1
        3 -> 2
        4 -> 2
}
```



Advantages

- Uses graphviz standard
- Structure so it can can work with other packages
- Highly customisable
- Visualisations can be iterated

Challenges

- Needs GAP function for everything in graphviz standard
- No immediate way to show results without additional installation on system

```
gap> Splash(gv);
Error: /tmp/gaptempdirv6HuGo/hgn.dot: syntax error in line 4 near '('
```

IntPic (by Manuel Delgado)

Visualises arrays of integers

Can be used to highlight different properties in large number of colours.

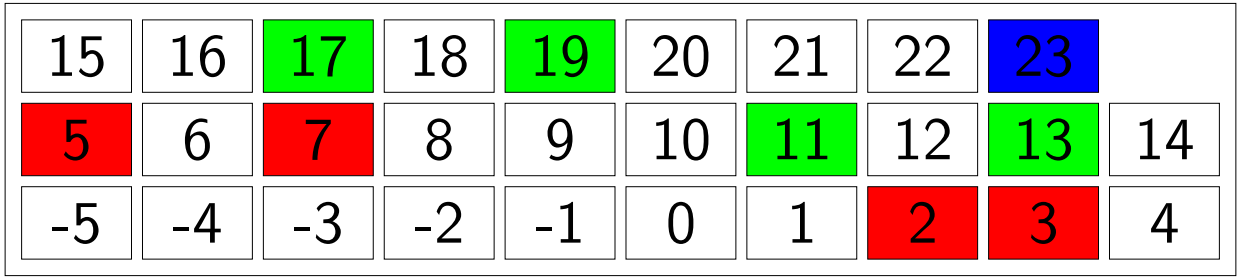
IntPic (by Manuel Delgado)

Visualises arrays of integers

Can be used to highlight different properties in large number of colours.

```
gap> rg := [-5..23];;
gap> len := 10;;
gap> hg := rec();;
gap> hg.highlights:=[[2,3,5,7],[11,13,17,19],[23]];;
gap> tkz := IP_TikzArrayOfIntegers(rg,len,hg);;
```

```
gap> Print(tkz);
%tikz
\begin{tikzpicture}[every node/.style={draw,scale=1pt,
minimum width=20pt,inner sep=3pt,
line width=0pt,draw=black}]
\matrix[row sep=2pt,column sep=2pt]
{\node[]{15};&
\node[]{16};&
\node[fill=green]{17};&
\node[]{18};&
\node[fill=green]{19};&
\node[]{20};&
\node[]{21};&
\node[]{22};&
\node[fill=blue]{23};\\
\node[fill=red]{5};&
\node[]{6};&
\node[fill=red]{7};&
\node[]{8};&
\node[]{9};&
\node[]{10};&
\node[fill=green]{11};&
\node[]{12};&
\node[fill=green]{13};&
\node[]{14};\\
\node[]{-5};&
\node[]{-4};&
\node[]{-3};&
\node[]{-2};&
\node[]{-1};&
\node[]{0};&
\node[]{1};&
\node[fill=red]{2};&
\node[fill=red]{3};&
\node[]{4};\\
};
\end{tikzpicture}
```



Advantages

- Uses tikz
- Highly customisable

Advantages

- Uses tikz
- Highly customisable

Challenges

- No immediate way to show results without compiling them

GAPic (by Alice C. Niemeyer and many more)

Visualising triangulated polyhedra

- Came from the SimplicialSurfaces package
- Expanded functionality like general polyhedra
- Has SimplicialSurfaces package as extension

```
gap> oct := Octahedron();;
gap> verticesPositions := [
> [ 0, 0, Sqrt(2.) ],
> [ 1, 1, 0 ],
> [ 1, -1, 0 ],
> [ -1, -1, 0 ],
> [ -1, 1, 0 ],
> [ 0, 0, -Sqrt(2.) ] ];;
gap> printRecord := SetVertexCoordinates3D(oct, verticesPositions, rec());;
gap> DrawComplexToJavaScript(oct, "octahedron.html", printRecord);;
```

Demonstration in browser

```
gap> oct := Octahedron();;
gap> verticesPositions := [
> [ 0, 0, Sqrt(2.) ],
> [ 1, 1, 0 ],
> [ 1, -1, 0 ],
> [ -1, -1, 0 ],
> [ -1, 1, 0 ],
> [ 0, 0, -Sqrt(2.) ] ];;
gap> printRecord := SetVertexCoordinates3D(oct, verticesPositions, rec());;
gap> DrawComplexToJavaScript(oct, "octahedron.html", printRecord);;
```

Demonstration in browser

```
gap> printRecord := SetFaceColour(oct, 3, "0x00FF00", printRecord);
rec( edgeThickness := 0.03, faceColours := [ ,, "0x00FF00" ],
  vertexCoordinates3D := [ [ 0, 0, 1.41421 ], [ 1, 1, 0 ], [ 1, -1, 0 ], [ -1, -1, 0 ], [ -1, 1, 0 ],
      [ 0, 0, -1.41421 ] ] )
gap> DrawComplexToJavaScript(oct, "octahedron-coloured.html", printRecord);;
```

Advantages

- Results can be opened with any browser
- High interactivity
- Results can be iterated
- Can use parameterised coordinates

Challenges

- Every new feature needs own GAP function
- No immediate way to show results without generating them

There are certainly more packages that include some visualisation which I did not mention.

(My) goals

Make visualisations in GAP

- more intuitive $\rightarrow$ accessible
- sustainable
- more visible

(My) goals

Make visualisations in GAP

- more intuitive $\rightarrow$ accessible
- sustainable
- more visible

Possible methods

- Use of package extensions?
- Use of existing standards

Questions and discussion

- What are your experiences with visualisations in GAP?
- What do you expect from a visualisation functionality?
- Is there anything you want to visualise but are not able to visualise yet?
- How can we provide libraries of examples?