Introduction to GAP:
A System for Computational Discrete Algebra

Inneke Van Gelder

January 14, 2011
What is GAP?

GAP is an open source system for computational discrete algebra. GAP = Groups, Algorithms, Programming.

GAP provides:
- Programming language
- Library of thousands of functions implementing algebraic algorithms
- Large data libraries of algebraic objects
What is GAP?

GAP is an open source system for computational discrete algebra
GAP is an open source system for computational discrete algebra
GAP = \textbf{G}roups, \textbf{A}lgorithms, \textbf{P}rogramming
What is GAP?

GAP is an open source system for computational discrete algebra
GAP = Groups, Algorithms, Programming
GAP provides:

- Programming language
- Library of thousands of functions implementing algebraic algorithms
- Large data libraries of algebraic objects
Who uses GAP?
Who uses GAP?

- Students
- Teachers
- Researchers
- ...

References
<table>
<thead>
<tr>
<th>How to install?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available for UNIX/Linux, Windows, and Macintosh operating systems</td>
</tr>
<tr>
<td>Go to <a href="http://www.gap-system.org">http://www.gap-system.org</a></td>
</tr>
<tr>
<td>Download the archives and follow the instructions</td>
</tr>
</tbody>
</table>
How to install?

- Available for **UNIX/Linux**, **Windows** and **Macintosh** operating systems
- Go to [http://www.gap-system.org](http://www.gap-system.org)
- **Download** the archives and follow the instructions
Getting started

GAP manuals can be found at http://www.gap-system.org/Doc/manuals.html including:
- Tutorial for getting started with GAP
- Reference manual
- Package manuals
Getting started

GAP manuals can be found at

http://www.gap-system.org/Doc/manuals.html

including:

- **Tutorial** for getting started with GAP
- Reference manual
- Package manuals
Starting and Leaving GAP

Start GAP by typing `gap` at the prompt of your operating system followed by the return key.

```
gap
```

GAP shows its own prompt `gap>` asking you for further input.

```
gap>
```

The usual way to end a GAP session is to type `quit;` at the `gap>` prompt. Do not omit the semicolon!

```
gap> quit;
```

$
Starting and Leaving GAP

Start GAP by typing `gap` at the prompt of your operating system followed by the return key

```
$ gap
```
Starting and Leaving GAP

Start GAP by typing `gap` at the prompt of your operating system followed by the return key

```
$ gap
```

GAP shows its own prompt `gap>` asking you for further input

`gap>`
Starting and Leaving GAP

Start GAP by typing `gap` at the prompt of your operating system followed by the return key

```
$  gap
```

GAP shows its own prompt `gap>` asking you for further input

```
gap>
```

The usual way to end a GAP session is to type `quit;` at the `gap>` prompt. Do not omit the semicolon!

```
gap> quit;
$  
```
Loading Source Code from a File

You can load a text file with GAP code into GAP using the `Read` function:

gap> Read("/home/ivgelder/Example.g");

This is useful when creating larger pieces of GAP code!
Loading Source Code from a File

You can load a text file with GAP code into GAP using the `Read` function

```gap
gap> Read("/home/ivgelder/Example.g");
```

This is useful when creating larger pieces of GAP code!
Constants and Operators

GAP can perform basic operations on integers such as subtraction, division, exponentiation, modulo, comparison, ...

\[ \text{gap> 17 - 23;} \]
\[ -6 \]

\[ \text{gap> 12345/25;} \]
\[ 2469/5 \]

\[ \text{gap> 3} \ ^\ 86; \]
\[ 107752636643058178097424660240453423951129 \]

\[ \text{gap> 17 \ mod \ 3;} \]
\[ 2 \]

\[ \text{gap> 10 > 0 \ and \ 10 < 100;} \]
\[ \text{true} \]
Constants and Operators

GAP can perform basic operations on integers such as subtraction, division, exponentiation, modulo, comparison, ...

gap> 17 - 23;
-6
GAP can perform basic operations on integers such as subtraction, division, exponentiation, modulo, comparison, ...

gap> 17 - 23;
-6

gap> 12345/25;
2469/5
GAP can perform basic operations on integers such as subtraction, division, exponentiation, modulo, comparison, ...

gap> 17 - 23;
-6

gap> 12345/25;
2469/5

gap> 3^86;
107752636643058178097424660240453423951129
Constants and Operators

GAP can perform basic operations on integers such as subtraction, division, exponentiation, modulo, comparison, ...

\[
gap> 17 \ - \ 23;
\]
\[-6\]
\[
gap> 12345/25;
\]
\[2469/5\]
\[
gap> 3 \ ^ 86;
\]
\[107752636643058178097424660240453423951129\]
\[
gap> 17 \ mod \ 3;
\]
\[2\]
Constants and Operators

GAP can perform basic operations on integers such as subtraction, division, exponentiation, modulo, comparison, ...

gap> 17 - 23;
-6

gap> 12345/25;
2469/5

gap> 3 ^ 86;
107752636643058178097424660240453423951129

gap> 17 mod 3;
2

gap> 10 > 0 and 10 < 100;
true
Constants and Operators

GAP can also deal with permutations:

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

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

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

```
gap> (1,2,3) ^ (1,2);
(1,3,2)
```
GAP can also deal with permutations

gap> (1,2,3) * (1,2);
(2,3)
gap> (1,2,3) ^ -1;
(1,3,2)
gap> 2 ^ (1,2,3);
3
gap> (1,2,3) ^ (1,2);
(1,3,2)
Assignment

Any GAP object can be assigned to a variable.

```plaintext
gap> a:= (9 - 7) * (5 + 6);
gap> a * (a + 1);
22
506
```
Assignment

Any GAP object can be assigned to a variable

```gap
gap> a := (9 - 7) * (5 + 6);
22
gap> a * (a + 1);
506
```
Assignment

Whenever GAP returns an object by printing it on the next line this object is assigned to the variable last

```
gap> (9 - 7) * (5 + 6);
gap> last * 2;
```

22
44

More variables: last2, last3
Help

For help just type `?topic` at the GAP prompt.
If there are several matches you get an overview.

gap> ?group ring
Help: several entries match this topic - type `?2` to get match [2].

[1] Reference: group ring
For help just type `?topic` at the GAP prompt
If there are several matches you get an overview

gap> ?group ring
Help: several entries match this topic - type ?2 to get match [2]
[1] Reference: group ring
Permutation groups

```plaintext
gap> S8 := Group( (1,2), (1,2,3,4,5,6,7,8) );
Group([ (1,2), (1,2,3,4,5,6,7,8) ])
gap> A8 := DerivedSubgroup( S8 );
Group([ (1,2,3), (2,3,4), (2,4)(3,5), (2,6,4), (2,4)(5,7), (2,8,6,4)(3,5) ])
gap> Size( A8 );
20160
gap> IsAbelian( A8 );
false
```
Permutation groups

Permutation groups are easy to input because we can type their elements

```
gap> S8 := Group( (1,2), (1,2,3,4,5,6,7,8) );
Group([ (1,2), (1,2,3,4,5,6,7,8) ])
gap> A8 := DerivedSubgroup( S8 );
Group([ (1,2,3), (2,3,4), (2,4)(3,5), (2,6,4), (2,4)(5,7), (2,8,6,4)(3,5) ])
gap> Size( A8 );
20160
gap> IsAbelian( A8 );
false
```
Permutation groups

Permutation groups are easy to input because we can type their elements

\begin{verbatim}
gap> S8 := Group( (1,2), (1,2,3,4,5,6,7,8) );
Group([ (1,2), (1,2,3,4,5,6,7,8) ])
gap> A8 := DerivedSubgroup( S8 );
Group([ (1,2,3), (2,3,4), (2,4)(3,5), (2,6,4), (2,4)(5,7), (2,8,6,4)(3,5) ])
\end{verbatim}
Permutation groups

Permutation groups are easy to input because we can type their elements

```gap
gap> S8 := Group( (1,2), (1,2,3,4,5,6,7,8) );
Group([ (1,2), (1,2,3,4,5,6,7,8) ])
gap> A8 := DerivedSubgroup( S8 );
Group([ (1,2,3), (2,3,4), (2,4)(3,5), (2,6,4), (2,4)(5,7), (2,8,6,4)(3,5) ])
gap> Size( A8 );
20160
gap> IsAbelian( A8 );
false
```
Free groups

Free groups are also easy.

```gap
gap> F2 := FreeGroup( "a", "b" );
<free group on the generators [ a, b ]>

gap> a:=F2.1;;b:=F2.2;;
```
Free groups are also easy

```gap
gap> F2 := FreeGroup( "a", "b" );
<free group on the generators [ a, b ]>
gap> a:=F2.1;; b:=F2.2;;
```
Finitely Presented Groups

We can use this to input finitely presented groups

```
gap> F2 := FreeGroup( "a", "b" );;
gap> a:=F2.1;;b:=F2.2;;
gap> G:=F2/[a^3,b^2,b*a*b*a];;
gap> a:=G.1;;b:=G.2;;
gap> Size(G); IsAbelian(G);
6 false
```

```
gap> H:=Subgroup(G,[a]);
gap> Normalizer(G,H);
group([ a, b ])
```
Finitely Presented Groups

We can use this to input finitely presented groups

gap> F2 := FreeGroup( "a", "b" );;
gap> a:=F2.1;;b:=F2.2;;
Finitely Presented Groups

We can use this to input finitely presented groups

```
gap> F2 := FreeGroup( "a", "b" );;
gap> a:=F2.1;;b:=F2.2;;
gap> G:=F2/[a^3,b^2,b*a*b*a];;
gap> a:=G.1;;b:=G.2;;
```
Finitely Presented Groups

We can use this to input finitely presented groups

gap> F2 := FreeGroup( "a", "b" );;
gap> a:=F2.1;; b:=F2.2;;
gap> G:=F2/[aˆ3,bˆ2,b*a*b*a];;
gap> a:=G.1;; b:=G.2;;
gap> Size(G); IsAbelian(G);
6
false
We can use this to input finitely presented groups

```gap
F2 := FreeGroup( "a", "b" );;
a := F2.1;; b := F2.2;;
G := F2/[ a^3, b^2, b*a*b*a ];;
a := G.1;; b := G.2;;
Size(G); IsAbelian(G);
6
false
H := Subgroup(G, [a]);
Group([ a ])
```
Finitely Presented Groups

We can use this to input finitely presented groups

gap> F2 := FreeGroup( "a", "b" );;
gap> a:=F2.1;;b:=F2.2;;
gap> G:=F2/[a^3,b^2,b*a*b*a];;
gap> a:=G.1;;b:=G.2;;
gap> Size(G); IsAbelian(G);
6
false
gap> H:=Subgroup(G,[a]);
Group([ a ])
gap> Normalizer(G,H);
Group([ a, b ])
The SmallGroups Library

gap> AllSmallGroups(6);  
[ <pc group of size 6 with 2 generators>,  
  <pc group of size 6 with 2 generators> ]

gap> AllSmallGroups(6,IsAbelian);  
[ <pc group of size 6 with 2 generators> ]

gap> SmallGroup(892,4);  
<pc group of size 892 with 3 generators>
The SmallGroups Library

The SmallGroups Library contains all groups of certain 'small' orders

```gap
gap> AllSmallGroups(6);
[ <pc group of size 6 with 2 generators>, <pc group of size 6 with 2 generators> ]
```
The SmallGroups Library contains all groups of certain 'small' orders

\begin{verbatim}
gap> AllSmallGroups(6);
[ <pc group of size 6 with 2 generators>,
  <pc group of size 6 with 2 generators> ]
gap> AllSmallGroups(6,IsAbelian);
[ <pc group of size 6 with 2 generators> ]
\end{verbatim}
The SmallGroups Library contains all groups of certain 'small' orders.

```gap
gap> AllSmallGroups(6);
[ <pc group of size 6 with 2 generators>,
  <pc group of size 6 with 2 generators> ]
gap> AllSmallGroups(6,IsAbelian);
[ <pc group of size 6 with 2 generators> ]
gap> SmallGroup(892,4);
<pc group of size 892 with 3 generators>
```
Characters

GAP can easily compute the character table of a group:

```gap
F:=FreeGroup("a","b");; a:=F.1;; b:=F.2;;
G:=F/[a^4,b^2,b*a*b*a];; a:=G.1;; b:=G.2;;
tbl:=CharacterTable(G);
ConjugacyClasses(tbl);
Display(tbl);
```

- `X`.1 1 1 1 1 1
- `X`.2 1 -1 -1 1 1
- `X`.3 1 -1 1 -1
- `X`.4 1 1 -1 -1
- `X`.5 2 . . -2 .
GAP can easily compute the character table of a group

```
gap> F:=FreeGroup("a","b");;a:=F.1;;b:=F.2;;
gap> G:=F/[a^4,b^2,b*a*b*a];;a:=G.1;;b:=G.2;;
```
GAP can easily compute the character table of a group

```gap
F:=FreeGroup("a","b");; a:=F.1;; b:=F.2;;
G:=F/[a^4,b^2,b*a*b*a];; a:=G.1;; b:=G.2;;
tbl:=CharacterTable(G);;
```
GAP can easily compute the character table of a group

```
gap> F:=FreeGroup("a","b");;a:=F.1;;b:=F.2;;
gap> G:=F/[a^4,b^2,b*a*b*a];;a:=G.1;;b:=G.2;;
gap> tbl:=CharacterTable(G);;
gap> ConjugacyClasses(tbl);
[ <identity ...>^G, a^G, b^G, a^2^G, a*b^G ]
gap> Display(tbl);
  X.1  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  .
```
More
GAP also provides

- group actions
- finite fields
- vector spaces
- semigroups
GAP also provides

- group actions
- finite fields
- vector spaces
- semigroups
- group rings (Package Wedderga)
GAP also provides

- group actions
- finite fields
- vector spaces
- semigroups
- group rings (Package Wedderga)
- **programming language**
References