Matrix group recognition in GAP

Max Neunhöffer

University of St Andrews

15.9.2007

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \le \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups
- (Maximal) subgroups?

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups
- (Maximal) subgroups?
- ...

Permutation groups and matrix groups

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

Permutation groups and matrix groups

Let $n \in \mathbb{N}$ and S_n be the symmetric group:

$$S_n = \{\pi : \{1, \dots, n\} \to \{1, \dots, n\} \mid \pi \text{ bijective}\}.$$

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \dots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

Permutation groups and matrix groups

Let $n \in \mathbb{N}$ and S_n be the symmetric group:

$$S_n = \{\pi : \{1, \ldots, n\} \to \{1, \ldots, n\} \mid \pi \text{ bijective}\}.$$

Given: $\pi_1, \ldots, \pi_k \in S_n$

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \le \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

Permutation groups and matrix groups

Let $n \in \mathbb{N}$ and S_n be the symmetric group:

$$S_n = \{\pi : \{1,\ldots,n\} \to \{1,\ldots,n\} \mid \pi \text{ bijective}\}.$$

Given: $\pi_1,\ldots,\pi_k \in S_n$

Then the π_i generate a group $G \le S_n$.

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1,\ldots,M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \le \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2}\prod_{i=1}^{n}(q^i - 1)$

Permutation groups and matrix groups

Let $n \in \mathbb{N}$ and S_n be the symmetric group:

$$S_n = \{\pi : \{1, \ldots, n\} \to \{1, \ldots, n\} \mid \pi \text{ bijective}\}.$$

Given: $\pi_1, \ldots, \pi_k \in S_n$

Then the π_i generate a group $G \leq S_n$.

It is finite, we have $|S_n| = n!$

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

Permutation groups

Let $n \in \mathbb{N}$ and S_n be the symmetric group:

$$S_n = \{\pi : \{1, \ldots, n\} \to \{1, \ldots, n\} \mid \pi \text{ bijective}\}.$$

Given: $\pi_1, \ldots, \pi_k \in S_n$

Then the π_i generate a group $G \leq S_n$.

It is finite, we have $|S_n| = n!$.

We can determine about G algorithmically (e.g.):

- The group order $|G|$
- Membership test: Is $M \in S_n$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups
- (Maximal) subgroups?
- ...

Matrix groups in GAP

In standard GAP:

```
gap> ugens;
[ <an immutable 56x56 matrix over GF2>,
  <an immutable 56x56 matrix over GF2> ]
gap> u := Group(ugens);;
gap> Size(u); time;
252000
341277
gap> Image(NiceMonomorphism(u));
<permutation group with 2 generators>
```

Matrix groups in GAP

In standard GAP:

```
gap> ugens;
[ <an immutable 56x56 matrix over GF2>,
  <an immutable 56x56 matrix over GF2> ]
gap> u := Group(ugens);;
gap> Size(u); time;
252000
341277
gap> Image(NiceMonomorphism(u));
<permutation group with 2 generators>
```

Using the upcoming genss package (with F. Noeske):

```
gap> Size(StabilizerChain(u)); time;
252000
1368
```

Matrix groups in GAP

In standard GAP:

```
gap> ugens;
[ <an immutable 56x56 matrix over GF2>,
  <an immutable 56x56 matrix over GF2> ]
gap> u := Group(ugens);;
gap> Size(u); time;
252000
341277
gap> Image(NiceMonomorphism(u));
<permutation group with 2 generators>
```

Using the upcoming genss package (with F. Noeske):

```
gap> Size(StabilizerChain(u)); time;
252000
1368
```

For “bigger” matrix groups both approaches do not work.

Constructive recognition — first formulation

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$ and
  - if so, expresses M as word in the M_i.

Constructive recognition — first formulation

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$ and
  - if so, expresses M as word in the M_i.

If this problem is solved, we call

$\langle M_1, \ldots, M_k \rangle$ recognised constructively.

Complexity of algorithms

To measure the efficiency of an algorithm, we consider a class $\mathcal{P}$ of problems, that the algorithm can solve.

We assign to each $P \in \mathcal{P}$ its size $g(P)$,

and prove an upper bound for the runtime $L(P)$ of the algorithm for P:

$$L(P) \leq f(g(P))$$

for some function f.

Complexity of algorithms

To measure the efficiency of an algorithm, we consider a class $\mathcal{P}$ of problems, that the algorithm can solve.

We assign to each $P \in \mathcal{P}$ its size $g(P)$,

and prove an upper bound for the runtime $L(P)$ of the algorithm for P:

$$L(P) \leq f(g(P))$$

for some function f.

The growth rate of f measures the complexity.

Complexity of algorithms

To measure the efficiency of an algorithm, we consider a class $\mathcal{P}$ of problems, that the algorithm can solve.

We assign to each $P \in \mathcal{P}$ its size $g(P)$,

and prove an upper bound for the runtime $L(P)$ of the algorithm for P:

$$L(P) \leq f(g(P))$$

for some function f.

The growth rate of f measures the complexity.

Example (Constructive matrix group recognition)

- Problem given by $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$.
- Size determined by n, k and $\log q$.
- Runtime should be $\leq$ a polynomial in n, k and $\log q$.

Randomised algorithms

Randomised algorithms

Definition (Monte Carlo algorithms)

A Monte Carlo algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it returns a wrong result is at most ϵ.

Randomised algorithms

Definition (Monte Carlo algorithms)

A Monte Carlo algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it returns a wrong result is at most ϵ.

Definition (Las Vegas algorithm)

A Las Vegas algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it fails is at most ϵ.

Randomised algorithms

Definition (Monte Carlo algorithms)

A Monte Carlo algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it returns a wrong result is at most ϵ.

Definition (Las Vegas algorithm)

A Las Vegas algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it fails is at most ϵ.

Example: Comp. of $|G| = 4\,089\,470\,473\,293\,004\,800$ for permutation group $G = \langle \pi_1, \pi_2 \rangle$ $(n = 137\,632)$:

Randomised algorithms

Definition (Monte Carlo algorithms)

A Monte Carlo algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it returns a wrong result is at most ϵ.

Definition (Las Vegas algorithm)

A Las Vegas algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it fails is at most ϵ.

Example: Comp. of $|G| = 4\,089\,470\,473\,293\,004\,800$ for permutation group $G = \langle \pi_1, \pi_2 \rangle$ ($n = 137\,632$):

deterministic alg.: 112s

Randomised algorithms

Definition (Monte Carlo algorithms)

A Monte Carlo algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it returns a wrong result is at most ϵ.

Definition (Las Vegas algorithm)

A Las Vegas algorithm with error probability ϵ is an algorithm, that is guaranteed to terminate after a finite time, such that the probability that it fails is at most ϵ.

Example: Comp. of $|G| = 4\,089\,470\,473\,293\,004\,800$ for permutation group $G = \langle \pi_1, \pi_2 \rangle$ ($n = 137\,632$):

deterministic alg.: 112s   Monte Carlo $\epsilon = 1\%$: 6s

Saving: 95% of runtime

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements und

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements und

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements und

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.
- A Monte Carlo Algorithmus is enough.

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements und

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.
- A Monte Carlo Algorithmus is enough. (Verification!)

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements und

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
    - decides, whether or not $M \in G$, and,
    - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.
- A Monte Carlo Algorithmus is enough. (Verification!)

If this problem is solved, we call

$\langle M_1, \ldots, M_k \rangle$ recognised constructively.

Troubles

The discrete logarithm problem

If $M_1 = [z] \in \mathbb{F}_q^{1\times 1}$ with z a primitive root of $\mathbb{F}_q$. Then:

Given $0 \neq [x] \in \mathbb{F}_q^{1\times 1}$, find $i \in \mathbb{N}$ such that $[x] = [z]^i$.

Troubles

The discrete logarithm problem

If $M_1 = [z] \in \mathbb{F}_q^{1\times 1}$ with z a primitive root of $\mathbb{F}_q$. Then:

Given $0 \neq [x] \in \mathbb{F}_q^{1\times 1}$, find $i \in \mathbb{N}$ such that $[x] = [z]^i$.

There is no solution in polynomial time in $\log q$ known!

Troubles

The discrete logarithm problem

If $M_1 = [z] \in \mathbb{F}_q^{1\times 1}$ with z a primitive root of $\mathbb{F}_q$. Then:

Given $0 \neq [x] \in \mathbb{F}_q^{1\times 1}$, find $i \in \mathbb{N}$ such that $[x] = [z]^i$.

There is no solution in polynomial time in $\log q$ known!

Integer factorisation

Some methods need a factorisation of $q^i - 1$ for an $i \leq n$.

Troubles

The discrete logarithm problem

If $M_1 = [z] \in \mathbb{F}_q^{1\times 1}$ with z a primitive root of $\mathbb{F}_q$. Then:

$$\text{Given } 0 \neq [x] \in \mathbb{F}_q^{1\times 1}, \text{ find } i \in \mathbb{N} \text{ such that } [x] = [z]^i.$$

There is no solution in polynomial time in $\log q$ known!

Integer factorisation

Some methods need a factorisation of $q^i - 1$ for an $i \leq n$.

There is no solution in polynomial time in $\log q$ known!

Troubles

The discrete logarithm problem

If $M_1 = [z] \in \mathbb{F}_q^{1\times 1}$ with z a primitive root of $\mathbb{F}_q$. Then:

$$\text{Given } 0 \neq [x] \in \mathbb{F}_q^{1\times 1}, \text{ find } i \in \mathbb{N} \text{ such that } [x] = [z]^i.$$

There is no solution in polynomial time in $\log q$ known!

Integer factorisation

Some methods need a factorisation of $q^i - 1$ for an $i \leq n$.

There is no solution in polynomial time in $\log q$ known!

In practice q is small $\Rightarrow$ no problem.

We ignore both!

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rcccl} \varphi & : & G & \to & H \\ & & M_i & \mapsto & P_i \qquad \text{for all } i \end{array}$$

with the following properties:

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rccc} \varphi : & G & \to & H \\ & M_i & \mapsto & P_i \qquad \text{for all } i \end{array}$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{cccccl} \varphi & : & G & \to & H & \\ & & M_i & \mapsto & P_i & \quad \text{for all } i \end{array}$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- φ is surjective: $H = \langle P_1, \ldots, P_k \rangle$

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \le \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{cccc} \varphi : & G & \to & H \\ & M_i & \mapsto & P_i \end{array} \qquad \text{for all } i$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- φ is surjective: $H = \langle P_1, \ldots, P_k \rangle$
- H is in some sense "smaller"
- or at least "easier to recognise constructively"

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rccc} \varphi : & G & \to & H \\ & M_i & \mapsto & P_i \end{array} \qquad \text{for all } i$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- φ is surjective: $H = \langle P_1, \ldots, P_k \rangle$
- H is in some sense “smaller”
- or at least “easier to recognise constructively”
- e.g. $H \leq S_m$ or $H \leq \mathrm{GL}_{n'}(\mathbb{F}_{q'})$ with $n' \log q' < n \log q$

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
3. express $\varphi(M)$ as word in the P_i,

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
3. express $\varphi(M)$ as word in the P_i,
4. evaluate the same word in the M_i,

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
3. express $\varphi(M)$ as word in the P_i,
4. evaluate the same word in the M_i,
5. get element $M' \in G$ with $M \cdot M'^{-1} \in N$.

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
3. express $\varphi(M)$ as word in the P_i,
4. evaluate the same word in the M_i,
5. get element $M' \in G$ with $M \cdot M'^{-1} \in N$.
6. If M is uniformly distributed in G
   then $M \cdot M'^{-1}$ is uniformly distributed in N

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
3. express $\varphi(M)$ as word in the P_i,
4. evaluate the same word in the M_i,
5. get element $M' \in G$ with $M \cdot M'^{-1} \in N$.
6. If M is uniformly distributed in G then $M \cdot M'^{-1}$ is uniformly distributed in N
7. Repeat.

Computing the kernel

Let $\varphi : G \to H$ be a reduction and assume that H is already recognised constructively.

Then we can compute the kernel N of φ:

1. Generate a (pseudo-) random element $M \in G$,
2. map it with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
3. express $\varphi(M)$ as word in the P_i,
4. evaluate the same word in the M_i,
5. get element $M' \in G$ with $M \cdot M'^{-1} \in N$.
6. If M is uniformly distributed in G then $M \cdot M'^{-1}$ is uniformly distributed in N
7. Repeat.

$\to$ Monte Carlo algorithm to compute N

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$$|G| = |H| \cdot |N|.$$

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$$|G| = |H| \cdot |N|. \text{ And for } M \in \mathrm{GL}_n(\mathbb{F}_q):$$

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
2. express $\varphi(M)$ as word in the P_i,

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
2. express $\varphi(M)$ as word in the P_i,
3. evaluate the same word in the M_i,

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
2. express $\varphi(M)$ as word in the P_i,
3. evaluate the same word in the M_i,
4. get element $M' \in G$ such that $M \cdot M'^{-1} \in N$,

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
2. express $\varphi(M)$ as word in the P_i,
3. evaluate the same word in the M_i,
4. get element $M' \in G$ such that $M \cdot M'^{-1} \in N$,
5. express $M \cdot M'^{-1}$ as word in the N_j,

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
2. express $\varphi(M)$ as word in the P_i,
3. evaluate the same word in the M_i,
4. get element $M' \in G$ such that $M \cdot M'^{-1} \in N$,
5. express $M \cdot M'^{-1}$ as word in the N_j,
6. get M as word in the M_i and N_j:
   $M' = \prod$ in the M_i, $\quad M \cdot M'^{-1} = \prod$ in the N_j
   $\Rightarrow M = \left(\prod \text{ in the } N_j\right) \cdot \left(\prod \text{ in the } M_i\right).$

Recognising image and kernel suffices

Let $\varphi : G \to H$ be a reduction and assume that both H and the kernel $N = \langle N_1, \ldots, N_m \rangle$ of φ are already recognised constructively.

Then we have recognised G constructively:

$|G| = |H| \cdot |N|$. And for $M \in \mathrm{GL}_n(\mathbb{F}_q)$:

1. map M with φ onto $\varphi(M) \in H = \langle P_1, \ldots, P_k \rangle$,
2. express $\varphi(M)$ as word in the P_i,
3. evaluate the same word in the M_i,
4. get element $M' \in G$ such that $M \cdot M'^{-1} \in N$,
5. express $M \cdot M'^{-1}$ as word in the N_j,
6. get M as word in the M_i and N_j:
   $M' = \prod$ in the M_i, $\quad M \cdot M'^{-1} = \prod$ in the N_j
   $$\Rightarrow M = \left(\prod \text{ in the } N_j\right) \cdot \left(\prod \text{ in the } M_i\right).$$
7. If $M \notin G$, then at least one step does not work.

Recursion: composition trees

We get a tree:

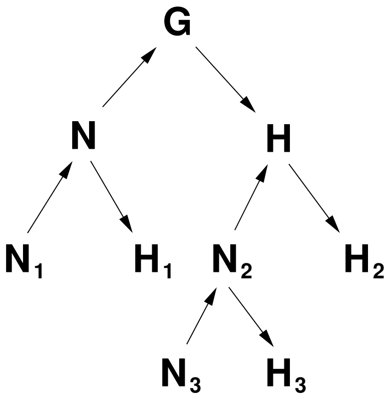
$$\begin{tikzcd} & \mathbf{G} \arrow[dr] & \\ \mathbf{N} \arrow[ur] & & \mathbf{H} \end{tikzcd}$$

Up arrows: inclusions

Down arrows: homomorphisms

Recursion: composition trees

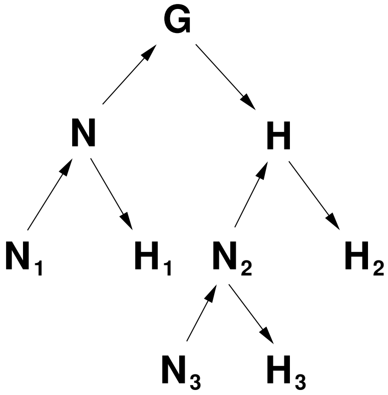
We get a tree:



Up arrows: inclusions
Down arrows: homomorphisms

Recursion: composition trees

We get a tree:



Up arrows: inclusions

Down arrows: homomorphisms

Old idea, substantial improvements: Seress & N. 2006

Example: invariant subspace

Let $V = \mathbb{F}_q^n$, then G acts on V.
Let $W \leq V$ be an invariant subspace, i.e.:

$$MW = W \quad \text{for all } M \in G$$

Example: invariant subspace

Let $V = \mathbb{F}_q^n$, then G acts on V.
Let $W \leq V$ be an invariant subspace, i.e.:

$$MW = W \quad \text{for all } M \in G$$

Choose basis $(w_1, \ldots, w_d)$ of W and extend to a basis

$$(w_1, \ldots, w_d, w_{d+1}, \ldots, w_n)$$

of V. After a base change the matrices in G look like this:

$$\left[\begin{array}{c|c} A & B \\ \hline \mathbf{0} & D \end{array}\right] \qquad \text{with } A \in \mathbb{F}_q^{d \times d}, B \in \mathbb{F}_q^{d \times (n-d)}, D \in \mathbb{F}_q^{(n-d) \times (n-d)}$$

Example: invariant subspace

Let $V = \mathbb{F}_q^n$, then G acts on V.
Let $W \leq V$ be an invariant subspace, i.e.:

$$MW = W \quad \text{for all } M \in G$$

Choose basis $(w_1, \ldots, w_d)$ of W and extend to a basis

$$(w_1, \ldots, w_d, w_{d+1}, \ldots, w_n)$$

of V. After a base change the matrices in G look like this:

$$\left[\begin{array}{c|c} A & B \\ \hline \mathbf{0} & D \end{array}\right] \qquad \text{with } A \in \mathbb{F}_q^{d \times d}, B \in \mathbb{F}_q^{d \times (n-d)}, D \in \mathbb{F}_q^{(n-d)\times(n-d)}$$

and

$$G \to \mathrm{GL}_{n-d}(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \mapsto D$$

is a homomorphism of groups.

Example: invariant subspace

$$G \to \mathrm{GL}_{n-d}(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \mapsto D$$

is a homomorphism of groups, its kernel is

$$N := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid D = \mathbf{1} \right\}.$$

Example: invariant subspace

$$G \to \mathrm{GL}_{n-d}(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \mapsto D$$

is a homomorphism of groups, its kernel is

$$N := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid D = \mathbf{1} \right\}.$$

The mapping

$$N \to \mathrm{GL}_d(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mapsto A$$

also is a homomorphism of groups and has kernel

$$N_2 := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid A = D = \mathbf{1} \right\}.$$

Example: invariant subspace

$$G \to \mathrm{GL}_{n-d}(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \mapsto D$$

is a homomorphism of groups, its kernel is

$$N := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid D = \mathbf{1} \right\}.$$

The mapping

$$N \to \mathrm{GL}_d(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mapsto A$$

also is a homomorphism of groups and has kernel

$$N_2 := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid A = D = \mathbf{1} \right\}.$$

This group is a p-group for $q = p^e$:

$$\begin{bmatrix} \mathbf{1} & B \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1} & B' \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & B + B' \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Example: invariant subspace

$$G \to \mathrm{GL}_{n-d}(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \mapsto D$$

is a homomorphism of groups, its kernel is

$$N := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid D = \mathbf{1} \right\}.$$

The mapping

$$N \to \mathrm{GL}_d(\mathbb{F}_q), \begin{bmatrix} A & B \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mapsto A$$

also is a homomorphism of groups and has kernel

$$N_2 := \left\{ \begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix} \in G \mid A = D = \mathbf{1} \right\}.$$

This group is a p-group for $q = p^e$:

$$\begin{bmatrix} \mathbf{1} & B \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1} & B' \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & B + B' \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Together with a reduction additional information is gained!

How to find reductions?

How to find reductions?

Aschbacher has defined classes C1 to C8 of subgroups of $\mathrm{GL}_n(\mathbb{F}_q)$.

How to find reductions?

Aschbacher has defined classes C1 to C8 of subgroups of $\mathrm{GL}_n(\mathbb{F}_q)$.

Theorem (Aschbacher, 1984)

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ and $Z := G \cap Z(\mathrm{GL}_n(\mathbb{F}_q))$ the subgroup of scalar matrices. Then G lies in at least one of the classes C1 to C8 or we have:

- *$T \subseteq G/Z \subseteq Aut(T)$*
  for a non-abelian simple group T, and
- *G acts absolutely irreducibly on $V = \mathbb{F}_q^n$.*

How to find reductions?

Aschbacher has defined classes C1 to C8 of subgroups of $\mathrm{GL}_n(\mathbb{F}_q)$.

Theorem (Aschbacher, 1984)

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ and $Z := G \cap Z(\mathrm{GL}_n(\mathbb{F}_q))$ the subgroup of scalar matrices. Then G lies in at least one of the classes C1 to C8 or we have:

- *$T \subseteq G/Z \subseteq Aut(T)$*
  for a non-abelian simple group T, and
- *G acts absolutely irreducibly on $V = \mathbb{F}_q^n$.*

(This last case is called C9.)

How to find reductions?

Aschbacher has defined classes C1 to C8 of subgroups of $\mathrm{GL}_n(\mathbb{F}_q)$.

Theorem (Aschbacher, 1984)

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ and $Z := G \cap Z(\mathrm{GL}_n(\mathbb{F}_q))$ the subgroup of scalar matrices. Then G lies in at least one of the classes C1 to C8 or we have:

- *$T \subseteq G/Z \subseteq Aut(T)$ for a non-abelian simple group T, and*
- *G acts absolutely irreducibly on $V = \mathbb{F}_q^n$.*

(This last case is called C9.)

Thus we can call in heavy artillery:

- the classification of finite simple groups

How to find reductions?

Aschbacher has defined classes C1 to C8 of subgroups of $\mathrm{GL}_n(\mathbb{F}_q)$.

Theorem (Aschbacher, 1984)

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ and $Z := G \cap Z(\mathrm{GL}_n(\mathbb{F}_q))$ the subgroup of scalar matrices. Then G lies in at least one of the classes C1 to C8 or we have:

- *$T \subseteq G/Z \subseteq \mathrm{Aut}(T)$ for a non-abelian simple group T, and*
- *G acts absolutely irreducibly on $V = \mathbb{F}_q^n$.*

(This last case is called C9.)

Thus we can call in heavy artillery:

- the classification of finite simple groups
- the modular representation theory of simple groups

Approach for leaves of the tree

If none of the algorithms for C1 to C8 has succeeded:

Approach for leaves of the tree

If none of the algorithms for C1 to C8 has succeeded:

1. For "small" groups compute direct isomorphism onto a permutation group.

Approach for leaves of the tree

If none of the algorithms for C1 to C8 has succeeded:

1. For "small" groups compute direct isomorphism onto a permutation group.
2. Determine, for which (simple) group $T \leq G/Z \leq \mathrm{Aut}(T)$ holds.

Approach for leaves of the tree

If none of the algorithms for C1 to C8 has succeeded:

1. For "small" groups compute direct isomorphism onto a permutation group.
2. Determine, for which (simple) group $T \leq G/Z \leq \mathrm{Aut}(T)$ holds.
3. Find an explicit isomorphism onto a "standard copy" of an intermediate group S.

Approach for leaves of the tree

If none of the algorithms for C1 to C8 has succeeded:

1. For "small" groups compute direct isomorphism onto a permutation group.
2. Determine, for which (simple) group $T \leq G/Z \leq \mathrm{Aut}(T)$ holds.
3. Find an explicit isomorphism onto a "standard copy" of an intermediate group S.
4. Finally use information about S to recognise G constructively.

Approach for leaves of the tree

If none of the algorithms for C1 to C8 has succeeded:

1. For "small" groups compute direct isomorphism onto a permutation group.
2. Determine, for which (simple) group $T \leq G/Z \leq \mathrm{Aut}(T)$ holds.
3. Find an explicit isomorphism onto a "standard copy" of an intermediate group S.
4. Finally use information about S to recognise G constructively.

This uses:

- the classification of finite simple groups
- information about their automorphism groups
- information about element orders
- information about conjugacy classes
- classifications of the irreducible representations
- information about the subgroup structure

Non-constructive recognition

Methods for non-constructive recognition:

- Knowledge about representations narrows down the possibilities

Non-constructive recognition

Methods for non-constructive recognition:

- Knowledge about representations narrows down the possibilities
- Statistics about orders of random elements

Non-constructive recognition

Methods for non-constructive recognition:

- Knowledge about representations narrows down the possibilities
- Statistics about orders of random elements

Usually this leads to Monte Carlo algorithms.

Standard generators

In G we can only multiply, invert and compute orders.

Standard generators

In G we can only multiply, invert and compute orders.

Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

- $S = \langle s_1, \ldots, s_r \rangle$,

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

- $S = \langle s_1, \ldots, s_r \rangle$,
- if $(s'_1, \ldots, s'_r) \in S^r$ with
  - $|s_i| = |s'_i|$ for $1 \leq i \leq r$,
  - $|p_j| = |p'_j|$ for $1 \leq j \leq m$
    (the p'_j are the same words in the s'_i),

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

- $S = \langle s_1, \ldots, s_r \rangle$,
- if $(s'_1, \ldots, s'_r) \in S^r$ with
  - $|s_i| = |s'_i|$ for $1 \leq i \leq r$,
  - $|p_j| = |p'_j|$ for $1 \leq j \leq m$
    (the p'_j are the same words in the s'_i),

  then $s_i \mapsto s'_i$ for $1 \leq i \leq r$ defines an automorphism of S.

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

- $S = \langle s_1, \ldots, s_r \rangle$,
- if $(s'_1, \ldots, s'_r) \in S^r$ with
  - $|s_i| = |s'_i|$ for $1 \leq i \leq r$,
  - $|p_j| = |p'_j|$ for $1 \leq j \leq m$
    (the p'_j are the same words in the s'_i),

  then $s_i \mapsto s'_i$ for $1 \leq i \leq r$ defines an automorphism of S.

Such elements are called "standard generators" of S.

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

- $S = \langle s_1, \ldots, s_r \rangle$,
- if $(s'_1, \ldots, s'_r) \in S^r$ with
    - $|s_i| = |s'_i|$ for $1 \leq i \leq r$,
    - $|p_j| = |p'_j|$ for $1 \leq j \leq m$
      (the p'_j are the same words in the s'_i),

  then $s_i \mapsto s'_i$ for $1 \leq i \leq r$ defines an automorphism of S.

Such elements are called "standard generators" of S.

We find $G \cong S$ explicitly by finding a tuple $(M_1, \ldots, M_r)$ of standard generators in G.

Standard generators

In G we can only multiply, invert and compute orders.
Suppose: $G \cong S$ with $T \leq S \leq \mathrm{Aut}(T)$ and T simple.

Find a tuple $(s_1, \ldots, s_r) \in S^r$ together with certain words $p_1, \ldots, p_m$ in the s_i, such that:

- $S = \langle s_1, \ldots, s_r \rangle$,
- if $(s'_1, \ldots, s'_r) \in S^r$ with
  - $|s_i| = |s'_i|$ for $1 \leq i \leq r$,
  - $|p_j| = |p'_j|$ for $1 \leq j \leq m$
    (the p'_j are the same words in the s'_i),

  then $s_i \mapsto s'_i$ for $1 \leq i \leq r$ defines an automorphism of S.

Such elements are called "standard generators" of S.

We find $G \cong S$ explicitly by finding a tuple $(M_1, \ldots, M_r)$ of standard generators in G.

Often this leads to efficient Las Vegas algorithms to find explicit isomorphisms.

Verification

Everywhere we used randomised methods:
Las Vegas and Monte Carlo.

$\Rightarrow$ We have to check whether our result is correct!

Verification

Everywhere we used randomised methods:

Las Vegas and Monte Carlo.

$\Rightarrow$ We have to check whether our result is correct!

Idea:

- Find (short) presentations for the leaf-groups,

Verification

Everywhere we used randomised methods:
Las Vegas and Monte Carlo.

$\Rightarrow$ We have to check whether our result is correct!

Idea:

- Find (short) presentations for the leaf-groups,
- put these together to one for the whole group.

Verification

Everywhere we used randomised methods:

Las Vegas and Monte Carlo.

$\Rightarrow$ We have to check whether our result is correct!

Idea:

- Find (short) presentations for the leaf-groups,
- put these together to one for the whole group.
- Check the relations and thus prove the result.

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively,

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively, Authors (currently): P. Brooksbank, M. Law, S. Linton, N., A. Niemeyer, E. O'Brien, Á. Seress,

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively, Authors (currently): P. Brooksbank, M. Law, S. Linton, N., A. Niemeyer, E. O'Brien, Á. Seress,
- complete asymptotically best methods to handle permutation groups,

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively, Authors (currently): P. Brooksbank, M. Law, S. Linton, N., A. Niemeyer, E. O'Brien, Á. Seress,
- complete asymptotically best methods to handle permutation groups,
- methods for most Aschbacher classes for matrix groups and projective groups (some improved algorithms still needed),

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively, Authors (currently): P. Brooksbank, M. Law, S. Linton, N., A. Niemeyer, E. O'Brien, Á. Seress,
- complete asymptotically best methods to handle permutation groups,
- methods for most Aschbacher classes for matrix groups and projective groups (some improved algorithms still needed),
- nearly ready non-constructive recognition,

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively, Authors (currently): P. Brooksbank, M. Law, S. Linton, N., A. Niemeyer, E. O'Brien, Á. Seress,
- complete asymptotically best methods to handle permutation groups,
- methods for most Aschbacher classes for matrix groups and projective groups (some improved algorithms still needed),
- nearly ready non-constructive recognition,
- a few leaf methods,

Status of our implementation

We have

- a package recogbase providing a framework to implement recognition algorithms and composition trees (Ákos Seress, N.),
- a package recog collecting methods to find reductions and recognise leafs constructively, Authors (currently): P. Brooksbank, M. Law, S. Linton, N., A. Niemeyer, E. O'Brien, Á. Seress,
- complete asymptotically best methods to handle permutation groups,
- methods for most Aschbacher classes for matrix groups and projective groups (some improved algorithms still needed),
- nearly ready non-constructive recognition,
- a few leaf methods,
- no verification.