The Cohomology of finite p–Groups

Simon King (NUIG)

Joint work with David J. Green

DFG project GR 1585/4–1
Friedrich–Schiller–Universität Jena

July 9, 2009

Outline

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)
- $\phi : G_1 \longrightarrow G_2 \rightsquigarrow \phi^* : H^*(G_2) \longrightarrow H^*(G_1)$, induced homomorphism

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)
- $\phi : G_1 \longrightarrow G_2 \rightsquigarrow \phi^* : H^*(G_2) \longrightarrow H^*(G_1)$,
  induced homomorphism $\rightsquigarrow$ restriction to subgroups of G.

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)
- $\phi : G_1 \longrightarrow G_2 \rightsquigarrow \phi^* : H^*(G_2) \longrightarrow H^*(G_1)$, induced homomorphism $\rightsquigarrow$ restriction to subgroups of G.
- G determines $H^*(G)$ up to isomorphism.

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)
- $\phi : G_1 \to G_2 \rightsquigarrow \phi^* : H^*(G_2) \to H^*(G_1)$, induced homomorphism $\rightsquigarrow$ restriction to subgroups of G.
- G determines $H^*(G)$ up to isomorphism.

Wanted: Minimal generating set, algebraic relations,

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)
- $\phi : G_1 \to G_2 \rightsquigarrow \phi^* : H^*(G_2) \to H^*(G_1)$, induced homomorphism $\rightsquigarrow$ restriction to subgroups of G.
- G determines $H^*(G)$ up to isomorphism.

Wanted: Minimal generating set, algebraic relations, depth, Poincaré series, a–invariants, ...

Group Cohomology

G: finite p-group (p prime, $|G| = p^n$). Compute $H^*(G;\mathbb{F}_p)$

- Finitely presented graded $\mathbb{F}_p$–algebra
- Graded commutative (commutative if $p = 2$)
- $\phi : G_1 \to G_2 \rightsquigarrow \phi^* : H^*(G_2) \to H^*(G_1)$,
  induced homomorphism $\rightsquigarrow$ restriction to subgroups of G.
- G determines $H^*(G)$ up to isomorphism.

Wanted: Minimal generating set, algebraic relations,
depth, Poincaré series, a–invariants, ...

J. F. Carlson, 1997-2001 (?)

Cohomology of all 267 groups of order 64 ($\sim$ 8 months / run)

Project "Computational Group Cohomology", 2007–2009

http://users.minet.uni-jena.de/~king/cohomology/
http://sage.math.washington.edu/home/SimonKing/Cohomology/

Project "Computational Group Cohomology", 2007–2009

http://users.minet.uni-jena.de/~king/cohomology/
http://sage.math.washington.edu/home/SimonKing/Cohomology/

Performance of our software (latest version)

Project "Computational Group Cohomology", 2007–2009

`http://users.minet.uni-jena.de/~king/cohomology/`
`http://sage.math.washington.edu/home/SimonKing/Cohomology/`

Performance of our software (latest version)

- 267 groups of order 64:
  $\sim$ 23 min CPU / $\sim$ 34 min real (Intel Pentium M, 1.73 GHz)

Project "Computational Group Cohomology", 2007–2009

`http://users.minet.uni-jena.de/~king/cohomology/`
`http://sage.math.washington.edu/home/SimonKing/Cohomology/`

Performance of our software (latest version)

- 267 groups of order 64:
  $\sim$ 23 min CPU / $\sim$ 34 min real (Intel Pentium M, 1.73 GHz)
- $<$ 2 months real time for the 2328 groups of order 128
  (Intel Xeon X7460 with 2.66 GHz, $<$ 4 Gb).

Project "Computational Group Cohomology", 2007–2009

`http://users.minet.uni-jena.de/~king/cohomology/`
`http://sage.math.washington.edu/home/SimonKing/Cohomology/`

Performance of our software (latest version)

- 267 groups of order 64:
  $\sim$ 23 min CPU / $\sim$ 34 min real (Intel Pentium M, 1.73 GHz)
- $<$ 2 months real time for the 2328 groups of order 128
  (Intel Xeon X7460 with 2.66 GHz, $<$ 4 Gb).
  First version needed $\sim$ 10 months.

Project "Computational Group Cohomology", 2007–2009

`http://users.minet.uni-jena.de/~king/cohomology/`
`http://sage.math.washington.edu/home/SimonKing/Cohomology/`

Performance of our software (latest version)

- 267 groups of order 64:
  $\sim$ 23 min CPU / $\sim$ 34 min real (Intel Pentium M, 1.73 GHz)
- $<$ 2 months real time for the 2328 groups of order 128
  (Intel Xeon X7460 with 2.66 GHz, $<$ 4 Gb).
  First version needed $\sim$ 10 months.
- (known) Sylow–2 of the Higman-Sims group (order 512),

Project "Computational Group Cohomology", 2007–2009

`http://users.minet.uni-jena.de/~king/cohomology/`
`http://sage.math.washington.edu/home/SimonKing/Cohomology/`

Performance of our software (latest version)

- 267 groups of order 64:
  $\sim$ 23 min CPU / $\sim$ 34 min real (Intel Pentium M, 1.73 GHz)
- $<$ 2 months real time for the 2328 groups of order 128
  (Intel Xeon X7460 with 2.66 GHz, $<$ 4 Gb).
  First version needed $\sim$ 10 months.
- (known) Sylow–2 of the Higman-Sims group (order 512),
- (new) Sylow–2 of the third Conway group (order 1024).

Project "Computational Group Cohomology", 2007–2009

`http://users.minet.uni-jena.de/~king/cohomology/`
`http://sage.math.washington.edu/home/SimonKing/Cohomology/`

Performance of our software (latest version)

- 267 groups of order 64:
  $\sim$ 23 min CPU / $\sim$ 34 min real (Intel Pentium M, 1.73 GHz)
- $<$ 2 months real time for the 2328 groups of order 128
  (Intel Xeon X7460 with 2.66 GHz, $<$ 4 Gb).
  First version needed $\sim$ 10 months.
- (known) Sylow–2 of the Higman-Sims group (order 512),
- (new) Sylow–2 of the third Conway group (order 1024).
- We also have all but 6 cohomology rings for 3-, 5-, and 7-groups of order at most 625

Approaches for computing Cohomology

Spectral Sequences

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Projective resolutions

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Projective resolutions

Yields approximation in increasing degree

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Projective resolutions

Yields approximation in increasing degree

Main problem: When is the computation finished?

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Projective resolutions

Yields approximation in increasing degree

Main problem: When is the computation finished?

- Carlson's Completeness Criterion (depends on a conjecture)

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Projective resolutions

Yields approximation in increasing degree

Main problem: When is the computation finished?

- Carlson's Completeness Criterion (depends on a conjecture)
- Use spectral sequences ($\rightsquigarrow$ HAP [G. Ellis 2006-2008])

Approaches for computing Cohomology

Spectral Sequences

- Lyndon–Hochschild–Serre
  $\rightsquigarrow$ extraspecial 2–groups [D. Quillen, 1971]
- Eilenberg–Moore $\rightsquigarrow$ groups of order 32 [D. J. Rusin, 1989]

Projective resolutions

Yields approximation in increasing degree
Main problem: When is the computation finished?

- Carlson's Completeness Criterion (depends on a conjecture)
- Use spectral sequences ($\rightsquigarrow$ HAP [G. Ellis 2006-2008])
- Benson's Completeness Criterion (see below)

Constructing minimal projective resolutions

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_p G$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3 C_3 \cong \mathbb{F}_3[t]/\langle t^3 \rangle$,

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$.

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$. Type II rule: $t\cdot a \rightsquigarrow t^2\cdot a - t\cdot b$.

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$. Type II rule: $t\cdot a \rightsquigarrow t^2\cdot a - t\cdot b$.
Reduce $t\cdot a + t\cdot b$:

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$. Type II rule: $t\cdot a \rightsquigarrow t^2\cdot a - t\cdot b$.
Reduce $t\cdot a + t\cdot b$: $\rightsquigarrow t^2\cdot a - t\cdot b + t\cdot b = t^2\cdot a$

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$. Type II rule: $t\cdot a \rightsquigarrow t^2\cdot a - t\cdot b$.
Reduce $t\cdot a + t\cdot b$: $\rightsquigarrow t^2\cdot a - t\cdot b + t\cdot b = t^2\cdot a$
$\rightsquigarrow t^3\cdot a - t^2\cdot b$

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$. Type II rule: $t\cdot a \rightsquigarrow t^2\cdot a - t\cdot b$.
Reduce $t\cdot a + t\cdot b$: $\rightsquigarrow t^2\cdot a - t\cdot b + t\cdot b = t^2\cdot a$
$\rightsquigarrow t^3\cdot a - t^2\cdot b \rightsquigarrow -t^2\cdot b$ (Type I precedes Type II!).

Constructing minimal projective resolutions

D. Green 2001

Use Gröbner / starndard basis techniques for *finite* $\mathbb{F}_pG$*–modules*.

- *Negative* monomial orders (for minimality)
- *Two-speed* replacement rules: *Type I* precedes *Type II*

$\mathbb{F}_3C_3 \cong \mathbb{F}_3[t]/\langle t^3\rangle$, $M = (\mathbb{F}_3C_3 \cdot a \oplus \mathbb{F}_3C_3 \cdot b)/\langle t\cdot a - t^2\cdot a + t\cdot b\rangle$

Type I rule: $t^3 \rightsquigarrow 0$. Type II rule: $t\cdot a \rightsquigarrow t^2\cdot a - t\cdot b$.
Reduce $t\cdot a + t\cdot b$: $\rightsquigarrow t^2\cdot a - t\cdot b + t\cdot b = t^2\cdot a$
$\rightsquigarrow t^3\cdot a - t^2\cdot b \rightsquigarrow -t^2\cdot b$ (Type I precedes Type II!).

Existence and uniqueness of reductions, Gröbner bases, computing kernels of homomorphisms...

Finding relations

Assume we know the cohomology out to degree n;

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.
- $I_n \subset R_n$, generated by degree–$\leq n$–part of $\ker(R_n \longrightarrow H^*(G))$.

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.
- $I_n \subset R_n$, generated by degree–$\leq n$–part of $\ker(R_n \longrightarrow H^*(G))$.

Compute the next degree as follows:

- Standard monomials of I_n of degree $n+1$

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.
- $I_n \subset R_n$, generated by degree–$\leq n$–part of $\ker(R_n \longrightarrow H^*(G))$.

Compute the next degree as follows:

- Standard monomials of I_n of degree $n+1$
  $\rightsquigarrow$ decomposable $(n+1)$–classes in cohomology.

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.
- $I_n \subset R_n$, generated by degree–$\leq n$–part of $\ker(R_n \longrightarrow H^*(G))$.

Compute the next degree as follows:

- Standard monomials of I_n of degree $n+1$
  $\rightsquigarrow$ decomposable $(n+1)$–classes in cohomology.
- Find new relations in degree $n+1$ $\quad\rightsquigarrow I_{n+1}$.

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.
- $I_n \subset R_n$, generated by degree–$\leq n$–part of $\ker(R_n \longrightarrow H^*(G))$.

Compute the next degree as follows:

- Standard monomials of I_n of degree $n+1$
  $\rightsquigarrow$ decomposable $(n+1)$–classes in cohomology.
- Find new relations in degree $n+1$ $\rightsquigarrow I_{n+1}$.
- Indecomposable classes

Finding relations

Assume we know the cohomology out to degree n; hence:

- R_n: Free graded–commutative algebra over $\mathbb{F}_p$, given by minimal generators of $H^*(G;\mathbb{F}_p)$ of degree $\leq n$.
- $I_n \subset R_n$, generated by degree–$\leq n$–part of $\ker(R_n \longrightarrow H^*(G))$.

Compute the next degree as follows:

- Standard monomials of I_n of degree $n+1$
  $\rightsquigarrow$ decomposable $(n+1)$–classes in cohomology.
- Find new relations in degree $n+1$ $\quad\rightsquigarrow I_{n+1}$.
- Indecomposable classes $\rightsquigarrow$ new generators $\rightsquigarrow R_{n+1}$

Special choice of generators

Special choice of generators

1. Nilpotent generators

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)
2. "Boring" generators:

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)
2. "Boring" generators: Not nilpotent, but restriction to the greatest central elem. ab. subgroup is nilpotent.

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)
2. "Boring" generators: Not nilpotent, but restriction to the greatest central elem. ab. subgroup is nilpotent.
3. Remaining: *Duflot regular* generators

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)
2. "Boring" generators: Not nilpotent, but restriction to the greatest central elem. ab. subgroup is nilpotent.
3. Remaining: *Duflot regular* generators

Reason for that choice of generators

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)
2. "Boring" generators: Not nilpotent, but restriction to the greatest central elem. ab. subgroup is nilpotent.
3. Remaining: *Duflot regular* generators

Reason for that choice of generators

David Green's monomial order on R_n relies on the generator types. It simplifies the computations by magic!

Special choice of generators

1. Nilpotent generators, *i.e.*, Restrictions to all maximal elem. ab. subgroups of G are nilpotent (easy to test!)
2. "Boring" generators: Not nilpotent, but restriction to the greatest central elem. ab. subgroup is nilpotent.
3. Remaining: *Duflot regular* generators

Reason for that choice of generators

David Green's monomial order on R_n relies on the generator types.
It simplifies the computations by magic!
Also, it is used in the completeness criterion below.

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$
- Let $P_1, \ldots, P_r \in R_n/I_n$ be a **filter-regular** HSOP, $\deg(P_i) \geq 2$

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$
- Let $P_1, \ldots, P_r \in R_n/I_n$ be a **filter-regular** HSOP, $\deg(P_i) \geq 2$
  I.e., the multiplication by P_i on $R_n/\left(I_n + \langle P_1, ..., P_{i-1}\rangle\right)$ has finite kernel, for $i = 1, ..., r$.

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$
- Let $P_1, \ldots, P_r \in R_n/I_n$ be a **filter-regular** HSOP, $\deg(P_i) \geq 2$
  I.e., the multiplication by P_i on $R_n/\left(I_n + \langle P_1, ..., P_{i-1}\rangle\right)$ has finite kernel, for $i = 1, ..., r$.
- Maximal degrees of the kernels and of $R_n/\left(I_n + \langle P_1, ..., P_r\rangle\right)$ $\rightsquigarrow$ "filter degree type" $(d_1, ..., d_{r+1})$ (after easy computation)

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$
- Let $P_1, \ldots, P_r \in R_n/I_n$ be a **filter-regular** HSOP, $\deg(P_i) \geq 2$
  I.e., the multiplication by P_i on $R_n/\left(I_n + \langle P_1, ..., P_{i-1}\rangle\right)$ has finite kernel, for $i = 1, ..., r$.
- Maximal degrees of the kernels and of $R_n/\left(I_n + \langle P_1, ..., P_r\rangle\right)$ $\rightsquigarrow$ "filter degree type" $(d_1, ..., d_{r+1})$ (after easy computation)

Theorem [D. J. Benson, 2004]

$$n > \max\left(0, d_i + i - 1\right)_{i=1,\ldots,r} + \sum_i \deg(P_i) - r \quad \Rightarrow \quad \boxed{R_n/I_n \cong H^*(G)}$$

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$
- Let $P_1, \ldots, P_r \in R_n/I_n$ be a **filter-regular** HSOP, $\deg(P_i) \geq 2$
  I.e., the multiplication by P_i on $R_n/\left(I_n + \langle P_1, ..., P_{i-1}\rangle\right)$ has finite kernel, for $i = 1, ..., r$.
- Maximal degrees of the kernels and of $R_n/\left(I_n + \langle P_1, ..., P_r\rangle\right)$ $\rightsquigarrow$ "filter degree type" $(d_1, ..., d_{r+1})$ (after easy computation)

Theorem [D. J. Benson, 2004]

$n > \max(0, d_i + i - 1)_{i=1,\ldots,r} + \sum_i \deg(P_i) - r \quad \Rightarrow \quad \boxed{R_n/I_n \cong H^*(G)}$

Remark "$n \geq ...$" suffices if $\mathrm{rk}(Z(G)) \geq 2$

Benson's Completeness Criterion

G abelian / gen. quaternion $\Rightarrow$ degree 2 / 4 suffices. Otherwise:

- Let r be the p-Rank of G and let R_n/I_n approximate $H^*(G)$
- Let $P_1, \ldots, P_r \in R_n/I_n$ be a **filter-regular** HSOP, $\deg(P_i) \geq 2$
  I.e., the multiplication by P_i on $R_n/\left(I_n + \langle P_1, ..., P_{i-1}\rangle\right)$ has finite kernel, for $i = 1, ..., r$.
- Maximal degrees of the kernels and of $R_n/\left(I_n + \langle P_1, ..., P_r\rangle\right)$ $\rightsquigarrow$ "filter degree type" $(d_1, ..., d_{r+1})$ (after easy computation)

Theorem [D. J. Benson, 2004]

$n > \max(0, d_i + i - 1)_{i=1,\ldots,r} + \sum_i \deg(P_i) - r \quad \Rightarrow \quad \boxed{R_n/I_n \cong H^*(G)}$

Remark "$n \geq ...$" suffices if $\mathrm{rk}(Z(G)) \geq 2$

Conj. If $R_n/I_n \cong H^*(G)$ then $(d_1, ..., d_{r+1}) = (-1, -2, ..., -r, -r)$

Constructing a filter-regular HSOP

- Let $r = p\text{-rk}(G)$ and $z = \text{rk}(Z(G))$.
  Find **Duflot regular** generators $g_1, ..., g_z \in H^*(G)$.

Constructing a filter-regular HSOP

- Let $r = p\text{-rk}(G)$ and $z = \text{rk}(Z(G))$.
  Find **Duflot regular** generators $g_1, ..., g_z \in H^*(G)$.
- Let $U_1, ..., U_m \subset G$ be the maximal elementary abelian subgroups.
  Using **Dickson invariants**: Compute classes $D_{i,j}$ in the polynomial part of $H^*(U_j)$, for $i = 1, ..., r - z$ and $j = 1, ..., m$.

Constructing a filter-regular HSOP

- Let $r = p\text{-rk}(G)$ and $z = \text{rk}(Z(G))$.
  Find **Duflot regular** generators $g_1, ..., g_z \in H^*(G)$.
- Let $U_1, ..., U_m \subset G$ be the maximal elementary abelian subgroups.
  Using **Dickson invariants**: Compute classes $D_{i,j}$ in the polynomial part of $H^*(U_j)$, for $i = 1, ..., r-z$ and $j = 1, ..., m$.
- There are classes $\Delta_i \in H^*(G)$ for $i = 1, ..., r-z$, simultaneously restricting to the p^{k_i}-th power of $D_{i,j}$ for $j = 1, ..., m$.

Constructing a filter-regular HSOP

- Let $r = p\text{-rk}(G)$ and $z = \text{rk}(Z(G))$.
  Find **Duflot regular** generators $g_1, ..., g_z \in H^*(G)$.
- Let $U_1, ..., U_m \subset G$ be the maximal elementary abelian subgroups.
  Using **Dickson invariants**: Compute classes $D_{i,j}$ in the polynomial part of $H^*(U_j)$, for $i = 1, ..., r - z$ and $j = 1, ..., m$.
- There are classes $\Delta_i \in H^*(G)$ for $i = 1, ..., r - z$, simultaneously restricting to the p^{k_i}-th power of $D_{i,j}$ for $j = 1, ..., m$.
  Very often, $k_i = 0$. Allow modification by nilpotent elements

Constructing a filter-regular HSOP

- Let $r = p\text{-rk}(G)$ and $z = \text{rk}(Z(G))$.
  Find **Duflot regular** generators $g_1, ..., g_z \in H^*(G)$.
- Let $U_1, ..., U_m \subset G$ be the maximal elementary abelian subgroups.
  Using **Dickson invariants**: Compute classes $D_{i,j}$ in the polynomial part of $H^*(U_j)$, for $i = 1, ..., r-z$ and $j = 1, ..., m$.
- There are classes $\Delta_i \in H^*(G)$ for $i = 1, ..., r-z$, simultaneously restricting to the p^{k_i}-th power of $D_{i,j}$ for $j = 1, ..., m$.
  Very often, $k_i = 0$. Allow modification by nilpotent elements
- $g_1, ..., g_z, \Delta_1, ..., \Delta_{r-z}$ is a filter-regular HSOP of $H^*(G)$ [consequence of D. J. Benson's results].

Constructing a filter-regular HSOP

- Let $r = p\text{-rk}(G)$ and $z = \text{rk}(Z(G))$.
  Find **Duflot regular** generators $g_1, ..., g_z \in H^*(G)$.
- Let $U_1, ..., U_m \subset G$ be the maximal elementary abelian subgroups.
  Using **Dickson invariants**: Compute classes $D_{i,j}$ in the polynomial part of $H^*(U_j)$, for $i = 1, ..., r-z$ and $j = 1, ..., m$.
- There are classes $\Delta_i \in H^*(G)$ for $i = 1, ..., r-z$, simultaneously restricting to the p^{k_i}-th power of $D_{i,j}$ for $j = 1, ..., m$.
  Very often, $k_i = 0$. Allow modification by nilpotent elements
- $g_1, ..., g_z, \Delta_1, ..., \Delta_{r-z}$ is a filter-regular HSOP of $H^*(G)$ [consequence of D. J. Benson's results].
  Computable in R_n/I_n !!

Improvement by existence proof

Problem: Dickson invariants may be of large degree

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.
- We use the Dickson classes *only* for computing the filter degree type

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.
- We use the Dickson classes *only* for computing the filter degree type (which is the same for any f. r. HSOP)!

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.
- We use the Dickson classes *only* for computing the filter degree type (which is the same for any f. r. HSOP)!
- We prove the presence of a small–degree filter–regular HSOP:

Lemma [D. J. Green, S. K. 2008/9]

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.
- We use the Dickson classes *only* for computing the filter degree type (which is the same for any f. r. HSOP)!
- We prove the presence of a small–degree filter–regular HSOP:

Lemma [D. J. Green, S. K. 2008/9]

In R_n/I_n, kill $g_1, ..., g_z$, possibly some $\Delta_1, ..., \Delta_{i_0}$, and all monomials of some degree d.

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.
- We use the Dickson classes *only* for computing the filter degree type (which is the same for any f. r. HSOP)!
- We prove the presence of a small–degree filter–regular HSOP:

Lemma [D. J. Green, S. K. 2008/9]

In R_n/I_n, kill $g_1, ..., g_z$, possibly some $\Delta_1, ..., \Delta_{i_0}$, and all monomials of some degree d. If the quotient is finite, then there exist parameters in degree d that extend $g_1, ..., g_z, \Delta_1, ..., \Delta_{i_0}$ to a filter-regular HSOP.

Improvement by existence proof

Problem: Dickson invariants may be of large degree

- We take minimal factors of the Δ_i, for decreasing the degrees.
- We use the Dickson classes *only* for computing the filter degree type (which is the same for any f. r. HSOP)!
- We prove the presence of a small–degree filter–regular HSOP:

Lemma [D. J. Green, S. K. 2008/9]

In R_n/I_n, kill $g_1, ..., g_z$, possibly some $\Delta_1, ..., \Delta_{i_0}$, and all monomials of some degree d. If the quotient is finite, then there exist parameters in degree d that extend $g_1, ..., g_z, \Delta_1, ..., \Delta_{i_0}$ to a filter-regular HSOP.

Conway(3) and others would be unfeasible without that trick!

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative*

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

Do not re-invent the wheel — build the car!

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

Do not re-invent the wheel — build the car!

- Includes AXIOM, **GAP**, GP/PARI, MACAULAY2, MAXIMA, OCTAVE, R, **Singular**, interfaces to commercial software, ...

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

Do not re-invent the wheel — build the car!

- Includes AXIOM, **GAP**, GP/PARI, MACAULAY2, MAXIMA, OCTAVE, R, **Singular**, interfaces to commercial software, ...
- Several hundred thousand lines of new code

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

Do not re-invent the wheel — build the car!

- Includes AXIOM, **GAP**, GP/PARI, MACAULAY2, MAXIMA, OCTAVE, R, **Singular**, interfaces to commercial software, ...
- Several hundred thousand lines of new code
- **Cython** (compiled Python) — fast code, inclusion of C-code.

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

Do not re-invent the wheel — build the car!

- Includes AXIOM, **GAP**, GP/PARI, MACAULAY2, MAXIMA, OCTAVE, R, **Singular**, interfaces to commercial software, ...
- Several hundred thousand lines of new code
- **Cython** (compiled Python) — fast code, inclusion of C-code.
- Notebook, 2d- and 3d-graphics

What is "SAGE"? www.sagemath.org

Initiated in 2005 by W. Stein (UCSD, now at Univ. of Washington)

Mission statement

Creating a *uniform* open source *viable alternative* to MAGMA, MAPLE, MATHEMATICA, and MATLAB

Do not re-invent the wheel — build the car!

- Includes AXIOM, **GAP**, GP/PARI, MACAULAY2, MAXIMA, OCTAVE, R, **Singular**, interfaces to commercial software, ...
- Several hundred thousand lines of new code
- **Cython** (compiled Python) — fast code, inclusion of C-code.
- Notebook, 2d- and 3d-graphics
- Very helpful **community**

How we use SAGE's features

- GAP functions of David Green create subgroup data.

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)
  - Cochains, Cup Product, Induced Homomorphisms, ...

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)
  - Cochains, Cup Product, Induced Homomorphisms, ...
  - Benson's test, with our improvements.

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)
  - Cochains, Cup Product, Induced Homomorphisms, ...
  - Benson's test, with our improvements.
- Do not re-invent the wheel — Use SINGULAR!

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)
  - Cochains, Cup Product, Induced Homomorphisms, ...
  - Benson's test, with our improvements.
- Do not re-invent the wheel — Use SINGULAR!

GB of relation ideal for group number 836 of order 128

>1 month with self-made implementation (David Green), but only few hours with SINGULAR

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)
  - Cochains, Cup Product, Induced Homomorphisms, ...
  - Benson's test, with our improvements.
- Do not re-invent the wheel — Use SINGULAR!

GB of relation ideal for group number 836 of order 128

>1 month with self-made implementation (David Green), but only few hours with SINGULAR

- Ask people when problems occur!

How we use SAGE's features

- GAP functions of David Green create subgroup data.
- Cython Extension Modules:
  - Compute resolutions (based on David Green's C-code)
  - Cochains, Cup Product, Induced Homomorphisms, ...
  - Benson's test, with our improvements.
- Do not re-invent the wheel — Use SINGULAR!

GB of relation ideal for group number 836 of order 128

>1 month with self-made implementation (David Green), but only few hours with SINGULAR

- Ask people when problems occur!

  $\rightsquigarrow$ Huge speed-up by better use of interface and Cython

Typical generating sets

Minimal number of generators

Typical generating sets

Minimal number of generators

#Gen	#gps
34	3
36	1
39	1
65	1

2322 groups have less than 34 minimal generators

Typical generating sets

Minimal number of generators

#Gen	#gps
34	3
36	1
39	1
65	1

2322 groups have less than 34 minimal generators

The winner is:
Group 836

Typical generating sets

Minimal number of generators

#Gen	#gps
34	3
36	1
39	1
65	1

2322 groups have less than 34 minimal generators

The winner is:
Group 836

Maximal degree of minimal generators

Typical generating sets

Minimal number of generators

#Gen	#gps
34	3
36	1
39	1
65	1

2322 groups have less than 34 minimal generators

The winner is:
Group 836

Maximal degree of minimal generators

Deg	#gps
13	1
14	3
16	22
17	1

2301 groups have generator degree less than 12

Typical generating sets

Minimal number of generators

#Gen	#gps
34	3
36	1
39	1
65	1

2322 groups have less than 34 minimal generators

The winner is:
Group 836

Maximal degree of minimal generators

Deg	#gps
13	1
14	3
16	22
17	1

2301 groups have generator degree less than 12

The winner is:
Group 562

Typical relation ideals

Minimal number of relations

Typical relation ideals

Minimal number of relations

#Rel	#gps
461	1
486	1
526	1
626	1
1859	1

2323 groups have less than 440 minimal relations

Typical relation ideals

Minimal number of relations

#Rel	#gps
461	1
486	1
526	1
626	1
1859	1

2323 groups have less than 440 minimal relations

The winner is:
Group 836

Typical relation ideals

Minimal number of relations

#Rel	#gps
461	1
486	1
526	1
626	1
1859	1

2323 groups have less than 440 minimal relations

The winner is:
Group 836

Maximal degree of minimal relations

Deg	#gps
28	4
30	3
32	1
34	1

2319 groups have relation degree less than 27

Typical relation ideals

Minimal number of relations

#Rel	#gps
461	1
486	1
526	1
626	1
1859	1

2323 groups have less than 440 minimal relations

The winner is:
Group 836

Maximal degree of minimal relations

Deg	#gps
28	4
30	3
32	1
34	1

2319 groups have relation degree less than 27

The winner is:
Group 562

Typical relation ideals

Minimal number of relations

#Rel	#gps
461	1
486	1
526	1
626	1
1859	1

2323 groups have less than 440 minimal relations

The winner is:
Group 836

Maximal degree of minimal relations

Deg	#gps
28	4
30	3
32	1
34	1

2319 groups have relation degree less than 27

The winner is:
Group 562
But 2298, 2300, 2327 are hardest!

How good is Benson's criterion?

Failure of Improved Benson's Completeness Criterion

$d_B - d_\infty$	#gps
0	1779
1	341
2	168
3	39
4	1

d_B: Benson's Criterion applies
d_∞: computation stabilises.

How good is Benson's criterion?

Failure of Improved Benson's Completeness Criterion

$d_B - d_\infty$	#gps
0	1779
1	341
2	168
3	39
4	1

d_B: Benson's Criterion applies
d_∞: computation stabilises.

The loser is: Group 2320,
which is the direct product of D_8
with an elementary abelian group.

How good is Benson's criterion?

Failure of Improved Benson's Completeness Criterion

$d_B - d_\infty$	#gps
0	1779
1	341
2	168
3	39
4	1

d_B: Benson's Criterion applies
d_∞: computation stabilises.

The loser is: Group 2320,
which is the direct product of D_8
with an elementary abelian group.

Failure of Duflot's Depth Bound (2313 non-abelian groups)

depth - $\mathrm{rk}(Z(G))$	#gps
0	1767
1	508
2	37
3	1

$\mathrm{rk}(Z(G))$ is a lower bound for the depth of $H^*(G)$.

How good is Benson's criterion?

Failure of Improved Benson's Completeness Criterion

$d_B - d_\infty$	#gps
0	1779
1	341
2	168
3	39
4	1

d_B: Benson's Criterion applies
d_∞: computation stabilises.

The loser is: Group 2320,
which is the direct product of D_8
with an elementary abelian group.

Failure of Duflot's Depth Bound (2313 non-abelian groups)

depth - $\mathrm{rk}(Z(G))$	#gps
0	1767
1	508
2	37
3	1

$\mathrm{rk}(Z(G))$ is a lower bound for the depth of $H^*(G)$.

The winner is: Group 2326,
extraspecial of type +

To Do

To Do

- Search counterexamples for Strong Benson Conjecture.

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...
- Add more features:

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...
- Add more features: Interesting invariants,

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...
- Add more features: Interesting invariants, Steenrod actions,

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...
- Add more features: Interesting invariants, Steenrod actions, $\mathbb{F}_p$ cohomology of *general* finite groups

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...
- Add more features: Interesting invariants, Steenrod actions, $\mathbb{F}_p$ cohomology of *general* finite groups
- Create a *relational* data base.

To Do

- Search counterexamples for Strong Benson Conjecture.
  Currently: Group 299 of order 256, which is of defect 4.
  It shows unexpected behaviour: 77 generators up to degree 11, $\sim$ 3000 relations up to degree 22, but still incomplete — and blocks 158 Gb hard disk...
- Add more features: Interesting invariants, Steenrod actions, $\mathbb{F}_p$ cohomology of *general* finite groups
- Create a *relational* data base.

Thank you!