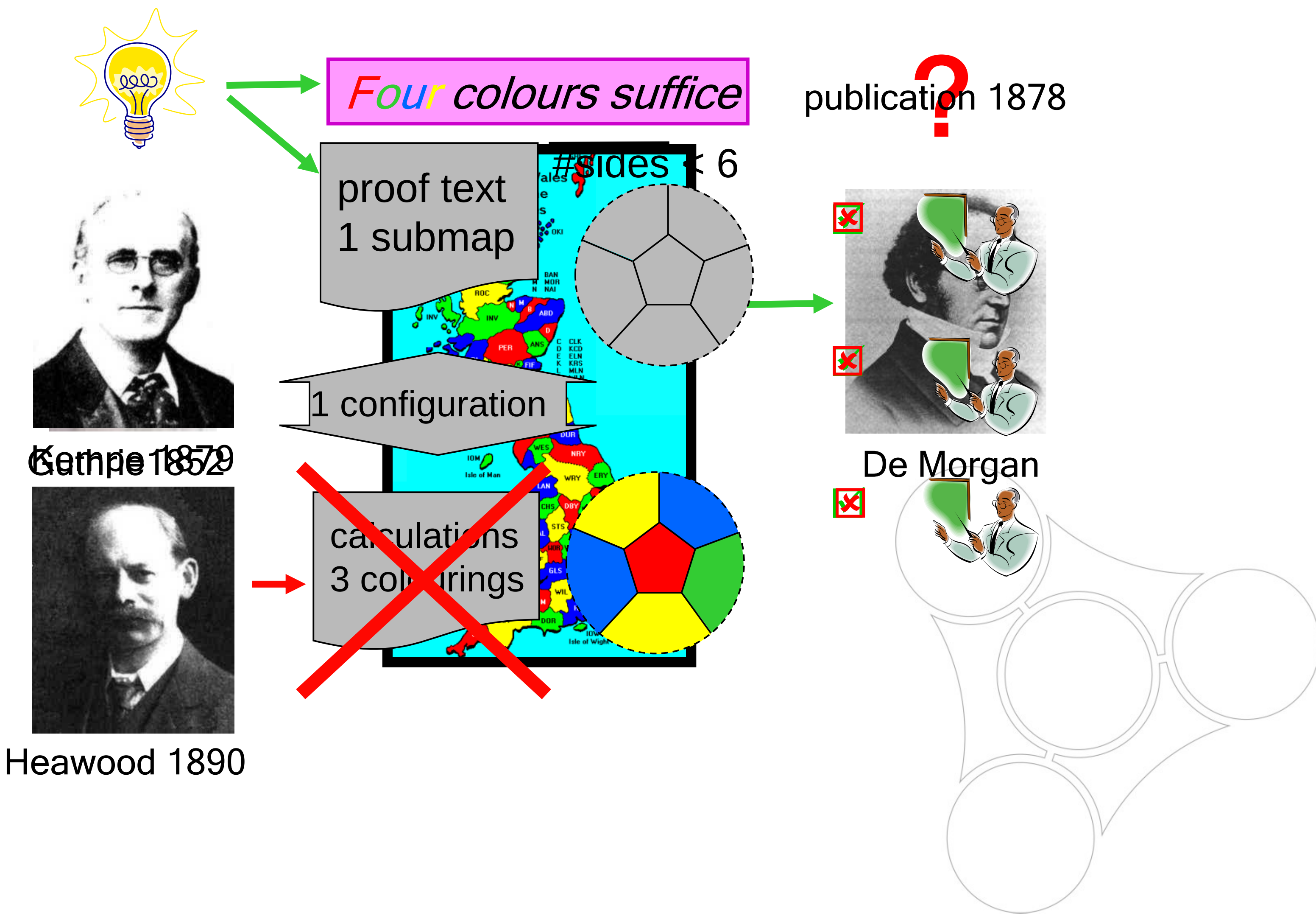
Proof engineering, from the Four Color to the Odd Order Theorem

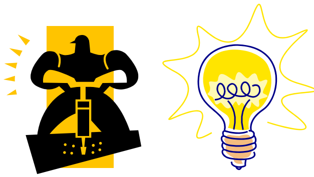
Georges Gonthier

Microsoft Research Cambridge

An old puzzle's story



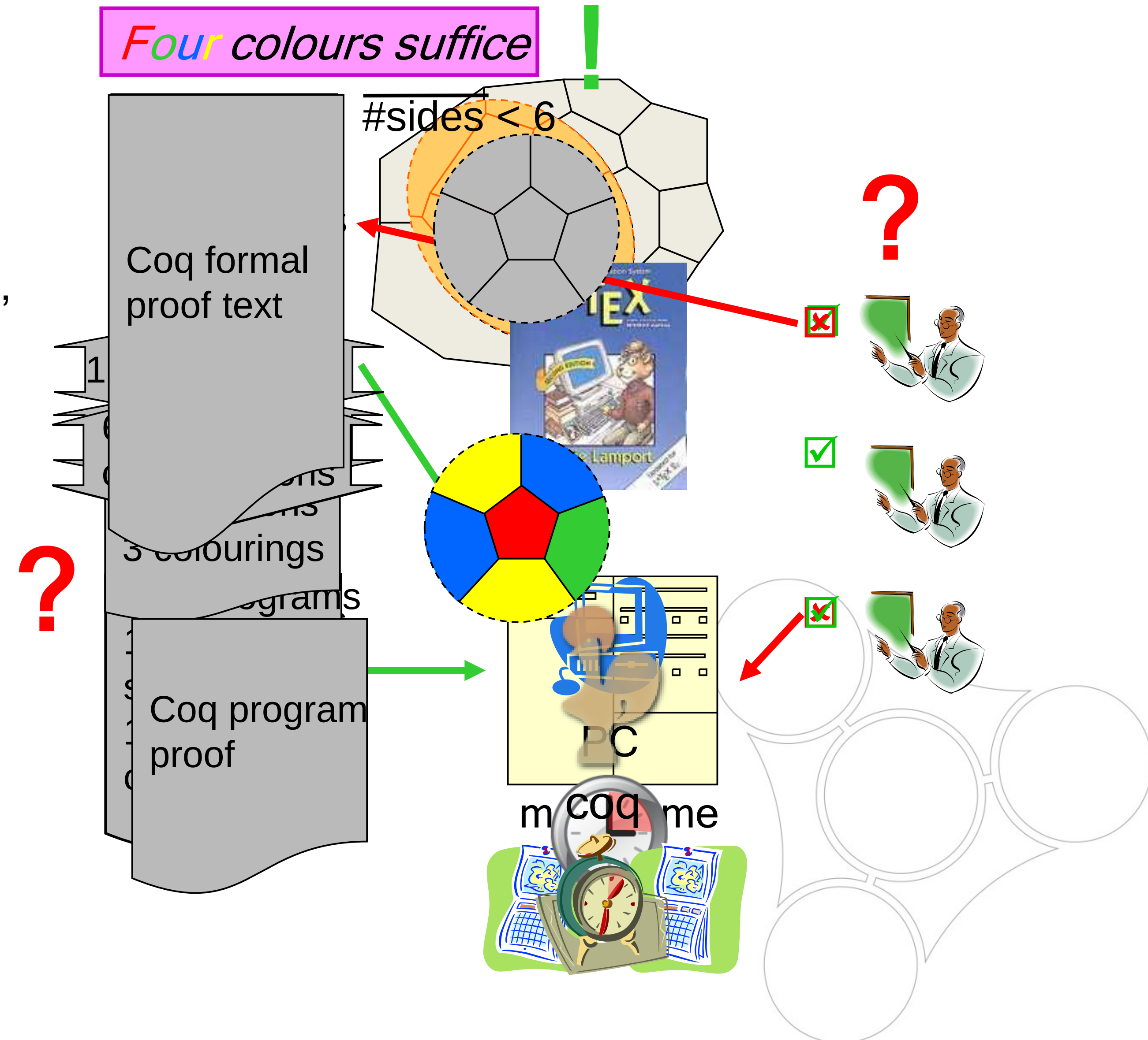
Saved by the computer?



Robertson & Sanders, Seymour & Thomas 1995

Gonthier & Werner, 2004

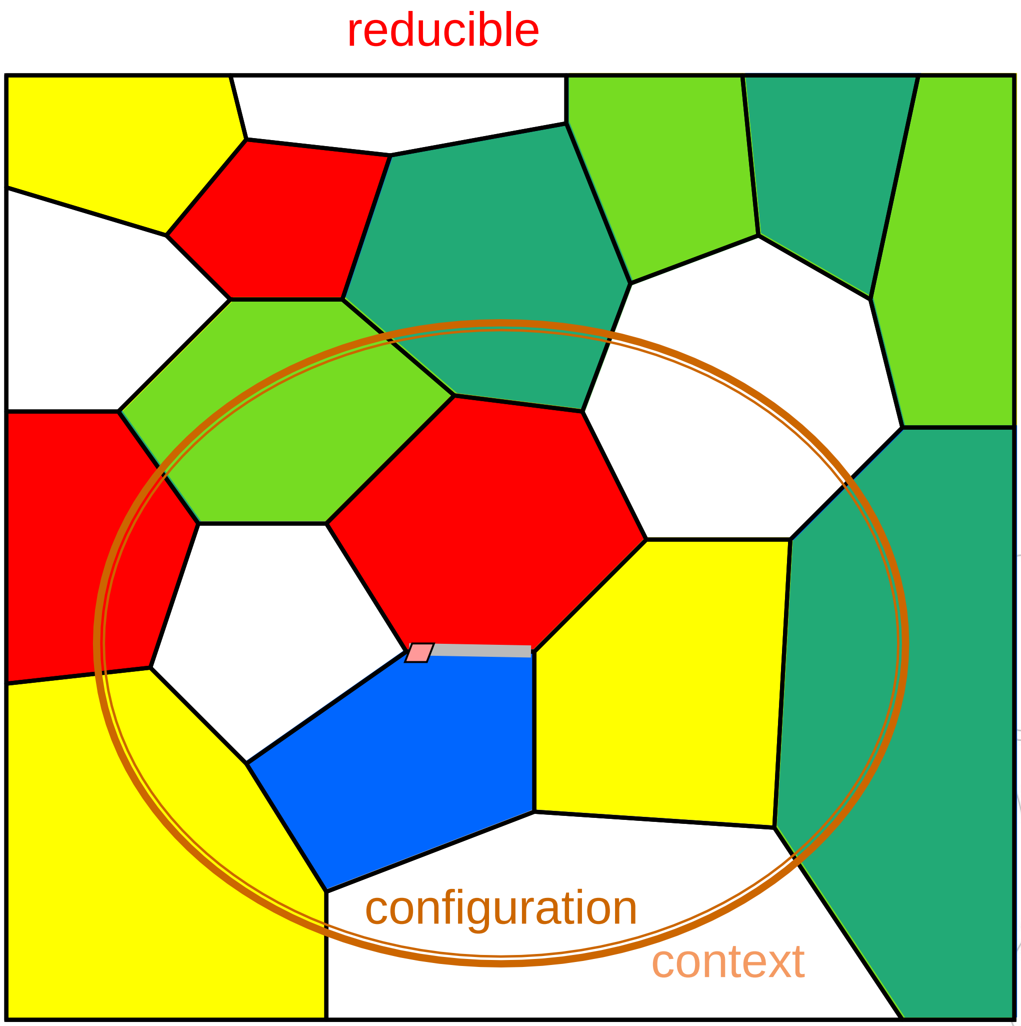
Appel & Haken 1976



Early lessons

- It is possible to build rigorously self-certifying program/proofs.
  - *proof by computation is feasible.*
- A computer proof assistant can be used to explore the logical structure of a proof.
  - *new math can be gleaned from a formalization*.
- Software Engineering matters in formal proofs.
  - old rules and new techniques.

Coloring by induction



The whole proof

- Find a set of configurations such that:

  (A) *unavoidability*: At least one appears in any planar map.

  10,000 cases

  (B) *reducibility*: Each one can be coloured to match any planar ring colouring.

  1,000,000,000 cases

- **Verify that the combinatorics fit the topology (graph theory + analysis).**

The Poincaré principle

- How do you prove: $2 + 2 = 4$?
- Given $2 \overset{\text{def}}{=} 1+ (1+ 0)$

$$4 \overset{\text{def}}{=} 1+ (1+ (1+ (1+ 0)))$$

$$n + m \overset{\text{def}}{=} \text{if } n \text{ is } 1+ n' \text{ then } 1+ (n' + m) \text{ else } m$$

(a recursive program)

a: $0 + 2 = 2$ *(neutral left)*

b: $(1 + 0) + 2 = 1 + (0 + 2)$ *(associativity)*

c: $2 + 2 = 1 + ((1 + 0) + 2)$ *(def, associativity)*

d: $2 + 2 = 1 + (1 + (0 + 2))$ *(replace b in c)*

e: *(def, calculation)* *(replace a in d, def)*

Reflecting reducibility

- Setup

  Variable cf : config.

  Definition cfreducible : Prop := …

  Definition check_reducible : bool := …

  Lemma check_reducible_valid : check_reducible -> cfreducible.

- Usage

  Lemma cfred232 : cfreducible (Config 11 33 37 H 2 H 13 Y 5 H 10 H 1 H 1 Y 3 H 11 Y 4 H 9 H 1 Y 3 H 9 Y 6 Y 1 Y 1 Y 3 Y 1 Y Y 1 Y).

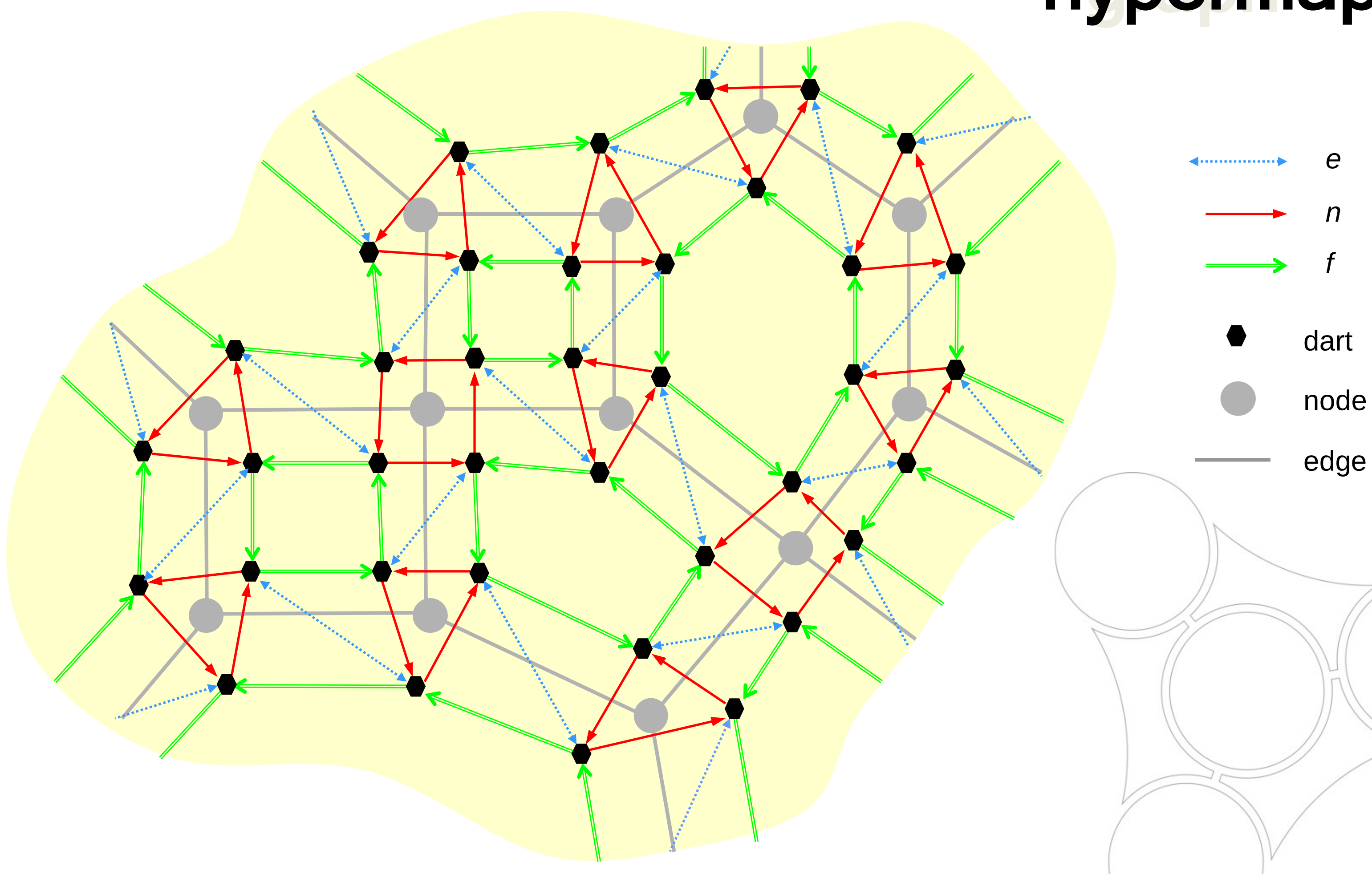
  Proof. apply check_reducible_valid; by compute. Qed.

20,000,000 cases

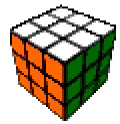
Describing a map

Euler: #edge + #node + #face = #dart + 2 * #comp

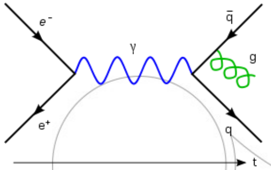
hypermap



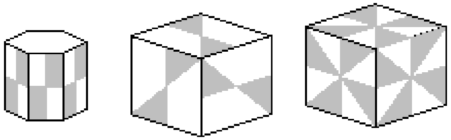
Group Theory

- The theory of invertible operators...
  - and of puzzles 

- Due to Évariste Galois
  - $x^5 + 3x^3 + 7 = 0$
- Explains quantum mechanics



- Cristallography, cryptography…

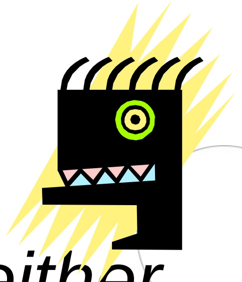


The Swiss army knife of Group Theory

- Theorem (Jordan-Hölder):

  Any finite group factors uniquely

  into a series of simple *groups*



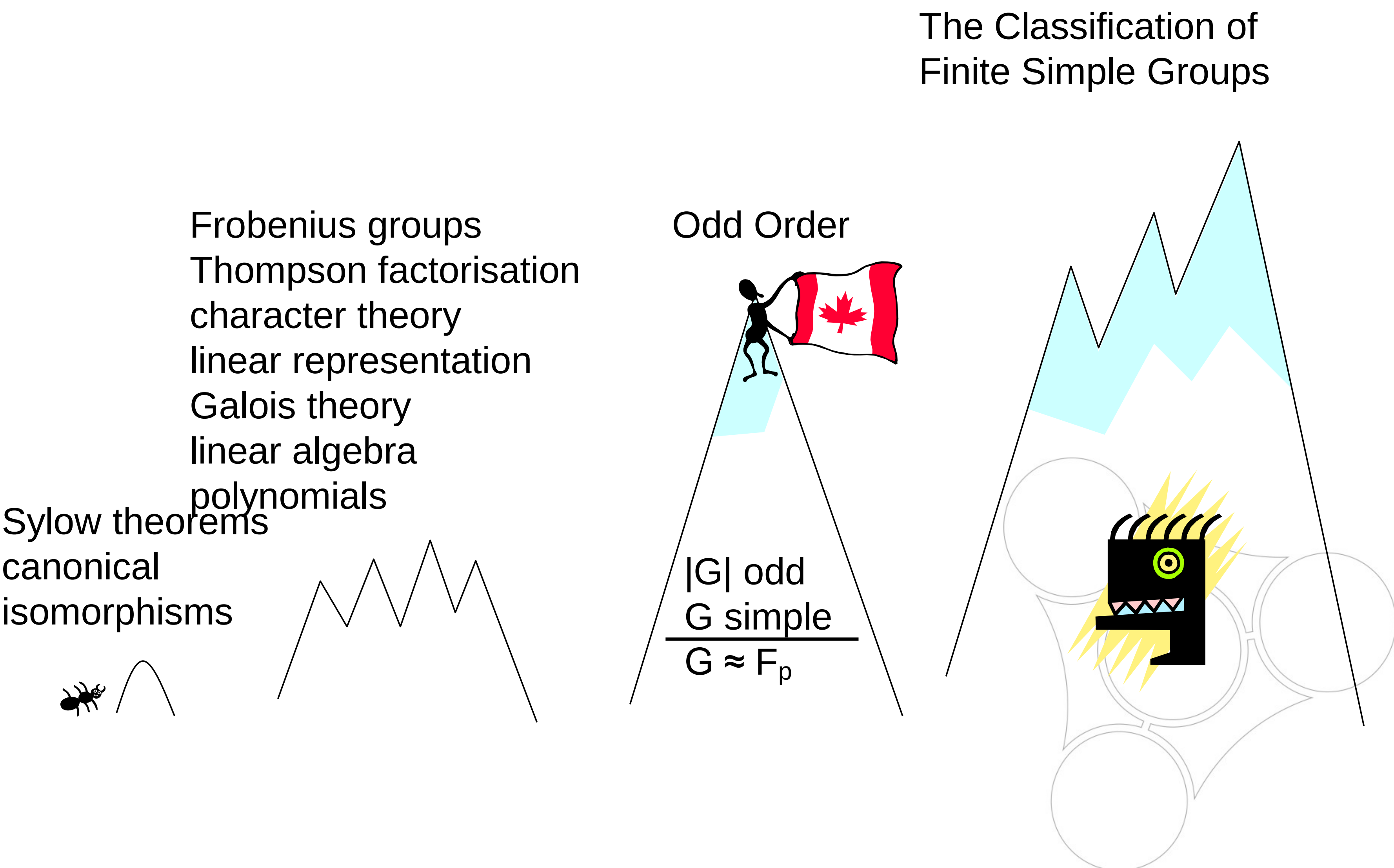
- Theorem (Classification):

  Finite simple groups belong to either

  one of 4 general classes,

  or one of 26 sporadic exceptions

The Finite Group Challenge



The Odd Order Theorem

Theorem (Feit & Thompson, 1963):

All finite groups of odd order are solvable.

Proof. – *255 pages, 50 years*

Proofread. – 240 pages, 20 years

```
Theorem Feit_Thompson (gT : finGroupType) (G : {group gT}) :
  odd #|G| -> solvable G.
```

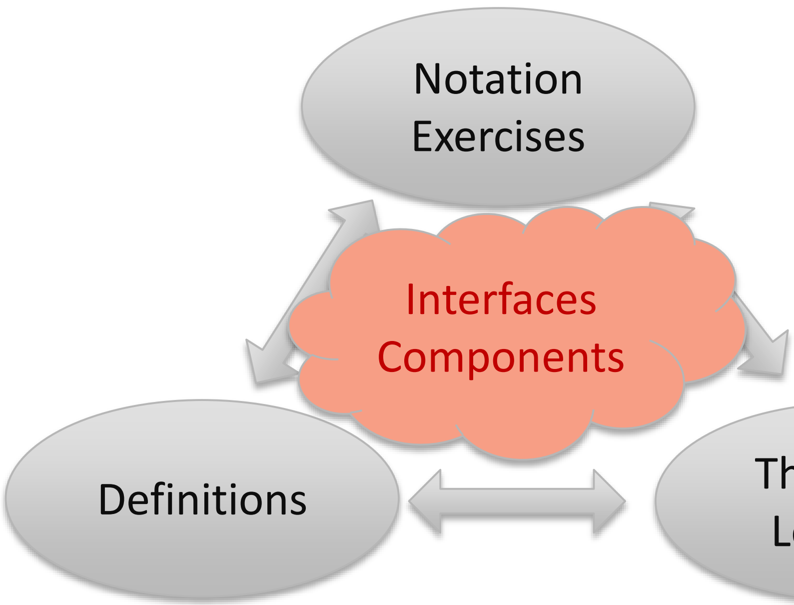
Definitions. – *54 LOC*

Proof. – *45,000 LOC, 2 years (+ 4 for the library)*

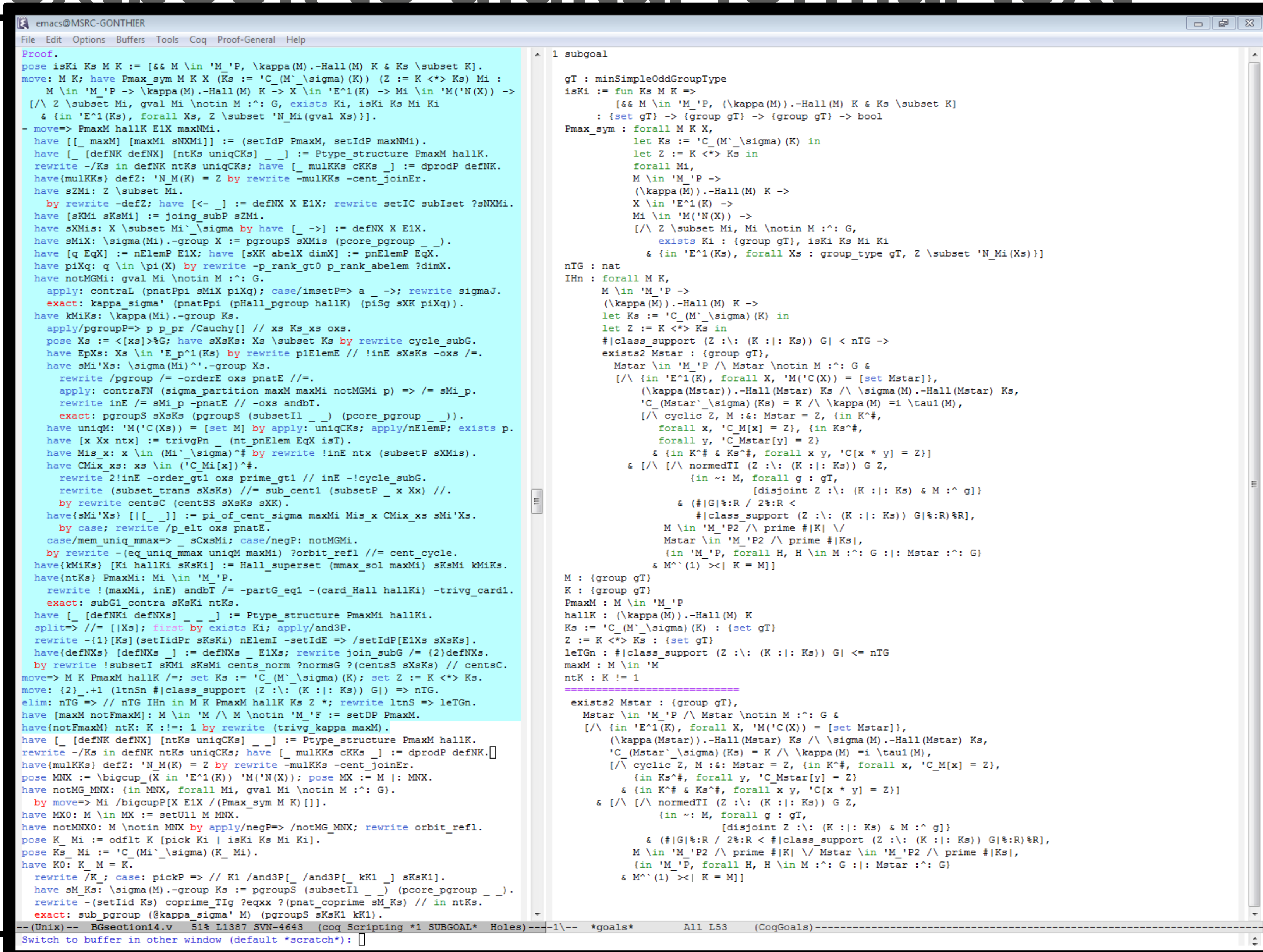
A mathematical library shelf

```
Section Lagrange.

Variable gT : finGroupType.
Implicit Types G H K : {group gT}.

Lemma LagrangeI G H : (#|G :&: H| * #|G : H|)%N = #|G|.
Proof.
rewrite -[#|G|]sum1_card (partition_big_imset (rcoset H)) /=.
rewrite mulnC -sum_nat_const; apply: eq_bigr => _ /rcosetsP[x Gx ->].
rewrite -(card_rcoset _ x) -sum1_card; apply: eq_bigl => y.
rewrite rcosetE eqEcard mulGS !card_rcoset leqnn andbT.
by rewrite group_modr sub1set // inE.
Qed.

Lemma divgI G H : #|G| %/ #|G :&: H| = #|G : H|.
Proof. by rewrite -(LagrangeI G H) mulKn ?cardG_gt0. Qed.

Lemma divg_index G H : #|G| %/ #|G : H| = #|G :&: H|.
Proof. by rewrite -(LagrangeI G H) mulnK. Qed.

Lemma dvdn_indexg G H : #|G : H| %| #|G|.
Proof. by rewrite -(LagrangeI G H) dvdn_mull. Qed.

Theorem Lagrange G H : H \subset G -> (#|H| * #|G : H|)%N = #|G|.
Proof. by move/setIidPr=> sHG; rewrite -{1}sHG LagrangeI. Qed.
```

Mathematics



Textbook to digital formal text



emacs@MSRC-GONTHIER

File Edit Options Buffers Tools Coq Proof-General Help

```
Proof.
pose isKi Ks M K := [&& M \in 'M_'P, \kappa(M).-Hall(M) K & Ks \subset K].
move: M K; have Pmax_sym M K X (Ks := 'C_(M`_\sigma)(K)) (Z := K <*> Ks) Mi :
    M \in 'M_'P -> \kappa(M).-Hall(M) K -> X \in 'E^1(K) -> Mi \in 'M('N(X)) ->
  [/\ Z \subset Mi, gval Mi \notin M :^: G, exists Ki, isKi Ks Mi Ki
   & {in 'E^1(Ks), forall Xs, Z \subset 'N_Mi(gval Xs)}].
- move=> PmaxM hallK E1X maxNMi.
  have [[_ maxM] [maxMi sNXMi]] := (setIdP PmaxM, setIdP maxNMi).
  have [_ [defNK defNX] [ntKs uniqCKs] _ _] := Ptype_structure PmaxM hallK.
  rewrite -/Ks in defNK ntKs uniqCKs; have [_ mulKKs cKKs _] := dprodP defNK.
  have{mulKKs} defZ: 'N_M(K) = Z by rewrite -mulKKs -cent_joinEr.
  have sZMi: Z \subset Mi.
    by rewrite -defZ; have [<- _] := defNX X E1X; rewrite setIC subIset ?sNXMi.
  have [sKMi sKsMi] := joing_subP sZMi.
  have sXMis: X \subset Mi`_\sigma by have [_ ->] := defNX X E1X.
  have sMiX: \sigma(Mi).-group X := pgroupS sXMis (pcore_pgroup _ _).
  have [q EqX] := nElemP E1X; have [sXK abelX dimX] := pnElemP EqX.
  have piXq: q \in \pi(X) by rewrite -p_rank_gt0 p_rank_abelem ?dimX.
  have notMGMi: gval Mi \notin M :^: G.
    apply: contraL (pnatPpi sMiX piXq); case/imsetP=> a _ ->; rewrite sigmaJ.
    exact: kappa_sigma' (pnatPpi (pHall_pgroup hallK) (piSg sXK piXq)).
  have kMiKs: \kappa(Mi).-group Ks.
    apply/pgroupP=> p p_pr /Cauchy[] // xs Ks_xs oxs.
    pose Xs := <[xs]>%G; have sXsKs: Xs \subset Ks by rewrite cycle_subG.
    have EpXs: Xs \in 'E_p^1(Ks) by rewrite p1ElemE // !inE sXsKs -oxs /=.
    have sMi'Xs: \sigma(Mi)^'.-group Xs.
      rewrite /pgroup /= -orderE oxs pnatE //=.
      apply: contraFN (sigma_partition maxM maxMi notMGMi p) => /= sMi_p.
      rewrite inE /= sMi_p -pnatE // -oxs andbT.
      exact: pgroupS sXsKs (pgroupS (subsetIl _ _) (pcore_pgroup _ _)).
    have uniqM: 'M('C(Xs)) = [set M] by apply: uniqCKs; apply/nElemP; exists p.
    have [x Xx ntx] := trivgPn _ (nt_pnElem EqX isT).
    have Mis_x: x \in (Mi`_\sigma)^# by rewrite !inE ntx (subsetP sXMis).
    have CMix_xs: xs \in ('C_Mi[x])^#.
      rewrite 2!inE -order_gt1 oxs prime_gt1 // inE -!cycle_subG.
      rewrite (subset_trans sXsKs) //= sub_cent1 (subsetP _ x Xx) //.
      by rewrite centsC (centSS sXsKs sXK).
    have{sMi'Xs} [|[_ _]] := pi_of_cent_sigma maxMi Mis_x CMix_xs sMi'Xs.
      by case; rewrite /p_elt oxs pnatE.
    case/mem_uniq_mmax=> _ sCxsMi; case/negP: notMGMi.
    by rewrite -(eq_uniq_mmax uniqM maxMi) ?orbit_refl //= cent_cycle.
  have{kMiKs} [Ki hallKi sKsKi] := Hall_superset (mmax_sol maxMi) sKsMi kMiKs.
  have{ntKs} PmaxMi: Mi \in 'M_'P.
    rewrite !(maxMi, inE) andbT /= -partG_eq1 -(card_Hall hallKi) -trivg_card1.
    exact: subG1_contra sKsKi ntKs.
  have [_ [defNKi defNXs] _ _ _] := Ptype_structure PmaxMi hallKi.
  split=> //= [|Xs]; first by exists Ki; apply/and3P.
  rewrite -{1}[Ks](setIidPr sKsKi) nElemI -setIdE => /setIdP[E1Xs sXsKs].
  have{defNXs} [defNXs _] := defNXs _ E1Xs; rewrite join_subG /= {2}defNXs.
  by rewrite !subsetI sKMi sKsMi cents_norm ?normsG ?(centsS sXsKs) // centsC.
move=> M K PmaxM hallK /=; set Ks := 'C_(M`_\sigma)(K); set Z := K <*> Ks.
move: {2}_.+1 (ltnSn #|class_support (Z :\: (K :|: Ks)) G|) => nTG.
elim: nTG => // nTG IHn in M K PmaxM hallK Ks Z *; rewrite ltnS => leTGn.
have [maxM notFmaxM]: M \in 'M /\ M \notin 'M_'F := setDP PmaxM.
have{notFmaxM} ntK: K :!=: 1 by rewrite (trivg_kappa maxM).
have [_ [defNK defNX] [ntKs uniqCKs] _ _] := Ptype_structure PmaxM hallK.
rewrite -/Ks in defNK ntKs uniqCKs; have [_ mulKKs cKKs _] := dprodP defNK.
have{mulKKs} defZ: 'N_M(K) = Z by rewrite -mulKKs -cent_joinEr.
pose MNX := \bigcup_(X in 'E^1(K)) 'M('N(X)); pose MX := M |: MNX.
have notMG_MNX: {in MNX, forall Mi, gval Mi \notin M :^: G}.
  by move=> Mi /bigcupP[X E1X /(Pmax_sym M K)[]].
have MX0: M \in MX := setU11 M MNX.
have notMNX0: M \notin MNX by apply/negP=> /notMG_MNX; rewrite orbit_refl.
pose K_ Mi := odflt K [pick Ki | isKi Ks Mi Ki].
pose Ks_ Mi := 'C_(Mi`_\sigma)(K_ Mi).
have K0: K_ M = K.
  rewrite /K_; case: pickP => // K1 /and3P[_ /and3P[_ kK1 _] sKsK1].
  have sM_Ks: \sigma(M).-group Ks := pgroupS (subsetIl _ _) (pcore_pgroup _ _).
  rewrite -(setIid Ks) coprime_TIg ?eqxx ?(pnat_coprime sM_Ks) // in ntKs.
  exact: sub_pgroup (@kappa_sigma' M) (pgroupS sKsK1 kK1).
```

```
1 subgoal

  gT : minSimpleOddGroupType
  isKi := fun Ks M K =>
          [&& M \in 'M_'P, (\kappa(M)).-Hall(M) K & Ks \subset K]
      : {set gT} -> {group gT} -> {group gT} -> bool
  Pmax_sym : forall M K X,
             let Ks := 'C_(M`_\sigma)(K) in
             let Z := K <*> Ks in
             forall Mi,
             M \in 'M_'P ->
             (\kappa(M)).-Hall(M) K ->
             X \in 'E^1(K) ->
             Mi \in 'M('N(X)) ->
             [/\ Z \subset Mi, Mi \notin M :^: G,
                exists Ki : {group gT}, isKi Ks Mi Ki
              & {in 'E^1(Ks), forall Xs : group_type gT, Z \subset 'N_Mi(Xs)}]
  nTG : nat
  IHn : forall M K,
        M \in 'M_'P ->
        (\kappa(M)).-Hall(M) K ->
        let Ks := 'C_(M`_\sigma)(K) in
        let Z := K <*> Ks in
        #|class_support (Z :\: (K :|: Ks)) G| < nTG ->
        exists2 Mstar : {group gT},
          Mstar \in 'M_'P /\ Mstar \notin M :^: G &
          [/\ {in 'E^1(K), forall X, 'M('C(X)) = [set Mstar]},
              (\kappa(Mstar)).-Hall(Mstar) Ks /\ \sigma(M).-Hall(Mstar) Ks,
              'C_(Mstar`_\sigma)(Ks) = K /\ \kappa(M) =i \tau1(M),
              [/\ cyclic Z, M :&: Mstar = Z, {in K^#,
                 forall x, 'C_M[x] = Z}, {in Ks^#,
                 forall y, 'C_Mstar[y] = Z}
                & {in K^# & Ks^#, forall x y, 'C[x * y] = Z}]
            & [/\ [/\ normedTI (Z :\: (K :|: Ks)) G Z,
                      {in ~: M, forall g : gT,
                                [disjoint Z :\: (K :|: Ks) & M :^ g]}
                    & (#|G|%:R / 2%:R <
                       #|class_support (Z :\: (K :|: Ks)) G|%:R)%R],
                  M \in 'M_'P2 /\ prime #|K| \/
                  Mstar \in 'M_'P2 /\ prime #|Ks|,
                  {in 'M_'P, forall H, H \in M :^: G :|: Mstar :^: G}
                & M^`(1) ><| K = M]]
  M : {group gT}
  K : {group gT}
  PmaxM : M \in 'M_'P
  hallK : (\kappa(M)).-Hall(M) K
  Ks := 'C_(M`_\sigma)(K) : {set gT}
  Z := K <*> Ks : {set gT}
  leTGn : #|class_support (Z :\: (K :|: Ks)) G| <= nTG
  maxM : M \in 'M
  ntK : K != 1
  ============================
   exists2 Mstar : {group gT},
     Mstar \in 'M_'P /\ Mstar \notin M :^: G &
    [/\ {in 'E^1(K), forall X, 'M('C(X)) = [set Mstar]},
        (\kappa(Mstar)).-Hall(Mstar) Ks /\ \sigma(M).-Hall(Mstar) Ks,
        'C_(Mstar`_\sigma)(Ks) = K /\ \kappa(M) =i \tau1(M),
        [/\ cyclic Z, M :&: Mstar = Z, {in K^#, forall x, 'C_M[x] = Z},
           {in Ks^#, forall y, 'C_Mstar[y] = Z}
          & {in K^# & Ks^#, forall x y, 'C[x * y] = Z}]
      & [/\ [/\ normedTI (Z :\: (K :|: Ks)) G Z,
                {in ~: M, forall g : gT,
                          [disjoint Z :\: (K :|: Ks) & M :^ g]}
              & (#|G|%:R / 2%:R < #|class_support (Z :\: (K :|: Ks)) G|%:R)%R],
            M \in 'M_'P2 /\ prime #|K| \/ Mstar \in 'M_'P2 /\ prime #|Ks|,
            {in 'M_'P, forall H, H \in M :^: G :|: Mstar :^: G}
          & M^`(1) ><| K = M]]
```

```
--(Unix)--  BGsection14.v   51% L1387 SVN-4643  (coq Scripting *1 SUBGOAL*  Holes)----1\--  *goals*       All L53     (CoqGoals)----
Switch to buffer in other window (default *scratch*):
```

Demonstration

```
mxtrace_mulC is defined
```

```
  trE : forall (m0 n0 : nat) (A0 : 'M_(m0, n0)) (B0 : 'M_(n0, m0)),
        \tr (A0 *m B0) = \sum_i \sum_j A0 i j * B0 j i
  ============================
\tr (A *m B) = \tr (B *m A)
subgoal 2 is:
 \tr (A *m B) = \tr (B *m A)
```

$$tr\ AB = \sum_i (AB)_{i,i} = \sum_i A_{i,i}$$

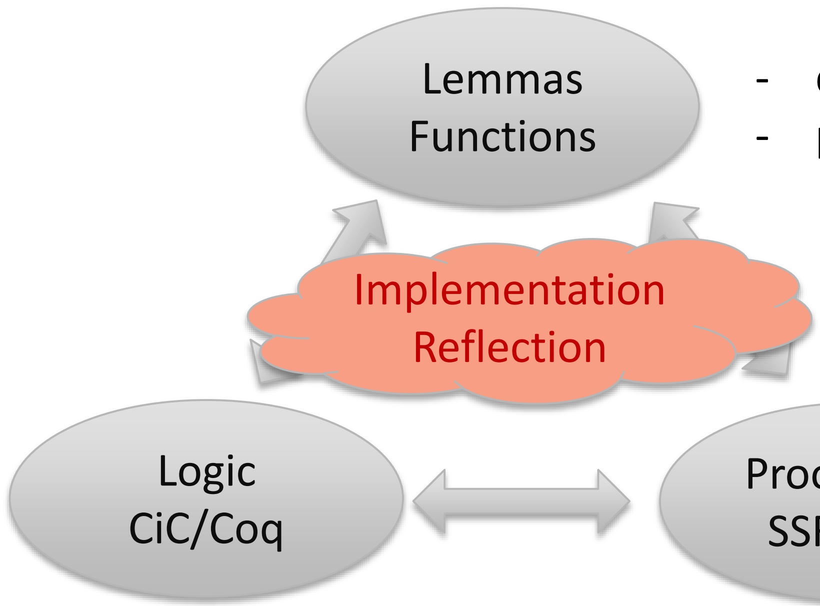
$$= \sum_i \sum_j A_{i,j} B_{j,i} \qquad (AB)_{i,j} = \sum_k A_{i,k} B_{k,j}$$

$$= \sum_j \sum_i A_{i,j} B_{j,i}$$

$$tr\ AB = \sum_j \sum_i B_{j,i} A_{i,j} = \sum_j (BA)_{j,j} = tr\ BA$$

```
Lemma mxtrace_mulC m n (A : 'M[R]_(m, n)) B :
   \tr (A *m B) = \tr (B *m A).
Proof.
gen have trE: m n A B / \tr (A *m B) = \sum_i \sum_j A i j * B j i.
  by apply: eq_bigr => i _; rewrite mxE.
rewrite {}!trE exchange_big.
by do 2!apply: eq_bigr => ? _; apply: mulrC.
Qed.
```

Formal mathematics



- define types
- package computation

- infer types
- compute types

- invoke computation
- control computation

Algebraic notation

$$\sum_{i<n} a_i\, x^i$$

$$\sum_{d\,|\,n} \Phi(n/d)\, m^d$$

$$\bigwedge_{i=1}^{n}{}_{GCD}\, Q_i(X)$$

$$\sum_{\sigma\in S_n} (-1)^\sigma \prod_i A_{i,i\sigma}$$

$$\bigcap_{H<G \atop H\ \mathrm{maximal}} H$$

$$\bigoplus_{V_i \approx W} V_i$$

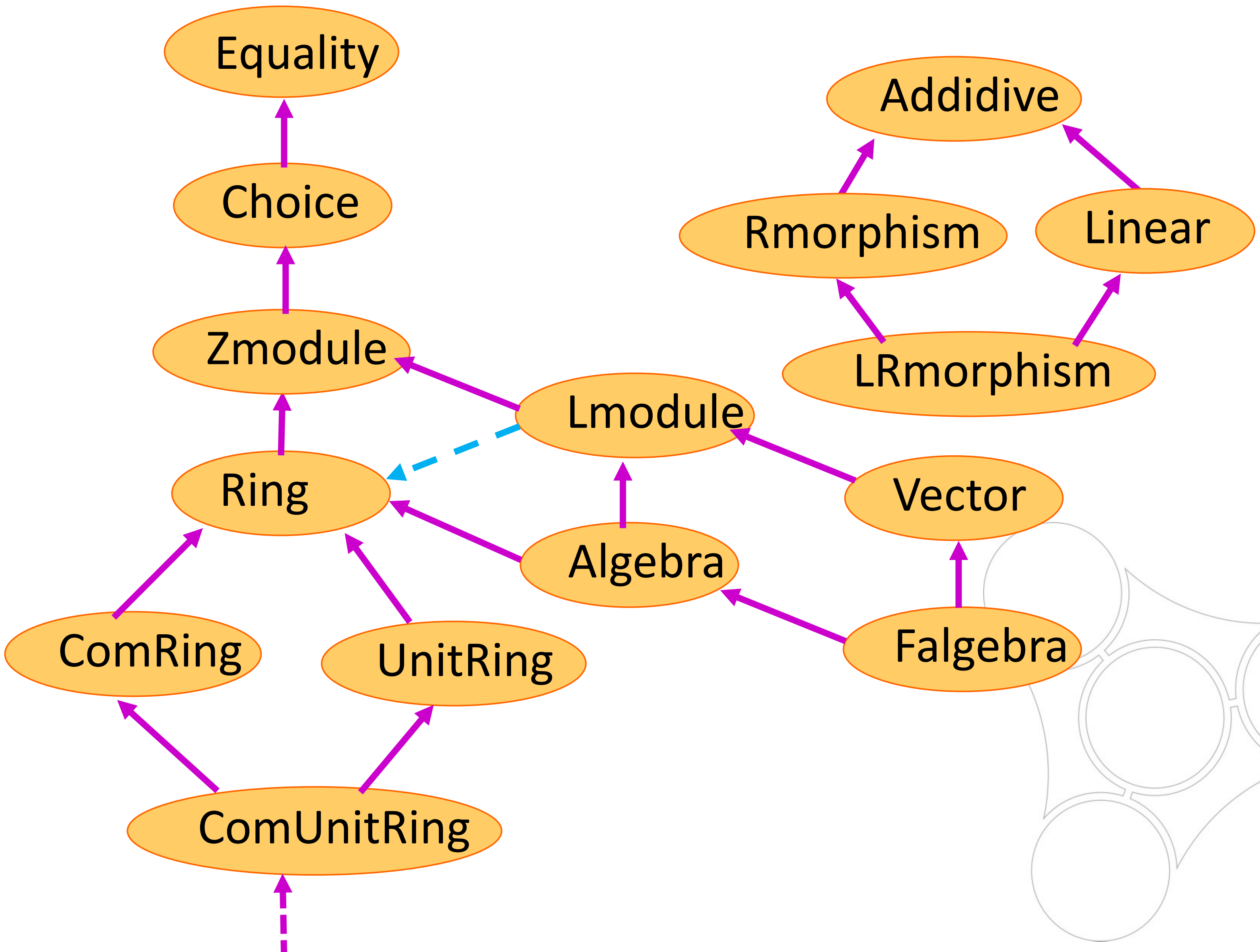
```
\bigcap_{H < G \atop H {\rm\ maximal}} H
```

```
Definition determinant n (A : 'M_n) : R :=
  \sum_(s : 'S_n) (-1) ^+ s * \prod_i A i (s i).
```

Implementing notation

```
Definition mxtrace (R : ringType) n (A : 'M[R]_n) :=
    @bigop R 'I_n 0 +%R (index_enum _)
        (fun i : 'I_n => fun_of_matrix A i i)
```

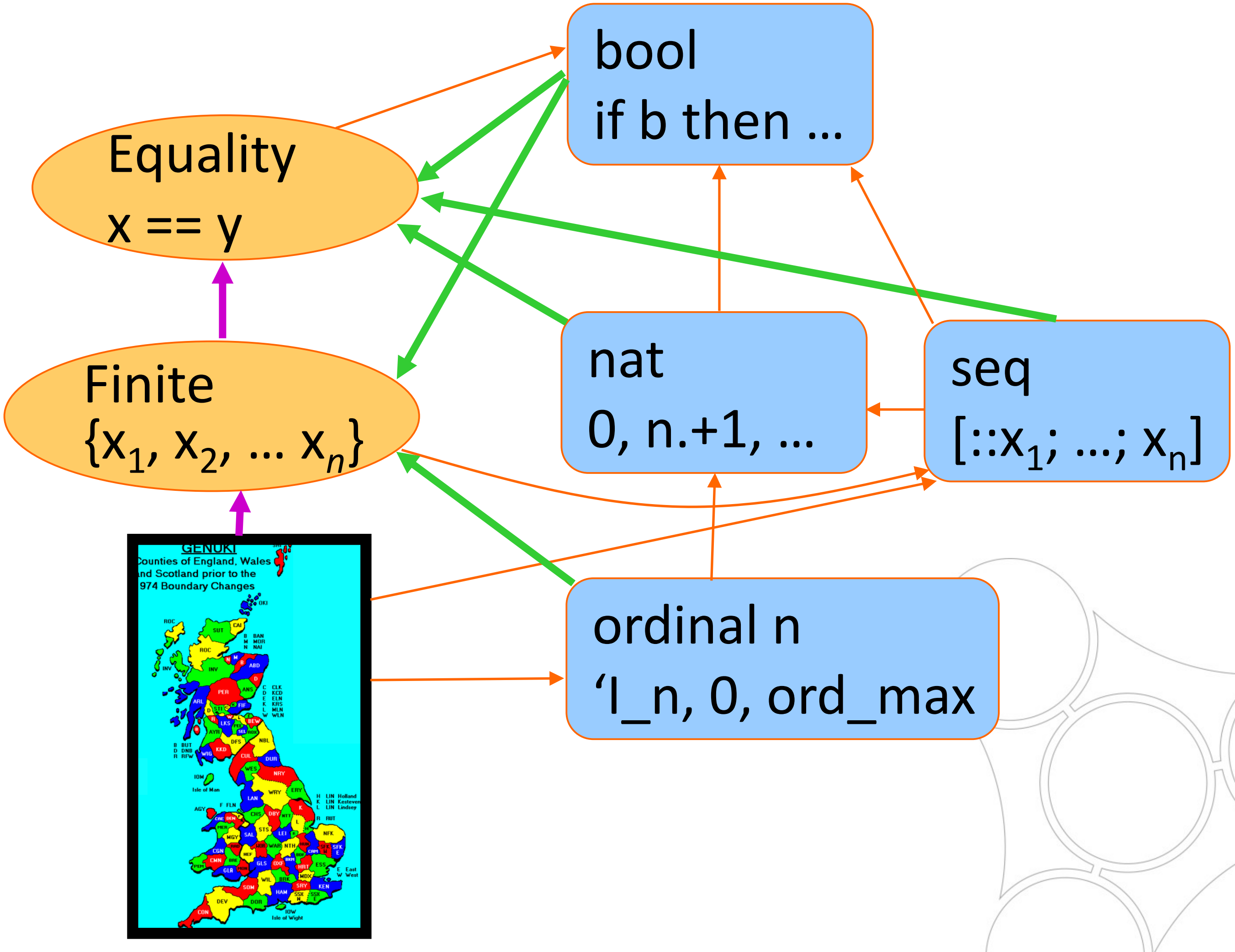
Algebra interfaces



Inferring notation

```
Definition mxtrace (R : ringType) n (A : 'M[R]_n) :=
   @bigop R 'I_n 0 (@Gring.add (Ring.ZmodType R))
        (index_enum _)
        (fun i : 'I_n => fun_of_matrix A i i)
```

Basic interfaces and objects



Ad hoc inference

```
Definition mxtrace (R : ringType) n (A : `M[R]_n) :=
   @bigop R `I_n 0 (@Gring.add (Ring.ZmodType R))
       (index_enum (ordinal_finType n))
       (fun i : `I_n => fun_of_matrix A i i)
```

Generic Lemmas

Pull, split, reindex, exchange …

```
Lemma bigD1 (I : finType) (j : I) P F :
  P j -> \big[*%M/1]_(i | P i) F i
    = F j * \big[*%M/1]_(i | P i && (i != j)) F i.

Lemma big_split I (r : list I) P F1 F2 :
  \big[*%M/1]_(i <- r | P i) (F1 i * F2 i) =
    \big[*%M/1]_(i <- r | P i) F1 i * \big[*%M/1]_(i <- r | P i) F2 i.

Lemma reindex (I J : finType) (h : J -> I) P F :
  {on P, bijective h} ->
  \big[*%M/1]_(i | P i) F i = \big[*%M/1]_(j | P (h j)) F (h j).

Lemma bigA_distr_bigA (I J : finType) F :
  \big[*%M/1]_(i : I) \big[+%M/0]_(j : J) F i j
    = \big[+%M/0]_(f : {ffun I -> J}) \big[*%M/1]_(i) F i (f i).
```

Operator structures

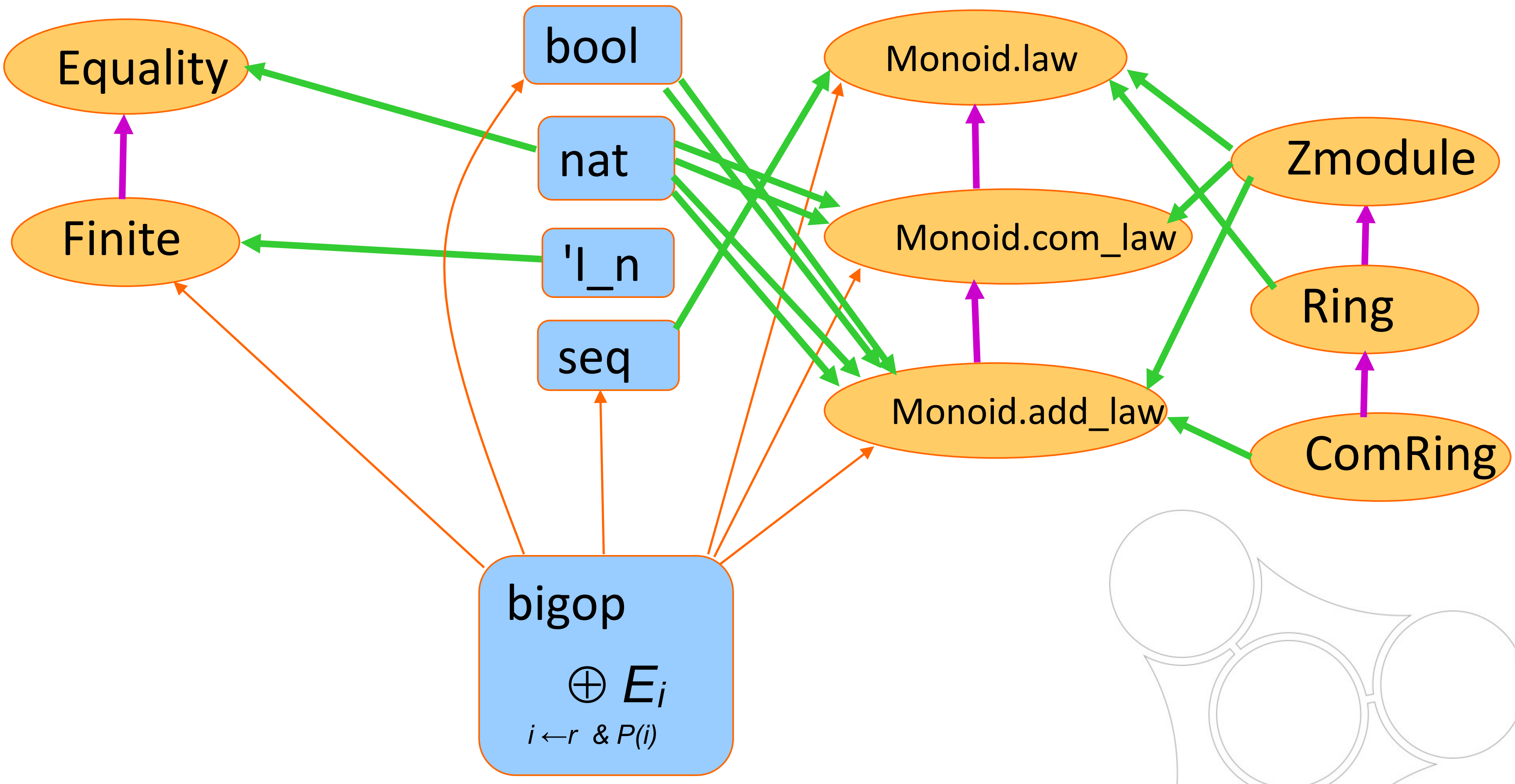
Polymorphism for values!

```
Structure law : Type :=
 Law {
  operator :> T -> T -> T;
  _ : associative operator;
  _ : left_id idx operator;
  _ : right_id idx operator
 }.
```

```
Structure com_law : Type :=
 AbelianLaw {
   com_operator :> law;
   _ : commutative com_operator
 }.
```

```
Canonical addn_monoid := Monoid.Law addnA add0n addn0.
Canonical addn_abeloid := Monoid.ComLaw addnC.
Canonical muln_monoid := Monoid.Law mulnA mul1n muln1.
…

Canonical ring_add_monoid := Monoid.Law addrA add0r addr0.
Canonical ring_add_abeloid := Monoid.ComLaw addrC.
…
```

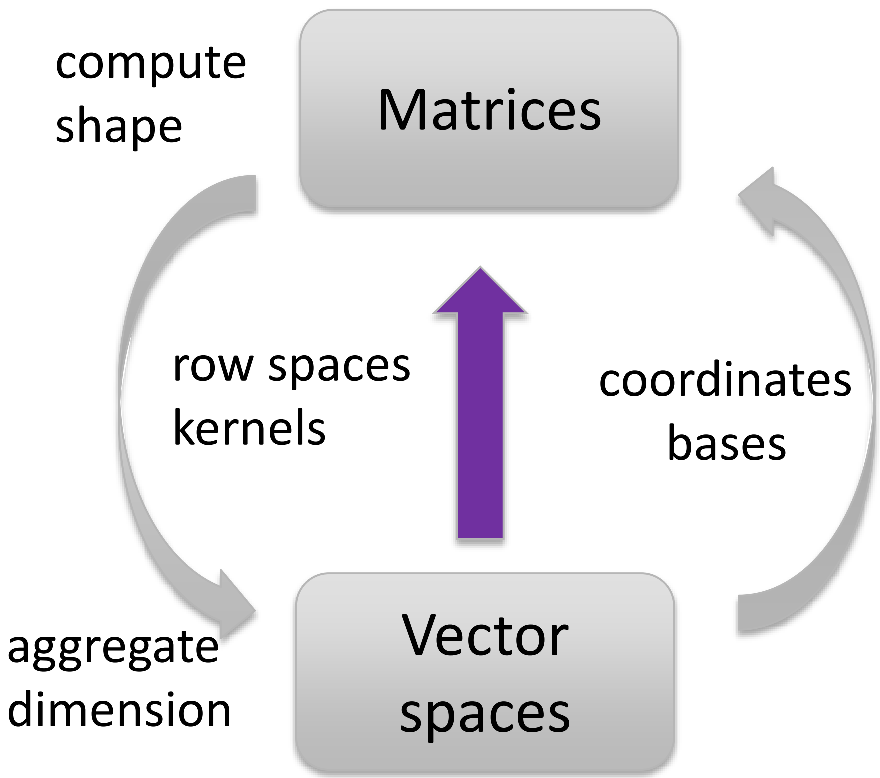
Interfacing big operators



More mathematical components…

- Finite group theory: morphisms, actions, characteristic & functor subgroups, p-groups, Frobenius & extremal groups…
- Character theory, representation and module theory, vector geometry.
- Finite field and Galois theory, algebraic number theory.
- Linear algebra, matrix rank.

Linear algebra interface?



group representation
$\Xi : G \to M_n(\mathbb{C})$

group character
$\chi = \mathrm{tr}\,\Xi : G \to \mathbb{C}$

Notation abuse

In math:

$S = A + \sum_i B_i$ is direct

iff $\text{rank } S = \text{rank } A + \sum_i \text{rank } B_i$

In Coq:

```
Lemma mxdirectP n (E : mxsum_expr n) :
   reflect (\rank E = mxsum_rank E) (mxdirect E).
```

This is generic in the *shape* of E

```
Let sumV := (\sum_(i < h) 'V_i)%MS.
(* This is B & G, Proposition 2.4(a) *)
Lemma mxdirect_sum_eigenspace_cycle :
   (sumV :=: 1%:M)%MS /\ mxdirect sumV.
```

Recurrenc

Proof. Again we use induction for (a). For $n = 0$ we know
(a) is true by hypothesis. Now suppose that $n > 0$ and $L(G$
Then

$$L(G) \to L_{2n}(H).$$

Hence

$$L_{2n}(H) \subseteq L_G(L(G)) = L_*(G).$$

Furthermore,

$$L_{2n}(H) \to L_G(L_*(G)) = L(G) \subseteq H.$$

Thus

$$L(G) \subseteq L_{2n+1}(H).$$

Again, (b) follows from Lemma B.1(c). □

By Step 1 and Step 2 we can now conclude that $L(G)$
sired. □

Lemma B.3. Assume p is odd, G is solvable of odd order, a
Suppose that S is a Sylow p-subgroup of G and $T = O_p(G)$.

$$L_*(S) \subseteq L_*(T) \subseteq L(T) \subseteq L(S).$$

Proof. First we show by induction on n that for all $n \geq 0$,

(B.1) $$L_{2n}(S) \subseteq L_{2n}(T) \subseteq L_{2n+1}(T) \subseteq L_{2n+1}(S).$$

For $n = 0$ the statement reduces to

$$1 \subseteq 1 \subseteq T \subseteq S,$$

which is trivial.

Assume (B.1) holds for some n. Since $L_{2n+1}(S) \to L_{2n+2}$

(B.2) $$L_{2n+1}(T) \to L_{2n+2}(S).$$

Now $L_{2n+1}(T)$ is a normal p-subgroup of G and, by Lemma

$$L_{2n+1}(T) \supseteq C_T(L_{2n+1}(T)).$$

Thus, by (B.2) and Theorem A.5,

$$L_{2n+2}(S) \subseteq T.$$

Hence, by (B.2),

(B.3) $$L_{2n+2}(S) \subseteq L_T(L_{2n+1}(T)) = L_{2n+2}(T).$$

Consequently, by Lemma B.1(a),

(B.4) $$L_{2n+3}(T) = L_T(L_{2n+2}(T)) \subseteq L_T(L_{2n+2}(S)) \subseteq L_S(L_{2n+2}(S))$$

By Lemma B.1(b),

```
Theorem Puig_center_normal : 'Z(L) <| G.
Proof.
have [sLiST sLTS] := pcore_Sylow_Puig_sub.
have sLiLT: 'L_*(T) \subset 'L(T) by exact: Puig_sub_even_odd.
have sZY: 'Z(L) \subset Y.
  rewrite subsetI andbC subIset ?centS ?orbT //=.
  suffices: 'C_S('L_*(S)) \subset 'L(T).
    by apply: subset_trans; rewrite setISS ?Puig_sub ?centS ?Puig_sub_even_odd.
  apply: subset_trans (subset_trans sLiST sLiLT).
  by apply: sub_cent_Puig_at pS; rewrite double_gt0.
have chY: Y \char G := char_trans (center_Puig_char _) (pcore_char _ _).
have nsCY_G: 'C_G(Y) <| G by rewrite char_normal 1?subcent_char ?char_refl.
have [C defC sCY_C nsCG] := inv_quotientN nsCY_G (pcore_normal p _).
have sLG: L \subset G by rewrite (subset_trans _ (pHall_sub sylS)) ?Puig_sub.
have nsL_nCS: L <| 'N_G(C :&: S).
  have sYLiS: Y \subset 'L_*(S).
    rewrite abelian_norm_Puig ?double_gt0 ?center_abelian //.
    apply: normalS (pHall_sub sylS) (char_normal chY).
    by rewrite subIset // (subset_trans sLTS) ?Puig_sub.
  have gYL: Y --> L := norm_abgenS sYLiS (Puig_gen _ _).
  have sLCS: L \subset C :&: S.
    rewrite subsetI Puig_sub andbT.
    rewrite -(quotientSGK _ sCY_C) ?(subset_trans sLG) ?normal_norm // -defC.
    rewrite odd_abelian_gen_stable ?char_normal ?norm_abgen_pgroup //.
      by rewrite (pgroupS _ pT) ?subIset // Puig_sub.
    by rewrite (pgroupS _ pS) ?Puig_sub.
  rewrite -[L](sub_Puig_eq _ sLCS) ?subsetIr //.
  by rewrite (char_normal_trans (Puig_char _)) ?normalSG // subIset // sSG orbT.
have sylCS: p.-Sylow(C) (C :&: S) := Sylow_setI_normal nsCG sylS.
have{defC} defC: 'C_G(Y) * (C :&: S) = C.
  apply/eqP; rewrite eqEsubset mulG_subG sCY_C subsetIl /=.
  have nCY_C: C \subset 'N('C_G(Y)).
    exact: subset_trans (normal_sub nsCG) (normal_norm nsCY_G).
  rewrite -quotientSK // -defC /= -pseries1.
  rewrite -(pseries_catr_id [:: p : nat_pred]) (pseries_rcons_id [::]) /=.
  rewrite pseries1 /= pseries1 defC pcore_sub_Hall // morphim_pHall //.
  by rewrite subIset ?nCY_C.
have defG: 'C_G(Y) * 'N_G(C :&: S) = G.
  have sCS_N: C :&: S \subset 'N_G(C :&: S).
    by rewrite subsetI normG subIset // sSG orbT.
  by rewrite -(mulSGid sCS_N) mulgA defC (Frattini_arg _ sylCS).
have nsZ_N: 'Z(L) <| 'N_G(C :&: S) := char_normal_trans (center_char _) nsL_nCS.
rewrite /normal subIset ?sLG //= -{1}defG mulG_subG /=.
rewrite cents_norm ?normal_norm // centsC.
by rewrite (subset_trans sZY) // centsC subsetIr.
Qed.
```

Telescopic algebra

150 C. The Final Contradiction

Thus, if $k \in \mathbf{F}_p$ and $\ell = k - 2$, after multiplying on the left and on the right by $t^{k-2} = t^\ell$ we have

(C.3)

$$\underbrace{t^{-\ell}s^{\ell}s^{-k}t^{k}}_{s^{-k}t^{k}t^{-\ell}s^{\ell}}(a^{-1})^{t^k}\underbrace{t^{-k}s^{k}s^{-k+1}t^{k-1}}_{s^{-k+1}t^{k-1}t^{-k}s^{k}}(ab^{-1})^{t^{k-1}}\underbrace{t^{-k+1}s^{k-1}s^{-\ell}}_{s^{-\ell}t^{\ell}t^{-k+1}s^{k-1}}$$

Now observe that

$$s^{-i}t^{i} = s^{-i}(s^{y})^{i} = s^{-i}(y^{-1}sy)^{i} = [s^{i}, y] \in Q$$

Since Q is commutative, (C.3) becomes

(C.4)

$$s^{-k}t^{2}\underbrace{s^{k-2}(a^{-1})^{t^k}s^{-k+1}}_{u_1 s_1 v_1}t^{-1}\underbrace{s^{k}(ab^{-1})^{t^{k-1}}s^{-k+2}}_{u_2 s_2 v_2}t^{-1}\underbrace{s^{k-1}b^{t}}_{u_3}$$

By Step 1, there are elements u_i, and $v_i \in U$ and $s_i \in P_0$, that

$$u_1 s_1 v_1 = s^{k-2}(a^{-1})^{t^k}s^{-k+1}$$

(C.5)

$$u_2 s_2 v_2 = s^{k}(ab^{-1})^{t^{k-1}}s^{-k+2}$$

$$u_3 s_3 v_3 = s^{k-1}b^{t^{k-2}}s^{-k}$$

and by Steps 2 and 3

(C.6) $$s_i \neq 1 \qquad (i = 1, 2, 3).$$

If we multiply equation (C.4) on the left by s^k and on the use equation (C.5) we have

$$t^{2}u_1 s_1 v_1 t^{-1}u_2 s_2 v_2 t^{-1}u_3 s_3 v_3 = 1, \text{ and henc}$$

$$u_1^{t^{-2}}t^{2}s_1\underbrace{v_1 u_2^{t}}_{w_3}t^{-1}s_2 t^{-1}\underbrace{v_2^{t^{-1}}u_3}_{w_1}s_3\underbrace{v_3 u_1^{t^{-2}}}_{w_2}(u_1^{-1})^{t^{-2}}$$

If we set

$$w_1 = v_2^{t^{-1}}u_3, \qquad w_2 = v_3 u_1^{t^{-2}}, \qquad \text{and} \qquad w_3 =$$

then $w_i \in U$ and

(C.7) $$t^{-1}s_2 t^{-1} = (w_1 s_3 w_2 t^{2} s_1 w_3)^{-1}.$$

Next we show that (C.5) holds with a, b, u_i, and v_i re u_i^p, and v_i^p, respectively. We prove only the first equation s

```
have [[Ua Uu1 Uv1 P0s1 Dusv1] /sUs_modP-Duv1] := (usv1P, usv1P).
have [[_ Uu2 Uv2 P0s2 _] [Ub Uu3 Uv3 P0s3 _]] := (usv2P, usv3P).
suffices /(congr1 sigma): s ^+ 2 = s ^ v1 * s ^ a^-1 ^ t ^+ 3.
  rewrite inE sigmaX // sigma_s sigmaM ?memJ_P -?psiE ?nUtn // => ->.
  by rewrite addrK -!im_psi !mem_imset ?nUtn.
rewrite groupV in Ua; have [Hs1 Hs3]: s1 \in H /\ s3 \in H by rewrite !sP0H.
have nt_s1: s1 != 1 by apply: nt_sUs usv1P.
have nt_s3: s3 != 1 by apply: nt_sUs usv3P.
have{sUsXp} Ds2p: s2def (w1 ^+ p) (w2 ^+ p) (w3 ^+ p).
  have [/sUsXp-usv1pP /sUsXp-usv2pP /sUsXp-usv3pP] := And3 usv1P usv2P usv3P.
  rewrite expUMp ?groupV // !expgVn in usv1pP usv2pP.
  rewrite !(=^~ conjXg _ _ p, expUMp) ?groupV -1?[t]expg1 ?nUtn ?nUtVn //.
  apply: Ds2 usv1pP usv2pP usv3pP => //.
  by rewrite !psiX // -!Frobenius_autE -rmorphD Dab rmorph_nat.
have{Ds2} Ds2: s2def w1 w2 w3 by apply: Ds2 usv1P usv2P usv3P.
wlog [Uw1 Uw2 Uw3]: w1 w2 w3 Ds2p Ds2 / [/\ w1 \in U, w2 \in U & w3 \in U].
  by move/(_ w1 w2 w3)->; rewrite ?(nUtVn, nUtVn 1%N, nUtn 1%N, in_group).
have{Ds2p} Dw3p: (w2 ^- p * w1 ^- p.-1 ^ s3 * w2) ^ t ^+ 2 = w3 ^+ p.-1 ^ s1^-1.
  rewrite -[w1 ^+ _](mulKg w1) -[w3 ^+ _](mulKg w3) -expgS -expgSr !prednK //.
  rewrite -(canLR (mulKg _) Ds2p) -(canLR (mulKg _) Ds2) 6!invMg !invgK.
  by rewrite mulgA mulgK [2]lock /conjg !mulgA mulVg mul1g mulgK.
have w_id w: w \in U -> w ^+ p.-1 == 1 -> w = 1.
  by move=> Uw /eqP/(canRL_in (expgK _) Uw)->; rewrite ?expg1n ?oU.
have{Uw3} Dw3: w3 = 1.
  apply: w_id => //; have:= @not_splitU s1^-1^-1 s1^-1 (w3 ^+ p.-1).
  rewrite !groupV mulVg eqxx andbT {2}invgK (negPf nt_s1) groupX //= => -> //.
  have /tiH_P1 <-: t ^+ 2 \in P1^#.
    rewrite 2!inE groupX // andbT -order_dvdn gtnNdvd // orderJ.
    by rewrite odd_gt2 ?order_gt1 // orderE defP0 (oddSg sP0P).
  by rewrite -mulgA -conjgE inE -{2}Dw3p memJ_conjg !in_group ?Hs1 // sUH.
have{Dw3p} Dw2p: w2 ^+ p.-1 = w1 ^- p.-1 ^ s3.
  apply/(mulIg w2)/eqP; rewrite -expgSr prednK // eq_mulVg1 mulgA.
  by rewrite (canRL (conjgK _) Dw3p) Dw3 expg1n !conj1g.
have{Uw1} Dw1: w1 = 1.
  apply: w_id => //; have:= @not_splitU s3^-1 s3 (w1 ^- p.-1).
  rewrite mulVg (negPf nt_s3) andbF -mulgA -conjgE -Dw2p !in_group //=.
  by rewrite eqxx andbT eq_invg1 /= => ->.
have{w1 w2 w3 Dw1 Dw3 w_id Uw2 Dw2p Ds2} Ds2: t * s2^-1 * t = s3 * t ^+ 2 * s1.
  by rewrite Ds2 Dw3 [w2]w_id ?mulg1 ?Dw2p ?Dw1 ?mul1g // expg1n invg1 conj1g.
```

Proof by reflecti

Assume that (3.5) has been shown. Set $\omega_{ij}^{\sigma} = \chi_{ij}$ and extend σ to CF(W) by linearity. Then (a) and (b) of Theorem (3.2) are established, and assertions (c) and (d) of Theorem (3.2) follow from (1.3).

Proof of (3.5).

(3.5.1) *Let $\beta_{ij} = \mathrm{Ind}_W^G \alpha_{ij} - 1_G$ $(1 \le i < w_1, 1 \le j < w_2)$. Then $(\beta_{ij}, 1_G) = 0$ and $\|\beta_{ij}\|^2 = 3$ for all i, j while $(\beta_{ij}, \beta_{ij'}) = (\beta_{ij}, \beta_{i'j}) = 1$ and $(\beta_{ij}, \beta_{i'j'}) = 0$ for $i \ne i'$, $j \ne j'$.*

Proof. That $(\mathrm{Ind}_W^G \alpha_{ij}, 1_G) = (\alpha_{ij}, 1_W) = 1$ follows from Frobenius reciprocity, and so $(\beta_{ij}, 1_G) = 0$. The other relations follow from the fact that Ind_W^G is an isometry on CF(W, V). by (2.6)(c) □

Let $1 \le i < w_1$, $1 \le j < w_2$. By (3.5.1) and the fact that $\beta_{ij} \in \mathbb{Z}[\mathrm{Irr}(G)]$, we see that $\beta_{ij} = \sum_{\chi \in A_{ij}} \chi$, where A_{ij} is a set of three pairwise orthogonal elements of $\pm(\mathrm{Irr}(G) - \{1_G\})$.

(3.5.2) *We have $|A_{11} \cap A_{12}| = 1$ and $A_{11} \cap (-A_{12}) = \emptyset$.*

Proof. Let $A_{11} = \{\chi_1, \chi_2, \chi_3\}$ and $a_i = (\beta_{12}, \chi_i)$ for $i = 1, 2, 3$. Then $(\beta_{12}, \beta_{11}) = a_1 + a_2 + a_3 = 1$ and $a_i \in \{0, 1, -1\}$. The numbers a_i are thus either $1, 0, 0$, or $1, 1, -1$. In the second case, we may assume that $\beta_{12} = \chi_1 + \chi_2 - \chi_3$ whence $2\chi_3 = \beta_{11} - \beta_{12} = \mathrm{Ind}_W^G(\alpha_{11} - \alpha_{12})$ vanishes on $1 \in G$, which is a contradiction. □

Lemma (3.5.2) clearly holds with A_{ij} and $A_{i'j'}$ in place of A_{11} and A_{12} if $i = i'$ and $j \ne j'$ or if $i \ne i'$ and $j = j'$. We refer to this lemma for A_{ij} and $A_{i'j'}$ as L($ij, i'j'$). We also refer to the statement $(\beta_{ij}, \beta_{i'j'}) = 0$ for $i \ne i'$ and $j \ne j'$ as O($ij, i'j'$).

By Hypothesis (3.1), $\sup(w_1, w_2) \ge 5$. By the symmetry between w_1 and w_2, we will assume

(3.5.3) $w_1 \ge 5$.

In the proof which follows, the functions χ_i and χ_{ij} are pairwise orthogonal elements of $\pm(\mathrm{Irr}(G) - \{1_G\})$.

(3.5.4) $|\bigcap_{1 \le i < w_1} A_{i1}| = 1$.

Proof. Suppose that (3.5.4) is false. By (3.5.2), we can then write, for some choice of indices $i = 1, 2, 3$,

$$\begin{aligned} \beta_{11} &= \chi_1 + \chi_2 + \chi_3, \\ \beta_{21} &= \chi_1 + \chi_4 + \chi_5, \\ \beta_{31} &= \chi_2 + \chi_4 + \chi_6. \end{aligned}$$

```
et unsat Ii : unsat |= & x1 in b11 & x1 in b21 & ~x1 in b31.
roof.
wlog Db11: (& b11 = x1 + x2 + x3) by do 2!fill b11.
wlog Db21: (& b21 = x1 + x4 + x5).
 by uhave ~x2, ~x3 in b21 as L(21, 11); do 2!fill b21; uexact Db21.
wlog Db31: (& b31 = x2 + x4 + x6).
 uwlog b31x2: x2 | ~x2 in b31 as L(31, 11).
   by uhave x3 in b31 as O(31, 11); symmetric to b31x2.
 uwlog b31x4: x4 | ~x4 in b31 as L(31, 21).
   by uhave x5 in b31 as O(31, 21); symmetric to b31x4.
 uhave ~x3 in b31 as O(31, 11); uhave ~x5 in b31 as L(31, 21).
 by fill b31; uexact Db31.
onsider b41; uwlog b41x1: x1 | ~x1 in b41 as L(41, 11).
 uwlog Db41: (& b41 = x3 + x5 + x6) => [|{b41x1}].
   uhave ~x2 | x2 in b41 as L(41, 11); last symmetric to b41x1.
   uhave ~x4 | x4 in b41 as L(41, 21); last symmetric to b41x1.
   uhave x3 in b41 as O(41, 11); uhave x5 in b41 as O(41, 21).
   by uhave x6 in b41 as O(41, 31); uexact Db41.
 consider b12; uwlog b12x1: x1 | ~x1 in b12 as L(12, 11).
   uhave ~x2 | x2 in b12 as L(12, 11); last symmetric to b12x1.
   by uhave x3 in b12 as O(12, 11); symmetric to b12x1.
 uwlog b12x4: -x4 | ~x4 in b12 as O(12, 21).
   by uhave -x5 in b12 as O(12, 21); symmetric to b12x4.
 uhave ~x2, ~x3 in b12 as L(12, 11); uhave ~x5 in b12 as O(12, 21).
 by uhave x6 in b12 as O(12, 31); counter to O(12, 41).
wlog Db41: (& b41 = x1 + x6 + x7).
 uhave ~x2, ~x3 in b41 as L(41, 11); uhave ~x4, ~x5 in b41 as L(41, 21).
 by uhave x6 in b41 as O(41, 31); fill b41; uexact Db41.
onsider b32; uwlog Db32: (& b32 = x6 - x7 + x8).
 uwlog b32x6: x6 | ~x6 in b32 as L(32, 31).
   uhave ~x2 | x2 in b32 as L(32, 31); last symmetric to b32x6.
   by uhave x4 in b32 as O(32, 31); symmetric to b32x6.
 uhave ~x2, ~x4 in b32 as L(32, 31).
 uhave -x7 | ~x7 in b32 as O(32, 41).
   uhave ~x1 in b32 as O(32, 41); uhave ~x3 in b32 as O(32, 11).
   by uhave ~x5 in b32 as O(32, 21); fill b32; uexact Db32.
 uhave -x1 in b32 as O(32, 41).
 by uhave x3 in b32 as O(32, 11); counter to O(32, 21).
onsider b42; uwlog Db42: (& b42 = x6 - x4 + x5).
 uhave ~x6 | x6 in b42 as L(42, 41).
   uhave ~x7 | x7 in b42 as L(42, 41); last counter to O(42, 32).
   uhave x1 in b42 as O(42, 41); uhave x8 in b42 as O(42, 32).
   uhave ~x2 | -x2 in b42 as O(42, 11); last counter to O(42, 21).
-(Unix)--  PFsection3.v   59% L1115 SVN-4447  (coq Scripting *3 SUBGOALS*
 b41x1 : unsat
           |= & b11 = x1 + x2 + x3
               & b21 = x1 + x4 + x5
               & b31 = x2 + x4 + x6
               & x1 in b41
 Db41 : unsat
          |= & b11 = x1 + x2 + x3
              & b21 = x1 + x4 + x5
              & b31 = x2 + x4 + x6
              & b41 = x3 + x5 + x6
 ============================
  unsat
    |= & b11 = x1 + x2 + x3
        & b21 = x1 + x4 + x5
        & b31 = x2 + x4 + x6
        & ~x1, ~x2 in b41
```

Wandering typo

- B & G 15.7
  - .. (e)(2) p = |X| is a prime in σ(M) – β(M), $O_p(H)$ is not abelian, $O_{p'}(\mathbf{H})$ is cyclic, …

Theorem 15.7. Suppose $F(M)$ is not a TI-subgroup of G. Let $H = M_F$ and choose $g \in G - M$ such that $X = F(M) \cap F(M)^g$ is not trivial. Take E, E_1, E_2, E_3 as in Sections 12-13. Then

(a) $M \in \mathcal{M}_{\mathcal{F}} \cup \mathcal{M}_{\mathcal{P}_1}$ and $H = M_\sigma$,
(b) $X \subseteq H$ and X is cyclic,
(c) $M' \subseteq F(M) = M_\sigma \times \mathcal{O}_{\sigma(M)'}(F(M))$,
(d) $E_3 = 1$, $E_2 \lhd E$, and $E/E_2 \cong E_1$, which is cyclic, and
(e) one of the following conditions holds:
  (1) $M \in \mathcal{M}_{\mathcal{F}}$ and H is abelian of rank two,
  (2) $p = |X|$ is a prime in $\sigma(M) - \beta(M)$, $\mathcal{O}_p(H)$ is not abelian, $\mathcal{O}_{p'}(H)$ is cyclic, and the exponent of M/H divides $q-1$ for every $q \in \pi(H)$,
  (3) $p = |X|$ is a prime in $\sigma(M) - \beta(M)$, $\mathcal{O}_{p'}(H)$ is cyclic, $\mathcal{O}_p(H)$ has order p^3 and is not abelian, $M \in \mathcal{M}_{\mathcal{P}_1}$, and $|M/H|$ divides $p+1$.

Things to look forward to

- Certification
  - of computer computations
  - of complex proofs
- Collaboration
  - safe contributions from diverse backgrounds
- Inspiration
  - explore logic, dependencies, and factoring