Computer-Checked Mathematics
A Formal Proof of the Odd Order Theorem

Assia Mahboubi
Joint work with

The Mathematical Components team:

(Finite) Groups

Abstract structure for:

- (finite) sets of things
- that can be reversed and combined.

Typical example: operators that preserve a shape.
Pocket finite group theory
Many other examples...

\[ e = x^0 = x^{n+1} \]

\[ \mathbb{Z}_n, \text{ the cycle of order } n + 1 \]

...and applications: combinatorics, chemistry, cryptography,...
A (finite) group $G$ is:
- A (finite) set: $G$;
- A binary law: $g \ast h$ or $gh$;
- A neutral element: $1$;

such that:
- The group law is associative;
- Every element $g$ has an inverse: $g^{-1}$. 
Formal definition

A (finite) group $G$ is:
- A (finite) set: $G$;
- A binary law: $g * h$ or $gh$;
- A neutral element: 1;

such that:
- The group law is associative;
- Every element $g$ has an inverse: $g^{-1}$.

Vocabulary:
- Abelian group = commutative group
- Order of a group = cardinal of its underlying set
Classification

Given a property on finite groups, enumerate the possible isomorphism classes for each cardinal.
Classification

Given a property on finite groups, enumerate the possible isomorphism classes for each cardinal.

• Multiplication tables?
Classification

Given a property on finite groups, enumerate the possible isomorphism classes for each cardinal.

- Multiplication tables? Not informative enough.
Classification

Given a property on \textit{finite} groups, enumerate the possible isomorphism classes for each cardinal.

- Multiplication tables? Not informative enough.
- For instance, decomposition in smaller groups is better.
Classification of abelian groups

A finite abelian group $G$ is (isomorphic to) a product of cycles:

$$G = \mathbb{Z} / p_1^{k_1} \mathbb{Z} \times \cdots \times \mathbb{Z} / p_s^{k_s} \mathbb{Z}$$

with $p_1, \ldots, p_k$ prime numbers.

Remarks:

- The order of $G$ is equal to $p_1^{k_1} \ldots p_s^{k_s}$.
- Here the $(p_i)_{i=1\ldots s}$ are not necessarily distinct.
Example: groups of order 4

\[ \mathbb{Z}_4 \]

\[ n = 4 = 2^2 \]

\[ \mathbb{Z}_2 \times \mathbb{Z}_2 \]

\[ n = 4 = 2 \times 2 \]
Example: groups of order 4

\[ \mathbb{Z}_4 \]
\[ n = 4 = 2^2 \]

\[ \mathbb{Z}_2 \times \mathbb{Z}_2 \]
\[ n = 4 = 2 \times 2 \]
Cartesian product

Let $G$ and $H$ be two finite groups.

Then $G \times H$ has a structure of groups with law:

$$(g_1, h_1) \ast (g_2, h_2) := (g_1 \ast g_2, h_1 \ast h_2)$$

If $G$ and $H$ are abelian, then so is $G \times H$. 
Dihedral group $D_6$

This is a group of order 6.
Dihedral group $D_6$

This is a group of order 6.
It contains: $\{e, r, r^2\} \simeq \mathbb{Z}_3$, 
Dihedral group $D_6$

This is a group of order 6.
It contains: $\{e, r, r^2\} \cong \mathbb{Z}_3$, $\{e, s\} \cong \mathbb{Z}_2$
Dihedral group $D_6$

This is a group of order 6. It contains: $\{e, r, r^2\} \simeq \mathbb{Z}_3$, $\{e, s\} \simeq \mathbb{Z}_2$ and $rs$ and $r^2s$. 
Dihedral group $D_6$

This is a group of order 6.
It contains: $\{e, r, r^2\} \cong \mathbb{Z}_3$, $\{e, s\} \cong \mathbb{Z}_2$ and $rs$ and $r^2s$. 
Dihedral group $D_6$

This is a group of order 6.
It contains: $\{e, r, r^2\} \simeq \mathbb{Z}_3$, $\{e, s\} \simeq \mathbb{Z}_2$ and $rs$ and $r^2s$. 
Dihedral group $D_6$

This is a group of order 6.
It contains: $\{e, r, r^2\} \cong \mathbb{Z}_3$, $\{e, s\} \cong \mathbb{Z}_2$ and $rs$ and $r^2s$. 
Dihedral group $D_6$

This is a group of order 6.
It contains: $\{e, r, r^2\} \cong \mathbb{Z}_3$, $\{e, s\} \cong \mathbb{Z}_2$ and $rs$ and $r^2s$.

But $rs = sr^{-1} \neq sr : D_6$ is not $\mathbb{Z}_3 \times \mathbb{Z}_2$
Dihedral group $D_{2n}$

The carrier set of $D_{2n}$ “is” indeed $\mathbb{Z}_n \times \mathbb{Z}_2$, but its law is:

$$(\mathbb{Z}_n \times \mathbb{Z}_2)^2 \rightarrow \mathbb{Z}_n \times \mathbb{Z}_2$$

$$(r_1, s_1) \ast (r_2, s_2) = (r_1 r_2^{\epsilon(s_1)}, s_1 s_2)$$

with $\epsilon(s) = -1$ and $\epsilon(s) = 1$.

$D_{2n}$ is (isomorphic to) the semi-direct product $\mathbb{Z}_n \rtimes \mathbb{Z}_2$. 
Example: groups of order 6

\[ \mathbb{Z}_6 = \mathbb{Z}_3 \times \mathbb{Z}_2 \]

\[ D_6 \]
Example: groups of order 6

\[ \mathbb{Z}_6 = \mathbb{Z}_3 \times \mathbb{Z}_2 \]

\[ D_6 \]
Normal subgroups

Subgroups $H \triangleleft G$ do not necessarily define quotients $G/H$.

This requires a normal subgroup $H \triangleleft G$:

$$(Hg_1)(Hg_2) = Hg_1g_2 \quad \text{for every } g_1, g_2 \in G$$

or equivalently:

$$g^{-1}Hg = H \quad \text{for every } g \in G$$
A group $G$ is simple if it has no trivial normal subgroup:

- the only quotients $G/H$ are $G/1 = G$ and $G/G = 1$.
- they are building blocks, like primes to natural numbers.
Simple groups

A group G is simple if it has no trivial normal subgroup:

• the only quotients $G/H$ are $G/1 = G$ and $G/G = 1$.
• they are building blocks, like primes to natural numbers.

Examples:

• $\mathbb{Z}_p$ with $p$ a prime: cyclic groups;
• $A_n$ for $n \leq 5$: even permutations on $n$ letters;
• ...
Decomposition of a group

• Existence of prime decomposition:
  Any number \( n \) is a product of prime numbers.
  These primes are called the prime factors of \( n \).
Decomposition of a group

• Existence of prime decomposition:
  Any number $n$ is a product of prime numbers.
  These primes are called the prime factors of $n$.

• Existence of composition series:
  For any finite group $G$, there exists a sequence of subgroups:
  \[ \{e\} = G_0 \triangleleft G_1 \triangleleft \cdots \triangleleft G_n = G \]
  such that for all $k$, $G_{k+1}/G_k$ is simple.
  Quotients $G_{k+1}/G_k$ are called factors of $G$. 
Uniqueness of the decomposition

- Prime decomposition uniqueness
  For any number \( n \), two decompositions of \( n \) into prime factors are the same up to permutation.

- Jordan-Hölder uniqueness
  For any group \( G \), two composition series for \( G \) have the same factors up to (isomorphism and) permutation.
Classification

Unfortunately the analogy with arithmetic stops pretty early:

- Same (multiset of) prime factors $\Rightarrow$ equal numbers.
- Same (multiset of) factors $\not\Rightarrow$ isomorphic finite groups.

$\Rightarrow$ Classifying finite groups is difficult.
$\Rightarrow$ Classifying finite simple groups is already very difficult.
Unfortunately the analogy with arithmetic stops pretty early:

- Same (multiset of) prime factors $\Rightarrow$ equal numbers.
Classification

Unfortunately the analogy with arithmetic stops pretty early:

- Same (multiset of) prime factors $\Rightarrow$ equal numbers.
- Same (multiset of) factors $\not\Rightarrow$ isomorphic finite groups.
Classification

Unfortunately the analogy with arithmetic stops pretty early:

- Same (multiset of) prime factors $\Rightarrow$ equal numbers.
- Same (multiset of) factors $\not\Rightarrow$ isomorphic finite groups.

$\Rightarrow$ Classifying finite groups is difficult.
Classification

Unfortunately the analogy with arithmetic stops pretty early:

- Same (multiset of) prime factors $\Rightarrow$ equal numbers.
- Same (multiset of) factors $\not \Rightarrow$ isomorphic finite groups.

$\Rightarrow$ Classifying finite groups is difficult.
$\Rightarrow$ Classifying finite simple groups is already very difficult.
Classification of finite simple groups

Finite simple groups are:

- Cycles of prime order $\mathbb{Z}/p\mathbb{Z}$
- Alternating groups $A_n$ for $n \leq 5$
- Lie-type groups
- 26 sporadic groups.

Classification of finite simple groups

Finite simple groups are:

- Cycles of prime order \( \mathbb{Z}/p\mathbb{Z} \)
- Alternating groups \( A_n \) for \( n \leq 5 \)
- Lie-type groups
- 26 sporadic groups.

Considered completed in 2004, after a false start in 1981.

(Aschbacher, 2004)
Classification of finite simple groups

Finite simple groups are:

- Cycles of prime order $\mathbb{Z}/p\mathbb{Z}$ by the Odd order theorem
- Alternating groups $A_n$ for $n \leq 5$
- Lie-type groups
- 26 sporadic groups.

Considered completed in 2004, after a false start in 1981.

(Aschbacher, 2004)
The Odd Order Theorem

Theorem (Feit - Thompson, 1963):

Every simple group of odd order is cyclic.
The Odd Order Theorem

Theorem (Feit - Thompson, 1963):

Every simple group of odd order is cyclic.

- Conjectured (at least) by Burnside in 1911.
The Odd Order Theorem

Theorem (Feit - Thompson, 1963):

Every simple group of odd order is cyclic.

- Conjectured (at least) by Burnside in 1911.
The Odd Order Theorem

Theorem (Feit - Thompson, 1963):

Every simple group of odd order is cyclic.

- Conjectured (at least) by Burnside in 1911.
- Original published proof:
  one entire volume of the Pacific Journal of Mathematics

A collective simplification work ⇒ two volumes
(Bender - Glauberman, 1994; Peterfalvi, 2000).

Assia Mahboubi – Computer-Checked Mathematics
The Odd Order Theorem

Theorem (Feit - Thompson, 1963) :

Every finite group of odd order is solvable.

Theorem (Suzuki, 1957) :

Every (CA) group of odd order is solvable.

• $G$ is (CA) := $x \sim y$ iff $xy = yx$ is an equivalence over $G^\times$. 
The Odd Order Theorem

Common structure to (Suzuki, 1957) and (FT, 1963):

- Postulate a minimal counterexample $G$
The Odd Order Theorem

Common structure to (Suzuki, 1957) and (FT, 1963):

- Postulate a minimal counterexample $G$
- Study its maximal proper subgroups:
  - **Local analysis**: study of subgroups of order $p^k$
The Odd Order Theorem

Common structure to (Suzuki, 1957) and (FT, 1963):

- Postulate a minimal counterexample $G$
- Study its maximal proper subgroups:
  - Local analysis: study of subgroups of order $p^k$
  - Character theory: study traces of representations of a group (as matrices over $\mathbb{C}$ or $\mathbb{F}_p$)

But Suzuki’s original proof is much more concise . . .
The Odd Order Theorem

Common structure to (Suzuki, 1957) and (FT, 1963):

- Postulate a minimal counterexample $G$
- Study its maximal proper subgroups:
  - Local analysis: study of subgroups of order $p^k$
  - Character theory: study traces of representations of a group (as matrices over $\mathbb{C}$ or $\mathbb{F}_p$)
- Obtain a contradiction via (character) norm inequalities.
The Odd Order Theorem

Common structure to (Suzuki, 1957) and (FT, 1963):

- Postulate a minimal counterexample \( G \)
- Study its maximal proper subgroups:
  - Local analysis: study of subgroups of order \( p^k \)
  - Character theory: study traces of representations of a group (as matrices over \( \mathbb{C} \) or \( \mathbb{F}_p \))
- Obtain a contradiction via (character) norm inequalities.

But Suzuki’s original proof is much more concise . . .
Proof

Maximal proper subgroups of a minimal counterexample:
Proof

Maximal proper subgroups of a minimal counterexample:

Local Analysis

Types I II III IV V
Proof

Maximal proper subgroups of a minimal counterexample:

Local Analysis

Types I II III IV V

Character Theory
Proof

Maximal proper subgroups of a minimal counterexample:

Local Analysis

Types I / II / III / IV / V

Character Theory
Proof

Maximal proper subgroups of a minimal counterexample:

- Character Theory
- Local Analysis
- Galois Theory

Types I, II, III, IV, V
Proof

Maximal proper subgroups of a minimal counterexample:

Local Analysis

Galois Theory

Character Theory

Types
Proof

Maximal proper subgroups of a minimal counterexample:

Local Analysis

Galois Theory

Character Theory

Types 1 1 1 1 1 1
Ingredients

- Combinatorics, finite group theory
- Elementary arithmetic, elementary modular arithmetic
- Linear algebra
- Complex (algebraic) numbers
- Representation theory (complex and modular) of finite groups
- Character theory
- Finite fields, Galois theory
The Coq Proof Assistant

Formal Logic
Proof Assistant
Libraries

Type Theory
Coq
MathComp, Compcert, LoCo,...
Types, intuitively

Labels for the locus/range of arguments and values of functions:

\[ f : \mathbb{R} \rightarrow \mathbb{R} \quad \text{g} : \mathbb{Z}/5\mathbb{Z} \rightarrow \mathbb{Z}/5\mathbb{Z} \]

\[ x \mapsto x + 1 \quad x \mapsto x + 1 \]
Types, intuitively

Labels for the locus/range of arguments and values of functions:

\[
\begin{align*}
f & : \mathbb{R} \rightarrow \mathbb{R} \\
x & \mapsto x + 1
\end{align*}
\]

\[
\begin{align*}
g & : \mathbb{Z}/5\mathbb{Z} \rightarrow \mathbb{Z}/5\mathbb{Z} \\
x & \mapsto x + 1
\end{align*}
\]

Sometimes functions have parameters like \( n \in \mathbb{N} \) in:

\[
\begin{align*}
f_n & : \mathbb{C} \rightarrow \mathbb{C} \\
x & \mapsto x^n
\end{align*}
\]
Coq kernel

The task of the Coq kernel is to check typing judgments:

\[ x_1 : T_1, \ldots, x_n : T_n \vdash t : T \]

- \( x_1, \ldots, x_n \) are variables;
- \( T_1, \ldots, T_n, T \) are types;
- \( x_1 : T_1, \ldots, x_n : T_n \) is a context;
- \( t \) is a term.

The judgment is read:

“In the context \( x_1 : T_1, \ldots, x_n : T_n \), the term \( t \) has type \( T \).”
Terms and Types

Terms include the usual terms of the $\lambda$-calculus:

- Variables: $x, A, \ldots$
- Functions: $(\text{fun } x \mapsto t)$, with $x$ a variable and $t$ a term
- Applications: $t_1 (t_2)$
- Constants: $c$

The rules defining what is a valid judgment explain how we can assign a type to a term, like:

$$n : \mathbb{N} \vdash \text{fun } x \mapsto x^n : \mathbb{C} \to \mathbb{C}$$
Two issues in formalization

- Find the right definition of objects;
- Find the right tools to assist formal proofs.
The Odd Order Theorem, formally

The statement of the formalized theorem is:

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

What do (gt : finGroupType) and (G : {group gT}) mean?
Subgroups

Most of finite group theory is about combining (sub)groups:

\[ G \cap H, \quad G \times H, \quad G \rtimes H, \frac{G}{H} \ldots \]
Subgroups

Most of finite group theory is about combining (sub)groups:

\[ G \cap H, \quad G \times H, \quad G \rtimes H, \quad G/H \ldots \]

- Mathematical notations often hide two levels:
  - sets
  - algebraic properties
Subgroups

Most of finite group theory is about combining (sub)groups:

\[ G \cap H, \quad G \times H, \quad G \rtimes H, \quad G/H \ldots \]

- Mathematical notations often hide two levels:
  - sets
  - algebraic properties
- Heterogeneous operations would be very inconvenient.
Group Types

A finGroupType is:

• A finite type;
• With an element called 1;
• Equipped with an associative binary operation $\ast$ and an inverse operation.
Group Types

A \texttt{finGroupType} is:

- A finite type;
- With an element called 1;
- Equipped with an associative binary operation \( \ast \) and an inverse operation.

A group \( G : \{\text{group } gT\} \), for \( gt : \text{finGroupType} \) is:

- A finite set of \( gT \);
- Which contains 1;
- And is stable under \( \ast \).
Group Types

• The (sub)set $K$ is a group of $gT : \text{finGroupType}$.
• The (sub)set $A$ is not a group.
Group Types

Distinct group types are defined for:

- Permutations of a set $T$;
- $\mathbb{Z}_n$ for $n > 0$;
- \ldots
- Quotients of the form $./H$ for each distinct group $H$. 
Group Types

Distinct group types are defined for:
- Permutations of a set $T$;
- $\mathbb{Z}_n$ for $n > 0$;
- ... 
- Quotients of the form $./H$ for each distinct group $H$.

Benefits:
- Homogeneous operations: $A \times B$, $A \cap B$, $A \rtimes B$, ...
- Constructions generalized to subsets (of a group): $N(A)$, $A/B$, ...
Finding the right representation

By:

• possibly generalizing the standard constructions;
• making operations total as often as possible;
• avoiding too fine-grained types.
Unification is an important component of the proof assistant.
Unification

Helps limiting the input required from the user in the commands building proofs:

\[ a : \text{nat} \]
\[ b : \text{nat} \]
\[ H : \text{forall } x : \text{nat}, x + 0 = x \]

\[ (a + b) + 0 = b + a \]
Unification

Helps limiting the input required from the user in the commands building proofs:

\[ a : \text{nat} \]
\[ b : \text{nat} \]
\[ H : \forall x : \text{nat}, x + 0 = x \]

\[ \text{rewrite } H. \]

\[ (a + b) + 0 = b + a \]
Unification

Helps limiting the input required from the user in the commands building proofs:

\[
\begin{align*}
a & : \text{nat} \\
b & : \text{nat} \\
H & : \text{forall } x : \text{nat}, x + 0 = x \\
\end{align*}
\]

\[\text{------------------------------- rewrite } H.\]

\[\begin{align*}
(a + b) = b + a \\
\end{align*}\]
Unification

Helps limiting the input required from the user in the commands building proofs:

\[ a : \text{nat} \]
\[ b : \text{nat} \]
\[ H : \text{forall } x : \text{nat}, x + 0 = x \]

\[ \text{rewrite } H. \]
\[ (a + b) = b + a \]

The pattern of the equation has been matched against the goal.
Unification is not enough

Lemma \texttt{cardG_gt0} \ gT \ (G : \{\text{group } gT\}) : 0 < \#\mid \text{set_of_grp } G \mid.
Unification is not enough

Lemma \texttt{cardG_gt0} \ gT \ (G : \{\text{group} \ gT\}) : 0 < \#| \text{set_of.grp} \ G|.

How to apply this theorem to:

\begin{align*}
G : & \{\text{group} \ gT\} \\
H : & \{\text{group} \ gT\} \\
\text{-----------------------------------------} \\
0 : & < \#| (\text{set.of.grp} \ H) \cap ('N(\text{set.of.grp} \ G)) | \\
\end{align*}
Unification is not enough

Lemma \texttt{cardG_gt0} \ gT \ (G : \{\text{group gT}\}) : 0 < \#|\text{set_of_grp G}|.

How to apply this theorem to:

\begin{align*}
G & : \{\text{group gT}\} \\
H & : \{\text{group gT}\} \\
\end{align*}

\begin{align*}
\text{-----------------------------------------} \\
0 & < \#|(\text{set_of_grp H}) \cap ('N(\text{set_of_grp G}))| \\
\end{align*}

which requires finding a group ? such that:

\begin{align*}
(\text{set_of_grp H})\cap (\text{'}N(\text{set_of_grp H})) & \equiv \text{set_of_grp ?} \\
\end{align*}
Unification is not enough

**Lemma** \texttt{cardG_gt0} \( gT \) (\( G : \{\text{group } gT\} \)) : \( 0 < \#| \text{set_of_grp } G| \).

How to apply this theorem to:

\[
\begin{align*}
& G : \{\text{group } gT\} \\
& H : \{\text{group } gT\} \\
& \text{----------------------------------}
\end{align*}
\]

\[
0 < \#| (\text{set_of.grp } H) \cap (\text{’N(set_of.grp G)}) |
\]

which requires finding a group \( ? \) such that:

\[
(\text{set_of.grp } H) \cap (\text{’N(set_of.grp H)}) \equiv \text{ set_of.grp } ?
\]

This requires **more** than unification...
Extending unification

Finding a group \( ? \) such that:
\[
(set_{of\_grp} H) \cap (\text{\texttt{'N}}(set_{of\_grp} G)) \equiv set_{of\_grp} ?
\]
is possible because we know that:
- For any groups \( K_1, K_2, K_1 \cap K_2 \) is a group;
- For any group \( K, N(K) \) is a group;

by canonical constructions.
Extending unification

Not only do we prove these theorems in the formal library, but we augment the unification algorithm with the following rules:

\[
\begin{align*}
K \text{ is a group} \\
N(K) \text{ is a group}
\end{align*}
\]

\[
\begin{align*}
K_1 \text{ is a group} & \quad K_2 \text{ is a group} \\
K_1 \cap K_2 \text{ is a group}
\end{align*}
\]

and let the unification algorithm trigger a Prolog like search.
Lemma \texttt{cardG_gt0 gT} \ G : \{\text{group gT}\} : \ 0 < \ | \ \text{set_of_grp} \ G|.

The command \texttt{(apply cardG_gt0)} solves the goal

\begin{align*}
G : \{\text{group gT}\} \\
H : \{\text{group gT}\}
\end{align*}

\begin{align*}
0 < |(\text{set_of_grp} \ H) \cap (\text{`}N(\text{set_of_grp} \ G))| \\
\end{align*}

using the canonical constructions of instances of groups.
Extending unification

Lemma \texttt{cardG_gt0} \ gT (G : \{\text{group gT}\}) : 0 < \#| \ G |.

The command \texttt{(apply cardG_gt0)} solves the goal

\begin{verbatim}
G : \{\text{group gT}\}
H : \{\text{group gT}\}
========================================
0 < \#|H \cap \ 'N(G)|
\end{verbatim}

and the function \texttt{set_of_grp} is in fact not displayed (nor input).
Check your favourite undergraduate linear algebra textbook:

“Let $A$ be a square matrix. Then:

$$\text{Det} \ (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i),i}.$$
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\text{Det} \ (A) = \sum_{\sigma \in S_n} \varepsilon_\sigma \prod_i a_{\sigma(i), i}$$
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\text{Det } (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i),i}$$

- $A$ is square
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\ Det \ (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i),i}$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$. 
Let $A$ be a square matrix. Then:

$$\text{Det } (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_i a_{\sigma(i),i}$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$.
- $\sum$
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\text{Det } (A) = \sum_{\sigma \in S_n} \epsilon_\sigma \prod_i a_{\sigma(i),i}$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$.
- $\Sigma$ denotes the iteration of a binary, commutative, associative operation with a neutral element.
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\det(A) = \sum_{\sigma \in S_n} \varepsilon_{\sigma} \prod_{i} a_{\sigma(i),i}$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$.
- $\sum$ denotes the iteration of a binary, commutative, associative operation with a neutral element.
- $\prod$
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\text{Det} (A) = \sum_{\sigma \in S_n} \epsilon_\sigma \prod_{i} a_{\sigma(i),i}$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$.
- $\Sigma$ denotes the iteration of a binary, commutative, associative operation with a neutral element.
- $\Pi$ denotes the iteration of a binary, commutative, associative operation with a neutral element, which is distributive over the one denoted by $\Sigma$. 
Let $A$ be a square matrix. Then:

$$\text{Det } (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i),i}$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$.
- $\Sigma$ denotes the iteration of a binary, commutative, associative operation with a neutral element.
- $\Pi$ denotes the iteration of a binary, commutative, associative operation with a neutral element, which is distributive over the one denoted by $\Sigma$.
- Obviously,
Mathematics on the paper

Let $A$ be a square matrix. Then:

$$\text{Det} (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i)},i$$

- $A$ is square of size $n \times n$, hence the type of permutations $S_n$ and the range of the index $i$.
- $\Sigma$ denotes the iteration of a binary, commutative, associative operation with a neutral element.
- $\Pi$ denotes the iteration of a binary, commutative, associative operation with a neutral element, which is distributive over the one denoted by $\Sigma$.
- Obviously, the coefficients of the matrix should live in a set equipped with a ring structure, and the iterated operations are the addition and product of that ring respectively.
Mathematics for the computer

Let $A$ be a square matrix. Then:

$$\text{Det} (A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_i a_{\sigma(i),i}$$
Let $A$ be a square matrix. Then:

$$\text{det}(A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i),i}$$

- In LaTeX:
  \[
  \textsf{det}(A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_{i} a_{\sigma(i),i}
  \]

- In Coq:
  [Coq code]

Assia Mahboubi – Computer-Checked Mathematics
Let $A$ be a square matrix. Then:

$$\text{Det}(A) = \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_i a_{\sigma(i),i}$$

- In $\LaTeX$:
  \begin{align*}
  \text{Det}(A) &= \sum_{\sigma \in S_n} \epsilon_{\sigma} \prod_i a_{\sigma(i),i} \\
  \text{In Coq:}
  \begin{align*}
  \text{Definition det} \ (R : \text{ringType}) \ n \ (A : \ 'M[R\_n]) : \ R := \\
  \sum_{s : \ 'S_n}
  \end{align*}
  \end{align*}
Programmable type inference

It acts like the mind of a trained mathematical reader as it:

- Restores (computationally) the assumptions left implicit in paper description;
- Decreases the amount of information provided by the user;
- Helps keeping the statements readable;
- Perform proof automation.

In practice: type inference + proof-search in unification.
Bookshelf for the Odd Order Theorem
Implementations of several algorithms of interest, together with correctness proofs:

- Depth-first search
- Factorization in primes
- (pseudo-)Euclidean division(s)
- Gaussian elimination, LUP matrix decomposition
- Quantifier elimination(s)
- ...

Most of them are used for constructing abstract witnesses rather than for performing large scale computations.
Specific issues

- No heavy computations
- No generic purpose automated reasoning was used
- No domain-specific proof search tool was needed
- No widely branching tree like structures and nested case/induction reasoning

But hopefully a large body of reusable libraries of formal algebra.
Thank you

(Picture courtesy of Alejandro Guijarro)