SSReflect, A Small Scale Reflection Extension for the Coq system

Robbert Krebbers [1]

June 26, 2009

[1] Student number: s0513229, e-mail: robbertkrebbers@student.ru.nl

Coq

- Development started in 1984 at INRIA
- Current version 8.2
- Based on intuitionistic type theory
- Written in Objective Caml with a bit of C
- Correctness relies on a not so small kernel (16587 lines)
- Distributed under the LGPL

SSReflect

- Coq extension
- Development started by George Gonthier for the formalization of the Four Colour theorem
- Currently maintained by the Mathematical Components team of Microsoft Research/INRIA
- Current version 1.1, compatible with Coq 8.1
- Distributed under the CeCill-B license

SSReflect

Download and documentation

- Home page
  http://www.msr-inria.inria.fr/Projects/math-components
- Documentation
  - Written by George Gonthier and Assia Mahboubi
  - 78 pages
  - Assumes you are highly experienced with Coq

Users

- Mainly used at Microsoft Research/INRIA
- Based in Orsay and Sophia Antipolis
- Respectively 5 and 6 researchers

George Gonthier

Team leader

Research interests:

- Programming language design and semantics
- Concurrency theory
- Its application to security
- Methods and tools for the formal verification

Benjamin Werner

Arithmetic leader

Research interests:

- Formalization of mathematical reasoning
- Mechanical verification through proof systems
- Proofs involving computations and evolutions of type theory

Projects

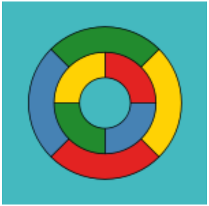
Mainly for *very long* and *non-trivial* formalizations

1. Four Colour Theorem
2. Cayley-Hamilton Theorem
3. Feit-Thompson Theorem

Four Colour Theorem

Four Colour Theorem: *The regions of any simple planar map can be coloured with only four colours, in such a way that any two adjacent regions have different colours.*

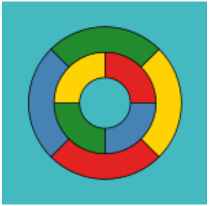
- First stated in 1852 by Francis Guthrie
- Lots of false proofs and counterexamples given

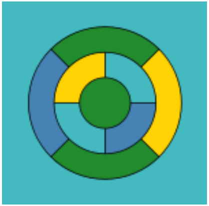


Four Colour Theorem

Four Colour Theorem: *The regions of any simple planar map can be coloured with only four colours, in such a way that any two adjacent regions have different colours.*

- First stated in 1852 by Francis Guthrie
- Lots of false proofs and counterexamples given





Four Colour Theorem

Heinrich Heesch

- Heinrich Heesch developed methods for proof search by a computer in 1970
- Developed a test for the four color theorem
- Did not have enough computer time

Four Colour Theorem

Appel and Haken

- Proven by Appel and Haken in 1976 using a computer
- Enormous case analysis
  - Checked 1936 configurations
  - 400 pages of microfiche had to be checked by hand
- Proof not accepted by many mathematicians
  - Unreadable IBM 370 assembly program
  - Computer programming is known to be error prone
- In 1980 rumours about a flaw in Appel and Haken's proof

Four Colour Theorem

Robertson, Sanders, Seymour and Thomas

- Proven by Robertson, Sanders, Seymour and Thomas in 1995
- Based on proof by Appel and Haken
- C program instead of assembly

Four Colour Theorem

George Gonthier

- Proven in 2005 by George Gonthier
- Using SSREFLECT for COQ 7.3.1
- Final step to remove all doubts
- 53282 lines COQ code
- ```
  Variable R : real_model.
  Theorem four_color : (m : (map R))
    (simple_map m) -> (map_colorable (4) m).
  Proof.
  Exact (compactness_extension four_color_finite).
  Qed.
  ```
  ```

Cayley-Hamilton Theorem

Cayley-Hamilton Theorem: *Every square matrix over the real or complex field satisfies its own characteristic equation.*

- Proven by Sidi Ould Biha in 2008 using SSReflect
- Resulted in a library to describe polynomials

Feit-Thompson Theorem

Feit-Thompson Theorem: *Every finite group of odd order is solvable*

Definition: *A group is* solvable *if it has a normal series whose factor groups are all abelian*

Feit-Thompson Theorem

- Historical proof of 255 pages
- It takes a professional group theorist a year to understand
- Unavoidable that flaws exist in the proof
- Start of the classification of finite simple groups
- George Gonthier et al. started a project to formalize this using SSReflect

Implementation

- Extension of the proof language
  4388 lines of Ocaml
- Basic Library
  6886 lines of Coq/Gallina

Proof language
Chaining

- Write very compact proofs
- Do a lot of bookkeeping meanwhile
- Regular Coq

```
generalize n m le_n_m.
clear n m le_n_m.
elim; [intros m _ | intros n IHn m lt_n_m].
```

- Becomes in SSReflect

```
elim: n m le_n_m => [|n IHn] m => [_ | lt_n_m].
```

Proof language

- `rewrite` tactic heavily extended
- `apply` more robust
- `last` ⟨goal⟩ `first` instead of `Focus` ⟨goal⟩
- `by` to terminate goals
- `have` for backwards reasoning
- Indentation and bullets allowed

Libraries

Propositions and booleans

- Coq is intuitionistic
- Logical propositions are of type `Prop`
- $\forall_{P:\texttt{Prop}}[P \vee \neg P]$ not provable

Libraries

Propositions and booleans

- Coq is intuitionistic
- Logical propositions are of type `Prop`
- $\forall_{P:\texttt{Prop}}[P \vee \neg P]$ not provable
- `bool` is an inductive type: `bool : true | false`
- $\forall_{b:\texttt{bool}}[b \mid\mid\ \sim b = \texttt{true}]$ is provable
- Because boolean functions are computable

Libraries

Propositions and booleans (2)

- In decidable domains this distinctions does not make sense
- Booleans are coerced to propositions

```
Coercion is true (b: bool) := b = true
```

- Propositions and booleans are related

```
Inductive reflect (P: Prop): bool  Type :=
  | Reflect true : P  reflect P true
  | Reflect false : P  reflect P false
```

Some other libraries

- `eqtype`: type with a decidable equality
- `choice`: type with choice operator
- `fintype`: type with finite elements
- `finfun`: type of function of finite domain
- `bigops`: generic indexed big operations
- `groups`: finite groups theory
- `ssralg`: algebraic structures
- `matrix`: determinant theory and matrix decomposition

SSReflect

Efficiency

- Standard Coq library
  - 93000 lines for 7000 objects
  - Average **13** lines per object
- Extended SSReflect library
  - 14400 lines for 1980 objects
  - Average **7** lines per object

Conclusion

- Only suitable for advanced Coq users
- Very effective way of doing proofs
- Mainly used for long and non-trivial proofs
- Classical flavour more familiar with Isabelle and Hol
- Decidable types
- Relies heavy on rewriting
- Most complete formalisation of finite group theory

Demo and questions

?