Advances in the Formalization of the Odd Order Theorem

Georges Gonthier

Microsoft Research Cambridge
gonthier@microsoft.com

Abstract. We present some of the proof techniques and library designs we used to formalize a large part of the proof of the Odd Order theorem.

Keywords: Formalization of Mathematics, Group Theory, Algebra, proof library, Coq, ssreflect.

The Odd Order theorem states that all finite groups of odd order are solvable. Due to Feit and Thompson, this very important and useful result in Group Theory is also historically significant because it initiated the large collective effort that lead to the full classification of finite simple groups twenty years later. It is also one of the first proofs to be questioned by prominent mathematicians because of its sheer length (255 pages) and complexity. These qualities make the Odd Order theorem a prime example for demonstrating the applicability of computer theorem proving to graduate and research-level mathematics.

As the Feit-Thompson proof draws on an extensive set of results spanning most of undergraduate algebra and graduate finite group theory, we would have to develop a substantial library of mathematical results to cover the prerequisites. We hoped that this library, its architecture, and the techniques supporting it, would provide practical outputs. Our starting point was the combinatorics library of the four-color theorem proof, and the small-scale reflection technique and the structured proof scripting language it used — these became the ssreflect extension to Coq.

Small-scale reflection consists in using the algorithmic fragment of the proof system logic (i.e., CiC for Coq) to capture more of the operational content of a mathematical theory (its "exercises"). Although reflection could not be used directly as often for general algebra than for combinatorics, we found that in association with extended type inference (supporting type classes) reflection could be used to create generic or "overloaded" theorems and theories. We used this everywhere, to create a variety of useful components, for, e.g., summations, algebraic structures, linear spaces, groups, group morphisms, characteristic subgroups...

This groundwork was tested in 2010, as we formalized more than half of the proof (the local analysis part). The smoothness of the process validated the library design: we could formalize nearly two pages a day when we had the right prerequisites. The proof language handled gracefully two unexpected difficulties in the graduate material: large lemmas (proving over 20 assertions under more than four separate assumptions), and non-structural proof steps (casual use of induction, abduction and symmetry), neither of which appeared in more basic material.

M. Van Eekelen et al. (Eds.): ITP 2011, LNCS 6898, p. 2, 2011.