Towards the formally validated crystallographic software

S. Gražulis1,2, A. Merkys1, A. Vaitkus1, K. Petrauskas2, L. Laibinis2

Institute of Biotechnology, Life Sciences Center, Vilnius University, Sauletekio 7, LT-10257 Vilnius, Lithuania
Institute of Computer Science, Faculty of Mathematics and Informatics, Vilnius University, Didlaukio 47, LT-08303 Vilnius, Lithuania
grazulis@ibt.lt

Keywords: crystallographic algorithms, formal verification, theorem proving

Digital computers have been used in crystallography since their very first applications for the purposes of pure science [1]. Software plays a crucial role in the determination, interpretation and analysis of crystal structures. As crystallographic models grow in size and complexity, so does the software that is needed to build and interpret them. Consequently, managing software “manually”, i.e. by using currently available software development methods, becomes problematic. Unfortunate errors may creep unnoticed into the software code and yield incorrect interpretation of crystallographic data [2]. It is thus desirable to develop new approaches that would allow to catch more software errors early enough and provide additional software correctness guarantees.

We propose to use formal software verification methods to document required behaviour of crystallographic software and to prove formally, with the help of proof assistants [3] such as Isabelle/HOL [4], that the software code used for crystallographic computations is indeed compliant with the requirements. This approach is used for mission critical software such as avionics and has been recently advocated as method of improving software reliability [5].

As a proof of concept, we have applied this method [6] to prove the correctness of a well-known and widely used algorithm of space group reconstruction [7]. The algorithm was first formulated as a pseudocode, and then Isabelle/HOL theory was created to describe the required pre- and postconditions of the algorithm. It was then proven using the Isabelle proof assistant that these conditions are indeed satisfied, and executable code was derived manually from the same pseudocode [6]. Since software transformations from pseudocode to executable code are simpler than the correctness proofs themselves, such approach facilitates development of correctly working computer code.

To further automate the process of the development of reliable software, and to reduce the likelihood of human error, we are investigating the use of Ada/SPARK software development and verification system for the development of crystallographic software. The SPARK system can generate proof obligations from the annotated software code in a subset of the Ada programming language and forward them to various automatic theorem provers and proof assistants. At the same time, executable code can be generated from the same program source code using a high quality open source Ada compiler such as GNAT. Such approach could further improve reliability of crystallographic software and allow the community to build well documented, reliable and formally verified crystallographic software libraries.