Towards verified usage of the C++ Standard Template Library[*]

Gergely Dévai, Norbert Pataki

Eötvös Loránd University, Department of Programming Languages and Compilers, Budapest, Hungary
{deva, patakino}@elte.hu

Abstract. In this paper we present ongoing research on formal specification of the *C++ Standard Template Library*. Our goal is to embed the *STL* datatypes and their operations into a system that is used to produce formally verified code.
From the point of view of the *C++* programmers such a tool is a great help when one writes safety critical applications. On the other hand, integration of well-known libraries makes formal methods more useful and more attractive.
We discuss possibilities of specifying basic operations of *STL* containers and iterators. We introduce a simple solution which can already be used to implement a wide range of algorithms. We also point out the limitations of that model and touch the core ideas of a possible improvement.

References

1. Home of LaCert: http://deva.web.elte.hu/LaCert.
2. J.-R. Abrial. *The B-book: assigning programs to meanings.* Cambridge University Press, New York, NY, USA, 1996.
3. S. Antoy and D. Hamlet. Automatically checking an implementation against its formal specification, 1999.
4. M. H. Austern. *Generic Programming and the STL.* Addison-Wesley, 1999.
5. J. Barnes. *High Integrity Software: The SPARK Approach to Safety and Security.* Addison Wesley, 2003.
6. Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. Springer Verlag, 2004.
7. M. de Mol, M. van Eekelen, and R. Plasmeijer. Theorem proving for functional programmers, Sparkle: A functional theorem prover. *LNCS*, page 55, 2001.
8. G. Dévai. Programming language elements for correctness proofs. *Acta Cybernetica (accepted for publication)*, 2007.
9. G. Dévai and Z. Csörnyei. Separation logic style reasoning in a refinement based language. In *Proceedings of the 7th International Conference on Applied Informatics (to appeare)*, 2007.

[*] This work is supported by "Stiftung Aktion Österreich-Ungarn (OMAA-ÖAU 66öu2)" and "ELTE IKKK (GVOP-3.2.2-2004-07-0005/3.0)".

10. C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata. Extended static checking for Java. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation (PLDI'2002)*, volume 37, pages 234–245, June 2002.
11. D. Gregor and S. Schupp. Stllint: lifting static checking from languages to libraries. *Software – Practice & Experience*, 36(3):225–254, 2006.
12. Z. Horváth, T. Kozsik, and M. Tejfel. Extending the Sparkle core language with object abstraction. *Acta Cybernetica*, 17:419–445, 2005.
13. D. Kapur, D. R. Musser, and X. Nie. An overview of the Tecton proof system. *Theoretical Computer Science*, 133:307–339, 1994.
14. J. McDonald and J. Anton. Specware - producing software correct by construction, 2001.
15. S. Meyers. *Effective STL.* Addison-Wesley, 2001.
16. J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Sci. Comput. Program.*, 9(3):287–306, 1987.
17. D. R. Musser. Tecton description of STL container and iterator concepts, 1998.
18. D. R. Musser and C. Wang. A basis for formal specification and verification of generic algorithms in the C++ standard template library, 1995.
19. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002.
20. P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. *Lecture Notes in Computer Science*, 2142, 2001.
21. N. Pataki, Z. Porkoláb, and Z. Istenes. Towards soundness examination of the C++ standard template library. In *Electronic Computers and Informatics*, pages 186–191, 2006.
22. J. Reynolds. Separation logic: a logic for shared mutable data structures, 2002.
23. J. Siek and A. Lumsdaine. Concept checking: Binding parametric polymorphism in C++. In *Proceedings of the First Workshop on C++ Template Programming*, 2000.
24. B. Stroustrup. *The C++ Programming Language.* Addison-Wesley, special edition, 2000.
25. K. Pásztor Varga and M. Várterész. Usability of some theorem proving systems. *PU.M.A.*, 15(2-3):273–284, 2004.
26. Ch. Wang and D. R. Musser. Dynamic verification of C++ generic algorithms. *Software Engineering*, 23:314–323, 1997.
27. J. Winkler. The frege program prover FPP. *Internationales Wissenschaftliches Kolloquium*, 42:116–121, 1997.
28. H. Yang and P. O'Hearn. A semantic basis for local reasoning. In *Foundations of Software Science and Computation Structure*, pages 402–416, 2002.