# A Comparative Study of C++ Standard Template Library's Formal Specification

**Norbert Pataki and Gergely Dévai**

The C++ Standard Template Library (STL) is an essential library that is used in C++ programs. The STL is built up according to the *generic programming* paradigm. It provides many useful containers (for example `vector` or `set`, etc.), iterators, algorithms (for instance `for_each()` or `sort()`) and functors. These work together and they are designed to be highly extensible.

The generic programming results in a special layout. Algorithms are fairly irrespective of the used container, because they work with iterators. For instance, we can use the `for_each()` algorithm with all containers. As a result of this layout the complexity of the library is greatly reduced and we can extend the library with new containers and algorithms simultaneously.

However, the STL itself is defined in the C++ Standard in an informal way. It may lead to misunderstood in some special cases and it does not help when one needs to formally prove the correctness of a safety-critical application. Unfortunately, ambigous specification can be found in the Standard too.

Formal techniques are needed to overcome the previous problems. Many formal specifications have been developed for the C++ STL and the generic programming paradigm. Various techniques have been worked out from the extension of well-known Hoare's method to the debugger-engined proof execution. Software tools also have been implemented to write safer STL-based code fragments.

In this paper we give an overview about formal techniques and tools that are used in C++ STL's specification. We examine the advantages and disadvantages of the known methods.

**References**

[1] Dévai, D., Pataki, N.: A Tool for Formally Specifying the C++ Standard Template Library, 2007, Annales (to appear)

[2] Meyers, S.: *Effective STL*, Addison-Wesley, 2001.

[3] Pataki, N., Porkoláb, Z., Istenes Z.: Towards soundness examination of the C++ standard template library, In *Electronic Computers and Informatics*, pages 186–191, 2006

[4] B. Stroustrup: *The C++ Programming Language*, Addison-Wesley, special edition, 2000.