Examining test coverage of C++ template metaprograms

Tamás Cséri[a], Gábor Horváth[a], Zalán Szűgyi[a]

[a]Eötvös Loránd University
{cseri,xazax,lupin}@casear.elte.hu

Abstract

Template metaprogramming is a new technique in C++ programming in which the algorithms are executed at compile-time. These programs are usually the foundations of libraries thus their testing is an essential issue. Standard coverage tools operate at runtime and are not applicable for template metaprograms. This paper examines the possibilities to check test coverage and show missing test inputs for template metafunctions.

Unit testing is a common way to test template metaprograms. To ensure that a test suite is sufficient we must examine the test coverage. For a good test suite we expect that every template specialization is checked at least once. In this paper we check if all specializations were instantiated during the compilation of the unit test by extending the compiler to show us information about template instantiations. For the missing cases we try to generate skeleton code that helps the programmer to identify the missing test cases.

We move on by examining if all decisions are tested. We define a decision by instantiating a template that has bool template parameter with a dependent type. It is expected that during the test suite both values of the bool template are tested.

Our goal is to aid the testing of complex template metaprograms by showing the programmer the untested cases.

Keywords: C++, template metaprogramming, testing

MSC: 68N15, 68N20