

CHALMERS | GÖTEBORG UNIVERSITY

Generic Programming with Concepts

Marcin Zalewski

The defense of this thesis for the degree of Doctor of Philosophy
will be held in room **EE**, ED&IT building,
Rännvägen 6B, Chalmers University of Technology,
on **Friday, November 28, 2008**, at **13.15**.

Faculty opponent: **Prof. Gregor Snelting**, Fakultät für Informatik, Universität
Karlsruhe, Germany.

The thesis is available at the Department of Computer Science and Engineering,
Chalmers University of Technology and Göteborg University.

Department of Computer Science and Engineering
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Telephone +46 (0)31-772 1000



Abstract

Generic programming is an indispensable ingredient of modern programming languages. In the C++ programming language, generic programming is implemented by the means of parameterized fragments of code, called *templates*, in which parameters are constrained by *concepts*. This thesis consists of six publications investigating different aspects of generic programming with concepts—formal semantics of the concepts language feature, the mathematical foundations of concepts as a specification tool, change impact analysis of generic libraries, and multi-paradigm, multi-language library development.

Formally specifying the semantics of programming languages is a difficult task, one that is taken up rarely due to its complexity. In this thesis we provide a formal semantics of the *separate type checking* with concepts. We also describe some potential problems we discovered while translating the informal wording into formal rules. A formal semantics, such as ours, makes it easier to discuss language design, to improve the quality of the informal specification, and may serve as a model for compilers and other tools. Concepts as *institutions* is another view on the semantics of concepts we provide in this thesis. Institutions describe the parts of logics: *signatures* are the vocabulary, *sentences* are phrases that can be said given a vocabulary, and *models* are the subjects of the phrases. The institutions of concepts we provide make concepts into logics; the most basic general institution frames concepts as signatures and concept maps as sentences and models at the same time. The logic view on concepts makes it possible to apply techniques from the field of algebraic specification in the realm of C++ programming.

Generic programming with concepts also requires more practical support: in this thesis we describe a *conceptual change impact analysis for generic libraries*. Our change impact analysis is applied at the specification level of a generic library, which comprises the underlying concept hierarchy and generic interfaces of library functions and data structures. We apply the analysis to a real and important problem, a proposed change to the iterator concepts hierarchy of the Standard Template Library of C++.

Finally, we consider multi-paradigm development of generic libraries. We investigate how the datatype-generic paradigm applies to an imperative language such as C++: we provide a possible solution and outline the important differences between C++ and a functional language that must be taken into account. We also consider similarities between generic programming in Haskell and C++: based on a particular library we provide a translation from Haskell generic interfaces to the corresponding interfaces in C++.