Metacasanova: a High-performance Meta-compiler for Domain-specific Languages

Francesco Di Giacomo

Abstract

Programming languages are at the foundation of computer science, as they provide abstractions that allow the expression of the logic of a program independent from the underlying hardware architecture. In particular scenarios, it can be convenient to employ Domain-Specific Languages, which are capable of providing an even higher level of abstraction to solve problems which are common in specific domains. Examples of such domains are database programming, text editing, 3D graphics, and game development. The use of a domain-specific language for the development of particular classes of software may drastically increase the development speed and the maintainability of the code, in comparison with the use of a general-purpose programming language. While the idea of having a domain-specific language for a particular domain may be appealing, implementing such a language tends to come at a heavy cost: as it is common to all programming languages, domain-specific languages require a compiler which translates their programs into executable code. Implementing a compiler tends to be an expensive and time-consuming task, which may very well be a burden which overshadows the advantages of having a domain-specific language.

To ease the process of developing compilers, a special class of compilers called "meta-compilers" has been created. Meta-compilers have the advantage of requiring only the definition of a language in order to generate executable code for a program written in that language, thus skipping the arduous task of writing a hard-coded compiler for the new language. A disadvantage of meta-compilers is that they tend to generate slow executables, so they are usually only employed for rapid prototyping of a new language. The main aim of this thesis is to create a meta-compiler which does not suffer from the disadvantage of inefficiency. It presents a meta-compiler called "Metacasanova", which eases the development cost of a compiler while simultaneously generating efficient executable code.

The thesis starts by analysing the recurring patterns of implementing a compiler, to define a series of requirements for Metacasanova. It then explains the architecture of the meta-compiler and provides examples of its usage by implementing a small imperative language called C--, followed by the reimplementation of a particular, existing domain-specific

language, namely Casanova, which has been created for use in game development. The thesis presents a novel way to optimize the performance of generated code by means of functors; it demonstrates the effect of this optimization by comparing the efficiency of Casanova code generated with and without it. Finally, the thesis demonstrates the advantages of having a meta-compiler like Metacasanova, by using Metacasanova to extend the semantics of Casanova to allow the definition of multiplayer online games.

Contents