Modeling C preprocessor metaprograms using purely functional languages

Máté Karácsony

Dept. of Programming Languages and Compilers,
Fac. of Informatics, Eötvös Loránd University, Budapest
`k_mate@inf.elte.hu`

Abstract

The preprocessor of the C language provides a convenient base to define relatively complex source-to-source transformations. Describing these transformations is possible by using parametrized, so called function-like macros. In fact, the structure of these macros can be almost arbitrarily compound, limited only by the actual preprocessor implementation [1]. However, writing and understanding these macros is usually fairly difficult. The lack of typing, the statelessness, and the uncommon syntax are the main reasons of this difficulty.

To support the development of these preprocessor metaprograms, a model for each of the macros can be implemented with a function in a purely functional language. If only a well-defined, restricted set of the chosen language is used, then a working functional implementation can be translated back into a preprocessor metaprogram. This translation leads to easily recognizable patterns in the resulting program, thus makes its maintenance and understanding easier. Furthermore, testing and prototyping may need less effort using an initial implementation in a functional language. This method is similar to the Haskell to C++ template compilation described in [2].

The absence of types in the macro system is expressed by the signature of these functions: they define mappings between strings. Because they are pure, the lack of a global state is also simulated. The abstraction level is increased by using basic arithmetic, conditionals, pattern matching, tuple- and list-manipulation functions. The process of the modeling and the translation of the functional elements into preprocessor macros will be presented using the abstractions provided by the preprocessor subset of the Boost MPL library [3].

Keywords: Preprocessor metaprogramming, functional programming

MSC: 68N15 Programming languages

References

[1] International Standard: Programming languages — C, Second edition, *ISO/IEC 9899:1999*, 1999.

[2] Zoltán Porkoláb, Ábel Sinkovics: C++ Template Metaprogramming with Embedded Haskell, *Proc. 8th Int. Conf. Generative Prog. & Component Engineering (GPCE 2009)(New York, NY, USA), ACM.*, 2009.

[3] David Abrahams, Aleksey Gurtovoy: C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond, *Addison Wesley Professional*, 2004.