Twiddle – A DSL for the Functional Bit-hacker

Michael Buch

January 2, 2019

Abstract

It is useful (and fun!) to bit twiddle i.e. perform arithmetic and manipulate data at the granularity of individual bits. Traditionally there is a compromise one has to make between using a low-level unsafe language that allows bit-twiddling versus a safe high-level language in which the type system or language design prohibit operations at bit-level (without additional complexity). The *Twiddle* domain-specific language (DSL) is an embedded language written in Scala that generates bit-twiddling style C code. The language offers several modes of operation, useful for quick prototyping, debugging and custom extensions: (1) Tracing interpreter (2) Scala evaluator (3) Twiddle AST generator (4) C code generator. Thus our language is a tool for the curious, a tool for safe bit-hackers and a tool for someone looking to get a bit more performance out of his high-level language.

1 Motivation

2 The Language

2.1 Core

The core of the language is split into modular pieces of functionality implemented as traits. As is common practice with tagless final interpreters (see section 4.3) we parameterize each set of language features with an evaluator that describes how each feature within its context. The core set of features is divided into following traits:

1. Arithmatic
2. Strings
3. Bools
4. Lambda
5. LispLike
6. CLike
7. CMathOps
8. CStrOps