Shine Documentation

Release 0.1

Christian S. Perone

Sep 27, 2017

Contents

Welcome to Shine's Documentation

Shine is a library that automatically makes the code generation of the LLVM IR (Intermediate Representation) from Genetic Programming trees. It also provides an easy API for optimizing and JITing Genetic Programming ASTs into native code.

Contents:

CHAPTER 1

Introduction

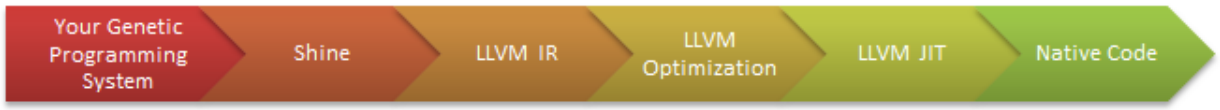
What is Shine ?

Shine is a library that automatically makes the code generation of the LLVM IR (Intermediate Representation) from the user Genetic Programming trees. It also provides an easy API for optimizing and JITing Genetic Programming ASTs into native code.

Note: Genetic Programming literature often call individuals as *trees*, *syntax trees* or *abstract syntax trees* (ASTs), we'll adopt the term AST (Abstract Syntax Tree) here and in the library itself.

Architecture Design

The follow image shows an overview of the Shine architecture:



Shine takes as input the individuals (ASTs) of your Genetic Programming system and then creates the *LLVM IR* assembly. Shine uses the LLVM Passes to optimize this generated assembly code; here is some examples showing how these transformation passes work on your Genetic Programming individuals:

Instruction Combine - *Combine redundant instructions*

It combines instructions like:

```
%Y = add i32 %X, 1
%Z = add i32 %Y, 1
```

into:

```
%Z = add i32 %X, 2
```

Constant Propagation

This transformation implements the constant propagation and merging, example:

```
add i32 1, 2
```

is transformed into:

```
i32 3
```

Note: These are just two examples of how the LLVM's Transformations works, for more information see the LLVM's Analysis and Transform Passes guide.

In the diagram below, you can see how Shine interacts with the user's Genetic Programming system:

The first step is the preparation of the non-terminal (funcions) code, actually, the definition of the functions that will be used in your ASTs can be done in any language that supports the LLVM IR generation (you can use Clang, llvm-gcc, etc...) to create the functions, in simple terms, you can define your functions using C/C++ for example.

After compiling your GP functions to LLVM IR (bitcode), you just need to plug them into the Shine library and link all of them together (if you have more than one bitcode object file). Shine library also has an optional step of LTO (*Link-time optimization*), in which you can optimize the linking between your multiple LLVM IR modules.

Now that you have your functions loaded into Shine library, all you need is to prepare an array with your AST nodes (this array is built using a **pre-order traversal** of your GP AST). After that, you will use Shine library to create, optimize and JIT the LLVM IR of your AST, to finally get a function pointer to your GP individual in native code.

See also:

LLVM's Analysis and Transform Passes

LLVM Assembly Language Reference

Requirements

In order to compile and use Shine, you'll need the follow libraries/tools:

A LLVM IR compatible compiler

This compiler will be used to create the LLVM IR of your Genetic Programming system, you can use Clang, llvm-gcc, etc.

LLVM - The Low-Level Virtual Machine

You'll need the version 2.9+ of the LLVM Compiler Infrastructure.

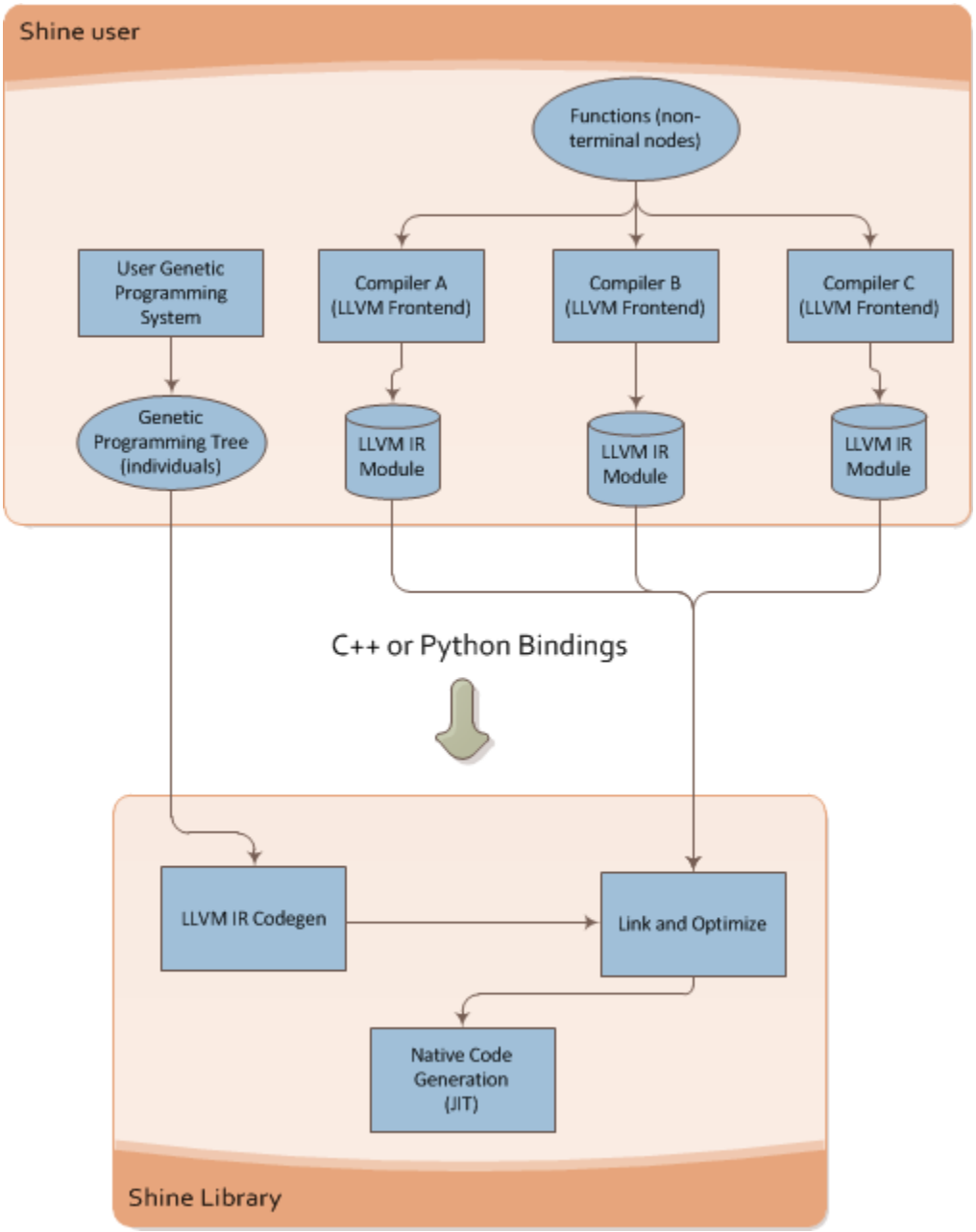
Version tested: 2.9

CMake

The CMake is used to create tha Makefiles.

Version tested: 2.8

GLib 2.x.x+ *(optional)*


Shine user
Functions (non-terminal nodes)
User Genetic Programming System
Compiler A (LLVM Frontend)
Compiler B (LLVM Frontend)
Compiler C (LLVM Frontend)
Genetic Programming Tree (individuals)
LLVM IR Module
LLVM IR Module
LLVM IR Module
C++ or Python Bindings
LLVM IR Codegen
Link and Optimize
Native Code Generation (JIT)
Shine Library

You'll only need the GLib if you'll need to compile the tests.

Version tested: 2.26.1

Sphinx *(optional)*

The Sphinx was used to create the non-API documentation of the library.

Version tested: 1.0.7

Doxygen *(optional)*

The Doxygen was used to create the API documentation of the library.

Version tested: 1.7.1

Installation

sdffd

CHAPTER 2

Getting Started

Todo

Write the doc !

CHAPTER 3

Contact the author

Christian S. Perone <christian.perone@gmail.com>

CHAPTER 4

License

Shine - The Symbolic Regression Machine

CHAPTER 5

Indices and tables

- genindex
- modindex
- search