**MCFromC Project Overview**

# Glossary

* AST – Abstract Syntax Tree – tree data structure that contains the given source code as syntax tree.
* C.F.G – Control Flow Graph
* MC – Monotonicity Constrains .
* A.C.F.G – Annotated Control Flow Graph is C.F.G that contains MCs on its edges.

# General Architecture overview

The project architecture is based on layers where each layer takes input process it and output it as input to the next layer. The Figure 1 describes shows a basic flow of the system.

Figure 1 - Syste Layers

Output

CFG

CFG

AST

C Source

Syntax Parser

CFG Creation

CFG Simplifier

MC Creation

## Syntax Parser

The syntax parser layer is responsible of converting text file containing subset of the C programming language (see Supported grammar (in BNF format) section for the exact language definition in BNF) into AST that then will be processed into C.F.G. This layer is implemented using the following tools:

* flex – This tool uses lex file and produces source file that holds function that takes input stream and returns tokens according to the lex file.
* Bison – This tool is implementation for YACC. The tool receives an yacc input file (basically EBNF that for each rule contains C code snippet) and produces C source file that contains code for parsing the given language's EBNF.

Since the above tools produce source files in C language (and not C++) this layer is implemented as separate DLL which takes a filename and returns a pointer to the root node of the AST.

## C.F.G Creation

The C.F.G Creation stage is responsible of creating a C.F.G from AST. The input for this layer is an AST and the output is C.F.G and the name of all the program's state variables. all our graphs implementation are based on 3rd party library named: boost graph library. The library is implemented in templates and contains only header files. In the following section the document will cover the algorithm for creating C.F.G from AST. This layer is implemented in separated DLL named CFGGen written in C++.

## CFG Simplifier

The C.F.G simplifier is a small layer implemented in the same DLL as the previous layer (C.F.G Creation) and is responsible to remove redundant flow points from the C.F.G . One example for this redundant information is the consequence of Expressions flow points that the current layer aggregates into a single flow point that contains all the other expression flowpoints.

## MC Creation

This MC Creation layer is responsible for trying to evaluate changes in the values of the program's state variables in each abstract transition of the program (presented as edge in the C.F.G). The input for this layer is C.F.G (simplified from the previous layer) and the names of all state variables (computed during the creation of the C.F.G). This layer uses Abstract Interpretation in order to track the changes in the values of the program's state variables, this layer will be covered in detail in MC generation section.

The following chart describes the normal flow of the application with regard to code projects that are involved.

Figure - Code Project Flow

# User Manual

## Installation

## Usage

# Algorithms

## converting AST to CFG

## MC generation

### Description of how expressions are evaluated

### Algorithm

# Supported grammar (in BNF format)

# Benchmark programs and result graphs and MCs.