Milestone #2: CFG & ILOC

Overview

For this milestone, your program must construct a control-flow graph for each function containing a linear representation (based on ILOC) for each basic block. Your program must also support printing the intermediate representation of each function to a file.

You can use an `antlr` tree-parser to construct the control-flow graph consisting of basic blocks and ILOC contained within. Though this milestone is not easily divided into parts, the following might suggest an approach to dividing the problem.

Part 1: CFG

The control-flow graph depends only on the control constructs in the language and, thus, can be created without concern for the contents of the basic blocks (the nodes of the CFG).

Construct a control-flow graph for each function. You will find it useful to have an entry node and an exit node distinct from all other nodes in the graph. You will also likely want to add labels (as appropriate) to the nodes at this point.

Part 2: ILOC

Complete the implementation of this milestone by filling each basic block with the appropriate ILOC instructions. This is done by converting each portion of the high-level language into its corresponding ILOC instructions. You may take a simplistic approach to this conversion based on the code shape discussion in lecture. You are, of course, free to use more advanced patterns to generate improved ILOC, but are not required to do so.

Command-line Options

Provide a `-dumpIL` command-line option. If this option is present, then your program will output the control-flow graph (in linear form) and its contents to a file. It is common that the name of the output file match the name of the input file, but with an extension of ".il".