1 The Relooper Algorithm

Relooper is an algorithm that turns unstructured control flow into structured control flow. Unstructured control flow can contain arbitrary branch instructions to anywhere else in the code, whereas structured control flow is bounded within higher-level structures such as if and while blocks.

The Relooper algorithm is described in the Emscripten paper [1].

1.1 Input

The Relooper algorithm takes a so-called 'soup of blocks' as input. Each block is a section of the instructions that starts at a label and ends with a branch instructions (which may be either a conditional branch followed by an unconditional branch, or just an unconditional branch).

```
label:
...
non-branch instructions
...
if ... goto x      (optional)
goto y
```

The list of instructions generated in the intermediate representation is processed into a set of these blocks. They're called a 'soup' of blocks because they're no longer stored as a single continuous list, instead each block is separate and points to the blocks to which it can branch.

To avoid overloading the term 'block', we call these input blocks 'labels'.

1.2 Output

The Relooper algorithm generates a set of structured blocks, which are nested to represent the structured control flow.

There are three types of block:

- **Simple** blocks, which contain
  - An **Internal** label
  - A **Next block**
- **Loop** blocks, which contain
  - An **Inner** block
  - A **Next** block
- **Multiple** blocks, which contain
  - Some number of **Handled** blocks
  - A **Next** block

Simple blocks are the basic building block, and just contain one of the input labels (which contains the actual code to execute), and point to the block to which execution should pass next. When this is translated into target code, a Simple block gets translated directly into the code inside the label, and the Next block is put right after it.

Loop blocks represent any kind of loop in the code. The Inner block contains any labels that can branch back to the start of the loop (along some execution path, which may be multiple labels long). The Next block contains all the rest of the labels, from which execution will never be able to get back to the start of the loop.

Multiple blocks represent any kind of conditional execution, e.g. 'if' or 'switch' statements. The Handled blocks are any blocks to which execution can directly pass when we enter this block. By use of the label variable, described below, we decide which (if any) of the Handled blocks should get executed when we enter the Multiple block. Once the selected

Handled block has finished executing, we go to the Next block. It's also possible to not execute any of the Handled blocks, for example to represent an 'if' statement without an 'else', in which case execution will go directly to the Next block.

References

[1] Alon Zakai. *Emscripten: An LLVM-to-JavaScript Compiler*. Mozilla, 2013. URL: https://raw.githubusercontent.com/emscripten-core/emscripten/main/docs/paper.pdf.