Bugpoint Redesign

Author: Diego Treviño (diegotf@google.com)
Date: 2019-06-07
Status: Draft

Introduction

As use of bugpoint has grown several areas of improvement have been identified through years of use: confusing to use, slow, it doesn't always produce high quality test cases, etc. This document proposes a new approach with a narrower focus: minimization of IR test cases.

Proposed New Design

Narrow focus: test-case reduction

The main focus will be a code reduction strategy to obtain much smaller test cases that still have the same property as the original one. This will be done via classic delta debugging and by adding some IR-specific reductions (e.g. replacing globals, removing unused instructions, etc), similar to what already exists, but with more in-depth minimization.

Granted, if the community differs on this proposal, the legacy code could still be present in the tool, but with the caveat of still being documented and designed towards delta reduction.

Command-Line Options

We are proposing to reduce the plethora of bugpoint's options to just two: an interesting-ness test and the arguments for said test, similar to other delta reduction tools such as CReduce, Delta, and Lithium; the tool should feel less cluttered, and there should also be no uncertainty about how to operate it.

The interesting-ness test that's going to be run to reduce the code is given by name:

```
--test=<test_name>
```

If a `--test` option is not given, the program exits; this option is similar to bugpoint's current `-compile-custom` option, which lets the user run a custom script.

The interesting-ness test would be defined as a script that returns 0 when the IR achieves a user-defined behaviour (e.g. failure to compile on clang) and a nonzero value when otherwise. Leaving the user the freedom to determine what is and isn't interesting to the tool, and thus, streamlining the process of reducing a test-case.

If the test accepts any arguments (excluding the input ll/bc file), they are given via the following flag:

`--test_args=<test_arguments>`

If unspecified, the test is run as given. It's worth noting that the input file would be passed as a parameter to the test, similar to `-compile-custom`'s current functionality

Implementation

The tool would behave similar to CReduce's functionality in that it would have a list of passes that try to minimize the given test-case. We should be able to modularize the tool's behavior, as well as making it easier to maintain and expand.

The first version of this redesign would try to:

- Split the code into chunks and discard those that fail the given test
- Discard functions, instructions and metadata that don't influence the interesting-ness test
- Remove unused parameters from functions
- Eliminate unvisited conditional paths
- Rename variables to more regular ones (such as "a", "b", "c", etc.)

Once these passes are implemented, more meaningful reductions (such as type reduction) would be added to the tool, to even further reduce IR.

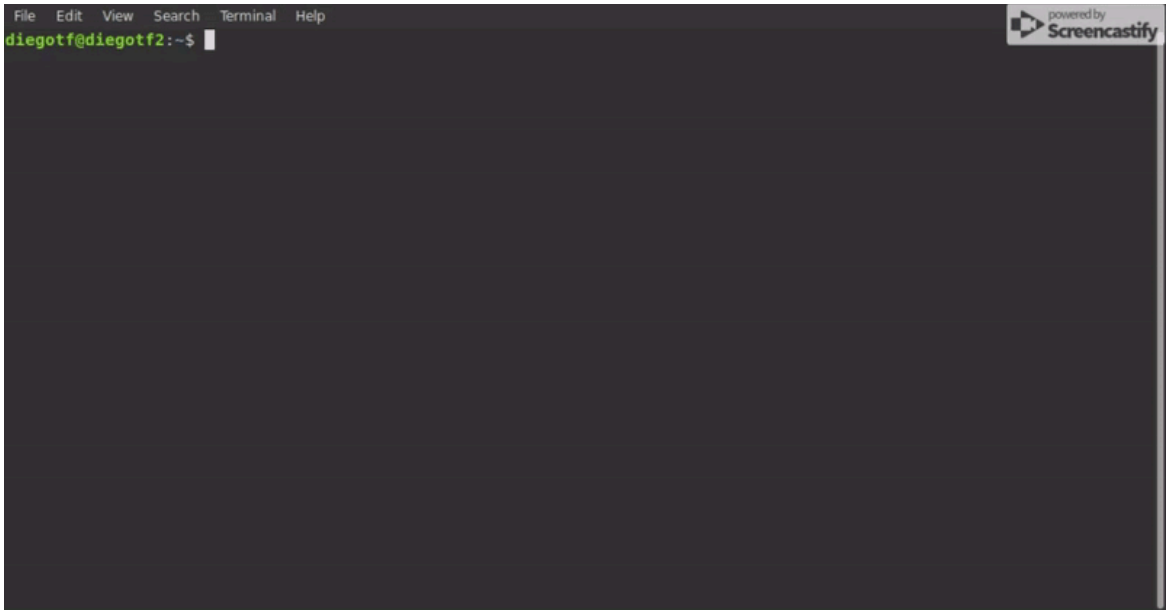
Background on historical bugpoint issues

Root Cause Analysis

Presently, bugpoint takes a long time to find the source problem in a given IR file, mainly due to the fact that it tries to debug the input by running various strategies to classify the bug, which in turn run multiple optimizer and compilation passes over the input, taking up a lot of time. Furthermore, when the IR crashes, it tries to reduce it by performing some sub-optimal passes (e.g. a lot of unreachable blocks), and sometimes even fails to minimize at all.

"Quirky" Interface

Bugpoint's current interface overwhelms and confuses the user, the help screen alone ends up confusing rather providing guidance, as seen below:



Not only are there numerous features and options, but some of them also work in unexpected ways and most of the time the user ends up using a custom script. Pruning and simplifying the interface will be worth considering in order to make the tool more useful in the general case and easier to maintain.