Detected non-deterministic results under various configurations

Open

Hi, I have recently been using Doop for an empirical study to detect non-deterministic behaviors in static analyzers. The experiments resulted in discovering some nondeterministic analysis results across multiple runs under various configurations of Doop.

The details of the experimental setup are as below:

- The experiments were conducted on the micro-benchmark CATS and a real-world benchmark DaCapo-2006.
- The experiments were conducted under 50 sample configurations which were generated using a 2-way covering array from the configuration space.
- The timeout set for Doop running on each CATS program was 60 minutes. For running on each DaCapo-2006 program, the timeout was set to 2 hours.
- We ran Doop on each program-configuration combination 5 times and compared the results across 5 runs for detecting non-deterministic behaviors.
- All experiments were conducted in docker containers. The hardware environment is a server with 376GB of RAM and 2 Intel Xeon Gold 5218 16-core CPUs@2.30GHz running Ubuntu 18.04.

In the end, the experiments detected non-deterministic results on 6 programs. None of the programs were from CATS, and all 6 programs were from the DaCapo-2006. These results were detected when the configuration sets its `analysis` option to `context-insensitive`.

The attached data is the detected nondeterministic results from CATS and DaCapo-2006 and configuration files
(*note1:* the configurations are hash-coded in the detected results, but the actual configuration options and values that each hash code stands for can be found in the attached configuration files.)

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requ

Notifications

You're receiving notificati thread.

Contributor

Hello and thanks for your work.
It's not surprising that Doop exhibits non-determinism. There are several sources of non-determinism. First, even the input facts of the analysis are non-deterministic, based on the Soot framework, whose output depends on the order of finding classes. Second, some algorithms in our reflection analysis (at least) are randomized: they elect a representative from an equivalence class of classes and process the classes based on the representatives.

The hope is that the very high-level outputs will be mostly deterministic. But if one wants strict determinism, they need to at least use a previously produced set of input facts, instead of re-deriving them. See `doop --help fact-generation`, probably the `--facts-only` flag and later the `--input-id` flag, over the previously-produced facts.

Even this may not be enough, depending on the reflection setting, but it will remove the main source of non-determinism.