Bottom-Up Synthesis of Memory Mutations with Separation Logic (Artifact)

Kasra Ferdowsi
University of California San Diego, CA, USA

Hila Peleg
Technion, Haifa, Israel

Abstract

Programming-by-Example (PBE) is the paradigm of program synthesis specified via input-output pairs. It is commonly used because examples are easy to provide and collect from the environment. A popular optimization for enumerative synthesis with examples is Observational Equivalence (OE), which groups programs into equivalence classes according to their evaluation on example inputs. Current formulations of OE, however, are severely limited by the assumption that the synthesizer's target language contains only *pure* components with no side-effects, either enforcing this in their target language, or ignoring it, leading to an incorrect enumeration. This limits their ability to use realistic component sets.

We address this limitation by borrowing from Separation Logic, which can compositionally reason about heap mutations. We reformulate PBE using a restricted Separation Logic: Concrete Heap Separation Logic (CHSL), transforming the search for programs into a proof search in CHSL. This lets us perform bottom-up enumerative synthesis without the need for expert-provided annotations or domain-specific inferences, but with three key advantages: we (i) preserve *correctness* in the presence of memory-mutating operations, (ii) *compact* the search space by representing many concrete programs as one under CHSL, and (iii) perform a provably correct OE-reduction.

We present SOBEQ (*Side-effects in OBservational EQuivalence*), a bottom-up enumerative algorithm that, given a PBE task, searches for its CHSL derivation. The SOBEQ algorithm is proved correct with no purity assumptions: we show it is guaranteed to lose no solutions. We also evaluate our implementation of SOBEQ on benchmarks from the literature and online sources, and show that it produces high-quality results quickly.

2012 ACM Subject Classification Software and its engineering → Automatic programming

Keywords and phrases Program synthesis, observational equivalence

Digital Object Identifier 10.4230/DARTS.11.2.2

Related Article Kasra Ferdowsi and Hila Peleg, "Bottom-Up Synthesis of Memory Mutations with Separation Logic", in 39th European Conference on Object-Oriented Programming (ECOOP 2025), LIPIcs, Vol. 333, pp. 10:1–10:32, 2025. `https://doi.org/10.4230/LIPIcs.ECOOP.2025.10`

Related Conference 39th European Conference on Object-Oriented Programming (ECOOP 2025), June 30–July 2, 2025, Bergen, Norway

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2025 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

The artifact is a docker image and contains the code and all dependencies for our implementation of SOBEQ, all benchmarks used in our evaluation, all parallel versions used in our evaluation, and the code for FRANGEL [1] that we use as a baseline.

In total, this replication package can be used to generate the raw data for all experiments appearing in the paper. It also includes a "kick the tires" script for a quick smoke test.

The artifact should be loaded using `docker load -i`, and all the scripts mentioned below should be run without an interactive mode, i.e., `docker run sobeq:ecoop2025 ./<script name>`.

1.1 Kick the tiers

The script `./ktt.sh` runs a kick the tires sequence: it runs each of the six main scripts in the artifact on one file in the directory `ktt-benchmark` with a timeout of 100 seconds. Even on a laptop, this should successfully terminate in under two minutes.

1.2 RQ1: Comparison to FrAngel

RQ1 is a comparison of the results of running `./run_sobeq.sh` and `./run_frangel.sh`. Each script prints intermediate results as it progresses, then calls `cat` on the more informative `results.csv` file.

Specifically, Figure 5 graphs the size columns from both result sets, and Figure 6 graphs the time column from both.

1.3 RQ2: Overhead of SObEq

RQ2 compares the classical bottom up enumeration, executed by running `./run_cbe.sh`, with SObEq on the *Pure* set. The *Pure* set is constructed by the `./run_immut.sh` script by adding an `immut` annotation to all variables in all *May* benchmarks, which are then run with SObEq.

Each script prints intermediate results as it progresses, then calls `cat` on the more informative `results.csv` file.

Figure 7 plots the time per benchmark in both outputs.

1.4 RQ3: SObEq's effectiveness in pruning

Figure 8 plots the results of `./run_sobeq.sh` (labeled "SObEq") against the results of varying OE techniques that are executed by running `./run_oe_comparison.sh`.

Each script prints intermediate results as it progresses, then calls `cat` on the more informative `results.csv` file.

One of the "Type" options is "Subsume (counters)", which is the same synthesizer behavior as `./run_sobeq.sh`, but records more information in order to plot Figure 9.

1.5 RQ4: Enumerating with preconditions vs. concrete states

RQ4 compares an enumeration with full concrete states, executed with `./run_concrete_states.sh`, with the run of "regular" SObEq (`./run_subeq.sh`). Each script prints intermediate results as it progresses, then calls `cat` on the more informative `results.csv` file.

Figure 10 plots the times for both of these runs: (a) and (b) as cumulative plots and (c) as a per-benchmark plot.

2 Content

The artifact package includes:

- `benchmarks/`: Contained the benchmarks described in the paper, divided into two folders:
  - `may/`: the *May* benchmark set in the paper
  - `must/`: the *Must* benchmark set in the paper
  - Notice: the *Pure* benchmark set does not have a directory but is constructed on the fly from *May* benchmarks in the course of the experiment for RQ2.
- `sobeq-main/`: Build directory for the "regular" version of SObEq.

- `frangel-comparison/`: Build directory for the wrapper to run FrAngel with SObEq's grammar. Also includes:
  - `lib/FrAngel`: Code for our fork of FrAngel: `https://github.com/KasraF/FrAngel/`.
- `classical-enumeration`: Build directory for our classical bottom-up enumeration (CBE) enumerator, based on the SObEq codebase, used in RQ2.
- `concrete-states`: Build directory for the concrete states enumerator, based on the SObEq codebase, used in RQ4.
- `run_*.sh`: Scripts to run the assorted experiments (detailed above)
- `ktt-benchmark/`: A directory with a single benchmark for the use of `ktt.sh`.
- `ktt.sh`: Kick the tires script.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://zenodo.org/records/15300517`.

4 Tested platforms

We tested the artifact on Windows 11 with an Intel x64 processor. Some benchmarks require large amounts of memory, we recommend at least 130GB for a full replication. There is no significant disk usage, and SObEq is single-threaded.

5 License

6 MD5 sum of the artifact

78d047d62b3cf6970b1bfa0f2442a022

7 Size of the artifact

945.4 MB

References

1 Kensen Shi, Jacob Steinhardt, and Percy Liang. FrAngel: component-based synthesis with control structures. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019. `doi:10.1145/3290386`.