# Validating SMT Solvers via Automatic Generation of Fusion Functions

Nicola Dardanis 21-960-851

Lucas Weitzendorf 21-947-130

## Motivation

SMT solvers are extensively used in formal methods, most notably in software verification, systematic test case generation, and program synthesis. Due to their high degree of complexity, even mature solvers still contain many correctness issues. Their wide applicability amplifies the severity of any such issue. In particular, soundness bugs in SMT solvers betray their users' trust and can have severe consequences in safety-critical and security-critical domains.

## Related Work

Previous research from Mansur et al. (STORM) [1] has explored the possibility of generating new satisfiable formulas by recombining sub-expressions from an initial pool of seeds. While their approach is applicable to formulas containing multiple theories, individual sub-expressions can only be combined at the Boolean operator level. Another approach that avoids differential testing is Semantic Fusion by Winterer et al. [4], which makes use of satisfiability-preserving metamorphism. Its implementation, yinyang, concatenatenates two equisatisfiable seed formulas. Next, fresh variables are introduced to connect the free variable sets from both seeds using manually designed fusion functions. Finally, some occurrences of the chosen seed variables are redefined as the inverse of the respective fusion function. Its main limitation is the necessity to manually specify fusion functions.

Other methods such as Type-Aware Operator Mutation [3] replace operators by randomly chosen ones of the same (sub)type. Generative Type-Aware Mutation [2] expands on this by replacing a sub-expression of type $T$ with the application of a randomly chosen operator returning the same type $T$ over sub-expressions of matching argument types. However, these techniques do not provide any satisfiability guarantees and therefore rely on differential testing. Despite this, operator mutation has proved to be highly effective at discovering critical bugs.

## Approach

To overcome the main limitation of Semantic Fusion, Type-Aware Operator Mutation and STORM, we propose a consolidation of ideas from all three. The goal of our approach is to automatically generate fusion functions, which can then be used by yinyang for Semantic Fusion. Critically, our approach will guarantee the existence of and find a corresponding inverse function to retrieve each input from the output of the fusion function. We achieve this in the following way.

A fusion function can be naturally represented by a tree of operators and their inputs. The internal nodes of such a tree consist of simple operators, while leaves are made up of constants and free variables, serving as inputs to the operators. The root of the tree represents the output of the fusion function. Each edge of the tree represents the result of the sub-tree rooted in the child node, thus having the type of the output of the child node operator. The tree will be subject to the following constraints:

1. Each operator node has a defined inverse operation.
2. Each free variable only appears in a single leaf.

3. Each edge is type-consistent, namely having a type matching the input parameter of the parent operator it is fed into.

From these three properties, we can guarantee that there exists a unique path from the root to each free variable. Informally speaking, the unique path property allows us to "undo" each operation in order to retrieve an expression for the variable values from the root output without introducing self-dependencies between branches. This allows us to automatically infer the inverse of the tree w.r.t. each free variable. More specifically, we inductively apply syntactic rewriting rules along the unique path from root to leaf.

Fulfilling the first tree constraint requires manual specification. For a given solver theory, we identify basic invertible operators. For example, the inverses of the addition operator (= c (+ a b)) are (= a (− c b)) and (= b (− c a)). These basic operators can then be used as operator nodes. To this end, we will encode a definition of invertible operators in a static file for each theory. We will refer to them as definition files.

The second part of our approach will consist of building a general tool, which will take the definition files as input. It is then able to automatically generate random operator trees fulfilling the last two constraints. Note that the first constraint has already been fulfilled by our choice of usable operators, which only needs to be defined once for each theory. Given the properties described above, the tool will also automatically generate an inversion function for each free variable.

The tool will be made compatible with yinyang by exporting generated functions and their inverses in SMT-LIB syntax. The tool can either be run ahead of time as stand-alone or integrated into the Semantic Fusion fuzzer of yinyang directly. This approach can be developed incrementally both in terms of the set of operators and theories supported.

# Timeline

- 15.04. Initial implementation
- 26.04. First evaluation results
- 29.04. Progress report
- 07.05. Refinement and further evaluation
- 27.05. Final evaluation results
- 06.06. Final report

# References


[1] Muhammad Numair Mansur, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Detecting critical bugs in SMT solvers using blackbox mutational fuzzing. *CoRR*, abs/2004.05934, 2020.

[2] Jiwon Park, Dominik Winterer, Chengyu Zhang, and Zhendong Su. Generative type-aware mutation for testing smt solvers. *Proc. ACM Program. Lang.*, 5(OOPSLA), oct 2021.

[3] Dominik Winterer, Chengyu Zhang, and Zhendong Su. On the unusual effectiveness of type-aware operator mutations for testing smt solvers. *Proc. ACM Program. Lang.*, 4(OOPSLA), nov 2020.

[4] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating smt solvers via semantic fusion. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, page 718–730, 2020.