AST - Project Proposal

PASCAL STREBEL and ROBIN SCHMIDIGER

1 MOTIVATION

SMT solvers are tools for determining the satisfiability of first-order logic formulas. They support a variety of different background theories such as arithmetic, bit vectors, strings, and uninterpreted functions [Monniaux 2016]. Their diverse applications, ranging from symbolic execution and test case generation to program verification, all rely on queries being answered correctly. But as SMT solvers are complex and highly optimised software systems, it is difficult to ensure their required correctness.

While most popular SMT solvers, including Z3 [de Moura and Bjørner 2008], are relatively robust in some theories such as arithmetic and bit-vectors, this is less the case for strings, partly because of the error-prone nature of string manipulations and partly because any reasonably comprehensive theory of strings is undecidable [Bjørner et al. 2009].

Hence, we will put the two string solvers available in Z3, Z3Seq [Berzish et al. 2021] and Z3Str3 [Berzish et al. 2017], under test. We will design simple satisfiability-preserving rewrite rules for formulas on strings and uninterpreted functions, and use them to generate inputs that potentially trigger soundness and performance issues in Z3.

2 RELATED WORK

While strings are among the most researched theories in the SMT community in the last decade, the methods presented are often complicated and support a rich vocabulary of operations. An example work for this is [Bugariu and Müller 2020], which, similar to what we intend to do, proposes a synthesis approach to generating formulas on strings that are (un)satisfiable by construction, with which they detected a total of 5 bugs in Z3. A more general approach is taken by Semantic Fusion [Winterer et al. 2020]. The rewriting rules presented there are very intuitive, but have nevertheless discovered 37 bugs in Z3. With both of these approaches, however, the generated formulas become increasingly complex, which can lead to timeouts and which is why generated formulas must first be reduced before reporting a bug.

[Zhang et al. 2023] focus less on soundness and more on performance regressions happening across multiple versions of a string solver. Although this will not be our main focus, the work shows performance aspects of Z3Seq and Z3Str3.

3 APPROACH

The classical dynamic programming algorithm for editing distance [Levenshtein 1965] between two strings gives us a basis for automatically generating formulas on strings and uninterpreted functions. In particular, we will leverage the problem of getting from one string to another by applying the operations insert, remove and replace for a single character repeatedly. We will model this idea in SMT and design a set of rewrite rules based on it, which generate formulas that are (un)satisfiable by construction.

We will then develop and implement a test suite that employs these rewriting rules to test the support of Z3 for the theories of strings and uninterpreted functions. Potentially, we will discover soundness and/or performance bugs.

Finally, we will draw conclusions from our analysis of the results and discuss the implications for the use of SMT solvers for strings. We will also discuss the potential for future research in this area, including the development of new sound rewrite rules or the testing of other SMT solvers.

Authors' address: Pascal Strebel, pstrebel@ethz.ch; Robin Schmidiger, schmrobi@ethz.ch.

4 WORK SCHEDULE

	Time	Task
26.03-09.04	(Week 1-2)	Formulate rewrite rules on paper.
02.04-09.04	(Week 2)	Plan software architecture for the formula generator.
09.04-13.05	(Week 3-7)	Implement the formula generator.
14.05-03.06	(Week 8-10)	Bug fixing and cleaning up code.
07.05-27.05	(Week 7-9)	Use generator to find bugs in Z3.
14.05-06.06	(Week 8-11)	Write report.
04.06-10.06	(Week 11)	Do presentation.

REFERENCES

Murphy Berzish, Vijay Ganesh, and Yunhui Zheng. 2017. Z3str3: A String Solver with Theory-aware Heuristics. In *2017 Formal Methods in Computer Aided Design (FMCAD)*. 55–59. https://doi.org/10.23919/FMCAD.2017.8102241

Murphy Berzish, Mitja Kulczynski, Federico Mora, Florin Manea, Joel D. Day, Dirk Nowotka, and Vijay Ganesh. 2021. An SMT Solver for Regular Expressions and Linear Arithmetic over String Length. arXiv:2010.07253 [cs.LO]

Nikolaj Bjørner, Nikolai Tillmann, and Andrei Voronkov. 2009. Path Feasibility Analysis for String-Manipulating Programs. In *Tools and Algorithms for the Construction and Analysis of Systems*, Stefan Kowalewski and Anna Philippou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 307–321.

Alexandra Bugariu and Peter Müller. 2020. Automatically Testing String Solvers. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 1459–1470. https://doi.org/10.1145/3377811.3380398

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 337–340.

V. I. Levenshtein. 1965. Binary codes with correction of deletions, insertions and symbol substitutions. , 845-848 pages.

David Monniaux. 2016. A Survey of Satisfiability Modulo Theory. arXiv:1606.04786 [cs.LO]

Dominik Winterer, Chengyu Zhang, and Zhendong Su. 2020. Validating SMT Solvers via Semantic Fusion. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (London, UK) *(PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 718–730. https://doi.org/10.1145/3385412.3385985

Yao Zhang, Xiaofei Xie, Yi Li, Yun Lin, Sen Chen, Yang Liu, and Xiaohong Li. 2023. Demystifying Performance Regressions in String Solvers. *IEEE Transactions on Software Engineering* 49, 3 (2023), 947–961. https://doi.org/10.1109/TSE.2022.3168373