Theory and Practice of String Solvers (Invited Talk Abstract)

Adam Kiezun
Amazon Inc.
USA
adam.kiezun@amazon.com

Philip J. Guo
University of California, San Diego
USA
philip@pgbovine.net

Pieter Hooimeijer
Facebook Inc.
USA
pieter@cs.virginia.edu

Michael D. Ernst
University of Washington, Seattle
USA
mernst@cs.washington.edu

Vijay Ganesh
University of Waterloo
Canada
vganesh@uwaterloo.ca

ABSTRACT

The paper titled "Hampi: A Solver for String Constraints" was published in the proceedings of the International Symposium on Software Testing and Analysis (ISSTA) 2009, and has been selected to receive the ISSTA 2019 Impact Paper Award. The paper describes HAMPI, one of the first practical solver aimed at solving the satisfiability problem for a theory of string (word) equations, operations over strings, and predicates over regular expressions and context-free grammars. HAMPI has been used widely to solve many software engineering and security problems, and has inspired considerable research on string solving algorithms and their applications.

In this talk, we review the state of research on the theory and practice of string solving algorithms, specifically highlighting key historical developments that have led to their widespread use. On the practical front, we discuss different kinds of algorithmic paradigms, such as word- and automata-based, that have been developed to solve string and regular expression constraints. We then focus on the many hardness results that theorists have proved for fragments of theories over strings. Finally, we conclude with open theoretical problems, practical algorithmic challenges, and future applications of string solvers.

CCS CONCEPTS

• **Software and its engineering** → Software verification and validation.

KEYWORDS

String SMT solvers, string equations, string solver-based analysis

ACM Reference Format:
Adam Kiezun, Philip J. Guo, Pieter Hooimeijer, Michael D. Ernst, and Vijay Ganesh. 2019. Theory and Practice of String Solvers (Invited Talk Abstract). In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '19), July 15–19, 2019, Beijing, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3293882.3338993

ISSTA '19, July 15–19, 2019, Beijing, China

ACM ISBN 978-1-4503-6224-5/19/07.
https://doi.org/10.1145/3293882.3338993

OVERVIEW

Over the last decade and a half, SMT solvers [2, 8, 10] have had a deep and sustained impact on software engineering, broadly construed to include testing, analysis, synthesis, verification, and software security [6, 9, 12, 20]. A particularly interesting case is that of string SMT solvers [13, 14, 23, 24], that support a rich quantifier-free first-order theory of string (word) equations, string operations such as sub-string, arithmetic over length of strings, string-to-number conversion, and predicates involving regular expressions and context-free grammars. While this theory is known to be undecidable [7], considerable progress has been made in developing practical solving algorithms for various fragments of this theory that work well for myriad applications such as dynamic symbolic analysis, testing, security analysis of web applications, automated exploit generation, and synthesis [1, 4, 5, 13, 18–21, 25]. As these solvers find new applications, the demand for evermore expressive and efficient solvers grows unabated. Today, string solver research is among the most active and exciting topics within the broader SMT context, with many leading solver teams such as those developing Z3str3 and CVC4. Further, a standardization effort has led to an official SMT-LIB theory of strings [3].

A decade ago, we published a paper in the proceedings of the International Symposium on Software Testing and Analysis (ISSTA) 2009, where we described one of the first practical string solving algorithm and its implementation called HAMPI [13]. The HAMPI string solver is a decision procedure for a quantifier-free first-order theory over strings with finite length. Specifically, HAMPI supports equations over finite-length strings, as well as common functions over strings (e.g., sub-string operation), and predicates over regular expressions and context-free grammars (e.g., contains predicate). Prior to HAMPI, there were few practical and efficient decision procedures for theories over strings, given the extraordinarily high complexity of deciding such theories in the worst-case [11, 15–17]. It became quickly apparent that the design of such solvers had to be driven by applications and *typical case* instances.

We made several careful pragmatic design choices that enabled HAMPI to scale for many software engineering and security applications. For example, we designed the HAMPI language to support a rich set of functions and predicates, while choosing to limit variables to finite-length strings. This naturally led to a SAT-based approach to solving string constraints, i.e., string formulas input to

HAMPI are reduced to Boolean formulas and a SAT solver is then used to solve them.

These and other pragmatic design choices enabled HAMPI to scale well for the analysis of large applications (e.g., security analysis of web applications), while retaining the expressive power of its rich input language. The ideas underpinning the design of HAMPI remain relevant, as some of them were used by us in the design of the Z3str3 string solver even though it supports unbounded strings.

In this talk, we review the state of both theoretical and practical research on string solving algorithms, specifically highlighting key historical developments that have led to their widespread use. On the practical front, we discuss different kinds of algorithmic paradigms, such as word-based [23, 24] and automata-based [22], that have been developed to solve string and regular expression constraints. We then focus on the many hardness results that theorists have proved for fragments of theories over strings [7, 11, 15–17]. Finally, we conclude with open theoretical problems, practical algorithmic challenges, and future applications of string solvers especially for security of web applications.

ACKNOWLEDGMENTS

We acknowledge the US National Science Foundation (NSF) and the Canadian National Science and Engineering Research Council (NSERC) for their generous support and funding for the research described here. We also acknowledge the feedback provided by users which has been crucial to improving our solvers.

REFERENCES

[1] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M.D. Ernst. 2010. Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking. *Software Engineering, IEEE Transactions on* 36, 4 (july-aug. 2010), 474 –494. https://doi.org/10.1109/TSE.2010.31

[2] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovi'c, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. CVC4. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV '11) (Lecture Notes in Computer Science)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.), Vol. 6806. Springer, 171–177. http://www.cs.stanford.edu/~barrett/pubs/BCD+11.pdf Snowbird, Utah.

[3] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. 2016. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`.

[4] Prithvi Bisht, Timothy Hinrichs, Nazari Skrupsky, Radoslaw Bobrowicz, and V. N. Venkatakrishnan. 2010. NoTamper: automatic blackbox detection of parameter tampering opportunities in web applications. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS '10)*. ACM, New York, NY, USA, 607–618. https://doi.org/10.1145/1866307.1866375

[5] Nikolaj Bjørner, Nikolai Tillmann, and Andrei Voronkov. 2009. Path Feasibility Analysis for String-Manipulating Programs. In *Proceedings of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '09)*. 307–321. https://doi.org/10.1007/978-3-642-00768-2_27

[6] C. Cadar, V. Ganesh, P.M. Pawlowski, D.L. Dill, and D.R. Engler. 2006. EXE: automatically generating inputs of death. In *ACM Conference on Computer and Communications Security*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM, 322–335.

[7] Joel Day, Vijay Ganesh, Paul He, Florin Manea, and Dirk Nowotka. 2018. The Satisfiability of Word Equations: Decidable and Undecidable Theories. In *Proceedings of the 12th International Conference on Reachability Problems*.

[8] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: an efficient SMT solver. In *Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems (TACAS'08)*. 337–340. http://dl.acm.org/citation.cfm?id=1792734.1792766

[9] M. Emmi, R. Majumdar, and K. Sen. 2007. Dynamic test input generation for database applications. In *ISSTA*. 151–162.

[10] Vijay Ganesh and David L. Dill. 2007. A Decision Procedure for Bit-vectors and Arrays. In *Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07)*. 519–531. http://dl.acm.org/citation.cfm?id=1770351.1770421

[11] Vijay Ganesh, Mia Minnes, Armando Solar-Lezama, and Martin Rinard. 2012. Word equations with length constraints: what's decidable?. In *HVC'12*.

[12] P. Godefroid, N. Klarlund, and K. Sen. 2005. DART: directed automated random testing. In *PLDI*. 213–223.

[13] Adam Kiezun, Vijay Ganesh, Philip J. Guo, Pieter Hooimeijer, and Michael D. Ernst. 2009. HAMPI: A Solver for String Constraints. In *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis (ISSTA '09)*. 105–116. https://doi.org/10.1145/1572272.1572286

[14] Tianyi Liang, Andrew Reynolds, Cesare Tinelli, Clark Barrett, and Morgan Deters. 2014. A DPLL(T) Theory Solver for a Theory of Strings and Regular Expressions. In *Proceedings of the 26th International Conference on Computer Aided Verification*. 646–662.

[15] G.S. Makanin. 1977. The problem of solvability of equations in a free semigroup. *Math. Sbornik* 103 (1977), 147–236. English transl. in Math USSR Sbornik 32 (1977).

[16] Yu. Matiyasevich. 2008. Computation Paradigms in Light of Hilbert's Tenth Problem. In *New Computational Paradigms*, S.B. Cooper, B. Löwe, and A. Sorbi (Eds.). Springer New York, 59–85.

[17] Wojciech Plandowski. 2006. An Efficient Algorithm for Solving Word Equations. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*. 467–476. https://doi.org/10.1145/1132516.1132584

[18] Prateek Saxena, Devdatta Akhawe, Steve Hanna, Feng Mao, Stephen McCamant, and Dawn Song. 2010. A Symbolic Execution Framework for JavaScript. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP '10)*. 513–528. https://doi.org/10.1109/SP.2010.38

[19] Takaaki Tateishi, Marco Pistoia, and Omer Tripp. 2013. Path- and Index-sensitive String Analysis Based on Monadic Second-order Logic. *ACM Trans. Softw. Eng. Methodol.* 22, 4, Article 33 (Oct. 2013), 33 pages. https://doi.org/10.1145/2522920.2522926

[20] G. Wassermann and Z. Su. 2007. Sound and precise analysis of web applications for injection vulnerabilities. In *PLDI*, J. Ferrante and K.S. McKinley (Eds.). ACM, 32–41.

[21] Xiaofei Xie, Yang Liu, Wei Le, Xiaohong Li, and Hongxu Chen. 2015. S-looper: Automatic Summarization for Multipath String Loops. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis (ISSTA 2015)*. ACM, New York, NY, USA, 188–198.

[22] Fang Yu, Muath Alkhalaf, and Tevfik Bultan. 2010. STRANGER: an automata-based string analysis tool for PHP. In *Proceedings of the 16th international conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*. 154–157. https://doi.org/10.1007/978-3-642-12002-2_13

[23] Yunhui Zheng, Vijay Ganesh, Sanu Subramanian, Omer Tripp, Murphy Berzish, Julian Dolby, and Xiangyu Zhang. 2016. Z3str2: an efficient solver for strings, regular expressions, and length constraints. *Formal Methods in System Design* (2016), 1–40. https://doi.org/10.1007/s10703-016-0263-6

[24] Yunhui Zheng, Vijay Ganesh, Sanu Subramanian, Omer Tripp, Julian Dolby, and Xiangyu Zhang. 2015. *Effective Search-Space Pruning for Solvers of String Equations, Regular Expressions and Length Constraints*. Springer International Publishing, Cham, 235–254. https://doi.org/10.1007/978-3-319-21690-4_14

[25] Yunhui Zheng and Xiangyu Zhang. 2013. Path Sensitive Static Analysis of Web Applications for Remote Code Execution Vulnerability Detection. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. 652–661. http://dl.acm.org/citation.cfm?id=2486788.2486874