Z3-Noodler 1.2.0
System description for SMT-COMP 2024

Yu-Fang Chen[2], David Chocholatý[1], Vojtěch Havlena[1],
Lukáš Holík[1], Ondřej Lengál[1], Juraj Síč[1]

[1] Brno University of Technology, Brno
[2]Academia Sinica, Taipei

Abstract

This is a brief overview of the string solver Z3-Noodler[1] 1.2.0 entering SMT-COMP 2024. It is based on the SMT solver Z3 [1] in which it replaces the string theory solver. Z3-Noodler implements multiple decision procedures for efficient handling of different fragments of string constraints with the stabilization-based procedure as the main one [2, 3]. Z3-Noodler heavily relies on nondeterministic finite automata (NFAs) whose handling is provided by the automata library Mata [4].

1 Overview

Z3-Noodler[1] is a string solver that targets string constraints such as those that occur at analysis of programs, regular filters, policy descriptions, etc. It is built on top of the SMT-solver Z3 [1] and automata library Mata[2]. The core of the string solver is the *stabilization-based* decision procedure from [2, 3] aiming at solving string (dis)equations with length and regular constraints.

Z3-Noodler replaces the string theory solver of Z3 with a new theory solver that uses infrastructure of Z3, in particular string theory rewriter for lightweight simplification of input formula and linear arithmetic solver (LIA) for solving length constraints. Z3-Noodler's string theory solver then interacts with the core of Z3 in an usual manner: based on an input conjunction of string constraints the solver reduces the conjunction to a LIA formula over string lengths and returns it to Z3 as a theory lemma. To speed-up the obtaining of a satisfiable case, the solver generates parts of the LIA formula lazily, when necessary. Z3-Noodler implements multiple decision procedures, which are using Mata library for handling of NFAs. Z3-Noodler is implemented in C++, as well as Z3 and Mata. For a more detailed description of Z3-Noodler's design, we refer to [5], which is the paper that this description is based on.

2 Core of the String Solver

From a high-level point of view the string solver of Z3-Noodler consists of several parts: (i) interface interacting with Z3's core, (ii) axiom saturation for string functions/predicates, (iii) preprocessing of string constrants, and (iv) decision procedures for satisfiability checking of a conjunction of string atoms. Below, we give a brief description of some important features of the string solver.

Axiom Saturation. In order to maximize the benefits of internal Z3's LIA solver during generation of satisfiable assignment, we saturate the input formula with length-aware axioms and specific axioms for string predicates/functions. For instance $|t_1.t_2| = |t_1| + |t_2|$ where t_1, t_2 are string terms. We also use different saturation rules for functions/predicates instantiated by concrete values, e.g., the axiom $substr(t_1, 4, 1) = at(4)$.

Preprocessing. The core string solver uses a set of simple rewriting rules that eliminate trivial equations such as $x = y$, simplify equations when a string variables is known to equal the empty string, transform equations to regular membership constraint when possible ($x = uv$ becomes $x \in L_u \cdot L_v$ if u, v do not appear elsewhere and are constraint by the languages L_u and L_v, respectively), and simplify equations such as $xyz = xuz$ into $y = u$. Besides the semantics preserving rewriting rules, we use one under-approximating rule that replaces a membership of a variable in a co-finite language by a length constraint that excludes all the lengths of words outside the language.

Decision procedures. From a high-level point of view, Z3-Noodler implements multiple complementary decision procedures. A particular decision procedure is selected according to features of input string constraint. The main decision procedure is *stabilization-based* procedure [2, 3] for efficient solving of string (dis)equations with length and regular constraints. The procedure interatively refines language of each variable (represented by a NFA) according to word equations until the stabilization condi-

[1] `https://github.com/VeriFIT/z3-noodler`
[2] `https://github.com/VeriFIT/mata`

tion is met in a way that concatenation of languages corresponding to left and right side of each equation is equal. When the stability is achieved, length constraints constructed from variable languages are passed to the LIA solver. The language refinement is done through *noodlification*, which is an automata operation for generating feasible variable alignments. The algorithm is complete for the *chain-free* [6] combinations of equations, lenght and regular constraints, together with unrestricted disequations making it as a larges known decidable fragment of these types of constraints.

For efficient handling of *quadratic* equations (at most two occurrences of each variable) with length constraints, Z3-Noodler implements Nielsen transformation [7]. The algorithm constructs a graph corresponding to the system and reasons about it to determine if the input formula is satisfiable or not [8, 9]. If the system contains length variables, we also create a counter automaton corresponding to the Nielsen graph (in a similar way as in [10]). In the subsequent step, we contract edges, saturating the set of self-loops and, finally, we iteratively generate flat counter sub-automata (a flat counter automaton only allows cycles that are self-loops), which are later transformed into LIA formulae describing lengths of all possible solutions. We also employ optimizations, such as state-space pruning, prioritization of promising paths, etc. In Z3-Noodler we use this procedure also for general cyclic equations (without guarantee of termination).

For efficient handling of difficult regular constraints, Z3-Noodler implements construction of NFAs from regular expressions with eager minimization/simulation-based reduction. To avoid useless constructions, we sort regular expression according to expected size of the corresponding NFA and eagerly combine obtained NFAs to get the answer as soon as possible.

Supported string predicates. From the SMT-lib string language, Z3-Noodler handles the basic string constraints, word equations, regular constraints, linear arithmetic (LIA) constraints on string lengths, and extended string predicates such as `str.replace`, `str.substring`, `str.at`, `str.indexof`, `str.prefixof`, `str.suffixof`, `str.contains`, `str.replace_re`, and string conversions `str.from_int`, `str.to_int`, `str.from_code`, `str.to_code`. These extended predicates are to the basic string constraints. The core solver does not support the `str.replace_all` predicate. The decision procedures used in Z3-Noodler make it complete for the chain-free fragment with unbounded disequations and regular constraints [3], and quadratic equations. Outside this fragment, our theory core is sound but incomplete.

References

[1] Leonardo Mendonça de Moura and Nikolaj Bjørner. "Z3: An Efficient SMT Solver". In: *TACAS'08*. Vol. 4963. LNCS. Springer, 2008, pp. 337–340. URL: https://doi.org/10.1007/978-3-540-78800-3%5C_24.

[2] František Blahoudek et al. "Word Equations in Synergy with Regular Constraints". In: *Formal Methods*. Ed. by Marsha Chechik, Joost-Pieter Katoen, and Martin Leucker. Cham: Springer International Publishing, 2023, pp. 403–423. ISBN: 978-3-031-27481-7.

[3] Yu-Fang Chen et al. "Solving String Constraints with Lengths by Stabilization". In: *Proc. ACM Program. Lang.* 7.OOPSLA2 (Oct. 2023). DOI: 10.1145/3622872.

[4] David Chocholatý et al. "Mata: A Fast and Simple Finite Automata Library". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Cham: Springer Nature Switzerland, 2024, pp. 130–151.

[5] Yu-Fang Chen et al. "Z3-Noodler: An Automata-based String Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Cham: Springer Nature Switzerland, 2024, pp. 24–33. ISBN: 978-3-031-57246-3.

[6] Parosh Aziz Abdulla et al. "Chain-Free String Constraints". In: *Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings*. Ed. by Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza. Vol. 11781. Lecture Notes in Computer Science. Springer, 2019, pp. 277–293. DOI: 10.1007/978-3-030-31784-3_16. URL: https://doi.org/10.1007/978-3-030-31784-3%5C_16.

[7] Jakob Nielsen. "Die Isomorphismen der allgemeinen, unendlichen Gruppe mit zwei Erzeugenden". In: *Mathematische Annalen* 78.1 (1917), pp. 385–397.

[8] John Michael Robson and Volker Diekert. "On quadratic word equations". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 1999, pp. 217–226.

[9] Yu-Fang Chen et al. "A symbolic algorithm for the case-split rule in solving word constraints with extensions". In: *Journal of Systems and Software* 201 (2023), p. 111673. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2023.111673. URL: https://www.sciencedirect.com/science/article/pii/S0164121223000687.

[10] Anthony W. Lin and Rupak Majumdar. "Quadratic Word Equations with Length Constraints, Counter Systems, and Presburger Arithmetic with Divisibility". In: *Automated Technology for Verification and Analysis*. Cham: Springer International Publishing, 2018, pp. 352–369. ISBN: 978-3-030-01090-4.