Is solver guidance redundant for strong SMT implementations?

An exploration of Z3's strings.

Odysseas Machairas
o.machairas@student.tudelft.nl
Supervised by D. Sprokholt and S. Chakraborty, TU Delft

1. Context and motivation

Z3 [1] is an SMT [2] solver, which finds satisfiable assignments to queries such as, for two numbers X and Y, $X + Y > 13$ and $X \cdot Y < 10$. This is a generalization of SAT, the archetypal NP-complete problem, which means that universally efficient solutions likely don't exist. Despite this, Z3 aims to quickly find answers to such questions.

There are two ways to improve performance:

Domain-specific guidance

Understand the structure of a problem, change the strategy of the solver (aka *tactics*, in Z3) or add constraints that refine the search space.

General purpose improvements

Make the solver better for most problems.

Given these two approaches, we ask whether *domain-specific* guidance becomes less useful if the underlying implementation is stronger in general (e.g., like trying to "help" a chess engine, which is futile). Namely, we compare:

Z3str3 (2017) [3]

- Official Z3 upstream solver
- Searches smaller subtrees first
- Relatively simple/intuitive

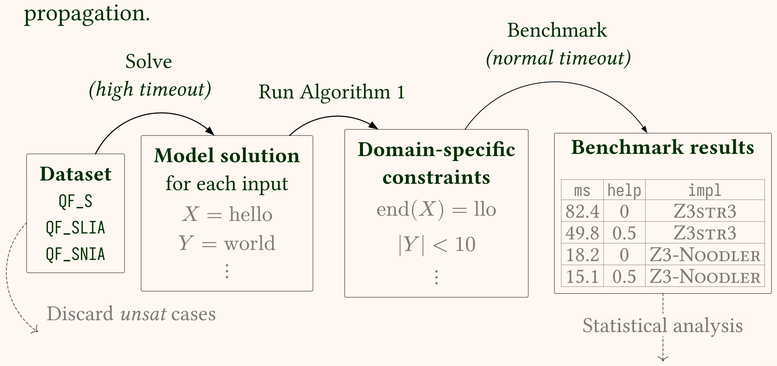
Z3-Noodler (2024) [4]

- State of the art, winner of SMT-COMP 2024
- Compares NFAs from regular expressions directly, and many other improvements
- Complicated and not intuitive

- **Practical use-cases**:
  - Whether invest time in problem understanding *vs* just letting the solver run (or improving it).
  - Both for research & industry.
  - General understanding of how the solvers behave and change.

2. Methodology

We ran an experiment on the SMT-LIB2 dataset for strings. Namely, we simulated domain-specific knowledge by adding constraints based on the solutions, which quickly cuts off infeasiable branches in regular constraint propagation.



2.1. Simulating domain-specific knowledge

- There are many constraints (Length greater than, Length less than, Length equals, Prefix (starts with), Suffix (ends with), Substring (contains))
- Question: How to give help fairly?
- Answer: Quantify the help, as the reduction of the search space
  - Guesser G chooses a length ℓ from $\mathrm{Exp}(\lambda)$, then a random string of length ℓ.
  - Probability p_s of guessing solution string s
  - Probability p_s^* of guessing solution given a constraint $*$
  - **Help = log-increase of probability**

$$h = -\frac{\ln(p_s^*) - \ln(p_s)}{\ln(p_s)}$$

- Sensible results in practice: `X = "hello"`, then help of `X startsWith "he"`
  - Expected $\sim \frac{2}{5} = 40\%$
  - Actual value: 38.87%

Environment reproducibility Note 1

Thanks to Nix, you can easily reproduce the testing environment with 100% accuracy.

Machine configuration Note 2

Programs were ran on an laptop with a M1 Max CPU with 8GB of RAM with no user programs running. Each combination was benchmarked multiple times.

3. Results

We find that Z3str3 is sped up more and more consistently than Z3-Noodler, as per Table 1.

	Mean speedup (weighted)	Mean speedup (unweighted)
Z3str3	$\mathbf{3.392} \pm 2.43$	$\mathbf{0.879} \pm 2.43$
Z3-Noodler	$\mathbf{0.282} \pm 4.64$	$\mathbf{0.294} \pm 4.64$

Table 1: Mean speedups of average runtime with vs without help, weighted by the original runtime (without help) and equally.

- Observation: "Reducing the search space" improves as the problems get harder (Figure 2)
  - Adds overhead to small, fast cases (the "slowdown zone").
- Z3-Noodler sees *huge* (70%!) slowdowns with domain-specific help, because...

1. ...it is already fast, so it doesn't leave the "slowdown zone" (Figure 2)
2. ...the additional constraints actively harm performance (Figure 3)

- Overall, **there does seem to be diminishing returns to domain-specific guidance as solvers get stronger**

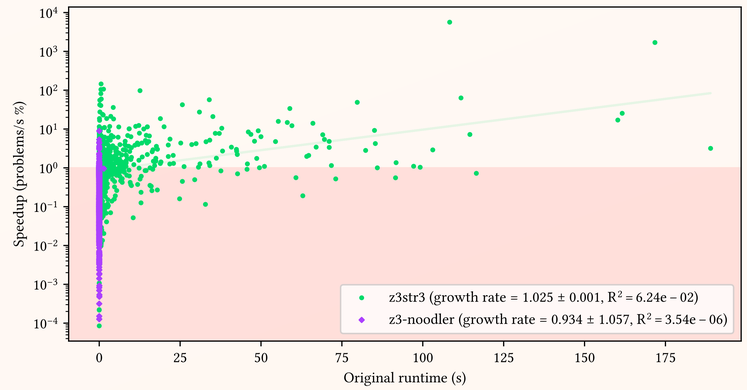


Figure 2: Speedup vs original time for Z3str3 and Z3-Noodler. Shaded area indicates slowdown. (density at the start not accurately represented)

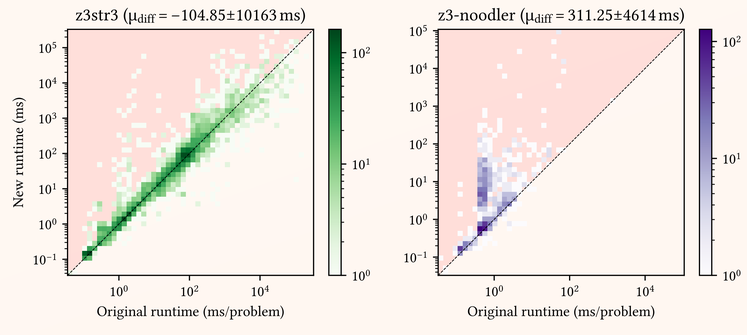


Figure 3: Heatmap comparison of runtime with and without help. Shaded area indicates slowdowns, diagonal line corresponds to no change in performance. μ_{diff} is the mean of the difference of the runtimes.

3.1. Limitations & future recommendations

- More implementations, more theories!
- Higher runtimes for Z3-Noodler (tricky because long runtimes for Z3-Noodler generally imply *very* long runtimes for Z3str3).
- Understand how exactly do these constraints harm performance.
- Add support for "soft constraints".
- Apply procedure to unsatisfiable cases.

Bibliography

[1] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 4963. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 337–340, 2008. doi: 10.1007/978-3-540-78800-3_24.

[2] C. Barrett, A. Stump, C. Tinelli, and others, "The Smt-Lib Standard: Version 2.0," in *Proceedings of the 8th International Workshop on Satisfiability modulo Theories (Edinburgh, UK)*, 2010, p. 14.

[3] M. Berzish, V. Ganesh, and Y. Zheng, "Z3str3: A String Solver with Theory-aware Heuristics," in *2017 Formal Methods in Computer Aided Design (FMCAD)*, Vienna: IEEE, Oct. 2017, pp. 55–59. doi: 10.23919/FMCAD.2017.8102241.

[4] Y.-F. Chen, D. Chocholatý, V. Havlena, L. Holík, O. Lengál, and J. Síč, "Z3-Noodler: An Automata-based String Solver," *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 14570. Springer Nature Switzerland, Cham, pp. 24–33, 2024. doi: 10.1007/978-3-031-57246-3_2.