

Z3-Noodler - An Automata based String Solver

Theory and Practice of SMT Solving

Luís Felipe Ramos Ferreira

What is Z3-Noodler

- ▶ Fork of the Z3 theorem prover

What is Z3-Noodler

- ▶ Fork of the Z3 theorem prover
- ▶ String theory solver replaced by *Noodler*

What is Z3-Noodler

- ▶ Fork of the Z3 theorem prover
- ▶ String theory solver replaced by *Noodler*
 - ▶ *Stabilization-based procedure*

What is Z3-Noodler

- ▶ Fork of the Z3 theorem prover
- ▶ String theory solver replaced by *Noodler*
 - ▶ *Stabilization-based procedure*
 - ▶ Heavy usage of *non deterministic automata*

Handling automata and regular expressions

- ▶ Hard task

Handling automata and regular expressions

- ▶ Hard task
- ▶ Mata library

Handling automata and regular expressions

- ▶ Hard task
- ▶ Mata library
 - ▶ Union, concatenation, intersection, etc

Handling automata and regular expressions

- ▶ Hard task
- ▶ Mata library
 - ▶ Union, concatenation, intersection, etc
- ▶ Very efficient and optimized

Architecture

- ▶ Takes advantage of Z3's whole DPLL(\mathcal{T}) architecture

Architecture

- ▶ Takes advantage of Z3's whole $\text{DPLL}(\mathcal{T})$ architecture
- ▶ Parser, rewriter and LIA solver

Architecture

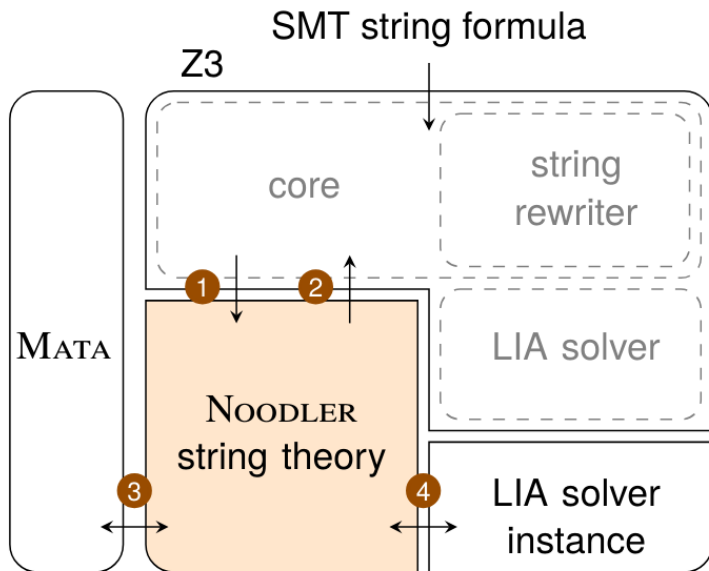


Figure 1: Z3-Noodler architecture

String Theory Core

- ▶ Axiom Saturation

String Theory Core

- ▶ Axiom Saturation
- ▶ Decision Procedures

String Theory Core

- ▶ Axiom Saturation
- ▶ Decision Procedures
- ▶ Preprocessing

String Theory Core

- ▶ Axiom Saturation
- ▶ Decision Procedures
- ▶ Preprocessing
- ▶ Supported String Predicates and Limitations

Axiom Saturation

- ▶ Goal: optimize the use of Z3's internal LIA solver

Axiom Saturation

- ▶ Goal: optimize the use of Z3's internal LIA solver
- ▶ Happens during preprocessing (before SAT solver first assignment)

Axiom Saturation

- ▶ Goal: optimize the use of Z3's internal LIA solver
- ▶ Happens during preprocessing (before SAT solver first assignment)
- ▶ $len(t_1) \geq 0, len(t_1.t_2) = len(t_1) + len(t_2)$

Axiom Saturation

- ▶ Goal: optimize the use of Z3's internal LIA solver
- ▶ Happens during preprocessing (before SAT solver first assignment)
- ▶ $len(t_1) \geq 0, len(t_1.t_2) = len(t_1) + len(t_2)$
- ▶ $t_1 = t_2 \implies len(t_1) = len(t_2)$

Axiom saturation

- ▶ Goal: deal with string functions/predicates

Axiom saturation

- ▶ Goal: deal with string functions/predicates
- ▶ Saturates the original formula with an equivalent one composed of:

Axiom saturation

- ▶ Goal: deal with string functions/predicates
- ▶ Saturates the original formula with an equivalent one composed of:
 - ▶ Word (dis)equations

Axiom saturation

- ▶ Goal: deal with string functions/predicates
- ▶ Saturates the original formula with an equivalent one composed of:
 - ▶ Word (dis)equations
 - ▶ Regular constraints

Axiom saturation

- ▶ Goal: deal with string functions/predicates
- ▶ Saturates the original formula with an equivalent one composed of:
 - ▶ Word (dis)equations
 - ▶ Regular constraints
- ▶ $\neg \text{contains}(s, "abc")$ becomes $s \notin \Sigma^* abc \Sigma^*$

Decision Procedures: Stabilization Based

- ▶ Solves word (dis)equations with regular and length constraints

Decision Procedures: Stabilization Based

- ▶ Solves word (dis)equations with regular and length constraints
- ▶ For every variable, creates a NFA encoding regular constraints of the variable

Decision Procedures: Stabilization Based

- ▶ Solves word (dis)equations with regular and length constraints
- ▶ For every variable, creates a NFA encoding regular constraints of the variable
- ▶ Iteratively refines the NFA according to word equations until stability is achieved

Decision Procedures: Stabilization Based

- ▶ Solves word (dis)equations with regular and length constraints
- ▶ For every variable, creates a NFA encoding regular constraints of the variable
- ▶ Iteratively refines the NFA according to word equations until stability is achieved
- ▶ Stability: for every word equation $s_1 = s_2$, the language of the NFA of s_1 equals the language of the NFA of s_2

Decision Procedures: Stabilization Based

- ▶ Solves word (dis)equations with regular and length constraints
- ▶ For every variable, creates a NFA encoding regular constraints of the variable
- ▶ Iteratively refines the NFA according to word equations until stability is achieved
- ▶ Stability: for every word equation $s_1 = s_2$, the language of the NFA of s_1 equals the language of the NFA of s_2
- ▶ After stability, length constraints are added and fed to the LIA solver

Decision Procedures: Nielsen Transformation

- ▶ Efficient for quadratic equations

Decision Procedures: Nielsen Transformation

- ▶ Efficient for quadratic equations
- ▶ Creates a graph encoding the system and reason about it

Decision Procedures: Nielsen Transformation

- ▶ Efficient for quadratic equations
- ▶ Creates a graph encoding the system and reason about it
- ▶ Complex

Preprocessing

- ▶ Each DP aforementioned employs different preprocessing steps

Preprocessing

- ▶ Each DP aforementioned employs different preprocessing steps
- ▶ Simple unsatisfiable patterns are checked for early termination

Preprocessing

- ▶ Each DP aforementioned employs different preprocessing steps
- ▶ Simple unsatisfiable patterns are checked for early termination
- ▶ $xy = zw \wedge \text{len}(x) = \text{len}(z) \implies y = w$

Supported String Predicates and Limitations

- ▶ *replace, substr, at, indexof, prefix, suffix, contains*

Supported String Predicates and Limitations

- ▶ *replace, substr, at, indexof, prefix, suffix, contains*
- ▶ Limited support for \neg *contains*

Supported String Predicates and Limitations

- ▶ *replace, substr, at, indexof, prefix, suffix, contains*
- ▶ Limited support for \neg *contains*
- ▶ Don't support *replace_all* and *to/from_int*

Supported String Predicates and Limitations

- ▶ *replace, substr, at, indexof, prefix, suffix, contains*
- ▶ Limited support for \neg *contains*
- ▶ Don't support *replace_all* and *to/from_int*
- ▶ Z3-Noodler is complete for the chain-free fragment with unbounded disequations, regular constraints and quadratic equations

Supported String Predicates and Limitations

- ▶ *replace, substr, at, indexof, prefix, suffix, contains*
- ▶ Limited support for \neg *contains*
- ▶ Don't support *replace_all* and *to/from_int*
- ▶ Z3-Noodler is complete for the chain-free fragment with unbounded disequations, regular constraints and quadratic equations
- ▶ Outside of this, the theory is sound but not complete

Experiments

- ▶ Three sets of benchmarks: *Regex*, *Equations* and *Predicates Small*

Experiments

- ▶ Three sets of benchmarks: *Regex*, *Equations* and *Predicates Small*
- ▶ Z3-Noodler is good in the *Regex* group in number of solved instances as well in average runtime

Experiments

- ▶ Three sets of benchmarks: *Regex*, *Equations* and *Predicates Small*
- ▶ Z3-Noodler is good in the *Regex* group in number of solved instances as well in average runtime
- ▶ The same happens in the *Equations* group

Experiments

- ▶ Three sets of benchmarks: *Regex*, *Equations* and *Predicates Small*
- ▶ Z3-Noodler is good in the *Regex* group in number of solved instances as well in average runtime
- ▶ The same happens in the *Equations* group
- ▶ cvc5 is the best one in *Predicates Small*, while Z3-Noodler performed poorly

Experiments

- ▶ Three sets of benchmarks: *Regex*, *Equations* and *Predicates Small*
- ▶ Z3-Noodler is good in the *Regex* group in number of solved instances as well in average runtime
- ▶ The same happens in the *Equations* group
- ▶ cvc5 is the best one in *Predicates Small*, while Z3-Noodler performed poorly
- ▶ Z3-Noodler could, however, be further improved by proper axiom saturation for predicates or lazy predicate evaluation

The end

► Thanks!