Solving String Constraints with Lengths by Stabilization

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

[1]Faculty of Information Technology, Brno University of Technology, Czech Republic
[2]Institute of Information Science, Academia Sinica, Taiwan

OOPSLA'23

Motivation: SMT and string constraint solving

- SMT solvers are in demand in the industry
- Reasoning about strings is important:
  - Analysis of string manipulations in programs
  - Prevention of attacks such as XSS, SQL-injection, . . . (scripting languages rely heavily on strings)
  - Regular-expression-based filters
  - AWS cloud access control policy analysis
    (Neha Rungta. A Billion SMT Queries a Day. CAV'22)
- Number of powerful SMT string constraint solvers.
  Z3, cvc5, OSTRICH, Z3-Trau, Z3str3RE, Z3str4, . . .

Motivation: SMT and string constraint solving

- SMT solvers are in demand in the industry
- Reasoning about strings is important:
  - Analysis of string manipulations in programs
  - Prevention of attacks such as XSS, SQL-injection, . . . (scripting languages rely heavily on strings)
  - Regular-expression-based filters
  - AWS cloud access control policy analysis
    (Neha Rungta. A Billion SMT Queries a Day. CAV'22)
- Number of powerful SMT string constraint solvers.
  Z3, cvc5, OSTRICH, Z3-Trau, Z3str3RE, Z3str4, . . .

$$\underbrace{x = yz \ \wedge \ y \neq u}_{\text{(dis)equations}} \ \wedge \ \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \ \wedge \ \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \ \wedge \ \underbrace{\text{contains}(u, \text{substr}(s, i, n))}_{\text{more complex operations}}$$

Our Approach

- Combination of two automata-based algorithms:

Our Approach

- Combination of two automata-based algorithms:
  1. Stabilization decision procedure (FM'23).
     - Tightly integrating equations with regular constraints
     - Fast, but does not handle length constraints

- Supports:
  - Our earlier versions

$$\underbrace{x = yz \wedge y \neq u}_{\text{(dis)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \wedge \underbrace{\text{contains}(u, \text{substr}(s, i, n))}_{\text{more complex operations}}$$

Our Approach

- Combination of two automata-based algorithms:
  1. Stabilization decision procedure (FM'23).
     - Tightly integrating equations with regular constraints
     - Fast, but does not handle length constraints
  2. Align&Split (CAV'14)
     - Slower, but handles length constraints
- Supports:
  - Our earlier versions

$$\underbrace{x = yz \wedge y \neq u}_{\text{(dis)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \wedge \underbrace{\text{contains}(u, \text{substr}(s, i, n))}_{\text{more complex operations}}$$

  - Now

$$\underbrace{x = yz \wedge y \neq u}_{\text{(dis)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \wedge \underbrace{\text{contains}(u, \text{substr}(s, i, n))}_{\text{more complex operations}}$$

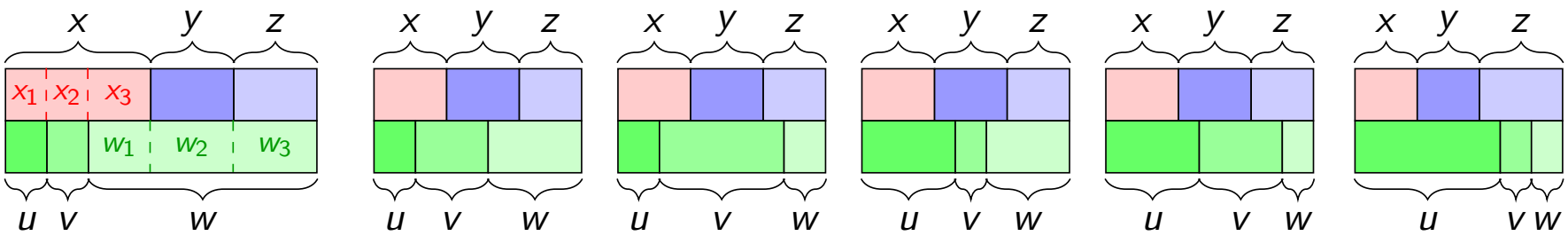
Align&Split

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Align&Split

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Equation alignments (6)



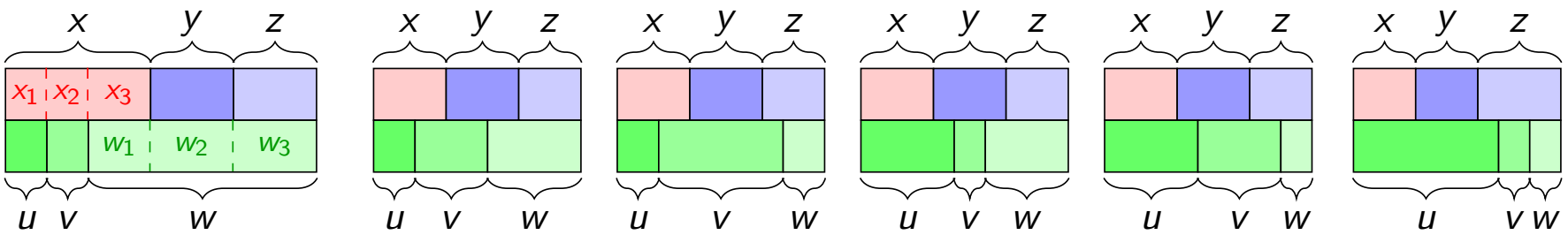
u/x_1
v/x_2
w/x_3yz
$x/x_1x_2x_3$

$x_1x_2x_3 \in \mathcal{R}_x$

Align&Split

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Equation alignments (6)



u/x_1
v/x_2
w/x_3yz
$x/x_1x_2x_3$

Automata splitting

Pure regular constraints formula:
$x_1 \in \mathcal{R}_{x_1} \wedge x_2 \in \mathcal{R}_{x_2} \wedge x_3 \in \mathcal{R}_{x_3}$

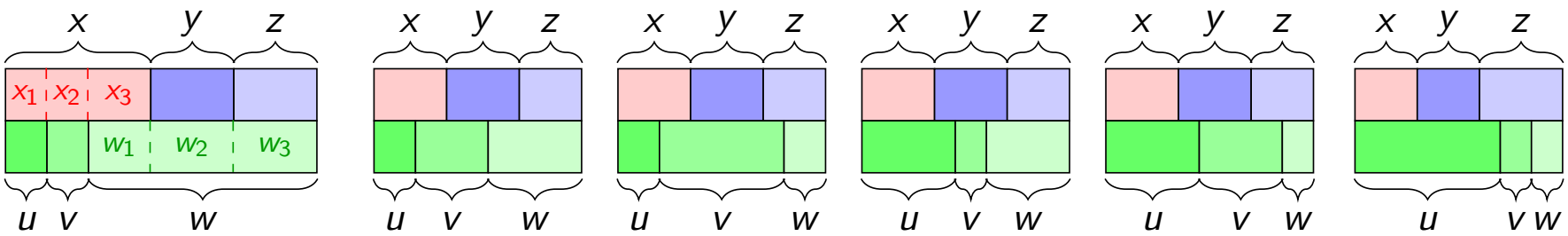
(Monadic decomposition)

$x_1x_2x_3 \in \mathcal{R}_x$

Align&Split

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Equation alignments (6)



u/x_1
v/x_2
w/x_3yz
$x/x_1x_2x_3$

Automata splitting

Pure regular constraints formula:
$x_1 \in \mathcal{R}_{x_1} \wedge x_2 \in \mathcal{R}_{x_2} \wedge x_3 \in \mathcal{R}_{x_3}$

(Monadic decomposition)

$x_1x_2x_3 \in \mathcal{R}_x$

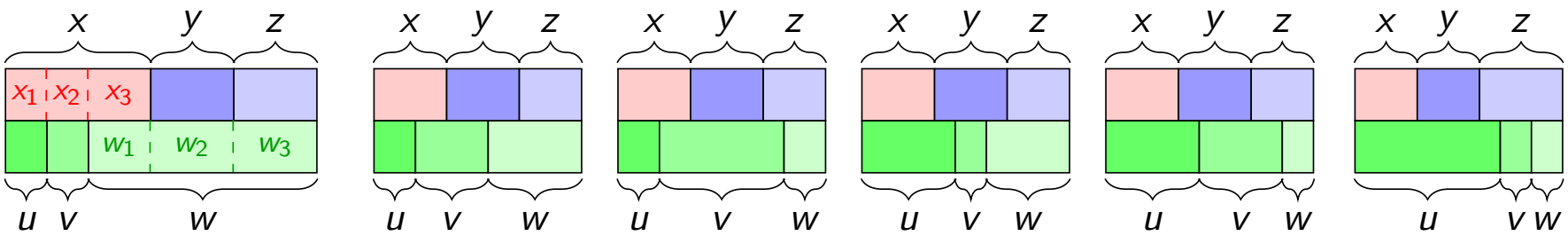
Length extraction

$\varphi[|x_1|, |x_2|, |x_3|, |y|, |z|] \wedge |x| = 100 + 4k$

Align&Split

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Equation alignments (6)



u/x_1
v/x_2
w/x_3yz
$x/x_1x_2x_3$

Automata splitting

Pure regular constraints formula:
$x_1 \in \mathcal{R}_{x_1} \wedge x_2 \in \mathcal{R}_{x_2} \wedge x_3 \in \mathcal{R}_{x_3}$

(Monadic decomposition)

$x_1x_2x_3 \in \mathcal{R}_x$

Length extraction

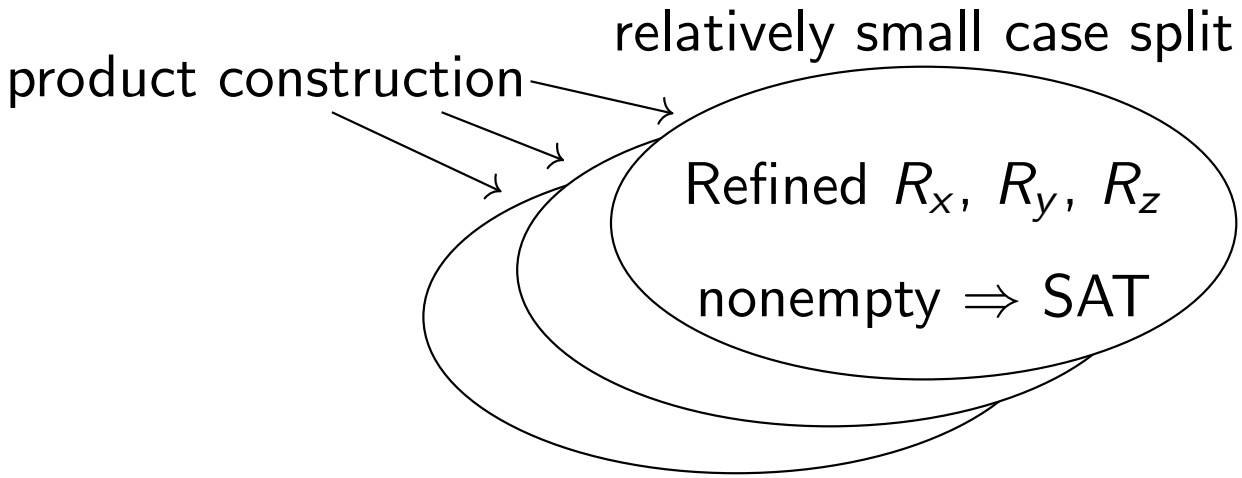
$\varphi[|x_1|, |x_2|, |x_3|, |y|, |z|] \wedge |x| = 100 + 4k$

SMT LIA

Stabilization

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

$$R_x \# R_y \# R_z \;\times\; R_u . R_v . R_w$$



- no new variables, no alignments
- no monadic decomposition

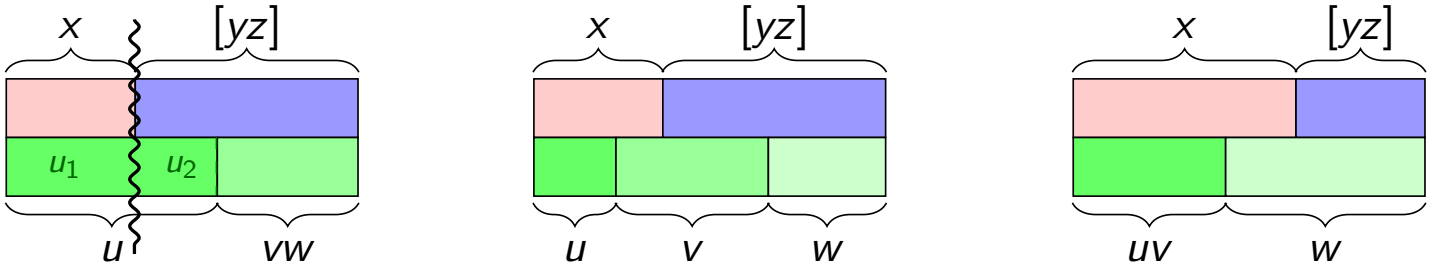
Combined algorithm

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Combined algorithm

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

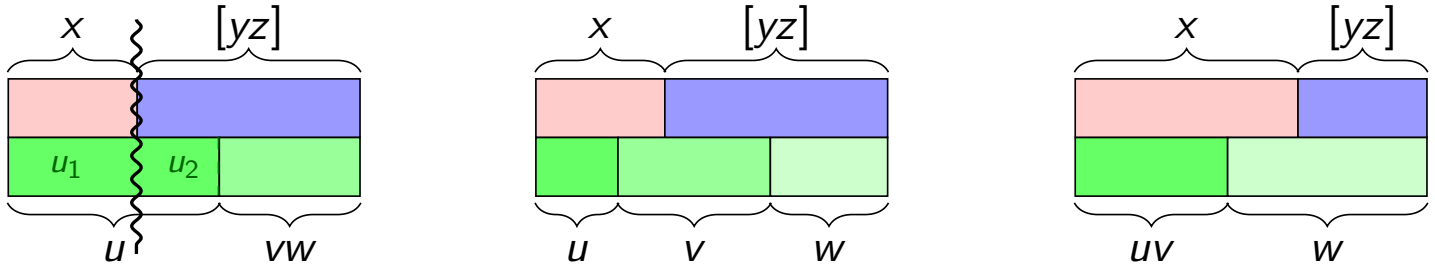
Partial alignments, smaller case split (3)



Combined algorithm

$$xyz = uvw \wedge x \in \mathcal{R}_x \wedge y \in \mathcal{R}_y \wedge z \in \mathcal{R}_z \wedge u \in \mathcal{R}_u \wedge v \in \mathcal{R}_v \wedge w \in \mathcal{R}_w \wedge \ldots \wedge |x| = 100 + 4k$$

Partial alignments, smaller case split (3)



$u / u_1 u_2$
u_1 / x

$[yz] = u_2 vw$

Monadic decomposition for length variables

Solved by stabilization

$[yz] = yz$
$[yz] = [yz]_1 [yz]_2$
$[yz]_1 = u_2$

Extensions and Completeness

- Extended string constr (substr(), indexof(), replace(), contains(), ...) rewritten to basic constr (equations, lengths, regular)
- Saturation with basic constraints further improves performance

Extensions and Completeness

- Extended string constr (substr(), indexof(), replace(), contains(), ...) rewritten to basic constr (equations, lengths, regular)
- Saturation with basic constraints further improves performance
- A new reduction of unrestricted disequations to length constraints
- Complete on the **chain-free s.c. + unrestricted disequations**
  - *(Abdulla, Atig, Diep, Holík, Janků. Chain-Free String Constraints. ATVA'19)*
  - yields a new largest known decidable fragment of
    - (dis)equations,
    - regular and
    - length constraints

Z3-Noodler

Z3-Noodler[1]:

- Based on DPLL(T) SMT solver Z3, replaces its theory of strings
- Relies on our fast automata library Mata[2]
- Very fast and competitive

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

Experimental Evaluation[3]

Unsolved instances

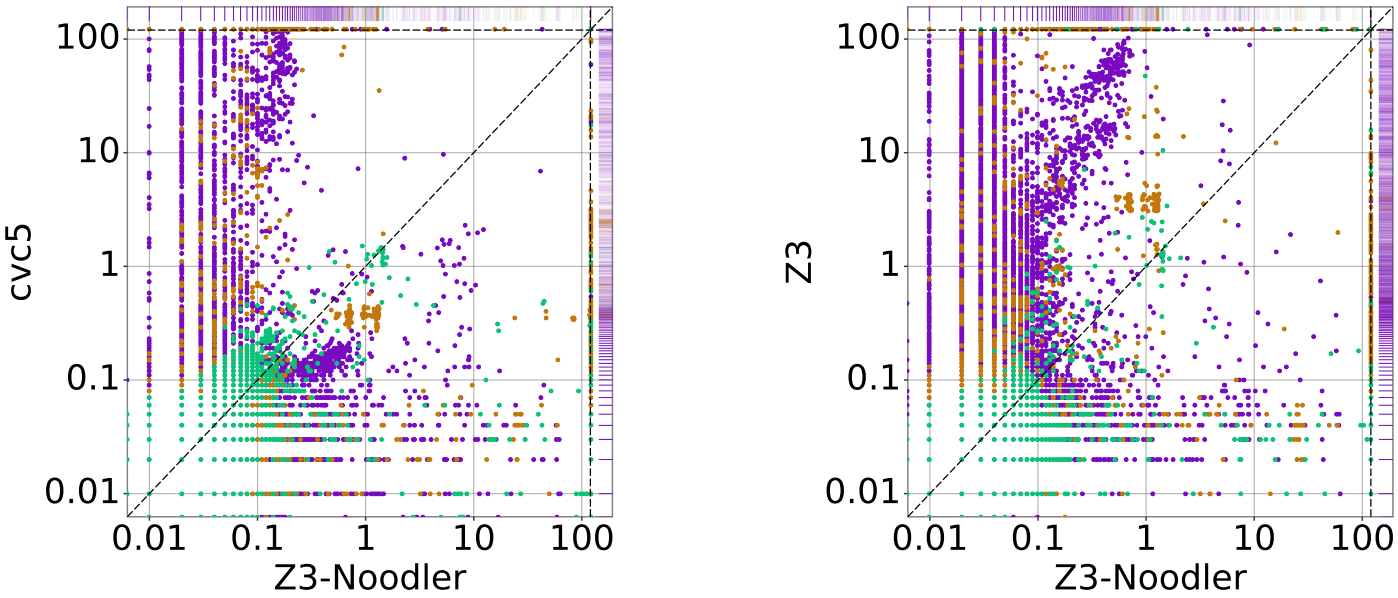
	Regex				Equations							Predicates			
	Aut	Den	StrFuzz	Syg	Kal	Kep	Norn	Slent	Slog	Web	Woo	StrInt	Leet	StrSm	PyEx
Included	15,995	999	10,050	343	19,432	587	1,027	1,128	1,976	267	809	11,669	2,652	1,670	23,845
Unsupported	0	0	1,568	0	0	0	0	0	0	414	0	5,299	0	210	0
Z3-Noodler	62	**0**	**0**	**0**	259	**4**	**0**	**5**	**0**	**0**	243	4	4	55	4,424
cvc5	94	18	1037	**0**	**0**	240	85	22	**0**	40	54	**0**	**0**	**4**	**34**
Z3	113	118	340	**0**	164	313	124	74	71	61	**25**	4	**0**	32	1,071
Z3str4	60	4	27	**0**	174	254	73	73	16	62	78	5	4	37	570
OSTRICH	**55**	15	229	**0**	288	387	1	130	7	65	53	37	26	*106	12,290
Z3str3RE	66	27	*143	1	*144	311	133	87	55	*104	*118	64	192	*179	17,764

Average runtimes

	Reg		Eq		Pred	
	avg	std	avg	std	avg	std
Z3-Noodler	**0.11**	1.35	**0.11**	2.13	0.11	2.16
cvc5	1.17	8.51	**0.11**	2.15	0.03	0.15
Z3	1.92	9.71	0.18	2.83	0.04	0.42
Z3str4	0.35	2.00	0.25	3.40	0.02	0.31
OSTRICH	4.29	8.67	4.28	9.28	12.71	15.08
Z3str3RE	0.31	3.28	0.13	2.72	**0.01**	0.08

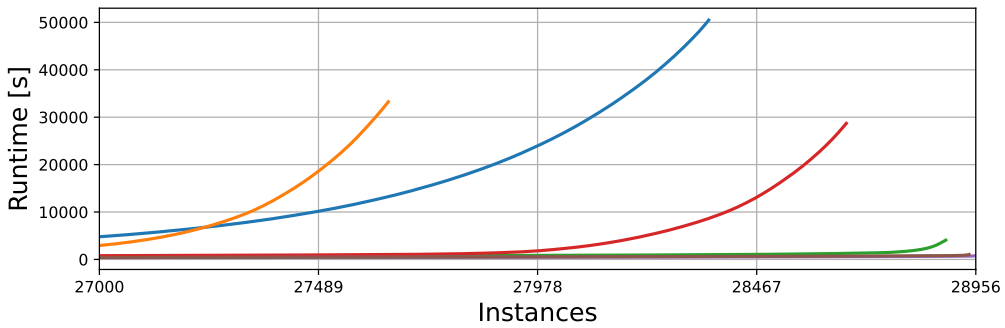
[3] newer results than in the paper

Comparison with CVC5 and Z3

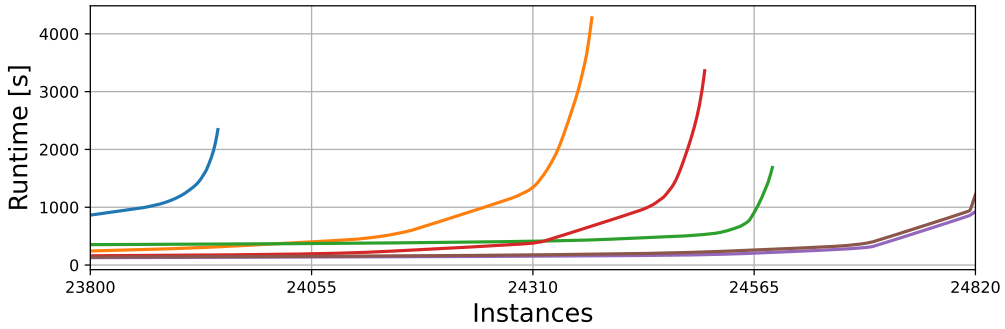


● **Regex**, ● **Equations**, ● **Predicates**.

Virtual Best Solver

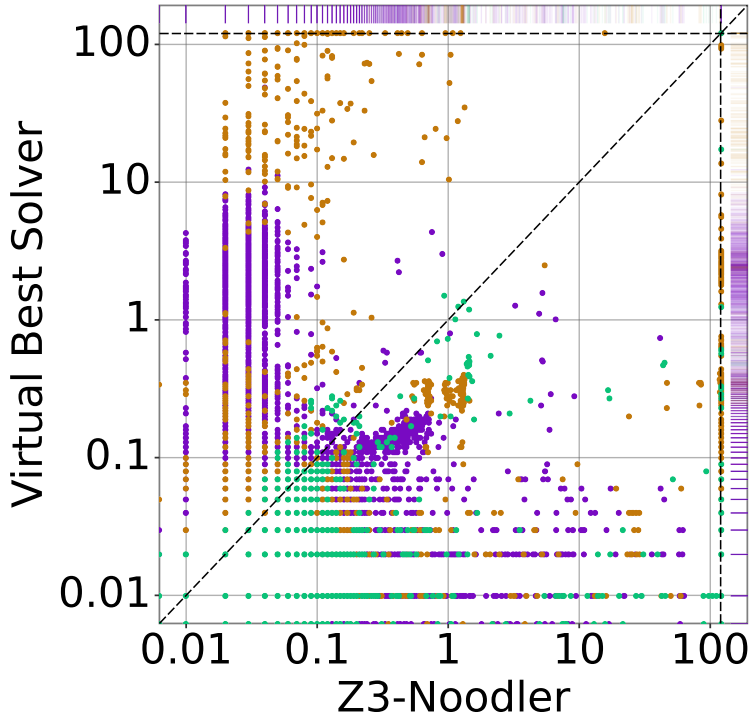


(a) VBS on regex-heavy



(b) VBS on equation-heavy

z3
cvc5
Noodler
cvc5+z3
Noodler+cvc5+z3
Noodler+cvc5



	Regex		Equations	
	Unsolved	Time	Unsolved	Time
VBS	1	427	19	1,304
VBS - Z3-Noodler	1	2,914	131	6,830
VBS - cvc5	1	549	145	1,401
VBS - Z3	1	430	29	1,579
VBS - Z3str4	1	473	19	1,416
VBS - OSTRICH	1	427	21	1,270
VBS - Z3str3RE	1	510	20	1,307
cvc5 + Z3 + Z3-Noodler	1	608	22	1,471
cvc5 + Z3	278	27,916	303	2,805

Conclusion

- Combination of
  - Stabilization (efficient synergy of equations and regular constr.)
  - Align&Split (to handle lengths, disequations, and extended constr.)
- New SMT string solver Z3-Noodler[4]
  - Fast and complementary to s.o.t.a. string solvers
  - Relying on our new fast automata library Mata[5]
- Ongoing work
  - other constraints – transducers, string/int conversion, ...
  - efficiency of automata
  - benchmark-specific heuristics
  - witness/certificate generation

[4] `https://github.com/VeriFIT/z3-noodler`

[5] `https://github.com/VeriFIT/mata`