A String, Regular Expression, and Integer Solver for Bug-finding and Security

Yunhui Zheng[1], Vijay Ganesh[2], Sanu Subramanian[2],
Omer Tripp[1], Murphy Berzish[2], Julian Dolby[1], Xiangyu Zhang[3]

IBM Research
UNIVERSITY OF WATERLOO
PURDUE UNIVERSITY

HCSS 2016

May 10, 2016

Z3str2

- **A solver for strings, regex and length constraints**

- **Currently the top performer on the Kaluza suite** [Abdulla@CAV'15]
  - The only large scale suite available now (generated from JavaScript analysis)

- **Invited by the Z3 team (Microsoft Research) and will be included in the mainstream (MIT License)**

Outline

01

Z3str2 Overview

Solving Word Equations

02

Search space pruning

Two techniques:

1. Overlapping variable detections
2. Theory integrations

03

Evaluation

Constraints from

1. Bug finding
2. Security analysis

String Constraint Solver

- **Constraint Solver**
  - A key piece of machinery for symbolic execution, test generation, etc.

- **Why Strings?**
  - Strings are pervasive in web applications and mobile apps
  - Path predicates, sanitizers, permission settings, …

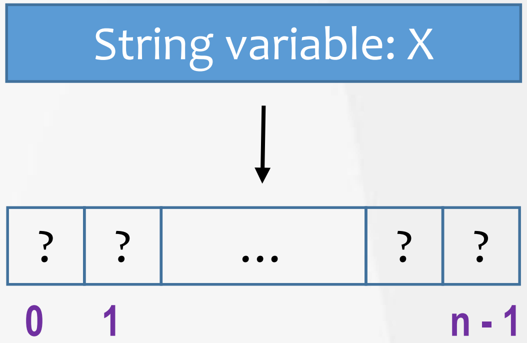
- **Why cohesive reasoning about strings and non-strings?**
  - String operations interact with non-string ones, e.g., length(), substring(), …
  - String equations, regular expressions and length constraints should be supported

Existing String Constraint Solvers

Bit-vector based

- HAMPI [ISSTA'09]
- The solver for Pex [TACAS'09]
- Kaluza [S&P'10]

01



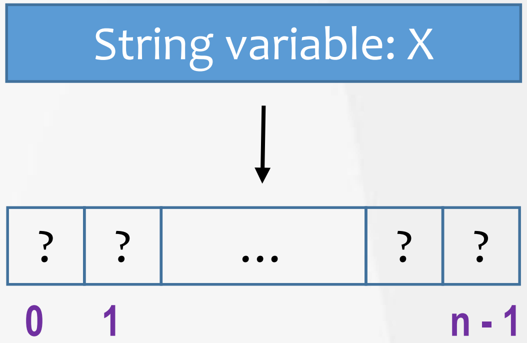
- **Fix Length First**

Existing String Constraint Solvers

01

Bit-vector based

- HAMPI [ISSTA'09]
- The solver for Pex [TACAS'09]
- Kaluza [S&P'10]


String variable: X

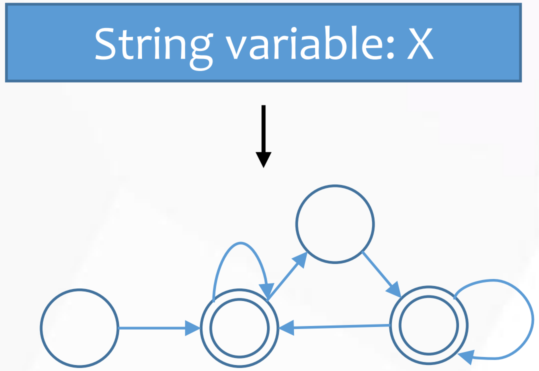
?	?	…	?	?
0	1			n - 1

- **Fix Length First**

02

Automata based

- JSA [SAS'03]
- Hooimeijer [ASE'10]
- PISA-MONA [ISSTA'11]
- Stranger [TACAS'10]
- PASS [HVC'13]
- JST [ICSE'13]

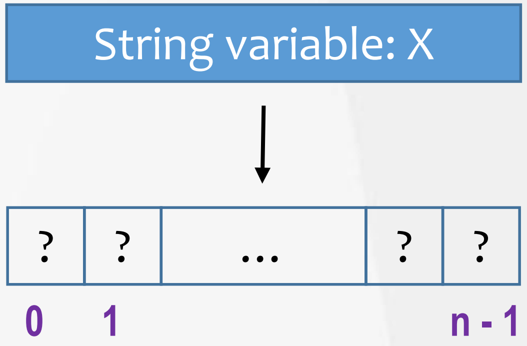

String variable: X

- **Over-approximations**
- **Hard to capture theory connections**

Existing String Constraint Solvers

Bit-vector based

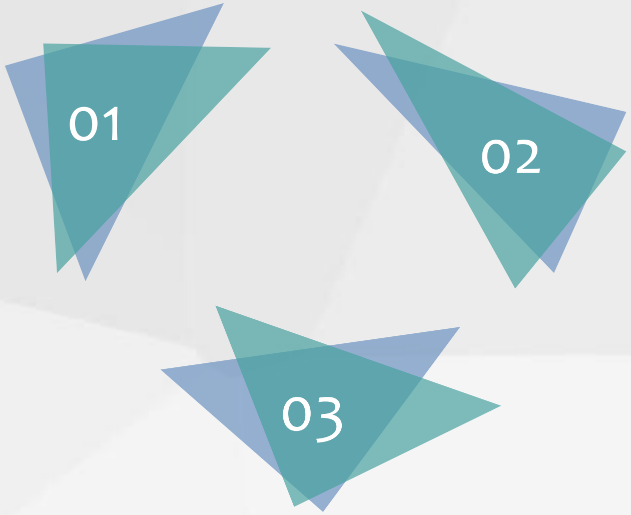
- HAMPI [ISSTA'09]
- The solver for Pex [TACAS'09]
- Kaluza [S&P'10]



String variable: X

?	?	…	?	?
0	1			n - 1

- **Fix Length First**



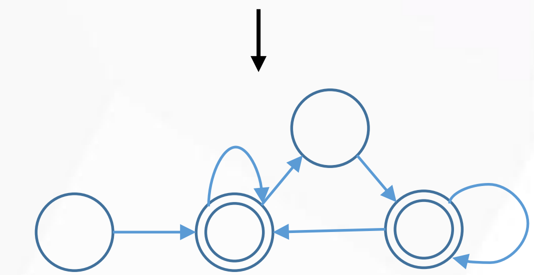
Word- / SMT-based

- **Z3-str [FSE'13]**
- CVC4 [CAV'14]
- Norn [CAV'14, CAV'15]
- S3 [CCS'14] (atop Z3-str)
- **Z3str2 [CAV'15]**

Automata based

- JSA [SAS'03]
- Hooimeijer [ASE'10]
- PISA-MONA [ISSTA'11]
- Stranger [TACAS'10]
- PASS [HVC'13]
- JST [ICSE'13]

String variable: X



- **Over-approximations**
- **Hard to capture theory connections**

Challenges

- **Solving string equations is known to be hard**
  - Makanin 1977, Plandowski 2006, …

- **Finding a consistent solution in multiple domains is expensive**
  - **An inconsistent “solution” (UNSAT):** X = “ab” /\ length(X) = 1
  - **Infinite** search space in both string and integer domains

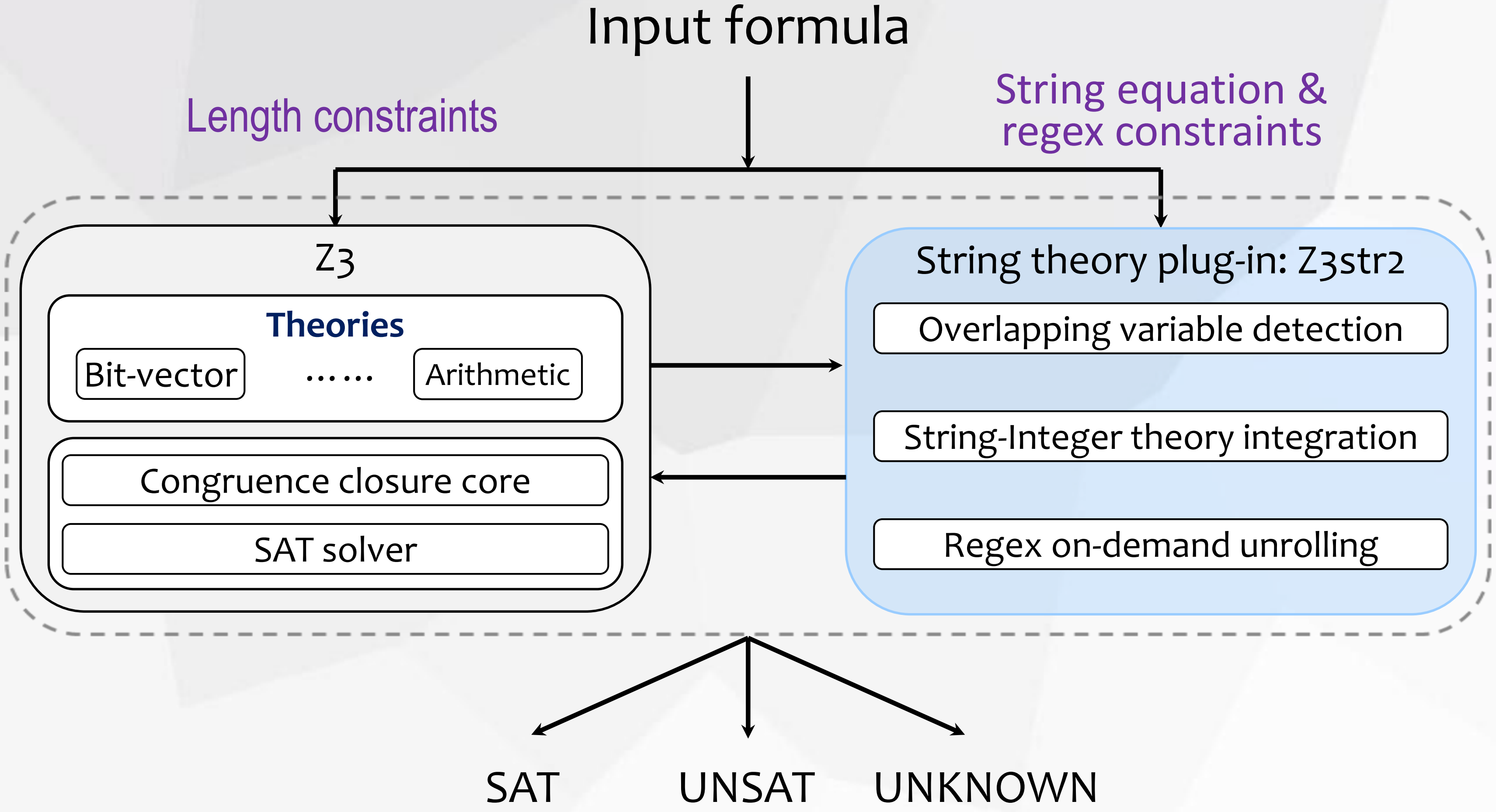
The Theory Considered

- **The quantifier-free theory over word equations, membership predicates, and length functions**

- **Multi-sorted theory**
  - String (strings over ASCII characters)
  - Integer (natural numbers)
  - Boolean

- **Common string related operators**
  - CharAt, Contains, StartsWith, Indexof, Substring, Regex….

Z3str2 Overview


Input formula
Length constraints
String equation & regex constraints
Z3
Theories
Bit-vector
……
Arithmetic
Congruence closure core
SAT solver
String theory plug-in: Z3str2
Overlapping variable detection
String-Integer theory integration
Regex on-demand unrolling
SAT
UNSAT
UNKNOWN

Sound but incomplete

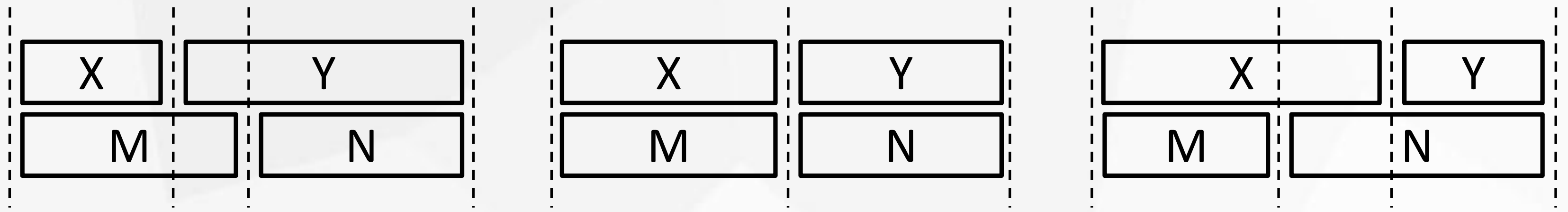
Supporting string operations

- **Three primitive string operations**
  - String equivalence
  - Concatenation
  - String length
- **Reduce other string operations to primitives**
  - Contains
  - IndexOf
  - Substring
  - Replace
  - Regex membership predicate
  - …

Solving String Equations

- **Basic idea**
  - Recursively split equations into smaller ones until they are directly solvable
  - Given an equation, identify all possible arrangements

X . Y = M . N



3 possible arrangements

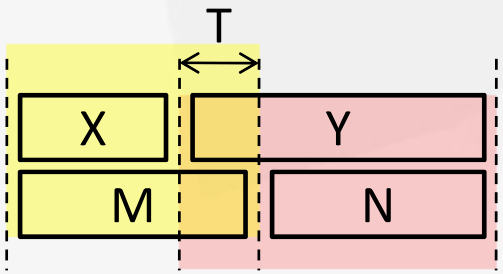
X, Y, M and N are all string variables

Solving String Equations

Basic idea

- Recursively split equations into smaller ones until they are directly solvable
- Given an equation, identify all possible arrangements
- Given an arrangement, generate smaller equations

$$X . Y = M . N$$



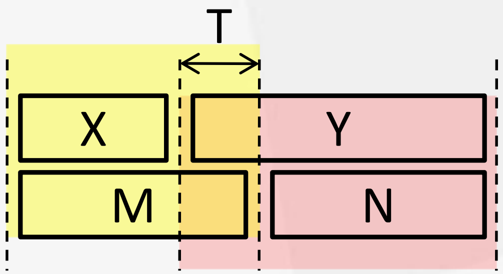
$M = X . T$ **$Y = T . N$** ⇦ **Smaller Equations**

- Keep splitting until solved
- If conflicts detected, rollback, try another arrangement

Sync with Integer Theory

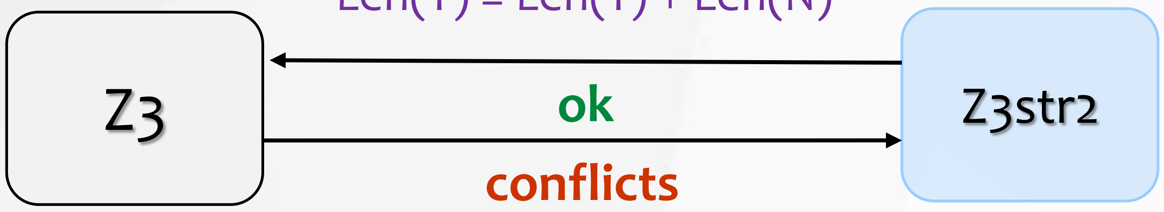
- **Consistent solutions in both theories**
  - Z3str2 asserts new length constraints during search

X . Y = M . N



M = X . T **Y = T . N**

Len(T) > 0
Len(M) = Len(X) + Len(T)
Len(Y) = Len(T) + Len(N)



- Keep splitting
- Rollback. Try another arrangement

Outline

01

Z3str2 Overview

Solving Word Equations

02

Search space pruning

Two techniques:

1. Overlapping variable detections
2. Theory integrations

03

Evaluation

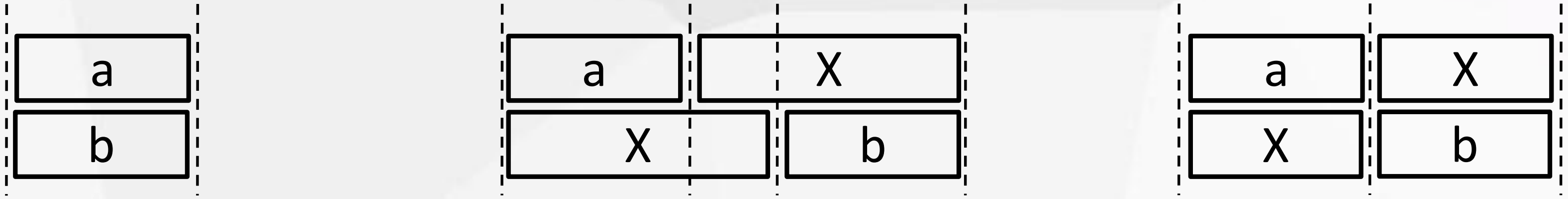
Constraints from

1. Bug finding
2. Security analysis

Overlapping Variables

- **Non-terminations caused by overlapping variables**

“a” . X = X . “b”



a
b

a X
X b

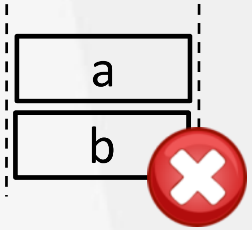
a X
X b

3 possible arrangements

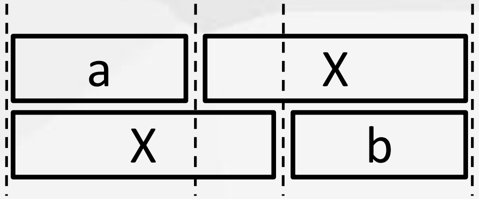
Overlapping Variables

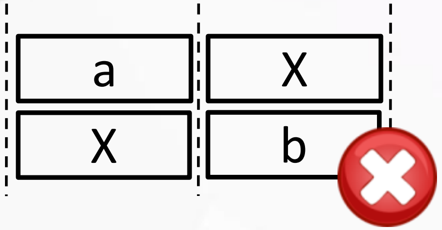
- Non-terminations caused by overlapping variables

“a” . X = X . “b”



X = ε

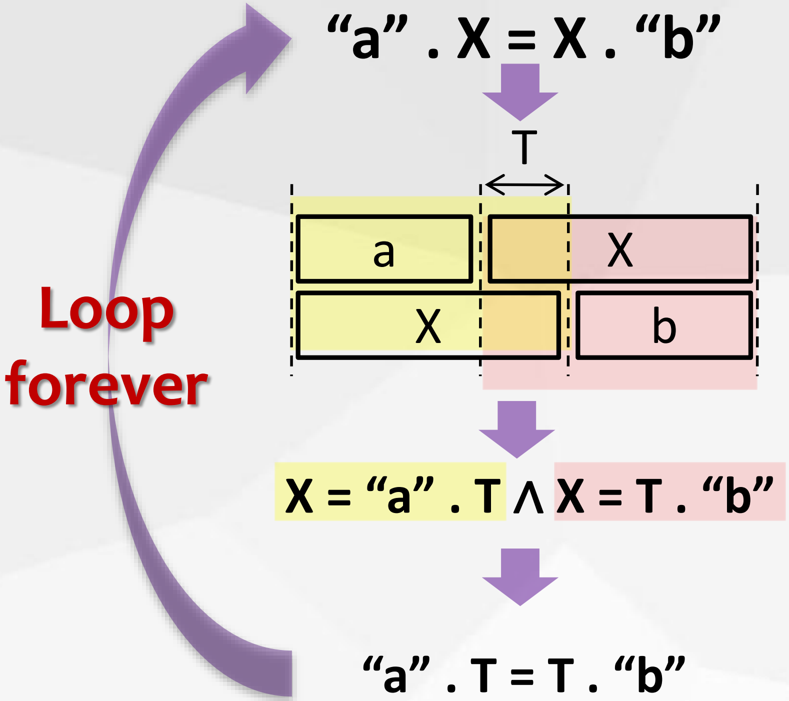




X = “a” ∧ x = “b”

Overlapping Variables

- Non-terminations caused by overlapping variables



“a” . X = X . “b”

X = “a” . T ∧ X = T . “b”

“a” . T = T . “b”

Reason

X splits itself

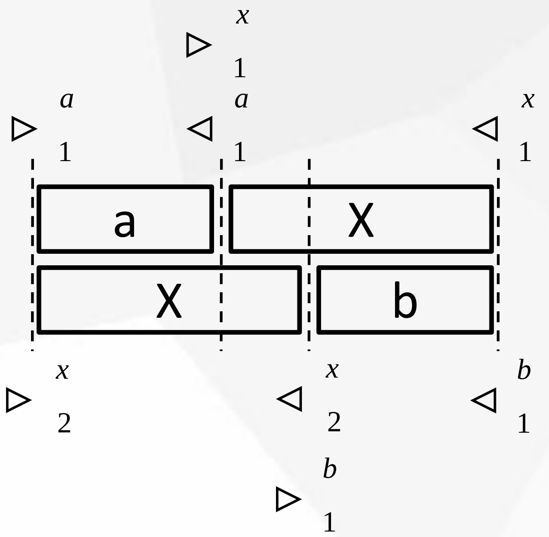
This is a simple example. In general, loops may be formed via multiple equations

Overlapping Variables

- **A boundary label based system**
  - Arrangement and sub-equation generation

$$\boxed{\quad a \quad}\ \boxed{\quad X \quad} = \boxed{\quad X \quad}\ \boxed{\quad b \quad}$$

$$\{\triangleright^{a}_{1}\}\ \{\triangleleft^{a}_{1}, \triangleright^{x}_{1}\}\ \{\triangleleft^{x}_{1}\}\ \{\triangleright^{x}_{2}\}\ \{\triangleleft^{x}_{2}, \triangleright^{b}_{1}\}\ \{\triangleleft^{b}_{1}\}$$



Arrangements:

X = T . “b”

$$\{\triangleright^{a}_{1}, \triangleright^{x}_{2}\} \bullet \{\triangleleft^{a}_{1}, \triangleright^{x}_{1}\} \bullet \{\triangleleft^{x}_{2}, \triangleright^{b}_{1}\} \bullet \{\triangleleft^{x}_{1}, \triangleleft^{b}_{1}\}$$

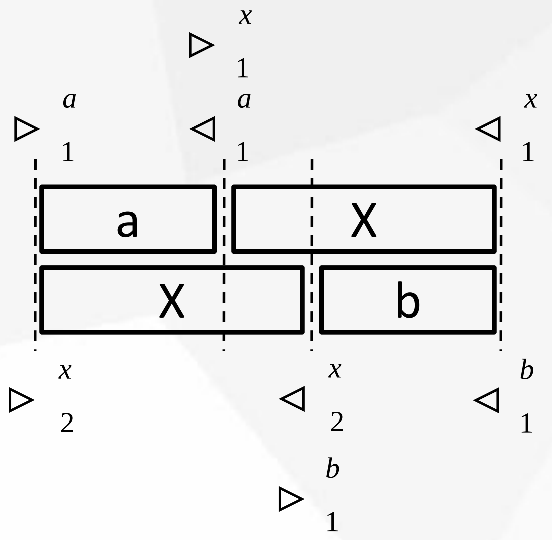
X = “a” . T

Overlapping Variables

- **A boundary label based system**
  - Arrangement and sub-equation generation
  - An algorithm detecting and avoiding such loops

$$\boxed{\;a\;}\ \boxed{\;X\;} = \boxed{\;X\;}\ \boxed{\;b\;}$$

$$\{\triangleright^{a}_{1}\}\ \{\triangleleft^{a}_{1}, \triangleright^{x}_{1}\}\ \{\triangleleft^{x}_{1}\}\ \{\triangleright^{x}_{2}\}\ \{\triangleleft^{x}_{2}, \triangleright^{b}_{1}\}\ \{\triangleleft^{b}_{1}\}$$



Arrangements:

$$\{\triangleright^{a}_{1}, \triangleright^{x}_{2}\} \bullet \{\triangleleft^{a}_{1}, \triangleright^{x}_{1}\} \bullet \{\triangleleft^{x}_{2}, \triangleright^{b}_{1}\} \bullet \{\triangleleft^{x}_{1}, \triangleleft^{b}_{1}\}$$

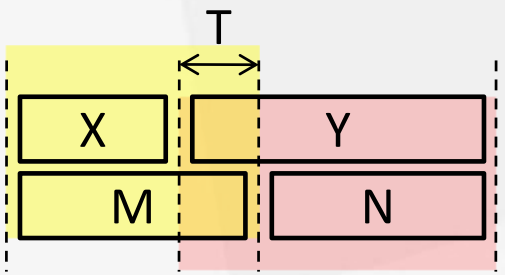
⬆

Overlap Detected

String and Integer Theory Integration

- Undesirable searches in the integer theory (1)

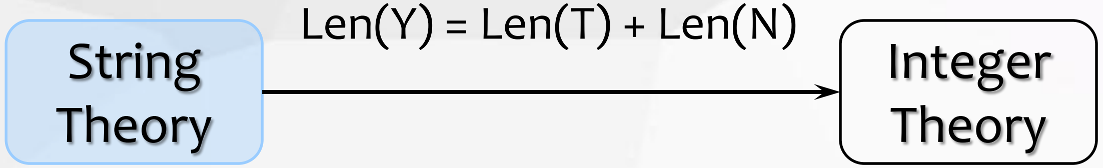
$X . Y = M . N$



$M = X . T$

$Y = T . N$

Len(T) > 0
Len(M) = Len(X) + Len(T)
Len(Y) = Len(T) + Len(N)



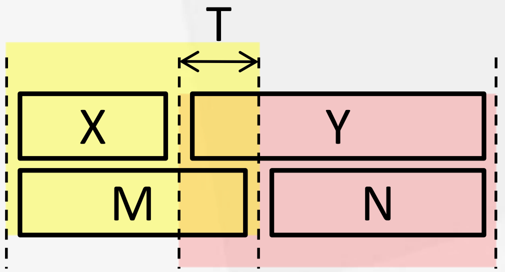
- Len(M) > Len(X) ?
- Len(M) = Len(X) ?
- Len(M) < Len(X) ?

Unnecessary Efforts

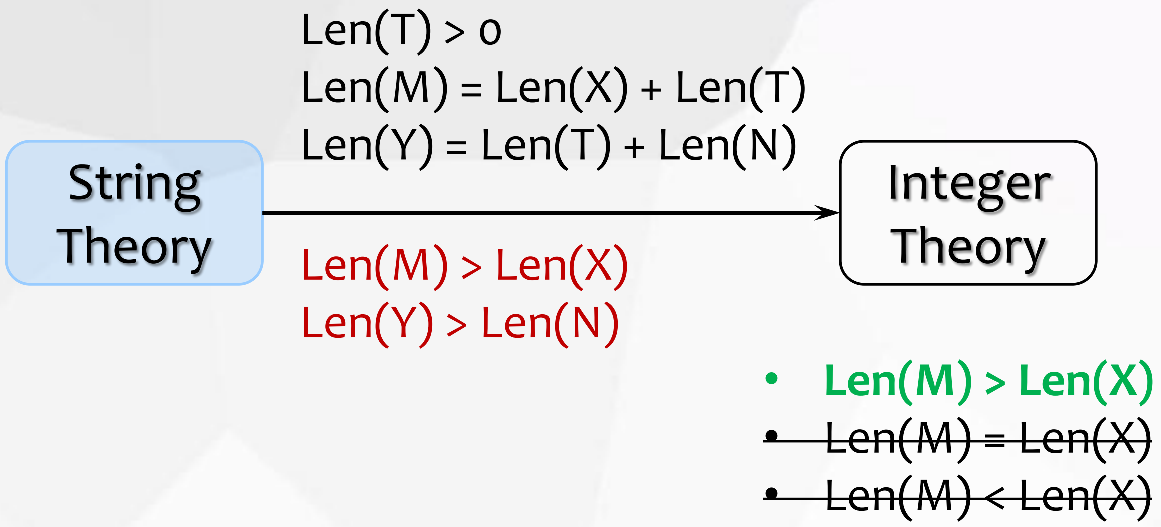
String and Integer Theory Integration

- Integration: string --> integer

X . Y = M . N



M = X . T **Y = T . N**


String Theory

Len(T) > 0
Len(M) = Len(X) + Len(T)
Len(Y) = Len(T) + Len(N)

Len(M) > Len(X)
Len(Y) > Len(N)

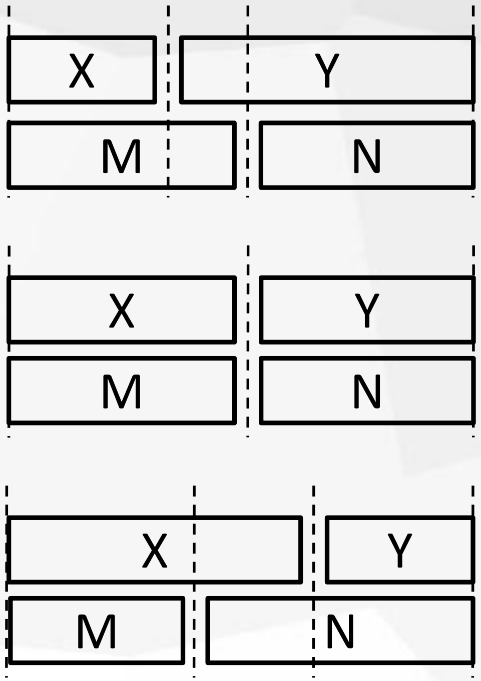
Integer Theory

- **Len(M) > Len(X)**
- ~~Len(M) = Len(X)~~
- ~~Len(M) < Len(X)~~

String and Integer Theory Integration

- Undesirable searches in the string theory (2)

X . Y = M . N



String Theory

Context:

Len(M) < Len(X)

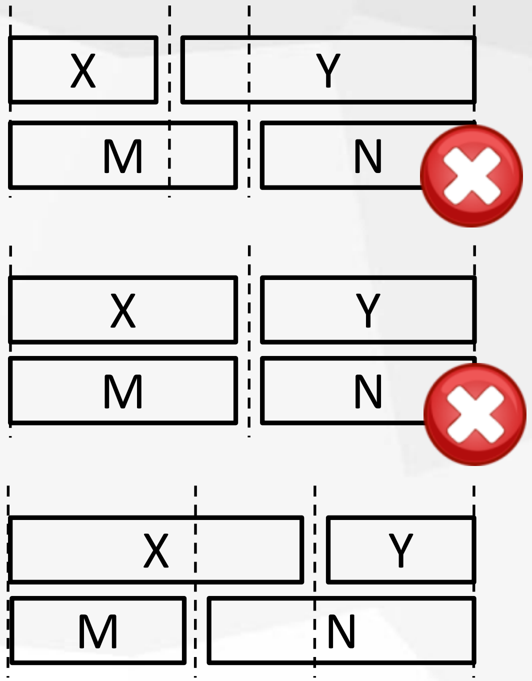
……

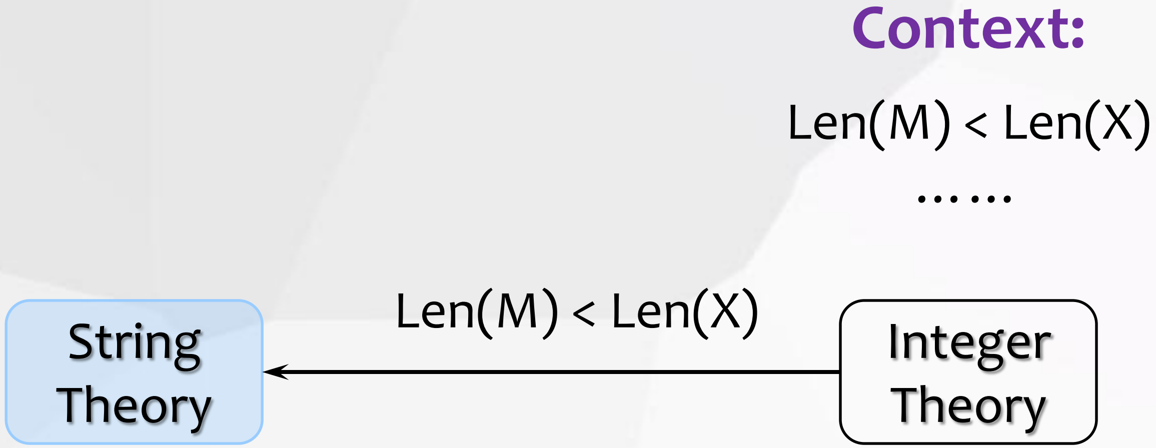
Integer Theory

String and Integer Theory Integration

- Undesirable searches in the string theory (2)

X . Y = M . N





- Explore 1 arrangement instead of 3
- Space pruning without concrete length assignments (thanks to the SMT engine)

Outline

01

Z3str2 Overview

Solving Word Equations

02

Search space pruning

Two techniques:

1. Overlapping variable detections
2. Theory integrations

03

Evaluation

Constraints from

1. Bug finding
2. Security analysis

Comparison Study Setting -- Solvers

- **Word-based SMT String Solvers**
  - Z3str2 [CAV’15]
  - Z3str2 without theory integration
  - CVC4 (V1.5-prerelease) [CAV’14]
  - S3 (built on top of Z3-str) [CCS’14]
- **Bit-vector based String Solvers**
  - Kaluza (the version from the CVC4 paper) [S&P’10]
- **Automata-based String Solvers**
  - PISA-MONA [ISSTA’11]
  - Stranger (a patched version by the Stranger team) [TACAS’10]

Comparison Study Setting -- Benchmarks

- **Bug finding / Testing**
  - **Kaluza Suite**: generated from JavaScript symbolic execution [S&P'10]
    - String operators: Length, Equivalence, Concat, Regex
  - **Kausler Suite**: Java dynamic symbolic execution [ASE'14]
    - Pure string constraints (constant indices and length bounds)
- **Security Analysis**
  - **PISA Suite** [ISSTA'11]
    - Java Sanitizer Analysis
  - **AppScan Suite**
    - JavaScript Security warning outputs of IBM Security AppScan Source Edition
  - A rich set of string operators

Bug finding / Testing: Kaluza Suite

	Z3str2		Z3str2 w/o integration		CVC4		S3		Kaluza	
Answer Verification	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
SAT	34,859	0	32,752	0	33,190	0	32,503	488	21,651	n/a+
UNSAT	11,799	0	11,313	0	11,625	0	11,351	412	12,099	10,909
Unknown	626 (1.3%)		395 (0.8%)		0		0		0	
Timeout (20s)	0		2,824		2,469		989		340	
Tool reports error	0		0		0		2		2285	
crash	0		0		0		1539		0	
Total time (sec)	4288.8 (1x)		61,232.8 (14.3x)		52,478.8 (12.2x)		22543.4 (5.3x)		46753.9 (10.9x)	
Avg time (sec)	0.091		1.295		1.110		0.477		0.989	

+ Kaluza only provides assignments for variable matching the query string, we cannot verify.

Bug finding / Testing: Kaluza Suite

	Z3str2		Z3str2 w/o integration		CVC4		S3		Kaluza	
Answer Verification	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
SAT	34,859	0	32,752	0	33,190	0	32,503	488	21,651	n/a+
UNSAT	11,799	0	11,313	0	11,625	0	11,351	412	12,099	10,909
Unknown	626 (1.3%)		395 (0.8%)		0		0		0	
Timeout (20s)	0		2,824		2,469		989		340	
Tool reports error	0		0		0		2		2285	
crash	0		0		0		1539		0	
Total time (sec)	4288.8 (1x)		61,232.8 (14.3x)		52,478.8 (12.2x)		22543.4 (5.3x)		46753.9 (10.9x)	
Avg time (sec)	0.091		1.295		1.110		0.477		0.989	

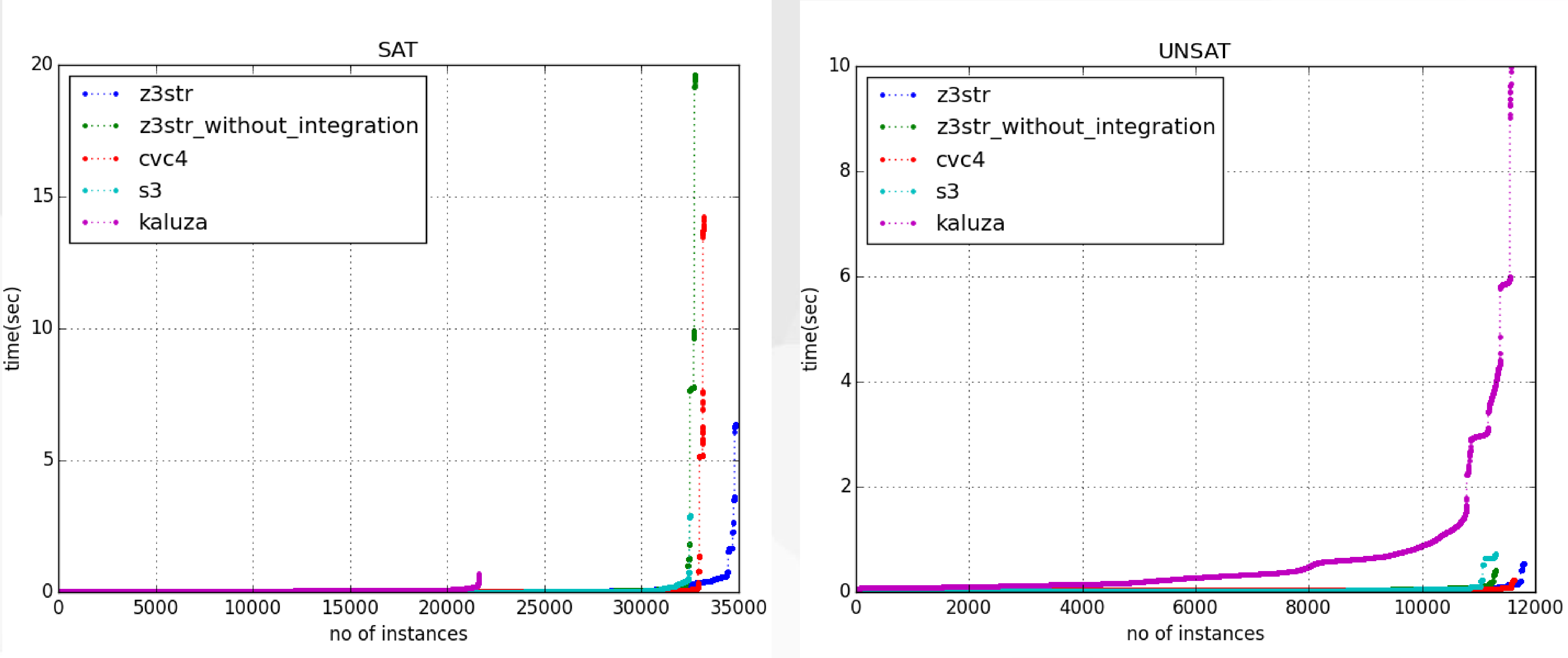
+ Kaluza only provides assignments for variable matching the query string, we cannot verify.

Bug finding / Testing: Kaluza Suite

	Z3str2		Z3str2 w/o integration		CVC4		S3		Kaluza	
Answer Verification	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
SAT	34,859	0	32,752	0	33,190	0	32,503	488	21,651	n/a+
UNSAT	11,799	0	11,313	0	11,625	0	11,351	412	12,099	10,909
Unknown	626 (1.3%)		395 (0.8%)		0		0		0	
Timeout (20s)	0		2,824		2,469		989		340	
Tool reports error	0		0		0		2		2285	
crash	0		0		0		1539		0	
Total time (sec)	4288.8 (1x)		61,232.8 (14.3x)		52,478.8 (12.2x)		22543.4 (5.3x)		46753.9 (10.9x)	
Avg time (sec)	0.091		1.295		1.110		0.477		0.989	

+ Kaluza only provides assignments for variable matching the query string, we cannot verify.

Bug finding / Testing: Kaluza Suite



Cactus Plots

(incorrect results excluded)

Bug finding / Testing: Kausler Suite

		Avg time per instance (ms)	
	Instance #	Z3str2	Stranger
Beasties	7230	6.4	51.8
jerichoHTMLParser	1275	10.7	5.9
mathParser	9138	39.9	1.4
mathQuizGame	21	7.1	9.4
naturalCLI	2367	23.4	3.0

- The cases where Stranger crashed or hanged are excluded
- Stranger over-approximates dis-equalities (≠) among variables that can represent multiple strings. We believe the constraints are thus easier for stranger.

Security analysis: PISA Suite

		Z3str2			CVC4			PISA-MONA		
				time(s)			time(s)			time(s)
PISA-000.smt2	contains, indexof, substring	**sat**	✓	**0.164**	**sat**	✓	**0.264**	**sat**	?	**0.029**
PISA-001.smt2	contains, indexof, substring	**sat**	✓	**0.114**	**sat**	✓	**0.032**	---+	---+	+---
PISA-002.smt2	contains	**sat**	✓	**0.114**	**sat**	✓	**50.871**	---+	---+	+---
PISA-003.smt2	contains, concat	**unsat**	✓	**0.064**	**timeout**		**200.00**	---+	---+	+---
PISA-004.smt2	contains, indexof, length, lastIndexof, substring	**unsat**	✓	**0.038**	**timeout**		**0.165**	**unsat**	✓	**0.041**
PISA-005.smt2	Indexof, lastIndexof, length, substring	**sat**	✓	**0.115**	**sat**	✓	**200.00**	---+	---+	+---
PISA-006.smt2	Indexof, lastIndexof, length, substring	**unsat**	✓	**0.039**	**timeout**		**200.00**	**unsat**	✓	**0.038**
PISA-007.smt2	Indexof, lastIndexof, length, substring, contains	**unsat**	✓	**0.042**	**timeout**		**200.00**	**unsat**	✓	**0.039**
PISA-008.smt2	replace, contains	**sat**	✓	**0.214**	**timeout**		**200.00**	**sat**	?*	**0.031**
PISA-009.smt2	replace, concat, contains	**sat**	✓	**0.447**	**sat**	✓	**0.046**	**sat**	?*	**0.054**
PISA-010.smt2	replace, concat	**sat**	✓	**0.165**	**timeout**		**200.00**	---+	---+	+---
PISA-011.smt2	replace, concat	**sat**	✓	**0.115**	**sat**	✓	**0.016**	---+	---+	+---

+ We could not generate constraints without changing PISA

* No string solutions are generated so it's not verifiable

Security analysis: PISA Suite

		Z3str2			CVC4			PISA-MONA		
				time(s)			time(s)			time(s)
PISA-000.smt2	contains, indexof, substring	**sat**	✓	**0.164**	**sat**	✓	**0.264**	**sat**	**?**	**0.029**
PISA-001.smt2	contains, indexof, substring	**sat**	✓	**0.114**	**sat**	✓	**0.032**	**---**[+]	**---**[+]	**---**[+]
PISA-002.smt2	contains	**sat**	✓	**0.114**	**sat**	✓	**50.871**	**---**[+]	**---**[+]	**---**[+]
PISA-003.smt2	contains, concat	**unsat**	✓	**0.064**	**timeout**		**200.00**	**---**[+]	**---**[+]	**---**[+]
PISA-004.smt2	contains, indexof, length, lastIndexof, substring	**unsat**	✓	**0.038**	**timeout**		**0.165**	**unsat**	✓	**0.041**
PISA-005.smt2	Indexof, lastIndexof, length, substring	**sat**	✓	**0.115**	**sat**	✓	**200.00**	**---**[+]	**---**[+]	**---**[+]
PISA-006.smt2	Indexof, lastIndexof, length, substring	**unsat**	✓	**0.039**	**timeout**		**200.00**	**unsat**	✓	**0.038**
PISA-007.smt2	Indexof, lastIndexof, length, substring, contains	**unsat**	✓	**0.042**	**timeout**		**200.00**	**unsat**	✓	**0.039**
PISA-008.smt2	replace, contains	**sat**	✓	**0.214**	**timeout**		**200.00**	**sat**	**?**[*]	**0.031**
PISA-009.smt2	replace, concat, contains	**sat**	✓	**0.447**	**sat**	✓	**0.046**	**sat**	**?**[*]	**0.054**
PISA-010.smt2	replace, concat	**sat**	✓	**0.165**	**timeout**		**200.00**	**---**[+]	**---**[+]	**---**[+]
PISA-011.smt2	replace, concat	**sat**	✓	**0.115**	**sat**	✓	**0.016**	**---**[+]	**---**[+]	**---**[+]

+ We could not generate constraints without changing PISA

* No string solutions are generated so it's not verifiable

Security analysis: AppScan Suite

		Z3str2			CVC4		
				time(s)			time(s)
t01.smt2	indexof, substring	**sat**	✓	**0.265**	**timeout**		200.00
t02.smt2	concat, membership, regexConcat, regexUnion, str2Regex, length	**sat**	✓	0.215	**sat**	✓	**0.026**
t03.smt2	concat, membership, regexConcat, regexUnion, str2Regex, length	**sat**	✓	**2.519**	**timeout**		200.00
t04.smt2	concat, membership, regexConcat, regexUnion, str2Regex, length	**sat**	✓	**4.574**	**timeout**		200.00
t05.smt2	concat, membership, regexConcat, regexUnion, str2Regex, length, substring	**sat**	✓	**2.779**	**timeout**		200.00
t06.smt2	concat, indexof, endsWith	**sat**	✓	**0.214**	**sat**	✓	3.021
t07.smt2	concat, regexStar, Str2Regex, endsWith, regexUnion, membership, startsWith	**sat**	✓	**0.114**	**sat**	✓	0.115
t08.smt2	concat, regexStar, Str2Regex, endsWith, regexUnion, membership, startsWith	**sat**	✓	**0.164**	**sat**	✓	151.66

Conclusion

- **Z3str2: An SMT string constraint solver**
  - A Solver for Strings, Regex and Length Constraints
  - Source code and benchmarks are available @ https://sites.google.com/site/z3strsolver/

- **Two search-space pruning techniques**
  - Overlapping variables detection
  - Tight bi-directional integer-string theory integration

- **We show the efficacy on two groups of benchmark sets**
  - Bug findings
  - Security analysis

THANKS!

www https://sites.google.com/site/z3strsolver/

✉ zhengyu@us.ibm.com