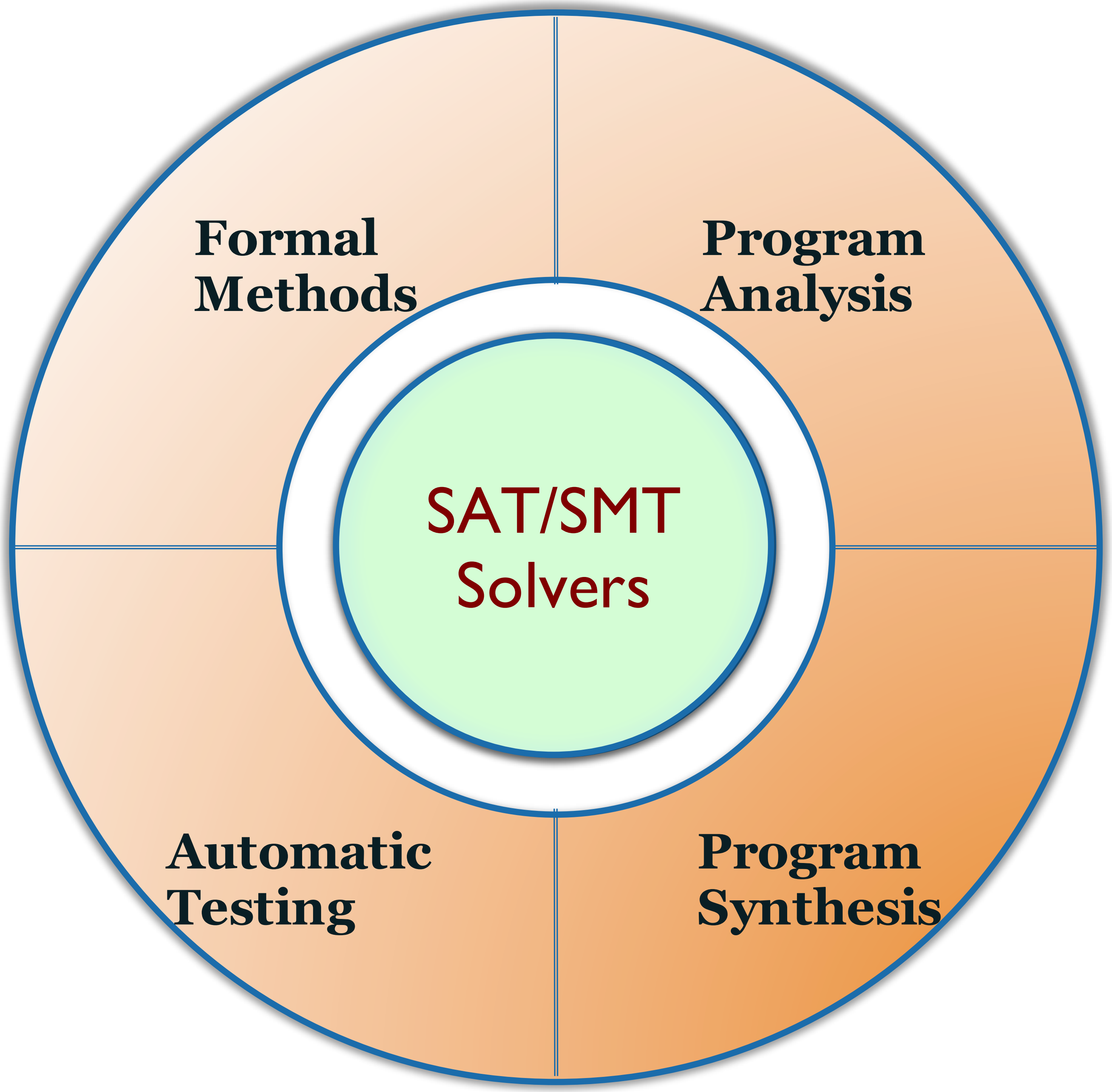
String Solvers for Web Security

Vijay Ganesh
Affiliation: University of Waterloo

Software Engineering & SMT Solvers

An Indispensable Tactic for Most Strategies



Why a String Solver?

Efficient Analysis of String Programs

- Strings are heavily used in Web applications
- Web applications plagued by string-related errors
- An SMT solver that natively reasons about strings can lower analysis burden

Programs that use strings	**Errors attributable to insufficient string analysis**
Traditional Applications C/C++ programs (string operations)	**Memory-related Errors** Buffer overflow due overly long strings
Web Applications PHP JavaScript String manipulation by Server or client-side code	**Improper Sanitization** SQL injection XSS scripting JavaScript Eval with user/attacker-supplied strings

Theory of Word Equations, Length and Membership

Symbol	String Sort	Number Sort
Constants	Finite-length strings defined over a finite alphabet Σ	Integers
Variables	Range over Σ^*	Range over integers
String functions	Concat: String × String ⇒ String Length: String ⇒ Integer	
Integer functions		Addition: Integer × Integer ⇒ Integer
String predicates	Equality over string terms (= : String × String ⇒ Bool) membership in regular expressions/CFGs (∈: String × regular-expression ⇒ Bool) Contains predicate: (Contains: String × String ⇒ Bool)	
Integer predicates		Equality over integer terms (=: Integer × Integer ⇒ Bool) Inequality over integer terms (<: Integer × Integer ⇒ Bool)

Theory of Strings
Example Constraints

- X = concat(“SELECT msg FROM msgs WHERE topicid = ”,v)
  AND
  (X $\in$ SQL_Grammar)

- input $\in$ RegExp([0-9]+)

- X = concat (str_term1, str_term2, “c”)[1:42]
  AND
  X contains “abc”

- Xa = aX, XabY = YbaX

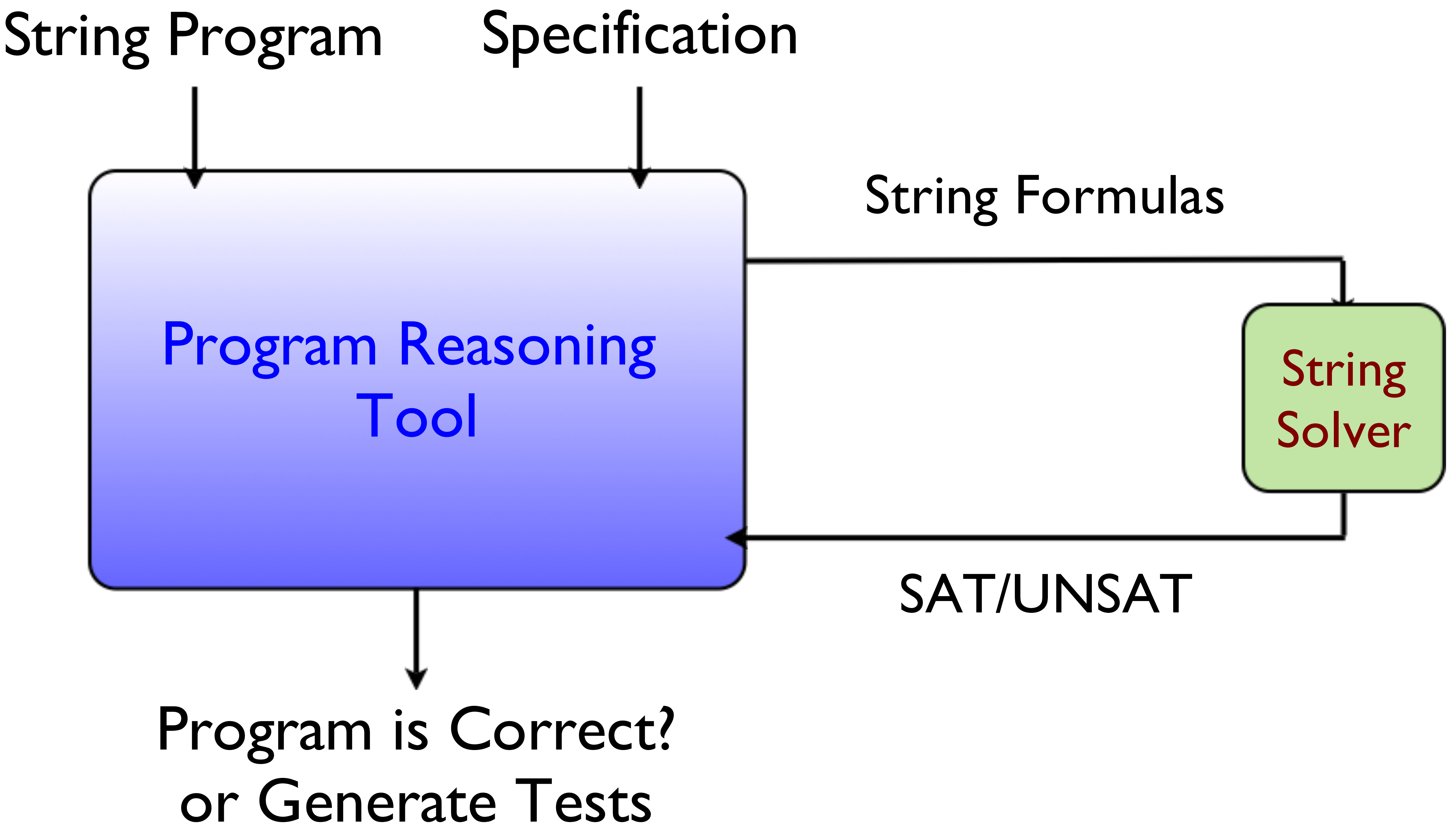
Word Equations, Membership, and Length

What is Known

Result	Person (Year)	Notes
Undecidability of Quantified Word Equations	Quine (1946)	Multiplication reduced to concat
Undecidability of Quantified Word Equations with single alternation	Durnev (1996), G. (2012)	2-counter machines reduced to words with single quantifier alter.
Decidability (PSPACE) of QF Theory of Word Equations	Makanin (1977) Plandowski (1996, 2002/06)	Makanin result very difficult Simplified by Plandowski
Decidability (PSPACE-complete) of QF Theory of Word Equations + RE	Schultz (1992)	RE membership predicate
QF word equations + Length() (?)	Matiyasevich (1971)	Unsolved

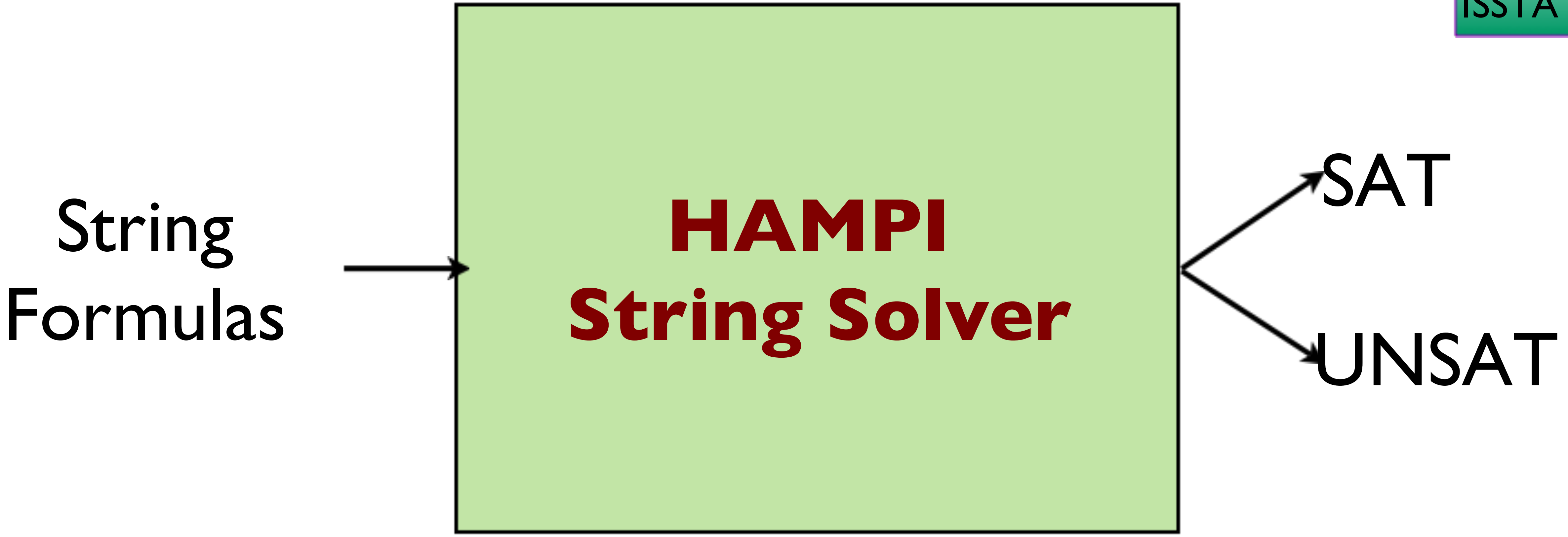
String Solver Problem Statement

Efficient Solver for Analysis of String Programs



HAMPI String Solver

TOSEM 2012
CAV 2011
ISSTA 2009



- X = concat(“SELECT...”,v) AND (X ∈ SQL_grammar)
- JavaScript, PHP, ... string expressions
- NP-complete
- ACM Distinguished Paper Award 2009
- Google Faculty Research Award 2011

Rest of the Talk

- HAMPI string solver
  - String equations and membership in regular expressions/CFGs
  - How HAMPI works
  - Experimental results
- Theoretical results
  - Undecidability of forallexists fragment of word equations
  - Conditional decidability results for word equations and length
  - Open theoretical problems
- Z3-str
  - A solver for string equations and length function

Theory of Strings

The Hampi Language

PHP/JavaScript/C++…	HAMPI: Theory of Strings	Notes

Var a;
$a = ‘name’
	Var a : 1...20;
	a = ‘name’
		Bounded String Variables
		String Constants

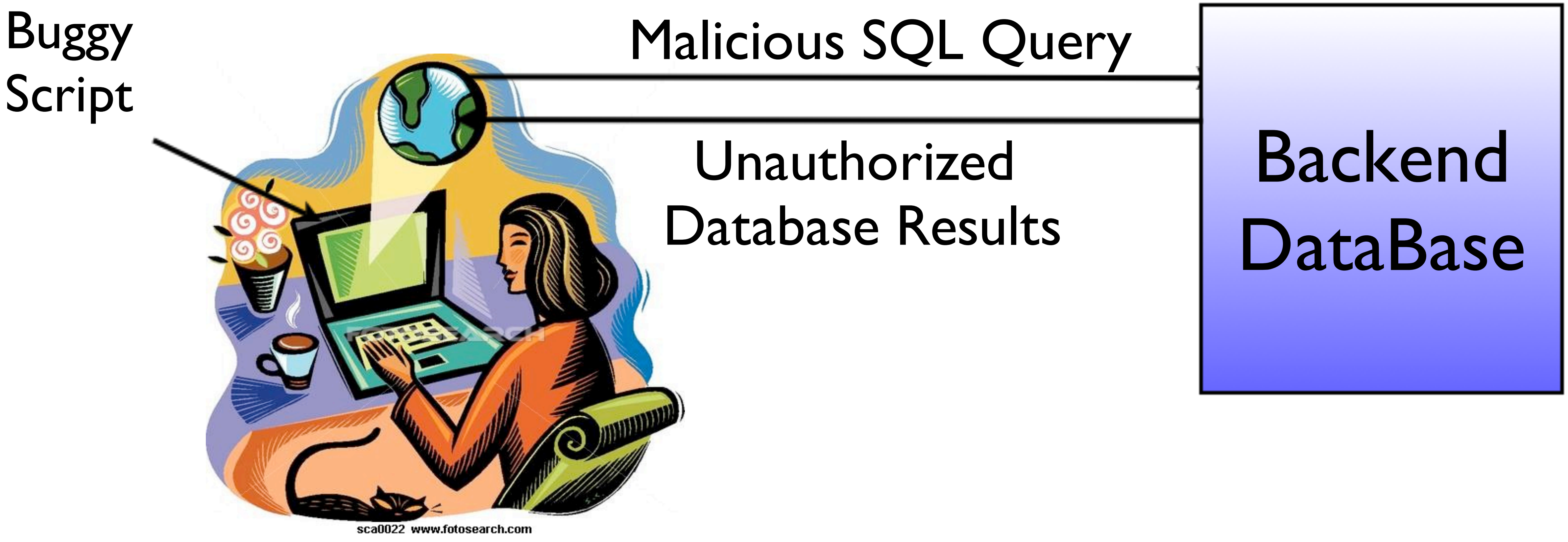
string_expr.” is ”	concat(string_expr, “ is “);	Concat Function
substr(string_expr,1,3)	string_expr[1:3]	Extract Function

assignments/strcmp
a = string_expr;
a /= string_expr;
	equality
	a = string_expr;
	a /= string_expr;
		Equality Predicate

Sanity check in regular expression RE Sanity check in context-free grammar CFG	string_expr in RE string_expr in SQL string_expr NOT in SQL	Membership Predicate
string_expr contains a sub_str string_expr does not contain a sub_str	string_expr contains sub_str string_expr NOT?contains sub_str	Contains Predicate (Substring Predicate)

HAMPI Solver Motivating Example

SQL Injection Vulnerabilities



Buggy Script

Malicious SQL Query

Unauthorized Database Results

Backend DataBase

sca0022 www.fotosearch.com

SELECT m FROM messages WHERE id=’1’ OR 1 = 1

HAMPI Solver Motivating Example

SQL Injection Vulnerabilities

Should be: “^[0-9]+$”

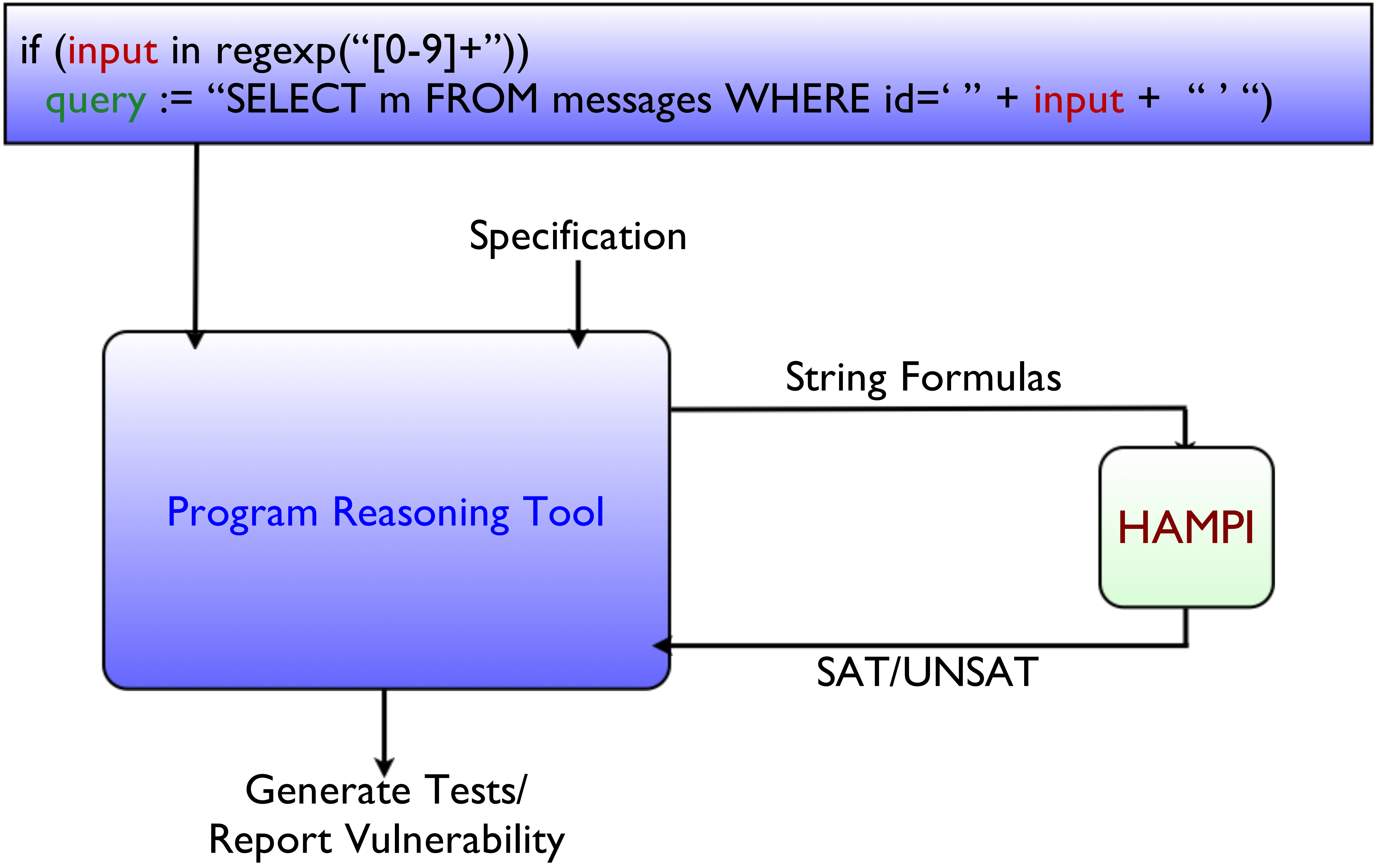
Buggy Script

```
if (input in regexp(“[0-9]+”))
  query := “SELECT m FROM messages WHERE id=‘ ” + input +  “ ’ “)
```

- input passes validation (regular expression check)
- query is syntactically-valid SQL
- query can potentially contain an attack substring (e.g., 1’ OR ‘1’ = ‘1)

HAMPI Solver Motivating Example

SQL Injection Vulnerabilities



```
if (input in regexp(“[0-9]+”))
  query := “SELECT m FROM messages WHERE id=‘ ” + input +  “ ’ “)
```

Expressing the Problem in HAMPI

SQL Injection Vulnerabilities

Input String

```
Var v : 12;
```

SQL Grammar

```
cfg SqlSmall := "SELECT ” [a-z]+ " FROM ” [a-z]+ " WHERE " Cond;

cfg Cond := Val "=" Val | Cond " OR " Cond;

cfg Val := [a-z]+ | "'” [a-z0-9]* "'" | [0-9]+;
```

SQL Query

```
val q := concat("SELECT msg FROM messages WHERE topicid='", v, "'");
```

```
assert v in [0-9]+;
```

SQLI attack conditions

```
assert q in SqlSmall;

assert q contains "OR ‘1'=‘1'";
```

“q is a valid SQL query”

“q contains an attack vector”

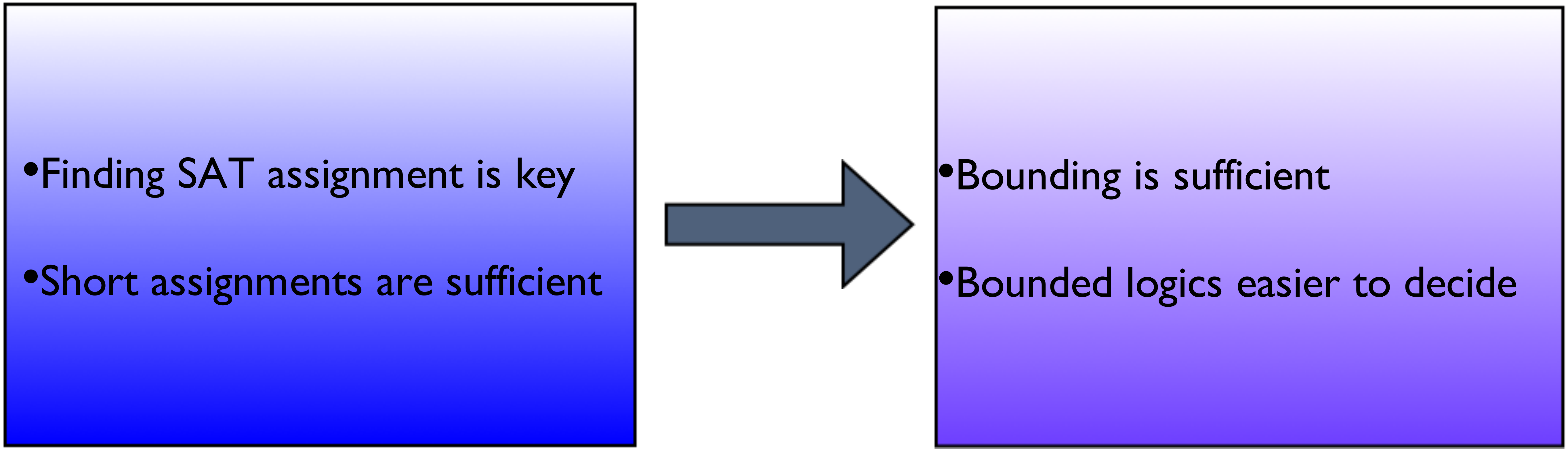
Hampi Key Conceptual Idea

Bounding, expressiveness and efficiency

L_i	Complexity of $\emptyset = L_1 \cap \ldots \cap L_n$	Current Solvers
Context-free	Undecidable	n/a
Regular	PSPACE-complete	Quantified Boolean Logic
Bounded	NP-complete	SAT Efficient in practice

Hampi Key Idea: Bounded Logics

Testing, Analysis, Vulnerability Detection,...



- Finding SAT assignment is key
- Short assignments are sufficient

- Bounding is sufficient
- Bounded logics easier to decide

HAMPI Solver Motivating Example

SQL Injection Vulnerabilities

Input String

```
Var v : 12;
```

SQL Grammar

```
cfg SqlSmall := "SELECT ” [a-z]+ " FROM ” [a-z]+ " WHERE " Cond;

cfg Cond := Val "=" Val | Cond " OR " Cond;

cfg Val := [a-z]+ | "'” [a-z0-9]* "'" | [0-9]+;
```

SQL Query

```
val q := concat("SELECT msg FROM messages WHERE topicid='", v, "'");

assert v in [0-9]+;
```

“q is a valid SQL query”

SQLI attack conditions

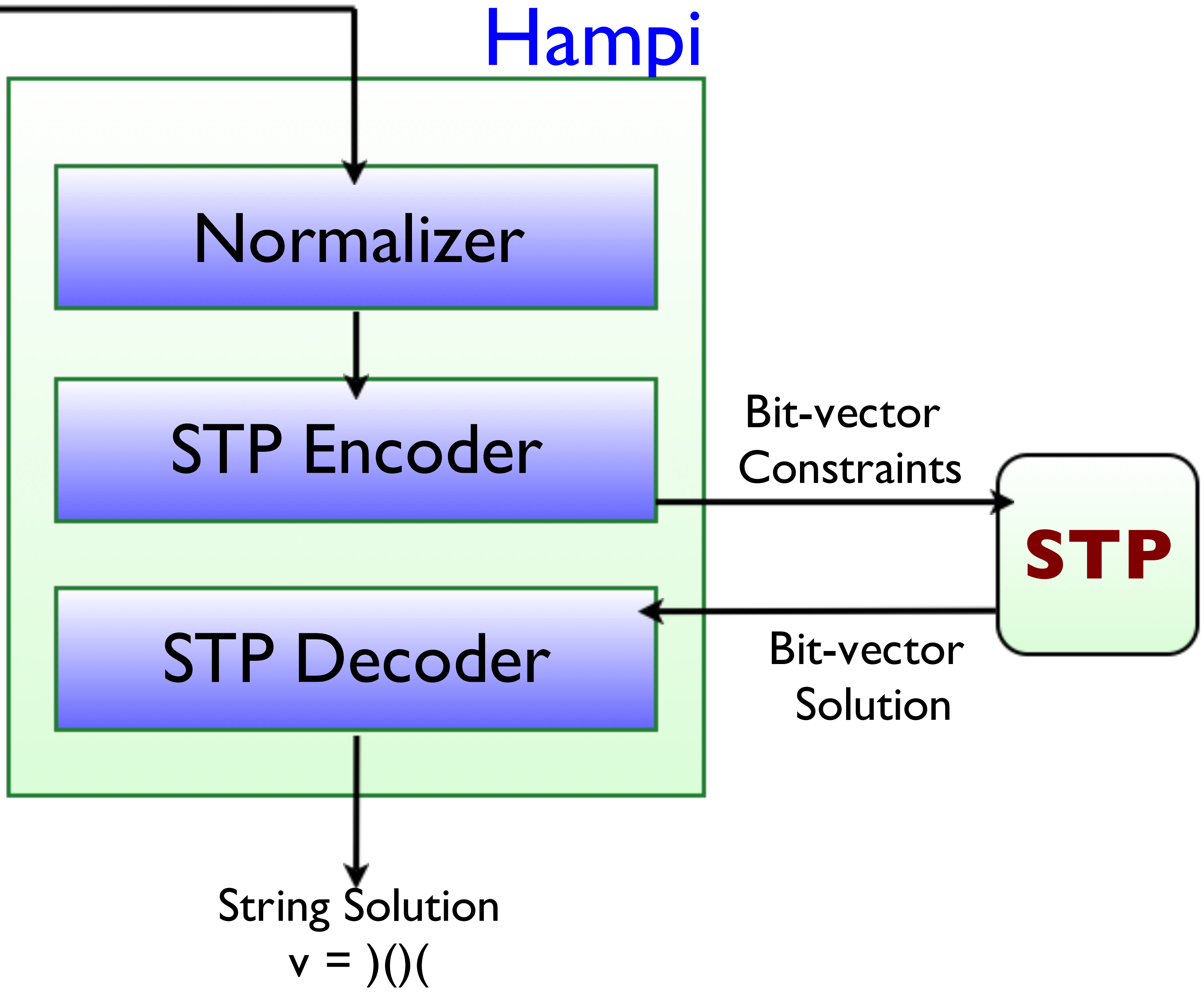
```
assert q in SqlSmall;

assert q contains "OR ‘1'=‘1'";
```

“q contains an attack vector”

How Hampi Works

Bird’s Eye View: Strings into Bit-vectors

```
var v : 4;

cfg E := “()” | E E | “(“ E
“)”;

val q := concat( “(“, v, ”)”);

assert q in E;
assert q contains “()()”;
```



Hampi

Normalizer

STP Encoder

STP Decoder

Bit-vector Constraints

STP

Bit-vector Solution

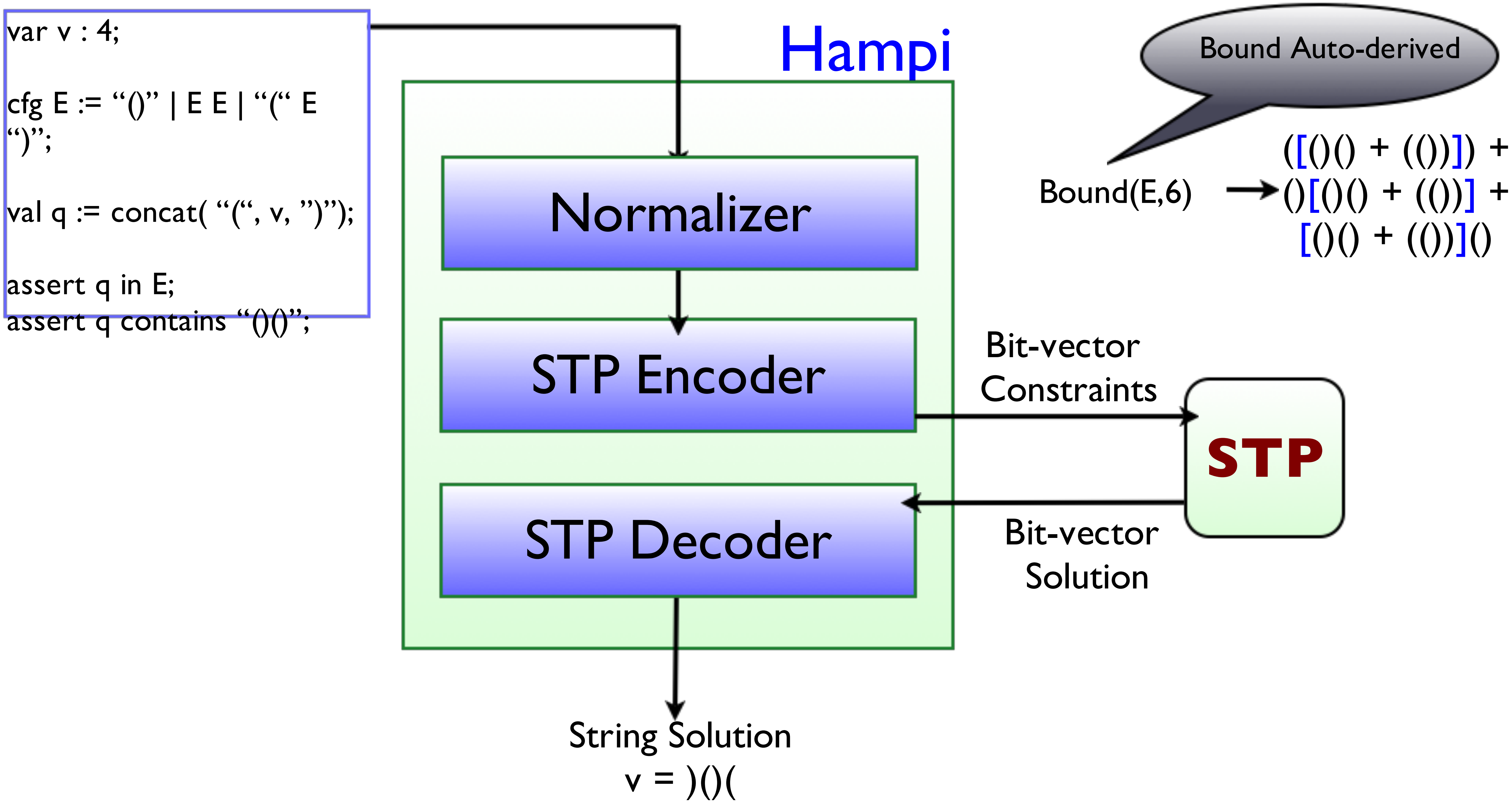
String Solution
v =)()(

Find a 4-char string v:

- (v) is in E
- (v) contains ()()

How Hampi Works

Unroll Bounded CFGs into Regular Exp.

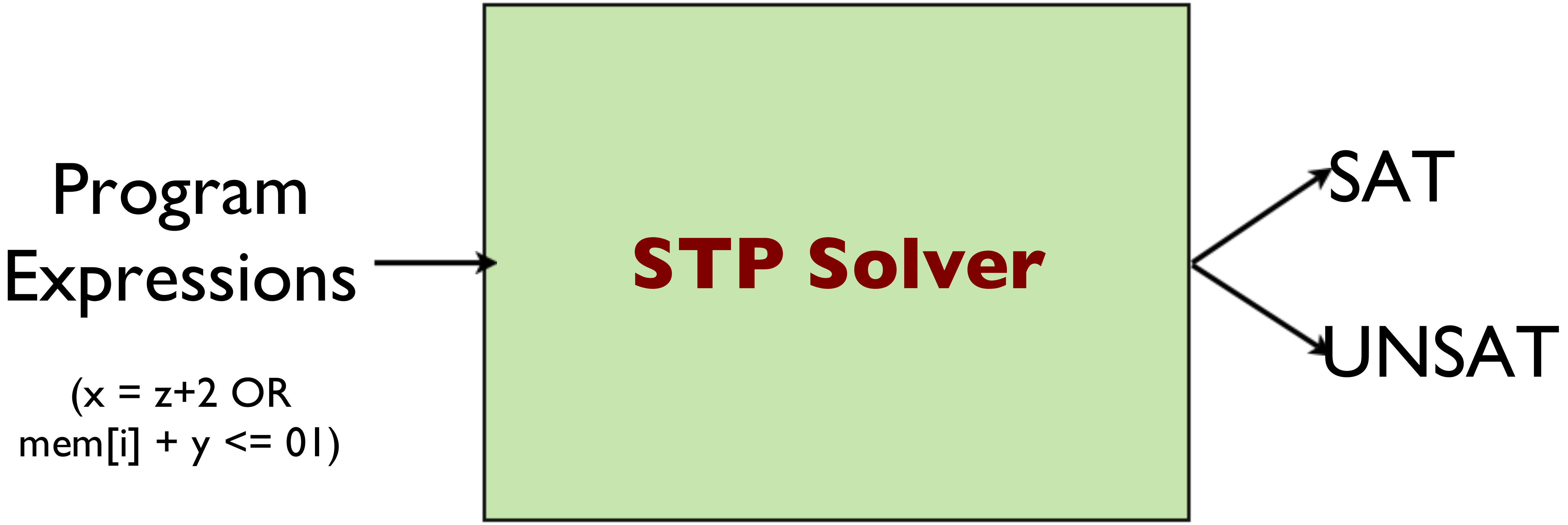


```
var v : 4;

cfg E := “()” | E E | “(“ E
“)”;

val q := concat( “(“, v, ”)”);

assert q in E;
assert q contains “()()”;
```

Bound(E,6) → ([()() + (())]) +
()[()() + (())] +
[()() + (())]()

String Solution
v =)()(

STP Bit-vector & Array Solver



Program Expressions

(x = z+2 OR mem[i] + y <= 01)

STP Solver

SAT

UNSAT

- Bit-vector or machine arithmetic
- Arrays for memory
- C/C++/Java expressions
- NP-complete

Impact of STP: Notable Projects

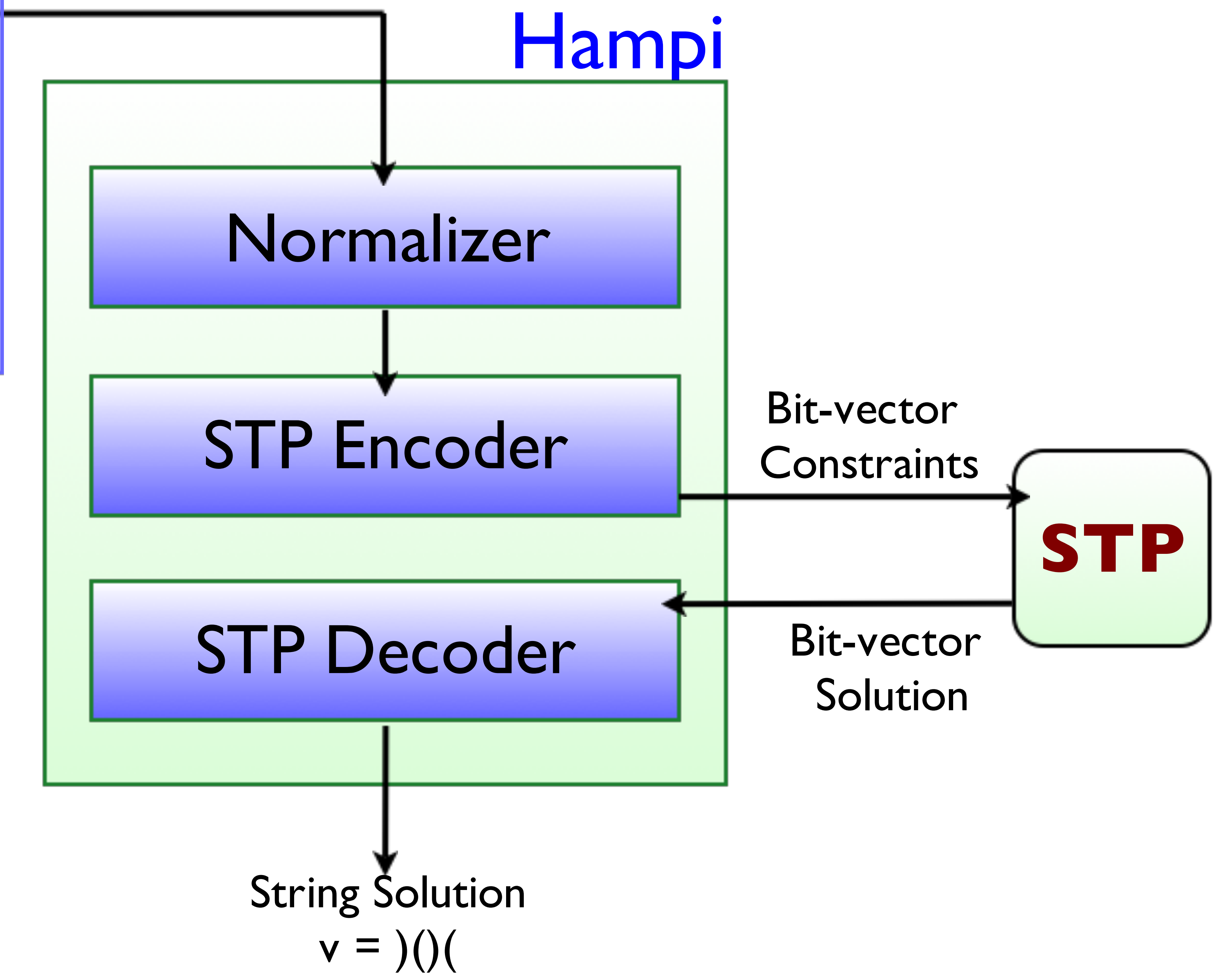
- Played an important role in the development of symbolic testing techniques
- 100+ reliability and security projects

Category	Research Project	Project Leader/Institution
Formal Methods	ACL2 Theorem Prover + STP Verification-aware Design Checker Java PathFinder Model Checker	Eric Smith & David Dill/Stanford Jacob Chang & David Dill/Stanford Mehlitz & Pasareanu/NASA
Program Analysis	BitBlaze & WebBlaze BAP	Dawn Song et al./Berkeley David Brumley/CMU
Automatic Testing Security	Klee, EXE SmartFuzz Kudzu S2E & Cloud9	Engler & Cadar/Stanford Molnar & Wagner/Berkeley Saxena & Song/Berkeley Bucur & Candea/EPFL
Hardware Bounded Model-cheking (BMC)	Blue-spec BMC BMC	Katelman & Dave/MIT Haimed/NVIDIA

How Hampi Works

Bird’s Eye View: Strings into Bit-vectors

```
var v : 4;

cfg E := “()” | E E | “(“ E
“)”;

val q := concat( “(“, v, ”)”);

assert q in E;
assert q contains “()()”;
```

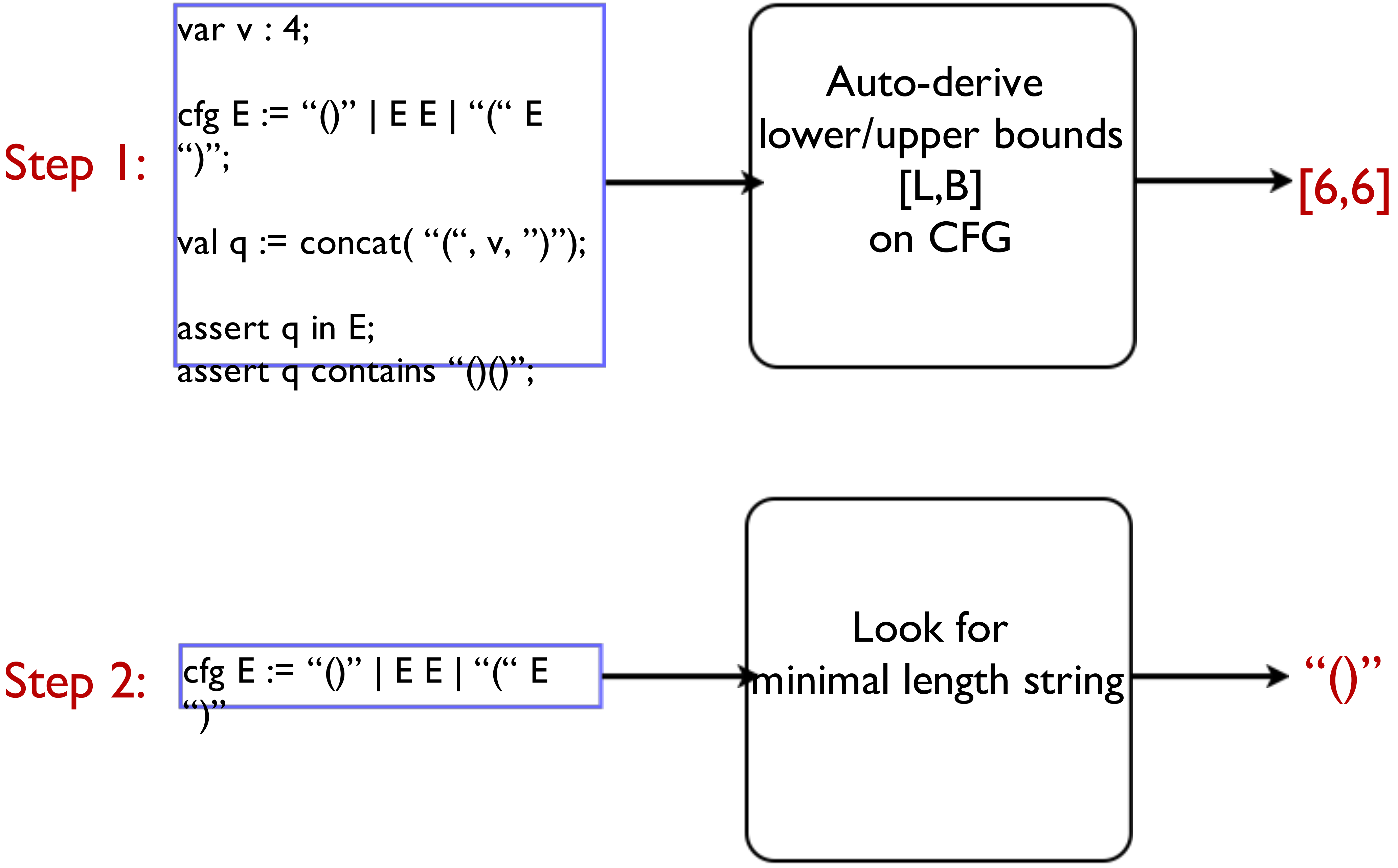


Find a 4-char string v:

- (v) is in E
- (v) contains ()()

How Hampi Works

Unroll Bounded CFGs into Regular Exp.



Step 1:

```
var v : 4;

cfg E := "()" | E E | "(" E
")";

val q := concat( "(", v, ")");

assert q in E;
assert q contains "()()";
```

Auto-derive
lower/upper bounds
[L,B]
on CFG

[6,6]

Step 2:

```
cfg E := "()" | E E | "(" E
")"
```

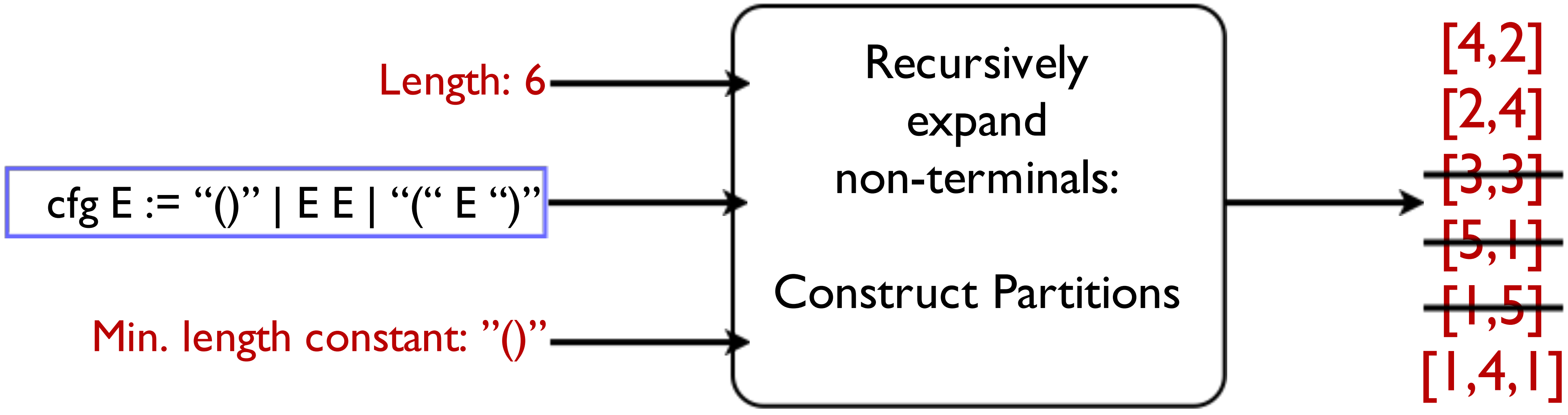
Look for
minimal length string

"()"

How Hampi Works

Unroll Bounded CFGs into Regular Exp.

Step 3:



Length: 6

cfg E := “()” | E E | “(“ E “)”

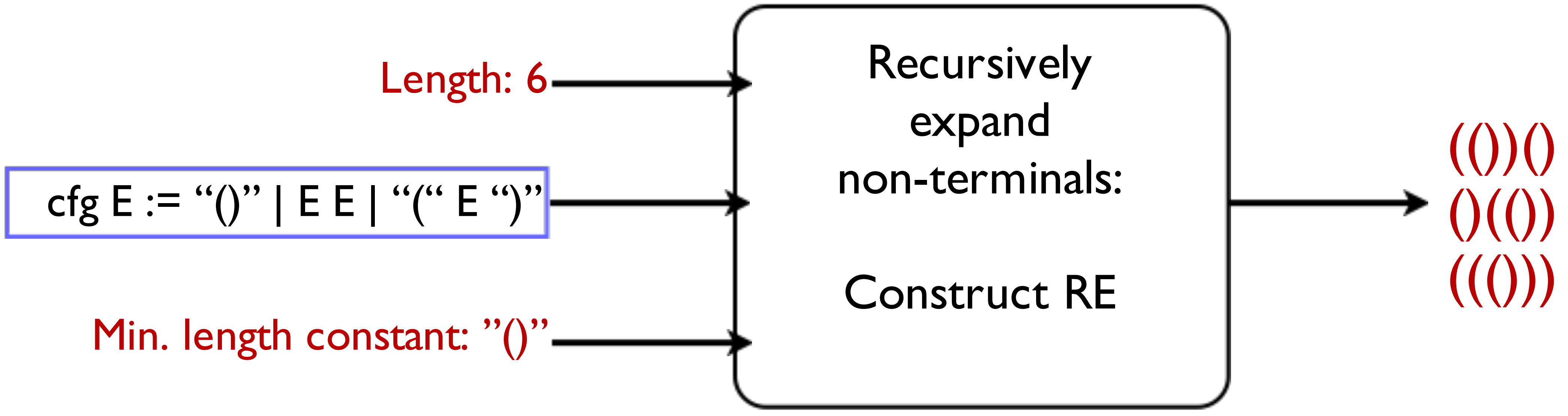
Min. length constant: ”()”

Recursively expand non-terminals:

Construct Partitions

[4,2]
[2,4]
~~[3,3]~~
~~[5,1]~~
~~[1,5]~~
[1,4,1]

Step 4:



Length: 6

cfg E := “()” | E E | “(“ E “)”

Min. length constant: ”()”

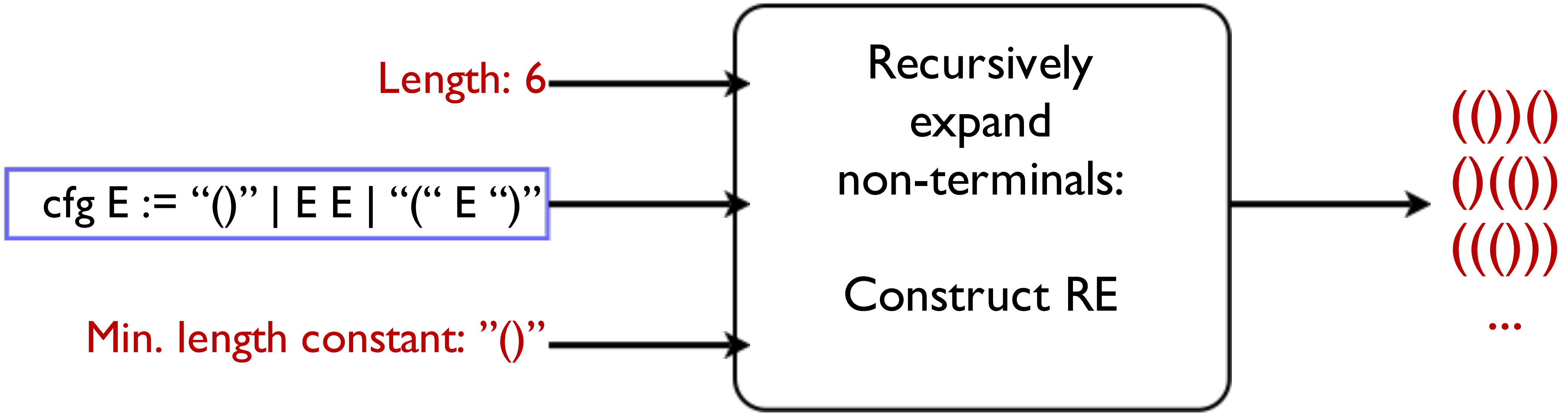
Recursively expand non-terminals:

Construct RE

(())()
()(())
((()))

Unroll Bounded CFGs into Regular Exp.

Managing Exponential Blow-up



Length: 6

cfg E := “()” | E E | “(“ E “)”

Min. length constant: ”()”

Recursively expand non-terminals:

Construct RE

(())()
()(())
((()))
...

- Memoize common sub-expressions
- Lots of redundant sub-expression in commonly occurring regular expressions
- Works well in practice

How Hampi Works

Converting Regular Exp. into Bit-vectors

Encode regular expressions recursively

- Alphabet { (,) } → 0, 1
- constant → bit-vector constant
- union + → disjunction $\vee$
- concatenation → conjunction $\wedge$
- Kleene star * → conjunction $\wedge$
- Membership, equality → equality

$$\boxed{(\ v\ ) \in ()[()() + (())] + [()() + (())]() + ([()() + (())])}$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

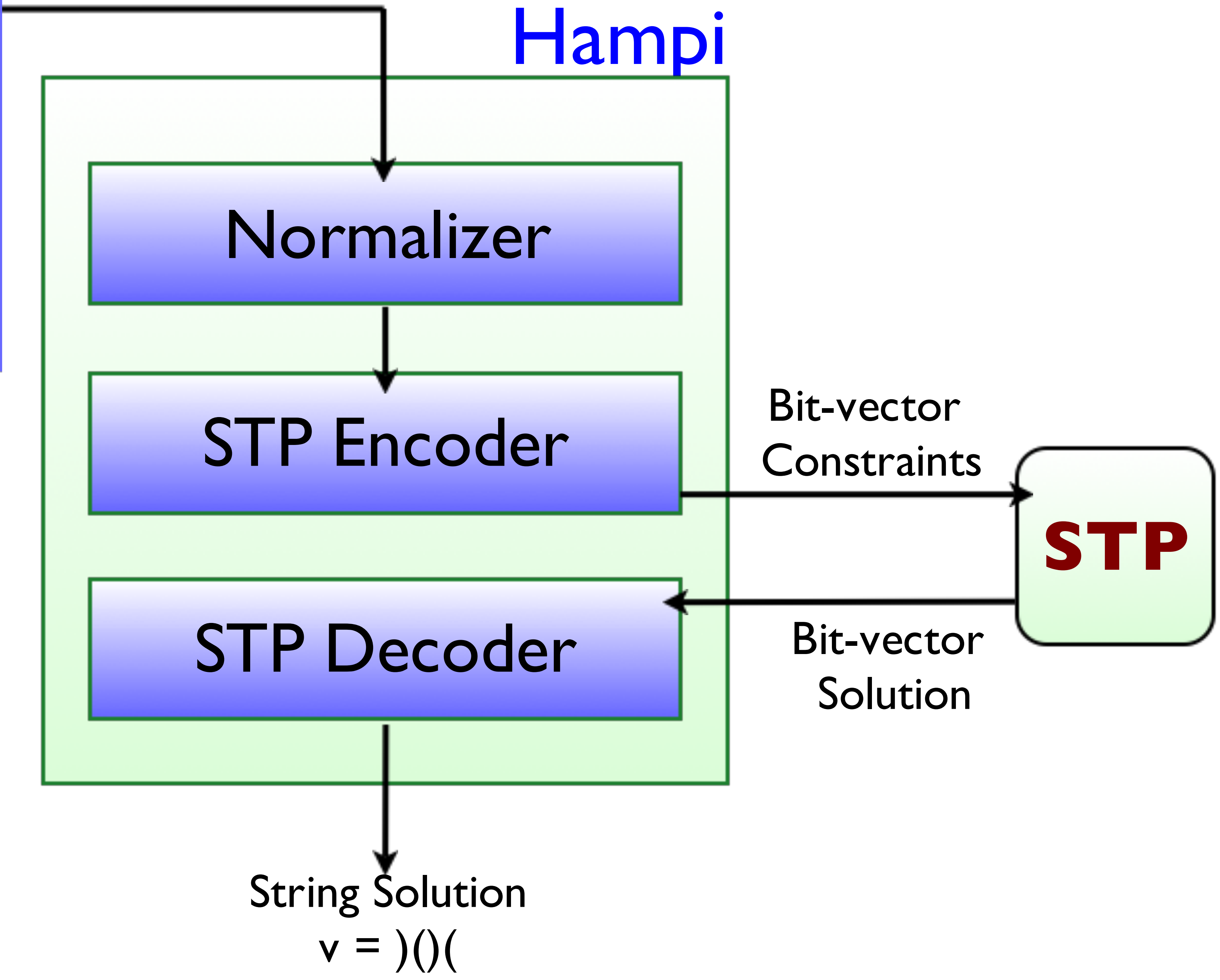
$$\text{Formula } \Phi_1 \quad \vee \quad \text{Formula } \Phi_2 \quad \vee \quad \text{Formula } \Phi_3$$

$$B[0]=0 \wedge B[1]=1 \wedge \{B[2]=0 \wedge B[3]=1 \wedge B[4]=0 \wedge B[5]=1 \ \vee \ldots$$

How Hampi Works

Decoder converts Bit-vectors to Strings

```
var v : 4;

cfg E := "()" | E E | "(" E
")";

val q := concat( "(", v, ")");

assert q in E;
assert q contains "()()";
```

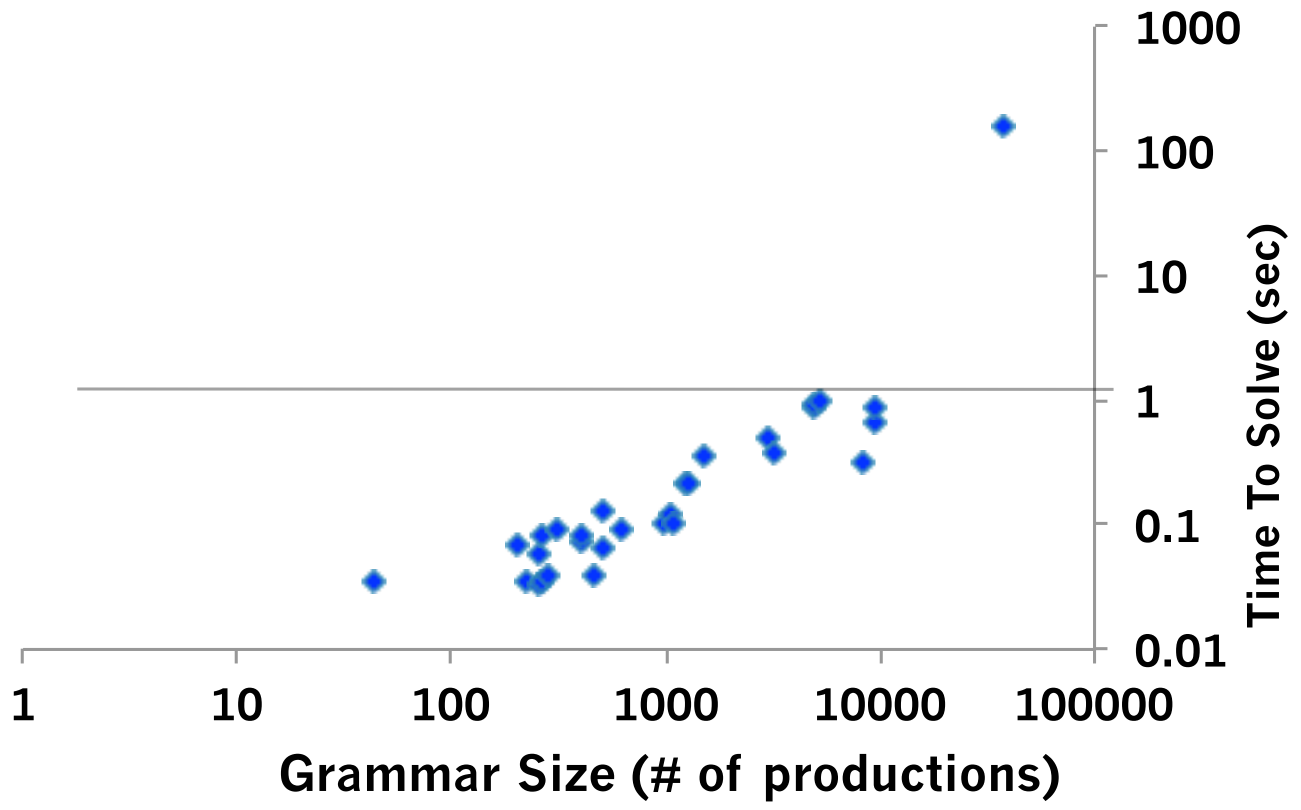


Find a 4-char string v:

- (v) is in E
- (v) contains ()()

HAMPI: Result 1

Static SQL Injection Analysis



- 1367 string constraints from Wasserman & Su [PLDI'07]
- Hampi scales to large grammars
- Hampi solved 99.7% of constraints in < 1sec
- All solvable constraints had short solutions

HAMPI: Result 2

Security Testing and XSS

- Attackers inject client-side script into web pages
- Somehow circumvent same-origin policy in websites
- echo “Thank you $my_poster for using the message board”;
- Unsanitized $my_poster
- Can be JavaScript
- Execution can be bad

HAMPI: Result 2

Security Testing

- Hampi used to build Ardilla security tester [Kiezun et al., ICSE’09]

- 60 new vulnerabilities on 5 PHP applications (300+ kLOC)
  - 23 SQL injection
  - 37 cross-site scripting (XSS)

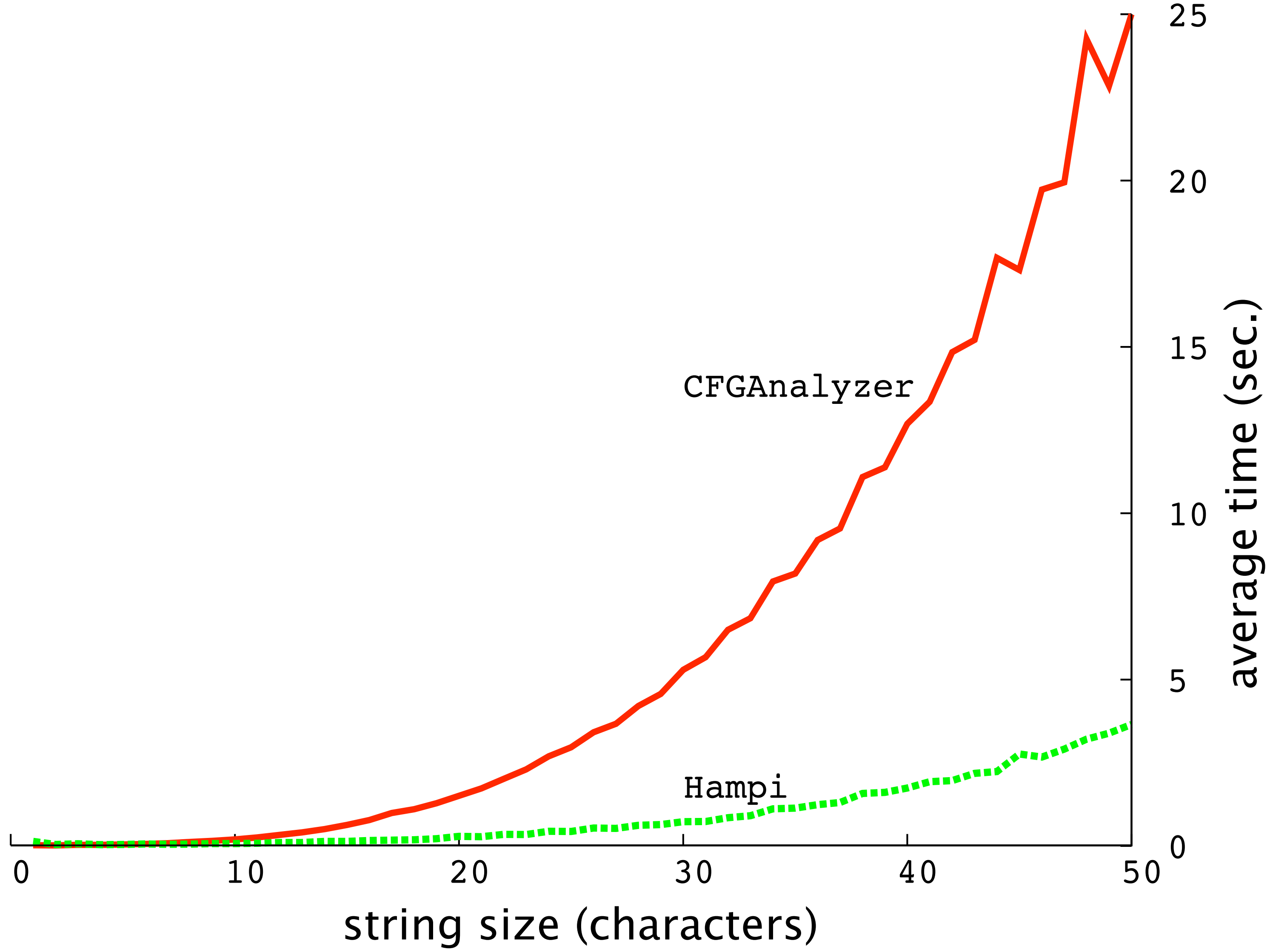
5 added to
US National Vulnerability DB

- 46% of constraints solved in < 1 second per constraint

- 100% of constraints solved in <10 seconds per constraint

HAMPI: Result 3

Comparison with Competing Tools

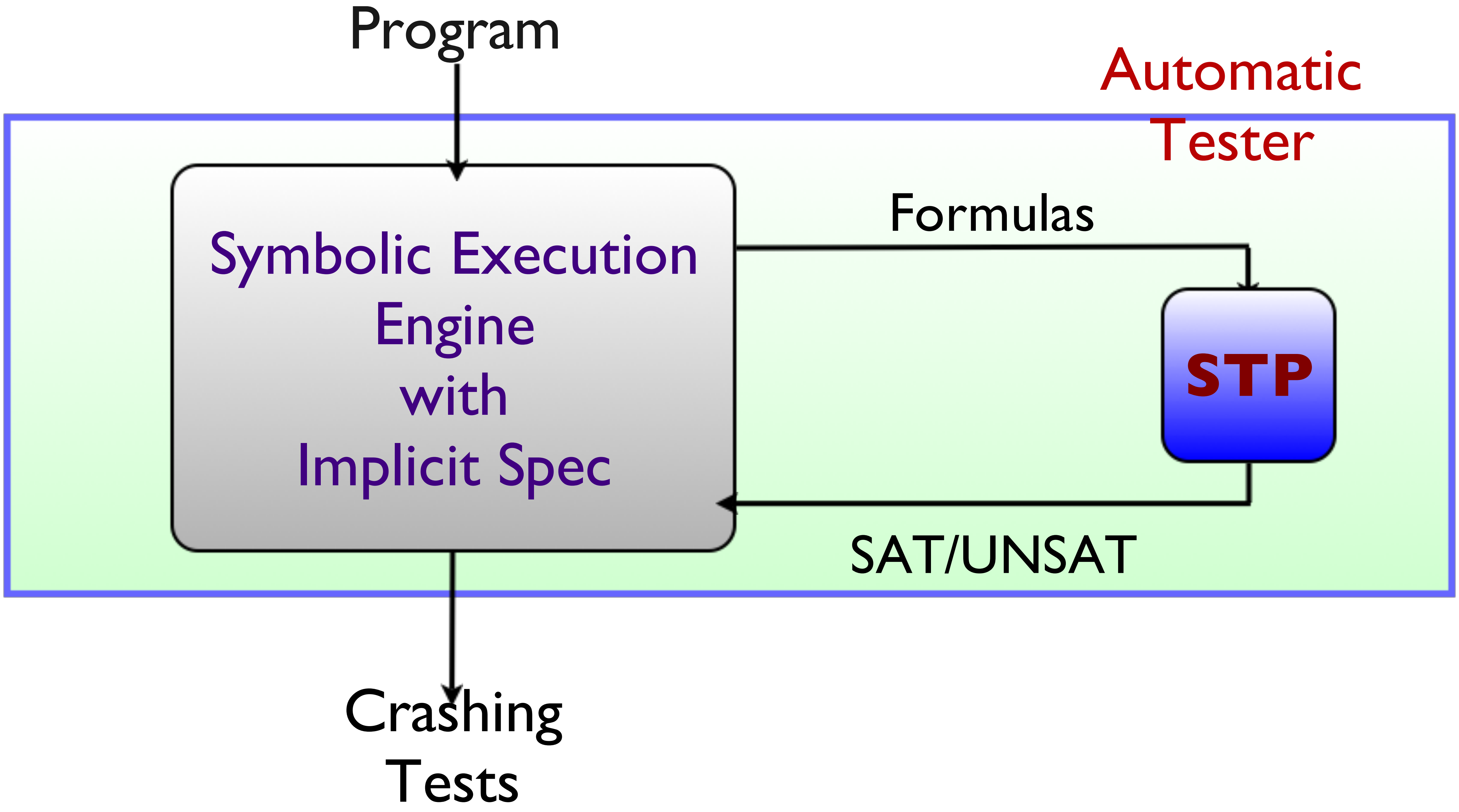


- HAMPI vs. CFGAnalyzer (U. Munich): HAMPI ~7x faster for strings of size 50+
- HAMPI vs. Rex (Microsoft Research): HAMPI ~100x faster for strings of size 100+
- HAMPI vs. DPRLE (U. Virginia): HAMPI ~1000x faster for strings of size 100+

How to Automatically Crash Programs?

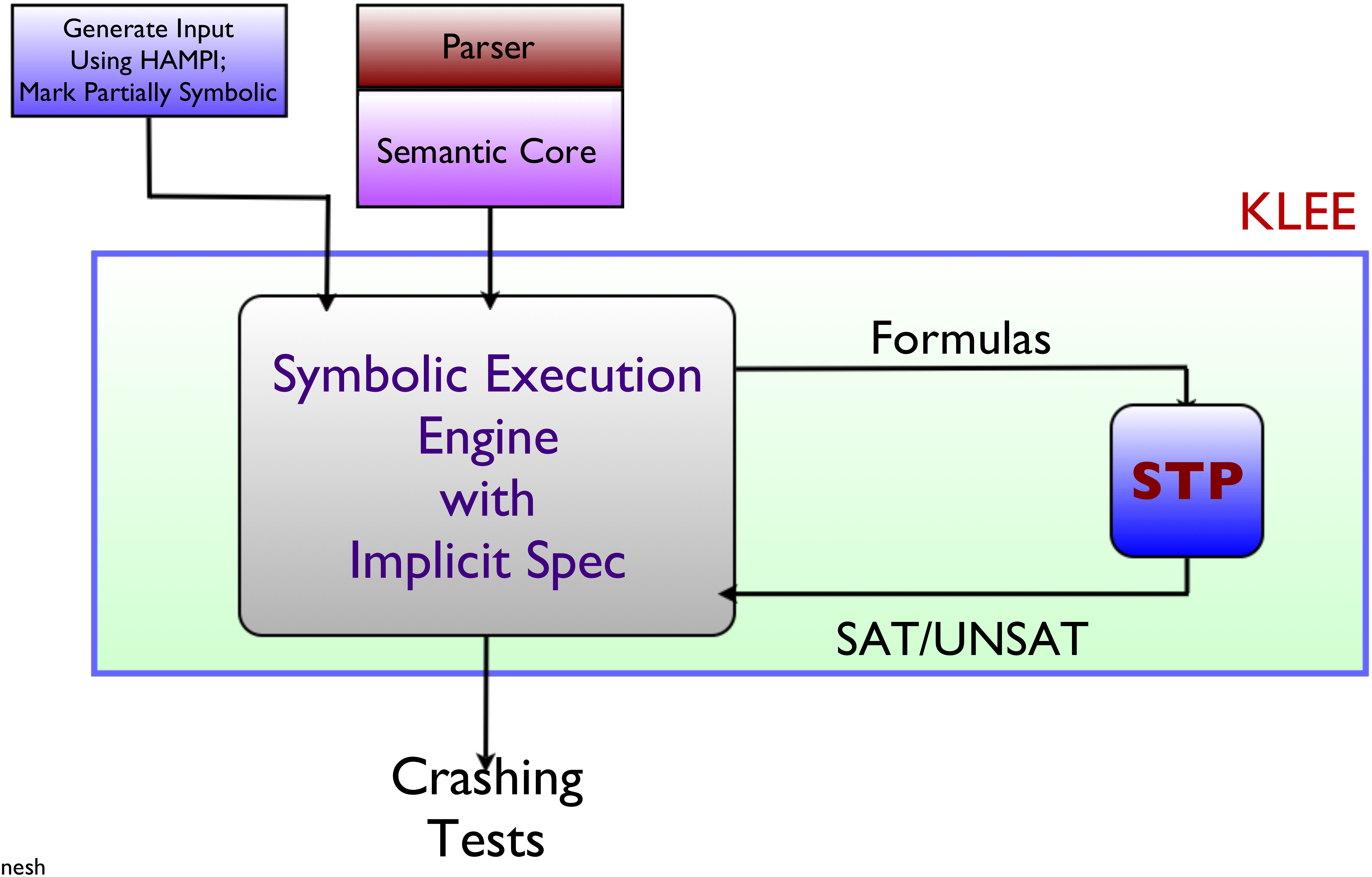
KLEE: Concolic Execution-based Tester

Problem: Automatically generate crashing tests given only the code



HAMPI: Result 4

Helping KLEE Pierce Parsers



Impact of Hampi: Notable Projects

Category	Research Project	Project Leader/Institution
Static Analysis	SQL-injection vulnerabilities	Wasserman & Su/UC, Davis
Security Testing	Ardilla for PHP (SQL injections, cross-site scripting)	Kiezun & Ernst/MIT
Concolic Testing	Klee Kudzu NoTamper	Engler & Cadar/Stanford Saxena & Song/Berkeley Bisht & Venkatakrishnan/U Chicago
New Solvers	Kaluza	Saxena & Song/Berkeley

Rest of the Talk

- HAMPI string solver
  - String equations and membership in regular expressions/CFGs
  - How HAMPI works
  - Experimental results
- Theoretical results
  - Undecidability of forallexists fragment of word equations
  - Conditional decidability results for word equations and length
  - Open theoretical problems
- Z3-str
  - A solver for string equations and length function

Theory of Word Equations, Length and Membership

Symbol	String Sort	Number Sort
Constants	Finite-length strings defined over a finite alphabet Σ	Integers
Variables	Range over Σ*	Range over integers
String functions	Concat: String × String ⇒ String Length: String ⇒ Integer	
Integer functions		Addition: Integer × Integer ⇒ Integer
String predicates	Equality over string terms (= : String × String ⇒ Bool) membership in regular expressions/CFGs (∈: String × regular-expression ⇒ Bool)	
Integer predicates		Equality over integer terms (=: Integer × Integer ⇒ Bool) Inequality over integer terms (<: Integer × Integer ⇒ Bool)

Word Equations, Reg Exp, and Length
What is known

Result	**Person (Year)**	**Notes**
Undecidability of Quantified Word Equations	Quine (1946)	Multiplication reduced to concat
Undecidability of Quantified Word Equations with single alternation	Durnev (1996), G. (2012)	2-counter machines reduced to words with single quantifier alter.
Decidability (PSPACE) of QF Theory of Word Equations	Makanin (1977) Plandowski (1996, 2002/06)	Makanin result very difficult Simplified by Plandowski
Decidability (PSPACE-complete) of QF Theory of Word Equations + RE	Schultz (1992)	RE membership predicate
QF word equations + Length() (?)	Matiyasevich (1971)	Unsolved

Theory of Word Equations and Length

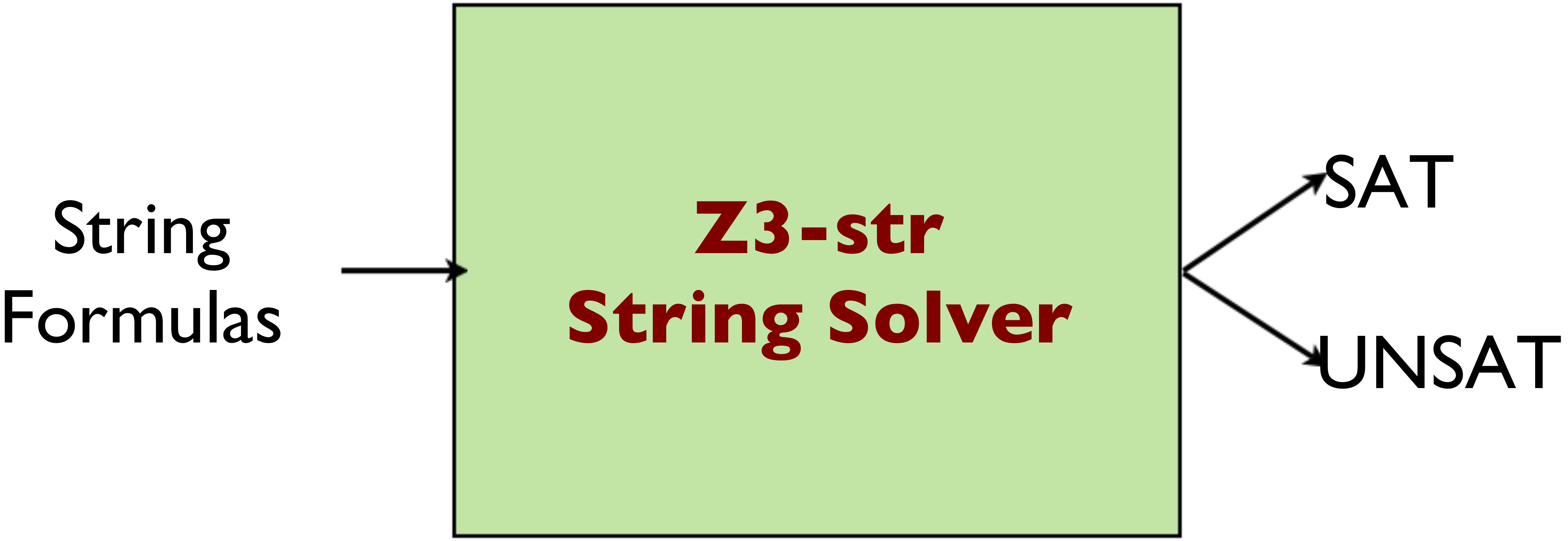
Our Results (HVC 2012)

Decidability/Undecidability	**Result**
Undecidability	• Theorem: The forall-exists fragment of quantifier-free word equations in undecidable. Proof Sketch: • Reduction from the halting problem for 2-counter machines to SAT problem for forall-exists fragment of word equations • Intuition is to encode computational histories of 2-counter machines into strings
Conditional decidability	• Theorem: The quantifier-free theory of word equations and length is decidable, if word equations can be converted into solved form • Theorem: The quantifier-free theory of word equations with length and regular expressions membership is decidable, if word equations can be converted into solved form

Rest of the Talk

- HAMPI string solver
  - String equations and membership in regular expressions/CFGs
  - How HAMPI works
  - Experimental results
- Theoretical results
  - Undecidability of forallexists fragment of word equations
  - Conditional decidability results for word equations and length
  - Open theoretical problems
- Z3-str
  - A solver for string equations and length function

Z3-str String Solver*



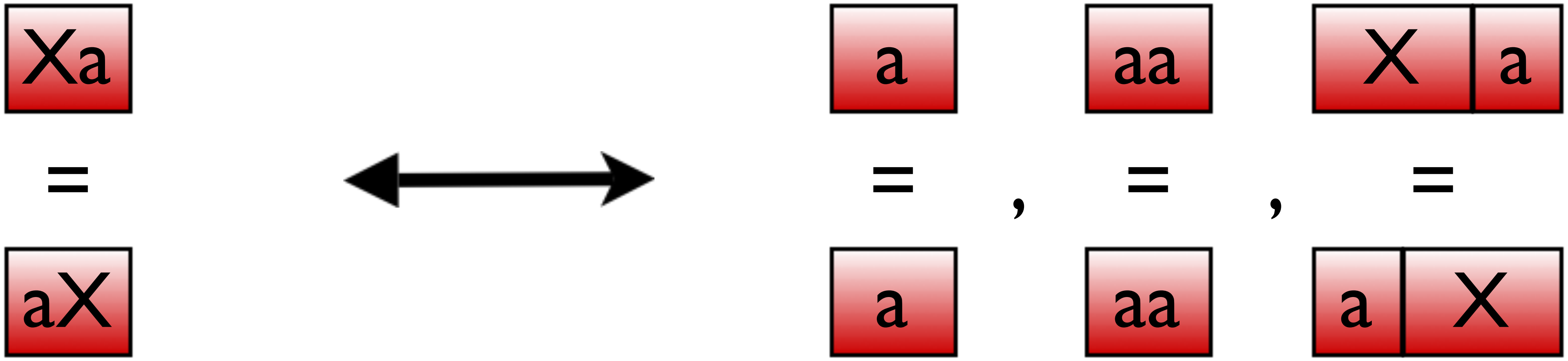
- Quantifier-free theory of word equations and length function
- Status: unknown
- Our partial decidability technique
  - Given a word equation partition its solutions space into finite buckets
  - Leverage Z3 for identifying equivalent expressions and length consistency checks
  - Approximate by heuristically solving "overlapping" equations

*** Joint work with Xiangyu Zhang and Yunhui Zheng (Purdue University)**

Z3-str String Solver*

- Quantifier-free theory of word equations and length function
- Status: unknown
- Our partial decidability technique
  - Given a word equation partition its solutions space into finite buckets
  - Leverage Z3 for identifying equivalent expressions and length consistency checks
  - Approximate by heuristically solving “overlapping” equations

* Joint work with Xiangyu Zhang and Yunhui Zheng (Purdue University)



Related Work (Practice)

Tool Name	Project Leader/Institution	Comparison with HAMPI
Rex	Bjorner, Tillman, Vornkov et al. (Microsoft Research, Redmond)	• HAMPI + Length+Replace(s_1,s_2,s_3) - CFG • Translation to int. linear arith. (Z3)
Mona	Karlund et al. (U. of Aarhus)	• Can encode HAMPI & Rex • User work • Automata-based • Non-elementary
DPRLE	Hooimeijer (U. of Virginia)	• Regular expression constraints

Some Future Directions

- A Predictive Theory for CDCL SAT solvers and DPLL(T)?
- Attack-resistance programs
  - Can we define a mathematical notion of partial reliability?
- Expanding the scope of testing
  - Automatic counter-example construction for math conjectures
- Open problems regarding theories of strings
  - Is the SAT problem for word equations in NP? Is the quantifier-free theory of word equations and length decidable?
- Richer string solvers
  - All-in-one: integrating word equations, length and membership into SMT

Key Contributions

https://ece.uwaterloo.ca/~vganesh

Name	**Key Concept**	**Impact**	**Pubs**
STP Bit-vector & Array Solver[1,2]	Abstraction-refinement for Solving	Concolic Testing	CAV 2007 CCS 2006 TISSEC 2008
HAMPI String Solver[1]	App-driven Bounding for Solving	Analysis of Web Apps	ISSTA 2009[3] TOSEM 2012 CAV 2011
(Un)Decidability results for Strings	Reduction from two-counter machine halting problem		HVC 2012
Taint-based Fuzzing	Information flow is cheaper than concolic	Scales better than concolic	ICSE 2009
Automatic Input Rectification	Acceptability Envelope: Fix the input, not the program	New way of approaching SE	ICSE 2012

1. **STP won the SMTCOMP 2006 and 2010 competitions for bit-vector solvers**
2. **HAMPI: ACM Best Paper Award 2009**
3. **Google Award 2011**
4. Retargetable Compiler (DATE 1999)
5. Proof-producing decision procedures (TACAS 2003)
6. Error-finding in ARBAC policies (CCS 2011)
7. Programmatic SAT Solvers (SAT 2012)