



Automata-based Model Counting for String Constraints

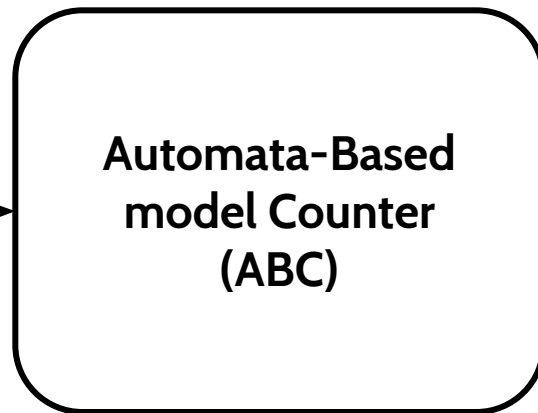
Abdulkaki Aydin, Lucas Bang, Tevfik Bultan
<https://vlab.cs.ucsb.edu>

Model Counting for String Constraints

INPUT

string
constraint:

C



OUTPUT

counting
function:

f_c

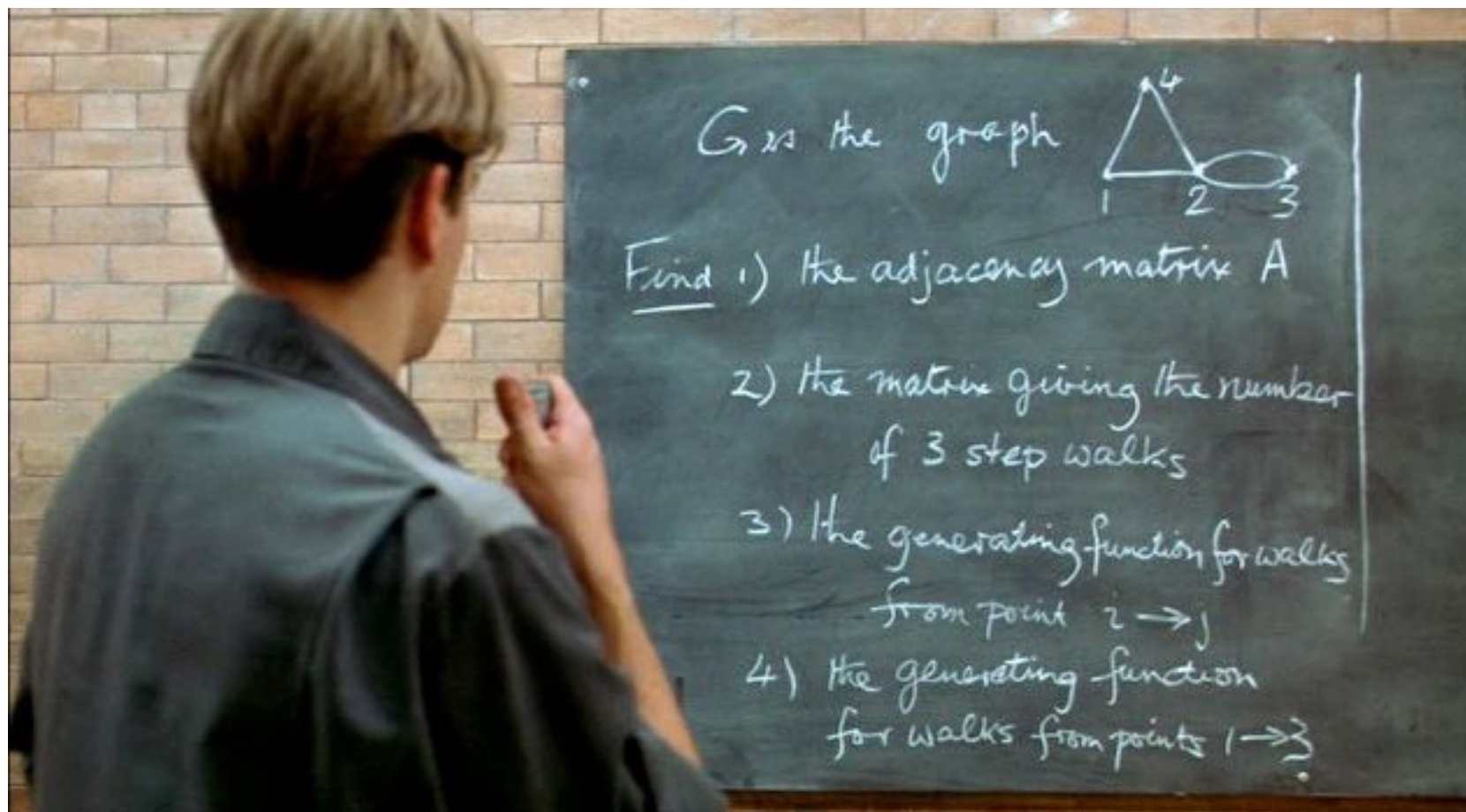


length bound: k



of strings with length $\leq k$
for which C evaluates to true

Can you solve it, Will Hunting?



Motivation

- ▶ **Why care about string constraints?**
 - ▶ String constraint solvers are essential for program analysis
 - ▶ Symbolic execution [Saxena et al., S&P'10]
 - ▶ Symbolic verification [Alkhalaf et al., ICSE'12]

- ▶ **Why care about model counting?**
 - ▶ Quantitative information flow analysis
 - ▶ [Clark et al., JCS'07] [McCamant et al., PLDI'08] [Phan et al., SEN'12] [Smith et al., FOSSACS'09]
 - ▶ Probabilistic symbolic execution
 - ▶ [Borges et al., PLDI'14] [Flieri et al., ICSE'13]

Outline

- ▶ **Automata-based Constraint Solver**
 - ▶ String constraint language
 - ▶ Constraint types
 - ▶ Automaton construction
- ▶ **Automata-based Model Counting**
 - ▶ Generating functions
 - ▶ Recurrences
- ▶ **Experimental Results**
- ▶ **Conclusion**

String Constraint Language

$$F \rightarrow C \mid \neg F \mid F \wedge F \mid F \vee F \quad (1)$$

$$C \rightarrow S \in R \quad (2)$$

$$\quad \mid S = S \quad (3)$$

$$\quad \mid S = S . S \quad (4)$$

$$\quad \mid \text{LEN}(S) \text{ O } n \quad (5)$$

$$\quad \mid \text{LEN}(S) \text{ O } \text{LEN}(S) \quad (6)$$

$$\quad \mid \text{CONTAINS}(S, s) \quad (7)$$

$$\quad \mid \text{BEGINS}(S, s) \quad (8)$$

$$\quad \mid \text{ENDS}(S, s) \quad (9)$$

$$\quad \mid n = \text{INDEXOF}(S, s) \quad (10)$$

$$\quad \mid S = \text{REPLACE}(S, s, s) \quad (11)$$

$$S \rightarrow v \mid s \quad (12)$$

$$R \rightarrow s \mid \varepsilon \mid R R \mid R|R \mid R^* \quad (13)$$

$$O \rightarrow = \mid < \mid > \mid \leq \mid \geq \quad (14)$$

Atomic
constraints

Constraint Types

▶ Single variable constraints

$$F \equiv \neg(x \in (01)^*) \wedge LEN(x) \geq 1$$

▶ Pseudo-relational constraints

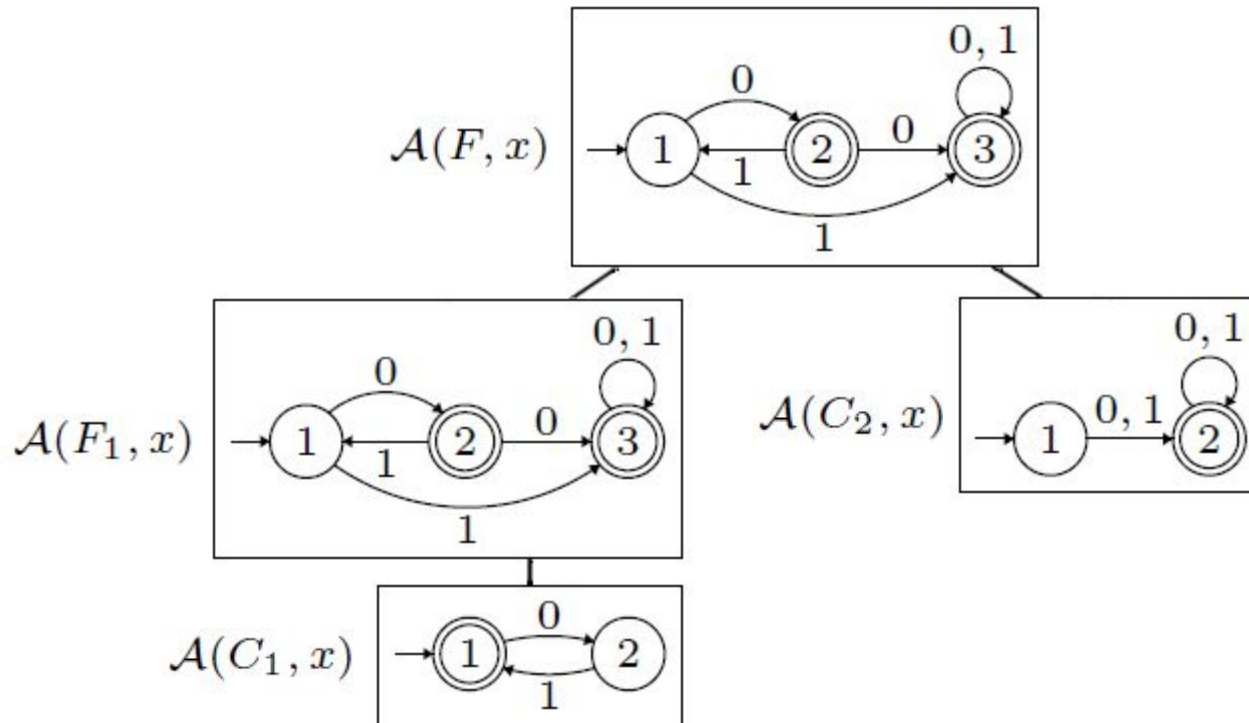
$$(x = y.z) \wedge (LEN(y) = 0) \wedge \neg(z \in (0|1)) \wedge \\ (x = t) \wedge \neg(t \in 0^*)$$

▶ Relational constraints

$$(x = y.z) \wedge \neg(x = y.t) \wedge (CONTAINS\ t\ z) \\ \wedge (LEN(y) > 5) \wedge \neg(z \in (0|1)) \wedge \neg(t \in 0^*)$$

Automata Construction

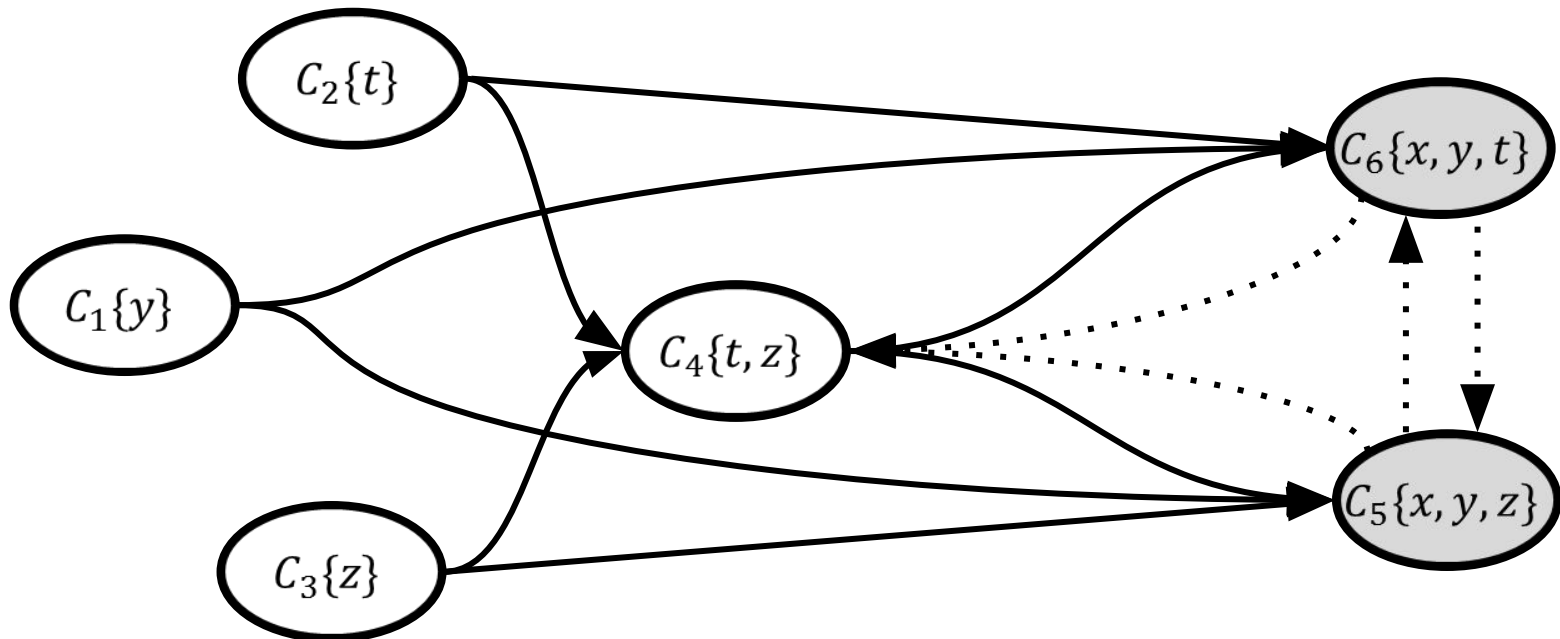
$$F \equiv \neg(x \in (01)^*) \wedge \text{LEN}(x) \geq 1$$



Automata Construction (Multi-variable)

- ▶ Pick a projection variable
- ▶ Generate a dependency graph between atomic constraints based on the variables that appear in them
- ▶ Solve constraints in topological order
- ▶ Cycles require iteration (which may not converge) leading to approximation

$(x = y.z) \wedge \neg(x = y.t) \wedge (\text{CONTAINS } t \ z) \wedge (\text{LEN}(y) > 5) \wedge \neg(z \in (0|1)) \wedge \neg(t \in 0^*)$

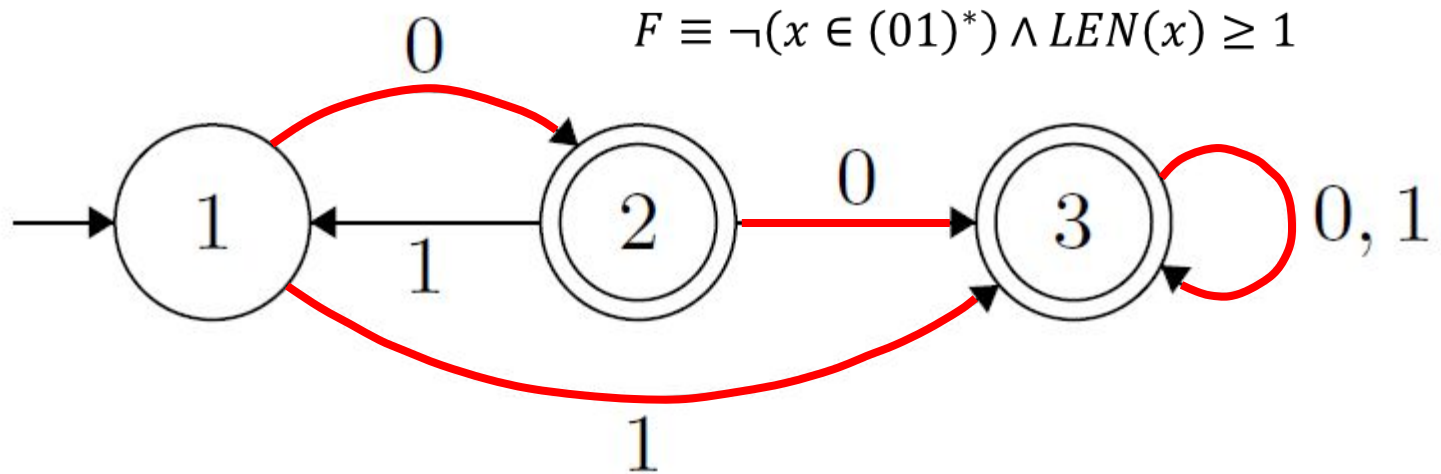


Outline

- ▶ Automata-based Constraint Solver
 - ▶ String constraint language
 - ▶ Constraint types
 - ▶ Automaton construction
- ▶ Automata-based Model Counting
 - ▶ Generating functions
 - ▶ Recurrences
- ▶ Experimental Results
- ▶ Conclusion

Automata-based Model Counting

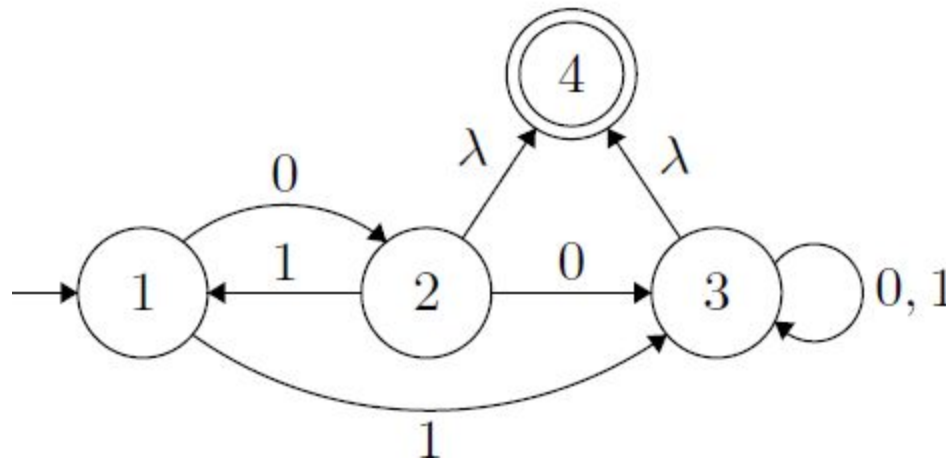
- ▶ Converting string constraints to automata reduces the model counting problem to path counting problem in graphs



- ▶ We will generate a function $f(k)$
 - ▶ Given length bound k , it will count the number of strings with length k .
 - ▶ $f(0) = 0, \{\}$
 - ▶ $f(1) = 2, \{0,1\}$
 - ▶ $f(2) = 3, \{00,10,11\}$

Path Counting

► $F = \neg(x \in (01)^*) \wedge LEN(x) \geq 1$



$$T = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, T^2 = \begin{bmatrix} 1 & 0 & 3 & 2 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix}, T^3 = \begin{bmatrix} 0 & 1 & 7 & 3 \\ 1 & 0 & 7 & 4 \\ 0 & 0 & 8 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}, T^4 = \begin{bmatrix} 0 & 1 & 15 & 8 \\ 1 & 0 & 15 & 7 \\ 0 & 0 & 16 & 8 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$f(0) = 0$$

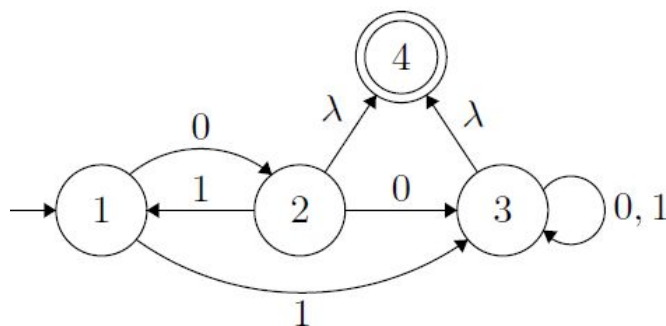
$$f(1) = 2$$

$$f(2) = 3$$

$$f(3) = 8$$

Counting Paths w Generating Functions

- ▶ We can compute the generating function, $g(z)$, for a DFA from the associated matrix



$$T = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(z) = (-1)^n \frac{\det(I - zT: n+1, 1)}{z \times \det(I - zT)} = \frac{2z - z^2}{1 - 2z - z^2 + 2z^3}$$

Counting Paths w Generating Functions

▶

$$g(z) = \frac{2z - z^2}{1 - 2z - z^2 + 2z^3}$$

- ▶ Each $f(i)$ can be computed by Taylor expansion of $g(z)$

$$g(z) = \frac{g(0)}{0!} z^0 + \frac{g^{(1)}(0)}{1!} z^1 + \frac{g^{(2)}(0)}{2!} z^2 + \dots + \frac{g^{(n)}(0)}{n!} z^n + \dots$$

$$g(z) = 0z^0 + 2z^1 + 3z^2 + 8z^3 + 15z^4 + \dots$$

$$g(z) = f(0)z^0 + f(1)z^1 + f(2)z^2 + f(3)z^3 + f(4)z^4 + \dots$$


Recurrence Relation

- ▶ An easier way to compute number of paths
- ▶ We can extract a homogeneous linear recurrence from $g(z)$ using linear algebraic techniques

$$f(0) = 0, f(1) = 2, f(2) = 3$$

$$f(k) = 2f(k - 1) + f(k - 2) - 2f(k - 3)$$

Good job Will Hunting!

G is the graph 

Find:

- 1) The adjacency matrix, A .
- 2) The matrix giving the number of 3 step walks
- 3) The generating function for walks from $i \rightarrow j$
- 4) The generating function for walks from $1 \rightarrow 3$

1.) $A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ 2.) $A^3 = \begin{pmatrix} 2 & 7 & 2 \\ 7 & 2 & 7 \\ 2 & 7 & 2 \end{pmatrix}$

3.) $P_i(z) = \sum_{n=0}^{\infty} w_n(i \rightarrow j) z^n$
 $= \frac{\det(\mathbb{1}_i, -zA_{ij})}{\det(\mathbb{1} - zA)}$
 $= \frac{\det \begin{pmatrix} 1 & -z & 0 \\ -z & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}{\det(\mathbb{1} - zA)}$
 $= \frac{1 - z^2}{1 - 7z^2 + 4z^4}$
 $= 1 + 7z^2 + 19z^4 + 47z^6 + \dots$

**This is correct.
Who did this ?**

Path Counting Methods

- ▶ Complexities of different path counting methods
 - ▶ n is the number of states
 - ▶ k is the length bound

	Matrix Exponentiation	Dynamic Programming	
Construction Time			
Evaluation Time			

Outline

- ▶ Automata-based Constraint Solver
 - ▶ String constraint Language
 - ▶ Constraint types
 - ▶ Automaton construction
- ▶ Automata-based Model Counting
 - ▶ Generating functions
 - ▶ Recurrences
- ▶ **Experimental Results**
- ▶ **Conclusion**

Experimental Evaluation

- ▶ We implemented the techniques described as a tool called Automata-Based model Counter (ABC)
- ▶ We conducted experiments on 4 benchmark sets
 - ▶ A java benchmark with wide range of string operations
 - ▶ Kaluza Small & Kaluza Big benchmarks for satisfiability check
 - ▶ SMC examples for direct comparison between SMC and ABC

	Frequency of Operations Per 1000 Formulas				
ASE	0.42	386.10	129.39	382.54	716.5
Kaluza Small	30.29	93.89	224.87	46.84	0
Kaluza Big	38.12	129.53	164.64	60.46	0

Satisfiability Check Comparison

- ▶ Compared with CVC4
 - ▶ Used SMT-lib format of Kaluza benchmarks from CVC4

	ABC-CVC4 sat-sat	ABC-CVC4 unsat-unsat	ABC-CVC4 sat-unsat	ABC-CVC4 unsat-sat	ABC-CVC4 sat-timeout
sat/small	19728	3	0	0	0
sat/big	1587	0	0	0	0
unsat/small	8139	3013	74	0	0
unsat/big	3419	5904	2385	0	2359

- ▶ Constraint solver performance for Kaluza benchmarks

	ABC Avg. Time (seconds)	CVC4 Avg. Time (seconds)
big	0.44	0.01
small	0.01	0.015

Model Counting Comparison

► Compared with SMC

	Max string length k	SMC lower bound (log)	SMC upper bound (log)	ABC count (log)
nullhttpd	500	3752	3760	3760
ghttpd	620	4880	4896	4896
csplit	629	4852	4921	4921
grep	629	4676	4763	4763
wc	629	4281	4284	4281
obscure	6	0	3	2

► Model counter performance for satisfiable constraints in the Kaluza benchmarks

	ABC Avg. Time (seconds)	SMC Avg. Time (seconds)
big	0.34	5.29
small	0.32	0.26

ASE Benchmark

- ▶ Extracted from 7 real-world server-side Java applications
 - ▶ Constraints were generated by extracting program path constraints through dynamic symbolic execution
 - ▶ SMC and CVC4 are not able to handle ASE benchmark; they do not support sanitization operations such as replace

# of satisfiable path constraints	Avg. # of BDD Nodes (each 16 bytes)	Avg. Running Time (seconds)
66236	69.51	0.0015

- ▶ We use MONA automata package where the transition relation is stored as a decision diagram

Conclusion

- ▶ Presented a model-counting string solver
 - ▶ Generates an automaton that accepts all solutions to a given constraint
 - ▶ Generates a model-counting function that, given a length bound, returns the number of solutions within that bound

- ▶ Future work
 - ▶ Extending ABC with new operations (e.g., lastIndexOf)
 - ▶ Extending ABC with Presburger arithmetic
 - ▶ Better approximation of relational constraints with multi-track automata

Thanks



Recurrence Relation

- ▶ An easier way to compute number of paths
- ▶ We can extract a linear recurrence from $g(z)$ using linear algebraic techniques

$$f(0) = 0, f(1) = 2, f(2) = 3$$

$$f(k) = 2f(k-1) + f(k-2) - 2f(k-3)$$

- ▶ A closed form solution can also be obtained from the recurrence relation

$$f(k) = \frac{2^{k+1} + (-1)^{k+1} - 1}{2}$$

Path Counting w Generating Functions

- ▶ $F = \neg(x \in (01)^*) \wedge LEN(x) \geq 1$
- ▶ Language over Σ that satisfy $F : \mathcal{L}^x = \Sigma^* - (01)^*$
- ▶ Let $\mathcal{L}_i^x = \{\omega \in \mathcal{L}^x : |\omega| = i\}$, $f(i) = |\mathcal{L}_i^x|$
- ▶ $|\mathcal{L}^x|$ is sum of the series $f(0), f(1), f(2), f(3) \dots$

$$f(0) = 0, f(1) = 2, f(2) = 3, f(3) = 8 \dots$$

- ▶ We can encode each $f(i)$ in a power series as a coefficient

$$g(z) = f(0)z^0 + f(1)z^1 + f(2)z^2 + f(3)z^3 + f(4)z^4 + \dots$$

$$g(z) = 0z^0 + 2z^1 + 3z^2 + 8z^3 + 15z^4 + \dots$$