Solving String Constraints via Hardware/Software Model Checking

Jie-Hong Roland Jiang[†], Fang Yu [§]

[†]National Taiwan University
[§]National Chengchi University

Meeting on String Constraints and Applications (MOSCA'19),
May 6-9, 2019, Bertinoro, Italy

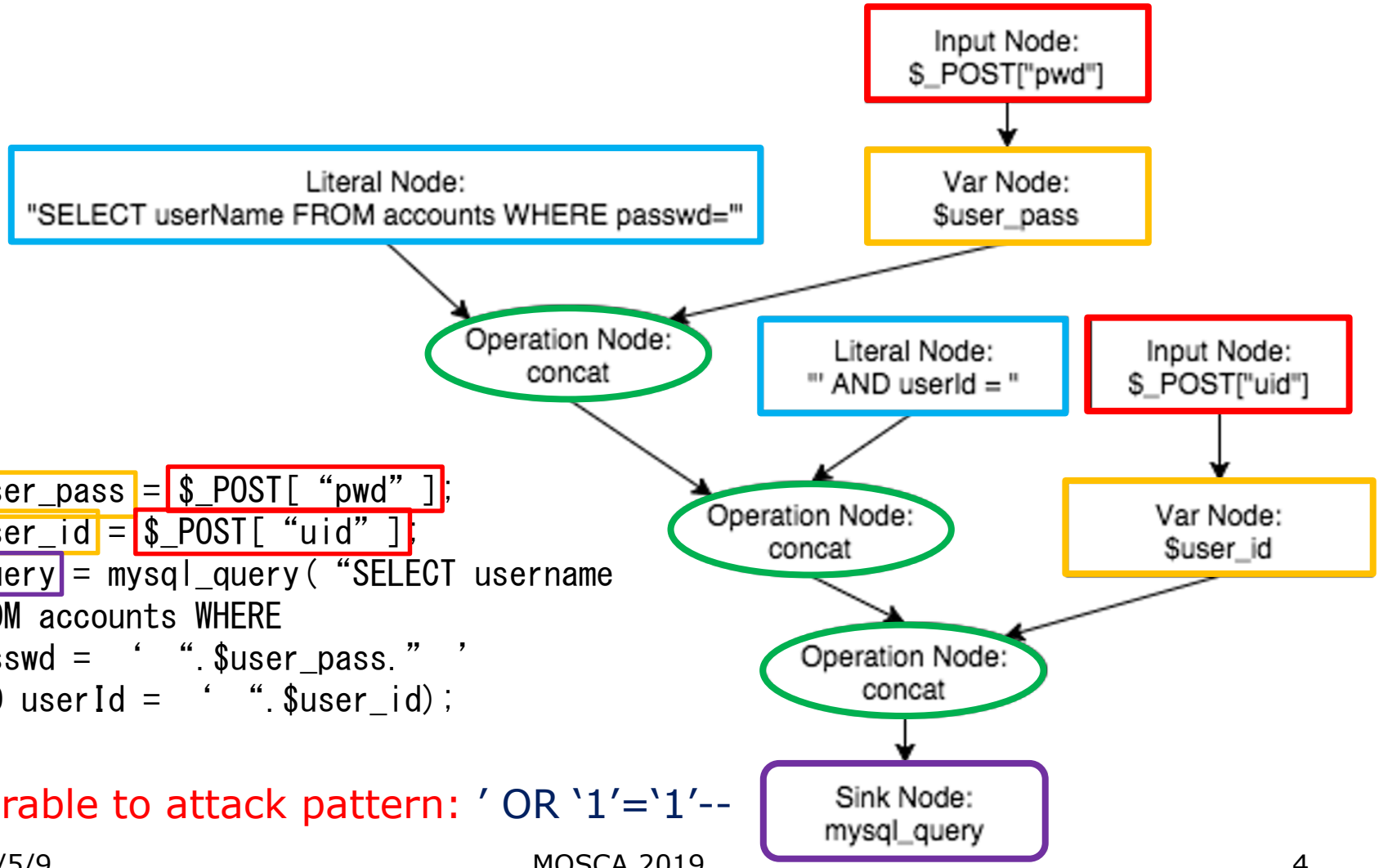
Outline

- Introduction
- Circuit based solving of string constraints (SLOG)
- Circuit based solving of string+length constraints (SLENT)
- Conclusions

Outline

- **Introduction**
- Circuit based solving of string constraints (SLOG)
- Circuit based solving of string+length constraints (SLENT)
- Conclusions

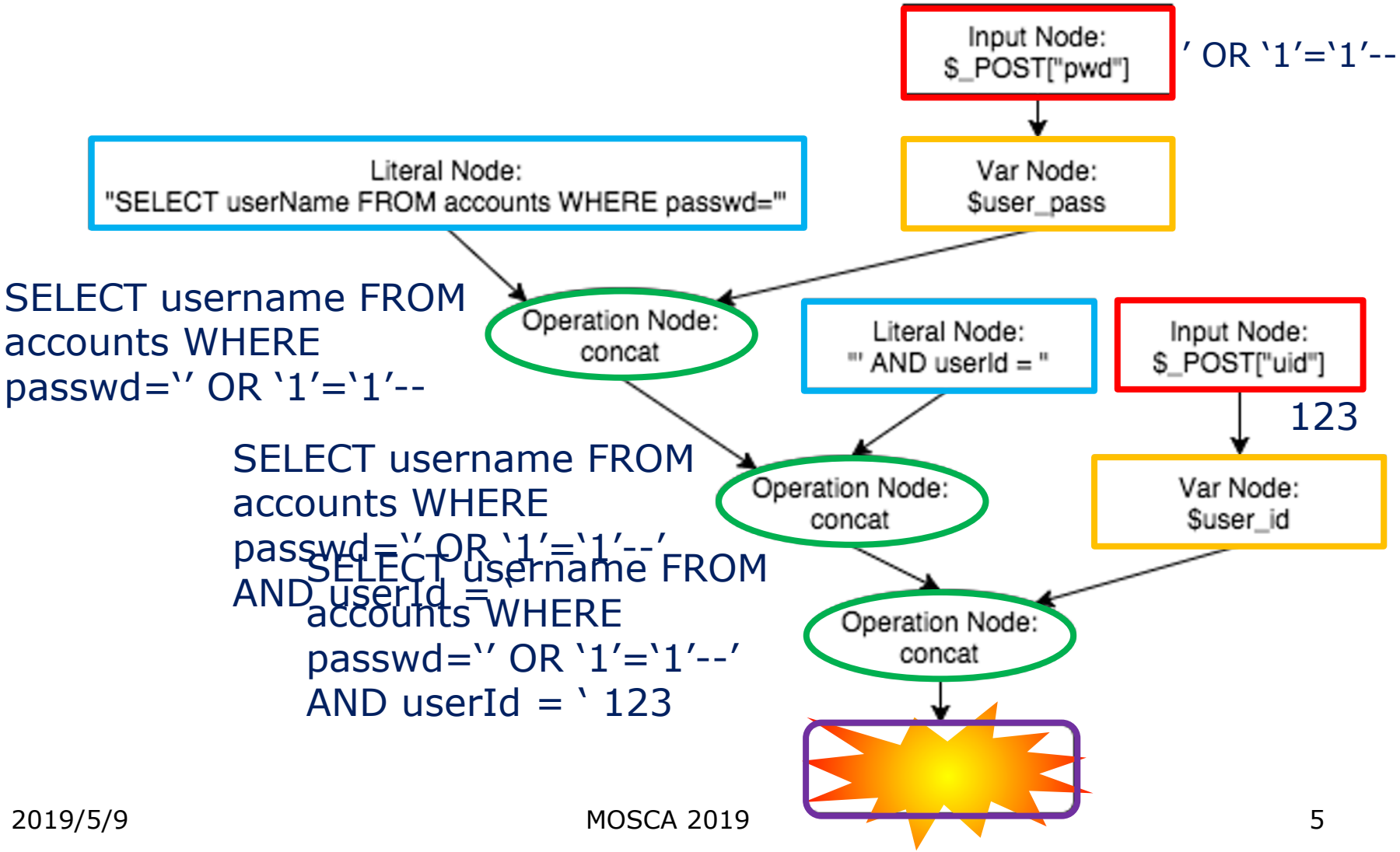
Motivating Example




```
<?PHP
   $user_pass = $_POST[ "pwd" ];
   $user_id = $_POST[ "uid" ];
   $query = mysql_query( "SELECT username
   FROM accounts WHERE
   passwd =  '  ".$user_pass."  '
   AND userId =  '  ".$user_id);
?>
```

Vulnerable to attack pattern: ' OR '1'='1'--

Motivating Example



Introduction

- String analysis
  - Problem formulation
    - Input: an acyclic dependency graph extracted from string manipulating program
    - Output: possible string values for each string variable under program execution

Related Work

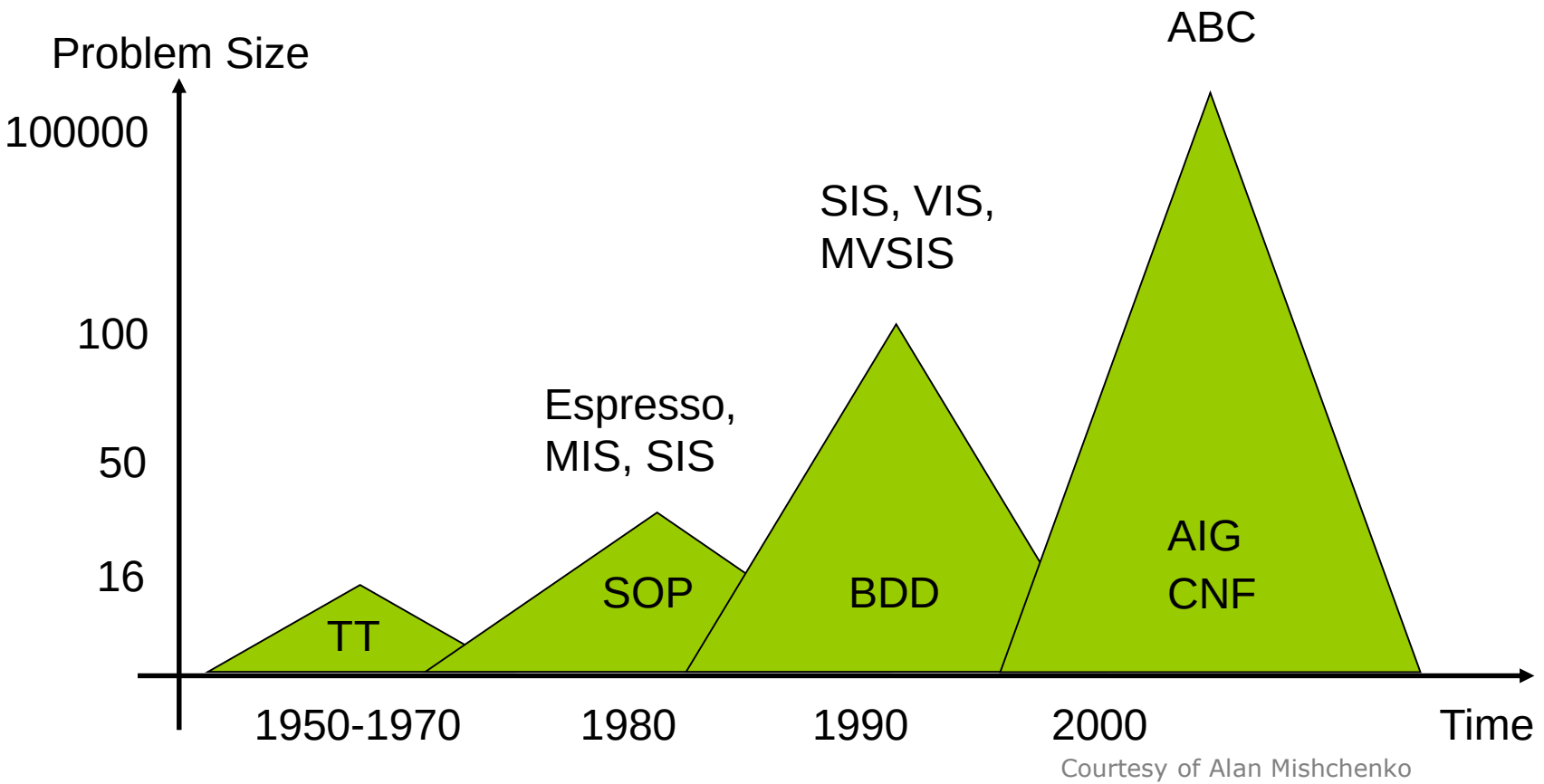
- Automata-based approaches
  - Use finite automata to characterize the set of possible string values in program execution
  - Can generate filter
  - Not support counterexample generation
- SMT-based approaches
  - Use string theory to decide whether a set of constraint is satisfiable
  - Can generate counterexample
  - Not support filter generation

Our Contribution

- We propose
  - Automata-based string analysis method using logic circuit representation
    - Advantages of logic circuit representation:
      - Support both counterexample generation and filter generation
      - Maintain circuit size linear in the input circuit sizes during string operations

Motivation of Circuit Representation

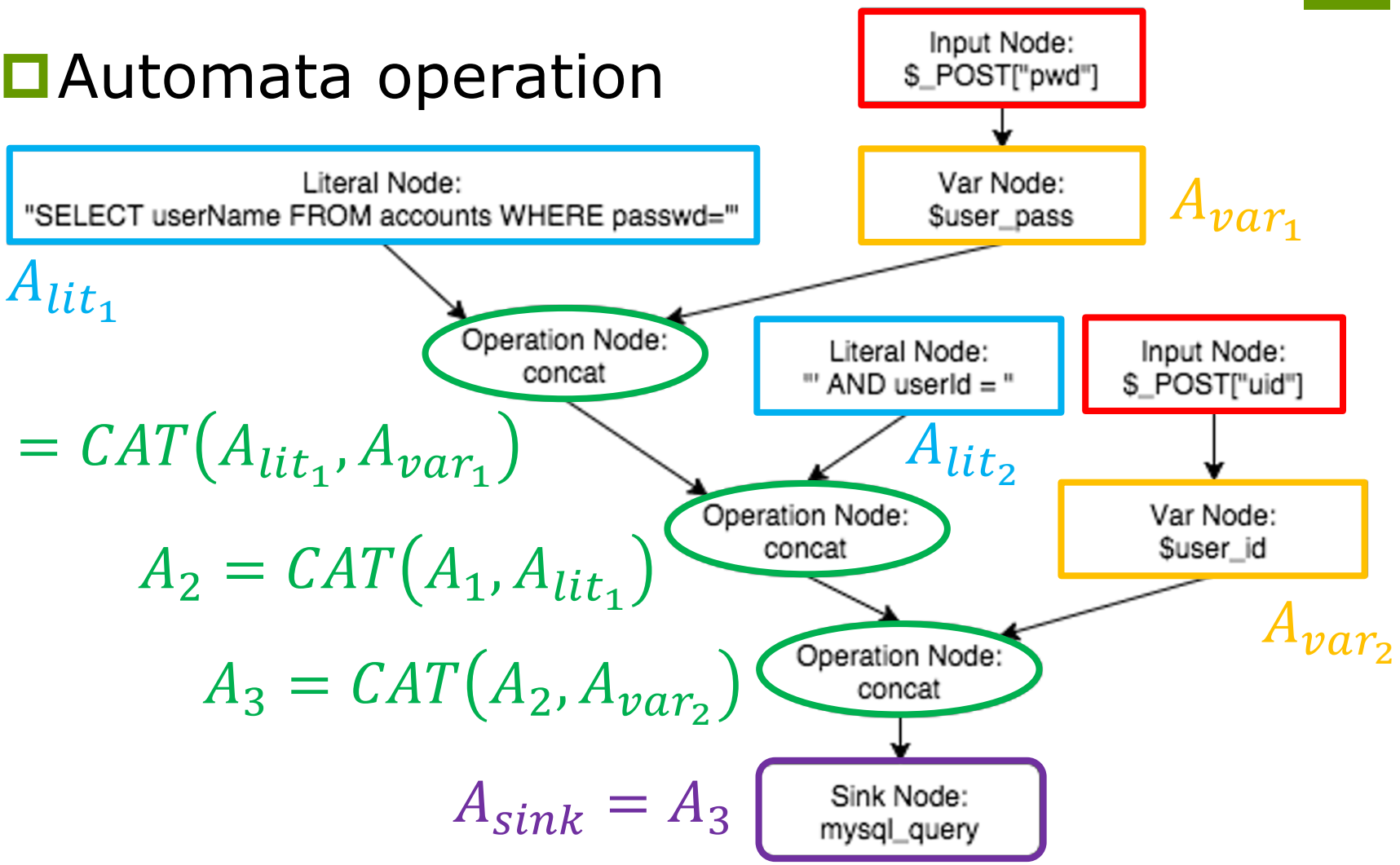
- Historic evolution of data structures and tools in logic synthesis and verification



Courtesy of Alan Mishchenko

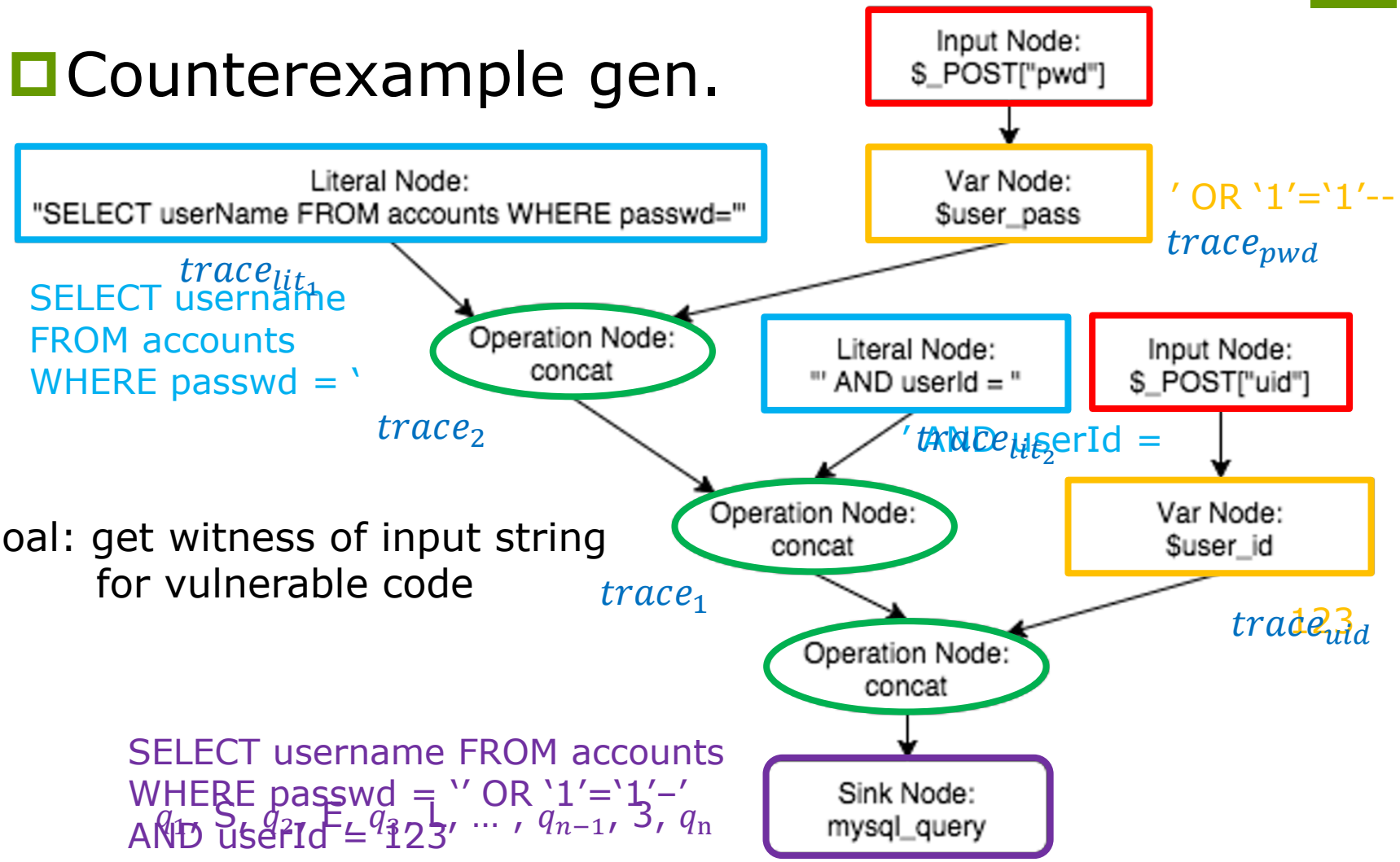
Introduction

- Automata operation



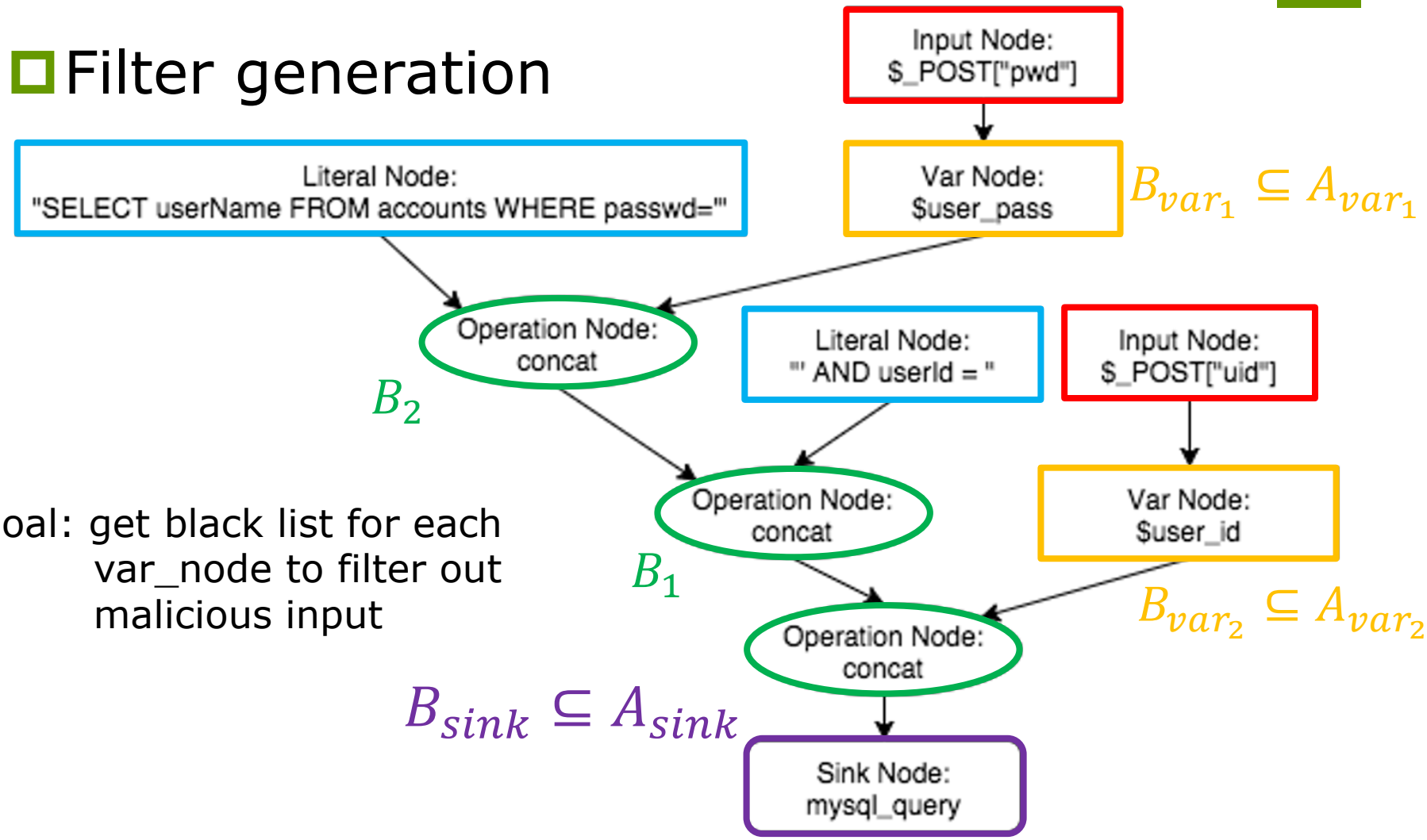
Introduction

- Counterexample gen.



Introduction

- Filter generation



Goal: get black list for each var_node to filter out malicious input

Outline

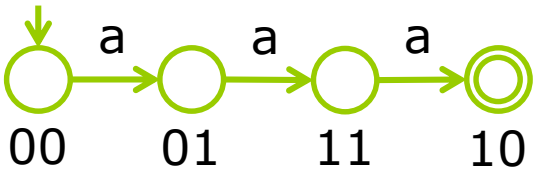
- Introduction
- **Circuit based solving of string constraints**
- Circuit based solving of string+length constraints
- Conclusions

Preliminaries

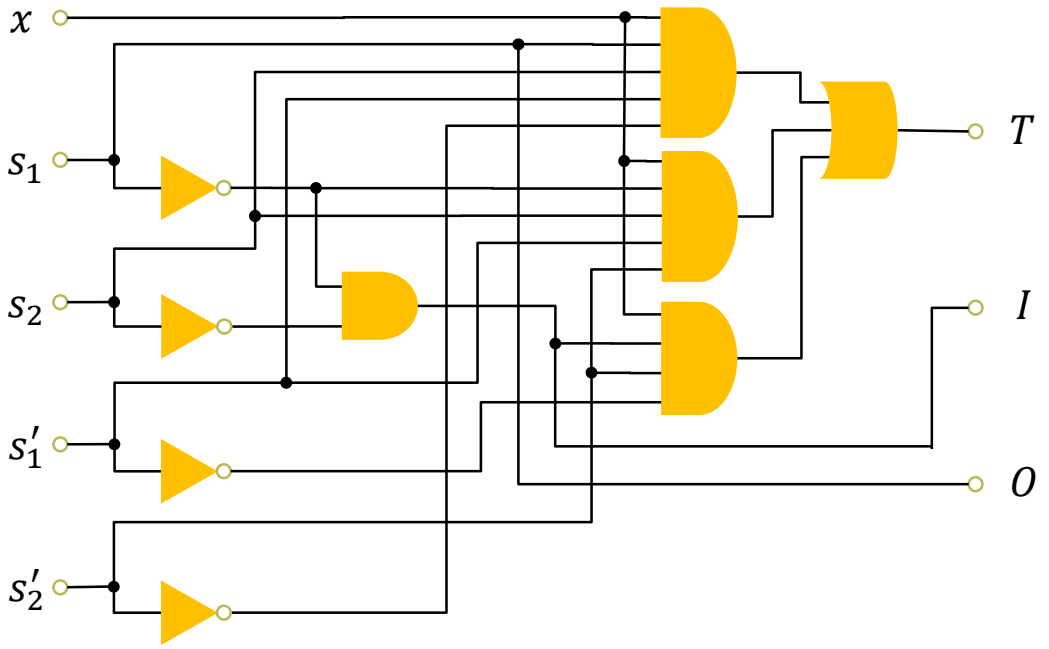
- A (nondeterministic) *finite automaton* $A = (Q, \Sigma, I, T, O)$
  - Q: finite state set
  - Σ: finite alphabet
  - $I \subseteq Q$: set of initial states
  - $T \subseteq \Sigma \times Q \times Q$: transition relation
  - $O \subseteq Q$: set of accepting states
- Reserve a symbol for ϵ in Σ
- $\mathcal{L}(A)$ denote the *language* (set of strings) accepted by A

Preliminaries

- Circuit representation
  - Boolean encoding on Q, Σ
  - Use characteristic functions $I(\vec{s})$, $T(\vec{x}, \vec{s}, \vec{s}')$, $O(\vec{s})$ to represent an automaton



$\Sigma = \{\epsilon, a\}$



String/Automata Operations

- Supported operations
  - Intersection
  - Union
  - Concatenation
  - Replacement
  - Reverse
  - Prefix
  - Suffix
  - Emptiness checking

Intersection

- Circuit construction
  - Construct automaton $A = INT(A_1, A_2)$ with $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$:

$$T_{INT}(\vec{x}, \vec{s}, \vec{s}') = T_1^{\epsilon}(\vec{x}, \vec{s}_1, \vec{s}_1') \wedge T_2^{\epsilon}(\vec{x}, \vec{s}_2, \vec{s}_2')$$

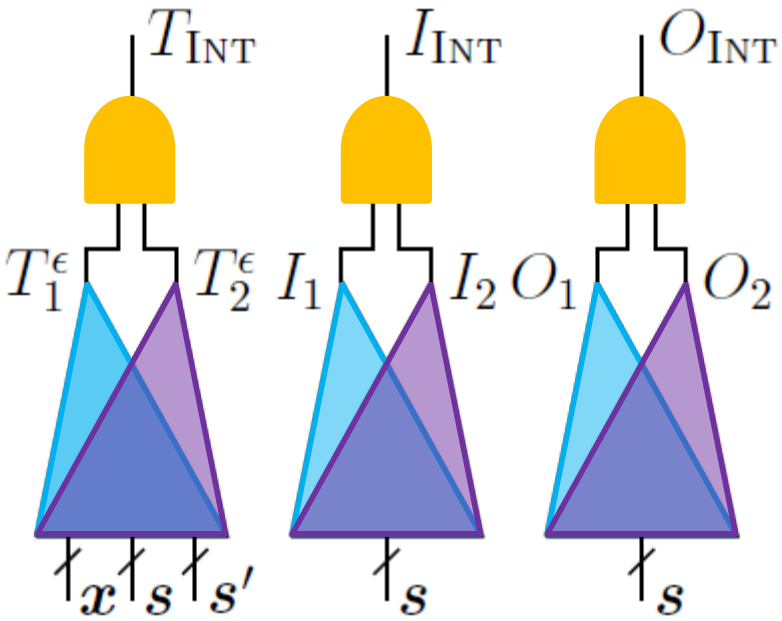
$$I_{INT}(\vec{s}) = I_1(\vec{s}_1) \wedge I_2(\vec{s}_2)$$

$$O_{INT}(\vec{s}) = O_1(\vec{s}_1) \wedge O_2(\vec{s}_2)$$

$\vec{s} = (\vec{s}_1, \vec{s}_2)$

Intersection

- Circuit construction



Intersection

- Counterexample generation

$$\frac{A_1\colon\ (p_1, \sigma_1, \ldots, p_\ell) \qquad A_2\colon\ (q_1, \sigma_1, \ldots, q_\ell)}{A\colon\ ((p_1, q_1), \sigma_1, (p_2, q_2), \sigma_2, \ldots, (p_\ell, q_\ell))}\ \textsc{IntCex}$$

Intersection

- Filter generation
  - Let B be the filter for $A = Int(A_1, A_2)$
  - B can directly applied as a filter for A_1 as well as A_2

Union

- Circuit construction
  - Construct automaton $A = UNI(A_1, A_2)$ with $\mathcal{L}(A) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$:

$$T_{UNI}(\vec{x}, \vec{s}, \vec{s}') = \left(\neg\alpha \wedge \neg\alpha' \wedge T_1(\vec{x}, \langle\vec{s}_2\rangle_m, \langle\vec{s}_2'\rangle_m)\right) \vee \left(\alpha \wedge \alpha' \wedge T_2(\vec{x}, \vec{s}_2, \vec{s}_2')\right)$$

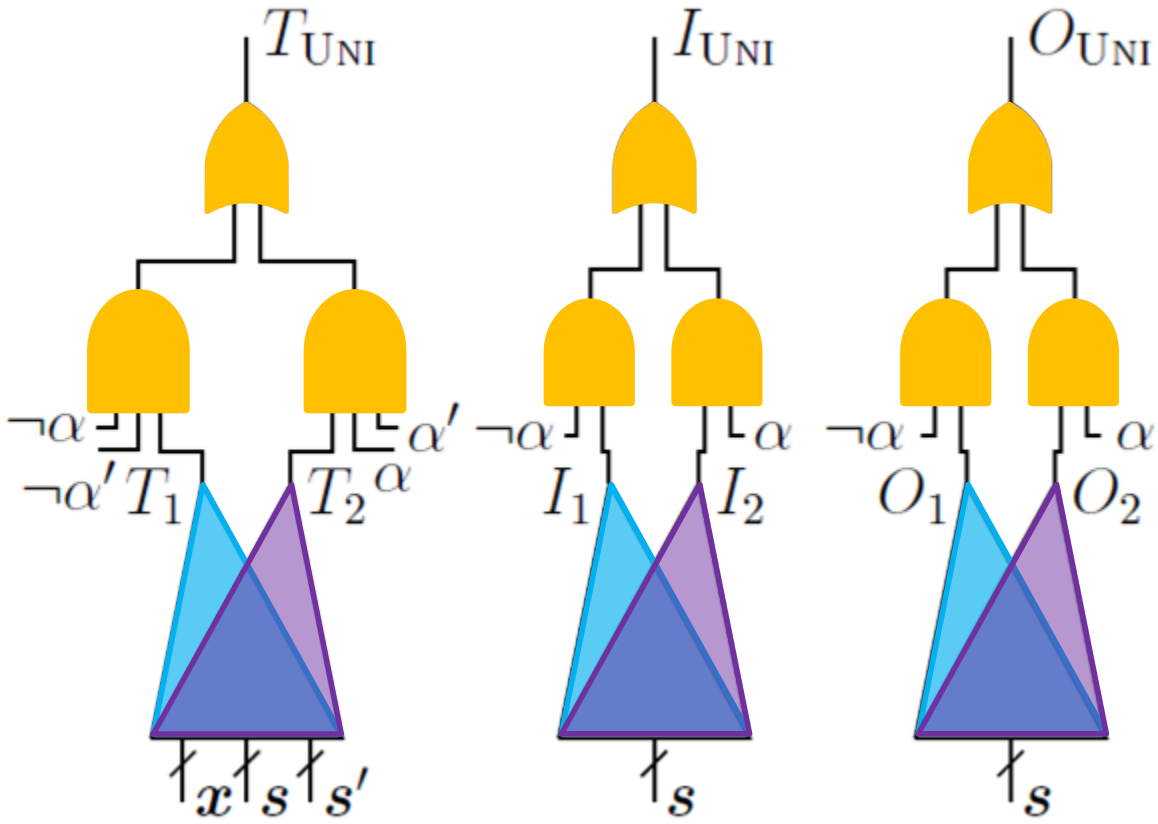
$$I_{UNI}(\vec{s}) = \left(\neg\alpha \wedge I_1(\langle\vec{s}_2\rangle_m)\right) \vee \left(\alpha \wedge I_2(\vec{s}_2)\right)$$

$$O_{UNI}(\vec{s}) = \left(\neg\alpha \wedge O_1(\langle\vec{s}_1\rangle_m)\right) \wedge \left(\alpha \wedge O_2(\vec{s}_2)\right)$$

$\vec{s} = (\vec{s}_2, \alpha)$
$\langle\vec{s}_2\rangle_m$: taking first m variables of $\vec{s}_2$
$\alpha = 0$: state in A_1, $\alpha = 1$: state in A_2

Union

- Circuit construction



Union

- Counterexample generation

$$\frac{A_1\colon\ (q_1, \sigma_1, q_2, \sigma_2, \ldots, q_\ell) \qquad A_2\colon\ (\bot)}{A\colon\ ((q_1, c), \sigma_1, (q_2, c), \sigma_2, \ldots, (q_\ell, c))}\ \textsc{UniCex},\ c = 0$$

$$\frac{A_1\colon\ (\bot) \qquad A_2\colon\ (q_1, \sigma_1, q_2, \sigma_2, \ldots, q_\ell)}{A\colon\ ((q_1, c), \sigma_1, (q_2, c), \sigma_2, \ldots, (q_\ell, c))}\ \textsc{UniCex},\ c = 1$$

Union

- Filter generation
  - Let B be the filter for $A = Uni(A_1, A_2)$
  - $B_1 = Int(B, A_1)$ and $B_2 = Int(B, A_2)$ form legitimate filter for A_1 and A_2, respectively

Concatenation

- Circuit construction
  - Construct automaton $A = CAT(A_1, A_2)$ with $\mathcal{L}(A) = \mathcal{L}(A_1).\mathcal{L}(A_2)$:

$$T_{CAT}(\vec{x}, \vec{s}, \vec{s}') = \left(\neg\alpha \wedge \neg\alpha' \wedge T_1(\vec{x}, \langle\vec{s}_2\rangle_m, \langle\vec{s}_2'\rangle_m)\right) \vee \left(\alpha \wedge \alpha' \wedge T_2(\vec{x}, \vec{s}_2, \vec{s}_2')\right) \vee \left((\vec{x} = \epsilon) \wedge \neg\alpha \wedge \alpha' \wedge O_1(\langle\vec{s}_2\rangle_m) \wedge I_2(\vec{s}_2')\right)$$

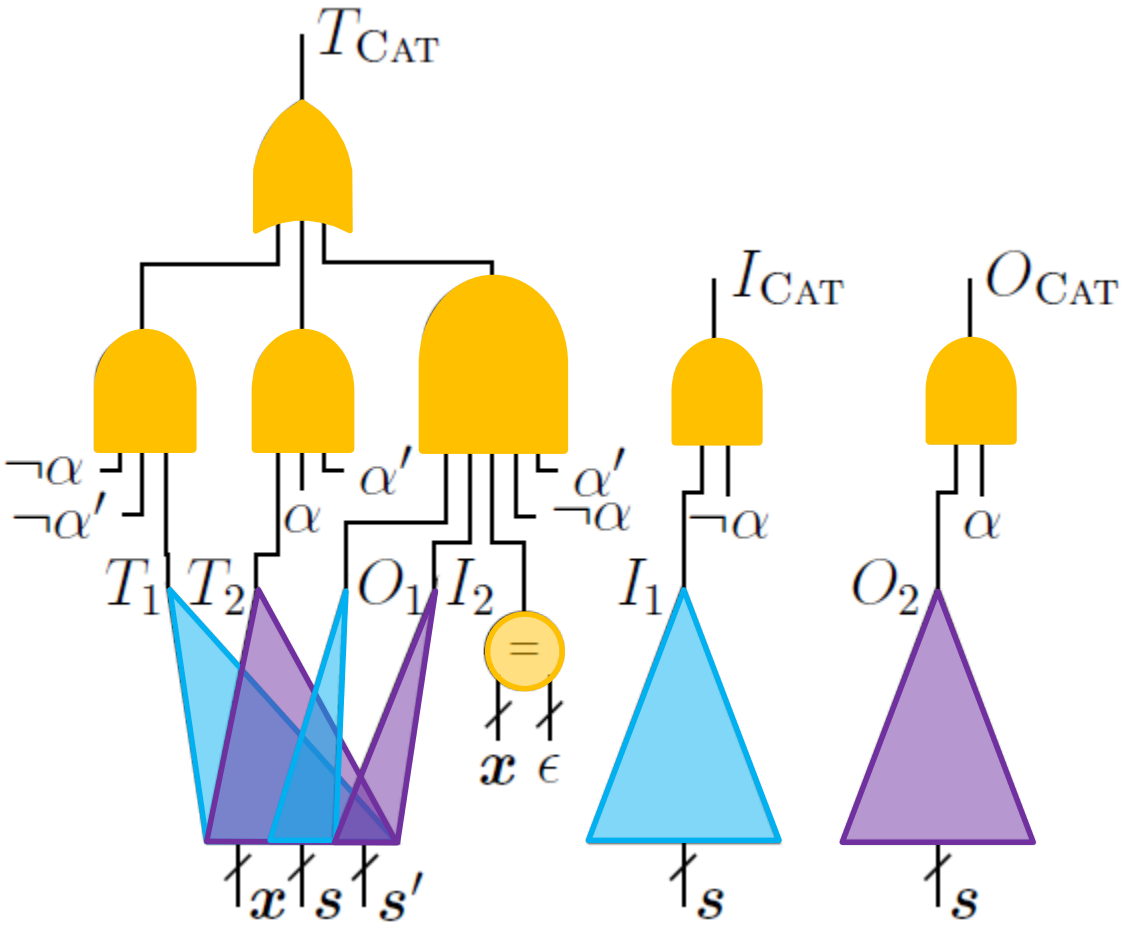
$$I_{CAT}(\vec{s}) = \left(\neg\alpha \wedge I_1(\langle\vec{s}_2\rangle_m)\right)$$

$$O_{CAT}(\vec{s}) = \left(\alpha \wedge O_2(\vec{s}_2)\right)$$

$\vec{s} = (\vec{s}_2, \alpha)$
$\langle\vec{s}_2\rangle_m$: taking first m variables of $\vec{s}_2$
$\alpha = 0$: state in A_1, $\alpha = 1$: state in A_2

Concatenation

- Circuit construction



Concatenation

- Counterexample generation

$$\frac{A_1\colon (q_1, \sigma_1, \ldots, q_i) \qquad A_2\colon (q_{i+1}, \sigma_{i+1}, \ldots, q_n)}{A_{\text{CAT}}\colon ((q_1, 0), \sigma_1 \ldots, (q_i, 0), \epsilon, (q_{i+1}, 1), \sigma_{i+1}, \ldots, (q_\ell, 1))}\ \textsc{CatCex}$$

Concatenation

- Filter generation
  - Let B be the filter for $A = Cat(A_1, A_2)$
  - First construct $B^{\dagger} = Int(A, B)$
  - Let B_1 be a copy of $B^{\dagger}$ but with all the transition between states of $\alpha = 1$ being replaced with ϵ-transition
  - Let B_2 be a copy of $B^{\dagger}$ but with all the transition between states of $\alpha = 0$ being replaced with ϵ-transition

Emptiness Checking

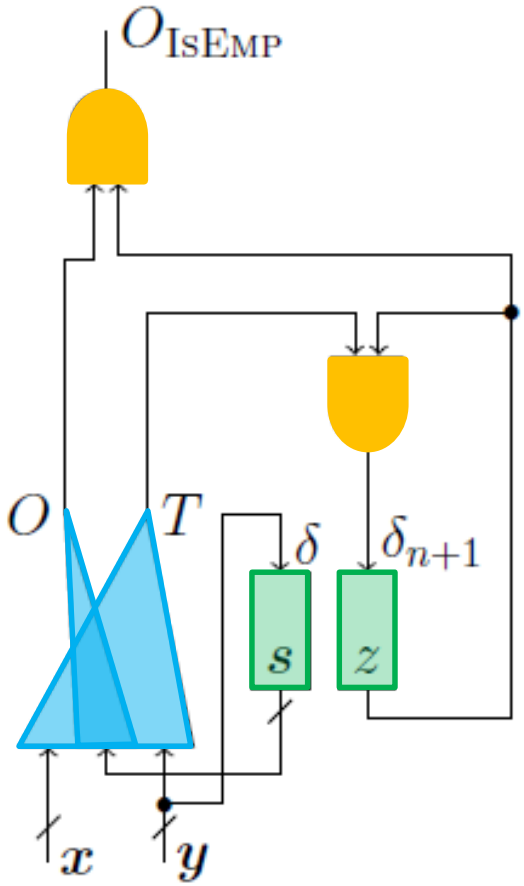
- Decide whether an automaton A accept any string or not
- T characterize one step of transition
- Convert into sequential circuit, then apply property directed reachability (PDR)
  - "pdr" command in Berkeley ABC only takes sequential circuits with *single initial state* and *transition functions*

Emptiness Checking

- Convert transition relation to transition functions
  - n new variables $\vec{y}$ for $n = |\vec{s}|$
  - One new state variable z with initial value 1
  - Output function: $\omega = (O(s) \wedge z)$
  - Next-state functions: $\delta_i = (y_i)$ for state variables s_i, $i = 1, \ldots, n$, and $\delta_{n+1} = (T(\vec{x}, \vec{s}, \vec{y}) \wedge z)$ for state variable z

Emptiness checking

- Circuit construction



Experiment Setting

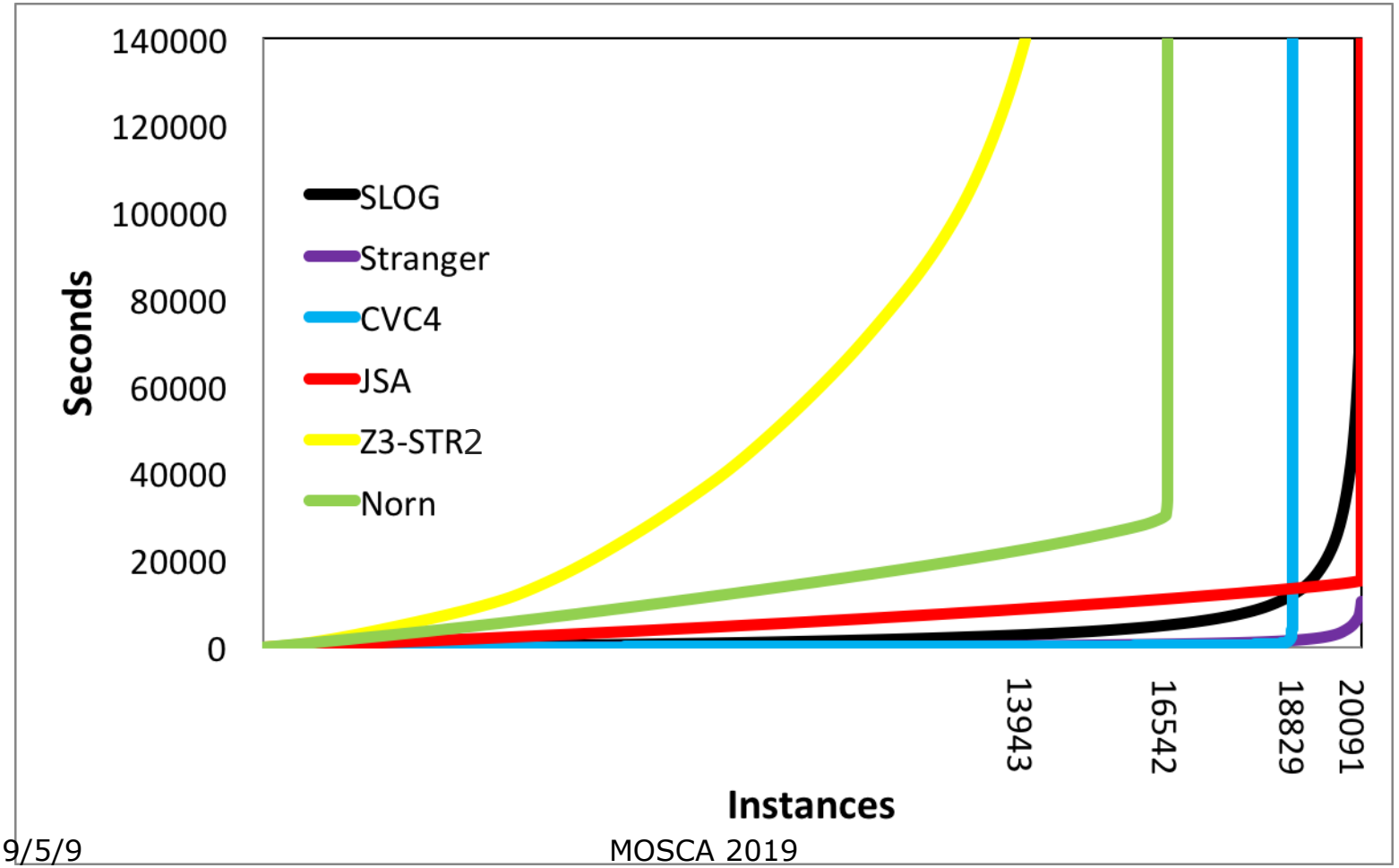
- Compared string constraint solvers
  - Our tool: SLOG
  - Automata-based tools: JSA, Stranger
  - SMT-based tools: CVC4, Norn, Z3-str2
- Environment
  - Intel Xeon 8-core CPU
  - 16GB memory
  - Ubuntu 12.04 LCS

Experimental Setting

- 20000+ string analysis instances generated from web applications
  - Replacement-free small instances
  - Replacement-free large instances
  - Instances with replacement operation

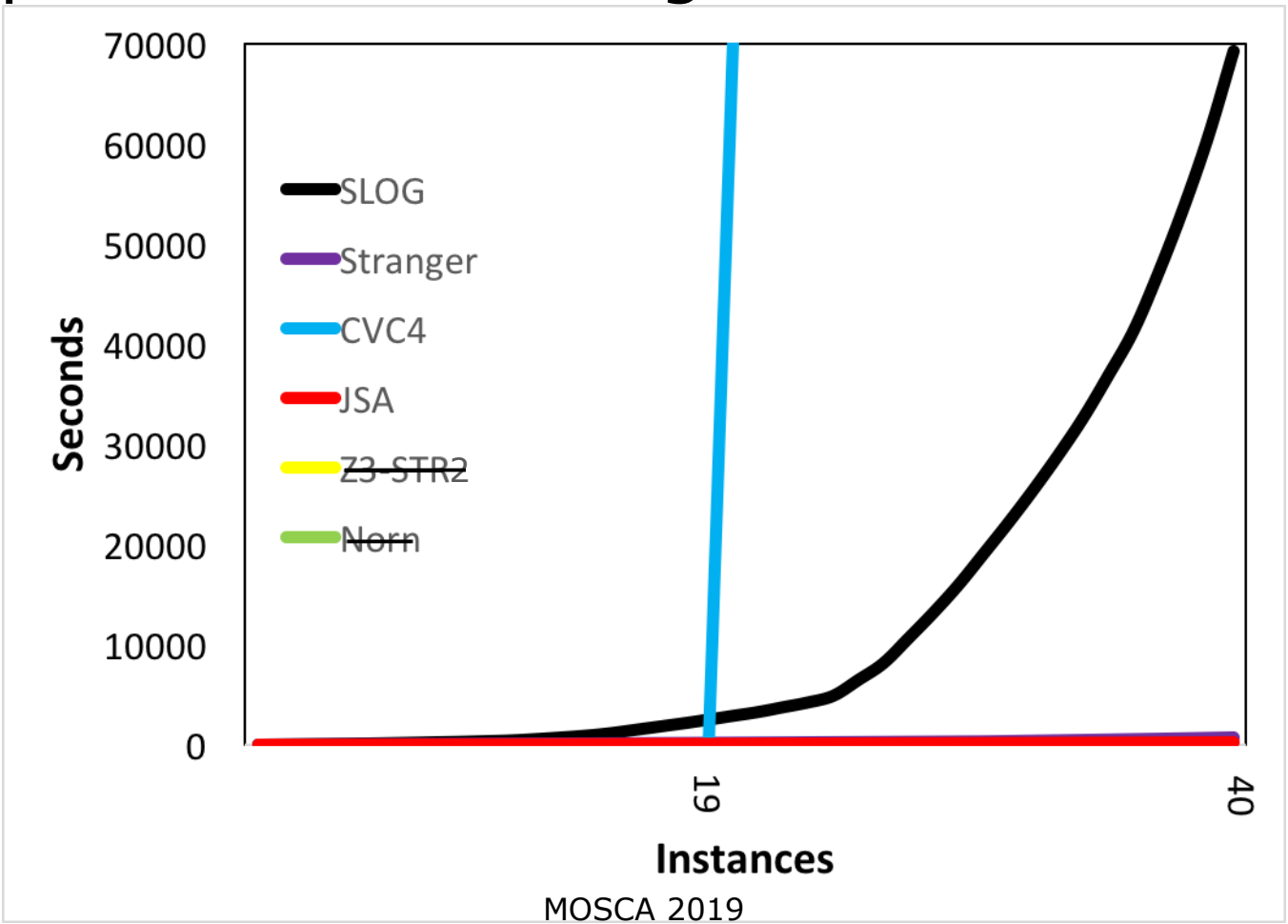
Experimental Results (1/4)

- Replacement-free small instances



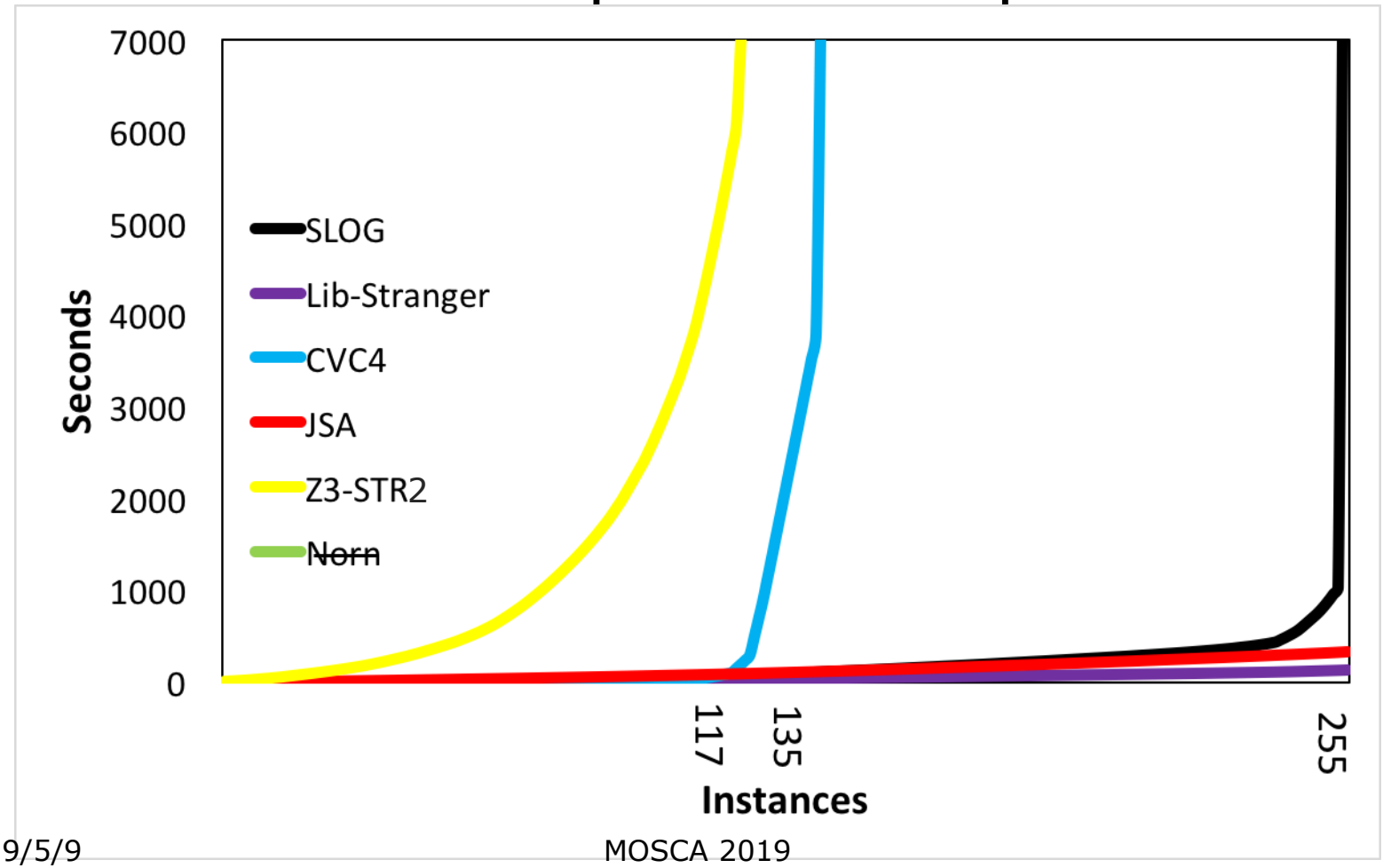
Experimental Results (2/4)

- Replacement-free large instances



Experimental Results (3/4)

- Instances with replacement operation



Experimental Results (4/4)

- SLOG performance on counterexample generation

Group	#SAT	SolveTime (s)	CexGenTime (s)
Small	8426	60664	481
Large	22	4236	18
Replacement	236	1015	25

accumulated solving and counterexample generation time

Outline

- Introduction
- Circuit based solving of string constraints (SLOG)
- **Circuit based solving of string+length constraints (SLENT)**
- Conclusions

Length Tracking

- Circuit construction
  - Construct automaton $A^L = TrkLen(A)$ with:

$$T^L(\vec{x}, \vec{s}, n, \overrightarrow{s'}, n) = T(\vec{x}, \vec{s}, \overrightarrow{s'}) \wedge$$
$$(((\vec{x} \neq \epsilon) \wedge (n' = n + 1)) \vee$$
$$((\vec{x} = \epsilon) \wedge (n' = n)))$$
$$I^L(\vec{s}) = I(\vec{s})$$
$$O^L(\vec{s}) = O(\vec{s})$$

Intersection

- Circuit construction
  - Construct automaton $A = INT(A_1, A_2)$ with $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$:

$$T_{INT}\left(\vec{x}, \vec{s}, \vec{n}, \vec{s}', \overrightarrow{n'}\right) = T_1^{\epsilon}\left(\vec{x}, \vec{s}, \vec{n}, \vec{s}', \overrightarrow{n'}\right) \wedge T_2^{\epsilon}\left(\vec{x}, \vec{s}, \vec{n}, \vec{s}', \overrightarrow{n'}\right)$$

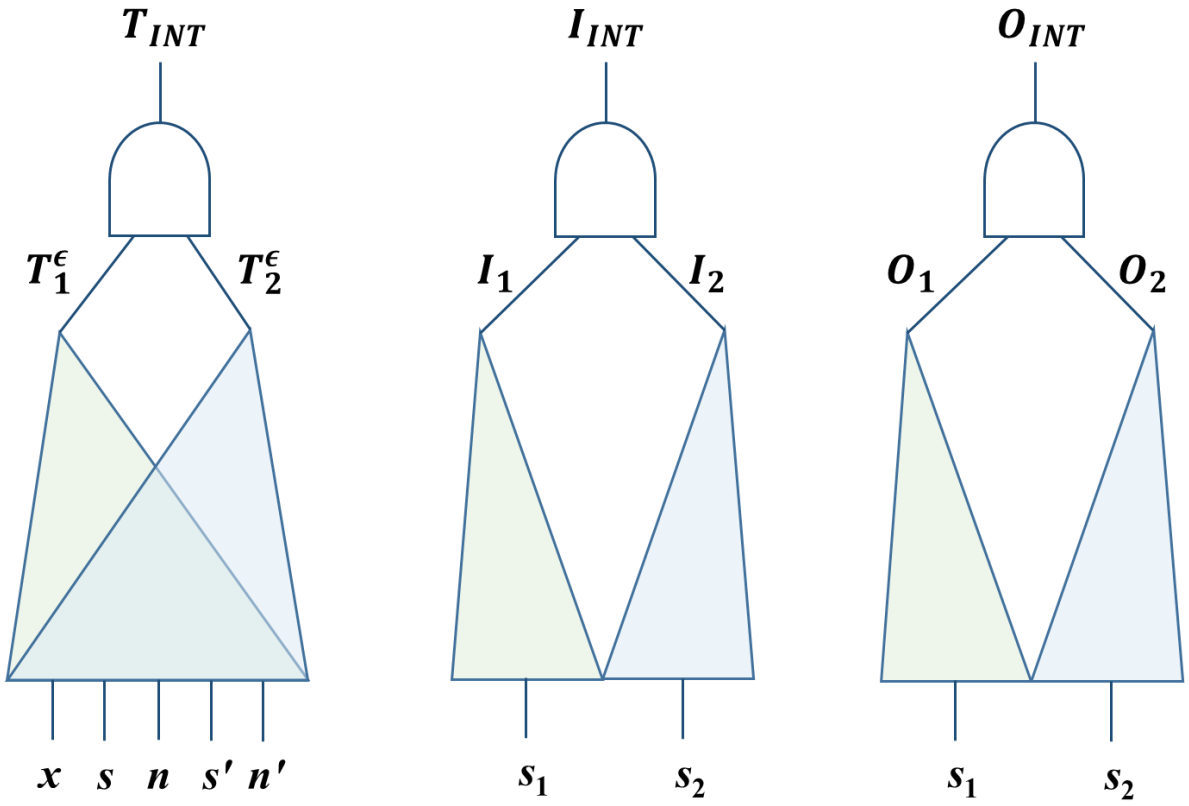
$$I_{INT}(\vec{s}) = I_1(\vec{s}_1) \wedge I_2(\vec{s}_2)$$

$$O_{INT}(\vec{s}) = O_1(\vec{s}_1) \wedge O_2(\vec{s}_2)$$

$$\vec{s} = (\vec{s}_1, \vec{s}_2)$$

Intersection

- Circuit construction



Union

- Circuit construction
  - Construct automaton $A = UNI(A_1, A_2)$ with $\mathcal{L}(A) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$:

$$T_{UNI}\left(\vec{x}, \vec{s}, \vec{n}, \vec{s}', \overrightarrow{n'}\right) = \left(\neg\alpha \wedge \neg\alpha' \wedge T_1\left(\vec{x}, \langle\vec{s}_2\rangle_m, \vec{n}, \langle\vec{s}_2'\rangle_m, \overrightarrow{n'}\right)\right) \vee \left(\alpha \wedge \alpha' \wedge T_2\left(\vec{x}, \vec{s}_2, \vec{n}, \vec{s}_2', \overrightarrow{n'}\right)\right)$$

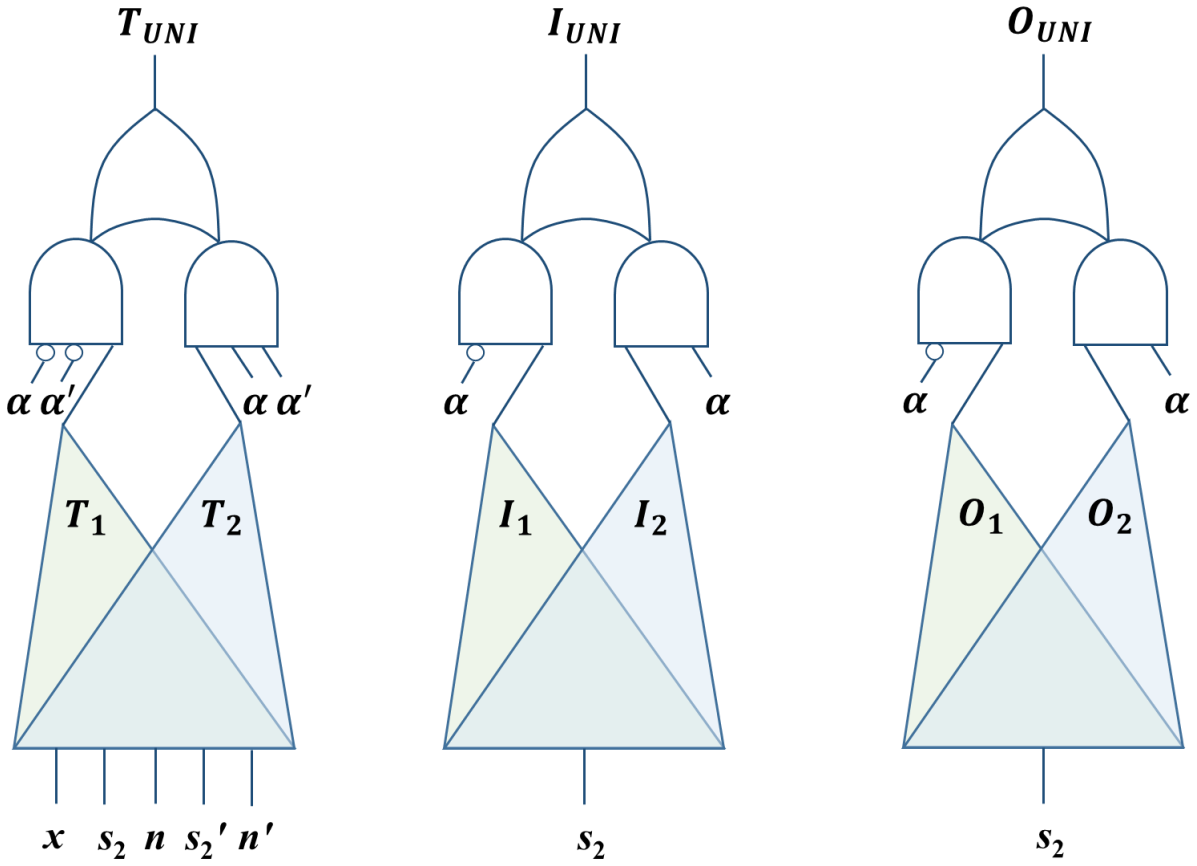
$$I_{UNI}(\vec{s}) = \left(\neg\alpha \wedge I_1(\langle\vec{s}_2\rangle_m)\right) \vee \left(\alpha \wedge I_2(\vec{s}_2)\right)$$

$$O_{UNI}(\vec{s}) = \left(\neg\alpha \wedge O_1(\langle\vec{s}_1\rangle_m)\right) \wedge \left(\alpha \wedge O_2(\vec{s}_2)\right)$$

$\vec{s} = (\vec{s}_2, \alpha)$
$\langle\vec{s}_2\rangle_m$: taking first m variables of $\vec{s}_2$
$\alpha = 0$: state in A_1, $\alpha = 1$: state in A_2

Union

- Circuit construction



Concatenation

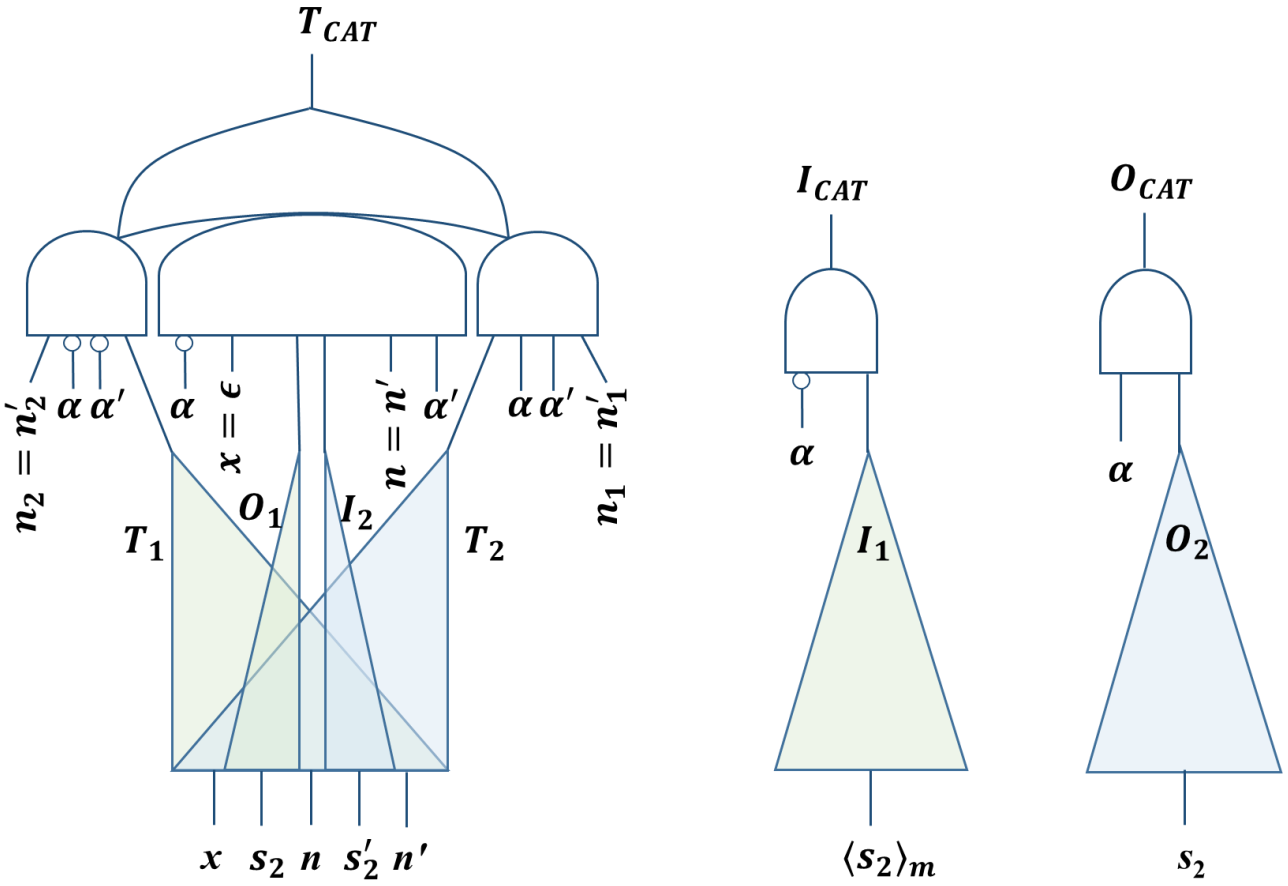
- Circuit construction
  - Construct automaton $A = CAT(A_1, A_2)$ with $\mathcal{L}(A) = \mathcal{L}(A_1).\mathcal{L}(A_2)$:

$$T_{CAT}\left(\vec{x}, \vec{s}, \vec{n}, \vec{s}', \overrightarrow{n'}\right) = (\neg\alpha \wedge \neg\alpha' \wedge T_1(\vec{x}, \langle\vec{s}_2\rangle_m, \vec{n}_1, \langle\vec{s}_2'\rangle_m, \vec{n}_1') \wedge (\vec{n}_2 = \vec{n}_2')) \vee$$
$$(\alpha \wedge \alpha' \wedge T_2(\vec{x}, \vec{s}_2, \vec{n}_2, \vec{s}_2', \vec{n}_2') \wedge (\vec{n}_1 = \vec{n}_1')) \vee$$
$$\left((\vec{x} = \epsilon) \wedge \neg\alpha \wedge \alpha' \wedge O_1(\langle\vec{s}_2\rangle_m) \wedge I_2(\vec{s}_2') \wedge (\vec{n} = \vec{n}')\right)$$
$$I_{CAT}(\vec{s}) = \left(\neg\alpha \wedge I_1(\langle\vec{s}_2\rangle_m)\right)$$
$$O_{CAT}(\vec{s}) = \left(\alpha \wedge O_2(\vec{s}_2)\right)$$

$\vec{s} = (\vec{s}_2, \alpha)$
$\langle\vec{s}_2\rangle_m$: taking first m variables of $\vec{s}_2$
$\alpha = 0$: state in A_1, $\alpha = 1$: state in A_2

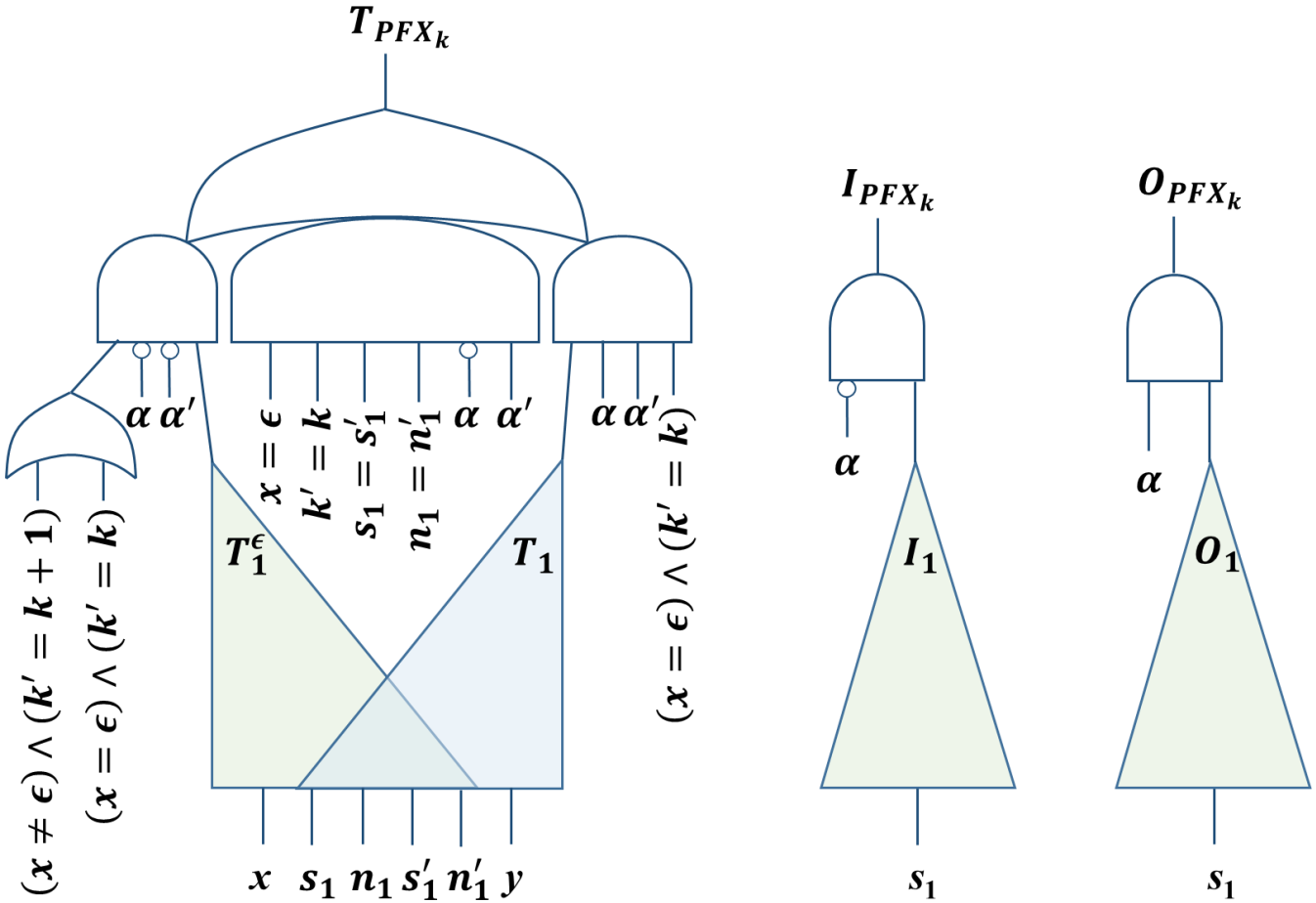
Concatenation

- Circuit construction



Prefix

- Circuit construction



Experiment Setting

- Compared string constraint solvers
  - Our tool: SLENT (using $IC3_{IA}$ model checker)
  - Automata-based tools: (UCSB) ABC
  - SMT-based tools: CVC4, Norn, S3P, Trau, Z3-str3
- Environment
  - Intel Xeon 8-core CPU
  - 16GB memory
  - Ubuntu 12.04 LCS

Experimental Results (1/3)

- 2000+ instances converted from Kaluza benchmarks involving only string concatenation and length constraints

solver	time (s)	#SAT	#UNSAT	#TO (200s)
Z3str3	56.46	1017	983	0
CVC4	88.89	1017	983	0
Norn	2025.30	1013	983	4
ABC	255.76	1013	983	4
S3P	137.90	1015	983	2
Trau	123.85	1017	983	0
SLENT	1397.82	1013	983	4

Experimental Results (2/3)

- 236 instances converted from Stranger benchmarks with involving string-to-string replaceall, concatenation, and length constraints

solver	time (s)	#SAT	#UNSAT	#TO (600s)	#abort
ABC	2282.84	109 (31)	111 (0)	0	16
S3P	605.79	30 (0)	114 (3)	22	70
Trau	687.49	54 (2)	139 (22)	5	38
SLENT	26692.55	88 (0)	141 (0)	7	0

Experimental Results (3/3)

- 101 instances converted from Stranger benchmarks with involving language-to-language replaceall, concatenation, and length constraints

solver	time (s)	#SAT	#UNSAT	#TO (600s)	#abort
ABC	977.80	46 (2)	41 (0)	1	13
SLENT	4413.25	44 (0)	38 (0)	19	0

9 under TO 1800s

Conclusions

- SLOG
  - Logic circuit based automata manipulation method for string analysis
    - Support both counterexample generation and filter synthesis
    - Maintain circuit size linear in input circuit sizes for string operations
    - Postpone language emptiness checking with model checking at the end

Conclusions (cont'd)

- SLENT
  - Encode length information to string automata as length-encoded automata
    - Construct characteristic functions of length-encoded automata through automata manipulations that correspond to string and length constraints
    - Leverage a symbolic model checker for infinite state systems as an engine for language emptiness checking

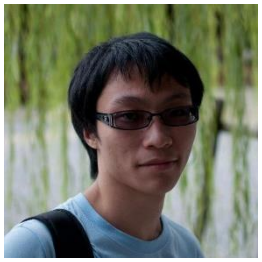
Conclusions (cont'd)

- SLOG and SLENT are based on scalable circuit representation and good at solving complex string constraints
- Counterexample generation and filter synthesis are possible
- Future work
  - Support complement operation
  - Allow relation on string variables

Acknowledgement

Shih-Yu Chen

Chun-Han Lin



Tsung-Lin Tsai

Hung-En Wang

Thanks for Your Attention!

- Questions?