A Symbolic Algorithm for the Case-Split Rule in String Constraint Solving

Yu-Fang Chen [1] **Vojtěch Havlena** [2]
Ondřej Lengál [2] Andrea Turrini [3]

Academia Sinica, Taiwan

Brno University of Technology, Czech Republic

Institute of Software, Chinese Academy of Sciences, China

30 November 2020 (APLAS'20)

String Constraints – Motivation

- SMT solving
  - FO logic fragments combining various theories
  - integers, reals, arrays, strings

String Constraints – Motivation

- SMT solving
  - FO logic fragments combining various theories
  - integers, reals, arrays, strings
- Analysis and verification of programs
  - symbolic execution; concolic testing
  - vulnerabilities in web applications (XSS)
    - bad strings manipulation
  - test-case generation

String Constraints

- Equations containing string variables from $\mathbb{X}$ ranging over Σ^*
- Syntax $\Sigma_{\mathbb{X}} = \mathbb{X} \cup \Sigma$

(equation) $\psi ::= t_\ell = t_r$ where $t_\ell, t_r \in \Sigma_{\mathbb{X}}^*$

(string constraint) $\varphi ::= \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg\varphi \mid \psi$

String Constraints

- Equations containing string variables from $\mathbb{X}$ ranging over Σ^*
- Syntax $\qquad \Sigma_{\mathbb{X}} = \mathbb{X} \cup \Sigma$

$$\begin{array}{rll} \text{(equation)} & \psi ::= t_\ell = t_r & \text{where } t_\ell, t_r \in \Sigma_{\mathbb{X}}^* \\ \text{(string constraint)} & \varphi ::= \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg\varphi \mid \psi & \end{array}$$

- Assignment $I : \mathbb{X} \to \Sigma^*$; lifted to strings $I(\epsilon) = \epsilon$, $I(a) = a$, $I(xw) = I(x)I(w)$ for $a \in \Sigma$, $x \in \Sigma_{\mathbb{X}}$, $w \in \Sigma_{\mathbb{X}}^*$
- An assignment I is a model of φ, $I \models \varphi$, if I satisfies φ
  - $I \models t_\ell = t_r$ iff $I(t_\ell) = I(t_r)$
  - $I \models \varphi_1 \land \varphi_2, \varphi_1 \lor \varphi_2, \neg\varphi_1$ defined as usual
- φ is satisfiable if it has a model

String Constraints

- Equations containing string variables from $\mathbb{X}$ ranging over Σ^*
- Syntax $\Sigma_{\mathbb{X}} = \mathbb{X} \cup \Sigma$

$$\text{(equation)} \quad \psi ::= t_\ell = t_r \qquad \text{where } t_\ell, t_r \in \Sigma_{\mathbb{X}}^*$$
$$\text{(string constraint)} \quad \varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \psi$$

- Assignment $I : \mathbb{X} \to \Sigma^*$; lifted to strings $I(\epsilon) = \epsilon$, $I(a) = a$, $I(xw) = I(x)I(w)$ for $a \in \Sigma$, $x \in \Sigma_{\mathbb{X}}$, $w \in \Sigma_{\mathbb{X}}^*$
- An assignment I is a model of φ, $I \models \varphi$, if I satisfies φ
  - $I \models t_\ell = t_r$ iff $I(t_\ell) = I(t_r)$
  - $I \models \varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \neg\varphi_1$ defined as usual
- φ is satisfiable if it has a model

Example

- $xx = yaz \wedge aza = y \rightsquigarrow$ unsatisfiable (incompatible lengths)
- $ax = yy \rightsquigarrow$ satisfiable with e.g. $I(x) = waw$, $I(y) = aw$ for $w \in \Sigma^*$

Nielsen Transformation

- Conjunction of word equations φ; is φ satisfiable?

Nielsen Transformation

- Conjunction of word equations φ; is φ satisfiable?
- Building of a proof graph using rewriting rules

$$\frac{\alpha u = \alpha v}{u = v}\ \text{(trim)} \qquad \frac{xu = v}{u[x \mapsto \epsilon] = v[x \mapsto \epsilon]}\ (x \hookrightarrow \epsilon)$$

$$\frac{xu = \alpha v}{x(u[x \mapsto \alpha x]) = v[x \mapsto \alpha x]}\ (x \hookrightarrow \alpha x)$$

  - where $x \in \mathbb{X}$, $\alpha \in \Sigma_{\mathbb{X}}$, and $u, v \in \Sigma_{\mathbb{X}}^*$

Nielsen Transformation

- Conjunction of word equations φ; is φ satisfiable?
- Building of a proof graph using rewriting rules

$$\frac{\alpha u = \alpha v}{u = v}\ \text{(trim)} \qquad \frac{xu = v}{u[x \mapsto \epsilon] = v[x \mapsto \epsilon]}\ (x \hookrightarrow \epsilon)$$

$$\frac{xu = \alpha v}{x(u[x \mapsto \alpha x]) = v[x \mapsto \alpha x]}\ (x \hookrightarrow \alpha x)$$

  - where $x \in \mathbb{X}$, $\alpha \in \Sigma_{\mathbb{X}}$, and $u, v \in \Sigma_{\mathbb{X}}^*$

- Starting from the node φ
- φ is satisfiable iff one of the leaf nodes is trivially valid, i.e, a node $\epsilon = \epsilon \wedge \cdots \wedge \epsilon = \epsilon$ is reachable from φ

Nielsen Transformation

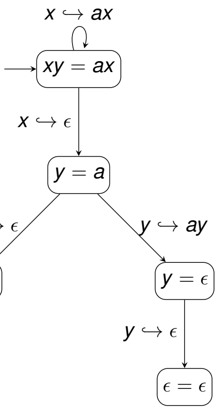
- Quadratic equations: at most two occurences of each variable
- Example: $\mathbb{X} = \{x, y\}, \Sigma = \{a, b\}$
  - $xay = xb \wedge y = b \quad \rightsquigarrow \quad$ quadratic
  - $xay = xb \wedge y = bx \quad \rightsquigarrow \quad$ not quadratic

Lemma (Makanin'77)

Nielsen transformation is sound. Moreover, it is complete when the systems of word equations is quadratic.

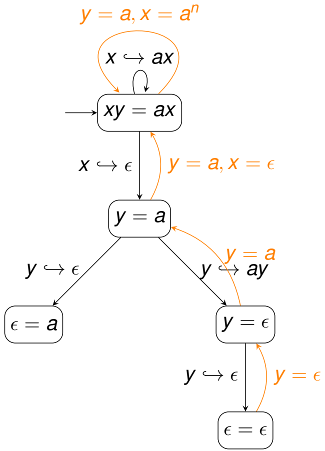
Nielsen Transformation *Cont.*

Example: Consider an equation ψ : $xy = ax$ (implicit trim)



Nielsen Transformation *Cont.*

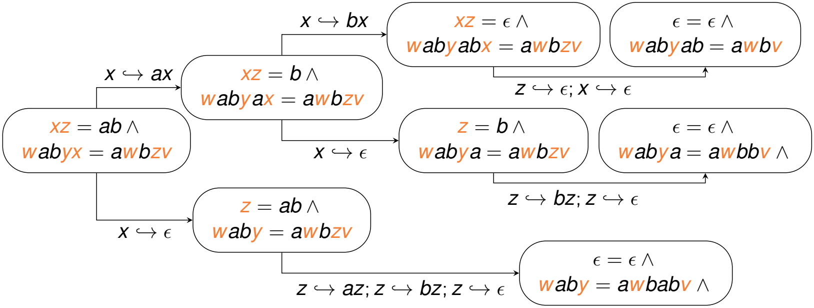
Example: Consider an equation ψ : $xy = ax$ (implicit trim)



- Model: $\{x \mapsto a^n, y \mapsto a\}$

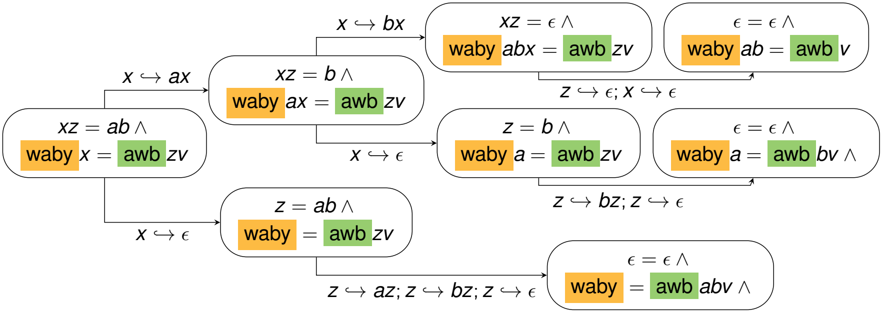
Proof Graphs

- Consider the equation $xz = ab \wedge wabyx = awbzv$
- Partial Nielsen proof graph (implicit trim; omitted contradictions)



Proof Graphs

- Consider the equation $xz = ab \land wabyx = awbzv$
- Partial Nielsen proof graph (implicit trim; omitted contradictions)



A lot of nodes have shared parts. Could we do it better?

Towards an Efficient Handling of the Proof Graph

- A string equation $\psi : t_\ell = t_r$ can be seen as a pair (t_ℓ, t_r)

Towards an Efficient Handling of the Proof Graph

- A string equation $\psi : t_\ell = t_r$ can be seen as a pair (t_ℓ, t_r)
- Nielsen rules are then word operations
  - *trim* $\rightsquigarrow$ τ_{trim} (trim as much as possible)
  - $x \hookrightarrow \epsilon$ $\rightsquigarrow$ $\tau_{x \mapsto \epsilon}$
  - $x \hookrightarrow \alpha x$ $\rightsquigarrow$ $\tau_{x \mapsto \alpha x}$

Towards an Efficient Handling of the Proof Graph

- A string equation $\psi : t_\ell = t_r$ can be seen as a pair (t_ℓ, t_r)
- Nielsen rules are then word operations
  - *trim* $\rightsquigarrow$ τ_{trim} (trim as much as possible)
  - $x \hookrightarrow \epsilon$ $\rightsquigarrow$ $\tau_{x \mapsto \epsilon}$
  - $x \hookrightarrow \alpha x$ $\rightsquigarrow$ $\tau_{x \mapsto \alpha x}$
- Example: Consider an equation $\psi : zb = ua$ with $\mathbb{X} = \{z, u\}$
  - $\tau_{z \mapsto uz}(zb, ua) = \{(uzb, ua)\}$
  - $\tau_{u \mapsto \epsilon}(zb, ua) = \{(zb, a)\}$

Towards an Efficient Handling of the Proof Graph

- A string equation $\psi : t_\ell = t_r$ can be seen as a pair (t_ℓ, t_r)
- Nielsen rules are then word operations
  - *trim* $\rightsquigarrow$ τ_{trim} (trim as much as possible)
  - $x \hookrightarrow \epsilon$ $\rightsquigarrow$ $\tau_{x \mapsto \epsilon}$
  - $x \hookrightarrow \alpha x$ $\rightsquigarrow$ $\tau_{x \mapsto \alpha x}$
- Example: Consider an equation $\psi : zb = ua$ with $\mathbb{X} = \{z, u\}$
  - $\tau_{z \mapsto uz}(zb, ua) = \{(uzb, ua)\}$
  - $\tau_{u \mapsto \epsilon}(zb, ua) = \{(zb, a)\}$
- Apply multiple word operations on a set of equations at once
  - $\langle v \mapsto \alpha v \rangle = \bigcup_{x \in \mathbb{X}, \alpha \in \Sigma_{\mathbb{X}}} (\tau_{trim} \circ \tau_{x \mapsto \alpha x})$
  - $\langle v \mapsto \epsilon \rangle = \bigcup_{x \in \mathbb{X}} (\tau_{trim} \circ \tau_{x \mapsto \epsilon})$

Towards an Efficient Handling of the Proof Graph

- A string equation $\psi : t_\ell = t_r$ can be seen as a pair (t_ℓ, t_r)
- Nielsen rules are then word operations
  - $trim \rightsquigarrow \tau_{trim}$ (trim as much as possible)
  - $x \hookrightarrow \epsilon \rightsquigarrow \tau_{x \mapsto \epsilon}$
  - $x \hookrightarrow \alpha x \rightsquigarrow \tau_{x \mapsto \alpha x}$
- Example: Consider an equation $\psi : zb = ua$ with $\mathbb{X} = \{z, u\}$
  - $\tau_{z \mapsto uz}(zb, ua) = \{(uzb, ua)\}$
  - $\tau_{u \mapsto \epsilon}(zb, ua) = \{(zb, a)\}$

- Apply multiple word operations on a set of equations at once
  - $\langle v \mapsto \alpha v \rangle = \bigcup_{x \in \mathbb{X}, \alpha \in \Sigma_{\mathbb{X}}} (\tau_{trim} \circ \tau_{x \mapsto \alpha x})$
  - $\langle v \mapsto \epsilon \rangle = \bigcup_{x \in \mathbb{X}} (\tau_{trim} \circ \tau_{x \mapsto \epsilon})$

- Example: Consider an equation $\psi : xay = yx$ with $\mathbb{X} = \{x, y\}$
  - $\langle v \mapsto \alpha v \rangle (xay, yx) = \{(xay, yx), (axy, yx)\}$ $(\tau_{x \mapsto yx}, \tau_{y \mapsto xy} + \tau_{trim})$
  - $\langle v \mapsto \epsilon \rangle (xay, yx) = \{(ay, y), (a, \epsilon)\}$ $(\tau_{x \mapsto \epsilon}, \tau_{y \mapsto \epsilon} + \tau_{trim})$

Towards an Efficient Handling of the Proof Graph *Cont.*

- Equation $\psi : t_\ell = t_r$
- BFS strategy of the proof graph generation
  - initial node $\mathcal{I} = \{(t_\ell, t_r)\}$
  - transformation function $\mathcal{T} = \langle v \mapsto \alpha v \rangle \cup \langle v \mapsto \epsilon \rangle$
    $\rightsquigarrow$ single step in a proof graph for all nodes
  - destination $\mathcal{D} = \{(\epsilon, \epsilon)\}$

Towards an Efficient Handling of the Proof Graph *Cont.*

- Equation $\psi : t_\ell = t_r$
- BFS strategy of the proof graph generation
  - initial node $\mathcal{I} = \{(t_\ell, t_r)\}$
  - transformation function $\mathcal{T} = \langle v \mapsto \alpha v \rangle \cup \langle v \mapsto \epsilon \rangle$
    $\rightsquigarrow$ single step in a proof graph for all nodes
  - destination $\mathcal{D} = \{(\epsilon, \epsilon)\}$
  - Compute $\mathcal{T}^0(\mathcal{I}), \mathcal{T}^1(\mathcal{I}), \mathcal{T}^2(\mathcal{I}), \dots$ until
    - $(\epsilon, \epsilon) \in \mathcal{T}^n(\mathcal{I}) \quad \rightsquigarrow \quad \psi$ is satisfiable
    - $\bigcup_{0 \leq i < n} \mathcal{T}^i(\mathcal{I}) \supseteq \mathcal{T}^n(\mathcal{I}) \quad \rightsquigarrow \quad \psi$ is unsatisfiable

Towards an Efficient Handling of the Proof Graph *Cont.*

- Equation $\psi : t_\ell = t_r$
- BFS strategy of the proof graph generation
  - initial node $\mathcal{I} = \{(t_\ell, t_r)\}$
  - transformation function $\mathcal{T} = \langle v \mapsto \alpha v \rangle \cup \langle v \mapsto \epsilon \rangle$
    $\rightsquigarrow$ single step in a proof graph for all nodes
  - destination $\mathcal{D} = \{(\epsilon, \epsilon)\}$
  - Compute $\mathcal{T}^0(\mathcal{I}), \mathcal{T}^1(\mathcal{I}), \mathcal{T}^2(\mathcal{I}), \ldots$ until
    - $(\epsilon, \epsilon) \in \mathcal{T}^n(\mathcal{I}) \quad \rightsquigarrow \quad \psi$ is satisfiable
    - $\bigcup_{0 \leq i < n} \mathcal{T}^i(\mathcal{I}) \supseteq \mathcal{T}^n(\mathcal{I}) \quad \rightsquigarrow \quad \psi$ is unsatisfiable
  - Regular model checking framework
    - verification of systems; can we reach a bad state?
    - $\exists n \in \mathbb{N} : \mathcal{T}^n(\mathcal{I}) \cap \mathit{Bad} \neq \emptyset$

Symbolic Encoding

- Encoding ensuring regular languages and rational relations

Symbolic Encoding

- Encoding ensuring regular languages and rational relations
- Encoding of a string equation $\psi : a_1 \ldots a_n = b_1 \ldots b_m$ $(n \geq m)$

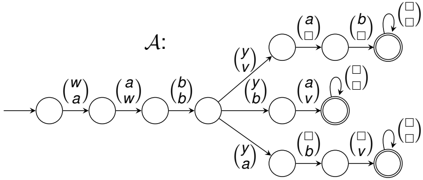
$$enc(\psi) = \binom{a_1}{b_1}\binom{a_2}{b_2}\ldots\binom{a_m}{b_m}\binom{a_{m+1}}{\square}\ldots\binom{a_n}{\square}\binom{\square}{\square}^*$$

Symbolic Encoding

- Encoding ensuring regular languages and rational relations
- Encoding of a string equation $\psi : a_1 \dots a_n = b_1 \dots b_m$ $(n \geq m)$

$$enc(\psi) = \binom{a_1}{b_1}\binom{a_2}{b_2}\dots\binom{a_m}{b_m}\binom{a_{m+1}}{\square}\dots\binom{a_n}{\square}\binom{\square}{\square}^*$$

  - Example: Equations
    wab $yab =$ awb v,
    wab $ya =$ awb bv,
    wab $y =$ awb $abv \rightsquigarrow$ repr.
    by a finite automaton $\mathcal{A}$

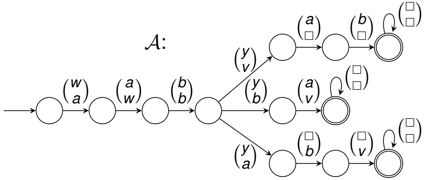


Symbolic Encoding

- Encoding ensuring regular languages and rational relations
- Encoding of a string equation $\psi : a_1 \dots a_n = b_1 \dots b_m$ $(n \geq m)$

$$enc(\psi) = \binom{a_1}{b_1}\binom{a_2}{b_2}\dots\binom{a_m}{b_m}\binom{a_{m+1}}{\square}\dots\binom{a_n}{\square}\binom{\square}{\square}^*$$

  - Example: Equations
    wab yab = awb v,
    wab ya = awb bv,
    wab y = awb abv $\rightsquigarrow$ repr.
    by a finite automaton $\mathcal{A}$



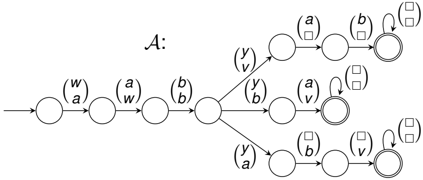
- Encoding can be straightforwardly extended to relations
  - $\mathcal{T}^{\leq i}_{v \mapsto \alpha v} = enc(\langle v \mapsto \alpha v \rangle_i)$ rewrites at most i occurrences

Symbolic Encoding

- Encoding ensuring regular languages and rational relations
- Encoding of a string equation $\psi : a_1 \dots a_n = b_1 \dots b_m$ $\qquad (n \geq m)$

$$enc(\psi) = \binom{a_1}{b_1}\binom{a_2}{b_2}\dots\binom{a_m}{b_m}\binom{a_{m+1}}{\square}\dots\binom{a_n}{\square}\binom{\square}{\square}^*$$

  - Example: Equations
    wab $yab =$ awb v,
    wab $ya =$ awb bv,
    wab $y =$ awb $abv \rightsquigarrow$ repr.
    by a finite automaton $\mathcal{A}$



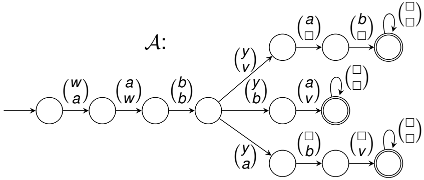
- Encoding can be straightforwardly extended to relations
  - $\mathcal{T}^{\leq i}_{v \mapsto \alpha v} = enc(\langle v \mapsto \alpha v \rangle_i)$ $\qquad$ rewrites at most i occurrences
  - Example: $\psi : xb = ab$, $x \hookrightarrow \epsilon$ transformation of ψ is encoded as
    $\left\{ \left(\binom{x}{a}\binom{b}{b}\binom{\square}{\square}^k, \binom{b}{a}\binom{\square}{b}\binom{\square}{\square}^\ell \right) \mid k, \ell \in \mathbb{N} \right\}$

Symbolic Encoding

- Encoding ensuring regular languages and rational relations
- Encoding of a string equation $\psi : a_1 \dots a_n = b_1 \dots b_m$ $(n \geq m)$

$$enc(\psi) = \binom{a_1}{b_1}\binom{a_2}{b_2}\dots\binom{a_m}{b_m}\binom{a_{m+1}}{\square}\dots\binom{a_n}{\square}\binom{\square}{\square}^*$$

  - Example: Equations
    wab $yab =$ awb v,
    wab $ya =$ awb bv,
    wab $y =$ awb $abv \rightsquigarrow$ repr.
    by a finite automaton $\mathcal{A}$



- Encoding can be straightforwardly extended to relations
  - $\mathcal{T}^{\leq i}_{v \mapsto \alpha v} = enc(\langle v \mapsto \alpha v \rangle_i)$ rewrites at most i occurrences
  - Example: $\psi : xb = ab$, $x \hookrightarrow \epsilon$ transformation of ψ is encoded as
    $\left\{ \left(\binom{x}{a}\binom{b}{b}\binom{\square}{\square}^k, \binom{b}{a}\binom{\square}{b}\binom{\square}{\square}^\ell \right) \mid k, \ell \in \mathbb{N} \right\}$
- The relations $\mathcal{T}^{\leq i}_{v \mapsto \alpha v}$ and $\mathcal{T}^{\leq i}_{v \mapsto \epsilon}$ are rational.

Symbolic Algorithm

- Satisfiability checking of a quadratic equation ψ

Symbolic Algorithm

- Satisfiability checking of a quadratic equation ψ
- Regular model checking framework
  - $\mathcal{T}_\psi = \mathcal{T}^{\leq 2}_{v \mapsto \alpha v} \cup \mathcal{T}^{\leq 2}_{v \mapsto \epsilon}$ $\rightsquigarrow$ rational language
  - $\mathcal{I}_\psi = \mathit{enc}(\psi)$ $\rightsquigarrow$ regular language
  - $\mathcal{D}_\psi = \binom{\square}{\square}^*$ $\rightsquigarrow$ regular language

Symbolic Algorithm

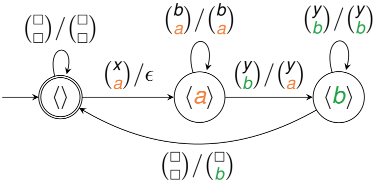
- Satisfiability checking of a quadratic equation ψ
- Regular model checking framework
  - $\mathcal{T}_\psi = \mathcal{T}^{\leq 2}_{v \mapsto \alpha v} \cup \mathcal{T}^{\leq 2}_{v \mapsto \epsilon} \quad \rightsquigarrow \quad$ rational language
  - $\mathcal{I}_\psi = \mathit{enc}(\psi) \quad \rightsquigarrow \quad$ regular language
  - $\mathcal{D}_\psi = \binom{\square}{\square}^* \quad \rightsquigarrow \quad$ regular language
- Efficiently repr. by automata and length-preserving transducers

Symbolic Algorithm

- Satisfiability checking of a quadratic equation ψ
- Regular model checking framework
  - $\mathcal{T}_\psi = \mathcal{T}^{\leq 2}_{v \mapsto \alpha v} \cup \mathcal{T}^{\leq 2}_{v \mapsto \epsilon}$ $\rightsquigarrow$ rational language
  - $\mathcal{I}_\psi = enc(\psi)$ $\rightsquigarrow$ regular language
  - $\mathcal{D}_\psi = \binom{\square}{\square}^*$ $\rightsquigarrow$ regular language
- Efficiently repr. by automata and length-preserving transducers
- Example: $\psi : xbyy = aabb$; part of $\mathcal{T}^{\leq 1}_{v \mapsto \epsilon}$ for $\mathbb{X} = \{x, y\}, \Sigma = \{a, b\}$

(a lot of transitions and states are omitted)

$$\binom{x}{a}\binom{b}{a}\binom{y}{b}\binom{y}{b}$$
$$\rightsquigarrow$$
$$\binom{b}{a}\binom{y}{a}\binom{y}{b}\binom{\square}{b}$$

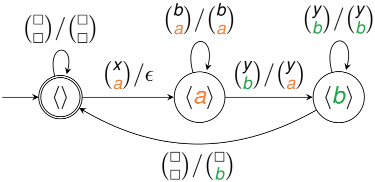


Symbolic Algorithm

- Satisfiability checking of a quadratic equation ψ
- Regular model checking framework
  - $\mathcal{T}_\psi = \mathcal{T}^{\leq 2}_{v \mapsto \alpha v} \cup \mathcal{T}^{\leq 2}_{v \mapsto \epsilon} \quad \rightsquigarrow \quad$ rational language
  - $\mathcal{I}_\psi = enc(\psi) \quad \rightsquigarrow \quad$ regular language
  - $\mathcal{D}_\psi = \binom{\square}{\square}^* \quad \rightsquigarrow \quad$ regular language
- Efficiently repr. by automata and length-preserving transducers
- Example: $\psi : xbyy = aabb$; part of $\mathcal{T}^{\leq 1}_{v \mapsto \epsilon}$ for $\mathbb{X} = \{x, y\}, \Sigma = \{a, b\}$

(a lot of transitions and states are omitted)

$$\binom{x}{a}\binom{b}{a}\binom{y}{b}\binom{y}{b} \rightsquigarrow \binom{b}{a}\binom{y}{a}\binom{y}{b}\binom{\square}{b}$$



- Sound; complete for quadratic equations
- Model obtained by a backward run and computing preimages

General String Constraints

Conjunction of quadratic equations $\Psi = \psi_1 \land \cdots \land \psi_n$

- Delimiter dividing encoded equations
- $enc(\Psi) = enc(\psi_1).\{\binom{\#}{\#}\}\ldots\{\binom{\#}{\#}\}.enc(\psi_n)$
- Modify $\mathcal{T}^{\leq 2}_{v \mapsto \epsilon}$ and $\mathcal{T}^{\leq 2}_{v \mapsto \alpha v}$ accordingly
- Sound and complete

- Example:
  - $xay = ya \land xb = z \land z = ba \quad \leadsto \quad$ quadratic
  - $xay = ya \land xb = z \land z = bx \quad \leadsto \quad$ cubic (3 occurrences of x)

General String Constraints

Conjunction of quadratic equations $\Psi = \psi_1 \wedge \cdots \wedge \psi_n$

- Delimiter dividing encoded equations
- $enc(\Psi) = enc(\psi_1).\{\binom{\#}{\#}\}\ldots\{\binom{\#}{\#}\}.enc(\psi_n)$
- Modify $\mathcal{T}^{\leq 2}_{v\mapsto\epsilon}$ and $\mathcal{T}^{\leq 2}_{v\mapsto\alpha v}$ accordingly
- Sound and complete
- Example:
  - $xay = ya \wedge xb = z \wedge z = ba \quad \rightsquigarrow \quad$ quadratic
  - $xay = ya \wedge xb = z \wedge z = bx \quad \rightsquigarrow \quad$ cubic (3 occurrences of x)

Conjunction of general equations Ψ

- Transformation of Ψ into equisatisfiable cubic system (using fresh variables)
- Integrate this transformation into $\mathcal{T}^{\leq 3}_{v\mapsto\alpha v}$ and $\mathcal{T}^{\leq 3}_{v\mapsto\epsilon}$
- Sound but incomplete

General String Constraints *Cont.*

Boolean combination of equations Ψ

- Inequalities: $t_\ell \neq t_r$; remove in a standard way using $\wedge, \vee$
  - $\bigvee_{c\in\Sigma}(t_\ell = t_r \cdot cx \vee t_\ell \cdot cx = t_r) \vee \bigvee_{c_1,c_2\in\Sigma,c_1\neq c_2}(t_\ell = x_3 c_1 x_1 \wedge t_r = x_3 c_2 x_2)$

General String Constraints *Cont.*

Boolean combination of equations Ψ

- Inequalities: $t_\ell \neq t_r$; remove in a standard way using $\wedge, \vee$
  - $\bigvee_{c \in \Sigma} (t_\ell = t_r \cdot cx \vee t_\ell \cdot cx = t_r) \vee \bigvee_{c_1, c_2 \in \Sigma, c_1 \neq c_2} (t_\ell = x_3 c_1 x_1 \wedge t_r = x_3 c_2 x_2)$
- Disjunction of equalities $\Delta = \psi_1 \vee \cdots \vee \psi_n$
  - $enc(\Delta) = \bigcup_{1 \leq i \leq n} enc(\psi_i)$

General String Constraints *Cont.*

Boolean combination of equations Ψ

- Inequalities: $t_\ell \neq t_r$; remove in a standard way using $\wedge, \vee$
  - $\bigvee_{c \in \Sigma}(t_\ell = t_r \cdot cx \vee t_\ell \cdot cx = t_r) \vee \bigvee_{c_1, c_2 \in \Sigma, c_1 \neq c_2}(t_\ell = x_3 c_1 x_1 \wedge t_r = x_3 c_2 x_2)$
- Disjunction of equalities $\Delta = \psi_1 \vee \cdots \vee \psi_n$
  - $enc(\Delta) = \bigcup_{1 \leq i \leq n} enc(\psi_i)$

1. Remove inequalities from Ψ

General String Constraints *Cont.*

Boolean combination of equations Ψ

- Inequalities: $t_\ell \neq t_r$; remove in a standard way using $\wedge, \vee$
  - $\bigvee_{c \in \Sigma}(t_\ell = t_r \cdot cx \vee t_\ell \cdot cx = t_r) \vee \bigvee_{c_1, c_2 \in \Sigma, c_1 \neq c_2}(t_\ell = x_3 c_1 x_1 \wedge t_r = x_3 c_2 x_2)$
- Disjunction of equalities $\Delta = \psi_1 \vee \cdots \vee \psi_n$
  - $enc(\Delta) = \bigcup_{1 \leq i \leq n} enc(\psi_i)$

1. Remove inequalities from Ψ
2. Convert to CNF $\Psi' = \Delta_1 \wedge \cdots \wedge \Delta_k$
   - No Tseitin $\rightsquigarrow$ introduces negations

General String Constraints *Cont.*

Boolean combination of equations Ψ

- Inequalities: $t_\ell \neq t_r$; remove in a standard way using $\wedge, \vee$
  - $\bigvee_{c\in\Sigma}(t_\ell = t_r \cdot cx \vee t_\ell \cdot cx = t_r) \vee \bigvee_{c_1,c_2\in\Sigma,c_1\neq c_2}(t_\ell = x_3 c_1 x_1 \wedge t_r = x_3 c_2 x_2)$
- Disjunction of equalities $\Delta = \psi_1 \vee \cdots \vee \psi_n$
  - $enc(\Delta) = \bigcup_{1\leq i\leq n} enc(\psi_i)$

1. Remove inequalities from Ψ
2. Convert to CNF $\Psi' = \Delta_1 \wedge \cdots \wedge \Delta_k$
   - No Tseitin $\rightsquigarrow$ introduces negations
3. $encode(\Psi') = enc(\Delta_1).\{\binom{\#}{\#}\}\ldots\{\binom{\#}{\#}\}.enc(\Delta_n)$

General String Constraints *Cont.*

Boolean combination of equations Ψ

- Inequalities: $t_\ell \neq t_r$; remove in a standard way using $\wedge, \vee$
  - $\bigvee_{c\in\Sigma}(t_\ell = t_r \cdot cx \vee t_\ell \cdot cx = t_r) \vee \bigvee_{c_1,c_2\in\Sigma, c_1\neq c_2}(t_\ell = x_3 c_1 x_1 \wedge t_r = x_3 c_2 x_2)$
- Disjunction of equalities $\Delta = \psi_1 \vee \cdots \vee \psi_n$
  - $enc(\Delta) = \bigcup_{1\leq i\leq n} enc(\psi_i)$

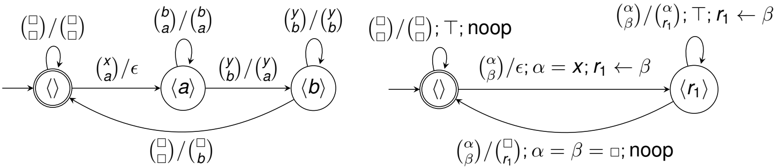
1. Remove inequalities from Ψ
2. Convert to CNF $\Psi' = \Delta_1 \wedge \cdots \wedge \Delta_k$
   - No Tseitin $\rightsquigarrow$ introduces negations
3. $encode(\Psi') = enc(\Delta_1).\{\binom{\#}{\#}\}\ldots\{\binom{\#}{\#}\}.enc(\Delta_n)$

- Sound but incomplete

Implementation

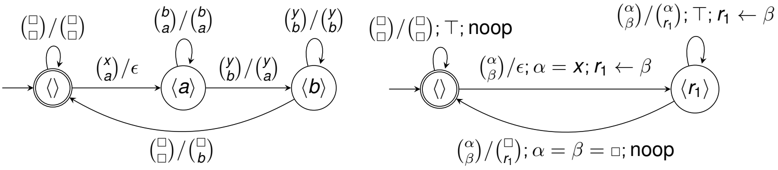
- Transducer $\mathcal{T}_\psi \rightsquigarrow$ branches for each choice of *x* and α
  $\rightsquigarrow$ huge transducer ($|Unicode| \geq 10^6$)
  - finite-alphabet register transducers
  - choice of *x* and α stored in registers, processed symbolically

- Transducer $\mathcal{T}_\psi \rightsquigarrow$ branches for each choice of x and α
  $\rightsquigarrow$ huge transducer $(|\mathit{Unicode}| \geq 10^6)$
  - finite-alphabet register transducers
  - choice of x and α stored in registers, processed symbolically
  - Example: Part of $\mathcal{T}^{\leq 1}_{x \mapsto \epsilon}$. Input variables: α, β, registers: r_1



Implementation

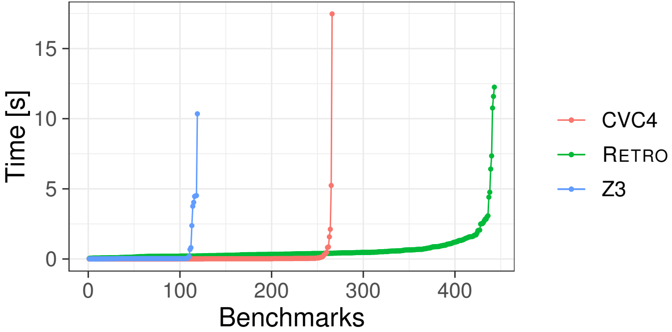
- Transducer $\mathcal{T}_\psi \rightsquigarrow$ branches for each choice of x and α
  $\rightsquigarrow$ huge transducer ($|Unicode| \geq 10^6$)
  - finite-alphabet register transducers
  - choice of x and α stored in registers, processed symbolically
  - Example: Part of $\mathcal{T}^{\leq 1}_{x \mapsto \epsilon}$. Input variables: α, β, registers: r_1



- Encoded proof graph represented by deterministic finite automata
  - eager minimization
  - on-the-fly language inclusion checking

Experimental Evaluation

1. `Kepler`$_{22}$ benchmark [LeHe'18]
   - 600 hand-crafted hard quadratic equations

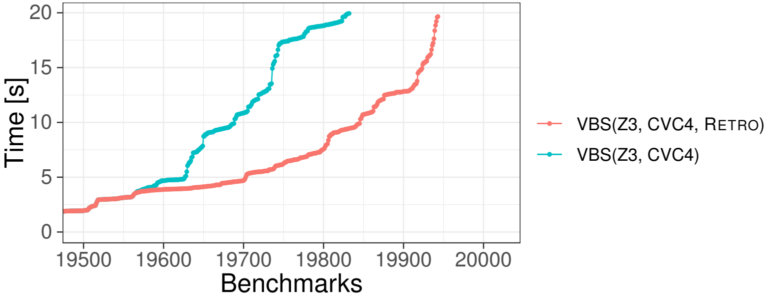


- Z3: 119 CVC4: 266 RETRO: 443 (TO 20 sec)

Experimental Evaluation *Cont.*

2 Conjunctions extracted from 967 difficult instances of PyEx
- symbolic execution of Python programs
- leads to 20,020 conjunctions of equations

- Z3: 16,788 CVC4: 19,823 Retro: 16,921 (TO 20 sec)
- orthogonal approach: Retro solved 82% instances where Z3 failed and 54% instances where CVC4 failed



Conclusion

- RMC framework for solving of word equations
  - efficiently implemented by register automata

- Future work
  - extensions with efficient handling of length and regular constraints
  - encoding of Makanin's algorithm (sound and complete for arbitrary equations)

Conclusion

- RMC framework for solving of word equations
  - efficiently implemented by register automata

- Future work
  - extensions with efficient handling of length and regular constraints
  - encoding of Makanin's algorithm (sound and complete for arbitrary equations)

THANK YOU!