Word Equations in Synergy with Regular Constraints
(based on FM'23 and OOPSLA'23 papers)

František Blahoudek[1], Yu-Fang Chen[2], David Chocholatý[1], Vojtěch Havlena[1], Lukáš Holík[1], **Ondřej Lengál[1]**, and Juraj Síč[1]

[1]Faculty of Information Technology, Brno University of Technology, Czech Republic
[2]Institute of Information Science, Academia Sinica, Taiwan

MOSCA'23

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^*a^+(b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \wedge \underbrace{\mathsf{contains}(u, \mathsf{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- String manipulation in programs
  - source of security vulnerabilities
  - scripting languages rely heavily on strings
- Analysis of AWS access policies
- ...

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \ \land \ y \neq u}_{\text{(in)equations}} \land \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \land \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \ \land \ \underbrace{\mathrm{contains}(u, \mathrm{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \ \land \ y \in a^+ b^+ \ \land \ z \in b^* \ \land \ x \in a^*$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^*a^+(b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\mathrm{contains}(u, \mathrm{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \;\wedge\; y \in a^+b^+ \;\wedge\; z \in b^* \;\wedge\; x \in a^*$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\text{contains}(u, \text{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \;\wedge\; y \in a^+ b^+ \;\wedge\; z \in b^+ \;\wedge\; x \in a^*$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\text{contains}(u, \text{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \;\wedge\; y \in a^+b^+ \;\wedge\; z \in b^+ \;\wedge\; x \in a^*$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\text{contains}(u, \text{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \;\wedge\; y \in a^+b^+ \;\wedge\; z \in b^+ \;\wedge\; x \in a^*$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\mathrm{contains}(u, \mathrm{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \;\wedge\; y \in a^+ b^+ \;\wedge\; z \in b^+ \;\wedge\; x = \epsilon$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\mathrm{contains}(u, \mathrm{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zy \;=\; z \;\wedge\; y \in a^+ b^+ \;\wedge\; z \in b^+$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)

String solving

- Satisfiability of formulas over string constraints such as:

$$\underbrace{x = yz \;\wedge\; y \neq u}_{\text{(in)equations}} \wedge \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \wedge \overbrace{|x| = 2|u| + 1}^{\text{length constraints}} \;\wedge\; \underbrace{\mathrm{contains}(u, \mathrm{replaceAll}(z, b, c))}_{\text{more complex operations}}$$

- A source of difficulty: equations with regular constraints
- Example: $zy = z \wedge y \in a^+b^+ \wedge z \in b^+$
  - results in an infinite case split
  - leads to failure for all current solvers (except ours!)
  - it is UNSAT

Our Approach

- Decision procedure tightly integrating regular constraints with equations
- Gradually refines regular constraints according to equations until:
  - an infeasible constraint is generated or
  - refinement becomes **stable**
- Complete on the chain-free fragment [AbdullaADHJ'19]
  - largest known decidable fragment for equations, regular, transducer, and length constraints
- Prototype tool Z3-Noodler
  - extension of Z3
  - competitive to existing solvers

Example

$$xyx = zu \ \wedge \ ww = xa \ \wedge \ u \in (baba)^*a \ \wedge \ z \in a(ba)^* \ \wedge \ x \in \Sigma^* \ \wedge \ y \in \Sigma^* \ \wedge \ w \in \Sigma^*$$

- $\Sigma = \{a, b\}$

Example

$$xyx = zu \ \land \ ww = xa \ \land \ u \in (baba)^*a \ \land \ z \in a(ba)^* \ \land \ x \in \Sigma^* \ \land \ y \in \Sigma^* \ \land \ w \in \Sigma^*$$

- $\Sigma = \{a, b\}$
- Use equations to refine regular constraints

Example

$$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^* a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$$

- $\Sigma = \{a, b\}$
- Use equations to refine regular constraints
- Start with $xyx = zu$

Example

$$xyx = zu \;\land\; ww = xa \;\land\; u \in (baba)^*a \;\land\; z \in a(ba)^* \;\land\; x \in \Sigma^* \;\land\; y \in \Sigma^* \;\land\; w \in \Sigma^*$$

- $\Sigma = \{a, b\}$
- Use equations to refine regular constraints
- Start with $xyx = zu$
- For any solution ν, the string $s = \nu(x) \cdot \nu(y) \cdot \nu(x) = \nu(z) \cdot \nu(u)$ satisfies

$$s \in \overbrace{\Sigma^*}^{x}\overbrace{\Sigma^*}^{y}\overbrace{\Sigma^*}^{x} \overset{=}{\cap} \overbrace{a(ba)^*}^{z}\overbrace{(baba)^*a}^{u}$$

Example

$$xyx = zu \ \wedge \ ww = xa \ \wedge \ u \in (baba)^*a \ \wedge \ z \in a(ba)^* \ \wedge \ x \in \Sigma^* \ \wedge \ y \in \Sigma^* \ \wedge \ w \in \Sigma^*$$

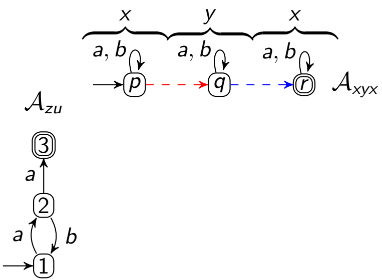
- $\Sigma = \{a, b\}$
- Use equations to refine regular constraints
- Start with $xyx = zu$
- For any solution ν, the string $s = \nu(x) \cdot \nu(y) \cdot \nu(x) = \nu(z) \cdot \nu(u)$ satisfies

$$s \in \overbrace{\Sigma^*}^{x}\overbrace{\Sigma^*}^{y}\overbrace{\Sigma^*}^{x} \ \overset{=}{\cap} \ \overbrace{a(ba)^*}^{z}\overbrace{(baba)^*a}^{u}$$

- Refine x, y from the left-hand side xyx using special intersection

Intersection with epsilon transitions [FM'23]

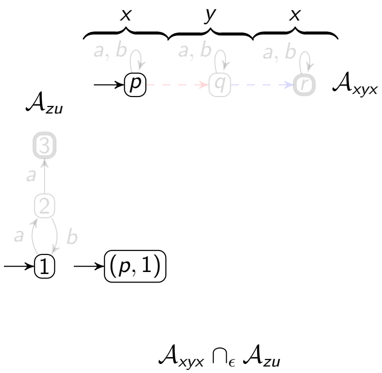
$xyx = zu \ \land \ ww = xa \ \land \ u \in (baba)^*a \ \land \ z \in a(ba)^* \ \land \ x \in \Sigma^* \ \land \ y \in \Sigma^* \ \land \ w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions

Intersection with epsilon transitions [FM'23]

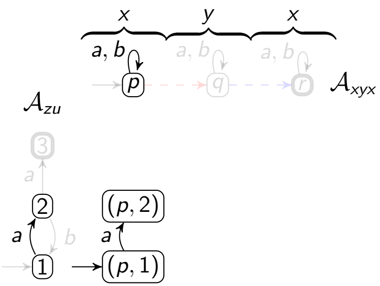
$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction

Intersection with epsilon transitions [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$

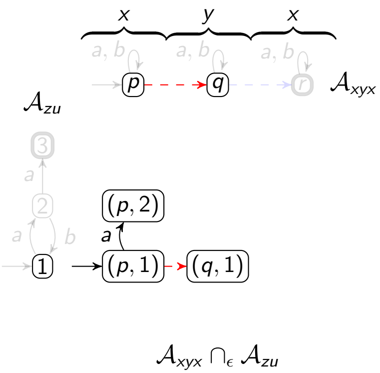


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction

Intersection with epsilon transitions [FM'23]

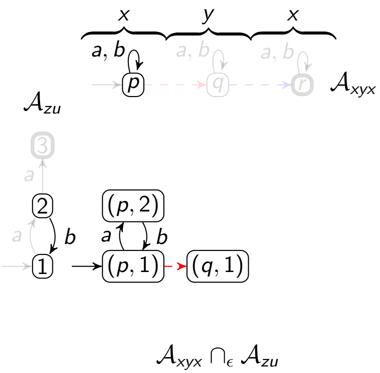
$xyx = zu \;\land\; ww = xa \;\land\; u \in (baba)^*a \;\land\; z \in a(ba)^* \;\land\; x \in \Sigma^* \;\land\; y \in \Sigma^* \;\land\; w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

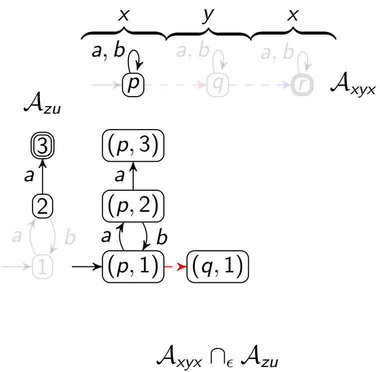
$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

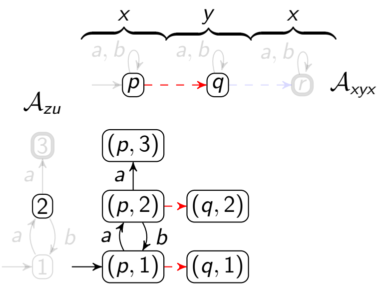
$xyx = zu \ \land \ ww = xa \ \land \ u \in (baba)^* a \ \land \ z \in a(ba)^* \ \land \ x \in \Sigma^* \ \land \ y \in \Sigma^* \ \land \ w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^* a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$

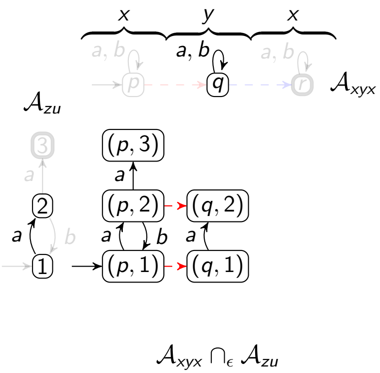


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

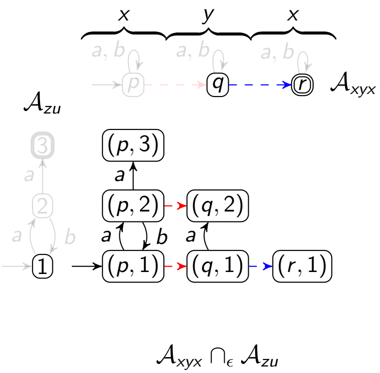
$xyx = zu \ \wedge \ ww = xa \ \wedge \ u \in (baba)^*a \ \wedge \ z \in a(ba)^* \ \wedge \ x \in \Sigma^* \ \wedge \ y \in \Sigma^* \ \wedge \ w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

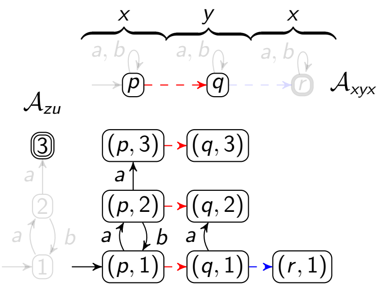
$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^* a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$

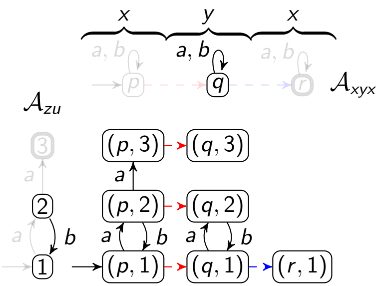


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

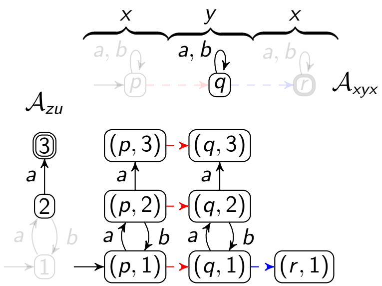
$xyx = zu \ \wedge\ ww = xa \ \wedge\ u \in (baba)^*a \ \wedge\ z \in a(ba)^* \ \wedge\ x \in \Sigma^* \ \wedge\ y \in \Sigma^* \ \wedge\ w \in \Sigma^*$



$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

$xyx = zu \ \wedge \ ww = xa \ \wedge \ u \in (baba)^*a \ \wedge \ z \in a(ba)^* \ \wedge \ x \in \Sigma^* \ \wedge \ y \in \Sigma^* \ \wedge \ w \in \Sigma^*$

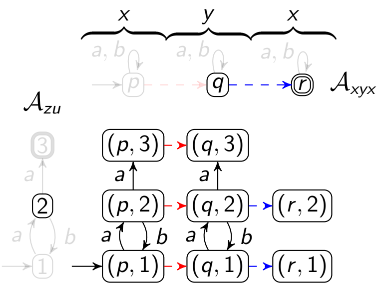


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$

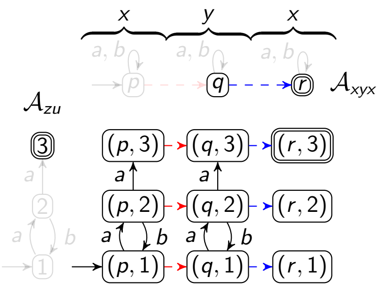


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \ \land \ ww = xa \ \land \ u \in (baba)^*a \ \land \ z \in a(ba)^* \ \land \ x \in \Sigma^* \ \land \ y \in \Sigma^* \ \land \ w \in \Sigma^*$

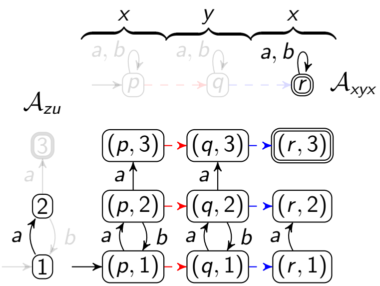


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$

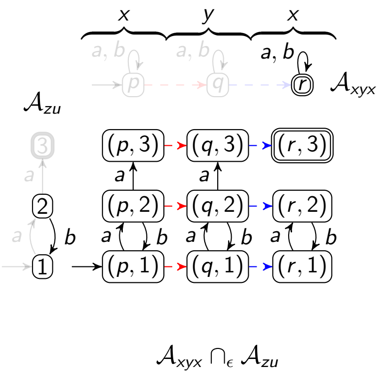


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

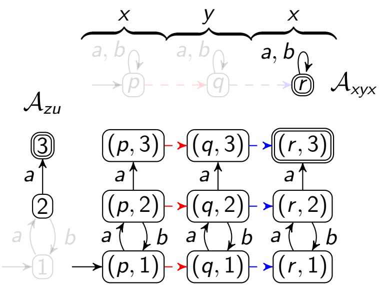
$xyx = zu \ \wedge\ ww = xa \ \wedge\ u \in (baba)^*a \ \wedge\ z \in a(ba)^* \ \wedge\ x \in \Sigma^* \ \wedge\ y \in \Sigma^* \ \wedge\ w \in \Sigma^*$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^* a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$

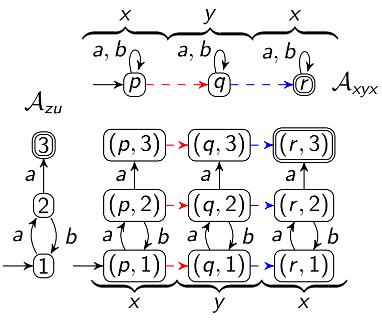


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions

Intersection with epsilon transitions [FM'23]

$xyx = zu \ \land \ ww = xa \ \land \ u \in (baba)^*a \ \land \ z \in a(ba)^* \ \land \ x \in \Sigma^* \ \land \ y \in \Sigma^* \ \land \ w \in \Sigma^*$

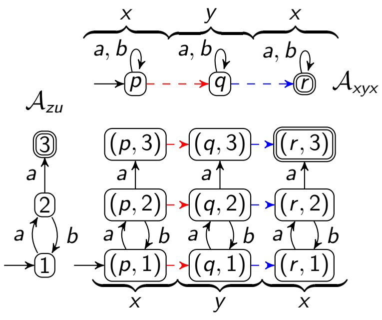


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side, $a(ba)^*a$, minimized
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions
- Construct intersection $\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$
  - synchronous product construction
  - keep ϵ transitions
- Variables x and y are nicely separated

Noodlification and unification [FM'23]

$\color{blue}{xyx = zu}$ $\wedge$ $\color{gray}{ww = xa \;\wedge}$ $u \in (baba)^* a$ $\wedge$ $z \in a(ba)^*$ $\wedge$ $x \in \Sigma^*$ $\wedge$ $y \in \Sigma^*$ $\wedge$ $\color{gray}{w \in \Sigma^*}$

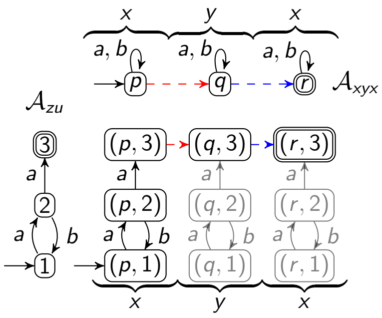


$\mathcal{A}_{xyx} \cap_{\epsilon} \mathcal{A}_{zu}$

- Split product into noodles
  - case split
  - values of y depend on values of x

Noodlification and unification [FM'23]

$$\color{blue}{xyx = zu} \;\land\; \color{gray}{ww = xa \;\land}\; u \in (baba)^*a \;\land\; z \in a(ba)^* \;\land\; x \in \Sigma^* \;\land\; y \in \Sigma^* \;\land\; \color{gray}{w \in \Sigma^*}$$

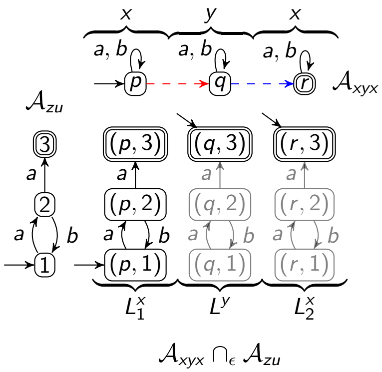


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Split product into noodles
  - case split
  - values of y depend on values of x

Noodlification and unification [FM'23]

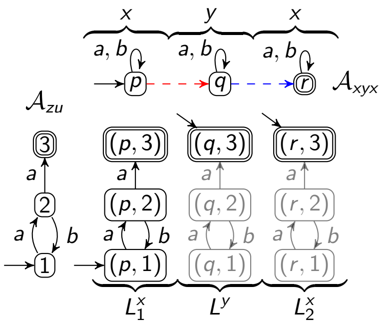
$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$



- Split product into noodles
  - case split
  - values of y depend on values of x
- Noodle languages:
  - $L_1^x = a(ba)^*a$
  - $L^y = \epsilon$
  - $L_2^x = \epsilon$

Noodlification and unification [FM'23]

$xyx = zu \ \wedge \ ww = xa \ \wedge \ u \in (baba)^*a \ \wedge \ z \in a(ba)^* \ \wedge \ x \in \Sigma^* \ \wedge \ y \in \Sigma^* \ \wedge \ w \in \Sigma^*$

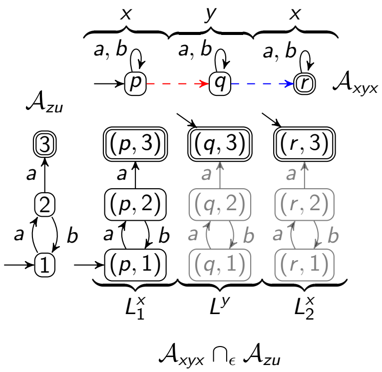


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Split product into noodles
  - case split
  - values of y depend on values of x
- Noodle languages:
  - $L_1^x = a(ba)^*a$
  - $L^y = \epsilon$
  - $L_2^x = \epsilon$
- Unification:
  - $\cap$ of langs for the same variable
  - $\text{Lang}(x) = L_1^x \cap L_2^x =$
  - $\text{Lang}(y) = L^y =$

Noodlification and unification [FM'23]

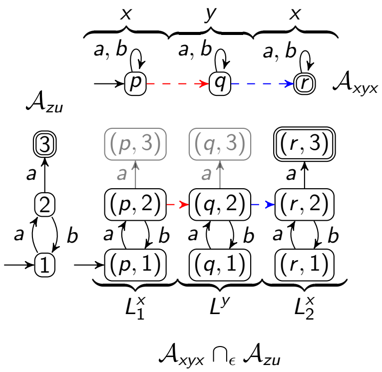
$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$



- Split product into noodles
  - case split
  - values of y depend on values of x
- Noodle languages:
  - $L_1^x = a(ba)^*a$
  - $L^y = \epsilon$
  - $L_2^x = \epsilon$
- Unification:
  - $\cap$ of langs for the same variable
  - $\mathrm{Lang}(x) = L_1^x \cap L_2^x = a(ba)^*a \cap \epsilon = \emptyset$
  - $\mathrm{Lang}(y) = L^y = \epsilon$

Noodlification and unification [FM'23]

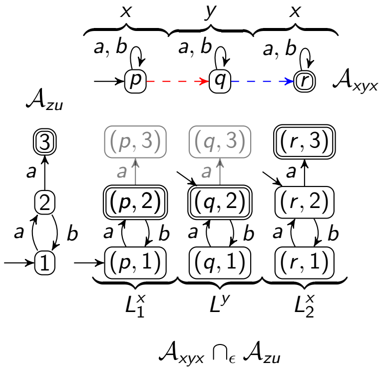
$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; w \in \Sigma^*$



- Split product into noodles
  - case split
  - values of y depend on values of x
- Noodle languages:
  - $L^x_1 =$
  - $L^y =$
  - $L^x_2 =$
- Unification:
  - $\cap$ of langs for the same variable
  - $\mathrm{Lang}(x) = L^x_1 \cap L^x_2 =$
  - $\mathrm{Lang}(y) = L^y =$

Noodlification and unification [FM'23]

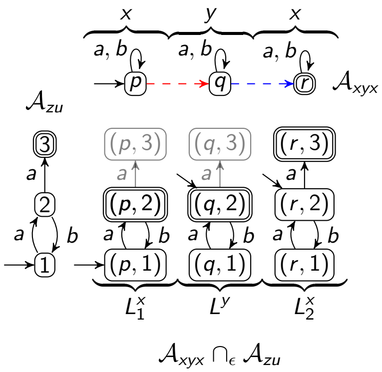
$\color{blue}{xyx = zu} \;\wedge\; \color{gray}{ww = xa} \;\wedge\; u \in (baba)^* a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in \Sigma^* \;\wedge\; y \in \Sigma^* \;\wedge\; \color{gray}{w \in \Sigma^*}$



- Split product into noodles
  - case split
  - values of y depend on values of x
- Noodle languages:
  - $L_1^x = a(ba)^*$
  - $L^y = (ba)^*$
  - $L_2^x = (ba)^* a$
- Unification:
  - $\cap$ of langs for the same variable
  - $\mathrm{Lang}(x) = L_1^x \cap L_2^x = a(ba)^* \cap (ba)^* a = \color{red}{a}$
  - $\mathrm{Lang}(y) = L^y = \color{red}{(ba)^*}$

Noodlification and unification [FM'23]

$xyx = zu \ \wedge \ ww = xa \ \wedge \ u \in (baba)^*a \ \wedge \ z \in a(ba)^* \ \wedge \ x \in a \ \wedge \ y \in (ba)^* \ \wedge \ w \in \Sigma^*$



- Split product into noodles
  - case split
  - values of y depend on values of x
- Noodle languages:
  - $L_1^x = a(ba)^*$
  - $L^y = (ba)^*$
  - $L_2^x = (ba)^*a$
- Unification:
  - $\cap$ of langs for the same variable
  - $\mathrm{Lang}(x) = L_1^x \cap L_2^x = a(ba)^* \cap (ba)^*a = a$
  - $\mathrm{Lang}(y) = L^y = (ba)^*$

Continuing [FM'23]

$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^*a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in a \;\wedge\; y \in (ba)^* \;\wedge\; w \in \Sigma^*$

- Refine further with $ww = xa$:

$$\overbrace{\Sigma^*}^{w}\;\overbrace{\Sigma^*}^{w} \;\overset{=}{\cap}\; \overbrace{a}^{x}\;\overset{a}{a}$$

Continuing [FM'23]

$$xyx = zu \;\wedge\; ww = xa \;\wedge\; u \in (baba)^* a \;\wedge\; z \in a(ba)^* \;\wedge\; x \in a \;\wedge\; y \in (ba)^* \;\wedge\; w \in a$$

- Refine further with $ww = xa$:

$$\overbrace{a}^{w}\;\overbrace{a}^{w} \overset{=}{\cap} \overbrace{a}^{x}\;\overset{a}{a}$$

Continuing [FM'23]

$$xyx = zu \ \land \ ww = xa \ \land \ u \in (baba)^* a \ \land \ z \in a(ba)^* \ \land \ x \in a \ \land \ y \in (ba)^* \ \land \ w \in a$$

- Refine further with $ww = xa$:

$$\overset{w}{\overbrace{a}} \ \overset{w}{\overbrace{a}} \ \overset{=}{\cap} \ \overset{x}{\overbrace{a}} \ \overset{a}{a}$$

- Languages in equations match:

$$\overset{x}{\overbrace{a}} \ \overset{y}{\overbrace{(ba)^*}} \ \overset{x}{\overbrace{a}} \ \overset{=}{=} \ \overset{z}{\overbrace{a(ba)^*}} \ \overset{u}{\overbrace{(baba)^* a}} \qquad \text{and} \qquad \overset{w}{\overbrace{a}} \ \overset{w}{\overbrace{a}} \ \overset{=}{=} \ \overset{x}{\overbrace{a}} \ \overset{a}{a}$$

Continuing [FM'23]

$$xyx = zu \ \land\ ww = xa \ \land\ u \in (baba)^*a \ \land\ z \in a(ba)^* \ \land\ x \in a \ \land\ y \in (ba)^* \ \land\ w \in a$$

- Refine further with $ww = xa$:

$$\overbrace{a}^{w}\ \overbrace{a}^{w} \ \overset{=}{\cap}\ \overbrace{a}^{x}\ \overset{a}{a}$$

- Languages in equations match:

$$\overbrace{a}^{x}\ \overbrace{(ba)^*}^{y}\ \overbrace{a}^{x} \ \overset{=}{=}\ \overbrace{a(ba)^*}^{z}\ \overbrace{(baba)^*a}^{u} \qquad \text{and} \qquad \overbrace{a}^{w}\ \overbrace{a}^{w} \ \overset{=}{=}\ \overbrace{a}^{x}\ \overset{a}{a}$$

- Because of stability (next slide), enough to decide SAT

Stability of equation system [FM'23]

- Single-equation system $\Phi\colon s = t \ \land \ \bigwedge_{x \in \mathbb{X}} x \in \mathrm{Lang}_\Phi(x)$ where $\mathrm{Lang}_\Phi\colon \mathbb{X} \to \mathcal{P}(\Sigma^*)$

System Φ is SAT iff there is refinement Lang of Lang_Φ where $\mathrm{Lang}(s) = \mathrm{Lang}(t)$.

Stability of equation system [FM'23]

- Single-equation system $\Phi\colon s = t \ \wedge\ \bigwedge_{x \in \mathbb{X}} x \in \mathrm{Lang}_\Phi(x)$ where $\mathrm{Lang}_\Phi\colon \mathbb{X} \to \mathcal{P}(\Sigma^*)$

System Φ is SAT iff there is refinement Lang of Lang_Φ where $\mathrm{Lang}(s) = \mathrm{Lang}(t)$.

- If all variables in t occur in $s = t$ exactly once:

System Φ is SAT iff there is refinement Lang of Lang_Φ where $\mathrm{Lang}(s) \subseteq \mathrm{Lang}(t)$.

Stability of equation system [FM'23]

- Single-equation system $\Phi\colon s = t \;\wedge\; \bigwedge_{x\in\mathbb{X}} x \in \mathrm{Lang}_\Phi(x)$ where $\mathrm{Lang}_\Phi\colon \mathbb{X} \to \mathcal{P}(\Sigma^*)$

System Φ is SAT iff there is refinement Lang of Lang_Φ where $\mathrm{Lang}(s) = \mathrm{Lang}(t)$.

- If all variables in t occur in $s = t$ exactly once:

System Φ is SAT iff there is refinement Lang of Lang_Φ where $\mathrm{Lang}(s) \subseteq \mathrm{Lang}(t)$.

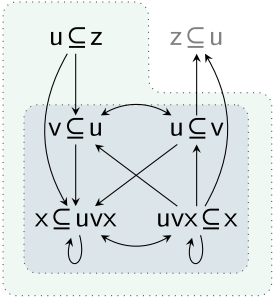
- Can be extended to multi-equation system

Inclusion Graph [FM'23]

Inclusion Graph:

- denotes how information should be propagated
- for each equation $s = t$, make two nodes: s⊆t and t⊆s
- Example:

$$u = z \quad \wedge \quad v = u \quad \wedge \quad x = uvx$$



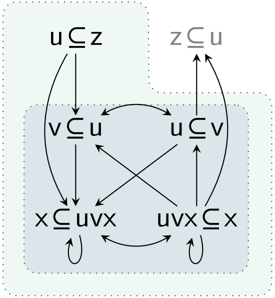
- We explore paths in the inclusion graph until all inclusions are satisfied, UNSAT, or T/O.

Inclusion Graph [FM'23]

Inclusion Graph:

- denotes how information should be propagated
- for each equation $s = t$, make two nodes: $s \subseteq t$ and $t \subseteq s$
- Example:

$$u = z \quad \wedge \quad v = u \quad \wedge \quad x = uvx$$



- We explore paths in the inclusion graph until all inclusions are satisfied, UNSAT, or T/O.

Chain-free equations & reg. constraints [AbdullaADHJ'19] (cf. their *splitting graph*):

Theorem

For chain-free constraint there exists an acyclic inclusion graph.

- $\rightsquigarrow$ completeness

Adding Length Constraints [OOPSLA'23]

$$\underbrace{x = yz}_{\text{equations}} \;\wedge\; \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \;\wedge\; \underbrace{|x| = 2|y| + 1}_{\text{length constraints}}$$

Length constraints:

- Needed for a tight integration within a DPLL(T) SMT solver
- Solved by translation of string solutions to a LIA formula
  - Each feasible branch of the computation tree outputs language assignment to string variables
  - Any combination of $w_x \in \mathrm{Lang}(x), w_y \in \mathrm{Lang}(y), \ldots$ is a solution (cf. *monadic decompos.*)
  - compute the Parikh image

Adding Length Constraints [OOPSLA'23]

$$\underbrace{x = yz}_{\text{equations}} \;\wedge\; \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \;\wedge\; \underbrace{|x| = 2|y| + 1}_{\text{length constraints}}$$

Length constraints:

- Needed for a tight integration within a DPLL(T) SMT solver
- Solved by translation of string solutions to a LIA formula
  - Each feasible branch of the computation tree outputs language assignment to string variables
  - Any combination of $w_x \in \mathrm{Lang}(x), w_y \in \mathrm{Lang}(y), \ldots$ is a solution (cf. *monadic decompos.*)
  - compute the Parikh image
- Variables appearing in length constraints need special handling

Adding Length Constraints [OOPSLA'23]

$$\underbrace{x = yz}_{\text{equations}} \;\wedge\; \overbrace{x \in (ab)^* a^+ (b|c)}^{\text{regular constraints}} \;\wedge\; \underbrace{|x| = 2|y| + 1}_{\text{length constraints}}$$

Length constraints:

- Needed for a tight integration within a DPLL(T) SMT solver
- Solved by translation of string solutions to a LIA formula
  - Each feasible branch of the computation tree outputs language assignment to string variables
  - Any combination of $w_x \in \mathrm{Lang}(x), w_y \in \mathrm{Lang}(y), \ldots$ is a solution (cf. *monadic decompos.*)
  - compute the Parikh image
- Variables appearing in length constraints need special handling
  - similar to the alignment procedure of Norn ...
  - ... but solve alignment only for length variables

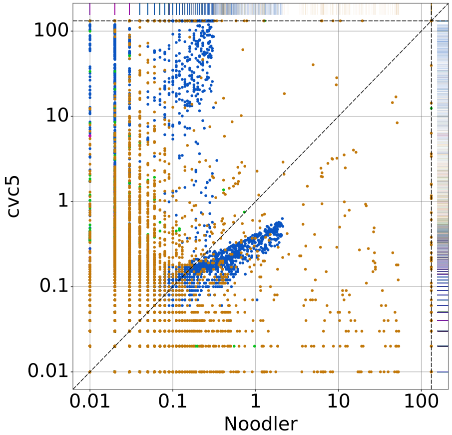
Experimental Evaluation [OOPSLA'23]

	Sygus-Qgen (343)					Norn (1 027)					Slent (1 128)				
	TOs	Es	Us	Time	Time−TOs	TOs	Es	Us	Time	Time−TOs	TOs	Es	Us	Time	Time−TOs
Z3-Noodler	**0**	**0**	**0**	**5.7**	**5.7**	**0**	**0**	**0**	**18.7**	**18.7**	**7**	**0**	**0**	982.3	142.3
CVC5	**0**	**0**	**0**	188.2	188.2	84	0	0	10 883.3	803.3	28	0	0	4 763.7	1 403.7
Z3	**0**	**0**	**0**	34.2	34.2	127	0	0	15 318.7	78.7	73	0	0	9 313.0	553.0
Z3str3RE	1	0	0	163.9	43.9	133	0	0	15 986.2	26.2	87	0	0	10 457.3	**17.3**
Z3-Trau	2	41	0	6 065.8	5 825.8				N/A		5	*53	4	**662.2**	62.2
Z3str4	**0**	**0**	**0**	65.9	65.9	75	0	0	9 113.6	113.6	77	0	0	9 271.5	31.5
OSTRICH	**0**	**0**	**0**	962.1	962.1	**0**	**0**	**0**	8 985.7	8 985.7	155	1	0	23 547.0	4 827.0

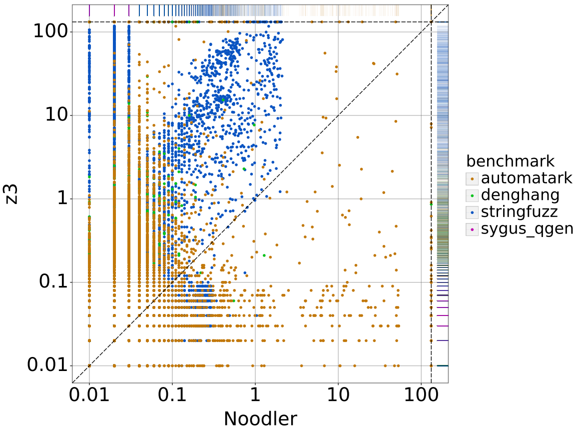
	Slog (1 976)					Leetcode (2 652)					Kaluza (19 432)				
	TOs	Es	Us	Time	Time−TOs	TOs	Es	Us	Time	Time−TOs	TOs	Es	Us	Time	Time−TOs
Z3-Noodler	**0**	**0**	**0**	36.2	36.2	35	0	0	4 779.2	579.2	192	0	0	24 226.9	1 186.9
CVC5	**0**	**0**	**0**	**12.1**	**12.1**	**0**	**0**	**0**	149.3	149.3	**6**	**0**	**0**	**1 914.4**	1 194.4
Z3	33	0	0	4 297.1	337.1	**0**	**0**	**0**	**142.4**	142.4	188	0	0	23 418.5	858.5
Z3str3RE	58	0	0	8 279.5	1 319.5	2	0	190	275.3	**35.3**	132	0	8	16 133.1	**293.1**
Z3-Trau	45	0	1	7 827.6	2 427.6	**0**	**0**	**0**	162.0	162.0	125	0	0	20 587.7	5 587.7
Z3str4	22	0	0	3 816.3	1 176.3	2	0	2	400.9	160.9	132	0	46	17 752.9	1 912.9
OSTRICH	6	*5	0	9 323.7	8 603.7	185	26	0	33 308.9	8 108.9	305	0	0	88 056.3	51 456.3

- T/Os = timeouts (120 s)
- time = total run time in seconds
- time−T/O = run time without timeouts
- best values are in **bold**

Comparison with CVC5 and Z3 on regex-heavy benchmarks

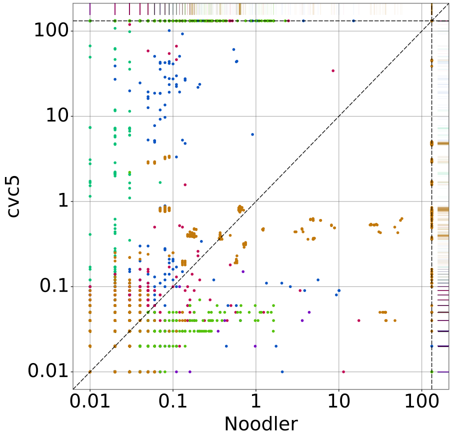


(a) Z3-Noodler vs. CVC5.

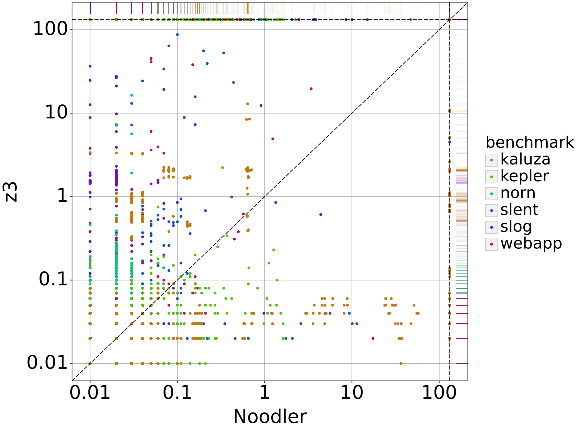


(b) Z3-Noodler vs. Z3.

Comparison with CVC5 and Z3 on equation-heavy benchmarks

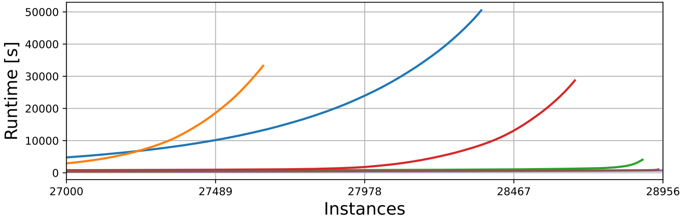


(a) Z3-Noodler vs. CVC5.

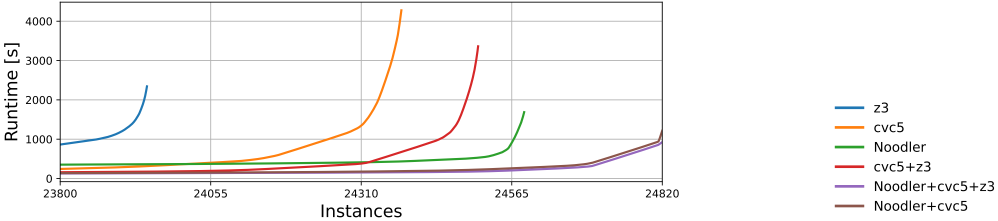


(b) Z3-Noodler vs. Z3.

Virtual Best Solver



(a) VBS on regex-heavy



(b) VBS on equation-heavy

Discussion

- Tight integration of word equations and regular constraints [FM'23].
- Extension to lengths and other predicates [OOPSLA'23].
- Can beat well established solvers
  - can solve more benchmarks
  - average time is low
- Often complementary to other solvers
- Preprocessing is important
- Need for efficient handling of automata $\rightsquigarrow$ efficient automata library (Mata)

Ongoing work

- Disequalities
  - can be rewritten to equations $\rightsquigarrow$ many disadvantages (breaking chain-freeness, etc.)
  - can be deferred to after stability and translated to LIA
- Transducers
- string $\leftrightarrow$ integer conversion
- other constraints