Word Equations in Synergy with Regular Constraints

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

String solving

- Satisfiability of formulas over string constraints such as:

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

- String manipulation in programs
  - source of security vulnerabilities
  - scripting languages rely heavily on strings
  - new examples of an intensive use of critical string operations

String solving

- Satisfiability of formulas over string constraints such as:

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

- 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:

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

- 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:

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

- 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:

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

- 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:

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

- 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:

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

- A source of difficulty: equations with regular constraints
- Example: $zyx = xxz \land y \in a^+b^+ \land z \in b^+ \land x = \epsilon$
  - 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:

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

- A source of difficulty: equations with regular constraints
- Example: $zy = z \land y \in a^+b^+ \land z \in b^+$
  - 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:

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

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

Our contribution

- Decision procedure tightly integrating regular constraints with equations
- Gradually refines languages until:
  - an infeasible constraint is generated or
  - refinement becomes **stable**
- Complete on chain-free fragment
  - largest known decidable fragment for equations, regular, transducer, and length constraints
  - terminates for all SAT instances
- Prototype tool Noodler
  - in Python
  - competitive with existing solvers

Example

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

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

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

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

- $\Sigma = \{a, b\}$
- Use equations to refine regular constraints
- Start with $xyx = zu$
- For any solution (assignment v) 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

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

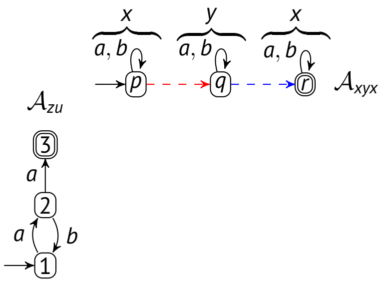
- $\Sigma = \{a, b\}$
- Use equations to refine regular constraints
- Start with $xyx = zu$
- For any solution (assignment ν) string $s = \nu(x) \cdot \nu(y) \cdot \nu(x) = \nu(z) \cdot \nu(u)$ satisfies:

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

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

Intersection with epsilon transitions

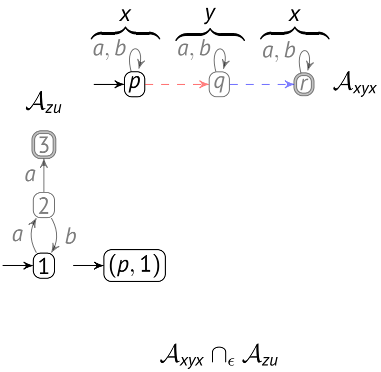
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – left side, keep ϵ transitions

Intersection with epsilon transitions

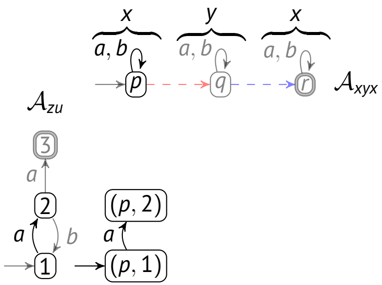
$$xyx = zu \land ww = xa \land u \in (baba)^*a \land z \in a(ba)^* \land x \in \Sigma^* \land y \in \Sigma^* \land w \in \Sigma^*$$



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

Intersection with epsilon transitions

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

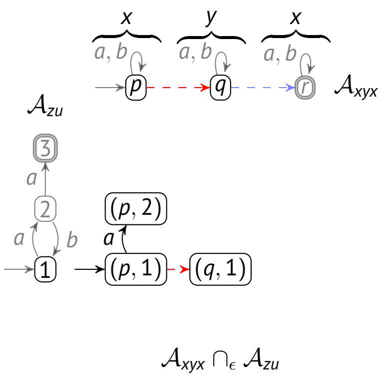


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

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

Intersection with epsilon transitions

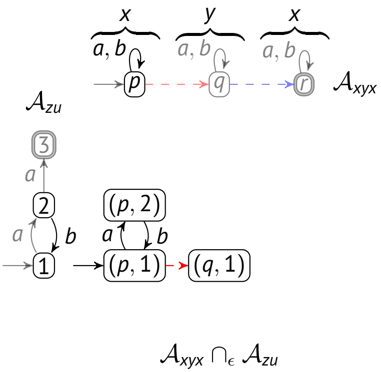
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

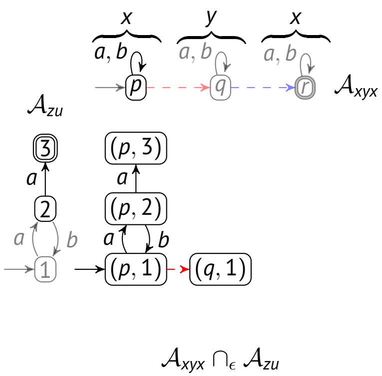
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

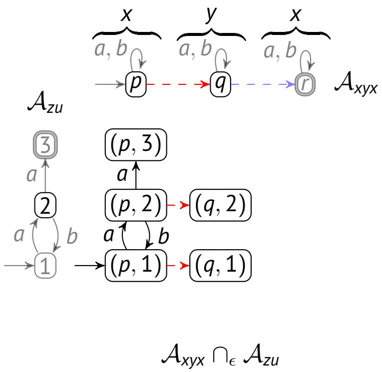
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

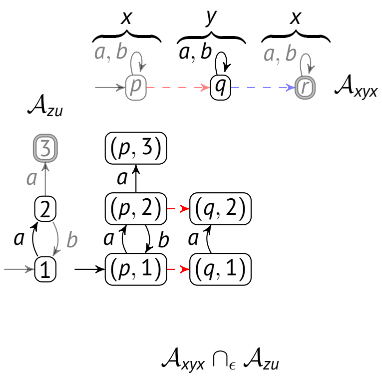
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

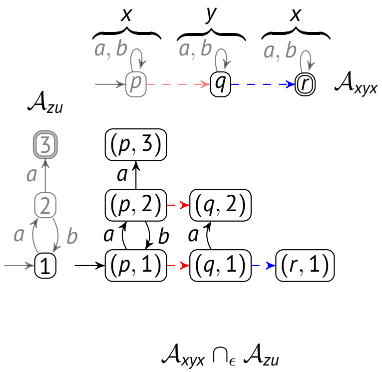
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

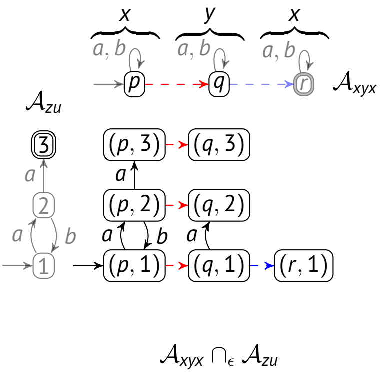
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

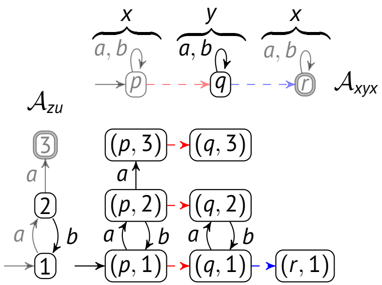
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

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

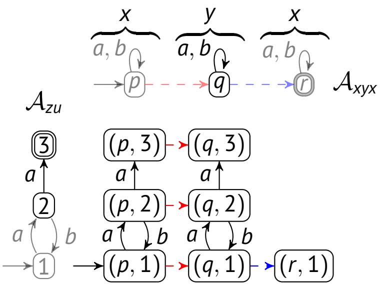


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

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

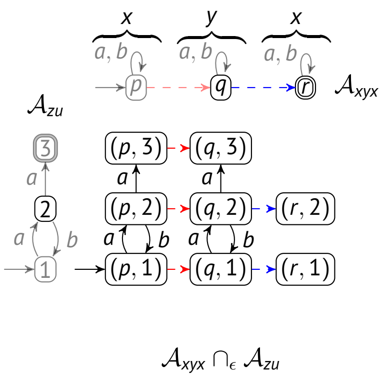


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

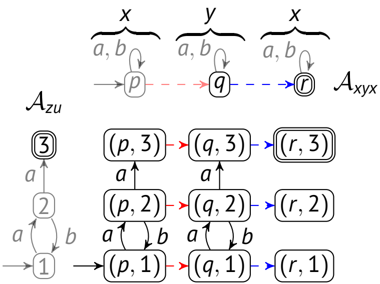
$$xyx = zu \land ww = xa \land u \in (baba)^* a \land z \in a(ba)^* \land x \in \Sigma^* \land y \in \Sigma^* \land w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

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

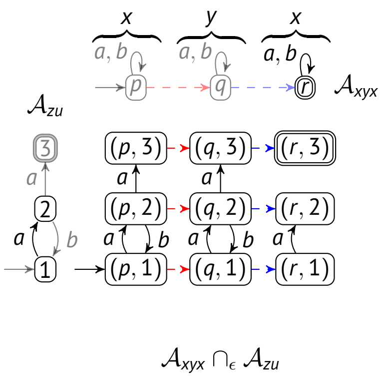


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

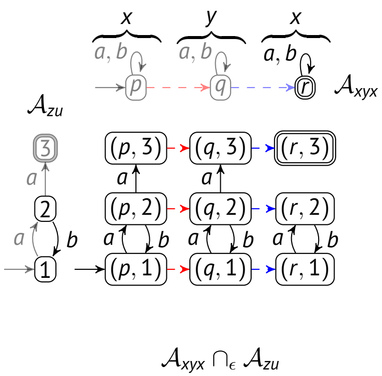
$$xyx = zu \land ww = xa \land u \in (baba)^*a \land z \in a(ba)^* \land x \in \Sigma^* \land y \in \Sigma^* \land w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

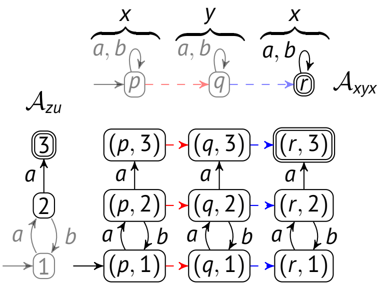
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

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

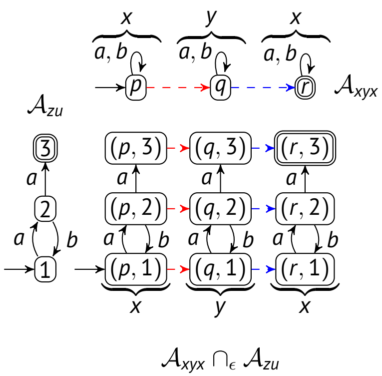


$\mathcal{A}_{xyx} \cap_\epsilon \mathcal{A}_{zu}$

- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

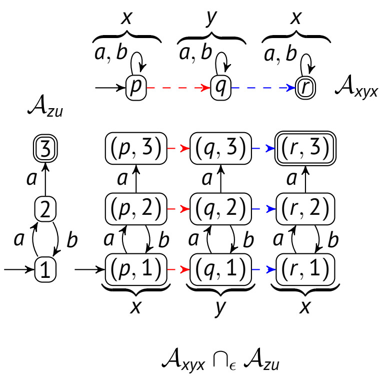
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Construct automata for both sides
  - $\mathcal{A}_{zu}$ – concatenation of right side
  - $\mathcal{A}_{xyx}$ – 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

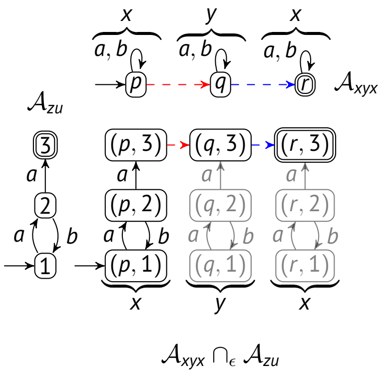
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^* a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Split product into noodles
  - values of y depends on values of x

Noodlification and unification

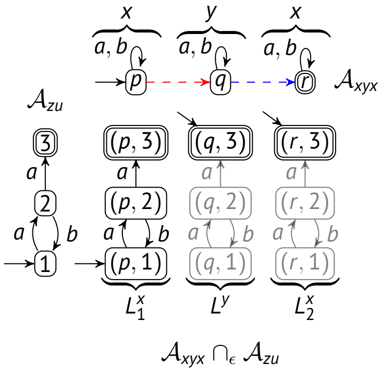
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Split product into noodles
  - values of y depends on values of x

Noodlification and unification

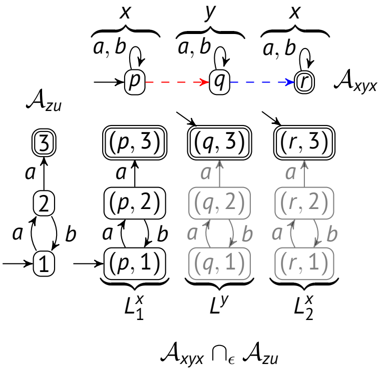
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Split product into noodles
  - values of y depends on values of x
- Noodle languages:
  - $L^x_1 = (ab)^*a$
  - $L^y = \epsilon$
  - $L^x_2 = \epsilon$

Noodlification and unification

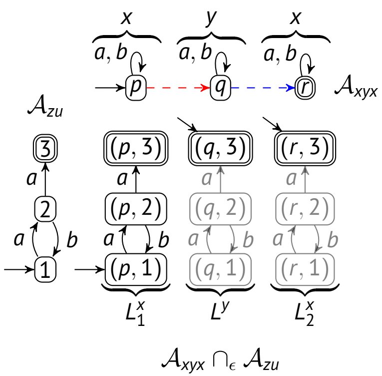
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^* a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



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

Noodlification and unification

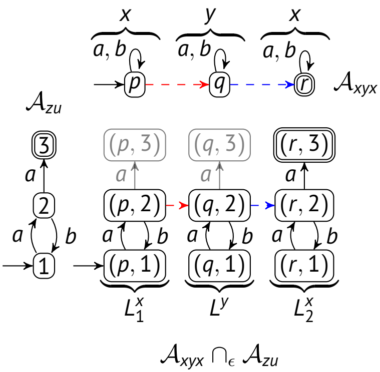
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^* a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



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

Noodlification and unification

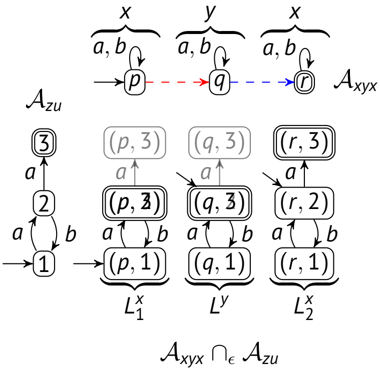
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^* a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



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

Noodlification and unification

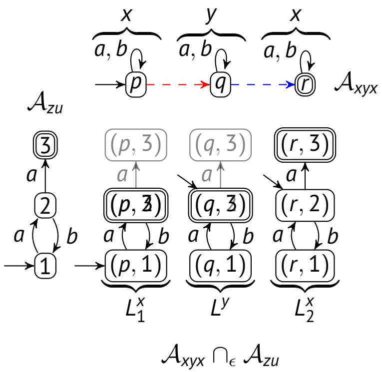
$$xyx = zu \wedge ww = xa \wedge u \in (baba)^*a \wedge z \in a(ba)^* \wedge x \in \Sigma^* \wedge y \in \Sigma^* \wedge w \in \Sigma^*$$



- Split product into noodles
  - values of y depends on values of x
- Noodle languages:
  - $L^x_1 = a(ba)^*$
  - $L^y = (ab)^*$
  - $L^x_2 = (ba)^*a$
- Unification:
  - intersect langs for the same variable
  - $\text{Lang}(x) = L^x_1 \cap L^x_2 = a(ba)^* \cap (ba)^*a = a$
  - $\text{Lang}(y) = L^y = (ab)^*$

Noodlification and unification

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



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

Continuing

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

- Refine further with $ww = xa$:

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

Continuing

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

- Refine further with $ww = xa$:

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

Continuing

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

- Refine further with $ww = xa$:

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

- Languages in equations match:

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

Continuing

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

- Refine further with $ww = xa$:

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

- Languages in equations match:

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

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

Stability of equation system

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

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

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

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

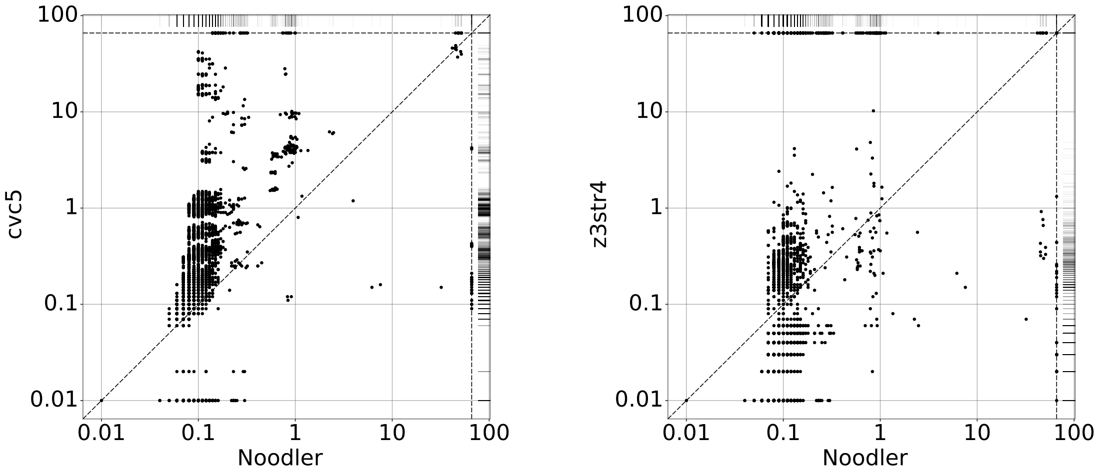
- Can be extended to multiple-equation system

Experimental evaluation

	PyEx-Hard (20,023)			Kaluza-Hard (897)			Str 2 (293)			Slog (1,896)		
	T/Os	time	time−T/O	T/Os	time	time−T/O	T/Os	time	time−T/O	T/Os	time	time−T/O
Noodler	**39**	**5,266**	2,926	**0**	**46**	46	3	**198**	18	**0**	165	165
Z3	2,802	178,078	9,958	207	15,360	2,940	149	8,955	15	2	332	212
CVC5	112	12,523	5,803	**0**	55	55	92	5,525	**5**	**0**	**14**	**14**
Z3str3RE	814	49,744	904	10	622	22	149	8,972	32	55	4,247	947
Z3str4	461	28,114	**454**	17	1,039	**19**	154	9,267	27	208	16,508	4,028
Z3-Trau	108	33,551	27,071	**0**	201	201	10	724	124	5	970	670
OSTRICH	2,979	214,846	36,106	111	14,912	8,252	238	14,497	217	2	13,601	13,481
Sloth	463	371,373	343,593	**0**	3,195	3,195		N/A		202	24,940	12,820
Retro	3,004	199,107	18,867	148	16,404	7,524	**1**	299	239		N/A	

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

Comparison with CVC5 and Z3str4 on PyEx-Hard



(a) Noodler vs. CVC5.

(b) Noodler vs. Z3str4.

Discussion

- Can beat well established solvers
  - can solve more benchmarks
  - average time is low
- Often complementary to other solvers
- Preprocessing is important

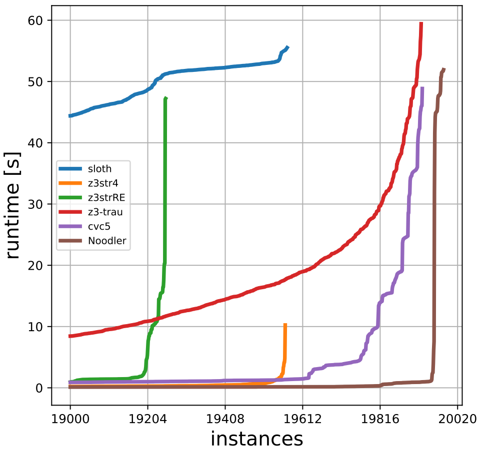


Figure: Hardest 1,023 formulae of PyEx-Hard

Future work

- Current status:

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

- Currently working on:
  - improved decision procedure handling other constraints
  - fast C++ implementation within Z3