Intelligent Software Development and Application Modernization Workshop
Saint-Petersburg, 27-29.06.2024

Introduction to Word Equations in the Context of Program Analysis

(A short overview)

Andrei Nemytykh

Antonina Nepeivoda

Program Systems Institute of RAS

nemytykh@math.botik.ru

a_nevod@mail.ru

This Talk

We are interested here in:

- Automated reasonings about words over some finite alphabet.
  - They are commonly used to perform analysis of string manipulating programs.
- A fundamental problem which such program analysis need to address is:
  - solving word equations, usually in combination with additional constraints.

Everywhere in this talk the “word” is used as a synonym of the term “finite string”.

This Talk

- A short overview on word equations: several
  - basic notions;
  - examples.
- The expressivity of the word equation language.
- Word-equational constraints in program analysis.

The First Example

Given two alphabets:

- Σ is a set of constant (e.g. lowercase latin) characters;
- $\mathcal{V}$ is a set of variables (e.g. capitalized latin characters).

Consider two constant words:

$$\mathit{aababaa} \qquad \mathit{aababaa}$$

The First Example

Given two alphabets:

- Σ is a set of constant (e.g. lowercase latin) characters;
- $\mathcal{V}$ is a set of variables (e.g. capitalized latin characters).

Replace some of occurrences of their subwords with variables:

$$aababaaba \qquad aabababaa$$

$$a^2 \mapsto X, \quad a^2ba \mapsto Y, \quad b \mapsto Z$$

$$XZabY \qquad YbaZX$$

The First Example

Given two alphabets:

- Σ is a set of constant (e.g. lowercase latin) characters;
- $\mathcal{V}$ is a set of variables (e.g. capitalized latin characters).

Insert the equality sign between the resulted words:

$$aababaaba \qquad aabababaa$$
$$a^2 \mapsto X, \quad a^2ba \mapsto Y, \quad b \mapsto Z$$
$$XZabY \qquad YbaZX$$
$$XZabY \quad = \quad YbaZX$$

The First Example

Given two alphabets:

- Σ is a set of constant (e.g. lowercase latin) characters;
- $\mathcal{V}$ is a set of variables (e.g. capitalized latin characters).

$$aababaaba \qquad aababababaa$$

$$a^2 \mapsto X, \quad a^2ba \mapsto Y, \quad b \mapsto Z$$

$$XZabY \quad = \quad YbaZX$$

We have obtained a word equation.

A solution of the equation is the following substitution:

$$X := a^2, \quad Y := a^2ba, \quad Z := b$$

Given two alphabets:

- Σ is a set of constant (e.g. lowercase latin) characters;
- $\mathcal{V}$ is a set of variables (e.g. capitalized latin characters).

$$aababaaba \qquad aabababaa$$

$$a^2 \mapsto X, \quad a^2ba \mapsto Y, \quad b \mapsto Z$$

$$XZabY \quad = \quad YbaZX$$

We have obtained a word equation.

A solution of the equation is the following substitution:

$$X := a^2, \quad Y := a^2ba, \quad Z := b$$

Another solution of the equation :

$X := \varepsilon$, $Y := a$, $Z := \varepsilon$, where ε is the empty word.

Word Equations

Given a finite alphabet Σ of constant characters and an alphabet $\mathcal{V}$ of variables. Let ε stand for the empty word.

Definition. A word equation is an equation of the form $\Psi = \Phi$, where $\Psi, \Phi \in \{\Sigma \cup \mathcal{V}\}^*$.

Definition. Given a word equation $\Psi = \Phi$, where $\Psi, \Phi \in \{\Sigma \cup \mathcal{V}\}^*$. A solution of the equation is a morphism $\sigma : \{\Sigma \cup \mathcal{V}\}^* \to \Sigma^*$ s.t. $\sigma(\Psi) = \sigma(\Phi)$, where the morphism σ respects the concatenation operation, and $\forall \xi \in \Sigma.\ \sigma(\xi) = \xi$.

Given a finite alphabet Σ of constant characters and an alphabet $\mathcal{V}$ of variables. Let ε stand for the empty word.

Definition. Given a word equation $\Psi = \Phi$, where $\Psi, \Phi \in \{\Sigma \cup \mathcal{V}\}^*$. A solution of the equation is a morphism $\sigma : \{\Sigma \cup \mathcal{V}\}^* \to \Sigma^*$ s.t. $\sigma(\Psi) = \sigma(\Phi)$, where the morphism σ respects the concatenation operation, and $\forall \xi \in \Sigma.\ \sigma(\xi) = \xi$.

- The first problem is to decide whether or not a given word equation has a solution.
- The second problem is to find all solutions of a given word equation.

Solution Sets

Given a word equation $\Psi = \Phi$, where $\Psi, \Phi \in \{\Sigma \cup \mathcal{V}\}^*$. Its solution-set may be empty, finite or infinite.

The set of all solutions to the equation

- $\mathsf{YabcbcdY} = \mathsf{XX}$ is empty;
- $\mathsf{XacXbc} = \mathsf{daYdbY}$ includes the only solution: $(\mathsf{X} := \mathsf{d}, \mathsf{Y} := \mathsf{c})$;
- $\mathsf{aX} = \mathsf{Xa}$ coincides with the infinite set a^*.

Word Equations in Program Analysis (Simplest Examples)

How do word equations naturally arise in the context of program analysis?

The variables X, X_1, Y, U, Z range over strings. Their values may be completely unknown or partially known in compile (analyse) time.

$$X = UbaYabZ$$

```
if ( includes(X, 'ba' + Y + 'ab') )
  then { ...... }
  else { ...... }
```

$$X = aX_1 = X_1a$$

```
while ( startsWith(X,'a') && endsWith(X,'a') ) {
  X := substring(X, 1, length(X))
  if is_empty(X) return true;
}
```

The Decision Problem

The first problem is to decide for *any* given word equation whether or not the given word equation has a solution.

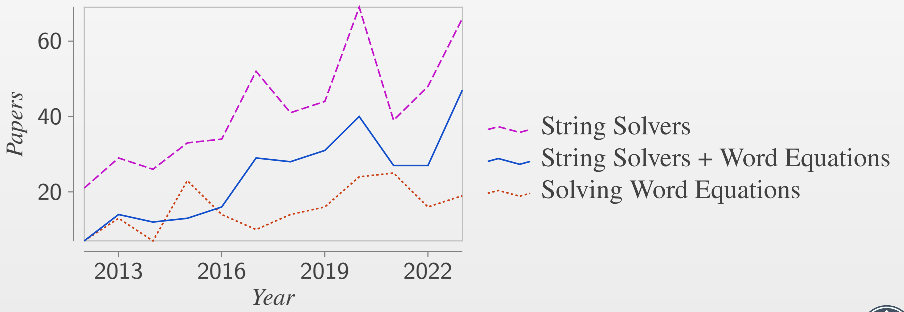
- The task to prove that this problem is undecidable was posed by A. A. Markov in the **1940**s.
- In the **1960**s Yu. I. Khmelevskii provided algorithms testing satisfiability of:
  1. any word equation with at most three variables;
  2. any word equation system consisting only of equations, each of which is at most two-variables equation.

The Decision Problem

- In **1977** G. S. Makanin provided an algorithm testing satisfiability for the whole set of word equations.
- In **1999** W. Plandowski showed, by his completely new algorithm, that the problem is actually in PSPACE.
- In **2013** A. Jez published a comparatively simple but nondeterministic algorithm for testing satisfiability of the word equations that works in time $\mathcal{O}(\log_2(N) \times poly(n))$ and in $\mathcal{O}(n \times \log_2 n)$ (bit) space, where n is the size of the input equation and N is the length of its length-minimal solution.

In recent years:

- Many studies are conducted on developing automated reasoning tools capable of proving or disproving statements involving words.
  - Their task is to automatically determine the satisfiability of that formula.



- Most currently available string-solvers focus rather on specific subsets to which their approaches are best suited.

- Many string solving tools are now available:
  - CVC5, Z3Str4, Norn, Z3-Trau, OSTRICH, Sloth, Woorpje, CertiStr, HAMPI.

Solution Sets

The second problem is to find all solutions of a given word equation.

Given a word equation, its solution-set may be infinite. The following questions, in general, are nontrivial.

- What does “to find all solutions of the equation” mean?
- How can the set be constructively described?
- Can the solution-set be described in mathematical terms more constructively as compared to the description provided by the original equation — in other words, by the formal language of word equations?

The second problem is to find all solutions of a given word equation.

- In the **1971** Yu. I. Khmelevskii published algorithms enumerating all solutions of:
  1. any word equation with at most three variables;
  2. any word equation system consisting only of equations, each of which is at most two-variables equation.

- In **1977** G. S. Makanin provided an algorithm enumerating all solutions of any word equation.

The second problem is to find all solutions of a given word equation.

- . . . .
- In **1999** W. Plandowski published completely new algorithm that is in PSPACE.
- In **2013** A. Jez published a comparatively simple but nondeterministic algorithm.

In **1977** G. S. Makanin provided an algorithm enumerating all solutions of any word equation.

- "The original Makanin's algorithm is one of the most complicated algorithms ever presented."
  - — J. Berstel, and J. Karhumaki. Combinatorics on words: a tutorial. Bulletin of the EATCS 79.178 (2003): 9.
- G. S. Makanin, The problem of solvability of equations in a free semigroup, Math. USSR-Sb., 32:2 (1977), pp: 129–198.

Simple algorithms for several classes of word equations.

- Word equations with constant right-hand sides:
  — $\Psi = C_0$, where $\Psi \in \{\Sigma \cup \mathcal{V}\}^*$, $C_0 \in \Sigma^*$.
- One-variable word equations.
- Quadratic word equations.

Example source: P.A. Abdulla, M.F. Atig, Y. Chen, B.P. Diep, L. Holik, A. Rezine, Ph. Rummer: Flatten and conquer: a framework for efficient analysis of string constraints, PLDI, 2017, ACM, pp. 602–617.

Initial program

```
main(X,Y,Z)   =
      eq(g(Z)++X, 'a'++g(Z)++Y);
g('i'++X)     =   'a'++g(X)++'b';
g('s'++X)     =   g(X)++'b';
g(ε)          =   ε;
eq(X, X)      =   true;
eq(X, Y)      =   false;
```

Simplified program

```
main'(X,Y,'i'++Z) = false;
main'(X,Y,'s'++Z) = false;
main'('a'++Y,Y,ε) = true;
main'(X,Y,ε)      = false;
```

- Quadratic equation $WX = aWY$ ($W =$ `g(Z)`) appears as a constraint generated by rule `eq(X, X) = true`, and its solution set includes $W \in a^*$, which is incompatible with any trace of g-computation except the trivial one.

One-Variable Equations

Examples. Let $\Sigma = \{a, b\}$.

- $Xab = abX$ has infinitely many solutions $X := (ab)^i$, where $\mathbb{N} \ni i \geq 0$.
- $XXbaaba = aabaXbX$ has exactly two solutions $X := a$ and $X := aaba$.
- $XaXbXaabbabaXbabaabbab =$ $abaabbabaXbabaabbXaXbX$ has exactly three solutions $X := \varepsilon, X := ab$, $X := abaabbab$.

One-Variable Equations

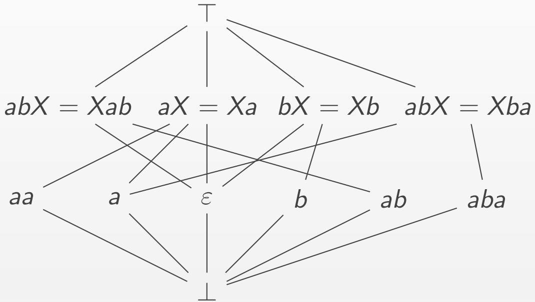
Theorem (Nowotka-Saarela, 2018). Every one-variable word equation has either infinitely many solutions or at most three.

- D. Nowotka, and A. Saarela. An optimal bound on the solution sets of one-variable word equations and its consequences. SIAM Journal on Computing 51.1 (2022): pp: 1-18.

Definition. A word $\eta \in \Sigma^+$ is said to be primitive, if for any $\xi \in \Sigma^*$, $n \in \mathbb{N}$ s.t. $\eta = \xi^n$ the equality $n = 1$ holds.

One-Variable-Equations Domain (Program Analysis*)

- One-variable equations with infinite solution sets and trivial equations of the form $X = \eta$, $\top$, $\bot$ form a complete lattice.



- A non-trivial one-variable word equation with an infinite solution set can be reduced to an equivalent one (where η_i stand for constant words)

$$\underbrace{\eta_1\eta_2 X = X\eta_2\eta_1 \ (\eta_1\eta_2 \text{ is primitive})}_{\text{normal-form condition}}.$$

- $\big((\eta_1\eta_2)^n X = X(\eta_2\eta_1)^n\big) \Leftrightarrow \big(\eta_1\eta_2 X = X\eta_2\eta_1\big)$ $(n \geq 1)$, hence the widening problem is resolved.

Constant-Free Equations

Definition. A word equation $\Psi = \Phi$ is said to be a constant-free word equation if $\Psi\Phi \in \mathcal{V}^*$.

Lemma (Khmelevskii, 1971). Given any solution (X, Y) of constant-free two-variables word equation of the form $X\Psi(X, Y) = Y\Phi(X, Y)$, where $\Psi(X, Y)\Phi(X, Y) \in \{X, Y\}^*$, there exist a primitive word $\eta \in \Sigma^*$ and $n, m \in \mathbb{N}$ s.t. $X = \eta^n$, $Y = \eta^m$.

- Yu. I. Khmelevskii, Equations in a free semigroup, Trudy Mat. Inst. Steklov., 107, 1971, 3–288; Proc. Steklov Inst. Math., 107 (1971), pp: 1–270.

Quadratic Equations

Given a word w and a character b, let $|w|_b$ stand for the number of occurences of b in w.

Definition. A word equation $\Psi = \Phi$ is said to be a quadratic word equation if for any $X \in \mathcal{V}$ $|\Psi\Phi|_X \leq 2$ holds.

The paper

- P.A. Abdulla, M.F. Atig, Y. Chen, L. Holik, A. Rezine, P. Rummer, J. Stenman: String Constraints for Verification. In: CAV 2014. LNCS, Vol. 8559, pp. 150–166 (2014).

presents a decision procedure for a logic that combines linear word (dis)equalities over string variables, together with constraints on the length of words, and on the regular languages to which words belong.

The set of the linear word equations is a subset of the quadratic word equations of the form $\Psi = \Phi$, where for any $X \in \mathcal{V}$ $|\Psi\Phi|_X \leq 1$.

Parametric Words

Definition. Let $\mathcal{A}$ be an alphabet, $\mathcal{N}$ be an alphabet of parameters ranging over $\mathbb{N}$. The set $\mathcal{P}$ of parametric words is defined inductively as follows: (1) $\mathcal{A}^* \subset \mathcal{P}$; (2) if $\phi \in \mathcal{P}$, $n \in \mathcal{N}$, then $\phi^n \in \mathcal{P}$; (3) if $\phi_1, \phi_2 \in \mathcal{P}$, then $\phi_1\phi_2 \in \mathcal{P}$.

We are interested in parametric words over the alphabet $\mathcal{A} = \Sigma \cup \Xi$. The terms $\xi \in \Xi$ are said to be word parameters.

Example. $\big((ab)^n b\big)^m a^n$, where n, m range over $\mathbb{N}$ while $a, b \in \Sigma$, is a parametric word.

We are interested in parametric words over the alphabet $\Sigma \cup \mathcal{V}$. The terms $\xi \in \mathcal{V}$ are said to be word parameters.

Example 1. The solution set of the conjugation equation $XZ = UX$ is parameterized: $X = (\xi\eta)^n\xi, Z = \eta\xi, U = \xi\eta$, where $n \geq 0$ ranges over $\mathbb{N}$ and the word parameters ξ, η range over Σ^*.

The conjugation equation $XZ = UX$ is both constant-free and quadratic.

Example 2. The solution set of the word equation system $\begin{cases} XY = YX \\ Y = ZZ \end{cases}$ is parameterized: $X = \xi^n, Y = \xi^{2m}, Z = \xi^m$, where $n, m \geq 0$ range over $\mathbb{N}$ and ξ ranges over Σ^*.

One-Variable Equations

Theorem. One-variable word equations are parametrizable.

- If an X-variable word equation has infinitely many solutions, then its whole solution-set is described with the following parametric word $\exists \xi \in \Sigma^* \ \exists \zeta \in \Sigma^*$ s.t. $X = (\xi\zeta)^n \xi$, where the constant word $\xi\zeta$ is primitive and $n \geq 0$ ranges over $\mathbb{N}$.

Simple example — built-in string concatenation in JavaScript.

$$\boxed{\text{x + y}} = \begin{cases} \{X = \xi_1\xi_2\}, \text{ if } x \text{ is } \{X = \xi_1\}, y \text{ is } \{X = \xi_2\}; \\ \dots \\ \{\xi_5\xi_6 X = X\xi_6\xi_5\}, \text{ if } x = \{\xi_1\xi_2 X = X\xi_2\xi_1\} \;\&\; y = \{\xi_3\xi_4 X = X\xi_4\xi_3\} \\ \qquad\qquad \text{and } \xi_2\xi_1 = \xi_3\xi_4 \text{ and } \xi_5\xi_6 = \xi_1\xi_2 \text{ and } \exists n\big(\xi_1\xi_3 = (\xi_1\xi_2)^n\xi_5\big); \\ \top, \text{ otherwise.} \end{cases}$$

$$\forall m, n\,\exists k\Big(\underbrace{(\xi_1\xi_2)^m\xi_1}_{x \text{ values}}\underbrace{(\xi_3\xi_4)^n\xi_3}_{y \text{ values}} = \underbrace{(\xi_5\xi_6)^k\xi_5}_{\boxed{\text{x+y}} \text{ values}}\Big)$$

$$\Downarrow$$

$$(\xi_1\xi_2 = \xi_5\xi_6) \;\&\; \forall n\,\exists k\big(\xi_1\underbrace{(\xi_3\xi_4)^n\xi_3}_{y \text{ values}} = \underbrace{\xi_1(\xi_2\xi_1)^k\xi_2}_{(\xi_5\xi_6)^{k+1}}\xi_5\big)$$

Theorem (Khmelevskii, 1971). For every constant-free three-variables word equation its solution-set may be described with finitely many parametric words.

- Yu. I. Khmelevskii, Equations in a free semigroup, Trudy Mat. Inst. Steklov., 107, 1971, 3–288; Proc. Steklov Inst. Math., 107 (1971), pp: 1–270.

Parametric solutions are particularly useful because they are a very explicit description of the solution-set.

The word equation language $\mathcal{WL}$ vs. the regular expression language $\mathcal{RL}$.

Example. The solution set of the equation $aXXbY = XaYbX$ is parameterized: $X = a^n$, $Y = (a^n b)^m a^n$, where both $n \geq 0$ and $m \geq 0$ range over $\mathbb{N}$.
The set of the Y-values $(a^n b)^m a^n$ can not be expressed with a regular expression.

Thus the $\mathcal{WL}$ language used for describing abstract values is able to express involved recursive program invariants as compared to $\mathcal{RL}$.

Khmelevskii's Equation

The following equation is both constant-free and quadratic.
Khmelevskii's equation: $XUY = YWX$.

Theorem (Khmelevskii, 1971). The word equation $XUY = YWX$ is not parametrizable.

- Yu. I. Khmelevskii, Equations in a free semigroup, Trudy Mat. Inst. Steklov., 107, 1971, 3–288; Proc. Steklov Inst. Math., 107 (1971), pp: 1–270.

In 2005 E. Czeizler published an alternative simple proof of this Khmelevskii theorem.

- E. Czeizler, The non-parametrizability of the word equation xyz = zvx: a short proof. Theoret. Comput. Sci. 345(2–3), 296–303 (2005).

Theorem (Khmelevskii, 1971). The word equation $XUY = YWX$ is not parametrizable.

Even the following instance of Khmelevskii's equation is not parametrizable.
Theorem (Ilie-Plandowski, 2000). The word equation $XabY = YbaX$ is not parametrizable.

- L. Ilie, W. Plandowski. Two-variable word equations. RAIRO-Theoret. Inform. Appl. 34(6), 467–501 (2000).

How can solution-sets to word equations be represented if not by parametric words?

- One can try to reveal algorithms producing complete descriptions of the full solution-sets.
- Another Makanin's algorithm for solving equations in a free group was used by Razborov (**1993**) in developing an algorithmic representation of all solutions to systems of equations in the free group.
  — A. A. Razborov. On systems of equations in free groups. In: Combinatorial and Geometric Group Theory, pp. 269–283 (1993).

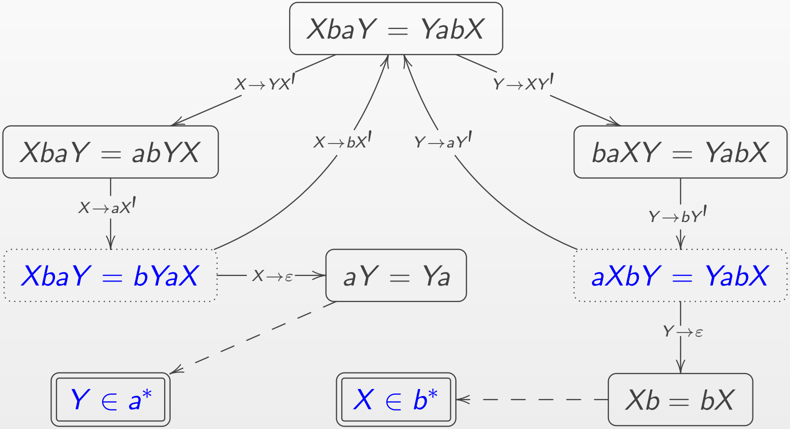
Graph Representations of Solution-Sets (Word Equations)

How can solution-sets to word equations be represented if not by parametric words?

- In **2006** Plandowski adapted his decidability algorithm for solving word equations to produce a finite graph representing all solutions.
  — W. Plandowski. An efficient algorithm for solving word equations. In: Proc. of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 467–476 (2006).
- In **2016** A. Jez published a comparatively simple but nondeterministic algorithm.
  — A. Jez. Recompression: a simple and powerful technique for word equations. J. ACM (JACM) 63(1), 1–51 (2016).

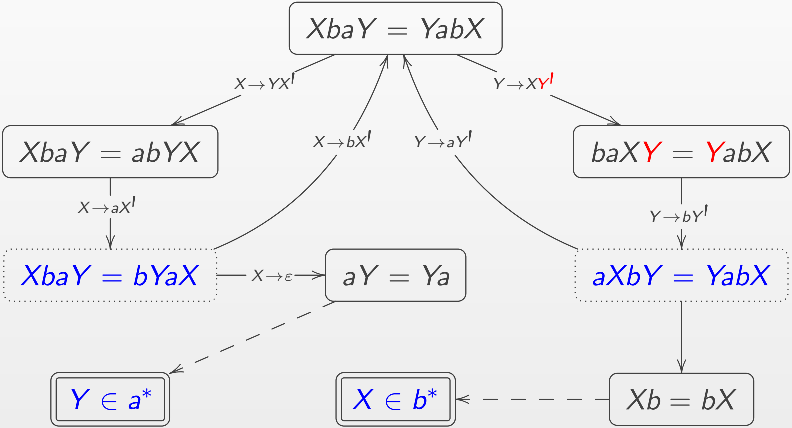
Word Equation $XbaY = YabX$

Graph representation of the solution-set of the non-parametrizable quadratic equation $XbaY = YabX$.

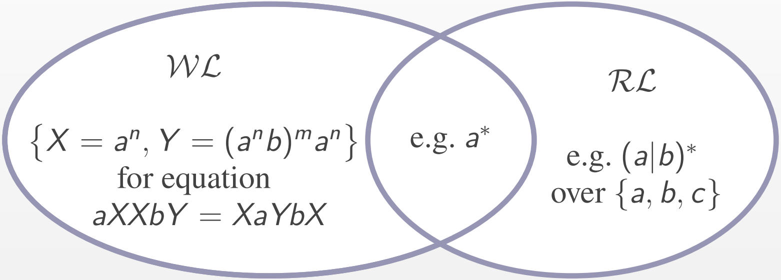


Word Equation $XabY = YbaX$

Here the prime sign labeling the fresh variables in the narrowings is dropped in the target vertices.



The word equation language $\mathcal{WL}$ vs. the regular expression language $\mathcal{RL}$: intersect but are not subsets each of other:



— There is no word equation, which solution-set is represented by $(a|b)^*$ over the alphabet $\{a, b, c\}$.

- J. Karhumaki, F. Mignosi, W. Plandowski. The expressibility of languages and relations by word equations. Journal of the ACM (JACM), 47(3), pp: 483-505, (2000).

Word Equations vs. Algebraic Equations

$\mathbb{R}[x_1, \ldots, x_n]$ stands for the set of real polynomials on $x_1, \ldots, x_n$ variables. Consider polynomials $P_i, Q_j \in \mathbb{R}[x_1, \ldots, x_n]$.

The set of real solutions to the algebraic-equation system:

$$\begin{cases} P_1 = Q_1 \\ P_2 = Q_2 \end{cases}$$

coincides with the set of real solutions to the equation

$$(P_1 - Q_1)^2 + (P_2 - Q_2)^2 = 0$$

Word Equations vs. Algebraic Equations

Lemma (A. B. Livchak, 1971). If $\#\Sigma \geq 2$, then for every four words $\Phi_1, \Phi_2, \Psi_1, \Psi_2 \in \{\Sigma \cup \mathcal{V}\}^*$ the solution-set to the word-equation system:

$$\begin{cases} \Phi_1 = \Psi_1 \\ \Phi_2 = \Psi_2 \end{cases}$$

coincides with the solution-set to the following equation, where $a, b \in \Sigma$:

$$\Phi_1 a \Phi_2 \Phi_1 b \Phi_2 = \Psi_1 a \Psi_2 \Psi_1 b \Psi_2$$

Lemma (A. B. Livchak, 1971). If $a, b \in \Sigma$, then $\begin{cases} \Phi_1 = \Psi_1 \\ \Phi_2 = \Psi_2 \end{cases} \iff$ $\Phi_1 a \Phi_2 \Phi_1 b \Phi_2 = \Psi_1 a \Psi_2 \Psi_1 b \Psi_2$

Proof. (1) The case $\Longrightarrow$ is trivial.
(2) Let us prove the case $\Longleftarrow$. The lengths of the equation sides are equal $|\Phi_1 a \Phi_2 \Phi_1 b \Phi_2| = |\Psi_1 a \Psi_2 \Psi_1 b \Psi_2|$ and even. Hence, the first halves of the sides equal each other.
We have: $\Phi_1 a \Phi_2 = \Psi_1 a \Psi_2$ **(i)** and $\Phi_1 b \Phi_2 = \Psi_1 b \Psi_2$ **(ii)**.

(2.A) The case $|\Phi_1| = |\Psi_1|$ is trivial.
(2.B) Assume that $|\Phi_1| > |\Psi_1|$. Then for any solution σ of **(i)** the $(|\Psi_1| + 1)$-st character of the word $\sigma(\Phi_1)$ is a, while **(ii)** requires that the same character should be b. Thus, the case (2.B) is impossible. $\square$

Word Equations vs. Algebraic Equations

For any polynomials $P_i, Q_j \in \mathbb{R}[x_1, \ldots, x_n]$.

$$(P_1 = Q_1) \vee (P_2 = Q_2)$$

$$\Updownarrow$$

$$(P_1 - Q_1) \times (P_2 - Q_2) = 0$$

Word Equations vs. Algebraic Equations

Theorem. If $\#\Sigma \geq 2$, then for any words $\Phi_1, \Phi_2, \Psi_1, \Psi_2 \in \{\Sigma \cup \mathcal{V}\}^*$

$$(\Phi_1 = \Psi_1) \vee (\Phi_2 = \Psi_2)$$
$$\Updownarrow$$
$$\exists X \in \mathcal{V}\ \exists Y \in \mathcal{V} : X\Upsilon^2\Psi\Upsilon^2 Y = \Upsilon^2\Phi_1\Upsilon^2\Phi_2\Upsilon^2$$

Where $\Psi = \Psi_1\Psi_2, \Delta_1 = \Phi_1\Psi_2, \Delta_2 = \Psi_1\Phi_2$ and
$\Upsilon = \Delta_1\Delta_2\Psi a\Delta_1\Delta_2\Psi b$, and a, $b \in \Sigma$, $a \neq b$.

Word Equations in Abstract Interpretation (Program Analysis)

Simple Example: abstract interpretation over $\mathcal{WL}$-lattice shows the sanitization given in line 5 is sound (line 7 is unreachable).

```
1   z = ξ;
2   x = '<script' + z;
3   if (x != z + z) {
4       x = x + x };
5   replaceAll(x, 'ip', 'ipv4');
6   if (contains(x, 'script')) {
7       return 'Possible code injection'; }
```

Word Equations in Abstract Interpretation (Program Analysis)

The sanitization given in line 5 is sound.

1	`z = ξ;`	$z \mapsto \{X = \xi\}$
2	`x = '<script' + z;`	$x \mapsto \{X = \eta\}$
3	`if (x != z + z) {`	(does not change)
4	`x = x + x };`	$\{X = \eta\} \vee \{X = \eta + \eta\}$
		$= \{\mathit{prm}(\eta)X = X\,\mathit{prm}(\eta)\}$; $x \mapsto \{\mathit{prm}(\eta)X = X\,\mathit{prm}(\eta)\}$
5	`replaceAll(x,'ip','ipv4');`	$x \mapsto^{*} \{\eta' X = X \eta'\}$
6	`if (contains(x,'script')) {`	(never holds for the abstract value of x)
7	`return 'Possible code injection'; }`	(is pruned)

- $\mathit{prm}(\eta)$ is a primitive* root of η;
- $\eta = \texttt{'<script'} + \xi$;

  $\eta' = \texttt{replaceAll}(\mathit{prm}(\eta), \texttt{'ip'}, \texttt{'ipv4'})$.
- predicate `contains(x,'script')` fails for any value of X satisfying $\eta' X = X \eta'$.

Thank for Attention

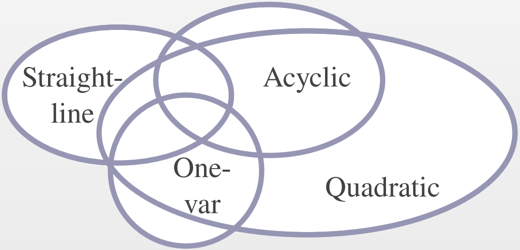
- Any questions?

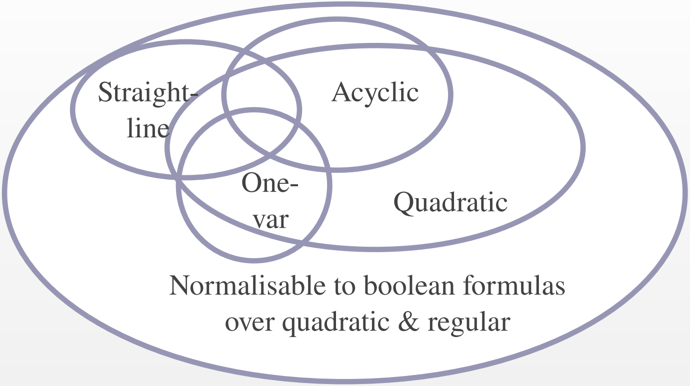
Possible Extensions

$\mathcal{WL}$ — existential string theory
(concatenation + equality = word equations).

Theory	Replace All (Const Args)	letter count	length count	$\mathcal{RL}$	Complexity
$\mathcal{WL} + \mathcal{RL}$	✗	✗	✗	✓	PSPACE
$\mathcal{WL}$+len	✗	✗	✓	✗	???
$\mathcal{WL}$+count	✗	✓	✗	✗	Undec.
$\mathcal{WL}$+repl	✓	✗	✗	✗	Undec.

- Many string solving tools are now available:
  - HAMPI — bounded
  - Norn — acyclic
  - OSTRICH — straight-line
  - Sloth — straight-line & acyclic
  - Woorpje — bounded
  - CertiStr — acyclic
  - CVC5, Z3Str3 — acyclic & bounded
  - Z3-Trau — chain-free





Straight-line & non-acyclic

$$\exists X, Z, n\Big((Z = XXX) \,\&\, (Z = (\mathtt{ab})^n\mathtt{a})\Big)$$

One-var & non-straight-line

$$\exists X\Big(\mathtt{ab}XX = XX\mathtt{ba}\Big)$$

For now, non-refutable by CVC5 & Z3.

Levi's lemma:

- given equation $X\Phi = a\Psi$, every element of X-solution set either
  - equals ε;
  - or starts by a.
- given equation $X\Phi = Y\Psi$, any pair (η_1, η_2) of (X, Y)-solutions either:
  - satisfies $\eta_1 = \eta_2$;
  - satisfies $\exists\xi(\eta_1 = \eta_2\xi)$;
  - satisfies $\exists\xi(\eta_2 = \eta_1\xi)$.

- Useful in constructing solution graphs of simple equations.
- Applied in a straightforward manner, causes infinite growth of non-quadratic equations.

Parametric solutions are particularly useful because they are a very explicit description of the solution-set.

$$\mathtt{a}\underbrace{(\boxed{\mathtt{a}^*\mathtt{a}})}_{\text{capture group 1}}\overbrace{\backslash 1}^{\text{reference}} \quad \text{vs} \quad \underbrace{(\boxed{\mathtt{a}\,\mathtt{a}^*})}_{\text{capture group 1}}\mathtt{a}\overbrace{\backslash 1}^{\text{reference}}$$

- Iterations over constants $\longrightarrow$ parametric words.
- Alternation blocks $\longrightarrow$ word parameters.
- Uniformity of the resulted equation verifies bisimulation:

$$\forall \mathtt{n} \in \mathbb{N}(\boxed{\mathtt{a}^{\mathtt{n}}\mathtt{a}} = \boxed{\mathtt{a}\mathtt{a}^{\mathtt{n}}})$$

No REDoS situation (unambiguous parsing):

- $\overbrace{(\mathtt{a}^*\mathtt{b})}^{\text{capture group 1}}\underbrace{(\mathtt{a}\backslash 1)^*(\backslash 1)^*}_{\text{unambiguous}}$

REDoS situation (non-constant parse paths number):

- $\overbrace{(\mathtt{a}^*)}^{\text{capture group 1}}\ \mathtt{b}\underbrace{(\mathtt{a}\backslash 1)^*(\backslash 1)^*}_{\text{ambiguity}}$

Ambiguity occurs only if the quadratic equation $WX = aWY$, where W is the capture group 1 value, has at least one W-solution satisfying the language of the capture group 1.

Closure Properties

$\mathcal{WL}$ are closed under:

- finite intersections;
- finite unions;
- string reversal and shift.

$\mathcal{WL}$ are not closed under:

- morphic & inverse morphic images;
- complementation;
- intersection with regular languages.

By Tarski–Knaster theorem, lattice fixpoints *wrt* endomorphisms form sublattices in the $\mathcal{WL}$-based lattice, being able to express additional string invariants.

References-I

1. P.A. Abdulla, et al.: String Constraints for Verification. In: CAV 2014. LNCS, Vol. 8559, pp. 150–166 (2014).
2. P.A. Abdulla, et al.: Norn: an SMT solver for string constraints. In: CAV 2015. LNCS, vol. 9206, pp. 462–469 (2015).
3. P.A. Abdulla, et al.: Flatten and conquer: a framework for efficient analysis of string constraints, PLDI, 2017, ACM, pp. 602–617.
4. H. Barbosa, et al.: CVC5: a versatile and industrial-strength SMT solver. In: TACAS 2022. LNCS, vol. 13243, pp. 415–442, (2022).
5. J. Berstel, and J. Karhumaki. Combinatorics on words: a tutorial. Bulletin of the EATCS 79.178 (2003): 9.
6. E. Czeizler, The non-parametrizability of the word equation xyz = zvx: a short proof. Theoret. Comput. Sci. 345(2–3), 296–303 (2005).

. . .

References-II

7. T. Chen, M. Hague, A.W. Lin, P. Rummer, Z. Wu: Decision procedures for path feasibility of string-manipulating programs with complex operations. Proc. ACM Program. Lang. 3(POPL), 1–30 (2019).
8. J.D. Day, et al.: On solving word equations using SAT. In: RP 2019. LNCS, vol. 11674, pp. 93–106 (2019).
9. J.D. Day. Word Equations in the Context of String Solving. In Int. Conf. on Developments in Language Theory. 2022, pp. 13-32.
10. L. Holik, P. Janku, A.W. Lin, P. Rummer, T. Vojnar: String constraints with concatenation and transducers solved efficiently. In: Proc. of the ACM on Programming Languages, vol. 2, pp. 1–32 (2018).
11. L. Ilie, W. Plandowski. Two-variable word equations. RAIRO-Theoret. Inform. Appl. 34(6), 467–501 (2000).

. . .

References-III

12. A. Jez. Recompression: a simple and powerful technique for word equations. J. ACM (JACM) 63(1), 1–51 (2016).
13. S. Kan, A.W. Lin, P. Rummer, M. Schrader: CertiStr: a certified string solver. In: Proc. of the 11th ACM SIGPLAN International Conf. on Certified Programs and Proofs, pp. 210–224 (2022).
14. J. Karhumaki, F. Mignosi, and W. Plandowski, The expressibility of languages and relations by word equations, J. ACM, vol. 47, no. 3, pp. 483–505, May 2000.
15. A. Kiezun, V. Ganesh, S. Artzi, P.J. Guo, P. Hooimeijer, M.D. Ernst: HAMPI: a solver for word equations over strings, regular expressions, and context-free grammars. ACM Trans. Softw. Eng. Methodol. (TOSEM) 21(4), 1–28 (2013).

. . .

References-IV

16. Yu. I. Khmelevskii, Equations in a free semigroup, Trudy Mat. Inst. Steklov., 107, 1971, 3–288; Proc. Steklov Inst. Math., 107 (1971), pp: 1–270.
17. N.K. Kosovskij, Some properties of solutions to equations in a semigroup (in Russian), Zap. Nauch. Sem. LOMI, vol. 32, pp. 21–28, 1972. Available at https://www.mathnet.ru/rus/znsl2560
18. G.S. Makanin, The problem of solvability of equations in a free semigroup, Math. USSR-Sb., 32:2 (1977), pp: 129–198.
19. F. Mora, M. Berzish, M. Kulczynski, D. Nowotka, V. Ganesh: Z3Str4: a multi-armed string solver. In: FM 2021. LNCS, vol. 13047, pp. 389–406, (2021).

References-V

20. D. Nowotka, and A. Saarela. An optimal bound on the solution sets of one-variable word equations and its consequences. SIAM Journal on Computing 51.1 (2022): pp: 1-18.
21. W. Plandowski. An efficient algorithm for solving word equations. In: Proc. of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 467–476 (2006).
22. W. Plandowski. An efficient algorithm for solving word equations. Theoretical Computer Science Volume 792, 5 November 2019, Pages 20-61.
23. A.A. Razborov. On systems of equations in free groups. In: Combinatorial and Geometric Group Theory, pp. 269–283 (1993).