Equations in free monoids, formal languages, and complexity

Laura Ciobanu (with V. Diekert and M. Elder)

Heriot-Watt University, Edinburgh

York Semigroup Seminar, October 17, 2018

HERIOT WATT UNIVERSITY

Equations in free monoids = Word equations

Let $A = \{a, b, c, \dots\}$ and $\Omega = \{X_1, X_2, X_3, \dots\}$ be two finite sets.

$M(A) = A^* =$ the free monoid over A.

- An *equation in $M(A)$* is an expression $U = V$ with $U, V \in (A \cup \Omega)^*$.
- A *solution* is an assignment $X_i \to u_i \in M(A)$ for each X_i that makes U equal V in $M(A)$.

Example - free monoid

- M - free monoid generated by a and b
- $\{X, Y, Z, U\}$ - variables.

A solution for the equation

$$XaUZaU = YZbXaabY$$

is $X \to abb$, $Y \to ab$, $Z \to ba$, $U \to bab$.

Then $XaUZaU = YZbXaabY = abbababbaabab$.

Equations in groups

- G - a group
- $\{X_1, \ldots, X_n\}$ - a set of variables.

An *equation* with coefficients g_j in G has the form

$$g_1 X_{i_1}^{\epsilon_1} g_2 X_{i_2}^{\epsilon_2} \ldots X_{i_m}^{\epsilon_m} g_{m+1} = 1_G$$

where $i_j \in \{1, \ldots, n\}, \epsilon_j \in \{1, -1\}$.

Examples

F - free group on a and b, X variable.

- The equation

$$X^2 = a$$

has no solutions.

- The equation

$$X^{-1}abX = ba$$

has solutions $X = (ab)^n a$, $n \in \mathbb{Z}$.

Example - free group

$F = F(a, b)$ is free group on a and b, X and Y variables.

The equation

$$XYX^{-1}Y^{-1} = aba^{-1}b^{-1}$$

has solutions

$$X = a,\ Y = b;$$

$$X = ab,\ Y = b;$$

$$\vdots$$

$$X = ab^n,\ Y = b$$

$$X = a,\ Y = ba^m$$

Questions

- **Diophantine Problem** - **a decision problem**

  Does an equation have solutions?

Questions

- **Diophantine Problem** - **a decision problem**

  Is a given equation satisfiable?

Questions

- **Diophantine Problem** - **a decision problem**

  Does there exist an algorithm which for any equation in a given semigroup or group can determine whether the equation has solutions?

- **Search Diophantine Problem**

  Give an algorithm that finds a solution (all solutions) for any satisfiable equation.

Motivation: Hilbert's Tenth Problem

(Markov, Hmelevskii, Malcev, Makanin, ...)

- The matrices $a = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $b = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ form a free monoid inside $SL_2(\mathbb{Z})$.

- Consider a word equation over $\{a, b\}^*$ with variables $\{X_1, \ldots, X_n\}$, where each $X_i = \begin{pmatrix} \alpha_{i1} & \alpha_{i2} \\ \alpha_{i3} & \alpha_{i4} \end{pmatrix}$, $\alpha_{ij} \in \mathbb{N}$.

- Consider an equation over $\{a, b\}^*$ with variables $\{X_1, \ldots, X_n\}$, where each $X_i = \begin{pmatrix} \alpha_{i1} & \alpha_{i2} \\ \alpha_{i3} & \alpha_{i4} \end{pmatrix}$, $\alpha_{ij} \in \mathbb{Z}$.
- The equation becomes

$$\begin{pmatrix} P_1 & P_2 \\ P_3 & P_4 \end{pmatrix} = \begin{pmatrix} Q_1 & Q_2 \\ Q_3 & Q_4 \end{pmatrix},$$

  where $P_1, \ldots, Q_4$ are polynomials in the α_{ij}.

- The equation has a solution if and only if the Diophantine system has a solution:

$$\alpha_{i1}\alpha_{i4} - \alpha_{i2}\alpha_{i3} = 1$$

$$P_j = Q_j.$$

The Diophantine Problem is:

- Unsolvable in general monoids;

  e.g. unsolvable in free inverse monoids (Rozenblat 1986)

- Solvable, but hard for free (semi)groups: not primitive recursive, EXPSPACE, G. Makanin (1982) and A. Razborov (1985)

- Solvable (theoretically) in the following classes of groups:

  - hyperbolic (Rips - Sela, Dahmani - Guirardel, 1995 - 2010)

  - relatively hyperbolic (Dahmani, Groves, 2009)

  - right-angled Artin (Diekert - Muscholl, 2005)

Connections to:

- Logic (Tarski's Conjecture, 1950's)

  The elementary theories of free groups with different number of generators coincide. (Sela, Kharlampovich & Myasnikov 2000)

- Geometry

  Quadratic equations (every variable appears twice) - well understood.

- Theoretical Computer Science

  Unification theory - solvability of free monoid equations.

Equations and formal languages

Description of solutions

1. Algebraic/algorithmic approach

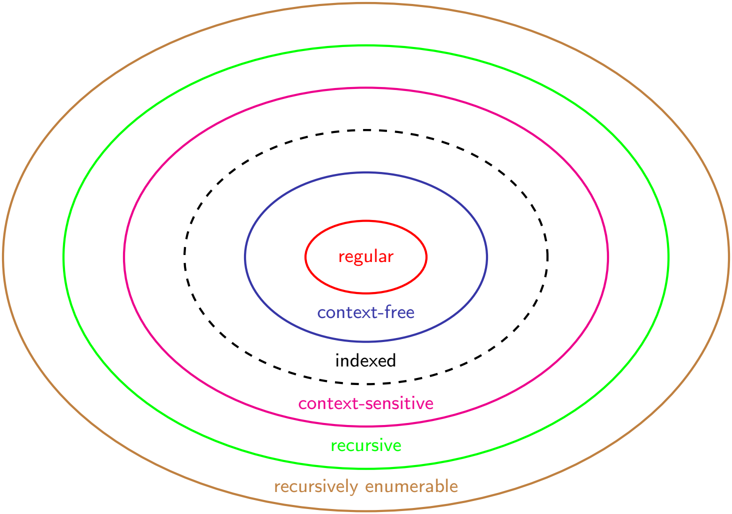
   1987 Razborov: Description of all solutions for an equation in a free group via "Makanin-Razborov" diagrams.

   2017 Sela: Description of all solutions for an equation in a free monoid via "Makanin-Razborov" diagrams.

2. Formal language approach

   - What is the formal language type of a solution set in a free (semi)group?
   - ANSWER: An indexed language!
     (C., Diekert, Elder)

Formal languages and the Chomsky hierarchy



How do we represent the solutions as a language?

Example: The equation $XYX^{-1}Y^{-1} = aba^{-1}b^{-1}$ has solutions

$$\{a\#b, ab\#b, ab^2\#b, \ldots\}.$$

- Let $U = V$ be an equation over $F(A)$ with variables $\Omega = \{X_1, \ldots, X_k\}$.
- Any solution of $U = V$ is given by a homomorphism

$$\sigma : F(\Omega \cup A) \to F(A)$$

  that fixes A, such that $\sigma(U) = \sigma(V)$.
- Let $\#$ be a symbol not in A. We encode a solution σ of $U = V$ as

$$\sigma(X_1)\#\cdots\#\sigma(X_k).$$

Theorem (C., Diekert, Elder)

Let $F(A)$ be the free group on A and $\Omega = \{X_1, \dots, X_k\}$ a set of variables.

A solution of $U = V$ will be denoted by $\sigma : F(\Omega \cup A) \to F(A)$.

- The set of all solutions in reduced words

$$Sol(U = V) = \{\sigma(X_1)\# \cdots \#\sigma(X_k) \mid \sigma \text{ solution}\}$$

  is an indexed language.

- We produce an algorithm which takes (U, V) as input and computes in NSPACE($n \log n$) a finite graph (an NFA) $\mathcal{A}$ where the transitions are monoid morphisms and

$$Sol(U = V) = \{\phi(\$) \mid \phi \in L(\mathcal{A})\}.$$

1999-2014: The CS approach to solving equations

- 1999 Plandowski: Decidability for word equations is in PSPACE.
- 2000 Gutiérrez: Decidability for free group equations is in PSPACE.
- 2001 Diekert, Gutiérrez, Hagenah: Decidability for free group equations with *rational constraints* is PSPACE-complete.
- 2013: Artur Jeż applied *recompression* and simplified all known proofs for decidability.
- 2014: Diekert, Jeż, Plandowski gave a new PSPACE algorithm that produces all solutions for an equation with rational constraints in free groups or free monoids with involution.

The more precise statement of our theorem

Let $U = V$ be an equation over $F(A)$ with variables $\Omega = \{X_1, \ldots, X_k\}$.

Then the set of all solutions in reduced words

$$Sol(U = V) = \{\sigma(X_1)\# \cdots \#\sigma(X_k) \mid \sigma \text{ solution}\}$$

is EDT0L (**E***xtended*, **D***eterministic*, **T***able*, **0** *interaction, and* **L***indenmayer*).

Lindenmeyer systems $\Longrightarrow$ L languages

- 1960s: Lindenmeyer came up with grammars whose purpose was to model the growth of organisms.
- Main feature: growth in parallel – leaves in a fern growth all in parallel, not sequentially.
- L languages: apply a family of morphisms to a fixed word*.

Remark: The class of EDT0L languages is a proper subclass of indexed languages, and is incomparable to the class of context-free languages.

Example: The language $\{uu \mid u \in A^*\}$ is EDT0L, but not context-free.

An example

Let $A = \{a, b\}$ and $C = \{a, b, \#\}$.

We let $H = \{f, g_a, g_b, h\}$ be a set of morphisms $C^* \to C^*$, where

- $f(\#) = \#\#$
- $g_a(\#) = a\#$
- $g_b(\#) = b\#$
- $h(\#) = 1$
- On all other letters f, g_a, g_b, h are the identity.

Let $R = h\{g_a, g_b\}^* f$.

Then $\{\phi(\#) \mid \phi \in R\} = \{uu \mid u \in A^*\}$ is EDT0L, but not context-free.

Theorem (C., Diekert, Elder)

Let $F(A)$ be the free group on A and $\Omega = \{X_1, \ldots, X_k\}$ a set of variables.

- The set of all solutions in reduced words to $U = V$ is an EDT0L language.
- There is an algorithm which takes (U, V) as input and computes in NSPACE($n \log n$) a finite graph (an NFA) $\mathcal{A}$ where the transitions are monoid morphisms and

$$Sol(U = V) = \{\phi(\$) \mid \phi \in L(\mathcal{A})\}.$$

The algorithm: an overview

1. First we translate an equation in a free group into a system of equations in a free monoid with involutions.

2. Then we solve equations in free monoids with RATIONAL constraints.

3. We ensure that the solutions are reduced words in the free group by using the rational constrains. (MR diagrams cannot produce reduced words!)

About the algorithm

1. It is easy to produce the graph that gives the solutions, HARD to show it is the correct graph.

2. Once the graph is produced from an initial vertex to final vertices, we start at the final vertices and go backwards to the initial ones to read off the solutions.

   This gives us the EDT0L description.

The proof: preprocessing

Step 1: transform equation into triangular system

Take the equation $U = V$ in $F(A)$, which is equivalent to $UV^{-1} = 1$,

and make a system of equations, using new variables Z_i, as follows:

$$\begin{aligned} UV^{-1} &= p_1 p_2 p_3 \ldots p_n = 1 \\ &\rightarrow p_1 p_2 = Z_1,\ Z_1 p_3 = Z_2,\quad Z_2 p_4 = Z_3,\ Z_3 p_5 = Z_4, \ldots \end{aligned}$$

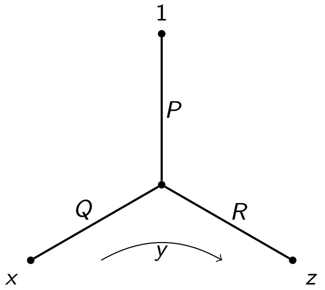
Each equation is now *triangular*, i.e. it has length 3.

Step 2. Free groups $\longrightarrow$ free monoids

In a free group, $xy = z$ holds if and only if there are reduced words P, Q, R with

$$x = PQ, y = Q^{-1}R, z = PR$$

as *word equations* in the free monoid over $A = \{a_1, a_1^{-1}, \ldots, a_d, a_d^{-1}\}$.



Step 3: Free monoids with involution

- Write a^{-1} as $\overline{a}$ and X^{-1} as $\overline{X}$.

  The map $a \mapsto \overline{a} : A \to A$ is an *involution*, i.e. $\overline{(\overline{a})} = a$ for all $a \in A$.

- We have a system of *word equations* $k_i l_i = m_i$ over $A \cup \Omega$

  (where $A = \{a_1, \overline{a_1}, \ldots, a_d, \overline{a_d}\}$ and Ω now includes $\overline{X}_i, Z_i, \overline{Z}_i, P, \overline{P}$, etc.)

  and we require that solutions do not contain $a\overline{a}$ or $\overline{a}a$.

- Finally put the system $k_i l_i = m_i$ into a single equation

$$k_1 l_1 \# k_2 l_2 \# \ldots \# k_s l_s = m_1 \# m_2 \# \ldots \# m_s$$

  and insist that the letter $\# \notin A$ does not appear in any solution.

Step 3: Free groups $\longrightarrow$ free monoids with constraints

Now let $A := \{a_1, \overline{a_1}, \ldots, a_d, \overline{a_d}, \#\}$.

How do we find solutions $\{X_i \rightarrow u_i\}$ to a *word equation* such that:

- $u_i \in A^*$ is not allowed to contain any subwords $a\bar{a}$,
- $u_i \in A^*$ is not allowed to use the letter $\#$?

Step 3: Constraints

Let $N = (A \times A) \cup \{0, 1\}$ be the *finite monoid* with multiplication:

$$x * 0 = 0 = 0 * x$$

$$x * 1 = x = 1 * x$$

$$(a, b) * (c, d) = \begin{cases} (a, d) & b \neq \overline{c} \\ 0 & b = \overline{c} \end{cases}$$

Define a *monoid homomorphism* (which respects the involution) $\rho : A^* \to N$ by $\rho(a) = (a, a)$ for all $a \in A \setminus \{\#\}$, $\rho(\#) = 0$ and $\rho(1) = 1$.

Example:

$$\begin{aligned} \rho(abca\overline{a}bc) &= \rho(abca) * \rho(\overline{a}bc) \\ &= (a, a) * (\overline{a}, c) \quad = \quad 0 \end{aligned}$$

So $\rho(u) \neq 0$ if and only if u is *reduced* and doesn't contain $\#$.

The proof: key process

Main idea of the proof: example $aXXb = YX$

- We start *guessing* the first and last letters of the variables, and substitute:

  $X \to aX$ $\qquad$ $aaXaXb = YaX$

  $X \to Xb$ $\qquad$ $aaXbaXbb = YaXb$

  $X \to aX$ $\qquad$ $aaaXbaaXbb = YaaaXb$

- We get *long segments* of constants in between variables.
- We *compress* these segments, to bring the equation length back down.
- Eventually, we will substitute $X \to 1$ and $Y \to 1$, and be left with two words just in constants. If they are identical we accept, else reject.

Goal: Find all solutions to the word equation $U = V$ satisfying the constraint ρ.

- We first *guess* $\rho(X) \in N \setminus \{0\}$ for each $X \in \Omega$.

- We then apply the following moves to the equation:

  - *pop* variables: $X \to aX$
  - *compress pairs* of constants $ab \to c$ where c is a new constant.
  - *compress blocks* of letters $aa \ldots a \to a_\ell$ where a_ℓ is a new constant.
  - Eventually, we will substitute $X \to 1$ and $Y \to 1$, and be left with two words just in constants. If they are identical we accept, else reject.

Comments

- The first move (pop) increases the length of the equation, but gets us closer to a solution.
- The two compression moves, applied many times, will reduce the length of the equation, at the expense of enlarging the set of constants.

Constructing the NFA: the vertices

We represent this process using a finite directed graph.

Vertices — labeled by the current state of the equation, plus some extra data.

- An initial vertex is a vertex containing the initial equation together with some guess for the constraint morphism ρ for each variable X.
- Final vertices — equation with no variables and both sides identical.

Constructing the NFA: the transitions

(I) As we move between vertices, variables will be replaced (*eg* $X \to aX$), and the value of ρ will be updated.

(II) Also, we have two types of compression:

- pair: $ab \to c_{ab}$
- block: $bbb \ldots b \to b_\ell$

The nondeterministic finite automaton

So we need *more constants* than the original set A. Call this set C.

We define several types of edges, such that solutions and constraints are preserved by an edge, and each edge is labeled by a morphism h of C^*.

To ensure the graph is *finite*, we must

- only use (and reuse) finitely many new constants,
- have a global bound on the length of any intermediate equation.

How do we get the solutions?

If there is a path from some

- initial vertex to a
- final vertex (with $P = P$ and no variables)

then we can apply the maps h labeling the path *backwards* from final to initial and recover a solution to $U = V$.

Is this graph the correct one?

The graph can then be turned into a finite state automaton, accepting a language R of morphisms. The set of all solutions becomes the (EDT0L) language $\{h(\$) \mid h \in R\}$.

The key of the proof is to show

(1) that every answer we get is indeed a solution, and

(2) we get all solutions.

Dealing with the two issues

(1) every answer we get is indeed a solution:

the graph was constructed to preserve solutions,

(2) we get all solutions:

the most technical and complicated part of the paper.

Thank you!