Combining Word Equations, Regular Languages and Arithmetic: (Some of) What We Know and What We Don't

Joel D. Day, Matthew Konefal, Vijay Ganesh, Nathan Grewal and Florin Manea

In this talk...

- $\Sigma = \{a, b, \ldots\}$ is a finite alphabet with $|\Sigma| \geq 2$
- $\mathcal{X} = \{X, Y, Z \ldots\}$ is an infinite set of variables
- $|w|$ is the length of a word w
- $w^n = \underbrace{w\,w\,\ldots\,w}_{n \text{ times}}$
- v is a factor (substring) of w if $w = uvx$ for some u, x
- A (QF) formula is a Boolean combination of atoms of some specified type(s)
- A (QF) theory is a set of all formulas containing atoms of some specified type(s)

Word Equations

Variables from $\mathcal{X}$

$$X\ ab\ Y\ c \doteq Z\ X\ c\ Y$$

Terminal Symbols from Σ

- $\alpha \doteq \beta$ where $\alpha, \beta \in (X \cup \Sigma)^*$
- True for $h : \mathcal{X} \to \Sigma^*$ if both sides become identical under h
- Let WE denote the set of all formulas whose atoms are word equations

Regular Constraints

Variable from $\mathcal{X}$ Regular Language

$$\underset{\uparrow}{X} \in (ab \mid ba \mid a)^*$$

- $X \in L$ where L can be given as a finite automaton or regular expression
- True for $h : \mathcal{X} \to \Sigma^*$ if $h(X) \in L$
- Let $\mathrm{WE} + \mathrm{REG}$ denote the set of all formulas whose atoms are word equations or regular constraints

Length Constraints

$$\underbrace{|X| = |Y|}_{}$$

Variables from $\mathcal{X}$

$$|X| = |Y|$$

- True for $h : \mathcal{X} \to \Sigma^*$ if $|h(X)| = |h(Y)|$
- Let $\mathrm{WE} + \mathrm{LEN}$ denote the set of all formulas whose atoms are word equations or length constraints
- Let $\mathrm{WE} + \mathrm{REG} + \mathrm{LEN}$ denote the set of all formulas whose atoms are word equations, regular constraints or length constraints

Summary of Theories

Theory	$\neg, \vee, \wedge$	$\alpha \doteq \beta$	$x \in L$	$\lvert X\rvert = \lvert Y\rvert$
WE	✓	✓		
WE + REG	✓	✓	✓	
WE + LEN	✓	✓		✓
WE + REG + LEN	✓	✓	✓	✓

- We can model $|X| > |Y|$ as $|X| = |Z| \wedge Z \doteq YW \wedge \neg(W \doteq \varepsilon)$
- Linear combinations like $2|X| + 3|Y| + 1 = |Z|$ can be modelled e.g. as $W \doteq XXYYYa \wedge |W| = |Z|$

What Do We Want to Know?

- Complexity/computability/algorithmic
  - Satisfiability
  - When can a given formula be rewritten in a smaller or alternative theory?
  - $\vdots$
- Design decisions
  - Understanding expressivity/complexity trade-offs
  - Search heuristics for satisfying assignments
- Expressivity
  - Which properties can(not) be expressed in a theory?
  - Pumping/structural properties for expressible relations/languages

Expressivity

Expressible Languages and Relations

Definition (Adapted from Karhumäki, et al. 2000)

Let φ be a formula and $S = \{X_1, X_2, \ldots, X_k\}$ be a subset of the variables occurring in φ. Then the relation expressed by S in φ is the set:

$$L(\varphi, S) = \{(h(X_1), h(X_2), ..., h(X_k)) \mid h \text{ satisfies } \varphi\}$$

A relation R is expressible in a theory $\mathfrak{T}$ if there exists a formula $\varphi \in \mathfrak{T}$ and S such that $R = L(\varphi, S)$.

E.g. $\{w \in \Sigma^* \mid |w| \text{ even}\}$ is expressible in $WE + LEN$ via X in $X \doteq YZ \wedge |Y| = |Z|$

A Natural Hierarchy

$$
\begin{tikzcd}
& \text{All R.E. Languages} \arrow[d] & \\
& \mathrm{WE}+\mathrm{REG}+\mathrm{LEN} \arrow[dl] \arrow[dr] & \\
\mathrm{WE}+\mathrm{REG} \arrow[dr] & & \mathrm{WE}+\mathrm{LEN} \arrow[dl] \\
& \mathrm{WE} &
\end{tikzcd}
$$

Inexpressibility in WE

Theorem (Büchi, Senger 1990, Karhumaki, Mignosi, Plandowski 2000)

The languages $a^n b^n$ and $(a \mid b)^* c$ are not expressible in WE.

- $a^n b^n$ is expressed by X in the $\mathrm{WE} + \mathrm{LEN}$-formula:

$$X \doteq YZ \wedge Ya \doteq aY \wedge Zb \doteq bZ \wedge |Y| = |Z|.$$

- $(a \mid b)^* c$ is expressed by X in the $\mathrm{WE} + \mathrm{REG}$-formula:

$$X \in (a \mid b)^* c.$$

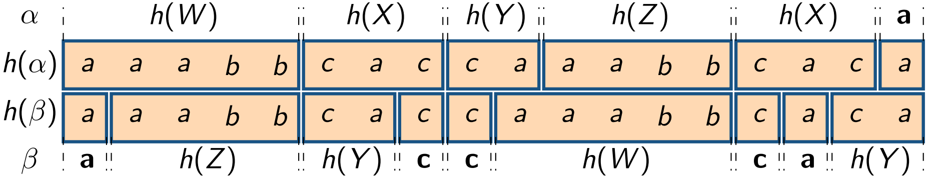
A Convenient Normal Form

Lemma (Folklore)

A language/relation is expressible in WE *if and only if it is expressible by a single positive word equation* $\alpha \doteq \beta$.

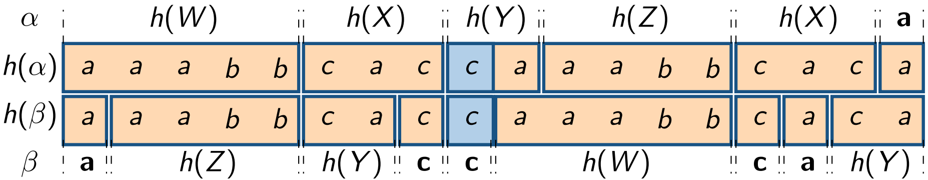
Filling the Positions and Unanchored Letters

$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



Filling the Positions and Unanchored Letters

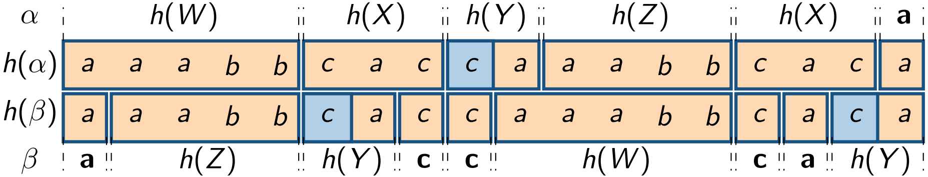
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



Vertically aligned positions must have the same letter

Filling the Positions and Unanchored Letters

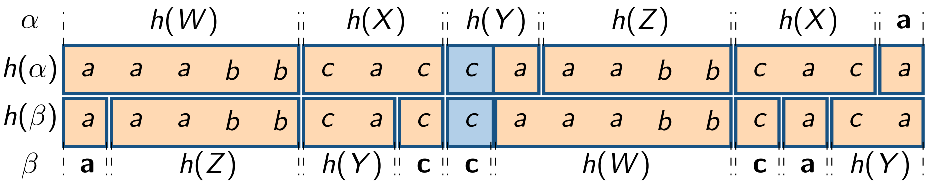
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



Positions occupying the same part of a variable must have the same letter

Filling the Positions and Unanchored Letters

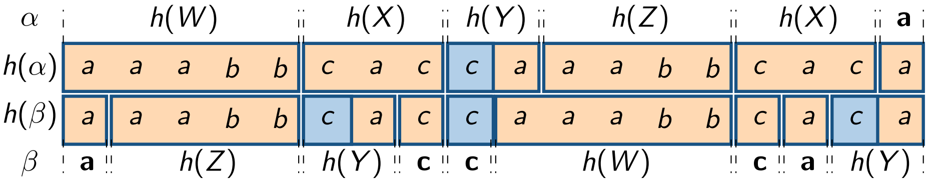
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



This leads to equivalence classes of positions which must have the same letter

Filling the Positions and Unanchored Letters

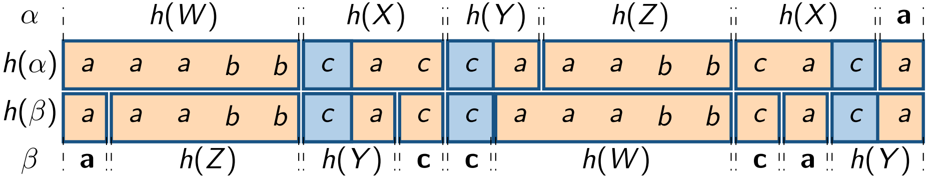
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



This leads to equivalence classes of positions which must have the same letter

Filling the Positions and Unanchored Letters

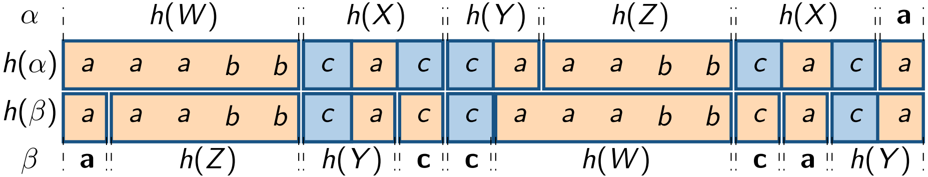
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



This leads to equivalence classes of positions which must have the same letter

Filling the Positions and Unanchored Letters

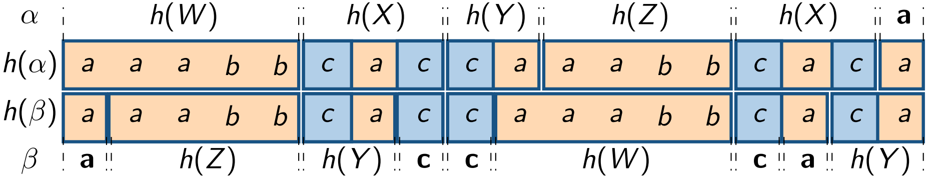
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



This leads to equivalence classes of positions which must have the same letter

Filling the Positions and Unanchored Letters

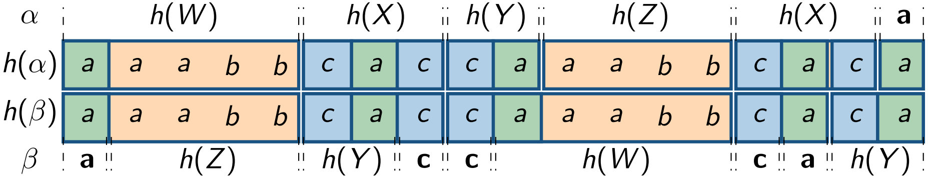
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



This leads to equivalence classes of positions which must have the same letter

Filling the Positions and Unanchored Letters

$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$



This leads to equivalence classes of positions which must have the same letter

Filling the Positions and Unanchored Letters

$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$

<table>
<tr><td>α</td><td colspan="5">$h(W)$</td><td colspan="3">$h(X)$</td><td colspan="2">$h(Y)$</td><td colspan="4">$h(Z)$</td><td colspan="3">$h(X)$</td><td>a</td></tr>
<tr><td>$h(\alpha)$</td><td>a</td><td>a</td><td>a</td><td>b</td><td>b</td><td>c</td><td>a</td><td>c</td><td>c</td><td>a</td><td>a</td><td>a</td><td>b</td><td>b</td><td>c</td><td>a</td><td>c</td><td>a</td></tr>
<tr><td>$h(\beta)$</td><td>a</td><td>a</td><td>a</td><td>b</td><td>b</td><td>c</td><td>a</td><td>c</td><td>c</td><td>a</td><td>a</td><td>a</td><td>b</td><td>b</td><td>c</td><td>a</td><td>c</td><td>a</td></tr>
<tr><td>β</td><td>a</td><td colspan="4">$h(Z)$</td><td colspan="2">$h(Y)$</td><td>c</td><td>c</td><td colspan="5">$h(W)$</td><td>c</td><td>a</td><td colspan="2">$h(Y)$</td></tr>
</table>

Some equivalence classes must take the value dictated by a constant from the equation (anchored)

Filling the Positions and Unanchored Letters

$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$

α	$h(W)$					$h(X)$			$h(Y)$		$h(Z)$				$h(X)$			**a**
$h(\alpha)$	a	a	a	??	b	c	a	c	c	a	a	a	??	b	c	a	c	a
$h(\beta)$	a	a	a	??	b	c	a	c	c	a	a	a	??	b	c	a	c	a
β	**a**	$h(Z)$				$h(Y)$		**c**	**c**	$h(W)$					**c**	**a**	$h(Y)$	

Others have no positions aligned to a constant, and can take any value (unanchored)

Filling the Positions and Unanchored Letters

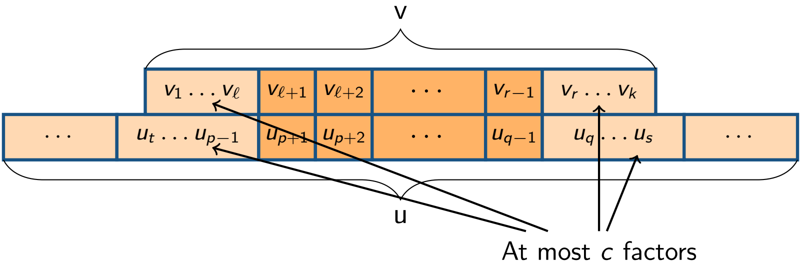
$$W\,X\,Y\,Z\,X\,a \doteq a\,Z\,Y\,cc\,W\,ca\,Y$$

<table>
<tr><td>α</td><td colspan="5">$h(W)$</td><td colspan="3">$h(X)$</td><td colspan="2">$h(Y)$</td><td colspan="4">$h(Z)$</td><td colspan="3">$h(X)$</td><td>**a**</td></tr>
<tr><td>$h(\alpha)$</td><td>a</td><td>a</td><td>a</td><td>??</td><td>??</td><td>c</td><td>a</td><td>c</td><td>c</td><td>a</td><td>a</td><td>a</td><td>??</td><td>??</td><td>c</td><td>a</td><td>c</td><td>a</td></tr>
<tr><td>$h(\beta)$</td><td>a</td><td>a</td><td>a</td><td>??</td><td>??</td><td>c</td><td>a</td><td>c</td><td>c</td><td>a</td><td>a</td><td>a</td><td>??</td><td>??</td><td>c</td><td>a</td><td>c</td><td>a</td></tr>
<tr><td>β</td><td>**a**</td><td colspan="4">$h(Z)$</td><td colspan="2">$h(Y)$</td><td>**c**</td><td>**c**</td><td colspan="5">$h(W)$</td><td>**c**</td><td>**a**</td><td colspan="2">$h(Y)$</td></tr>
</table>

Others have no positions aligned to a constant, and can take any value (unanchored)

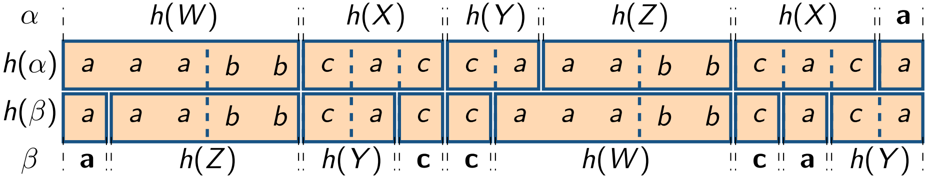
Synchronising Factorisation Schemes

- A **factorisation scheme** provides a unique way of splitting any given word $u \in \Sigma^+$ into factors $u = u_1 \cdot u_2 \cdot \ldots \cdot u_k$.
- It is **synchronising** if the factorisations of two overlapping words always align after a constant number of factors.



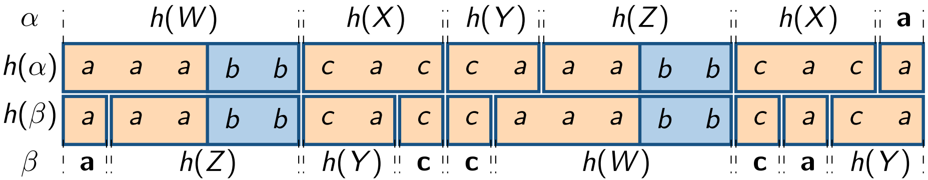
Filling the Positions and Unanchored **Factors**

- Dividing a word into runs of individual letters is **synchronising**
- We can generalise the filling the position methods to work for the factors of a synchronising factorisation scheme
- "Most" factors will line up nicely, but some will still overlap



Filling the Positions and **Unanchored Factors**

- It is still possible for some factors to be “unanchored”, meaning we can freely swap them to obtain other solutions



Existence of Unanchored Factors

Lemma (Karhumaki, Mignosi, Plandowski 2000, adapted)

Let $\mathfrak{F}$ be a synchronising factorisation scheme and let E be a word equation. There is a constant $C_{E,\mathfrak{F}}$ depending only on $\mathfrak{F}$ and $|E|$ such that if h is a solution to E and $h(X)$ has more than $C_{E,\mathfrak{F}}$ distinct factors in its $\mathfrak{F}$-factorisation, then at least one is unanchored.

Showing Inexpressibility: WE (Karhumäki et al. 2000)

$(a|b)^*c$

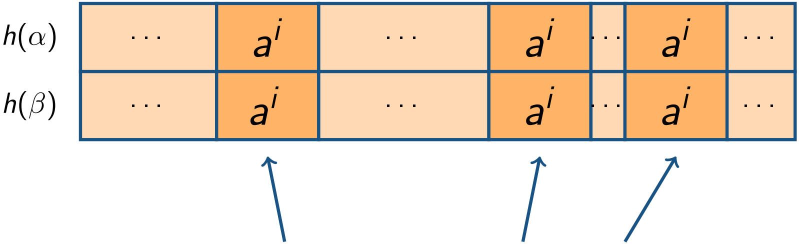
1. Choose a "good" factorisation scheme $\mathfrak{F}$
   E.g. blocks of letters, so $abbbaabaaa \rightarrow a \ \ bbb \ \ aa \ \ b \ \ aaa$

2. Assume L is expressed by X in E. Pick a word $w \in L$ such that w has more than $C_{E,\mathfrak{F}}$ distinct factors w.r.t. $\mathfrak{F}$
   E.g. $aba^2b^2a^3b^3 \ldots a^nb^nc$ for $n > C_{E,\mathfrak{F}}$

3. Take any solution h such that $h(X) = w$. At least one of the factors in w will be "unanchored" and we can freely replace it with any word $u \in \Sigma^*$
   E.g. swapping a^i for c

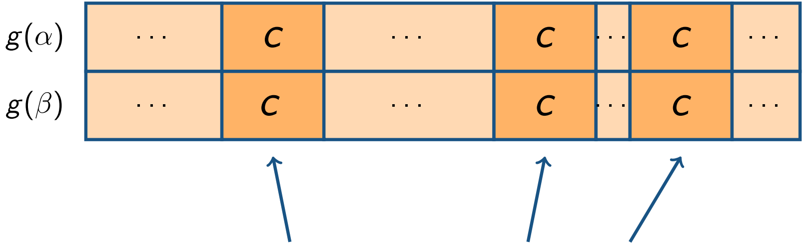
4. If we chose w, $\mathfrak{F}$ and u well, we get a new solution g such that $g(X) = w'$ for some $w' \notin L$ (a contradiction)

Showing Inexpressibility: WE (Karhumäki et al. 2000)



All occurrences of a^i line up exactly

Showing Inexpressibility: WE (Karhumäki et al. 2000)



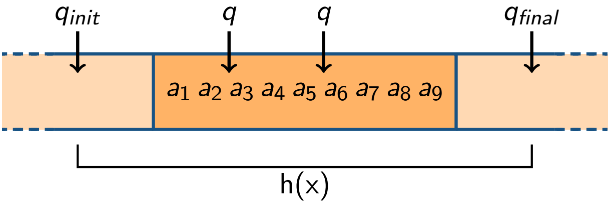
So we can swap a^i for c without affecting the equality of both sides

Showing Inexpressibility: WE + LEN

Adapting this approach to work for WE + LEN is straightforward, we just need to preserve the lengths when swapping factors

E.g. swapping a^i for c^i

Adapting the same approach to work for WE + REG requires a bit more care, but can be done by an involved pumping argument.

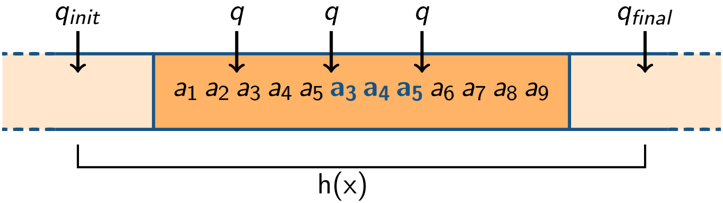


Showing Inexpressibility: WE + LEN

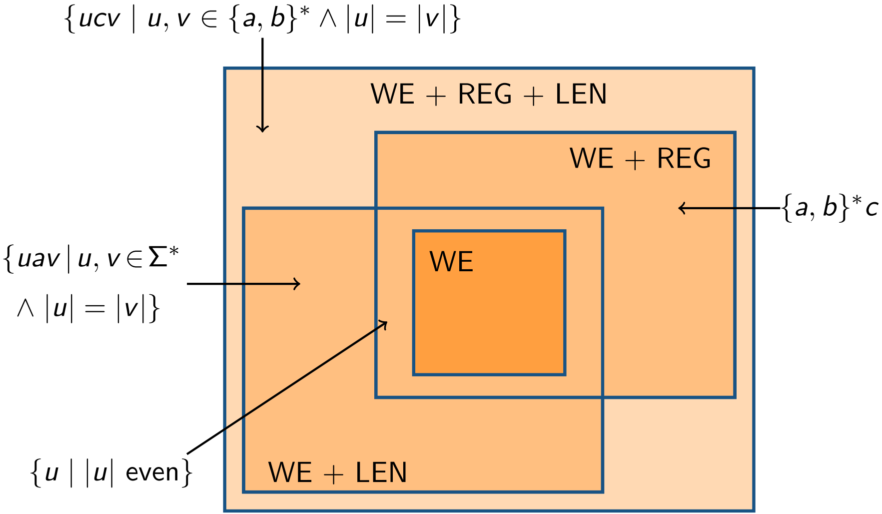
Adapting this approach to work for WE + LEN is straightforward, we just need to preserve the lengths when swapping factors

E.g. swapping a^i for c^i

Adapting the same approach to work for WE + REG requires a bit more care, but can be done by an involved pumping argument.



Separating the Theories



Showing Inexpressibility: WE + LEN + REG

Unfortunately, preserving lengths and pumping are incompatible when swapping out factors in a solution

Showing Inexpressibility: WE + LEN + REG

Unfortunately, preserving lengths and pumping are incompatible when swapping out factors in a solution

Theorem (Day, Ganesh, Grewal and Manea 2022)

There exist recursively enumerable languages which are not expressible in WE+REG+LEN.

Showing Inexpressibility: WE + LEN + REG

Unfortunately, preserving lengths and pumping are incompatible when swapping out factors in a solution

Theorem (Day, Ganesh, Grewal and Manea 2022)

There exist recursively enumerable languages which are not expressible in WE+REG+LEN.

Idea: Pump the "width" of the language (# of words of length n)

A Convenient Normal Form

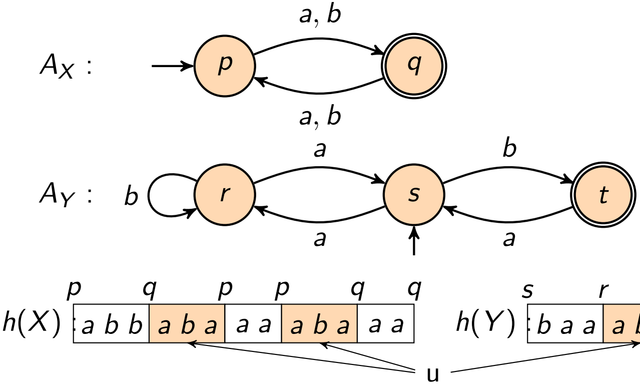
We can rewrite any $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN}$ formula expressing a given language into the form:

$$\bigvee_{1 \leq i \leq N} \left(E_i \wedge \psi_i^{len} \wedge \psi_i^{reg} \right)$$

where each E_i is a single word equation, ψ_{len_i} is a Boolean combination of length constraints and ψ_i^{reg} is a conjunction of regular constraints

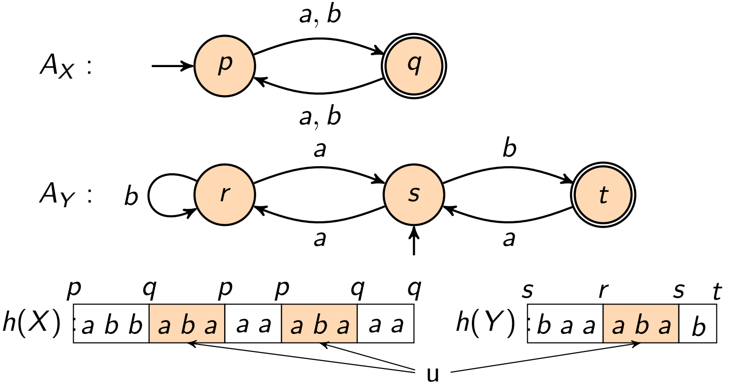
Inexpressibility for $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN}$

Suppose h is a solution to an equation E which satisfies some length constraints ψ^{len} and regular constraints given by A_X, A_Y.



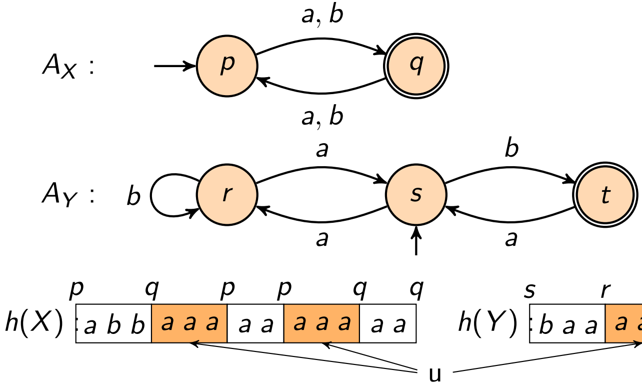
Inexpressibility for WE + REG + LEN

Suppose $u = aba$ is our unanchored factor. We can swap u for $v = aaa$ while still satisfying all constraints.



Inexpressibility for WE + REG + LEN

Suppose $u = aba$ is our unanchored factor. We can swap u for $v = aaa$ while still satisfying all constraints.

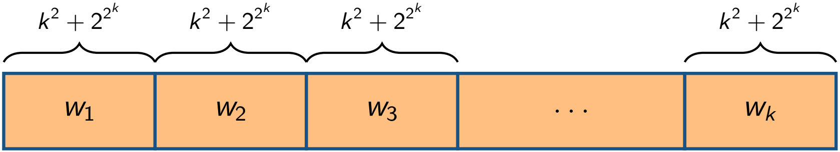


Inexpressibility for WE + REG + LEN

- Let Q be the set of pairs of states for which an occurrence of u starts/ends ($Q = \{(q,p),(p,q),(r,s)\}$ in the previous example)
- The set of words v which start/end in the same combinations of states as u is a regular language R_Q which can be computed from the original automata using the product construction.
- Swapping u for some $v \in R_Q$ means the equation and regular constraints remain satisfied.

A P(l)umping Argument

- We construct a R.E. language L so that each word $\in L$ contains k near-copies of some word $w \in \{a, b\}^k$, subject to different encodings over the same alphabet $a, b, c, d, @, \$$. We "pad" each copy so it has length $k^2 + 2^{2^k}$.
- The words in L have lengths $k^3 + k2^{2^k}$ for each $k \in \mathbb{N}$.
- Since there are 2^k choices of w for each k, there are $\Theta(\log(n))$ words of length n in L .



A P(l)umping Argument

- Suppose (for contradiction) that L is expressible by some formula φ from $\mathsf{WE} + \mathsf{LEN} + \mathsf{REG}$.
- The encoding means we can design a synchronising factorisation scheme which divides a word into its "copies" w_i.
- For all k large enough, at least one copy w_i of w is "unanchored". We associate each unanchored copy with the set Q of pairs of states it's occurrences start/end in w.r.t. to the regular constraints.

- The number of different sets Q is bound by aconstant C_{reg} depending only on φ^{reg}

- For sufficiently large k, there are at least $\frac{2^k}{C_{reg}} = \Omega(2^k)$ words of length $k^2 + 2^{2^k}$ whose occurrences start/end in pairs from Q.

- In other words, R_Q has at least $\Omega(2^k)$ words of length $\Theta(2^{2^k})$.

- Properties of regular languages dictate that the width of R_Q cannot be logarithmic, so R_Q must have $\Omega(2^{2^k})$ words of length $\Theta(2^{2^k})$.
- Since this means that for long-enough words in L, there is an unanchored factor which may be swapped for a near-linear number of alternatives while still satisfying the formula φ. This means that L contains a near-linear number of words of a given length.
- A contradiction, so L is not expressible.

Undecidability From Above

Generalising WE + REG + LEN

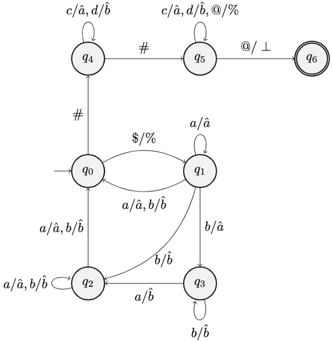
- It is a long-standing open problem if satisfiability is decidable for WE + LEN or WE + REG + LEN.
- Let WE + CF denote the set of formulas whose atoms are word equations or $X \in L$ where L is a context free language (CFL)
- Then WE + CF is powerful enough to model length constraints and regular constraints, but unfortunately satisfiability is undecidable

Theorem

Every R.E. language is expressible in WE + CF.

Generalising WE + REG + LEN

- What about languages between CFL and REG?
- We want a decidable intersection problem
- And to have enough "memory" to compare lengths
- Visibly Pushdown Languages (VPLs) fit the bill...



Visibly Pushdown Languages

- Partition Σ into Σ_{call}, Σ_{return} and $\Sigma_{internal}$.
- A language $L \subseteq \Sigma^*$ is a VPL if it is accepted by a pushdown automaton which
  - pushes when reading a letter from Σ_{call},
  - pops when reading a letter from Σ_{return},
  - leaves the stack unchanged when reading a letter from $\Sigma_{internal}$,
- VPLs are closed under intersection, union, complement, ...
  and have decidable emptiness, equivalence, inclusion problems

Generalising WE + LEN + REG

Let WE + VPL denote the set of formulas whose atoms are word equations or $X \in L$ where L is a visibly pushdown language

Theorem (Day, Ganesh, Grewal and Manea 2022)

All R.E. languages are expressible in WE + VPL.

Corollary (Day, Ganesh, Grewal and Manea 2022)

Satisfiability for WE + VPL *is undecidable.*

Decision Problems

Rewriting Problems: $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN} \rightarrow \mathsf{WE} + \mathsf{REG}$

Theorem (Day, Ganesh, Grewal, Manea 2022)

The following problem is undecidable:

Given a $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN}$*-formula* φ *and a non-empty subset* S *of the variables of* φ*, does there exist a* $\mathsf{WE} + \mathsf{REG}$*-formula* ψ *such that the relations expressed by* S *in* φ *and* ψ *are the same?*

Rewriting Problems: $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN} \to \mathsf{WE} + \mathsf{LEN}$

Open Problem

Is the following problem is decidable?

Given a $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN}$*-formula* φ *and a non-empty subset* S *of the variables of* φ*, does there exist a* $\mathsf{WE} + \mathsf{LEN}$*-formula* ψ *such that the relations expressed by* S *in* φ *and* ψ *are the same?*

Rewriting Problems: WE $\rightarrow$ REG

Theorem (Day, Ganesh, Grewal, Manea 2022)

The following problem is undecidable:

Given a WE*-formula* φ *and a variable* X *occurring in* φ *is the language expressed by* X *in* φ *regular?*

Rewriting Problems: REG $\rightarrow$ WE

Open Problem

Is the following problem decidable?

Given a regular language L, is L expressible in WE*?*

Rewriting Problems: REG $\to$ WE

A language L is **thin** if there is some word u which does not occur as a factor of any word in L.

Theorem (Day et al 2023)

Let e be a regular expression which does not contain $\emptyset$ and such that $L(e)$ is thin. Then $L(e)$ is expressible in WE *if and only if, for every subexpression of the form f^* of e, there exists w such that $L(f) \subseteq \{w\}^*$.*

Corollary (Day et al 2023)

It is decidable whether a thin regular language is expressible in WE.

Open Problem

Open Problem

Are languages expressible in $\mathsf{WE} + \mathsf{REG} + \mathsf{LEN}$ *decidable? Are they Context Sensitive?*

Thank You!