The Formal Language Theory Column

by

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20133 Milano, Italy
pighizzini@di.unimi.it

Formal Languages via Theories over Strings: An Overview of Some Recent Results

Joel D. Day [*] Vijay Ganesh [†] Florin Manea [‡]

Abstract

In this note, we overview a series of results that were obtained in [16, 15]. In these papers, we have investigated the properties of formal languages expressible in terms of formulas over quantifier-free theories of word equations, arithmetic over length constraints, and language membership predicates for the classes of regular, visibly pushdown, and deterministic context-free languages. As such, we have considered 20 distinct theories and decidability questions for problems such as emptiness and universality for formal languages over them. In this note, we first present the relative expressive power of the approached theories. Secondly, we discuss the decidability status of several important decision problems, some of them with practical applications in the area of string solving, such as the emptiness and the universality problem. To this end, it is worth noting that the emptiness problem for some theory is equivalent to the satisfiability problem over the corresponding theory. Finally, we discuss the problem of deciding whether a language expressible in one theory is also expressible in another one, and show several undecidability results; these investigations are particularly relevant in the context of normal forms for string constraints, and, as such, they are relevant to both the practical and theoretical side of string solving.

The current note is heavily based on the contents of [16, 15], and the readers are encouraged to check these references for complete details.

1 Introduction

Logical theories based on strings (or words) over a finite alphabet have been an important topic of study for decades, as described, for instance, in the two fundamental handbooks in the area of combinatorics on words [33, 32] as well as

[*]Loughborough University, United Kingdom, J.Day@lboro.ac.uk
[†]University of Waterloo, Canada, vijay.ganesh@uwaterloo.ca
[‡]Universität Göttingen, Germany, florin.manea@cs.informatik.uni-goettingen.de

in the volumes 1 and 3 of the handbook of formal languages [39, 40]. Connections to arithmetic (see, e.g.,Q̃uine [38]) and fundamental questions about free (semi)groups underpinned interest in logics involving concatenation and equality. Combining these two topics leads to word equations: expressions $\alpha = \beta$ where α and β are terms obtained by concatenating variables and concrete words over a finite alphabet. For example, if x and y are variables, and our alphabet is $\Sigma = \{a, b\}$, then $xaby = ybax$ is a word equation. Its solutions are variable-substitutions unifying the two sides: $x \to bb, y \to b$ would be one such a solution for the previous example; other solutions are $x \to b^{n+1}, y \to b^n$, for $n \geq 0$.

The existential theory of a finitely generated free semigroup consists of formulas made up of Boolean combinations of word equations. In fact, the problem of deciding whether such a formula is true is equivalent to determining satisfiability of word equations, since any such formula can be transformed into a single word equation without disrupting satisfiability (see [33, 27]). While it was originally hoped that the problem of deciding if a word equation has a solution could facilitate an undecidability proof for Hilbert's famous Tenth problem, by providing an intermediate step connecting Diophantine equations to the computations of Turing Machines, Makanin showed in 1977 that satisfiability of word equations is algorithmically decidable [35], putting an end to these hopes, but also opening a new line of research. Since then, several improvements to the algorithm proposed by Makanin have been proposed. From a complexity point of view, Plandowski [37] showed that this satisfiability problem can be solved in PSPACE, which has been refined to nondeterministic linear space by Jeż via the Recompression technique [25]. On the other hand, Schulz [41] showed that the problem remains decidable even when the variables are constrained by regular languages, limiting the possible substitutions (see Chapter 12 of [33]). On the other hand, if length constraints (requiring that some pairs of variables are substituted for words of the same length) are permitted, then the (un)decidability of the problem is a long-standing open problem.

Word equations, and logics involving strings more generally, have remained a topic of interest within the Theoretical Computer Science community, in particular in the areas of Combinatorics on Words and Formal Languages, where they play a fundamental role. More recently, word equations became interesting to the Formal Methods community as well. This interest can be attributed to increasing popularity and influence of software tools called string-solvers, which seek to algorithmically solve constraint problems involving strings [6, 22]. In this setting, a string constraint formalizes a property of an unknown string (or string variable), and the string solvers try to determine whether strings (over a potentially infinite domain) exist which satisfy logical combinations of string constraints of various types. Word equations, regular language membership, and comparisons between lengths are all among the most prominent building blocks of string constraints

(as described in [6]), and when combined are sufficient to model several others (see, for instance, some examples in [16]). Various other string constraints are discussed in, e.g., [14]. String-solvers are also useful in other areas like Database Theory, particularly for evaluating path queries in graph databases [7] and in connection with Document Spanners [19, 18]. Recently, to overcome some difficulties related to solving string constraints over infinite domains, a finite-model version of the theory of concatenation was considered [20].

Many string-solvers are now available [26, 8, 11, 36, 28, 1, 43, 9, 2] (see also [6, 22] for an overview). However, the underlying task of determining the satisfiability of string constraints remains a challenging problem and a barrier to more effective implementations. Motivated in part by the applications in string-solving, and by the desire to make progress on seemingly very difficult open theoretical problems, some results already exist addressing the computability, complexity, and expressibility of combinations of string constraints. [34, 21, 29, 31, 30] identify restrictions on word equations which result in a decidable satisfiability problem even when length constraints are present. Several further ways of augmenting word equations (i.e., additional predicates or constraints on the variables), are shown to be undecidable in [13, 14, 12, 23].

Nevertheless, despite results such as those mentioned above, little is known about the true expressive power of word equations and of string logics involving word equations in conjunction with other common types of string constraints. A greater understanding in this regard would be of great help in settling open problems (such as for whether satisfiability for word equations with length constraints is decidable), and also with devising string solving strategies: often simply finding a solution to one constraint is not enough and the set of solutions must be considered more generally to account for other constraints which might be present, or to determine that no solution exists. Moreover, a common tactic is to rewrite constraints into some normal form before solving and understanding when and how this can be done also requires knowledge of the relative expressive power of subsets of constraints.

Our work [16] filled some gaps in the understanding of the properties and expressivity of some of the most important combinations of string constraints by considering languages expressible in the sense of [27]. In this regard, our results can be seen as extending [27] to a more general (and more practical) setting, where word equations are combined with language membership constraints and length constraints.

The framework: In [16], a landscape of string-based logics was considered, incorporating various types of atoms inspired by and strongly related to prominent varieties of string-constraints. In particular, we considered logics with different combinations of the following four types of predicates: equality between strings, concatenations of strings, membership of formal languages, and linear

arithmetic over string-lengths. In total, this covered 20 distinct families of logical theories (each family containing a different theory for each possible underlying alphabet Σ). We will overview them in detail in Section 3. Based on [27] (and partly on [10], where relation-definability by logics over strings was studied in a database-theory centred framework), we have analysed these logics from a formal language perspective by looking at the set of values a variable may take while preserving satisfiability of a formula. Specifically, given a formula f from a quantifier-free logical theory $\mathfrak{T}$, we say that the language expressed by a variable x occurring in f is the set of concrete values w such that substituting x for w in f yields a satisfiable formula. In the general case, we can think of the property that the formula f defines via the variable x. However, since we deal with logics in which x is substituted for finite strings, we get a formal language.

In our approach, we were interested both in the expressive power of the logical theories with respect to what languages they can express, and in their computational properties with respect to canonical decision problems within formal languages such as emptiness, universality, equivalence and inclusion.

Getting more into details, the 4 types of fundamental predicates we allowed in these logics cover many of the most prominent types of string constraints, as listed in [6]. While predicates related to equality between strings, concatenations of strings, and linear arithmetic over string-lengths do not need more explanations, given the motivation presented above, a discussion is in order with respect to our choice of language membership predicates. In this case, they are considered for the classes of regular, deterministic context-free, or as an intermediary between the two, visibly pushdown languages. While there are many classes of languages we might have chosen to consider between regular and deterministic context-free, there are several advantages to choosing the visibly pushdown languages in particular. Firstly, they exhibit an attractive balance of being computationally reasonable (they have many of the desirable closure and algorithmic properties of the regular languages) while simultaneously being powerful enough to provide a reasonable model in many verification and software analysis applications, in line with our motivations from string-solving. Moreover, since they directly generalise the regular languages, but with sufficient memory capabilities to model certain types of length comparisons, the combination of word equations and visibly pushdown language constraints generalises the combination of word equations with both length and regular constraints. The latter is of particular interest in the context of string-solving, but is also a case for which the decidability of satisfiability remains open and is likely to be difficult to resolve. In [16], we have shown that the satisfiability for the former is undecidable and thus that already a very limited extension to regular and length constraints is enough to reach this negative result.

The results: Firstly, a comparison of the relative expressive power of the dif-

ferent theories was obtained. On the one hand, we managed to group certain families together, where they express the same class of languages. We have shown that adding linear arithmetic over string-lengths to a theory allowing only language membership predicates for a class of languages with good language theoretic properties does not alter its expressive power. Thus, the theories in which only regular language (or visibly pushdown language) membership predicates are allowed and the theories in which length comparison is added to those membership predicates are equivalent. While in the case of theories based on regular language membership predicates we can also add concatenation without changing the expressive power, we have shown that adding this operation to theories based on visibly pushdown language membership predicates strictly increases their expressive power, and they can express all recursively enumerable languages. Moreover, we also discuss several separation results between the classes of language expressed by various theories. One of the ways this can be achieved (see [16] for a detailed discussion) is by non-trivially extending pumping-lemma style tools for word equations from [27] to our more general settings, as well as by developing a novel technique for showing inexpressibility by word equations with both regular and length constraints. The overall hierarchy of classes of languages expressible in our theories is depicted in Figure 2.

While these results seem already interesting from a language-theoretic point of view, they are also relevant for the emptiness problem for classes of languages expressed by our theories, which is equivalent to the satisfiability problem for formulas over those theories. As such, our results allowed us to non-trivially extend the state-of-the-art related to the satisfiability of string constraints. In particular, we settled the previously mentioned interesting case in which word equations are combined with visibly pushdown language membership constraints. When combined with existing results, our results establish a relatively complete description of when the emptiness problem is (un)decidable (see upper part of Figure 2). The cases left open are the combinations of word equations with length constraints with or without regular constraints, which remain long-standing open problems.

Further, we have considered the universality problem and a related variant, namely the subset universality problem in which we want to test whether a language is exactly S^* for a subset S of the underlying alphabet. Again, our results filled in gaps in the knowledge and allowed us to paint a comprehensive picture of the decidability status of these problems for our theories (see the right part of Figure 2). Since the universal language is expressible in all our theories, in combination with results from Section 4 and from the literature, we have obtained a complete picture for the equivalence and inclusion problems. However, a substantial further benefit (and a large part of our motivation for studying this problem) is that it allows us to use Greibach's theorem in numerous instances (as stated in Theorem 8) to establish further undecidability results (e.g., Theorems 10 and 9).

In particular, Theorem 10 is part of a larger line of thought, developed in Section 6, in which we have considered the question of when it is (un)decidable if a language expressed in one theory can be expressed in another. Such problems are particularly interesting in the context of practical string solving because they essentially ask whether a property defined by one kind of string constraint can be algorithmically converted to another. Often, it is the combinations of different kinds of string constraints which lead to high complexities in solving, so being able to rewrite constraints in different forms can be a powerful pre-processing technique. We also identify some interesting cases where Greibach's theorem is not applicable, and thus where other approaches are needed (e.g., Theorem 11). The potential implications our results might have in practice are discussed in [16].

The structure of the presentation: Our aim in [16, 15] was to obtain a more complete understanding of the computational properties and expressivity of languages expressed by various combinations of commonly occurring types of string constraints. Naturally, we were able to account for several cases by recalling, or extending existing results from literature, so at the beginning of each section, we give a single theorem that summarizes existing results and discuss their consequences. This allowed us to subsequently focus on the most interesting remaining cases, many of which we were able to resolve by drawing on a range of techniques rooted in formal languages, automata theory, combinatorics on words and computability theory. The results reported in [16, 15] were a substantial improvement of the state of understanding of the theories considered, particularly with respect to their expressive power. In those cases, we were unable to resolve, we have identified several interesting new open problems, which we list here as well.

The proofs of the results overviewed in this paper are given in [16, 15].

2 Preliminaries

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$. Let $\mathbb{Z}$ denote the set of integers. Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. We denote by Σ^* the set of all words over Σ including the empty word, which we denote ε. In other words, Σ^* is the free monoid generated by Σ under the operation of concatenation. For words $u, v \in \Sigma^*$ we denote their concatenation either by $u \cdot v$ or simply as uv. Given a set of variables $\mathcal{X} = \{x_1, x_2, \ldots\}$ and an alphabet Σ, a *word equation* is a pair $(\alpha, \beta) \in (X \cup \Sigma)^* \times (X \cup \Sigma)^*$, usually written as $\alpha = \beta$. A solution to a word equation is a substitution of the variables for words in Σ^* such that both sides of the equation become identical. Formally, we model solutions as morphisms. That is, we say a substitution is a (homo)morphism $h : (X \cup \Sigma)^* \to \Sigma^*$ satisfying $h(a) = a$ for all $a \in \Sigma$, and a solution to a word equation $\alpha = \beta$ is a substitution h such that $h(\alpha) = h(\beta)$.

We refer to [24] for standard definitions and well-known results from formal language theory regarding, for example, recursively enumerable languages (RE), regular languages (REGLang), context free languages (CFLang), deterministic context-free languages (DCFLang), finite and pushdown automata, etc.

In addition, we refer to [3, 4, 5] for background on visibly pushdown automata and visibly pushdown languages (VPLang) but also give here the main definitions. More precisely, a pushdown alphabet $\widetilde{\Sigma}$ is a triple $(\Sigma_c, \Sigma_i, \Sigma_r)$ of pairwise-disjoint alphabets known as the call, internal and return alphabets respectively. A visibly pushdown automaton (VPA) is a pushdown automaton for which the stack operations (i.e. whether a push, pop or neither is performed) are determined by the input symbol which is read. In particular, any transition for which the input symbol a belongs to the call alphabet Σ_c, must push a symbol to the stack while any transition for which $a \in \Sigma_r$ must pop a symbol from the stack unless the stack is empty and any transition for which $a \in \Sigma_i$ must leave the stack unchanged. Acceptance of a word is determined by the state the automaton is in after reading the whole word. The stack does not need to be empty for a word to be accepted. A $\widetilde{\Sigma}$-visibly pushdown language is the set of words accepted by a visibly pushdown automaton with pushdown alphabet $\widetilde{\Sigma}$. A language L is a visibly pushdown language (and is part of the class VPLang) if there exists a pushdown alphabet $\widetilde{\Sigma}$ such that L is a $\widetilde{\Sigma}$-visibly pushdown language. The class VPLang is a strict superset of the class of regular languages and a strict subset of the class of deterministic context-free languages, which retains many of the nice decidability and closure properties of regular languages. In particular, it is shown in [3] that VPLang is closed under union, intersection and complement and moreover that the emptiness, universality, inclusion and equivalence probelms are all decidable for VPLang.

By a *theory*, we mean a set $\mathfrak{T} = \{f_1, f_2, \ldots\}$ of formulas adhering to given syntax and to which we associate a particular semantics. The theories we consider (introduced in Section 3) consist of quantifier-free formulas. The typical computational questions one might consider with respect to a given theory $\mathfrak{T}$ are the following:

- *Satisfiability*: given formula $f \in \mathfrak{T}$, does there exist an assignment of the variables in f such that f becomes true under the associated semantics? and
- *Validity*: given formula $f \in \mathfrak{T}$, is f true under all assignments of the variables occurring in f?

The questions we overview here have a slightly different flavour: given formula $f \in \mathfrak{T}$ and variable x occurring in f, we are interested in properties of the set of all values w for which there is an assignment mapping x to w which makes the formula true. Thus, we consider the set of concrete values w for which f remains satisfiable once the variable x has been replaced by w. Since we shall focus on

theories in which variables represent words, we refer to the set of all such values w as the *language* expressed by the variable x in the formula f. In this respect, we extend the notion of languages expressible by word equations [27] to arbitrary string-based logical theories. We say a language L is *expressed* by a formula if it contains a variable x such that L is the language expressed by x in f. We say that L is *expressible* in a theory $\mathfrak{T}$ if there exists a formula $f \in \mathfrak{T}$ and variable x occurring in f such that L is expressed by x in f.

We shall discuss typical decision problems such as emptiness and universality for languages expressed by formulas in a given theory $\mathfrak{T}$. In this context, the input is a formula $f \in \mathfrak{T}$ and a variable x occurring in f. So, e.g, in the case of emptiness, we might be given a formula $x = aba \wedge x \cdot y = ababba$ along with the variable y, and we must decide whether the language L_y expressed by y in that formula is the empty set or not. In this case, $L_y = \{bba\} \neq \emptyset$ so the answer is no. Clearly, for any formula f and variable x, the emptiness problem for the language expressed by x in f is equivalent to the satisfiability problem for f. Thus, we consider a set of problems which directly generalise the satisfiability problem.

For theories containing word equations, we use and extend notions and results from [27] to reason about (in)expressibility of languages, such as the notion of a *synchronising* $\mathfrak{F}$-factorisation. The technical details about such factorisations can be found in [27], as well as in [16].

3 Logical Theories Over Strings Constraints

In this section, we present a variety of logical theories encompassing the most common kinds of string constraints (as overviewed in [6]). Consider three sets of terms, defined as follows. Let $\mathcal{X} = \{x_1, x_2, \ldots\}$ be an infinite set of string variables. Let Σ be a finite alphabet. Let $\mathcal{T}^{\Sigma}_{str} = \mathcal{X} \cup \Sigma^*$ be the set of *basic string terms*. Let $\mathcal{T}^{\Sigma}_{str,con} = (\mathcal{X} \cup \Sigma)^*$ be the set of *extended string terms*. Note that $\mathcal{T}^{\Sigma}_{str,con}$ is the closure of $\mathcal{T}^{\Sigma}_{str}$ under the concatenation $(\cdot)$ operation. Let $\mathcal{T}^{\Sigma}_{arith} = \{k_0 + k_1|s_1| + k_2|s_2| + \ldots + k_n|s_n| \mid n \in \mathbb{N}_0, k_i \in \mathbb{Z}, \text{ and } s_i \in \mathcal{T}^{\Sigma}_{str}\}$ be the set of *length terms*. We interpret $|s|$ as the length of the string term s, so $\mathcal{T}^{\Sigma}_{arith}$ is the set of linear combinations of lengths of string terms. Note that since we can express the length of a concatenation of string terms as a linear combination of lengths of basic string terms, it is not a restriction the fact that $s_i \in \mathcal{T}^{\Sigma}_{str}$ rather than $\mathcal{T}^{\Sigma}_{str,con}$ (this allows us to consider theories containing length terms both with and without concatenation). We construct three types of atoms from terms as follows:

(A1) Language membership constraints of the form $s \in L$ where $s \in \mathcal{T}^{\Sigma}_{str(,con)}$ and $L \subseteq \Sigma^*$ is a formal language,

(A2) Length constraints of the form $\ell_1 = \ell_2$ where $\ell_1, \ell_2 \in \mathcal{T}^{\Sigma}_{arith}$,

(A3) Word equations (string-equality constraints) of the form $s_1 = s_2$ where $s_1, s_2 \in \mathcal{T}^{\Sigma}_{str,con}$.

Formulas in our theories are constructed in general as follows:

(F1) Any atom is a well-formed formula,

(F2) If f_1, f_2 are well-formed formulas then $\neg f_1$ is a well-formed formula and $f_1 \oplus f_2$ is a well-formed formula for each $\oplus \in \{\wedge, \vee, \implies, \iff\}$.

Note that all formulas are quantifier-free. The semantics associated with these formulas are defined in the natural way: given a substitution for the variables $x_1, x_2, \ldots$ for words in Σ^*, each string term evaluates to a word in Σ^* (possibly as the result of concatenating several smaller words in the case of extended string terms). Each length term is a linear combination of lengths of strings and evaluates to an integer. Atoms of type A1 evaluate to "true" if the string term s evaluates to a word in the language L and false otherwise. Atoms of type A2 evaluate to true if the two length terms ℓ_1, ℓ_2 evaluate to the same integer and false otherwise. Atoms of type A3 evaluate to true if the string terms s_1 and s_2 evaluate to the same word and false otherwise. Finally, Boolean combinations of the form F2 are evaluated in the canonical way.

The most general logical theory we consider includes all of the above and we consider language membership constraints $s \in L$ where L is a deterministic context-free language, given, for instance, as a deterministic push-down automaton or a context-free grammar. However, we are not just interested in this theory alone, rather we want to consider various sub-theories in order to compare their expressive power and computability-related properties.

We have two ways of restricting expressive power. The first is to restrict the types of terms/atoms we allow, while the second is to restrict the kind of languages we allow in the language membership constraints (atoms of type A1). For the latter, we focus on three main possibilities: regular languages, visibly push-down languages, and deterministic context-free languages. For technical completeness, we can assume that all language constraints are given as automata (NFA, Visibly-PDA, or Deterministic-PDA respectively), however, since we do not focus on precise complexity-related issues, equivalent language descriptors such as grammars could equally be used. In particular, we might use simpler descriptors where convenient to do so and where it is obvious that an equivalent automaton could be constructed.

We consider all combinations of atom-types A1, A2 and A3, and in each case define versions in which only basic string terms from $\mathcal{T}^{\Sigma}_{str}$ are allowed and versions in which concatenations of string terms (i.e. terms from $\mathcal{T}^{\Sigma}_{str,con}$) are allowed. Note that whenever we allow word equations (so, atoms of type A3), we might as well

Theory Name	A1-Atoms ($s \in L$)	A2	A3	Example
REG	s basic, $L \in$ REGLang	×	×	$x_1 \in a^*b^* \vee x_1 \in \{c\}^*$
VPL	s basic, $L \in$ VPLang	×	×	$x_1 \in \{a^nb^n \mid n \in \mathbb{N}\}$
DCF	s basic, $L \in$ DCFLang	×	×	$x_1 \in \{a^nb^{2n} \mid n \in \mathbb{N}\}$
REG + CON	s extended, $L \in$ REGLang	×	×	$x_1abx_2 \in (ab)^* \wedge x_2 \in b^*$
VPL + CON	s extended, $L \in$ VPLang	×	×	$x_1abx_2 \in \{a^nb^n \mid n \in \mathbb{N}\}$
DCF + CON	s extended, $L \in$ DCFLang	×	×	$x_1cx_2 \in \{ucv \mid u, v \in \{a, b\}^*, \lvert u\rvert = \lvert v\rvert\}$
REG + LEN	s basic, $L \in$ REGLang	✓	×	$x_1 \in a^* \wedge x_2 \in b^* \wedge \lvert x_1\rvert = \lvert x_2\rvert$
VPL + LEN	s basic, $L \in$ VPLang	✓	×	$x_1 \in \{a^nb^n \mid n \in \mathbb{N}\} \wedge \lvert x_1\rvert = 8$
DCF + LEN	s basic, $L \in$ DCFLang	✓	×	$x_1 \in \{a^nb^{2n} \mid n \in \mathbb{N}\} \vee \lvert x_1\rvert = 3\lvert x_2\rvert$
REG+LEN+CON	s extended, $L \in$ REGLang	✓	×	$x_1x_2 \in a^*b^* \wedge x_2 \in b^* \wedge \lvert x_1\rvert = \lvert x_2\rvert$
VPL+LEN+CON	s extended, $L \in$ VPLang	✓	×	$x_1x_2 \in \{a^nb^n \mid n \in \mathbb{N}\} \wedge \lvert x_1\rvert = \lvert x_2\rvert$
DCF+LEN+CON	s extended, $L \in$ DCFLang	✓	×	$x_1x_2 \in \{a^nb^{2n} \mid n \in \mathbb{N}\} \wedge \lvert x_2\rvert = 2\lvert x_1\rvert$
WE	×	×	✓	$x_1abx_2 = x_2bax_1$
WE + REG	s basic, $L \in$ REGLang	×	✓	$x_1abx_2 = x_2bax_1 \wedge x_1 \in a^*$
WE + VPL	s basic, $L \in$ VPLang	×	✓	$x_1 = x_2x_3 \wedge x_1 \in \{a^nb^n \mid n \in \mathbb{N}\}$
WE + DCF	s basic, $L \in$ DCFLang	×	✓	$x_1 = x_2x_3 \wedge x_2 \in \{a^nb^{2n} \mid n \in \mathbb{N}\} \wedge x_3 \in c^*$
WE + LEN	×	✓	✓	$x_1abx_2 = x_2bax_1 \wedge \lvert x_1\rvert = 2\lvert x_2\rvert$
WE+REG+LEN	s basic, $L \in$ REGLang	✓	✓	$x_1x_2 = x_3 \wedge x_1 \in a^*b^* \wedge \lvert x_2\rvert = \lvert x_3\rvert$
WE+VPL+LEN	s basic, $L \in$ VPLang	✓	✓	$x_1x_2 = x_3 \wedge x_3 \in \{a^nb^n \mid n \in \mathbb{N}\} \wedge \lvert x_1\rvert = \lvert x_3\rvert$
WE+DCF+LEN	s basic, $L \in$ DCFLang	✓	✓	$x_1x_2 = x_3 \wedge x_3 \in \{a^nb^{2n} \mid n \in \mathbb{N}\} \wedge \lvert x_2\rvert = \lvert x_3\rvert$

Figure 1: A list of all the theory-families addressed in the current work, along side descriptions of the allowed atom-types and an example of a formula belonging to that theory family, in the case that the underlying alphabet $\Sigma = \{a, b, c\}$. Language membership constraints are given in shorthand for readability. In the case of visibly pushdown languages, $\{a^nb^n \mid n \in \mathbb{N}\}$ is a typical example under an alphabet-partition $\Sigma = (\Sigma_c, \Sigma_i, \Sigma_r)$ satisfying $a \in \Sigma_c$ and $b \in \Sigma_r$. For each theory-family, permitted atom-types are indicated by a ✓ in the case of A2 and A3, and, for A1-atoms, the class of languages and which kind of string terms are allowed are written explicitly. Atom-types which are not permitted are indicated with ×.

allow concatenations of string terms. If we allow concatenations in word equation terms, then we can model concatenation in all string terms anyway and if we were to restrict equality between string terms to basic string terms only, then we could easily eliminate all string equalities by direct substitution.

Moreover, we are not going to consider explicitly the case that only length constraints (atoms of type A2) are allowed, since this reduces to the existential fragment of Presburger arithmetic and is therefore not really a string-based logic. With these exclusions, we are left with a total of 20 theories to consider; see Figure 1. In fact, since the theories themselves depend on the underlying alphabet Σ, we have 20 families of theories. As such, it is convenient to introduce a naming convention for these (families of) theories.

If atoms of type A1 are allowed, we add either REG, VPL, or DCF to the name of the theory-family depending on the class of languages permitted: REGLang,

VPLang, or DCFLang, respectively. If atoms of type A2 are allowed, we add the abbreviation LEN, separated if necessary by a "+". Likewise, if atoms of type A3 are allowed, we add the abbreviation WE. Finally, if atoms of type A3 are not allowed, but extended string terms are (so we have concatenation but not equality between string terms), then we add the abbreviation CON. Note that CON is superseded by WE due to reasons explained above. For example, the most general theory which allows all three atom types (with deterministic context-free languages for atoms of type A1) is denoted by WE + DCF + LEN. Similarly, REG + LEN + CON describes the theory in which atoms of type A1 (where L is a regular language and s is an extended string term) and A2 are allowed.

For theories allowing VPLang membership constraints (i.e. belonging to families of the form VPL + ...), we assume a fixed partition of the alphabet Σ into the call, return and internal alphabets $\Sigma_c, \Sigma_r, \Sigma_i$. We conclude this section with the following remark.

Remark 1. *Since* REGLang *(respectively,* VPLang*) is closed under union, intersection and complement, the set of languages expressible in* REG *(respectively,* VPL*) is exactly* REGLang *(respectively,* VPLang*). However, the same is not true for* DCF *and* DCFLang*, since that class is not closed, for instance, under intersection. For* DCF *the expressible languages are exactly the Boolean closure of the deterministic context-free languages. Moreover, it can be inferred from well-known results on word equations (see, e.g., [27, 33]) that the languages expressed by* WE *are exactly those expressible by a single word equation in the sense of [27].*

4 Separation and Grouping of Theories

We are interested primarily in whether we can decide properties of a language expressed by a given formula and variable. Therefore, the first thing we consider is the relative expressive power of the various theories defined in the previous section. In particular, we want to understand how the classes of languages which may be expressed by a formula/variable from a given theory relate to each other. To make these comparisons formally, we define the following relation(s) on two logical theories $\mathfrak{T}_1, \mathfrak{T}_2$ whose formulas contain string variables.

Definition 1. *Let $\mathfrak{T}_1, \mathfrak{T}_2$ be theories whose formulas contain string-variables. We say that $\mathfrak{T}_1 \preceq \mathfrak{T}_2$ if, for every formula $f \in \mathfrak{T}_1$ and every (string) variable x occurring in f, there exists a formula $f' \in \mathfrak{T}_2$ and variable x' in f' such that the languages expressed by x in f and x' in f' are identical. Moreover, we say that $\mathfrak{T}_1 \sim \mathfrak{T}_2$ if both $\mathfrak{T}_1 \preceq \mathfrak{T}_2$ and $\mathfrak{T}_2 \preceq \mathfrak{T}_1$ hold. We write $\mathfrak{T}_1 \prec \mathfrak{T}_2$ if $\mathfrak{T}_1 \preceq \mathfrak{T}_2$ and $\mathfrak{T}_1 \not\sim \mathfrak{T}_2$.*

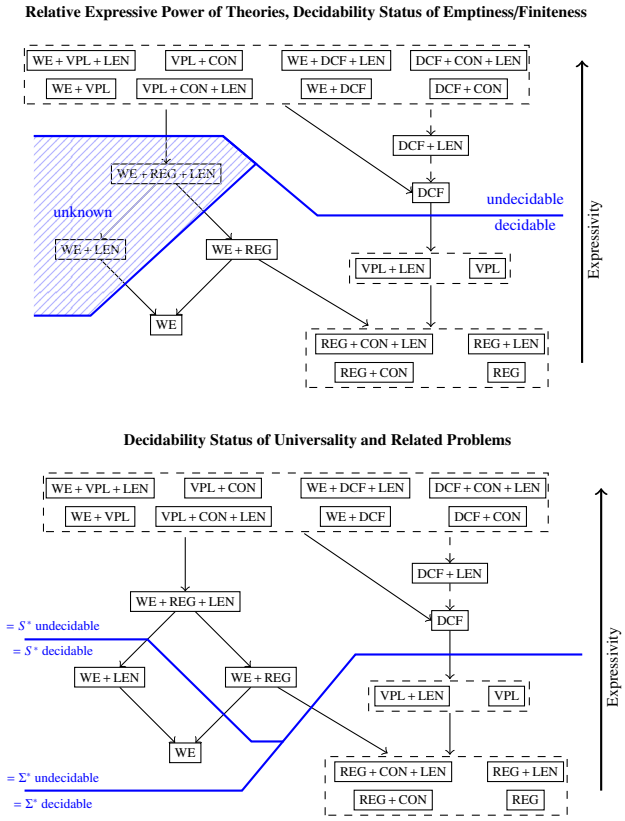

Figure 2: Visual representations of all 20 families of string-based logical theories considered. Theory-families are depicted in solid square boxes containing their names (see Section 3). Dashed square-boxes around multiple theory-families show equivalence with respect to the class of expressible languages (so equivalence under $\sim$). The arrows between theory-families and their transitive closure represent inclusion with respect to the class of expressible languages. Solid arrows indicate that the inclusion is known to be strict, while dashed arrows indicate that we do not know whether the inclusion is strict or not. The most expressive group of theories (i.e. those equivalent to VPL + CON) are able to express RE. The upper figure indicates for which (families of) theories the emptiness and finiteness problems are decidable or undecidable. The lower figure depictes (families) of theories for which the universality ($= \Sigma^*$) and subset-universality ($= S^*$) problems are decidable/undecidable. Equivalence and inclusion (where the two languages might come from different theories) are decidable if and only if both theories fall into cases where universality ($= \Sigma^*$) decidable.

Hence, $\mathfrak{T}_1 \preceq \mathfrak{T}_2$ if the class of languages expressible in $\mathfrak{T}_1$ is a subset of the class of languages expressible in $\mathfrak{T}_2$, and $\mathfrak{T}_1 \sim \mathfrak{T}_2$ if the two classes are equal. Note that the relation $\sim$ is an equivalence relation that is a weaker notion of equivalence than being isomorphic. That is, two theories need not be isomorphic to satisfy the equivalence $\sim$.

We extend Definition 1 for the families of theories defined in Section 3 as follows. Recall that each family contains all the theories consisting of a particular set of formulas, but whose underlying alphabet Σ may vary.

Definition 2. *Let $\mathfrak{F}_1, \mathfrak{F}_2$ be families of theories as defined in Section 3. We say that $\mathfrak{F}_1 \preceq \mathfrak{F}_2$ if, for every theory $\mathfrak{T}_1 \in \mathfrak{F}_1$, there is a theory $\mathfrak{T}_2 \in \mathfrak{F}_2$ such that $\mathfrak{T}_1 \preceq \mathfrak{T}_2$. The relations $\sim$ and $\prec$ are then defined analogously,*

Before moving on, let us make some remarks. It will often be the case that there exist formulas such that the language expressed by a variable x occurring in both formulas is the same, but the sets of satisfying assignments, when considered as a whole, are not identical (see Remark 2 below). This has an important implication for what conclusions we can and cannot draw from a statement of the form, e.g., $\mathfrak{T}_1 \sim \mathfrak{T}_2$. For example, while we will later show that REG $\sim$ REG + LEN, this does not imply that WE + REG + LEN $\sim$ WE + REG. Indeed we shall also show explicitly that the latter does not hold.

Remark 2. *Consider the* LEN *formula $|x| = 2|y|$ where x, y are string variables. Then the language expressed by x is the set of all even-length words over the underlying alphabet Σ, and the language expressed by y is simply Σ^*. Both of these languages are regular, and can be expressed in* REG. *However, if we were to consider, for example, a* WE + LEN *formula $x = yyy \wedge |x| = 2|y|$, then we cannot replace the condition $|x| = 2|y|$ with constraints based on the aforementioned regular languages. The problem with doing so would be that it allows us to decouple the sets of values for x and y satisfying the length constraint (so we get an x, y, x', y' such that $|x| = |y'|$ and $|x'| = |y|$ and $x = yyy$ holds, but where x' might be different from x and y' might be different from y.*

In [27] the authors consider expressibility of languages (and relations) by word equations and show that a language is expressible by WE if and only if it is expressible by a single word equation. The authors of [27] also show that, for $\Sigma \supseteq \{a, b, c\}$, the regular language $\{a, b\}^*$ is not expressible by a single word equation, and thus not in WE. The same holds for the language $\{a^n b^n \mid n \in \mathbb{N}_0\}$. Since these languages are clearly expressible in WE + REG and WE + LEN respectively, we may immediately conclude the following.

Theorem 1 ([27])**.** *The following hold:* WE $\prec$ WE + REG *and* WE $\prec$ WE + LEN.

On the one hand, all languages expressed in our theories are clearly recursively enumerable. On the other hand, in our first main result, we show that, in fact, all recursively enumerable languages can be expressed with only concatenation and VPL-membership.

Theorem 2. *The class of languages expressible in the familiy* VPL + CON *is exactly* RE.

Consequently, the class of languages expressible in each of VPL + CON + LEN, WE + VPL, WE + VPL + LEN, DCF + CON, DCF + CON + LEN, WE + DCF, and WE + DCF + LEN is the class of recursively enumerable languages RE. Thus, all these theories are equivalent under $\sim$. Clearly, we immediately get that the satisfiability problem for all these theories is undecidable. In fact, by Rice's Theorem, any non-trivial property is undecidable for languages expressible in these theories. As mentioned before, the case of WE + VPL is particularly interesting in the context of string solving and word equations. Since visibly pushdown languages are seemingly very close to regular languages, with many of the same positive closure and algorithmic properties, it is perhaps surprising to see that while satisfiability for WE + REG is decidable, satisfiability for WE + VPL is undecidable. Moreover, WE + VPL is a very natural extension of WE + LEN + REG. Since the satisfiability problem(s) for WE + LEN + REG and WE + LEN are both long standing open problems of significant interest both to the word equations and string-solving communities, the negative result for WE + VPL is both relevant and ominous.

In this context, it is now natural to ask whether we can separate the classes of languages expressible by WE + LEN + REG and WE + VPL, respectively. The existence of examples of recursively enumerable languages which are not expressible in the former is a necessary condition for having a decidable satisfiability problem, and if we wish to settle this open problem we must also settle the existence of such examples.

The next result does exactly this. In [16] we have established, with some involved argumentation, a sufficient criterion for languages to not be expressible in WE + LEN + REG and have used it to identify a concrete example of such language which is clearly recursively enumerable. To achieve this, we have first used techniques from [27], which were developed to show that certain languages are inexpressible by word equations only. We have first adapted these techniques in the presence of length constraints (so, to obtain languages which are not expressible in WE + LEN). With some care, we have also extended them for regular constraints (so to obtain languages which are not expressible in WE + REG). However, such techniques are altogether unsuitable for direct application in the presence of both length and regular constraints (so for WE + LEN + REG) and a novel approach was required. This new technique, as well as examples not expressible in the

aforementioned theories, are described in details in [16]. The result we obtain is the following.

Theorem 3. *There exist recursively enumerable languages which are not expressible in* WE + LEN + REG *and* WE + LEN + REG $\prec$ WE + VPL. *Moreover, we have* WE + LEN $\prec$ WE + REG + LEN *and* WE + REG $\prec$ WE + REG + LEN, *while the classes of languages expressible in* WE + LEN *and* WE + REG *are incomparable.*

With this result, the relations between the classes of languages expressed by the theories involving word equations is completely clarified; see the left side of Figure 2. Next, we turn our attention to the remaining theories which do not extend the expressive power of word equations. Since we have already seen that concatenation together with visibly pushdown (or deterministic context-free) membership constraints is enough to model recursively enumerable languages, and therefore word equations, the remaining theories consist of language membership without concatenation (but possibly with length constraints) and all combinations consisting of regular language membership constraints without word equations (so including either concatenation, length constraints, both, or neither).

In the following lemma, we state another important result. For this, let C be a class of formal languages which contains REGLang, is contained in CFLang, and is effectively closed under intersection and complement. We assume that the languages of C are specified by an accepting or generating mechanism which allows the construction of a context-free grammar generating that language. Let C_t be the theory defined as in Section 2 which allows only language membership predicates (of type A1) for the class of languages C. Let C_t + LEN be the theory which also allows length constraints. In this framework, the following holds (see the proof in [15]).

Lemma 1. $C_t + \mathrm{LEN} \sim C_t$.

As VPLang is a class which fulfills the properties of the class C from the above lemma, and is strictly included in RE, we immediately get the first claim of the following theorem. The second claim can also be shown with some additional effort (again, see [15]).

Theorem 4. *(1)* VPL $\sim$ VPL + LEN $\prec$ VPL + CON.
(2) REG $\sim$ REG + LEN $\sim$ REG + LEN + CON.

Recall that the languages expressible in REG (as well as the languages expressible in REG + LEN and REG + CON + LEN) and VPL (and VPL + LEN) are exactly the classes REGLang and, respectively, VPLang, and for each formula in one of these theories we can effectively construct a corresponding automaton accepting the language expressed by a given variable. See Remark 3 below.

Remark 3. *In fact, for a formula f in the theory* VPL + LEN *(which includes the theories* REG*,* REG + LEN + CON*, and* VPL*) we can effectively construct a formula g' which is a disjunction of conjunctions g_σ involving at most one membership predicate $x \in L_x$ per variable, where each language L_x is in* VPLang*. We can remove from g' the conjunctions g_σ which contain at least one membership predicate $x \in L_x$ with $L_x = \emptyset$. Now, it is easy to see that for the language expressed by x is exactly the union of the languages L_x for all membership predicates $x \in L_x$ occurring in g'. Therefore, this language is in the class* VPLang*, and we can effectively compute an automaton accepting it. Therefore, we can easily conclude that for two given formulae f and ϕ from* VPL + LEN *and a variable x occurring in f and a variable ϕ occurring in ϕ, we can decide whether the language expressed by x is the same as (respectively, included in) the language expressed by y.*

Let us now turn our attention to the theory DCF. The result of Lemma 1 does not apply in this case, as the class of languages DCFLang is not closed under intersection. In fact, for Lemma 1 to work, it would be enough to have that if L is a finite intersection of languages from the class C then the set $S = \{|w| \mid w \in L\}$ is semi-linear. However, this still does not hold for DCFLang. See Example 1 below.

Example 1. *Let $U_1 = \{a^n b^{2n} \mid n \geq 1\}$ and $L_1 = U_1^+$. Let $U_2 = \{b^n a^n \mid n \geq 1\}$ and $L_2 = aU_2^+ b^+$. It is clear that L_1 and L_2 are in* DCF*. Let $L = L_1 \cap L_2$. It is not hard to observe that $L = \{ab^2a^2b^4a^4b^8 \cdots a^{2^k}b^{2^{k+1}} \mid k \geq 1\}$. Further, let $S = \{|w| \mid w \in L\}$. We have that $S = \{2^{k+2} + 2^{k+1} - 3 \mid k \geq 1\}$. Clearly, S is not a semi-linear set (and it is not a deterministic context-free language either).*

In [15] we show an additional lemma.

Lemma 2. *$L = \{wcw \mid w \in \{a, b\}^*\}$ is expressible in* WE + REG *and not in* DCF*.*

By Theorem 2 and the existence RE-languages which are not expressible in DCF (see [45], as well as Lemma 2 or Example 1), we may infer the following relations: DCF $\preceq$ DCF + LEN $\preceq$ DCF + CON and DCF $\prec$ DCF + CON.

This also shows that at least one of the relations DCF $\preceq$ DCF + LEN and DCF + LEN $\preceq$ DCF + CON is strict. In fact, we can observe (see [15]) that the language $L = \{wcw \mid w \in \{a, b\}^*\}$, which is expressible in DCF + CON and not DCF, is not expressible by a restricted set of formulas in DCF + LEN. Accordingly, this indicates that the separation might occur between DCF + LEN and DCF + CON. We leave the following problem as open.

Open Problem 1. *Investigate whether the relations* DCF $\preceq$ DCF + LEN *and* DCF + LEN $\preceq$ DCF + CON *are strict or not (and note that* DCF $\prec$ DCF + CON*).*

As said before, REG + LEN + CON and VPL + LEN express exactly the classes of regular languages and VPL languages, respectively. Since the regular languages are a strict subset of the VPL languages, which in turn are a strict subset of the deterministic context-free languages, we may conclude the following strict inclusions in terms of expressibility: REG + LEN + CON $\prec$ VPL + LEN $\prec$ DCF.

Finally, note that there are languages expressible in WE (such as $\{xx \mid x \in \Sigma^*\}$) which are not regular nor visibly pushdown, and thus not expressible in REG or VPL or theories with equivalent expressibility. So, REG $\prec$ WE + REG holds. Moreover, we have already seen examples of regular languages which are not expressible in WE or WE + LEN.

Based on the previous results, we can now also discuss the emptiness problem, and the closely related finiteness problem. This is particularly interesting since emptiness for a language expressed by a formula f and variable x corresponds exactly to the satisfiability problem for f. Based on existing literature [3, 24, 41, 33], it is not hard to show that emptiness and finiteness are decidable for VPL and WE + REG but undecidable for DCF.

On the other hand, two cases where it seems particularly difficult to settle the decidability status of the satisfiability and, therefore, emptiness problems are WE + LEN and WE + REG + LEN. Emptiness for the former in particular is equivalent to the satisfiability problem for word equations with length constraints which is a long-standing and important open problem in the field. Similarly, the latter is prominent in the context of string-solving and as such satisfiability/emptiness also presents an important open problem which is likely to be closely related to that of WE + LEN. Consequently, WE + VPL presents a particularly interesting case as a "reasonable" generalisation of WE + REG + LEN and, in the absence of answers regarding this theory, it makes sense to consider the same problems for theories with slightly more or slightly less expressive power. If we extend the expressive power as far as WE + DCF, then undecidability is inherited directly from DCF. However, satisfiability and emptiness remain decidable for VPL. Moreover, visibly pushdown languages share many of the desirable computational properties of regular languages, and, as discussed, we can view WE + VPL as a slighter generalisation of WE + REG + LEN. Nevertheless, due to our result from Theorem 2 of the previous section, we know that VPL + CON expresses already RE, so emptiness and finiteness are undecidable for VPL + CON, and consequently for WE + VPL and other families $\mathfrak{F}$ of theories satisfying VPL + CON $\preceq \mathfrak{F}$. The left part of Figure 2 summarizes the understanding of the emptiness and finiteness problems, as resulting from our results.

5 Universality, Greibach's Theorem, and Expressibility Problems

Universality is an important problem for a number of reasons. Firstly, undecidability of universality implies undecidability of equivalence and inclusion for any theory in which the universal language Σ^* is expressible (which is true in any string-based theory containing at least one tautology). Secondly, an undecidable universality problem is the foundation for Greibach's theorem, which is helpful for proving that many other problems are undecidable. For instance, we shall make use of Greibach's theorem to show several problems concerning expressibility of languages in different theories are undecidable. We recall Greibach's theorem below.

Theorem 5 ([24])**.** *Let C be a class of formal languages over an alphabet $\Sigma \cup \{\#\}$ such that each language in C has some associated finite description. Suppose $\mathcal{P} \subsetneq C$ with $\mathcal{P} \neq \emptyset$ and suppose that all the following hold:*

1. *C and $\mathcal{P}$ both contain all regular languages over $\Sigma \cup \{\#\}$,*
2. *$\mathcal{P}$ is closed under quotient by a single letter,*
3. *Given (descriptions of) $L_1, L_2 \in C$ descriptions of $L_1 \cup L_2$, L_1R and RL_1 can be computed for any regular language $R \in C$,*
4. *It is undecidable whether, given $L \in C$, $L = \Sigma^*$.*

Then the problem of determining, for a language $L \in C$, whether $L \in \mathcal{P}$ is undecidable.

Note that in order to apply Greibach's theorem, we need a variant of the universality problem to be undecidable which refers to a sub-alphabet, rather than the whole alphabet.

Definition 3. *Let $\mathcal{T}$ be a theory defined in Section 3 with underlying alphabet Σ and such that $|\Sigma| \geq 3$. The subset-universality problem is: given a formula $f \in \mathcal{T}$, variable x occurring in f and $S \subset \Sigma$ with $|S| > 1$, is the language expressed by x in f equal to S^*?*

We recall the following results:

Theorem 6 ([21, 17, 3, 24])**.** *Universality is undecidable for* WE *and* DCF*, and decidable for* VPL*. Subset-universality is decidable for* VPL *but not* DCF*.*

To discuss the equivalence and inclusion problems, it makes sense to consider them in a general setting where the two languages may be taken from different theories. We therefore consider equivalence and inclusion problems for pairs of theories $(\mathfrak{T}_1, \mathfrak{T}_2)$. Combining the known results above with the constructive equivalences pointed out in Remark 3, we easily get that equivalence and inclusion for $(\mathfrak{T}_1, \mathfrak{T}_2)$ are undecidable whenever at least one of $\mathfrak{T}_1, \mathfrak{T}_2$ contains WE or DCF, but they are decidable for all other pairs of theories. In a similar way, one can show that cofiniteness is undecidable for WE.

We may, clearly, propagate undecidability of universality and related problems upwards through families of theories containing WE (or DCF) as a syntactic subset, or apply Rice's theorem to get such results for all theories expressing RE. In [15], we also show the following.

Theorem 7. *Subset-universality is decidable for* WE + LEN *and undecidable for* WE + REG. *In particular, for S large enough, for any theory $\mathfrak{T}$ from* WE + REG *with underlying alphabet $\Sigma \supset S$, the problem of whether a language expressed in $\mathfrak{T}$ is exactly S^* is undecidable.*

Theorem 7 allows us to apply Greibach's Theorem to many theories defined in Section 3.

Theorem 8. *Let $\mathfrak{F}$ be a family of theories defined in Section 3 which contains* WE + REG. *For large enough alphabets Σ, if C is the class of languages expressible by the theory $\mathfrak{T} \in \mathfrak{F}$ with underlying alphabet Σ, then the conditions of Greibach's theorem are satisfied by C.*

In the following, we give an example application of Theorem 8 with respect to the pumping lemma for regular languages (see, e.g., [24]). Aside from defining an interesting superclass of the regular languages itself, there are many reasons to be interested in notions of pumping. For example, when considering (in)expressibility questions (even beyond the regular languages), as well as part of a strategy for producing satisfiability results in the context of length constraints or other restrictions. We use the pumping lemma for regular languages because it is well-known, but the ideas are easily adapted to other useful notions of pumping and closure properties more generally. We recall first this lemma.

Lemma 3 ([24])**.** *Let L be a regular language. Then there exists a constant c such that for every $w \in L$ with $|w| > c$, there exist x, y, z such that (i) $|xy| < c$, and (ii) $w = xyz$, and (iii) $xy^n z \in L$ for all $n \in \mathbb{N}_0$.*

Now, Theorem 8 can be applied in this context (see [15]).

Theorem 9. *It is undecidable whether a language expressed by a formula in a theory from* WE + REG *satisfies the pumping lemma for regular languages.*

Theorem 7 also tells us that we cannot use Greibach's theorem as stated to show that properties of languages expressible in WE + LEN are undecidable. We leave as an open problem whether an equivalent of Greibach's theorem can be adapted to this context:

Open Problem 2. *Is there an equivalent of Theorem 5 for the classes of languages expressible in* WE + LEN *or* WE*?*

6 Expressivity Problems

Further, we consider decision problems related to expressivity. These problems have the general form: given a language L expressed by a formula in a theory $\mathfrak{T}_1$ and given a second theory $\mathfrak{T}_2$, can we decide whether or not L can be expressed by a formula in $\mathfrak{T}_2$?

We begin by noting that since it is decidable whether or not a deterministic context-free language is regular (see [42, 44]), the same holds true for visibly pushdown languages, and hence whether a language expressed in VPL can be expressed in REG. Therefore, it is clearly decidable whether a language expressed in VPL is expressible in REG. The same holds for theories from families equivalent to VPL and REG under the relation $\sim$.

Naturally, since we have already seen that VPL + CON is capable of expressing all RE-languages, it is undecidable whether a language expressed in a theory from VPL + CON is expressible in a theory from any of the families which have strictly less expressive power.

The separation results from Section 4 and Theorem 8 together mean we can get the following negative results as a consequence of Greibach's theorem. They have a particularly relevant interpretation in the context of string solving in practice. Specifically, it is often the case that string-solvers will perform some preprocessing of string constraints in order to put them in some sort of normal form which will make them easier to solve. One natural thing to want to do in this process is to reduce the number of combinations of sub-constraints of differing types by converting constraints of one type to another. This is useful particularly in cases where the combinations are difficult to deal with together in general. Word equations, regular constraints and length constraints are one such combination (recall from Section 4 that satisfiability for the corresponding theory including all three types of constraint is an open problem, but if length constraints are removed then satisfiability becomes decidable). Unfortunately, the following theorem reveals that we cannot, in general, decide whether length constraints can be eliminated by rewriting them using only regular membership constraints and word equations.

Theorem 10. *It is undecidable whether a language expressed in* WE + REG + LEN *can be expressed in* WE + REG.

The same undecidability result holds if, instead of removing length constraints by rewriting them as regular membership constraints and word equations, we want to remove word equations constraints by rewriting them as regular language membership constraints (possibly also with length constraints which, in the absence of word equations, do not increase the expressive power due to Theorem 4). While this result can also be obtained via Greibach's theorem, we can, in fact, state a stronger version for which we need a novel approach, detailed in [15]. In particular, we can show that it is already undecidable whether a language expressible by word equations (without additional constraints) is a regular language (i.e., can be expressed in REG).

Theorem 11. *It is undecidable whether a language expressed in* WE *is regular. In other words, it is undecidable whether a language expressed by a formula from* WE *is regular.*

Just as interesting as the result reported in Theorem 11 is the converse problem, which remains open.

Open Problem 3. *Is it decidable whether a regular language is expressible by word equations?*

Although a trivial consequence of Theorem 11, it is somehow surprising that it remains undecidable if a word equation combined with regular constraints expresses a regular language. Clearly, every regular language is trivially expressible in WE + REG.

Finally, we note the remaining cases which correspond to removing regular language membership constraints in the presence of word equations, and removing length constraints in the presence of word equations but without regular constraints. Thus, we leave the following questions open:

Open Problem 4. *Is it decidable whether a language expressed in* WE + REG *(respectively, in* WE + REG + LEN*) can be expressed in* WE *(respectively, in* WE + LEN*)? Is it decidable whether a language expressed in* WE + LEN *can be expressed in* WE*?*

7 Conclusions

Logics based on strings or words are an important topic in fields such as combinatorics on words and formal methods. Motivated primarily by tasks arising in

the automated analysis of software, string-solving, a collection of infinite-domain constraint satisfaction problems whose primary underlying objects are strings, is an area of increasing importance. However, despite considerable improvement in our understanding of this topic, there remains a wealth of open problems and many theoretical topics, particularly involving word equations, are still wide open for new developments.

In [16, 15], which are overviewed in this note, we have studied a variety of string-based logics inspired by typical types of constraints occurring in string-solving applications, such as word equations, length equality constraints and language membership constraints. By considering the formal languages obtained by looking at the set of values a single variable in these logics might take as part of a satisfying assignment for a given formula, we were able to obtain several novel results regarding the relative expressive power of these theories resulting in the hierarchy depicted in Figure 2. Within this broader picture, we have been able to also add new results regarding the computability of canonical decision problems for formal languages such as emptiness, finiteness, universality, equivalence and inclusion (see also Figure 2). Together with existing results, this has allowed us to portray a relatively complete picture of when these problems are and are not decidable within our framework.

Our results on decision problems - in particular (a variant of) the universality problem - created also a framework allogin us to apply Greibach's theorem to obtain further undecidability results. We made use of this tool alongside results overviewed in Section 4 to prove that in various cases, it is undecidable whether or not a language expressed by one theory can also be expressed in another.

On the other hand, we have also highlighted several interesting new open problems in the cases where we are not able to settle the decidability status of certain problems. We expect that studying these problems will lead to valuable new insights and techniques for the theory of word equations, as well as in the theory of formal languages and string-solving, more generally.

References

[1] P. A. Abdulla, M. F. Atig, Y. Chen, L. Holík, A. Rezine, P. Rümmer, and J. Stenman. Norn: An SMT solver for string constraints. In *Proc. Computer Aided Verification (CAV)*, volume 9206 of *Lecture Notes in Computer Science (LNCS)*, pages 462–469, 2015.

[2] Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Bui Phi Diep, Julian Dolby, Petr Janků, Hsin-Hung Lin, Lukáš Holík, and Wei-Cheng Wu. Efficient handling of string-number conversion. In *Proceedings of the 41st ACM SIGPLAN*

Conference on Programming Language Design and Implementation, pages 943–957, 2020.

[3] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. 36th ACM Symposium on Theory of Computing (STOC)*, STOC '04, pages 202–211, 2004.

[4] Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for visibly pushdown languages. In *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114. Springer, 2005.

[5] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, 2009.

[6] Roberto Amadini. A survey on string constraint solving. *ACM Computing Surveys (CSUR)*, 55(1):1–38, 2021.

[7] P. Barceló Baeza and P. Muñoz. Graph logics with rational relations: the role of word combinatorics. *ACM Trans. Comput. Logic*, 18(2), 2017.

[8] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In *TACAS*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022. doi:10.1007/978-3-030-99524-9_24.

[9] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli. CVC4. In *Proc. Computer Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science (LNCS)*, pages 171–177, 2011.

[10] Michael Benedikt, Leonid Libkin, Thomas Schwentick, and Luc Segoufin. Definable relations and first-order query languages over strings. *J. ACM*, 50(5):694–751, 2003.

[11] Murphy Berzish, Mitja Kulczynski, Federico Mora, Florin Manea, Joel D Day, Dirk Nowotka, and Vijay Ganesh. An smt solver for regular expressions and linear arithmetic over string length. In *International Conference on Computer Aided Verification*, pages 289–312. Springer, 2021.

[12] J. R. Büchi and S. Senger. Definability in the existential theory of concatenation and undecidable extensions of this theory. In *The Collected Works of J. Richard Büchi*, pages 671–683. Springer, 1990.

[13] Taolue Chen, Yan Chen, Matthew Hague, Anthony W Lin, and Zhilin Wu. What is decidable about string constraints with the replaceall function. *Proceedings of the ACM on Programming Languages*, 2(POPL):1–29, 2018.

[14] J. D. Day, V. Ganesh, P. He, F. Manea, and D. Nowotka. The satisfiability of word equations: Decidable and undecidable theories. In I. Potapov and P. Reynier, editors, *In Proc. 12th International Conference on Reachability Problems, RP 2018*, volume 11123 of *Lecture Notes in Computer Science (LNCS)*, pages 15–29, 2018.

[15] Joel D. Day, Vijay Ganesh, Nathan Grewal, and Florin Manea. Formal languages via theories over strings. *CoRR*, abs/2205.00475, 2022. arXiv:2205.00475, doi:10.48550/arXiv.2205.00475.

[16] Joel D. Day, Vijay Ganesh, Nathan Grewal, and Florin Manea. On the expressive power of string constraints. *Proc. ACM Program. Lang.*, 7(POPL):278–308, 2023. doi:10.1145/3571203.

[17] V. G. Durnev. Undecidability of the positive $\forall\exists^3$-theory of a free semigroup. *Siberian Mathematical Journal*, 36(5):917–929, 1995.

[18] D. D. Freydenberger. A logic for document spanners. *Theory of Computing Systems*, 63(7):1679–1754, 2019.

[19] D. D. Freydenberger and M. Holldack. Document spanners: From expressive power to decision problems. *Theory of Computing Systems*, 62(4):854–898, 2018.

[20] D. D. Freydenberger and L. Peterfreund. The theory of concatenation over finite models. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 198 of *LIPIcs*, pages 130:1–130:17, 2021.

[21] V. Ganesh, M. Minnes, A. Solar-Lezama, and M. C. Rinard. Word equations with length constraints: What's decidable? In *Haifa Verification Conference*, 2012.

[22] Matthew Hague. Strings at mosca. *ACM SIGLOG News*, 6(4):4–22, 2019.

[23] Simon Halfon, Philippe Schnoebelen, and Georg Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

[24] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[25] A. Jeż. Word equations in nondeterministic linear space. In *Proc. International Colloquium on Automata, Languages and Programming (ICALP)*, volume 80 of *LIPIcs*, pages 95:1–95:13, 2017.

[26] Shuanglong Kan, Anthony Widjaja Lin, Philipp Rümmer, and Micha Schrader. Certistr: a certified string solver. In *CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 210–224. ACM, 2022.

[27] J. Karhumäki, F. Mignosi, and W. Plandowski. The expressibility of languages and relations by word equations. *Journal of the ACM*, 47:483–505, 2000.

[28] A. Kiezun, V. Ganesh, P. J. Guo, P. Hooimeijer, and M. D. Ernst. HAMPI: a solver for string constraints. In *Proc. ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pages 105–116. ACM, 2009.

[29] Quang Loc Le and Mengda He. A decision procedure for string logic with quadratic equations, regular expressions and length constraints. In *Asian Symposium on Programming Languages and Systems*, pages 350–372. Springer, 2018.

[30] Tianyi Liang, Nestan Tsiskaridze, Andrew Reynolds, Cesare Tinelli, and Clark Barrett. A decision procedure for regular membership and length constraints over unbounded strings. In *International Symposium on Frontiers of Combining Systems*, pages 135–150. Springer, 2015.

[31] Anthony W Lin and Pablo Barceló. String solving with word equations and transducers: towards a logic for analysing mutation xss. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 123–136, 2016.

[32] M. Lothaire. *Combinatorics on words*, volume 17. Cambridge university press, 1997.

[33] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, Cambridge, New York, 2002.

[34] Rupak Majumdar and Anthony W Lin. Quadratic word equations with length constraints, counter systems, and Presburger arithmetic with divisibility. *Logical Methods in Computer Science*, 17, 2021.

[35] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Sbornik: Mathematics*, 32(2):129–198, 1977.

[36] Federico Mora, Murphy Berzish, Mitja Kulczynski, Dirk Nowotka, and Vijay Ganesh. Z3str4: A multi-armed string solver. In Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan, editors, *Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings*, volume 13047 of *Lecture Notes in Computer Science*, pages 389–406. Springer, 2021. doi:10.1007/978-3-030-90870-6_21.

[37] W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *Proc. Foundations of Computer Science (FOCS)*, pages 495–500. IEEE, 1999.

[38] W. V. Quine. Concatenation as a basis for arithmetic. *The Journal of Symbolic Logic*, 11(4):105–114, 1946.

[39] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*. Springer, 1997. doi:10.1007/978-3-642-59136-5.

[40] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, Volume 3: Beyond Words*. Springer, 1997. doi:10.1007/978-3-642-59126-6.

[41] K. U. Schulz. Makanin's algorithm for word equations-two improvements and a generalization. In *International Workshop on Word Equations and Related Topics*, pages 85–150. Springer, 1990.

[42] R.E. Stearns. A regularity test for pushdown machines. *Information and Control*, 11(3):323–340, 1967.

[43] Minh-Thai Trinh, Duc-Hiep Chu, and Joxan Jaffar. Progressive reasoning over recursively-defined strings. In *International Conference on Computer Aided Verification*, pages 218–240. Springer, 2016.

[44] L. G. Valiant. Regularity and related problems for deterministic pushdown automata. *Journal of the ACM (JACM)*, 22(1):1–10, 1975.

[45] Detlef Wotschke. The boolean closures of the deterministic and nondeterministic context-free languages. In *GI Gesellschaft für Informatik e. V.*, pages 113–121. Springer, 1973.