

# Language equations

*Michal Kunc*<sup>1</sup>

*Alexander Okhotin*<sup>2,3</sup>

<sup>1</sup> Ústav matematiky a statistiky, Masarykova univerzita,  
Kotlářská 2, CZ–611 37 Brno, Czech Republic  
email: kunc@math.muni.cz

<sup>2</sup> Suomen Akatemia, Helsinki, Finland

<sup>3</sup> Matematiikan laitos, Turun yliopisto, Turku, FI–20014, Finland  
email: alexander.okhotin@utu.fi

**Abstract.** Equations with formal languages as unknowns naturally appear whenever sets of words are being used. In particular, they are fundamental for the theory of formal grammars, with systems of equations of the form  $X_i = \varphi(X_1, \dots, X_n)$  representing the inductive nature of the context-free grammars and their natural variants. Some variants of these systems naturally represent finite automata and a basic class of cellular automata. Equations of the general form are notable for their computational completeness, with universal computation encoded in extremely simple examples. The chapter provides a survey of the known results on language equations, classifying them according to the methods of research, and comparing similar properties of different families.

2010 Mathematics Subject Classification: 68Q45, 68Q42, 03D05, 03D35

Key words: Language equations, formal grammars, computability.

## Contents

1	Introduction	2
2	General properties of operations	3
3	Equations with one-sided concatenation	5
3.1	Representation of finite automata . . . . .	6
3.2	Regularity and decidability . . . . .	6
3.3	Complexity of decision problems . . . . .	7
3.4	Inequalities with union and one-sided concatenation . . . . .	7
4	Resolved systems of equations	8
4.1	Union and concatenation: context-free languages . . . . .	9
4.2	Union, intersection and concatenation: conjunctive languages . . . . .	10
4.3	Union, intersection and linear concatenation: trellis automata . . . . .	13
4.4	Complementation and concatenation . . . . .	14
4.5	Concatenation and various sets of Boolean operations . . . . .	15

5	Equations with constant sides	15
5.1	Regularity of maximal solutions	16
5.2	Complexity of decision problems	16
5.3	Generalizations to other operations	17
6	Equations of the general form	18
6.1	Upper bounds	18
6.2	Computationally universal solutions	18
6.3	Equations $XK = LX$ and related systems	19
6.4	Equations over a unary alphabet	22
6.5	Infinite systems of equations	24
6.6	Well quasi-orders	24
6.7	Inequations and identity checking	27
7	Equations with erasing operations	28

## 1 Introduction

Language equations are equations of the form  $\varphi(X_1, \dots, X_n) = \psi(X_1, \dots, X_n)$ , where the variables  $X_i$  represent formal languages, and the expressions  $\varphi$  and  $\psi$  are comprised of the variables, constant languages, and some language-theoretic operations. The first and the best known use of language equations was to define the semantics of the context-free grammars (Ginsburg and Rice [24]). Many other applications of language equations arise in virtually all areas where formal languages appear. Numerous theoretical results in this area have been obtained in recent years, ending with several characterizations of effective computability by equations of a very simple form.

In this chapter, the work on language equations is arranged according to the general form of equations and the sets of allowed operations. The properties of language equations particularly depend upon *monotonicity* and *continuity* of operations, defined in Section 2. The typical operations used in equations are concatenation and the Boolean set-theoretic operations. Concatenation is sometimes restricted to be linear<sup>1</sup> or one-sided linear.

Equations with one-sided concatenation and Boolean operations are notable for being directly expressible in monadic second-order logic over infinite  $k$ -ary trees ( $SkS$ ), described by Löding in another chapter of this handbook<sup>2</sup>. It follows that whenever such an equation has a unique solution, all languages in it are regular. The basic problems for these equations (such as testing whether a given system of equations has any solutions, a unique solution, etc.) are decidable by expressing them in  $SkS$  and using its decision procedure. Some more efficient special methods for dealing with these equations are considered in Section 3.

Section 4 is concerned with systems of the *resolved* form  $X_i = \varphi_i(X_1, \dots, X_n)$  with  $i \in \{1, \dots, n\}$ . These systems define languages inductively, and hence are naturally connected to formal grammars and parsing. Context-free grammars are the most well-known kind of such equations, with the operations restricted to union and concatenation.

<sup>1</sup>Defined in any previous chapters?

<sup>2</sup>\*\*cross-reference\*\*, Sect. 6.1

Using other sets of operations leads to natural variants and generalizations of the context-free grammars, such as conjunctive grammars [58].

The next kind of equations are those of the form  $\varphi(X_1, \dots, X_n) = C$  with constant right-hand sides, which are considered in Section 5. If the operations in the left-hand sides are limited to union and concatenation, then such a system can be analyzed within the syntactic monoid of all constants on the right-hand side. In particular, the regularity of all maximal solutions can be established, and these solutions can be constructed by an algorithm. Furthermore, solution existence can be effectively decided, and the computational complexity of testing this property is known.

A more general kind of system with left- and right-hand sides of an unrestricted form, and with the only limitation that the operations are continuous, is considered in Section 6. The upper bound on the expressive power of unique, least and greatest solutions of such equations is given by recursive, recursively enumerable (r.e.) and co-r.e. sets, respectively. This upper bound is actually reached by all but the very simplest equations. In particular, equations over a one-letter alphabet and using concatenation as the only operation are already computationally universal. However, there exist some special cases of equations, in which the solutions are known to be regular.

Finally, there are language equations with non-continuous operations such as homomorphisms and the quotient, which are discussed in Section 7. Such operations are typically erasing. Under the weak assumption that the operations are definable in first-order arithmetic, unique solutions of such equations must be hyper-arithmetical sets and conversely, an arbitrary such set over an arbitrary alphabet can be represented by a unique solution of a system using union, concatenation and quotient.

## 2 General properties of operations

Most of the language equations studied in the literature are limited to the basic operations on languages: concatenation, Boolean operations, occasionally the Kleene star. This section assumes an abstract outlook on the operations, regarding them simply as functions  $\varphi: (2^{A^*})^n \rightarrow 2^{A^*}$ , and presents two general properties of such functions that are particularly important in language equations: *continuity* and *monotonicity*.

Continuity of functions is generally understood as preserving limits. Define a *convergent sequence* of languages  $\{L_k\}_{k=1}^{\infty}$  by imposing that for every word  $w$  there is a number  $k_w$  with  $w$  belonging either to all  $L_k$  with  $k \geq k_w$ , or to none of them; denote  $\lim L_k = \{w \mid w \in L_{k_w}\}$ . Then an operation  $\varphi: 2^{A^*} \rightarrow 2^{A^*}$  is *continuous* if  $\varphi(\lim L_k) = \lim \varphi(L_k)$  for every convergent sequence of languages  $\{L_k\}_{k=1}^{\infty}$ . Convergence of a sequence of  $n$ -tuples of languages is defined componentwise, and the definition of continuity accordingly extends to functions of multiple arguments. Proving that Boolean operations and concatenation are continuous is an exercise. On the other hand, erasing homomorphisms are not continuous: if  $h(a) = \varepsilon$ , then  $h(\lim\{a^{\geq \ell}\}_{\ell=1}^{\infty}) = h(\emptyset) = \emptyset$ , but  $\lim\{h(a^{\geq \ell})\} = \lim\{\varepsilon\} = \{\varepsilon\}$ .

The notion of continuity of operations on languages defined above is actually the continuity of functions in a metric space defined by  $d(K, L) = 2^{-\min\{|w| : w \in K \Delta L\}}$ , where  $K \Delta L$  denotes the symmetric difference of  $K$  and  $L$ . This definition extends to  $n$ -tuples

of languages by setting  $d((K_1, \dots, K_n), (L_1, \dots, L_n)) = \max d(K_i, L_i)$ . This metric is an *ultrametric* because it satisfies the inequality  $d(K, L) \leq \max(d(K, M), d(M, L))$ , and the set of  $n$ -tuples of languages equipped with  $d$  forms a *compact ultrametric space*.

Two languages,  $K$  and  $L$ , are said to be *equal modulo*  $A^{\leq \ell}$ , for some  $\ell \geq 0$ , if a string  $w \in A^*$  of length at most  $\ell$  is in  $K$  if and only if it is in  $L$  [68]. This definition is naturally extended to  $n$ -tuples of languages. The set of languages equal to a given language modulo  $A^{\leq \ell}$  forms an open ball of radius  $2^{-\ell}$  in the metric space. Note that since the space is compact, continuity is equivalent to uniform continuity, and the latter can be expressed in terms of these balls as follows: an operation  $\varphi$  is continuous if for every  $\ell \geq 0$  there exists  $m = m(\ell) \geq 0$ , such that  $K = L \pmod{A^{\leq m}}$  implies  $\varphi(K) = \varphi(L) \pmod{A^{\leq \ell}}$ . For concatenation, Kleene star and Boolean operations, this definition of uniform continuity uses  $m(\ell) = \ell$ , that is, equivalence of arguments modulo  $A^{\leq \ell}$  implies the equivalence of results modulo  $A^{\leq \ell}$ . On the other hand, the quotient with a letter,  $\varphi(L) = a^{-1}L = \{w \mid aw \in L\}$ , is uniformly continuous with  $m(\ell) = \ell + 1$ . Examples of continuous operations with  $m(\ell)$  increasing faster than any given function can also be constructed.

Consider a system of language equations with continuous operations. An  $n$ -tuple of languages  $L$  is its *solution modulo*  $A^{\leq \ell}$ , if each equation holds modulo  $A^{\leq \ell}$  under the substitution of these languages for variables. Due to the continuity, this property depends only on the value of  $L$  modulo  $A^{\leq m}$ , for an appropriate  $m = m(\ell)$ . This notion leads to the following important characterization of solutions in the proper sense by solutions modulo  $A^{\leq \ell}$  for different numbers  $\ell$ .

A solution  $L$  modulo  $A^{\leq \ell}$  is said to be *extendable to*  $A^{\leq \ell'}$ , for a given  $\ell' \geq \ell$ , if there exists such a solution  $L'$  modulo  $A^{\leq \ell'}$ , that  $L = L' \pmod{A^{\leq \ell}}$ . Such a solution  $L$  modulo  $A^{\leq \ell}$  is said to be *extendable to a solution*, if the system has a solution (in the proper sense), that equals  $L$  modulo  $A^{\leq \ell}$ .

**Lemma 2.1** (cf. Okhotin [68]). *Let  $\varphi_i = \psi_i$  be a system of equations, in which  $\varphi_i$  and  $\psi_i$  are continuous functions. Then for every number  $\ell \geq 0$  there exists a number  $\ell' \geq \ell$ , such that all solutions modulo  $A^{\leq \ell}$  that are extendable to  $A^{\leq \ell'}$  are extendable to solutions.*

In the cited paper, this lemma is proved only for the special case of  $\varphi_i, \psi_i$  using Boolean operations and concatenation. However, with minimal modifications, the proof applies to any compact ultrametric space and any continuous operations.

From this lemma, one can infer the following necessary and sufficient conditions for solution existence and solution uniqueness:

**Theorem 2.2** (cf. Okhotin [68]). *A system of language equations with continuous operations has a solution if and only if for every  $\ell \geq 0$  there exists a solution modulo  $A^{\leq \ell}$ . A system has a unique solution if and only if for every  $\ell \geq 0$  there exists  $\ell' \geq \ell$ , such that the system has at least one solution modulo  $A^{\leq \ell'}$ , and all the solutions modulo  $A^{\leq \ell'}$  are equal modulo  $A^{\leq \ell}$ .*

The second important abstract property of operations is *monotonicity*. It is defined with respect to the partial order of componentwise inclusion of  $n$ -tuples of languages, under which  $(K_1, \dots, K_n) \sqsubseteq (L_1, \dots, L_n)$  if  $K_i \subseteq L_i$  for all  $i$ . An  $n$ -ary operation on languages  $\varphi: (2^{A^*})^n \rightarrow 2^{A^*}$  is *monotone* if  $\varphi(K) \subseteq \varphi(L)$  whenever  $K \sqsubseteq L$ .

For example, union, intersection and concatenation are both monotone and continuous, complementation is continuous but not monotone, while erasing homomorphisms are monotone but not continuous.

A large, commonly used class of monotone operations on languages are the *word-based operations*. Every such operation is induced by a multiple-valued operation on words  $f: (A^*)^m \rightarrow 2^{A^*}$  and defined as  $\hat{f}(L_1, \dots, L_m) = \bigcup_{w_i \in L_i} f(w_1, \dots, w_m)$ . Concatenation, quotient and shuffle are examples of such operations; union and intersection also fall under this definition. Note that though every word-based operation is monotone, it is not necessarily continuous: for example, the quotient operation on two languages is not continuous.

The partial order of  $n$ -tuples of languages by componentwise inclusion is also commonly applied to the set of solutions of a given system of equations. This allows talking about *minimal* and *maximal* solutions of an equation (that is, those, for which there is no smaller and no greater solution, respectively) as well as *least* and *greatest* solutions (those that are less than all solutions and greater than all solutions, respectively). Not every equation has a least or a greatest solution: for instance, the equation  $XY = \{a\}$  has two incomparable solutions:  $X = \{\varepsilon\}$ ,  $Y = \{a\}$  and vice versa.

**Lemma 2.3.** *Let  $\varphi_i = \psi_i$  be a system of equations, in which  $\varphi_i$  and  $\psi_i$  are continuous functions. Then every solution  $L$  of the system is contained in a maximal solution and contains a minimal solution.*

Such a maximal solution can be obtained as a componentwise union of an ascending sequence of solutions as follows (a minimal solution is obtained in the same way as a componentwise intersection). Let  $L^{(0)} = L$ . If a solution  $L^{(k)}$  is not yet maximal, then find a shortest word  $w$  which belongs to some component of some solution  $L^{(k+1)}$  greater than  $L^{(k)}$ , while not belonging to the corresponding component of  $L^{(k)}$ . Proceeding in this way either eventually leads to a maximal solution, or produces an infinite ascending sequence of solutions  $L^{(k)}$ , which converges to their componentwise union. Then this limit is also a solution, since continuity of operations implies

$$\varphi_i(\lim(L^{(k)})) = \lim(\varphi_i(L^{(k)})) = \lim(\psi_i(L^{(k)})) = \psi_i(\lim(L^{(k)})).$$

Because the length of the words  $w$  considered during the construction of the sequence  $L^{(k)}$  increases, if a word belongs to some component of a solution greater than  $\lim(L^{(k)})$ , then it must have been added to this component of  $L^{(k)}$  before longer words  $w$  were considered. Therefore, the solution  $\lim(L^{(k)})$  is maximal.

### 3 Equations with one-sided concatenation

In language equations of the simplest kind, the concatenation is *one-sided*, that is, may be used only in expressions  $C \cdot \xi$ , where  $C$  is a constant language and  $\xi$  is a subexpression (or, alternatively, only in expressions  $\xi \cdot C$ , but these two types of expressions may not be mixed in a single system). These equations are closely connected to finite automata.

### 3.1 Representation of finite automata

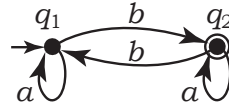
The most basic connection between language equations and computation is the representation of finite automata as systems of equations  $X_i = \varphi_i(X_1, \dots, X_n)$ , with union and one-sided concatenation. Each state  $q_i$  of an NFA  $\mathcal{A} = (Q, q_1, \delta, F)$  with  $Q = \{q_1, \dots, q_n\}$  is represented by a variable  $X_i$ , with its equation transcribing the outgoing transitions:

$$X_i = \bigcup_{(q_i, a, q_j) \in \delta} \{a\} \cdot X_j \quad \overbrace{\cup \{\varepsilon\}}^{\text{if } q_i \in F}$$

This construction is illustrated in the following example:

**Example 3.1.** The following system of language equations transcribes a two-state finite automaton recognizing the language  $L = \{w \in \{a, b\}^* \mid \text{the number of } b\text{s in } w \text{ is odd}\}$ :

$$\begin{cases} X_1 &= aX_1 \cup bX_2 \\ X_2 &= aX_2 \cup bX_1 \cup \{\varepsilon\} \end{cases}$$



It has a unique solution with  $X_1 = L$ .

This representation was first proposed by Bondarchuk [10] and further explored in the monograph by A. Salomaa [79]. Brzozowski and Leiss [11] extended these equations with all Boolean operations to define alternating finite automata.

### 3.2 Regularity and decidability

Consider equations of the general form  $\varphi(X_1, \dots, X_n) = \psi(X_1, \dots, X_n)$ , with all Boolean operations, one-sided concatenation of the form  $\xi(X_1, \dots, X_n) \cdot \{a\}$ , and constant  $\{\varepsilon\}$  (other regular constants may be represented as in the previous subsection). These equations can be directly expressed in the MSO logic on infinite trees, described in the chapter by Löding<sup>3</sup>. Nodes of these trees correspond to words, and each arc represents appending a letter to the end of the current word. Every expression  $\xi(X_1, \dots, X_n)$  occurring in an equation is transcribed as an MSO formula  $f_\xi(x)$ , with a first-order variable  $x$  and with second-order variables  $X_1, \dots, X_n$ , so that  $f_\xi(x)$  is true if and only if the word  $x$  is in  $\xi(X_1, \dots, X_n)$ . This is done inductively, with  $f_{X_i}(x) = (x \in X_i)$ ,  $f_{\xi \cdot a} = (\exists y)(x = ya \wedge f_\xi(y))$ ,  $f_{\xi \cup \theta}(x) = f_\xi(x) \vee f_\theta(x)$  and with the rest of the Boolean operations defined similarly. Finally, an equation  $\varphi = \psi$  is represented as a formula  $(\forall x) f_\varphi(x) \leftrightarrow f_\psi(x)$ . Then, by the general results on this logic, if a system of equations has any solutions, there must be a regular solution among them. All natural properties, such as uniqueness of a solution, can be directly expressed in the logic, and then decided by applying its decision procedure.

However, the above approach is too general to yield any efficient algorithms.

<sup>3</sup>\*\*cross-ref\*\*

### 3.3 Complexity of decision problems

Several efficient algorithms for testing properties of language equations with one-sided concatenation, as well as for computing their solutions, have been designed. In the more general case of set constraints, an exponential-time algorithm for testing solution existence was given by Aiken et al. [1]. This problem was proved to be EXPTIME-complete for language equations by Baader and Küsters [3], while Baader and Narendran [4] established EXPTIME-completeness of the existence of a finite solution. The idea of these arguments was generalized by Baader and Okhotin [5] to a general method of dealing with these equations.

The method is based upon converting a system of language equations to a certain structure representing all its solutions. This structure is a nondeterministic tree automaton, operating on an unlabelled infinite tree without any acceptance conditions. Furthermore, the state in each successor of a node is determined independently of the choice of the states in its siblings. Such a tree automaton is known as an *independent looping tree automaton* (ILTA), and it is induced by an NFA  $(Q, I, \delta, F)$  with an input alphabet  $A$  as follows. A *run* of the ILTA is any function  $r: A^* \rightarrow Q$  with  $r(\varepsilon) \in I$  and  $(r(w), a, r(wa)) \in \delta$  for all  $w \in A^*$  and  $a \in A$ . Every run defines a language  $L(r) = \{w \in A^* \mid r(w) \in F\}$ , and the ILTA defines the set of languages corresponding to all valid runs. Using multiple sets of accepting states  $F_1, \dots, F_n$  allows defining a set of  $n$ -tuples of languages.

**Theorem 3.1** (Baader, Okhotin [5]). *For every system of language equations using one-sided concatenation and Boolean operations, one can construct in exponential time an ILTA representing the set of solutions of this system.*

Using this representation, the basic questions about the set of solutions can be answered by simple algorithms analyzing the graph structure of the ILTA. This, in particular, yields exponential-time algorithms for testing whether an equation has (a) any solutions, (b) a unique solution, (c) finitely many solutions, (d) countably many solutions, (e) least or greatest solutions. All these problems are EXPTIME-complete [5]. Finite automata for unique, least and greatest solutions can be constructed as well.

### 3.4 Inequalities with union and one-sided concatenation

Consider systems of inequalities with one-sided concatenation, where the only allowed Boolean operation is union, that is, where inequalities are of the form

$$K_0 \cup X_1 K_1 \cup \dots \cup X_n K_n \subseteq L_0 \cup X_1 L_1 \cup \dots \cup X_n L_n .$$

Such a system always possesses a greatest solution, and this solution is regular, as long as all constant languages  $L_0, L_1, \dots, L_n$  in the right-hand sides are regular, and with *no restrictions* on the constants  $K_0, K_1, \dots, K_n$  (Kunc [44]). Moreover, for any given right-hand sides, there are only finitely many candidate languages, which may occur as components of greatest solutions of such systems for arbitrary constants  $K_1, \dots, K_n$ . These languages are regular and can be algorithmically calculated.

Once the list of candidate languages is compiled, there are finitely many possible greatest solutions to be checked. In order to check each of them, one should have an

effective procedure for testing the containment of the languages on the left-hand sides in regular languages. Hence, for instance, the greatest solution can be effectively computed for context-free constants on the left-hand sides.

## 4 Resolved systems of equations

This section is about systems of equations of the form  $X_i = \varphi_i(X_1, \dots, X_n)$  with  $i \in \{1, \dots, n\}$ , resolved with respect to their variables. These systems are known in the literature as *resolved* or *explicit*, and their most important quality is that they represent *inductive definition of languages*, in the sense that the properties of longer strings are expressed as Boolean combinations of concatenations of shorter strings. If the only allowed Boolean operation is union, these are the systems defining the semantics of the context-free grammars, and some other sets of operations yield their natural variants and generalizations.

**Lemma 4.1.** *If  $\varphi_1, \dots, \varphi_n$  are monotone and continuous, then the least solution of a system  $X_i = \varphi_i(X_1, \dots, X_n)$  with  $i \in \{1, \dots, n\}$  is the  $n$ -tuple  $L = \lim \varphi^k(\perp)$ , where  $\varphi = (\varphi_1, \dots, \varphi_n)$  is the right-hand side as an operator on  $(2^{A^*})^n$ , and  $\perp = (\emptyset, \dots, \emptyset)$ . The greatest solution is similarly obtained as  $\lim \varphi^k(\top)$ , where  $\top = (A^*, \dots, A^*)$ .*

*Sketch of a proof.* This well-known argument can be regarded as folklore. First, the sequence  $\{\varphi^k(\perp)\}_{k=0}^\infty$  monotonically increases. Then  $L$  is a solution of the system, because

$$\varphi(L) = \varphi(\lim \varphi^k(\perp)) = \lim \varphi(\varphi^k(\perp)) = \lim \varphi^{k+1}(\perp) = L,$$

which essentially uses the continuity of  $\varphi$ . In order to show that this solution is the least one, consider any other  $n$ -tuple  $K$  with  $\varphi(K) = K$ . Then each element of the sequence  $\{\varphi^k(K)\}_{k=0}^\infty$  is a superset of the corresponding element of  $\{\varphi^k(\perp)\}_{k=0}^\infty$ , and this inequality is extended to the least upper bounds of the sequences.  $\square$

Another important folklore result is the following sufficient condition of solution uniqueness. Let a system  $X_i = \varphi_i(X_1, \dots, X_n)$  be called *strict*, if for every two  $n$ -tuples of languages  $K, L$  that are equal modulo  $A^{\leq \ell}$ , the languages  $\varphi_i(K)$  and  $\varphi_i(L)$  are equal modulo  $A^{\leq \ell+1}$ .

**Lemma 4.2.** *Every strict system has at most one solution.*

In other words,  $\varphi$ , as a function on the metric space of  $n$ -tuples of languages defined in Section 2, is a *contraction*, and the fact that every contraction has at most one fixed point is a known result of basic analysis.

The most typical case of strict systems are those with each  $\varphi_i$  being a Boolean combination of concatenations, where every concatenation either contains a constant language without  $\varepsilon$ , or is a single constant language. The proof of Lemma 4.2 for such systems was provided by Autebert et al. [2].

## 4.1 Union and concatenation: context-free languages

Resolved systems of the most well-known kind are limited to the operations of union and concatenation. These equations constitute one of the two definitions of the semantics of the context-free grammars.

A *context-free grammar* is a quadruple  $G = (A, N, R, S)$ , in which  $A$  is a finite alphabet of *terminal symbols*,  $N$  is a finite set of *variables*, also known as *nonterminal symbols*, disjoint with  $A$ ;  $R$  is a finite set of *rules*, each of the form  $X \rightarrow \xi$  with  $X \in N$  and  $\xi \in (A \cup N)^*$ ;  $S \in N$  is a variable designated as the *start symbol*.

According to Ginsburg and Rice [24], such a grammar is interpreted as a system of language equations with the following equation for each variable  $X \in N$ :

$$X = \bigcup_{X \rightarrow \xi \in R} \xi,$$

where each terminal symbol  $a \in A$  in each concatenation  $\xi$  represents a constant language  $\{a\}$ , while  $\xi = \varepsilon$  is transcribed as a constant  $\{\varepsilon\}$ . By Lemma 4.1, this system always has a least solution, and the value of each variable  $X$  in this least solution is taken as *the language generated by  $X$* , denoted by  $L_G(X)$ . The language generated by the grammar is  $L(G) = L_G(S)$ .

Another definition of the semantics of the context-free grammars was given by Chomsky [15], who considered a rewriting system operating on strings over the alphabet  $A \cup N$ . Each rule  $X \rightarrow \xi$  can be used to rewrite a symbol  $X$  by a substring  $\xi$ . Formally, define a binary relation  $\Longrightarrow$  of one-step derivability on  $(A \cup N)^*$ , by  $\eta X \theta \Longrightarrow \eta \xi \theta$  for all  $\eta, \theta \in (A \cup N)^*$  and  $X \rightarrow \xi \in R$ . Its reflexive and transitive closure  $\Longrightarrow^*$  represents derivability in zero or more steps, and it is used to define the languages  $L_G(\alpha) = \{w \in A^* \mid \alpha \Longrightarrow^* w\}$ , for  $\alpha \in (A \cup N)^*$ , and  $L(G) = L_G(S)$ .

These two definitions are known to be equivalent. Consider the following basic examples of context-free grammars, each presented in the form of language equations:

**Example 4.1.** The equation  $S = aSb \cup \varepsilon$  has the unique solution  $L = \{a^n b^n \mid n \geq 0\}$ . It is the limit of the sequence  $\emptyset, \{\varepsilon\}, \{\varepsilon, ab\}, \{\varepsilon, ab, aabb\}$ , etc. (as in Lemma 4.1).

**Example 4.2.** The least solution of the equation  $S = SS \cup aSb \cup \varepsilon$  is the language of well-nested brackets, known as the Dyck language. Its greatest solution is  $S = A^*$ .

Because of their practical importance, context-free grammars became one of the most widely studied formalisms in the theory of computation. For an account of their properties, the reader is directed to a survey by Autebert et al. [2], and also to a survey of formal grammars by Okhotin [70].

As far as language equations are concerned, it is worth mentioning that every context-free grammar that does not generate  $\varepsilon$  can be transformed to an equivalent grammar in *Chomsky normal form*, with all rules of the form  $X \rightarrow YZ$  with  $Y, Z \in N$ , or  $X \rightarrow a$  with  $a \in A$ . The system of language equations corresponding to such a grammar always has a unique solution in  $\varepsilon$ -free languages. Another important form is *Greibach normal form* with rules  $X \rightarrow a\xi$  with  $\xi \in (A \cup N)^*$  or  $X \rightarrow \varepsilon$ ; the corresponding system of language equations is strict, and hence has a unique solution.

Context-free languages over a one-letter alphabet are known to be regular [24].

Some properties of the context-free languages are already undecidable, and they are important as the germs of the undecidability found in more general language equations. In particular, it is undecidable whether a given context-free grammar generates  $A^*$ , and this directly implies that testing whether a given resolved system with union and concatenation has a unique solution is undecidable; in contrast, testing the existence of a solution is trivially decidable (every system has one). A general undecidability technique for context-free languages was introduced by Hartmanis [26].

A generalization of context-free grammars to their *probabilistic*, *stochastic* or *fuzzy* variants is important in some applications, where the most probable parse of a word is being sought. These grammars can be defined by equations over a generalization of languages to mappings from  $A^*$  to a semiring, known as *formal power series* in non-commuting variables. Ordinary formal languages are then regarded as mappings from  $A^*$  to the Boolean semiring. For an introduction to the language-theoretic treatment of formal power series, the reader is referred to Sakarovitch [78]<sup>4</sup>.

## 4.2 Union, intersection and concatenation: conjunctive languages

The next kind of language equations is a direct generalization of the previous one, featuring an extra operation of intersection. Systems of such equations always have a least and a greatest solution due to Lemma 4.1. These systems can be used to define formal languages similarly to the context-free grammars, and hence are naturally regarded as another, more general family of formal grammars.

*Conjunctive grammars* were introduced by Okhotin [58] as a generalization of the context-free grammars with an explicit conjunction operation in the rules. A conjunctive grammar is again a quadruple  $G = (A, N, R, S)$ , in which  $A$ ,  $N$  and  $S$  are as in the context-free case, while the rules in  $R$  are of the form

$$X \rightarrow \xi_1 \& \dots \& \xi_m \quad (\text{with } X \in N, m \geq 1 \text{ and } \xi_1, \dots, \xi_m \in (A \cup N)^*),$$

and such a rule informally means that every string generated by each of the conjuncts  $\xi_i$  is therefore generated by  $X$ . If  $\xi_1, \dots, \xi_m \in A^* N A^* \cup A^*$  in every rule, such a grammar is called *linear conjunctive*.

The system of language equations associated to a grammar has equations of the form

$$X = \bigcup_{X \rightarrow \xi_1 \& \dots \& \xi_m \in R} \bigcap_{i=1}^m \xi_i \quad (\text{for all } X \in N).$$

Its least solution defines the languages  $L_G(X)$  for  $X \in N$ , and  $L(G) = L_G(S)$ .

An equivalent definition of the semantics of conjunctive grammars is given by *term rewriting*, which generalizes the string rewriting used for context-free grammars. Given a conjunctive grammar  $G$ , consider terms with concatenation and conjunction as operation symbols and with symbols from  $A \cup N$  as atomic terms. A term  $\psi$  is derivable in one step from a term  $\varphi$ , written as  $\varphi \Longrightarrow \psi$ , if  $\psi$  is obtained from  $\varphi$

- either by rewriting a subterm  $X \in N$  with  $(\xi_1 \& \dots \& \xi_m)$ , using a rule  $X \rightarrow$

<sup>4</sup>\*\*\*cross-ref\*\*\*: maybe something is in Chapter 2 by Sakarovitch? also, chapters 4 and 5 are about the closely related probabilistic automata

$\xi_1 \& \dots \& \xi_m$  from  $R$ ,

- or by rewriting, for  $w \in A^*$ , a subterm  $(w \& \dots \& w)$  with a single string  $w$ .

The language generated by a term  $\varphi$  is  $L_G(\varphi) = \{w \in A^* \mid \varphi \Longrightarrow^* w\}$ . The language generated by the grammar is  $L(G) = L_G(S) = \{w \in A^* \mid S \Longrightarrow^* w\}$ .

The two definitions of the semantics are again equivalent [59]. Since intersection is explicit in these equations, any finite intersection of context-free languages, such as  $\{a^n b^n c^n \mid n \geq 0\}$ , can be represented by a conjunctive grammar. However, conjunctive grammars can represent many languages outside of this intersection closure, such as the language  $\{w c w \mid w \in \{a, b\}^*\}$  [82]:

**Example 4.3** (Okhotin [58]). The system of language equations

$$\begin{aligned} S &= U \cap T & X &= aXa \cup aXb \cup bXa \cup bXb \cup cWa \\ U &= aUa \cup aUb \cup bUa \cup bUb \cup c & Y &= aYa \cup aYb \cup bYa \cup bYb \cup cWb \\ T &= (aX \cap aT) \cup (bY \cap bT) \cup cW & W &= aW \cup bW \cup \varepsilon \end{aligned}$$

has a unique solution with  $S = \{w c w \mid w \in \{a, b\}^*\}$  and  $T = \{u c z u \mid u, z \in \{a, b\}^*\}$ . The equation for  $T$  matches a single symbol in the left part of a string  $u c v$  to the corresponding symbol in its right part using  $X$  or  $Y$ , and the recursive reference to  $aT$  or  $bT$  makes the remaining symbols be compared in the same way. The intersection with the language  $U = \{u c v \mid u, v \in \{a, b\}^*, |u| = |v|\}$  completes the definition of the language.

Three normal forms for conjunctive grammars are known. One of them is a direct generalization of the Chomsky normal form [58], with the rules of the form  $X \rightarrow Y_1 Z_1 \& \dots \& Y_m Z_m$  with  $m \geq 1$  and  $Y_i, Z_i \in N$ , or  $X \rightarrow a$  with  $a \in A$ . Another normal form [71] has all rules of the form  $X \rightarrow Y_1 a_1 Z_1 \& \dots \& Y_m a_m Z_m$  with  $m \geq 1$ ,  $Y_i, Z_i \in N$  and  $a_i \in A$ , or  $X \rightarrow a$  with  $a \in A$ , or  $S \rightarrow aX$  or  $S \rightarrow Xa$  with  $a \in A$  and  $X \in N$ ; the system of equations corresponding to such a grammar is strict. One more normal form theorem [71] converts an arbitrary conjunctive grammar to one with the corresponding system of language equations of the form  $X_i = F_i \cup \bigcap_{j=1}^{m_i} \xi_{i,j}$ , where  $F_i \subset A^*$  is a finite set,  $m_i \geq 1$  and  $\xi_{i,j} \in (A \cup N)^*$ , that is, with union restricted to union with a singleton constant.

The known bounds on the time complexity of conjunctive languages are the same as in the context-free case: there is a straightforward algorithm working in time  $O(n^3)$  [61], which can be accelerated to  $O(n^\omega)$  with  $\omega < 3$  by offloading some computations to a matrix multiplication procedure [69]. These algorithms actually apply to the larger family of *Boolean grammars* [61], which allows an explicit negation in the rules. The most general definition of Boolean grammars, given by Kountouriotis et al. [40], involves a generalization of languages to mappings  $L: A^* \rightarrow \{0, \frac{1}{2}, 1\}$ , where  $\frac{1}{2}$  represents uncertainty, and all operations are redefined according to the three-valued logic. A further generalization to a semiring-valued formalism was devised by Ésik and Kuich [22]. The properties of Boolean grammars are beyond the scope of this chapter, and an interested reader is directed to a recent survey [70].

*Conjunctive grammars over a unary alphabet* form a special area of study. Such grammars are completely irrelevant to the parsing applications, but on the other hand they are important as a nontrivial special case of language equations, and form the basis for the study of more general language equations over a unary alphabet described in Section 6.4.

The following grammar for  $\{a^{4^n} \mid n \geq 0\}$  became the first evidence of their non-triviality:

**Example 4.4** (Jež [29]). The least solution of the system

$$\begin{aligned} X_1 &= (X_1X_3 \cap X_2X_2) \cup a & X_3 &= (X_1X_2 \cap X_6X_6) \cup aaa \\ X_2 &= (X_1X_1 \cap X_2X_6) \cup aa & X_6 &= (X_1X_2 \cap X_3X_3) \end{aligned}$$

is  $X_i = \{a^{i \cdot 4^n} \mid n \geq 0\}$  for  $i \in \{1, 2, 3, 6\}$ .

This example is best explained in terms of the base-4 representation of the lengths of the strings. Then each variable  $X_i$  represents all base-4 numbers  $i0 \dots 0$ . Substituting these four languages into the first equation, the first concatenation  $X_1X_3$  produces all numbers with the notation  $10^*30^*$ ,  $30^*10^*$  and  $10^+$ . The second concatenation  $X_2X_2$  yields  $20^*20^*$  and  $10^+$ . The intersection of these two concatenations is the set of strings with base-4 length  $10^+$ ; in other words, both concatenations contain some garbage, yet the garbage in the concatenations is disjoint, and is accordingly filtered out by the intersection. Finally, the union with  $\{a\}$  yields the language  $\{a^{4^n} \mid n \geq 0\}$ , and thus the first equation turns into an equality. The rest of the equations are verified similarly.

The idea of manipulating the positional notation of numbers was extended to the following general result:

**Theorem 4.3** (Jež, Okhotin [30]). *Let  $A_k = \{0, 1, \dots, k-1\}$  with  $k \geq 2$  be an alphabet of  $k$ -ary digits, and let  $L \subseteq A_k^*$  be a language generated by a linear conjunctive grammar, which contains no words beginning with 0. Then there exists and can be effectively constructed a conjunctive grammar generating  $\{a^n \mid \text{the } k\text{-ary representation of } n \text{ is in } L\}$ .*

A similar technique was used to construct an EXPTIME-complete set of numbers with its unary representation generated by a conjunctive grammar [31].

Theorem 4.3 is used to establish undecidability results for conjunctive grammars over  $\{a\}$ . Consider the language of valid accepting computations of a Turing machine  $\mathcal{T}$ ,  $\text{VALC}(\mathcal{T})$ , described in more detail in Section 6.2. It is known to be linear conjunctive. Let it be defined over some  $k$ -letter alphabet. Then the symbols of this alphabet can be re-interpreted as digits in  $A_k$ , which leads to a conjunctive grammar for the language of unary representations of the numbers, whose  $k$ -ary representations are in  $\text{VALC}(\mathcal{T})$ . Since  $\text{VALC}(\mathcal{T})$  is empty if and only if  $L(\mathcal{T})$  is empty, the emptiness problem for conjunctive grammars is undecidable. The following stronger result holds:

**Theorem 4.4** (Jež, Okhotin [30]). *For every fixed conjunctive language  $L_0 \subseteq a^*$ , the problem of whether a given conjunctive grammar over  $\{a\}$  generates  $L_0$  is  $\Pi_1^0$ -complete.*

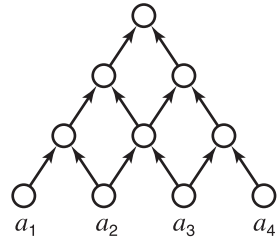
Consider conjunctive grammars over a unary alphabet with a single variable. The first example of the nontriviality of such grammars was given by Okhotin and Rondogiannis [72], by encoding the four languages  $X_i$  with  $i \in \{1, 2, 3, 6\}$  in Example 4.4 within a single language  $\bigcup_{i \in \{1, 2, 3, 6\}} \{a^{np-d_i} \mid a^n \in X_i\}$ , for some  $p, d_1, d_2, d_3, d_6 \geq 1$ . This example was extended by Jež and Okhotin [34] to a general technique of encoding any unary conjunctive language in a solution of a univariate language equation  $X = \varphi(X)$ , leading to the following undecidability results: testing whether a given conjunctive grammar over

a unary alphabet and with a single variable generates a finite language is  $\Sigma_1^0$ -complete, testing co-finiteness is  $\Sigma_1^0$ -complete as well, and testing equivalence of two given grammars is  $\Pi_1^0$ -complete.

### 4.3 Union, intersection and linear concatenation: trellis automata

Resolved systems of language equations with union, intersection and linear concatenation correspond to linear conjunctive grammars (Okhotin [62]). For instance, the grammars in the above Examples 4.1 and 4.3 are linear. The grammar for the Dyck language given in Example 4.2 is not linear, but a linear conjunctive grammar for this language can be obtained [62, Ex. 2] using the method of Dyer [20].

An important property of this family of language equations is their computational equivalence to the simplest type of cellular automata, known as *one-way real-time cellular automata*, or *trellis automata*, studied by Dyer [20], Čulík et al. [18], Ibarra and Kim [27], and others. A trellis automaton [18, 62], defined as a quadruple  $(Q, I, \delta, F)$ , processes an input string  $a_1 \dots a_m$  of length  $m \geq 1$  using a uniform array of  $\frac{m(m+1)}{2}$  nodes, as presented in the figure. Each node



computes a value from the finite set of states  $Q$ . The nodes in the bottom row obtain their values directly from the input symbols using a function  $I: A \rightarrow Q$ . The value of all other nodes is computed from the values of their predecessors using the function  $\delta: Q \times Q \rightarrow Q$ . The string is accepted if and only if the value computed by the topmost node belongs to the set of accepting states  $F \subseteq Q$ .

**Theorem 4.5** (Okhotin [62, 60]). *A language  $L \subseteq A^+$  is recognized by a trellis automaton if and only if it is generated by a linear conjunctive grammar. Every such language can be represented by a linear conjunctive grammar with two variables, and with the equations  $S = \varphi(X)$ ,  $X = \psi(X)$ .*

A general method for proving non-representability of languages by trellis automata was discovered by Terrier [80], and it is based on counting the number of cases that the trellis automaton needs to distinguish in the last few levels of its computation. Let  $L \subseteq A^*$  be a language, let  $k \geq 1$ , and consider the last  $k$  levels of a possible computation of a trellis automaton on a string  $w = a_1 \dots a_m$  with  $m \geq k$ . In these  $k$  levels, an automaton would have to decide on the membership of  $\frac{k(k+1)}{2}$  subwords of  $w$  in  $L$ , as represented in the following set:

$$S_{L,k,w} = \{(i, j) \mid i, j \geq 0, i + j < k, a_{i+1} \dots a_{m-j} \in L\}.$$

Hence, for each  $k$ , a trellis automaton recognizing  $L$  must distinguish between different sets  $S_{L,k,w}$  that occur for different strings  $w \in A^*$ , that is, between as many as

$$f_L(k) = |\{S_{L,k,w} \mid w \in A^{\geq k}\}|$$

cases, and must do so on the basis of  $k$  states in the  $k$ -th last line.

**Theorem 4.6** (Terrier [80]). *If  $L$  is linear conjunctive, then  $f_L(k) \leq p^k$  for some  $p \geq 2$ .*

In particular, Terrier [80] used this result to show that the square of a certain linear context-free language is not recognized by any trellis automaton, and hence this language family is not closed under concatenation. Another interesting result concerning this family is the existence of a trellis automaton for  $\{a^m b^{m+n} a^n \mid m, n \geq 1\}$ , demonstrated by Čulík [17] by simulating a cellular automaton solving the well-known *firing squad synchronization problem*. The existence of P-complete languages recognized by trellis automata was discovered by Ibarra and Kim [27]. All the above results apply to solutions of language equations via Theorem 4.5.

## 4.4 Complementation and concatenation

Turning to systems of equations of the form  $X_i = \varphi_i(X_1, \dots, X_n)$  with the operations of concatenation and complementation, the first thing to note is that they are not subject to Lemma 4.1, as complementation is not monotone. Such systems need not have a solution, as demonstrated by an equation  $X = \overline{X}$ . However, this possibility of expressing a contradiction works exclusively modulo the empty string. Once a system has a solution modulo  $\{\varepsilon\}$ , it can be extended to a solution:

**Lemma 4.7** (Okhotin, Yakimova [73]). *If a system  $X_i = \varphi_i(X_1, \dots, X_n)$  ( $1 \leq i \leq n$ ) with concatenation and complementation and with any constant languages has a solution  $(L_1, \dots, L_n)$  modulo  $A^{\leq \ell}$  for some  $\ell \geq 0$ , then the system has a solution  $(\widehat{L}_1, \dots, \widehat{L}_n)$ , which is equal to  $(L_1, \dots, L_n)$  modulo  $A^{\leq \ell}$ .*

The following is the first example of nontriviality in these equations (and actually the first known language equation over a unary alphabet with a non-regular unique solution):

**Example 4.5** (Leiss [49]). The strict language equation  $X = a \cdot \overline{\overline{\overline{X^2}}}$  has the unique solution  $\{a^n \mid \exists i \geq 0 : 2^{3i} \leq n < 2^{3i+2}\}$ .

**Example 4.6** (Okhotin, Yakimova [74]). The strict equation  $X = \overline{a\overline{X}b}$  over an alphabet  $A = \{a, b\}$  has a non-regular unique solution  $\{a^n w b^n \mid w = \varepsilon \text{ or } w \in bA^*\}$ .

There exists a method of proving non-representability of languages by such equations, based upon the notion of a prime language (a language  $L$  is called *prime*, if  $L = MN$  implies  $M = \{\varepsilon\}$  or  $N = \{\varepsilon\}$ ).

**Lemma 4.8** (Okhotin, Yakimova [74]). *Let  $L \subseteq A^*$  and its complement  $\overline{L}$  be prime languages. Then, if a system of language equations  $X_i = \varphi_i(X_1, \dots, X_n)$  with concatenation and complementation has a unique solution with  $L$  or  $\overline{L}$  among its components, one of these languages must be among the constant languages used in the system.*

For example, the regular language  $L = aA^*b \cup bA^*a \cup \varepsilon \subseteq \{a, b\}^*$  and its complement  $\overline{L} = aA^*a \cup bA^*b \cup \{a, b\}$  are both prime, and thus the lemma asserts that neither of them is representable by these equations with finite constants. Similarly, the non-regular

language  $L' = (aA^*b \cup bA^*a \cup \varepsilon) \setminus \{a^n b^n \mid n > 1\}$  and its complement  $\overline{L'}$  are prime, and therefore non-representable by equations with concatenation and complementation and regular constants.

Turning to the basic decision problems for this class of language equations, *solution existence* can be checked modulo  $\{\varepsilon\}$  according to Lemma 4.7, and it is NP-complete for large classes of constants [73]. For the *solution uniqueness*, it is not known whether the problem is decidable; it is only known to be co-r.e. and PSPACE-hard. In the special case of a one-letter alphabet, testing solution uniqueness is complete for the complexity class US (which stands for *unique satisfiability*) studied by Blass and Gurevich [9].

## 4.5 Concatenation and various sets of Boolean operations

Consider resolved systems of language equations with concatenation and all Boolean operations. These equations are powerful enough to simulate equations of the general form  $\varphi = \psi$ . Indeed, such an equation is equivalent to  $X = \overline{X} \cap (\varphi \Delta \psi)$ , where  $X$  is a new variable and  $\Delta$  denotes symmetric difference.

A systematic study of the families of languages represented by resolved systems  $X_i = \varphi_i(X_1, \dots, X_n)$  with concatenation, singleton constants and any possible fixed set of Boolean operations was done by Okhotin [67]. The study is based upon Post's classification of all functions of Boolean logic. For language equations, there are exactly seven distinct classes of languages represented by unique solutions of equations over different bases of Boolean functions. First, there are three trivial subregular classes generated by the sets of operations  $\emptyset$ ,  $\{\top\}$  and  $\{\cap, \top\}$ , where the logical truth  $\top$  represents the language  $A^*$ . Next come the context-free languages with  $\{\cup\}$ , the conjunctive languages with  $\{\cup, \cap\}$ , and the class generated by complementation only. Finally, if all Boolean operations are used, then these equations define exactly the recursive languages by their unique solutions; furthermore, rather than all Boolean operations, it is sufficient to use any of the three functions  $f_1(K, L, M) = K \cup (L \cap \overline{M})$ ,  $f_2(K, L, M) = K \cap (L \cup \overline{M})$  and  $f_3(K, L, M) = K \Delta L \Delta M$ . The properties of these equations are considered in Section 6.

## 5 Equations with constant sides

This section is about systems of equations of the form  $\varphi_i(X_1, \dots, X_n) = C_i$ , sometimes referred to as *implicit equations*. The most well-studied case are equations with only rational operations, which can be analyzed within the syntactic monoid of  $C_i$ . This technique, along with its implications for computability of solutions and decidability of their properties, is described in the first subsection. The next subsection surveys some precise results on the complexity of solvability, while equations with other sets of operations are considered in the last Subsection 5.3.

## 5.1 Regularity of maximal solutions

Consider equations  $\varphi_i(X_1, \dots, X_n) = C_i$ , with regular constants  $C_i$  and with  $\varphi_i$  using union, concatenation, Kleene star and regular constants. Since  $\varphi_i$  is monotone, every maximal solution of such a system is also a maximal solution of the system of inequalities  $\varphi_i(X_1, \dots, X_n) \subseteq C_i$ . It shall be proved that the latter system has finitely many maximal solutions, which are regular and can be algorithmically computed, and among them one can find the solutions of the original system. Such a result was first formulated by Conway [16]. It is based on the fact that all maximal solutions of the system can be calculated within any monoid recognizing all constant languages on the right-hand sides. This idea can be used to prove regularity of maximal solutions also when the left-hand sides employ more general operations, such as infinitary union, and arbitrary constants. All assertions of the following theorem also hold if any inequalities are replaced by equations.

**Theorem 5.1.** *Let  $\bigcup_{j \in I_i} E_{ij} \subseteq C_i$ ,  $i = 1, \dots, m$ , be a system of language inequalities, where  $C_i$  are constant regular languages,  $I_i$  are (possibly infinite) index sets and expressions  $E_{ij}(X_1, \dots, X_n)$  are products of arbitrary constant languages and variables. Then the system has only finitely many maximal solutions and every maximal solution has all components regular. If all left-hand sides in the system are regular expressions, then all maximal solutions can be algorithmically computed.*

*Proof.* The idea of the proof is to consider some fixed congruence  $\sim$  of  $A^*$  of finite index, under which all languages  $C_i$  are closed, and to show that every solution can be extended to a solution whose every component is a union of  $\sim$ -classes. If  $(K_1, \dots, K_n)$  is an arbitrary solution, the new solution  $(L_1, \dots, L_n)$  is obtained as the closure of the original solution under  $\sim$ , that is,  $L_j = \{w \in A^* \mid \exists v \in K_j: v \sim w\}$ . To see that  $(L_1, \dots, L_n)$  is a solution of the system, consider any  $w \in E_{ij}(L_1, \dots, L_n)$ . Then there exists  $v \in E_{ij}(K_1, \dots, K_n)$  satisfying  $v \sim w$ . Since  $(K_1, \dots, K_n)$  is a solution,  $v$  belongs to  $C_i$ . This shows that  $w$  belongs to  $C_i$ , because  $C_i$  is closed under  $\sim$ .

Now, once it is known that every solution is contained in some  $\sim$ -closed solution, all statements of the theorem can be easily verified; for instance, in order to find all maximal solutions, it is sufficient to test finitely many candidates for being a solution.  $\square$

Note that the state of the minimal DFA of a language  $L$  reached by a word  $w \in A^*$  can be identified with the greatest solution of the language inequality  $wX \subseteq L$  with constant right-hand side<sup>5</sup>. Similarly, maximal solutions of the inequality  $XY \subseteq L$  in two variables  $X$  and  $Y$  are exactly states of the so-called universal (non-deterministic) automaton of the language  $L$  [78]<sup>6</sup>. Therefore, the existence of a finite universal automaton of every regular language can be viewed as a special case of the above theorem.

## 5.2 Complexity of decision problems

The method used to prove Theorem 5.1 was employed by Bala [6] to analyze the computational complexity of solvability of systems of equations or inequalities with constant

<sup>5</sup>\*\*\*refer to the appropriate chapter

<sup>6</sup>\*\*\*refer to the appropriate chapter, if universal automaton is defined elsewhere, otherwise omit

right-hand sides. These constants are given by nondeterministic automata, and the left-hand sides use the operations of union, concatenation and star.

In order to decide the existence of a non-empty solution of a system of inequalities, due to the monotonicity of operations, only singleton solutions  $(\{w_1\}, \dots, \{w_n\})$  need to be considered. According to the proof of Theorem 5.1, such a solution can be considered as an  $n$ -tuple of classes of the congruence, to which the words  $w_j$  belong. The algorithm [6] guesses an  $n$ -tuple of such congruence classes, represents these classes by relations on the states of NFAs defining the constants  $C_i$ , and checks them for being a solution using a polynomial-space algorithm for testing containment of NFAs.

In order to decide solvability for equations, it is sufficient to look for maximal solutions. According to the proof of Theorem 5.1, every component of a maximal solution is a union of classes of the congruence. Therefore, the existence of a solution can be decided in exponential space by calculating such a congruence, non-deterministically choosing some of its classes for every component of the solution, and verifying the equality of the resulting regular languages. On the other hand, the EXPSPACE-complete problem of universality of rational expressions with intersection, where intersections are not nested, can be expressed as solvability of systems of equations, so the solvability problem is EXPSPACE-complete [6]. Additionally, using a known polynomial-space algorithm for the limitedness problem of *desert automata* due to Bala [6] and Kirsten [39], one can test whether a solution of a system can be replaced by a finite one, and so the problem of existence of finite solutions turns out to be EXPSPACE-complete as well [6].

For the particular equation  $XY = C$ , where  $C$  is given by a DFA, PSPACE-completeness of testing whether it has any nontrivial solution has recently been proved by Martens et al. [54].

### 5.3 Generalizations to other operations

The above results on systems  $\varphi_i(X_1, \dots, X_n) = C_i$  with union and concatenation essentially depend on having union as the only Boolean operation. If, for instance, the symmetric difference  $\Delta$  is allowed, then an arbitrary equation  $\varphi = \psi$  can be expressed as  $\varphi \Delta \psi = \emptyset$ , and hence such equations fall under the more general case described in the next section. The same happens if both union and intersection are allowed.

Consider a variant of these equations, in which concatenation is replaced by a different word operation  $\diamond: A^* \times A^* \rightarrow 2^{A^*}$ . Then one can define another word operation  $\square$  by the rule  $w \in u \square v \iff u \in w \diamond v$ , for all words  $u, v$  and  $w$ . For example, if  $\diamond$  is the concatenation, then  $\square$  is the quotient. As proved by L. Kari [38], the greatest solution of an inequality  $X \diamond L \subseteq C$  is  $X = \overline{C} \square L$ . The solution existence problem for equations of this special form, with regular or context-free constants  $C$  and  $L$  and with different pairs of operations  $(\diamond, \square)$ , were studied by L. Kari [38] and Domaratzki and K. Salomaa [19]; most of the operations are derived from the shuffle along trajectories [56]. In the case of shuffle along letter-bounded regular sets of trajectories, a result analogous to Theorem 5.1 was proved [19], which allowed dealing with equations of the form  $X \diamond Y = C$ , that is, with the problem of existence of a decomposition of a given regular language. This decomposition problem was particularly studied for the ordinary shuffle operation, where only very little is known [13, 8].

## 6 Equations of the general form

Consider systems of equations of the general form  $\varphi_i(X_1, \dots, X_n) = \psi_i(X_1, \dots, X_n)$ , with different allowed operations. It is assumed that all operations are continuous (the case of non-continuous operations is considered in Section 7), and furthermore, computable in the sense defined in Subsection 6.1. Typically, these will be concatenation and Boolean operations. In almost all cases, these equations are computationally universal.

### 6.1 Upper bounds

Let  $\varphi: (2^{A^*})^n \rightarrow 2^{A^*}$  be a continuous operation on languages, that is, for every  $\ell \geq 0$  there exists  $m = m(\ell) \geq 0$ , such that  $K = L \pmod{A^{\leq m}}$  implies  $\varphi(K) = \varphi(L) \pmod{A^{\leq \ell}}$ . Further assume that this definition is algorithmically effective, that is, that  $m(\ell)$  can be effectively computed, and that  $\varphi(L) \cap A^{\leq \ell}$  can be computed from  $L \cap A^{\leq m}$  and  $\ell$ . Such an operation is called *computable*, in accordance with a more general definition given in the theory of computable analysis on metric spaces [81]. All standard continuous operations on languages, such as concatenation, Boolean operations, Kleene star, shuffle, etc., are computable.

Consider a system of language equations with continuous and computable sides. Then the characterizations of solution existence and solution uniqueness given in Theorem 2.2 can be regarded as first-order formulae of the form  $(\forall \ell)R(\ell)$  and  $(\forall \ell)(\exists \ell')R_1(\ell, \ell')$ , respectively, where the predicates  $R$  and  $R_1$  are *recursive* due to the computability condition. This immediately implies that, as decision problems, solution existence and solution uniqueness are in the arithmetical hierarchy: in  $\Pi_1^0$  and in  $\Pi_2^0$ , respectively. Some further analysis leads to the following upper bounds on the sets representable as solutions:

**Theorem 6.1** (cf. Okhotin [68]). *If a system of language equations with continuous and computable operations and with any recursive constants has a unique (least, greatest) solution, then the components of this solution are recursive (r.e., co-r.e., respectively).*

The cited paper establishes the theorem for equations with Boolean operations and concatenation, but the proof can be extended to computable ultrametric spaces. It will now be demonstrated that these bounds are tight for all but the simplest language equations.

### 6.2 Computationally universal solutions

The first signs of computational universality in language equations were discovered by Parikh et al. [75], who used a concise *ad hoc* argument to prove that testing whether a language equation with Boolean operations and concatenation has any solutions is undecidable. Later systematic studies of language equations adopted a uniform method for encoding universal computation in formal languages, which is based upon the language of *computation histories* of a Turing machine  $\mathcal{T}$ , introduced by Hartmanis [26]. This language is denoted by

$$\text{VALC}(\mathcal{T}) = \{w \dagger C_{\mathcal{T}}(w) \mid w \in L(\mathcal{T})\},$$

where the string  $C_{\mathcal{T}}(w)$  over a second alphabet  $\Gamma$  somehow lists all consecutive configurations in the computation of  $\mathcal{T}$  on the input  $w$ , and the symbol  $\natural \notin A \cup \Gamma$  is used as a separator. For a suitable encoding  $C_{\mathcal{T}}$ , this language is an intersection of two context-free languages [26] (actually, these can be LL(1) linear context-free languages [66]), and hence can be defined by language equations with a unique solution  $X = \text{VALC}(\mathcal{T})$ . These equations use the operations of union, intersection and linear concatenation, though intersection can be eliminated [63]. Adding an extra equation  $X = \emptyset$ , the system has a solution if and only if  $L(\mathcal{T}) = \emptyset$ .

The same construction can be used to represent the language  $L(\mathcal{T})$  as a component of the least solution of the system. Once equations defining  $X = \text{VALC}(\mathcal{T})$  are constructed, it remains to “extract”  $L(\mathcal{T})$  out of  $X = \text{VALC}(\mathcal{T})$  using additional equations. Let  $Y$  be a new variable and consider the inequality

$$\text{VALC}(\mathcal{T}) \subseteq Y\natural\Gamma^*,$$

which can be formally rewritten as an equation  $X \cup Y\natural\Gamma^* = Y\natural\Gamma^*$ . This inequality states that for every  $w \in L(\mathcal{T})$ , the string  $w\natural C_{\mathcal{T}}(w)$  should be in  $Y\natural\Gamma^*$ , that is,  $w$  should be in  $Y$ . This makes  $L(\mathcal{T})$  the least solution of this inequality.

By a dual argument involving the complement of  $\text{VALC}(\mathcal{T})$ , the complement of an arbitrary r.e. set can be represented by a greatest solution of some system. These two constructions can then be combined to represent every recursive set by a unique solution.

**Theorem 6.2** (Okhotin [68, 63, 66]). *Every recursive (r.e., co-r.e.) set is representable as a component of a unique solution (least, greatest, respectively) of a system of language equations with union, linear concatenation and singleton constants. The same result holds for equations with intersection or symmetric difference instead of the union.*

Using the same construction based on  $\text{VALC}(\mathcal{T})$ , the undecidability level of the main decision problems can be precisely determined. Already for systems  $\varphi_i = \psi_i$  with union (or intersection), linear concatenation and singleton constants, (a) solution existence is co-r.e.-complete, (b) solution uniqueness is  $\Pi_2^0$ -complete, (c) existence of a least/greatest solution is  $\Pi_2^0$ -complete [68, 63], and (d) having finitely many solutions is  $\Sigma_3^0$ -complete [64]. All problems maintain the same complexity for equations with all Boolean operations, unrestricted concatenation and recursive constants.

### 6.3 Equations $XK = LX$ and related systems

The equations  $XL = LX$ , known as the *commutation equations*, were first considered by Conway [16]. Since then, the commutation equations, as well as the more general *conjugacy equation*  $XK = LX$ , have attracted attention as the simplest examples of language equations, which cannot be handled using the known methods. For any constant language  $L$ , the union of any solutions of these equations is a solution as well, and hence there is a greatest solution among them. It was anticipated that this greatest solution could be analyzed using the methods of combinatorics on words, which was achieved for some very special classes of constants: by Karhumäki et al. [36] for regular codes, and by Massazza and Salmela [55] for languages whose lexicographically minimal word is suf-

ficiently distinguished. In a related work, Frid [23] characterized all pairs of commuting languages over a binary alphabet, which are closed under taking subwords.

Yet in the end it turned out that the greatest solution of the equation  $XL = LX$  can encode universal computation already for a finite constant  $L$ :

**Theorem 6.3** (Kunc [45]). *There exists a finite language  $L$  over a binary alphabet, such that the greatest solution of the equation  $XL = LX$  is co-r.e.-complete.*

The proof is best explained using the following intuition on the commutation equation. This equation can be viewed as a game of two players, the attacker and the defender, in which the defender tries to prove that a word belongs to a solution of the equation. Therefore, a position of the game is an arbitrary word  $w \in A^*$ . In each round of the game, the attacker first chooses any word from  $L$  and adds it to one or the other side of  $w$ . In this way, the attacker decides, which of the two inclusions he wants to verify. Now the defender has to respond by removing some word belonging to  $L$  from the opposite side of the resulting word. The word thus obtained is a new position of the game. The attacker wins the game if the defender has no move available, and the defender wins if he manages to continue playing forever. Then  $w$  belongs to the greatest solution of  $XL = LX$  if and only if the defender has a winning strategy when the game begins with  $w$ , and all positions of the game that can be reached from  $w$  when the defender follows his winning strategy form a solution of the equation that contains  $w$ .

The proof of the theorem consists of two steps. At the first step, the complement of an arbitrary language recognized by a computationally universal machine is encoded into the greatest solution of a commutation equation with a regular constant  $L$ . The following example shows the principles of this construction by sketching an encoding of two counters and the operation of simultaneous decrementation of both counters, together with testing whether both counters are equal to zero. This example is already sufficient to prove that the greatest solution need not be regular.

**Example 6.1** (Kunc [45]). There exists such a regular language  $L$  over an alphabet  $A = \{a, b, e, \hat{e}, f, \hat{f}, g, \hat{g}\}$ , that the greatest solution of the equation  $XL = LX$  contains a string  $a^m b a^n$  with  $0 \leq m \leq n$  if and only if  $m = n$ .

This is a game, in which the defender has to prove that the values of two counters are equal, while the attacker tries to show that the value of the first counter is in fact smaller. If the values of the two counters are  $m, n \geq 0$ , this is represented by the word  $a^m b a^n$ . Stages of the computation are encoded using auxiliary letters  $e, \hat{e}, f, \hat{f}, g$  and  $\hat{g}$ , placed around the word representing counter values. The game is played on words of the form  $u a^m b a^n v$ , where  $u$  is a suffix of the word  $efg$ , and  $v$  is a prefix of the word  $\hat{g}\hat{f}\hat{e}$ . Furthermore, it is required that the words  $u$  and  $v$  together contain exactly 3 letters, that is, the only allowed pairs are  $(efg, \varepsilon)$ ,  $(g, \hat{g}\hat{f})$ ,  $(\varepsilon, \hat{g}\hat{f}\hat{e})$  and  $(fg, \hat{g})$ . Finally, words of the form  $efg b a^n$  are disallowed, since they represent the situation when the first counter already has value zero, and the attacker has been successful in proving that the value of the first counter is smaller. Let  $M$  denote the set of all these correct configurations.

A configuration can be modified by adding or removing one of the words from the set

$$N = \{ef, ga, e, fg, a\hat{g}, \hat{f}\hat{e}, \hat{g}\hat{f}, \hat{e}\}.$$

The attacker verifies the correctness of a current configuration by placing the letter  $c$  at either side of the word. If  $c$  is added to an incorrect configuration, then the defender is unable to remove a word belonging to  $L$  from the other side, and loses. All words obtained from a correct configuration by adding  $c$  are placed into the language  $L$ , so that the defender immediately wins, whenever the attacker plays  $c$  on a correct word. To ensure that the attacker does not produce incorrect configurations himself, all incorrect words, which can be possibly produced by the attacker (either containing two occurrences of  $b$  or with a wrong ordering of letters), are collected in a set  $L_0$ . Finally, the configuration  $f g b a \hat{g}$  is added to  $L$  to make it winning for the defender, since it can be reached only right before successfully decrementing both counters to zero. Altogether, the regular language  $L$  is defined as

$$L = N \cup \{c, f g b a \hat{g}\} \cup c M \cup M c \cup L_0.$$

If the game begins with the word  $e f g a^m b a^n$ , it has to proceed according to the scenario described below (possibly changing the direction of computation in any position). Whenever one of the players performs a different action, he immediately loses the game: the attacker can use the letter  $c$  to verify correctness of the configuration, while the defender can delete the whole incorrect word produced by the attacker, because it belongs to  $L_0$ .

The attacker begins by appending  $\hat{g} \hat{f}$  to produce the word  $e f g a^m b a^n \hat{g} \hat{f}$ , from which the defender has to remove  $e f$  to preserve the number of auxiliary letters. The resulting position of the game is  $g a^m b a^n \hat{g} \hat{f}$ . Then the attacker appends  $\hat{e}$ , and the defender removes  $g a$ , and so the next position is  $a^{m-1} b a^n \hat{g} \hat{f} \hat{e}$ , with the first counter decremented. By performing the symmetric moves, that is, adding  $f g$  and  $e$ , the attacker then forces the defender to decrement also the second counter, and the resulting position is  $e f g a^{m-1} b a^{n-1}$ .

If  $m < n$ , then a position of the form  $e f g b a^n$  will be eventually reached, and the attacker can win the game, because this word does not belong to  $M$ . If  $m = n$ , then the word  $f g b a \hat{g}$  will be obtained as the last position before both counters reach zero. Because this word belongs to  $L$ , it represents a winning position for the defender. Altogether, this means that the set of defender's winning positions cannot be regular due to the pumping lemma.

Jeandel and Ollinger [28] describe how this first step of the construction can be split into several more transparent constructions using certain special combinatorial games.

Once an infinite regular language  $L$ , for which the equation  $XL = LX$  has a co-r.e.-complete greatest solution, is constructed, the second step of the proof of Theorem 6.3 consists in encoding the finite automaton for  $L$  into a finite language. This is achieved by splitting the words from  $L$  into finitely many fragments annotated by special symbols, which describe the computation of this automaton. These fragments are incorporated into a finite language  $L'$ , along with auxiliary words, so that the attacker-defender game on  $L'$  ensures that the fragments are concatenated to a word accepted by the automaton for  $L$ .

Note that the equation  $XL = LX$  can be viewed as a system of two inequalities  $XL \subseteq LX$ ,  $LX \subseteq XL$ . This suggests that similar universality results as for commutation can be obtained also for other systems employing inequalities of the form  $XX \subseteq LX$ .

**Theorem 6.4** (Kunc [43]). *There exist finite languages  $K$ ,  $P$  and regular languages  $L$ ,  $M$ ,  $R$  such that the greatest solutions of both systems  $\{XK \subseteq LX, X \subseteq M\}$  and*

$\{XK \subseteq LX, XP \subseteq RX\}$  are co-r.e.-complete.

The commutation equation is generalized to the conjugacy equation  $XK = LX$ . Languages  $K$  and  $L$  are called *conjugates* via the language  $M$  if  $M$  is a non-empty solution of this equation. Conjugacy of languages was proved decidable in the case of finite biprefix codes by Cassaigne et al. [14]. On the other hand, the expressive power of commutation can be used to show that one cannot decide whether given regular languages are conjugates via a language containing the empty word:

**Theorem 6.5** (Kunc [45]). *One cannot algorithmically decide whether for two given regular languages  $K$  and  $L$  there exists a language  $M$ , which contains the empty word and satisfies  $MK = LM$ .*

This result exhibits a class of simple systems of language equations with regular constants and the only operation of concatenation, where solvability is not algorithmically decidable, namely systems of the form  $XK = LX, XA^* = A^*$ . The present proof of this result is deeply based on the construction for the commutation equation, and no proof, which would either be independent or would use only the computational universality of commutation, is known. It is still an open question whether a similar result can be proved also for conjugacy in general or for conjugacy of finite languages.

The results on commutation of languages contrast with the known properties of words, polynomials and formal series in non-commuting variables over a field, where two elements commute if and only if they are powers of the same element (see Lothaire [51, Sect. 1.2] and Lothaire [52, Ch. 9]).

## 6.4 Equations over a unary alphabet

It was already shown in Section 4.2 that language equations over a unary alphabet  $A = \{a\}$  have quite a significant expressive power, and in particular they can represent the unary encoding of the language  $\text{VALC}(T)$  of computation histories of a Turing machine  $T$ . In the multiple-letter case, the logical dependency of  $L(T)$  upon  $\text{VALC}(T)$  can be expressed in a simple language equation explained in Section 6.2. This argument was recreated for the case of a unary alphabet by using a very special encoding of  $\text{VALC}(T)$ , leading to the following theorem:

**Theorem 6.6** (Jež, Okhotin [32]). *For every recursive (r.e., co-r.e.) language  $L \subseteq a^*$  there is a system of language equations of the form  $\varphi_i(X_1, \dots, X_n) = \psi_i(X_1, \dots, X_n)$  over the alphabet  $\{a\}$ , with  $\varphi_i, \psi_i$  using union, concatenation and singleton constants, which has a unique (least, greatest, respectively) solution with  $X_1 = L$ . The same results hold for unresolved systems with intersection, concatenation and singleton constants.*

The construction is relatively compact if both union and intersection can be used, where Theorem 4.3 can be directly applied. However, using a single Boolean operation, whether it is union or intersection, requires remaking the construction in Theorem 4.3 using equations of this particular form.

The systems constructed in Theorem 6.6 can be further transformed to eliminate the union, leaving the only operation of concatenation. This is done by encoding both union and concatenation by concatenation of unary languages of a specific form, and using additional equations to ensure that specific form. Every variable  $X$  of the original system is represented by a new variable  $X'$  of the new system, with each solution  $X_i = L_i \subseteq a^*$  of the original system corresponding to another solution  $X'_i = \sigma(L_i)$ , where  $\sigma: 2^{a^*} \rightarrow 2^{a^*}$  is a certain *encoding function*. As proved by Jež and Okhotin [33], the mapping  $\sigma$  can be chosen so that  $a^n \in L$  if and only if  $a^{16n+13} \in \sigma(L)$ , while  $\sigma(L) \setminus a^{13}(a^{16})^*$  is a regular constant that does not depend upon  $L$  (called *representing  $L$  on track 13 of  $\sigma(L)$* ). The encoding is checked by an equation  $X'C_1 = C_2$  with fixed regular constants  $C_1, C_2 \subseteq a^*$ , which is satisfied by  $X' = \sigma(L)$  for any  $L \subseteq a^*$ , and which does not hold for  $X'$  of any other form. The simulation of equations is based on the second property of the encoding, that for certain finite constants  $C_3$  and  $C_4$ , the concatenation  $KL$  is represented on track 10 of  $\sigma(K)\sigma(L)C_3$  (and hence  $\sigma(K)\sigma(L)C_3$  depends entirely on  $KL$ ), and similarly  $\sigma(K)\sigma(L)C_4$  represents  $K \cup L$  on track 13. Then an equation  $XY = UV$  of the original system can be replaced by  $X'Y'C_3 = U'V'C_3$ , every equation  $X \cup Y = U \cup V$  is simulated by  $X'Y'C_4 = U'V'C_4$ , and  $X = C$  becomes  $X' = \sigma(C)$ .

Applying one more layer of a similar encoding to the resulting system allows further simplification by encoding all variables into one [48], with a string  $a^n$  in the variable  $X_i$  of the original system represented by the string  $a^{pn+d_i}$  in the unique variable of the new system, for some suitable numbers  $p$  and  $d_i$ . Finally, all equations can be transcoded into just two equations by yet another encoding of the same kind. Since all these encodings are applicable to a system with an arbitrary solution given by Theorem 6.6, the following computational universality result holds:

**Theorem 6.7** (Jež, Okhotin [33]; Lehtinen, Okhotin [48]). *For every recursive (r.e., co-r.e.) set of numbers  $S \subseteq \mathbb{N}_0$  there exist numbers  $p, d \geq 1$ , finite languages  $K, L \subset a^*$  and regular languages  $M, N \subseteq a^*$ , such that the system of two equations*

$$\begin{cases} XXK &= XXL \\ XM &= N \end{cases}$$

*with a variable  $X \subseteq a^*$  has a unique (least, greatest, respectively) solution  $X = L$ , such that  $n \in S$  if and only if  $a^{pn+d} \in L$ . Given a Turing machine recognizing  $S$  (the complement of  $S$  in the case of a greatest solution), such  $p, d, K, L, M$  and  $N$  can be effectively constructed.*

Thus the systems of such a simple form may already have computationally universal solutions. The same technique is used to prove that their decision problems are undecidable: solution existence is co-r.e.-complete, uniqueness is  $\Pi_2^0$ -complete, and existence of finitely many solutions is  $\Sigma_3^0$ -complete [32, 33, 48].

Although encoded forms of computationally complete sets can be represented by equations as simple as in Theorem 6.7, equations with concatenation and no Boolean operations have some combinatorial limitations, which make some simple unary languages non-representable. The known non-representability method is based upon the notions of prime languages (as in Section 4.4) and fragile languages.

A unary language  $L \subseteq a^*$  is called *fragile*, if  $L \cdot \{a^{n_1}, a^{n_2}\}$  is co-finite for all  $n_1, n_2 \in$

$\mathbb{N}$  with  $n_1 \neq n_2$ , while  $L$  itself is not co-finite. Examples of fragile sets representable by these equations are known, and prime sets can be represented as well. However, no set that possesses both properties can be represented:

**Theorem 6.8** (Lehtinen, Okhotin [47]). *Let  $L \subseteq a^*$  be prime and fragile. Then it is not representable by a least or a greatest solution of any system of language equations over a unary alphabet, with concatenation as the only operation and using regular constants.*

An example of a fragile and prime language is the set of all strings  $a^n$ , for which the  $(n + 1)$ -th bit of the binary sequence  $100011110000 \prod_{k=2}^{\infty} (1^k 0)^{2^k}$  is 1.

## 6.5 Infinite systems of equations

A system of infinitely many language equations can be regarded as a binary relation between left-hand sides and right-hand sides, and effectively described by any suitable means, such as a finite transducer recognizing the set of equations. Though one could consider different sets of operations and constants, it is known that satisfiability is undecidable already for very simple infinite systems without any occurrences of variables and using only concatenation and finite constants. In this case, the satisfiability problem actually asks whether a given infinite system of equalities holds true. This was first proved by Lisovik [50] and later improved by Karhumäki and Lisovik [37] and Kunc [46] to systems of the simplest form for which the problem is not trivially decidable:

**Theorem 6.9.** *It is undecidable whether three given finite languages  $K, L, M$  satisfy  $K^n M = L^n M$  for every integer  $n \geq 0$ . In particular, the system of language equations  $\{ X^n Z = Y^n Z \mid n \in \mathbb{N}_0 \}$  in variables  $X, Y, Z$  is not equivalent to any finite system.*

Consider infinite systems of language equations with constants restricted to be singletons and with concatenation as the only operation. Such language equations are similar to *word equations*, with words as unknowns. An arbitrary solution of a word equation constitutes a solution of a language equation in singleton languages. Conversely, for any non-empty solution of a language equation with singleton constants and concatenation, one can construct a solution in words by taking the lexicographically smallest word of minimal length from each component of the original solution. Therefore, solvability of infinite systems of language equations with singleton constants and concatenation, given by finite transducers, is reduced to solvability of the corresponding word equations, which is decidable (see the survey by Harju and Karhumäki [25, Sect. 6.1–7.2]).

## 6.6 Well quasi-orders

This section presents a powerful technique of proving regularity of maximal solutions, which generalizes the method using congruences of finite index, employed in Section 5.1. This technique is based on the fact that in order to prove regularity of a given language, it is sufficient to show that the language is upward closed with respect to a monotone well quasi-order on  $A^*$ .

A quasi-order  $\leq$  is *monotone* if from the inequalities  $u \leq v$  and  $\tilde{u} \leq \tilde{v}$  it follows that  $u\tilde{u} \leq v\tilde{v}$ . The *upward closure* of a language  $K \subseteq A^*$  with respect to a quasi-order  $\leq$  on  $A^*$  is  $\langle K \rangle_{\leq} = \{u \in A^* \mid \exists v \in K : v \leq u\}$ . A quasi-order  $\leq$  on  $A^*$  is called a *well quasi-order* (wqo) if for every language  $L \subseteq A^*$  upward closed with respect to  $\leq$  there exists a finite subset  $K$  of  $L$  such that  $L = \langle K \rangle_{\leq}$ .

**Theorem 6.10** (Ehrenfeucht et al. [21]). *A language  $L \subseteq A^*$  is regular if and only if it is upward closed with respect to some monotone wqo on  $A^*$ .*

The goal is to show that every solution of a certain system of language inequalities is contained in some solution upward closed with respect to a certain monotone wqo. Then Theorem 6.10 implies that all its maximal solutions are regular. The advantage of using wqos, compared to the use of congruences of finite index in Section 5.1, is that they can also be employed when variables and concatenation occur in right-hand sides of inequalities. This is because a quasi-order can classify words according to their decompositions into several factors, taking into account the contexts of these factors in the constant languages occurring in the inequalities.

**Case study I: Restrictions on operations and constants.** Bucher et al. [12] introduced an important class of monotone quasi-orders of words, based on homomorphisms to finite ordered semigroups, that is, semigroups equipped with a monotone ordering of their elements. Given a homomorphism  $\sigma : A^+ \rightarrow S$  to a finite ordered semigroup  $(S, \leq)$ , a monotone quasi-order  $\leq_{\sigma}$  on  $A^*$  is defined as follows: for  $u, v \in A^*$ , where  $v = a_1 \dots a_m$ , with  $a_k \in A$ , let  $v \leq_{\sigma} u$  if there exists a decomposition  $u = u_1 \dots u_m$ , with  $u_k \in A^+$ , such that  $\sigma(u_k) \leq \sigma(a_k)$  for  $k = 1, \dots, m$ . Then the languages upward closed with respect to  $\leq_{\sigma}$  are precisely those which are generated from some set of initial words using the context-free rewriting rules  $a \rightarrow u$ , for  $a \in A$  and  $u \in A^+$  satisfying  $\sigma(u) \leq \sigma(a)$ . Note that  $\leq_{\sigma}$  saturates the quasi-order induced on  $A^+$  by  $\sigma$ , that is, if  $v \leq_{\sigma} u$  then either  $u = v = \varepsilon$  or  $u, v \in A^+$  and  $\sigma(v) \geq \sigma(u)$ . This means that every language  $L \subseteq A^+$  recognized by  $\sigma$  is upward closed with respect to  $\leq_{\sigma}$ .

Characterizing all homomorphisms  $\sigma : A^+ \rightarrow S$  to an ordered semigroup, for which  $\leq_{\sigma}$  is a wqo, is an open problem. The answer is known only for semigroups  $(S, =)$  ordered by the equality relation.

**Theorem 6.11** (Kunc [42]). *For an arbitrary homomorphism  $\sigma : A^+ \rightarrow S$  to a finite semigroup ordered by equality, the relation  $\leq_{\sigma}$  is a well quasi-order on  $A^*$  if and only if for arbitrary words  $u, v \in A^+$  there exist  $x, y \in A^*$  such that  $\sigma(xwvy)$  is equal either to  $\sigma(u)$  or to  $\sigma(v)$  (algebraically, this means that  $\sigma(A^+)$  is a chain of simple semigroups).*

The following generalization of a result of Kunc [42] states that all maximal solutions of very general systems of inequalities are always regular, provided that all operations used in these inequalities (including constants) respect some well quasi-order of the form  $\leq_{\sigma}$ , where  $\sigma$  is a fixed homomorphism to a finite ordered semigroup.

**Theorem 6.12.** *Let  $\sigma : A^+ \rightarrow S$  be a homomorphism to a finite ordered semigroup  $(S, \leq)$ , such that  $\leq_{\sigma}$  is a wqo on  $A^*$ . Let  $\varphi_i(X_1, \dots, X_n) \subseteq$*

$\psi_i(X_1, \dots, X_n)$ , for  $i \in I$ , be a (possibly infinite) system of language inequalities, where  $\varphi_i(\langle L_1 \rangle_{\leq \sigma}, \dots, \langle L_n \rangle_{\leq \sigma}) \subseteq \langle \varphi_i(L_1, \dots, L_n) \rangle_{\leq \sigma}$  and  $\langle \psi_i(L_1, \dots, L_n) \rangle_{\leq \sigma} \subseteq \psi_i(\langle L_1 \rangle_{\leq \sigma}, \dots, \langle L_n \rangle_{\leq \sigma})$ , for every  $n$ -tuple of languages  $(L_1, \dots, L_n)$ . Then each component of every maximal solution of the system is regular and can be expressed as a finite union of concatenations of languages recognized by  $\sigma$ .

*Proof.* The proof proceeds similarly to the proof of Theorem 5.1, but instead of taking the closure of a given solution  $(L_1, \dots, L_n)$  with respect to a congruence relation, one has to take its upward closure with respect to  $\leq_\sigma$ , that is,  $(\langle L_1 \rangle_{\leq \sigma}, \dots, \langle L_n \rangle_{\leq \sigma})$ . To verify that this  $n$ -tuple is also a solution, it is sufficient to combine the assumptions on the functions  $\varphi_i$  and  $\psi_i$  with the inequality  $\langle \varphi_i(L_1, \dots, L_n) \rangle_{\leq \sigma} \subseteq \langle \psi_i(L_1, \dots, L_n) \rangle_{\leq \sigma}$ , which holds because  $(L_1, \dots, L_n)$  is a solution. Since  $\leq_\sigma$  is a wqo, every language  $\langle L_j \rangle_{\leq \sigma}$  is a union of finitely many languages of the form  $\langle a_1 \dots a_m \rangle_{\leq \sigma}$  with  $a_k \in A$ , and for this particular wqo, the latter language is equal to  $\{u_1 \dots u_m \mid \sigma(u_k) \leq \sigma(a_k)\} = \langle a_1 \rangle_{\leq \sigma} \dots \langle a_m \rangle_{\leq \sigma}$ . Since each language  $\langle a_k \rangle_{\leq \sigma}$  is equal to  $\sigma^{-1}\{s \in S \mid s \leq \sigma(a_k)\}$ , it is recognized by  $\sigma$ . Therefore, the language  $\langle L_j \rangle_{\leq \sigma}$  is of the required form.  $\square$

If the sides of inequalities are given as formulae over some basic operations, then the next corollary gives sufficient conditions on these operations, which ensure that their compositions satisfy the theorem.

**Corollary 6.13.** *Assume that a system  $\varphi_i \subseteq \psi_i$  has all  $\varphi_i, \psi_i$  composed of (1) monotone operations  $f$ , which satisfy  $\rho(f(L_1, \dots, L_n)) = f(\rho(L_1), \dots, \rho(L_n))$  for all substitutions  $\rho: A \rightarrow 2^{A^+}$  and for all languages  $L_1, \dots, L_n$ , and (2) constant languages recognized by a fixed homomorphism  $\sigma: A^+ \rightarrow S$  to a finite ordered semigroup  $(S, \leq)$ , such that  $\leq_\sigma$  is a wqo on  $A^*$ . Then the conclusions of Theorem 6.12 hold.*

Since the operations of concatenation, Kleene star and (infinitary) union satisfy the requirements of the corollary, their unrestricted use in equations agrees with the assumptions of Theorem 6.12. Moreover, the theorem also applies if shuffle and (infinitary) intersection are used in the right-hand sides and arbitrary constants are used on the left.

If constants occurring in the system are finitely many group languages, then  $\sigma$  can be taken as a homomorphism to a finite group, and Theorem 6.12 can be applied, because  $\leq_\sigma$  is a wqo by Theorem 6.11. Therefore, one of the consequences of Theorem 6.12 is that the class of polynomials of group languages (that is, regular languages open in the group topology [76], see also Beaudry et al. [7, Cor. 6.1]) is closed under taking maximal solutions of arbitrary systems with rational operations.

**Case study II: Inequalities  $XK \subseteq LX$ .** Using well quasi-orders, it can be also proved, that, unlike in the case of systems of inequalities  $XK \subseteq LX$  considered in Section 6.3, the greatest solution of a single inequality of this form is always regular.

**Theorem 6.14** (Kunc [42]). *If  $K \subseteq A^*$  is arbitrary and  $L \subseteq A^*$  is regular, then the greatest solution of the inequality  $XK \subseteq LX$  is regular.*

The basic idea of this result is to think of the inequality  $XK \subseteq LX$  as a game, as in the case of equations  $XL = LX$  in Section 6.3, and encode all strategies of the defender on

a given word as a labelled tree. Then the standard quasi-ordering of trees, which is a wqo due to Kruskal's Tree Theorem [41], corresponds to simulating the defender's winning strategies. Therefore, the wqo of words, induced by this wqo of the corresponding trees, recognizes the greatest solution.

Theorem 6.14, in particular, implies that for a context-free language  $K$  and a regular language  $L$ , one can algorithmically decide whether a given word belongs to the greatest solution of  $XK \subseteq LX$ . On the one hand, the greatest solution is co-r.e. by Theorem 6.1. On the other hand, it is r.e., because one can test every regular language for being a solution, and any element of the greatest solution shall be found in one of these solutions. However, it is not known whether an automaton for the greatest solution can be algorithmically constructed, except for the special case when both  $K$  and  $L$  are finite and all words in  $K$  are longer than all words in  $L$  (Ly [53]).

## 6.7 Inequations and identity checking

A language equation is an *identity*, if any substitution of languages for its variables turns it into an equality. A typical decision problem is testing whether a given language equation is an identity, and the state of the art on this problem is its decidability for equations with singleton constants, union, concatenation, Kleene star and shuffle, established by Meyer and Rabinovich [57]. Without constants, this problem is decidable if intersection is allowed as well [57]. On the other hand, it remains open whether identity testing is decidable for equations allowing intersection and regular constants along with the above operations.

Note that  $\varphi = \psi$  is an identity if and only if the *inequation*  $\varphi \neq \psi$  has no solutions. This suggests the study of such inequations, as well as of mixed systems of equations  $\varphi = \psi$  and inequations  $\varphi \neq \psi$ , with the same research problems as for any other language equations. The framework of Sections 2 and 6.1 is directly applicable to equations in such systems, and can be applied to inequations as to logical negations of equations.

Thus, a necessary and sufficient condition of solution existence was obtained [64]: a mixed system with concatenation and Boolean operations has a solution if and only if there exists a number  $k \geq 0$ , such that for every  $\ell \geq k$  there exists a solution of the equations modulo  $A^{\leq \ell}$  that satisfies the inequations modulo  $A^{\leq k}$ . In order to characterize solution uniqueness, consider that a mixed system has at most one solution if and only if for every  $\ell \geq 0$ , there exists  $m \geq \ell$ , such that all solutions of the equations modulo  $A^{\leq m}$  that satisfy the inequations modulo  $A^{\leq \ell}$  coincide modulo  $A^{\leq \ell}$ . Uniqueness of a solution is a conjunction of these two conditions [64].

One can infer from the above conditions, that unique solutions of mixed systems using Boolean operations and concatenation must be recursive languages, as in the case of systems of standard equations considered in Theorem 6.1. However, there is an important distinction. Unique solutions of systems of equations are *effectively recursive*, in the sense that one can construct a halting Turing machine recognizing the solution. But for mixed systems, they are *non-effectively recursive*: it is already undecidable whether the unique solution contains the empty word [64].

Decision problems for mixed systems have the following complexity [64]: testing whether a system has a unique solution is complete for the Boolean closure of  $\Sigma_2^0$ , while

solution existence is  $\Sigma_2^0$ -complete.

## 7 Equations with erasing operations

All equations considered so far were restricted to continuous operations. However, some non-continuous operations, such as erasing homomorphisms and the quotient, are commonly used, and equations involving them are also of interest. An early paper by Ruohonen [77] considered equations with homomorphisms and union in connection with Lindenmayer systems. Later, Okhotin [65] briefly investigated equations with quotient, and used them to represent all languages in the arithmetical hierarchy. But this was, in fact, still below the actual expressive power of such equations.

Consider an obvious upper bound on the unique solutions of language equations, assuming that each operation is representable in first-order arithmetic, which is about the weakest reasonable assumption one can make. Then every equation  $\varphi(X_1, \dots, X_n) = \psi(X_1, \dots, X_n)$  can be transcribed by an arithmetical formula  $f(X_1, \dots, X_n)$  using free second-order variables  $X_1, \dots, X_n$  (as well as any quantified first-order variables). Assume the equation has a unique solution; then the membership of words in this solution can be equivalently represented either as  $g(x) = (\exists X_1) \dots (\exists X_n) f(X_1, \dots, X_n) \wedge x \in X_1$ , or as  $g'(x) = (\forall X_1) \dots (\forall X_n) f(X_1, \dots, X_n) \rightarrow x \in X_1$ . Therefore, this solution belongs to the class  $\Sigma_1^1 \cap \Pi_1^1$ , known as the class of *hyper-arithmetical sets*.

On the other hand, an arbitrary hyper-arithmetical subset of  $a^*$  can be represented by a unique solution of a system of language equations of a rather simple form:

**Theorem 7.1** (Jež, Okhotin [35]). *For every hyper-arithmetical set  $S \subseteq \mathbb{N}$  there exists a system of language equations over the alphabet  $A = \{a\}$ , using the operations of concatenation, quotient and union, as well as singleton constants, which has a unique solution with  $X_1 = \{a^n \mid n \in S\}$ . Testing solution existence for such equations is  $\Sigma_1^1$ -complete.*

The argument is technically based upon the constructions of Theorem 6.6. It proceeds by expressing the known definition of hyper-arithmetical sets by infinite unions and intersections directly in a system of language equations. The same kind of result can be proved for any multiple-letter alphabet, by a technically much simpler construction.

## References

- [1] Aiken, A., Kozen, D., Vardi, M., Wimmers, E.: The complexity of set constraints. In *Proc. CSL '93*, LNCS 832, Springer (1994), 1–17.
- [2] J. Autebert, J. Berstel, L. Boasson, “Context-free languages and pushdown automata”, in: Rozenberg, Salomaa (Eds.), *Handbook of formal languages*, Vol. 1, 111–174, Springer, 1997.

- [3] Baader, F., Küsters, R.: Unification in a description logic with transitive closure of roles. In *Proc. LPAR 2001*, LNCS 2250, Springer (2001), 217–232.
- [4] Baader, F., Narendran, P.: Unification of concept terms in description logics. *J. Symbolic Comput.* **31**(3) (2001), 277–305.
- [5] Baader, F., Okhotin, A.: Complexity of language equations with one-sided concatenation and all Boolean operations. In *Proc. UNIF'06* (2006), 59–73.
- [6] Bala, S.: Complexity of regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. *Theory Comput. Syst.* **39**(1) (2006), 137–163.
- [7] Beaudry, M., Lemieux, F., Thérien, D.: Finite loops recognize exactly the regular open languages. In *Proc. ICALP 1997*, LNCS 1256, Springer (1997), 110–120.
- [8] F. Biegler, M. Daley, M. Holzer, I. McQuillan: On the uniqueness of shuffle on words and finite languages. *Theoret. Comput. Sci.* **410**(38-40) (2009) 3711–3724.
- [9] A. Blass, Yu. Gurevich, “On the unique satisfiability problem”, *Information and Control*, 55 (1982), 80–88.
- [10] V. G. Bondarchuk, “Sistemy uravnenii v algebre sobytii” (Systems of equations in the event algebra), in Russian, *Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki* (Journal of Computational Mathematics and Mathematical Physics), 3:6, 1963.
- [11] J. A. Brzozowski, E. L. Leiss, “On equations for regular languages, finite automata, and sequential networks”, *Theoretical Computer Science*, 10 (1980), 19–35.
- [12] W. Bucher, A. Ehrenfeucht, D. Haussler, On total regulators generated by derivation relations, *Theoret. Comput. Sci.* **40** (1985) 131–148.
- [13] Câmpeanu, C., Salomaa, K., Vágvölgyi, S.: Shuffle decompositions of regular languages. *Internat. J. Found. Comput. Sci.* **13**(6) (2002), 799–816.
- [14] Cassaigne, J., Karhumäki, J., Salmela, P.: Conjugacy of finite biprefix codes. *Theoret. Comput. Sci.* **410**(24-25) (2009) 2345–2351.
- [15] N. Chomsky, “Three models for the description of language”, *IRE Transactions on Information Theory*, 2 (1956), 113–124.
- [16] Conway, J.H.: *Regular Algebra and Finite Machines*. Chapman and Hall (1971).
- [17] K. Čulík II, “Variations of the firing squad problem and applications”, *Information Processing Letters*, 30:3 (1989), 152–157.
- [18] K. Čulík II, J. Gruska, A. Salomaa, “Systolic trellis automata”, I and II, *International Journal of Computer Mathematics*, 15 (1984), 195–212, and 16 (1984), 3–22.
- [19] Domaratzki, M., Salomaa, K.: Decidability of trajectory-based equations. *Theoret. Comput. Sci.* **345**(2–3) (2005), 304–330.
- [20] C. Dyer, “One-way bounded cellular automata”, *Information and Control*, 44 (1980), 261–281.
- [21] Ehrenfeucht, A., Haussler, D., Rozenberg, G.: On regularity of context-free languages. *Theoret. Comput. Sci.* **27**(3) (1983), 311–332.
- [22] Z. Ésik, W. Kuich, “Boolean fuzzy sets”, *International Journal of Foundations of Computer Science*, 18:6 (2007), 1197–1207.
- [23] A.E. Frid: Simple equations on binary factorial languages. *Theoret. Comput. Sci.* **410**(30-32) (2009), 2947–2956.

- [24] Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. *J. ACM* **9**(3) (1962), 350–371.
- [25] T. Harju, J. Karhumäki, “Morphisms”, in: Rozenberg, Salomaa (Eds.), *Handbook of formal languages*, Vol. 1, 439–510, Springer, 1997.
- [26] J. Hartmanis, “Context-free languages and Turing machine computations”, *Proceedings of Symposia in Applied Mathematics*, Vol. 19, AMS, 1967, 42–51.
- [27] O. H. Ibarra, S. M. Kim, “Characterizations and computational complexity of systolic trellis automata”, *Theoretical Computer Science*, **29** (1984), 123–153.
- [28] E. Jeandel, N. Ollinger, “Playing with Conway’s problem”, *Theoretical Computer Science* **409**:3 (2008), 557–564.
- [29] A. Jež, “Conjunctive grammars can generate non-regular unary languages”, *International Journal of Foundations of Computer Science*, **19**:3 (2008), 597–615.
- [30] A. Jež, A. Okhotin, “Conjunctive grammars over a unary alphabet: undecidability and unbounded growth”, *Theory of Computing Systems*, **46**:1 (2010), 27–58.
- [31] A. Jež, A. Okhotin, “Complexity of equations over sets of natural numbers”, *Theory of Computing Systems*, **48**:2 (2011), 319–342.
- [32] A. Jež, A. Okhotin, “On the computational completeness of equations over sets of natural numbers”, *Automata, Languages and Programming (ICALP 2008)*, Reykjavík, Iceland, July 6–13, 2008), part II, LNCS 5126, 63–74.
- [33] A. Jež, A. Okhotin, “Equations over sets of natural numbers with addition only”, *STACS 2009* (Freiburg, Germany, 26–28 February, 2009), 577–588.
- [34] A. Jež, A. Okhotin, “One-nonterminal conjunctive grammars over a unary alphabet”, *Theory of Computing Systems*, **49**:2 (2011), 319–342.
- [35] A. Jež, A. Okhotin, “Representing hyper-arithmetical sets by equations over sets of integers”, *Theory of Computing Systems*, accepted.
- [36] Karhumäki, J., Latteux, M., Petre, I.: Commutation with codes. *Theoret. Comput. Sci.* **340**(2) (2005), 322–333.
- [37] Karhumäki, J., Lisovik, L.P.: The equivalence problem of finite substitutions on  $ab^*c$ , with applications. *Internat. J. Found. Comput. Sci.* **14**(4) (2003), 699–710.
- [38] Kari, L.: On language equations with invertible operations. *Theoret. Comput. Sci.* **132**(1–2) (1994), 129–150.
- [39] Kirsten, D.: A Burnside approach to the finite substitution problem. *Theory of Computing Systems* **39**(1) (2006), 15–50.
- [40] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, “Well-founded semantics for Boolean grammars”, *Information and Computation*, **207**:9 (2009), 945–967.
- [41] Kruskal, J.B.: Well-quasi-ordering, the tree theorem, and Vazsonyi’s conjecture. *Trans. Amer. Math. Soc.* **95**(2) (1960), 210–225.
- [42] Kunc, M.: Regular solutions of language inequalities and well quasi-orders. *Theoret. Comput. Sci.* **348**(2–3) (2005), 277–293.
- [43] Kunc, M.: On language inequalities  $XK \subseteq LX$ . In *Proc. DLT 2005*, LNCS 3572, Springer (2005), 327–337.
- [44] Kunc, M.: Largest solutions of left-linear language inequalities. In *Proc. AFL 2005*, University of Szeged (2005), 178–186.

- [45] M. Kunc, The power of commuting with finite sets of words, *Theory of Computing Systems*, 40:4 (2007), 521–551.
- [46] M. Kunc, “The simplest language where equivalence of finite substitutions is undecidable”, *Fundamentals of Computation Theory (FCT 2007, Budapest, Hungary, August 27–30, 2007)*, LNCS 4639, 365–375.
- [47] T. Lehtinen, A. Okhotin, “On equations over sets of numbers and their limitations”, *International Journal of Foundations of Computer Science*, 22:2 (2011), 377–393.
- [48] T. Lehtinen, A. Okhotin, “On language equations  $XXK = XXL$  and  $XM = N$  over a unary alphabet”, *Developments in Language Theory (DLT 2010, London, Ontario, Canada, 17–20 August 2010)*, LNCS 6224, 291–302.
- [49] E. L. Leiss, “Unrestricted complementation in language equations over a one-letter alphabet”, *Theoretical Computer Science*, 132 (1994), 71–93.
- [50] Lisovik, L.P.: The equivalence problem for finite substitutions on regular languages. *Dokl. Akad. Nauk* **357**(3) (1997), 299–301.
- [51] M. Lothaire: *Combinatorics on Words*. Cambridge University Press (1997).
- [52] M. Lothaire: *Algebraic Combinatorics on Words*. Cambridge University Press (2002).
- [53] O. Ly, “A constructive solution of the language inequation  $XA \subseteq BX$ ”, *Theory and Applications of Language Equations (TALE 2007, Turku, Finland, July 2, 2007)*, Turku Centre for Computer Science GP 44, 76–84.
- [54] W. Martens, M. Niewerth, T. Schwentick, “Schema design for XML repositories: complexity and tractability”, *PODS 2010 (Indianapolis, USA, June 6–11, 2010)*, 239–250.
- [55] Massazza, P., Salmela, P.: On the simplest centralizer of a language. *Theor. Inform. Appl.* **40**(2) (2006), 295–301.
- [56] A. Mateescu, G. Rozenberg, A. Salomaa: Shuffle on trajectories: syntactic constraints. *Theoret. Comput. Sci.* **197**(1–2) (1998), 1–56.
- [57] A. R. Meyer, A. M. Rabinovich, “Valid identity problem for shuffle regular expressions”, *Journal of Automata, Languages and Combinatorics*, 7:1 (2002), 109–125.
- [58] Okhotin, A.: Conjunctive grammars. *J. Autom. Lang. Comb.* **6**(4) (2001), 519–535.
- [59] Okhotin, A.: Conjunctive grammars and systems of language equations. *Program. Comput. Software* **28**(5) (2002), 243–249.
- [60] A. Okhotin, “On the number of nonterminals in linear conjunctive grammars”, *Theoretical Computer Science*, 320:2–3 (2004), 419–448.
- [61] Okhotin, A.: Boolean grammars. *Inform. and Comput.* **194**(1) (2004), 19–48.
- [62] Okhotin, A.: On the equivalence of linear conjunctive grammars and trellis automata. *Theor. Inform. Appl.* **38**(1) (2004), 69–88.
- [63] Okhotin, A.: Unresolved systems of language equations: Expressive power and decision problems. *Theoret. Comput. Sci.* **349**(3) (2005), 283–308.
- [64] Okhotin, A.: Strict language inequalities and their decision problems. In *Proc. MFCS 2005*, LNCS 3618, Springer (2005), 708–719.
- [65] Okhotin, A.: Computational universality in one-variable language equations. *Fund. Inform.* **74**(4) (2006), 563–578.
- [66] Okhotin, A.: Language equations with symmetric difference. In *Proc. CSR 2006*, LNCS 3967, Springer (2006), 292–303.

- [67] A. Okhotin, “Seven families of language equations”, *AutoMathA 2007*, Palermo, Italy, June 18–22, 2007.
- [68] A. Okhotin, “Decision problems for language equations”, *Journal of Computer and System Sciences*, 76:3–4 (2010), 251–266.
- [69] A. Okhotin, “Fast parsing for Boolean grammars: a generalization of Valiant’s algorithm”, *Developments in Language Theory (DLT 2010, London, Ontario, Canada, August 17–20, 2010)*, LNCS 6224, 340–351.
- [70] A. Okhotin, “Conjunctive and Boolean grammars: the true general case of the context-free grammars”, manuscript.
- [71] A. Okhotin, C. Reitwießner, “Conjunctive grammars with restricted disjunction”, *Theoretical Computer Science*, 411:26–28 (2010), 2559–2571.
- [72] A. Okhotin, P. Rondogiannis, “On the expressive power of univariate equations over sets of natural numbers”, *IFIP Intl. Conf. on Theoretical Computer Science (TCS 2008, Milan, Italy, 8–10 September, 2008)*, IFIP vol. 273, 215–227.
- [73] A. Okhotin, O. Yakimova, “Language equations with complementation: Decision problems”, *Theoretical Computer Science*, 376:1–2 (2007), 112–126.
- [74] A. Okhotin, O. Yakimova, “Language equations with complementation: Expressive power”, *Theoretical Computer Science*, to appear.
- [75] Parikh, R., Chandra, A., Halpern, J., Meyer, A., “Equations between regular terms and an application to process logic”, *SIAM J. Comput.* **14**(4) (1985), 935–942.
- [76] Pin, J.-É.: Polynomial closure of group languages and open sets of the Hall topology. *Theoret. Comput. Sci.* **169**(2) (1996), 185–200.
- [77] K. Ruohonen, “A note on language equations involving morphisms”, *Information Processing Letters*, 7:5 (1978), 209–212.
- [78] Sakarovitch, J.: *Elements of Automata Theory*. Cambridge University Press (2009).
- [79] A. Salomaa, *Theory of Automata*, Pergamon Press, Oxford, 1969.
- [80] V. Terrier, “On real-time one-way cellular array”, *Theoretical Computer Science*, 141 (1995), 331–335.
- [81] K. Weihrauch, *Computable Analysis: An Introduction*, Springer, 2000.
- [82] D. Wotschke, “The Boolean closures of deterministic and nondeterministic context-free languages”, In: W. Brauer (ed.), *Gesellschaft für Informatik e. V., 3. Jahrestagung 1973*, LNCS 1, 113–121.

# Index

- arithmetical hierarchy, 28
- automata
  - desert, 17
- automaton
  - looping tree
  - independent (ILTA), 7
- cellular automaton
  - one-way real-time, 13
- computable operation, *see* language operation, computable
- conjugates, 22
- continuous operation, *see* language operation, continuous
- finite automaton
  - alternating (AFA), 6
  - nondeterministic (NFA), 6
  - universal, 16
- grammar
  - Boolean, 11
  - conjunctive, 10
    - linear, 13
  - context-free, 9
- hyper-arithmetical set, 28
- language
  - conjunctive, 10–13
    - linear, 13–14
  - context-free, 9–10
  - Dyck, 13
  - fragile, 23
  - group, 26
    - polynomial of, 26
  - P-complete, 14
  - prime, 14, 23
  - recursive, 15, 18–24
  - recursively enumerable (r.e.), 18–24
  - sequence, limit of, 3
- language equation, 2
  - explicit, 8
  - resolved, 8
  - strict, 8
- language operation
  - computable, 18
  - continuous, 3–4
  - monotone, 4–5, 26
  - word-based, 5
- Lindenmayer system, 28
- monotone operation, *see* language operation, monotone
- semigroup
  - ordered, 25
  - simple, 25
- set constraints, 7
- trellis automaton, *see* cellular automaton, one-way real-time
- well quasi-order (wqo), 24–27
- word equation, 24
- word-based operation, *see* language operation, word-based