Alexander Okhotin

Seven families of language equations

Turku Centre *for* Computer Science

TUCS Technical Report
No 844, June 2015

Seven families of language equations

Alexander Okhotin
alexander.okhotin@utu.fi
Department of Mathematics and Statistics, University of Turku,
Turku FI–20014, Finland

TUCS Technical Report
No 844, June 2015

Abstract

Systems of equations of the form $X_i = \varphi_i(X_1, \ldots, X_n)$, for $1 \leqslant i \leqslant n$, in which the unknowns X_i are formal languages, and the right-hand sides φ_i may contain concatenation and union, are known for representing context-free grammars. If, instead of union only, another set of Boolean operations is used, the expressive power of such equations may change: for example, using both union and intersection leads to conjunctive grammars (Okhotin, 2001), whereas using all Boolean operations allows all recursive sets to be expressed by unique solutions (Okhotin, 2003). This paper investigates the expressive power of such equations with any possible set of Boolean operations. It is determined that different sets of Boolean operations give rise to exactly seven families of formal languages: the recursive languages, the conjunctive languages, the context-free languages, a certain family incomparable with the context-free languages, as well as three subregular families.

This paper is an extended version of an invited talk given at the *AutoMathA 2007* conference held in Palermo, Italy on June 18–22, 2007.

Keywords: Language equations, Boolean operations, Post's lattice.

TUCS Laboratory
FUNDIM, Fundamentals of Computing and Discrete Mathematics

1 Introduction

Equations with formal languages as unknowns are among the natural objects of study in formal language theory. The most frequently used class of equations are systems of the following form.

$$\begin{cases} X_1 &= \varphi_1(X_1, \ldots, X_n) \\ &\vdots \\ X_n &= \varphi_n(X_1, \ldots, X_n) \end{cases} \tag{*}$$

Here the unknowns $X_1, \ldots, X_n$ are formal languages over a certain alphabet Σ, and the right-hand sides φ_i may use singleton constant languages, the concatenation operation, as well as some Boolean operations on languages.

If the only allowed Boolean operation is union, then, as shown by Ginsburg and Rice [8], these systems represent the basic mathematical model of syntax, known in the literature as a *context-free grammar*. To be precise, every grammar can be transcribed as such a system of equations, with nonterminal symbols becoming variables, so that the least solution of that system (with respect to inclusion) is exactly the vector of languages generated by those nonterminal symbols. For example, consider the following grammar over the alphabet $\Sigma = \{a, b\}$, and the corresponding one-variable equation.

$$X \to aXb \mid \varepsilon \qquad\qquad X = (\{a\} \cdot X \cdot \{b\}) \cup \{\varepsilon\}$$

In this equation, X is an unknown language, while $\{a\}$, $\{b\}$ and $\{\varepsilon\}$ are singleton constant languages, and the least solution of the equation is the language $\{a^n b^n \mid n \geqslant 0\}$. In this particular case, the solution is actually unique; in general, any grammar can be tranformed to the Greibach normal form, in which the solution is always unique. Therefore, the class of languages defined by unique solutions of equations (*) is exactly the class of the context-free languages.

The idea behind these equations—and behind formal grammars in general—is *inductive definition* of strings possessing certain properties. Each variable (nonterminal symbol) represents a property that each string may have or not have, and the equations (rules of a grammar) describe the structure of strings with a certain property as a combination of shorter strings with known properties. In ordinary (Chomsky's "context-free") grammars, longer strings are obtained by concatenating shorter ones, and each property is defined as a *disjunction* of such concatenations, This disjunction is represented in language equations as the union operation, and using other sets of Boolean operations could lead (and occasionally leads) to new classes of formal grammars.

The most obvious choice is to allow a conjunction operation alongside the disjunction. The resulting family of *conjunctive grammars* [26] is notable for

inheriting most of the parsing algorithms from ordinary grammars [34], in particular, the subcubic parsing through matrix multiplication [35]. At the same time, conjunctive grammars can represent a few syntactic constructs beyond the scope of ordinary grammars [34]. Conjunctive grammars are characterized by language equations (*) with concatenation, union and intersection [27].

The next obvious step is to add the negation operation. In terms of language equations, these will be systems (*) with concatenation and all Boolean operations. Having seen the conjunctive grammars, one could expect these systems to be another slightly more powerful variant of formal grammars, with expressive power well within polynomial time. However, it turned out that these equations can represent logical dependence of shorter strings upon longer ones, thus violating the principle of inductive definition of strings, and allowing every recursive set to be described by a unique solution of some system (*). Conversely, every representable set is recursive [28, 32].

The purpose of this paper is to consider systems (*) with concatenation, singleton constants and *any possible sets of Boolean operations*. For each set of Boolean operations, there is a corresponding family of formal languages defined by unique solutions of these systems. How many distinct language families could be obtained in that way?

The main result of this paper is that there exist exactly seven such classes (six for a unary alphabet). An essential tool for this study is the fundamental work by Post [42] on the classes of Boolean functions closed under composition, reviewed in Section 2 and adapted to language equations in Section 3. Even though Post's lattice of closed classes of Boolean functions contains countably many clases, this lattice is split into seven regions, giving rise to distinct families of formal languages defined by language equations. This partition is carried out in Section 4, where each of the seven regions is painted over Post's lattice, and the corresponding family of languages is characterized. These families are denoted by O, I, K, D, M, N and P, more or less after their respective generating classes of Boolean functions, and their hierarchy is established in Section 5.

The last Section 6 reviews the previous research on language equations of the form other than (*), and elaborates on possible applications of Post's lattice to that research.

2 Post's lattice

Denote the set of Boolean constants by $\mathbb{B} = \{0, 1\}$, and consider Boolean functions $f\colon \mathbb{B}^k \to \mathbb{B}$, where $k \geqslant 0$ is the number of arguments. The basic examples of Boolean functions are the standard propositional connectives, such as conjunction $f_1(x, y) = x \wedge y$, disjunction $f_2(x, y) = x \vee y$, implication

$f_3(x, y) = x \to y$ and sum modulo two $f_4(x, y) = x \oplus y$ (with two arguments each), negation $f_5(x) = \neg x$ and the identity function $f_6(x) = x$ (with one argument each), as well as constants 0 and 1 (with no arguments). The set of all Boolean functions is denoted by P_2, where the number 2 indicates binary logic.

Definition 1. *Let $f\colon \mathbb{B}^k \to \mathbb{B}$, with $k \geqslant 1$, be a Boolean function, and consider a substitution of Boolean functions $g_i\colon \mathbb{B}^{\ell_i} \to \mathbb{B}$, with $\ell_i \geqslant 1$, for all $i \in \{1, \ldots, k\}$, into the arguments of f. The resulting* composition *is any function $h\colon \mathbb{B}^n \to \mathbb{B}$ representable in the form*

$$h(x_1, \ldots, x_n) = f\big(g_1(x_{m_{1,1}}, \ldots, x_{m_{1,\ell_1}}), \ldots, g_k(x_{m_{k,1}}, \ldots, x_{m_{k,\ell_k}})\big),$$

where the subscripts $m_{i,j} \in \{1, \ldots, n\}$ are numbers of any arguments of f.

A set of functions $\mathcal{F} \subseteq P_2$ is said to be closed *(under composition), if $f, g_1, \ldots, g_k \in \mathcal{F}$ implies $h \in \mathcal{F}$.*

Post referred to sets of functions closed under composition as "closed systems", whereas some of the subsequent literature adopted the term "clone". For every set of functions $\mathcal{F} \subseteq P_2$, its *closure* (under composition), denoted by $[\mathcal{F}]$, is the smallest set of functions containing every function from $\mathcal{F}$ and closed under composition.

Consider the following five closed classes of Boolean functions.

- T_0: *functions preserving zero*, that is, with $f(0, \ldots, 0) = 0$.
- T_1: *functions preserving one*, that is, with $f(1, \ldots, 1) = 1$.
- S: *self-dual functions*, that is, those that satisfy the identity $\neg f(\neg x_1, \ldots, \neg x_n) = f(x_1, \ldots, x_n)$ for all $x_1, \ldots, x_n \in \mathbb{B}$.
- M: *monotone functions*, for which $f(b_1, \ldots, b_n) \leqslant f(c_1, \ldots, c_n)$ whenever $b_i \leqslant c_i$ for all i.
- L: *linear functions*, representable in the form $f(x_1, \ldots, x_n) = x_{i_1} \oplus \ldots \oplus x_{i_m} \oplus c$, for some $m \geqslant 0$, $1 \leqslant i_1 < \ldots < i_m \leqslant n$ and $c \in \mathbb{B}$.

These classes are collectively known as *the five pre-complete classes*, because of the following noteworthy result.

Post's Little Theorem ([41]). *Let $\mathcal{F} \subseteq P_2$ be a set of Boolean functions. Then $[\mathcal{F}] = P_2$ if and only if $\mathcal{F}$ is not contained in any of the classes T_0, T_1, S, M, L.*

For instance, the well-known result that every Boolean function is representable as a formula over the single base function, the *Sheffer stroke*, $f(x, y) = \neg(x \wedge y)$, follows from this theorem, because f belongs to none of the five pre-complete classes.

Post's research on Boolean functions eventually led to a complete description of all classes of Boolean functions closed under composition.

Post's Theorem ([42]). *The (countably many) classes listed in Table 1 are all closed classes of Boolean functions. Each class has a finite basis. Their lattice of containment is of the form given in Figure 1.*

The names of the classes are given in the notation of Yablonski et al. [45], who gave a simplified proof and explanation of Post's results. In total, there are 8 infinite (countable) hierarchies and 44 individual classes. For a proof of Post's theorem, the reader is directed to the cited book by Yablonski et al. [45], as well as to a more recent text by Lau [20].

The class P_2 at the top of Figure 1 is the class of all Boolean functions, which is generated, for instance, as $[x \vee y,\ \neg x]$. Each of the rest of the families has its own basis, such as $D_{01} = [x \vee y]$. Each line specifies a proper containment of a class located lower in the figure within a higher-located class.

In part of the literature, such as in the monograph by Lau [20], Post's classes are defined slightly differently, so that projections are implicitly included in every basis, thus collapsing a few bottom classes in the hierarchy. However, the application of Post's theory to language equations developed in this paper is not affected by these fine details.

3 Language equations with Boolean operations

Let Σ be a finite alphabet and let $\mathcal{F} \subseteq P_2$ be any set of Boolean functions. Consider systems of language equations of the resolved form.

$$\left\{ \begin{array}{lcl} X_1 & = & \varphi_1(X_1, \ldots, X_n) \\ & \vdots & \\ X_n & = & \varphi_n(X_1, \ldots, X_n) \end{array} \right. \qquad (*)$$

Here the unknowns X_i are formal languages over Σ, and the expressions φ_i may contain these variables, singleton constant languages, the operation of concatenation, as well as any Boolean operations from $\mathcal{F}$ defined on sets. In particular, Boolean constant 0 defines the empty set, constant 1 defines the set Σ^*, disjunction represents union, sum modulo two represents symmetric difference, etc.

Formally, the set of expressions admissible on the right-hand sides of equations is defined as follows:

- every variable X_i is an expression;
- a constant language $\{a\}$, with $a \in \Sigma$, is an expression;
- a concatenation of two expressions is an expression;

P_2	$x \vee y$, $\neg x$
M	0, 1, $x \vee y$, $x \wedge y$
L	1, $x \oplus y$
D	0, 1, $x \vee y$
K	0, 1, $x \wedge y$
U	1, $\neg x$
MU	0, 1, x
C	0, 1

T_{01}	$x \vee (y \wedge \neg z)$, $x \wedge y$
S_{01}	$d_3(\neg x, y, z)$
M_{01}	$x \vee y$, $x \wedge y$
L_{01}	$x \oplus y \oplus z$
D_{01}	$x \vee y$
K_{01}	$x \wedge y$
U_{01}	x
S	$\neg x$, d_3
SM	d_3
SL	$x \oplus y \oplus z \oplus 1$
SU	$\neg x$
O_0^m $(m \geqslant 2)$	$x \vee (y \wedge \neg z)$, d_{m+1}
MO_0^2	$x \vee y$, d_3
MO_0^m $(m \geqslant 3)$	d_{m+1}
I_1^m $(m \geqslant 2)$	$x \wedge (y \vee \neg z)$, d^*_{m+1}
MI_1^2	$x \wedge y$, d_3
MI_1^m $(m \geqslant 3)$	d^*_{m+1}
O_0^∞	$x \vee (y \wedge \neg z)$
MO_0^∞	$x \vee (y \wedge z)$
I_1^∞	$x \wedge (y \vee \neg z)$
MI_1^∞	$x \wedge (y \vee z)$
$\varnothing$	$\varnothing$

T_0	$x \wedge \neg y$, $x \vee y$
M_0	0, $x \vee y$, $x \wedge y$
L_0	$x \oplus y$
D_0	0, $x \vee y$
K_0	0, $x \wedge y$
U_0	0, x
C_0	0
I^m $(m \geqslant 2)$	$x \wedge \neg y$, d^*_{m+1}
MI^m $(m \geqslant 2)$	0, d^*_{m+1}
I^∞	$x \wedge \neg y$
MI^∞	0, $x \wedge (y \vee z)$

T_1	$x \vee \neg y$, $x \wedge y$
M_1	1, $x \vee y$, $x \wedge y$
L_1	$x \oplus y \oplus 1$
D_1	1, $x \vee y$
K_1	1, $x \wedge y$
U_1	1, x
C_1	1
O^m $(m \geqslant 2)$	$x \vee \neg y$, d_{m+1}
MO^m $(m \geqslant 2)$	1, d_{m+1}
O^∞	$x \vee \neg y$
MO^∞	1, $x \vee (y \wedge z)$

Table 1: Post's classes of Boolean functions and their finite bases. Notation: $d_m(x_1, \ldots, x_m) = \bigvee_{1 \leqslant i < j \leqslant m}(x_i \wedge x_j)$ and $d^*_m(x_1, \ldots, x_m) = \bigwedge_{1 \leqslant i < j \leqslant m}(x_i \vee x_j)$.

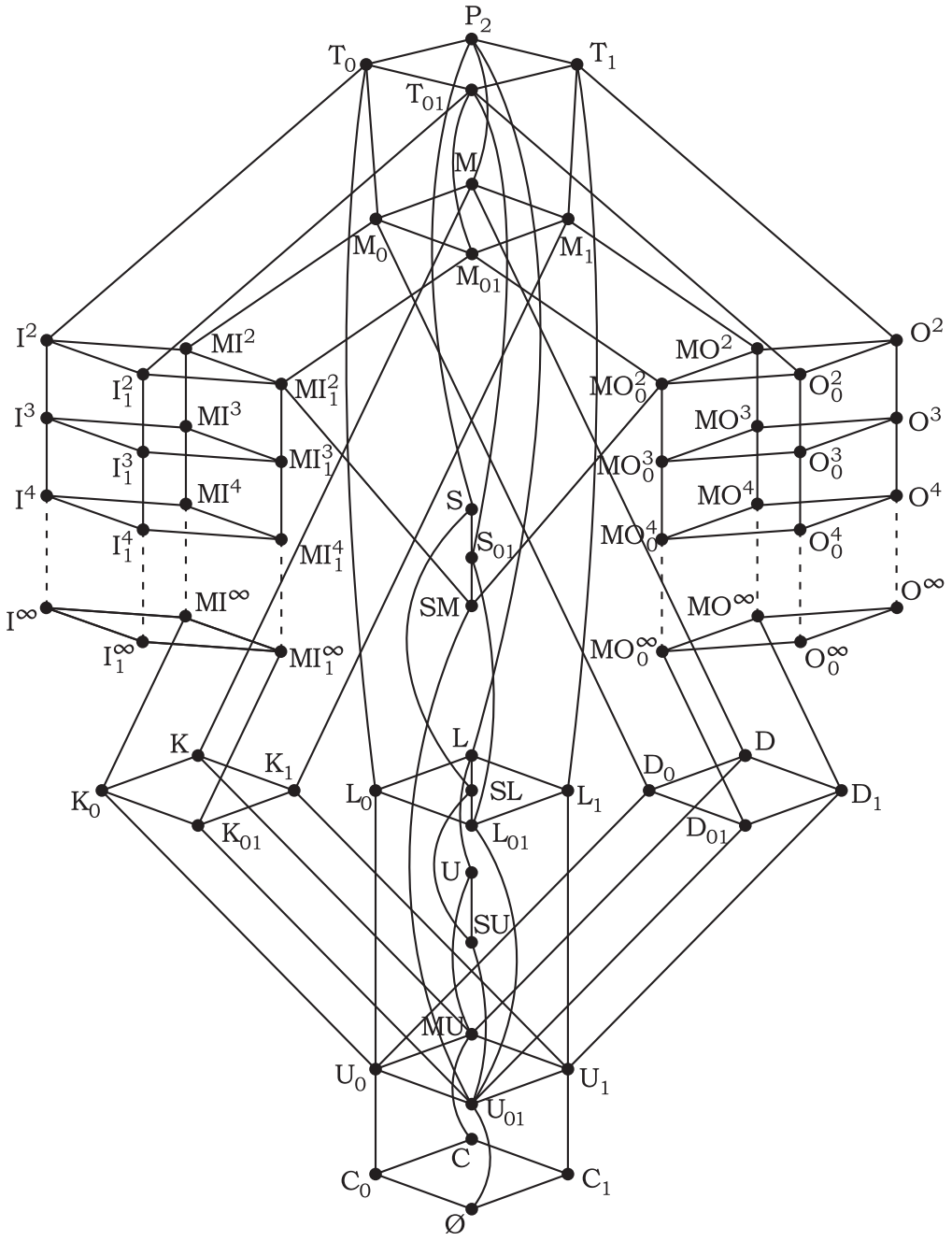

Figure 1: Post's lattice, presented in the notation of Yablonski et al. [45].

- if $f\colon \mathbb{N}^k \to \mathbb{N}$ is a Boolean function from $\mathcal{F}$ and $\eta_1, \ldots, \eta_k$ are expressions, then $f(\eta_1, \ldots, \eta_k)$ is an expression.

The value of an expression on a substitution $X_1 = L_1, \ldots, X_n = L_n$ is defined inductively on its structure. In particular, if $\eta_1, \ldots, \eta_k$ are expressions with values $M_1, \ldots, M_k \subseteq \Sigma^*$, then the value of $f(\eta_1, \ldots, \eta_k)$ is the language $\{w \mid f(x_1, \ldots, x_k) = 1, \text{ where } x_i = 1 \text{ if } w \in M_i, \text{ and } x_i = 0 \text{ if } w \notin M_i\}$. A vector of languages $(L_1, \ldots, L_n)$ is a solution of the system (*) if the value of each expression φ_i under the substitution $X_1 = L_1, \ldots, X_n = L_n$ is exactly L_i.

Let $\mathcal{L}_{\Sigma,\mathcal{F}} \subseteq 2^{\Sigma^*}$ be the family of languages representable by unique solutions of such systems; that is, $L \in \mathcal{L}_{\Sigma,\mathcal{F}}$ if and only if there exists a system (*) with a unique solution $X_1 = L$, $X_2 = L_2$, $\ldots$, $X_n = L_n$, for some languages $L_2, \ldots, L_n \subseteq \Sigma^*$. The question studied in this paper is, how many distinct language families can be obtained by using different sets $\mathcal{F}$, and what are these families?

First of all, note that the syntax of language equations allows any function composition to be expressed in the right-hand side of any equation. Therefore, one can always implement any Boolean operation from the closure $[\mathcal{F}]$ by combining operations from $\mathcal{F}$. Accordingly, one can assume that $\mathcal{F}$ is one of Post's classes.

Furthermore, in some cases, one can construct a system of equations using Boolean operations from $\mathcal{F}$ that implements a Boolean function not in the closure $[\mathcal{F}]$. For instance, Boolean constant 0 can be expressed by the equation $X = aX$ with a unique solution $X = \varnothing$, which is effectively constant 0. This is something that, according to Post's theorem, cannot be achieved by function composition.

In this paper, Boolean functions shall often be expressed in this way, in order to prove that some Boolean operations (such as constant 0) may be eliminated in a given system of language equations. The necessary notion of expressibility is formally defined as follows.

Definition 2. *A Boolean function $f(x_1, \ldots, x_k)$ is said to be expressible by language equations with Boolean operations $\mathcal{F}$ over an alphabet Σ, if there exists a system of language equations (*) in variables $X_1, \ldots, X_k, Y, Z_1, \ldots, Z_n$, for some $n \geqslant 0$, using functions from $\mathcal{F}$, and some functions on languages $\varphi_1, \ldots, \varphi_n\colon (2^{\Sigma^*})^k \to 2^{\Sigma^*}$, and that system is equivalent to the following sys-*

tem of equations (in the sense of having the same set of solution).

$$\begin{aligned} X_1 &= X_1 \\ &\vdots \\ X_k &= X_k \\ Y &= f(X_1, \ldots, X_n) \\ Z_1 &= \varphi_1(X_1, \ldots, X_k) \\ &\vdots \\ Z_n &= \varphi_n(X_1, \ldots, X_k) \end{aligned}$$

In other words, the system imposes no restrictions on the values of $X_1, \ldots, X_k$, and ensures that Y is their desired Boolean combination. The remaining auxiliary variables $Z_1, \ldots, Z_n$ functionally depend on $X_1, \ldots, X_k$, so that solution uniqueness is preserved in all constructions involving Definition 2. The basic construction is the one given below.

Proposition 1. *Let $\mathcal{F}$ be a class of Boolean functions, let f be a function not in $\mathcal{F}$, which is expressible by language equations with operations $\mathcal{F}$ over an alphabet Σ. Then, for every system using concatenation and operations from $\mathcal{F} \cup \{f\}$ that has a unique solution, with a language $L \subseteq \Sigma^*$ as one of its components, there is another system using concatenation and operations from $\mathcal{F}$, which also has a unique solution with L among its components.*

Indeed, every occurrence of f can be substituted with the construction in Definition 2, which produces a system with the desired properties.

4 The seven families

This study proceeds by splitting Post's lattice into seven fragments, centered around the following classes: D_{01} (disjunction only), M_{01} (disjunction and conjunction), P_2 (all Boolean operations), U (complementation only), $\varnothing$ (no Boolean operations), C_1 (only constant 1) and K_1 (conjunction and constant 1). For each of these base classes, it is shown that several neighbouring classes in Post's lattice, when used in language equations, define the same family of languages. This is presented in Lemmata 1–7 below, which, together, cover Post's lattice completely, as shown in Figure 2.

D: Disjunction only

If the only allowed Boolean operation is disjunction, one obtains the well-known equations of Ginsburg and Rice [8]. These equations constitute one of the definitions of the context-free grammars that is equivalent to Chomsky's

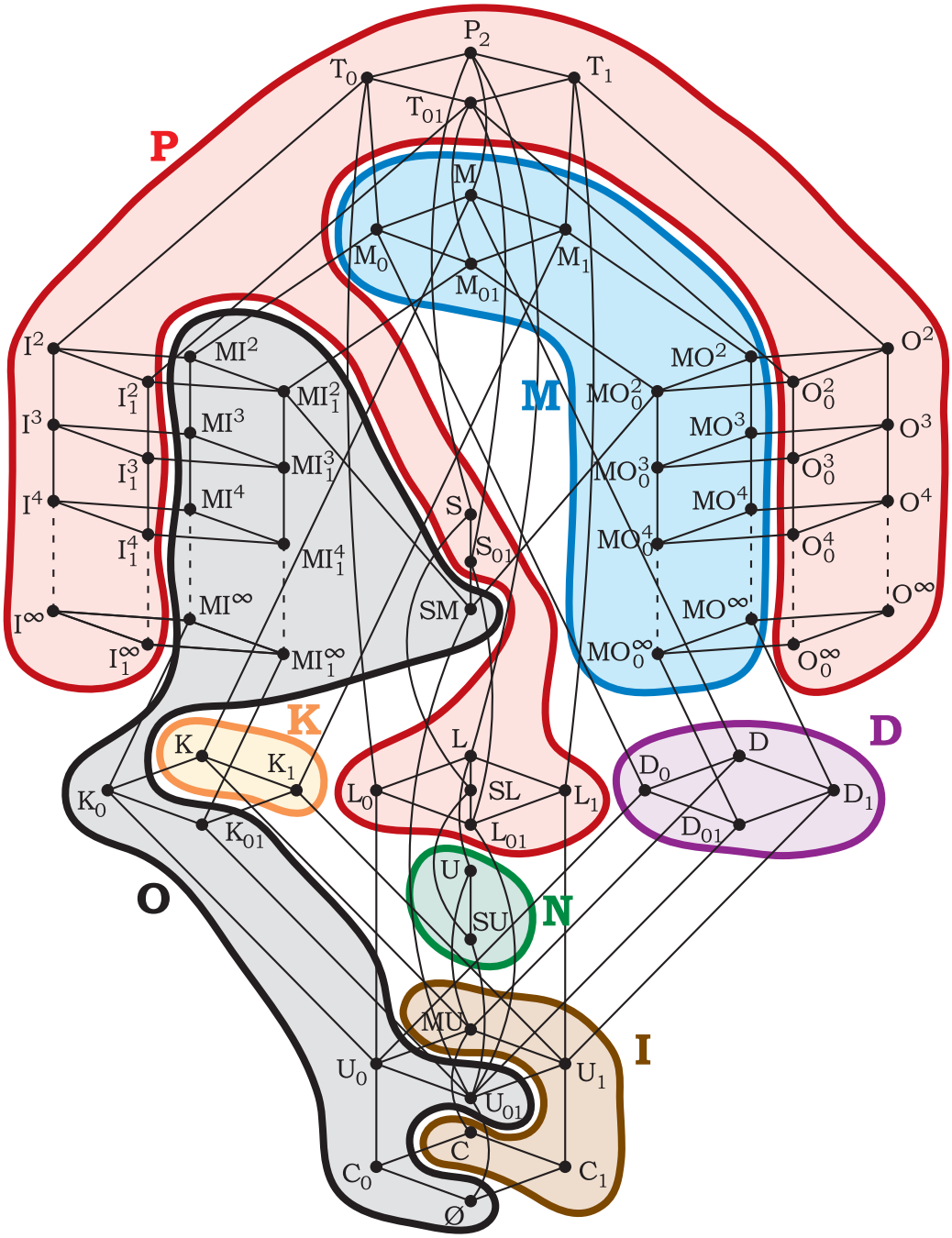

Figure 2: A variant of Post's lattice for language equations.

definition by string rewriting[1]. In the framework of language equations, this is the family of languages generated by the disjunction. Following the modern logical understanding of grammars developed by Rounds [44], this is a fragment of the FO(LFP) logic, whereas the definition of the entire FO(LFP) logic can be regarded as a far-going generalization of language equations.

A grammar is a quadruple $G = (\Sigma, N, R, S)$, where N is the set of nonterminal symbols or variables, $S \in N$ is the initial symbol, and every rule in R gives a possible representation of a nonterminal symbol as a concatenation.

$$X \to \alpha \qquad\qquad (X \in N,\ \alpha \in (\Sigma \cup N)^*)$$

Multiple rules for the same symbol on the left-hand side implicitly describe disjunction of syntactic conditions. The system of language equations corresponding to a grammar [8] has the following equation for each variable $X \in N$.

$$X = \bigcup_{X \to Y_1 \ldots Y_\ell \in R} Y_1 \cdot \ldots \cdot Y_\ell$$

Here each Y_i may be either a variable or a symbol from Σ; in the latter case, it is represented in the equation as a singleton constant language.

These equations use the basis comprised of single Boolean function, the disjunction, which generates Post's class D_{01}. Adding constants 0 and 1 to this basis does not increase the expressive power of language equations.

Lemma 1. *Let $\mathcal{F}$ be a class of Boolean functions, with its closure contained within the following bounds.*

$$[x \vee y] \subseteq [\mathcal{F}] \subseteq [x \vee y, 0, 1]$$

Then, for every alphabet Σ, the family of languages definable by unique solutions of systems (*), with Boolean operations from $\mathcal{F}$, concatenation and singleton constants, is exactly the family described by ordinary (context-free) grammars over Σ.

The four Post's classes satisfying these conditions are D, D_0, D_1, and D_{01}. They are marked in Figure 2 by the letter D.

Proof. Assume that the disjunction can be expressed in the basis $\mathcal{F}$. For every language described by some grammar, consider a grammar $G =$

[1] Actually, Chomsky's term "context-free" has no meaning outside of the definition by string rewriting, and does not characterize these grammars in relation to other currently used grammar models. To a modern reader, these grammars would rather be called *ordinary grammars*, because of their central position in the theory.

(Σ, N, R, S) in the Greibach normal form describing that language—that is, with all rules of the following form.

$$X \to aY_1 \dots Y_\ell \qquad (a \in \Sigma,\ \ell \geqslant 0,\ X, Y_1, \dots, Y_\ell \in N)$$

Accordingly, every concatenation in the corresponding system of language equations involves a singleton constant language $\{a\}$.

$$X = \bigcup_{X \to aY_1 \dots Y_\ell \in R} \{a\} \cdot Y_1 \cdot \ldots \cdot Y_\ell$$

Autebert et al. [1] called such systems *strict*, and showed that every such system has a unique solution. This system can be rewritten in the basis $\mathcal{F}$ by replacing each union operation with the expression for disjunction in $\mathcal{F}$. Thus, the language generated by the grammar is representable by a unique solution of a system over the basis $\mathcal{F}$.

Conversely, if a language L is defined by a unique solution of a system over the basis $\mathcal{F} = [x \vee y, 0, 1]$, then let the system first be transformed to use only the basis functions $x \vee y$, 0 and 1. Then, every occurrence of constant 0 can be expressed by the equation $X = \{a\}X$, for any symbol $a \in \Sigma$, which has a unique solution $X = \varnothing$. Expressing constant 1 means describing the language of all strings by an equation: if $\Sigma = \{a_1, \dots, a_k\}$, this is done by the following equation with a unique solution $X = \Sigma^*$.

$$X = \{a_1\}X \cup \ldots \cup \{a_k\}X \cup \{\varepsilon\}$$

By Proposition 1, the resulting system still has a unique solution, with L among its components, and it uses only union and concatenation. It remains to decompose complex right-hand sides to obtain a system with equations of the form $X = Y \cup Z$, $X = YZ$ and $X = \{w\}$, which can be directly translated to a grammar generating L. □

M: Disjunction and conjunction

Equations with disjunction and conjunction correspond to another family of formal grammars: the *conjunctive grammars* [26, 34].

A conjunctive grammar is a quadruple $G = (\Sigma, N, R, S)$, where N is the set of nonterminal symbols (as in ordinary grammars), $S \in N$ is the initial symbol, and each rule in R defines a representation of a nonterminal symbol as a conjunction of concatenations.

$$X \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_m \qquad (X \in N,\ m \geqslant 0,\ \alpha_1, \dots, \alpha_m \in (\Sigma \cup N)^*)$$

These grammars can define such an important syntactic construct as *declaration before use* [34, Ex. 3], as well as quite a few interesting abstract

languages, including $\{a^n b^n c^n \mid n \geqslant 0\}$, $\{wcw \mid w \in \{a,b\}^*\}$ [26, 34], $\{(wc)^{|w|} \mid w \in \{a,b\}^*\}$ and $\{a^{2^n} \mid n \geqslant 0\}$ [9].

The semantics of conjunctive grammars can be equivalently defined by a certain kind of term rewriting [26, 34] and by language equations [27], where the equation for each variable $X \in N$ is of the following form.

$$X = \bigcup_{X \to Y_{1,1} \ldots Y_{1,\ell_1} \,\&\, \ldots \,\&\, Y_{m,1} \ldots Y_{m,\ell_m} \in R} \bigcap_{i=1}^{m} Y_{i,1} \cdot \ldots \cdot Y_{i,\ell_i}$$

Note that some $Y_{i,j}$ may be symbols from Σ, in which case they represent the corresponding singleton constant languages.

Post's class corresponding to these equations $M_{01} = [x \vee y, x \wedge y]$. However, some other classes generate the very same language family.

Lemma 2. *Let $\mathcal{F}$ be a class of Boolean functions, with its closure contained within the following bounds.*

$$[x \vee (y \wedge z)] \subseteq [\mathcal{F}] \subseteq [x \vee y,\ x \wedge y,\ 0,\ 1]$$

Then, for every alphabet Σ, the family of languages definable by unique solutions of systems (*), with Boolean operations from $\mathcal{F}$, concatenation and singleton constants, is exactly the family described by conjunctive grammars over Σ.

The upper bound is given by Post's class $M = [x \vee y, x \wedge y, 0, 1]$, which contains all monotone Boolean functions. The lower bound is $MO_0^\infty = [x \vee (y \wedge z)]$. Two of the eight Post's infinite hierarchies are located between these classes, and, with respect to language equations, they collapse as shown in Figure 2, marked with the letter M.

Proof. Assume that the ternary function $f(x,y,z) = x \vee (y \wedge z)$ can be expressed in $\mathcal{F}$. Let a language L be described by a conjunctive grammar. Then, as shown by Okhotin and Reitwießner [36], it can be defined by a conjunctive grammar $G = (\Sigma, N, R, S)$ in the so-called *odd normal form*, with all rules of the following form.

$$\begin{aligned}
&X \to Y_1 a_1 Z_1 \,\&\, \ldots \,\&\, Y_m a_m Z_m && (m \geqslant 1,\ X, Y_i, Z_i \in N,\ a_i \in \Sigma)\\
&X \to a && (a \in \Sigma)\\
&S \to aX && (a \in \Sigma,\ X \in N)\\
&S \to \varepsilon
\end{aligned}$$

(the latter two types of rules are allowed only if S never occurs in the right-hand sides of any rules) Like in Lemma 1, the corresponding system of language equations is *strict*, in the sense that every concatenation involves a singleton constant language, and therefore the solution is unique.

This is a system over conjunction and disjunction, and it remains to express these operations through the function $f(x, y, z) = x \vee (y \wedge z)$. The disjunction can be expressed on the level of Boolean functions by identifying y and z. Zero can be expressed as usual, using the equation $X = \{a\}X$. Then, substituting this zero as x in f expresses the conjunction. The system is then converted to the desired basis by substituting each occurrence of f with its expressions in $\mathcal{F}$. The resulting system over $\mathcal{F}$ represents L by its unique solution.

In the other direction, let a language L be defined by a unique solution of a system using Boolean functions in $\mathcal{F} = [x \vee y,\, x \wedge y,\, 0,\, 1]$. First, the system is transformed to use only the basis functions $x \vee y$, $x \wedge y$, 0 and 1. Every occurrence of constants 0 and 1 is then expressed as in the proof of Lemma 1, using disjunction in the equation for constant 1. Finally, complex right-hand sides are decomposed, so that only equations $X = Y \cup Z$, $X = Y \cap Z$, $X = YZ$ and $X = \{w\}$ are left. Then, the corresponding conjunctive grammar generating L uses the rules $X \to Y$, $X \to Z$, $X \to Y \,\&\, Z$, $X \to YZ$ and $X \to w$. □

A survey of conjunctive grammars, their known properties and their open problems has recently appeared [34].

P: All Boolean operations

Language equations with all Boolean operations, that is, over the basis $P_2 = [x \vee y,\, \neg x]$, were first investigated by the author [28], with the original intention to use them in the definition of formal grammars with a negation operator: *Boolean grammars*. However, it turned out that these equations can represent logical dependence of a shorter string upon a longer one, such as in the following system of two equations.

$$\begin{array}{lcl} X & = & \overline{X} \cap \{a\}Y \\ Y & = & Y \end{array}$$

The system has a unique solution $X = Y = \varnothing$, because if any string w is in Y, then the first equation expresses a contradiction of the form "$aw \in X$ *if and only if* $aw \notin X$". However, in order to determine that contradiction for w, one has to consider a longer string aw, contrary to the intuition behind grammars. For that reason, the definition of Boolean grammars, given by Okhotin [29] and improved by Kountouriotis et al. [17], has to use specially modified language equations, which are beyond the scope of the present paper.

What is theoretically important about this dependence of shorter strings on longer strings, is that such dependencies can be used to express every recursive set—that is, a set recognized by a Turing machine that halts on

every input—by a unique solution of a system of language equations (*) with concatenation and all Boolean operations [28, 31, 32]. What is actually important about these equations is the possibility of using intersection and complementation to express containment of one arbitrary expression in another. For example, an inequality $XY \subseteq UV$ can be expressed by the following equation for an auxiliary variable Z, which turns into a contradiction if one concatenation is not a subset of the other.

$$Z = \overline{Z} \cap XY \cap \overline{UV}$$

An inclusion of this kind can be used to extract the language recognized by a Turing machine from the language of its computation histories by a language equation [28, 30, 32], and if the Turing machine indeed halts on every input, that equation will have a unique solution. A converse result, that unique solutions of language equations are always vectors of recursive sets, has also been established [28, 32]. For details, an interested reader is directed to the cited papers.

As long as the alphabet Σ contains at least two symbols, this computational universality construction can use the same alphabet to represent both a Turing machine's input string and its computation history, and concatenate them in the way that they could be separated from each other [28, 30, 32]. Later, Jeż and Okhotin [15] re-implemented this construction over a one-symbol alphabet $\Sigma = \{a\}$, that is, without using any auxiliary symbols to encode computation histories. This was done by representing both the input string and the computation history as sequences of digits in some base-k positional notation, and by manipulating unary representations of these numbers, using the tools developed for conjunctive grammars by Jeż [9] and by Jeż and Okhotin [11, 13].

As a small addendum to these results, the construction, both in its unary and non-unary cases, was adapted to use resolved systems (*) with the operation of symmetric difference of sets [33], which corresponds to Boolean *exclusive OR*, also known as *sum modulo two*, $x \oplus y$.

The following theorem presents all classes of Boolean operations, for which resolved language equations are already known to be computationally universal.

Theorem A (Okhotin [28, 32]; Jeż and Okhotin [15]; Okhotin [33])**.** *For every finite alphabet Σ and for every language $L \subseteq \Sigma$ over that alphabet, there exists a system of language equations (*) over the same alphabet Σ, using the operations of union, intersection, complementation and concatenation, which has a unique solution with L as one of its components.*

The result still holds true if the only allowed Boolean operation is symmetric difference.

Now the task is to determine the minimal classes of Boolean functions necessary and sufficient to implement this construction.

Lemma 3. *Let $\mathcal{F}$ be such a class of Boolean functions, that one of the three functions $x \vee (y \wedge \neg z)$, $x \wedge (y \vee \neg z)$, or $x \oplus y \oplus z$ is in its closure $[\mathcal{F}]$. Then, for every alphabet Σ, the family of languages definable by unique solutions of systems (*) over Σ, with Boolean operations from $\mathcal{F}$, concatenation and singleton constants, is exactly the family of recursive sets over Σ.*

These three functions generate Post's classes O_0^∞, I_1^∞ and L_{01}, respectively. Their upward closure is shown in Figure 2, in the area marked with P, and it covers four infinite hierarchies and a number of individual classes.

Proof. It is known that equations with any Boolean operations can define only recursive sets by their unique solutions [28, 32]. The rest of the proof shows that either of the bases O_0^∞ and I_1^∞ is sufficient to express all Boolean operations in language equations, whereas the base L_{01} can express sum modulo 2 of two arguments. Thus the representations of any recursive set given in Theorem A shall be adapted for the available bases.

$\boxed{O_0^\infty}$ Assume that the function $f(x, y, z) = x \vee (y \wedge \neg z)$ is representable in the basis $\mathcal{F}$. Consider a system as in Theorem A that defines a recursive set $L \subseteq \Sigma^*$ using disjunction and negation (conjunction could be eliminated through de Morgan laws). First, one can express the disjunction through f by identifying x and z, that is, as $f(x, y, x) = x \vee y$. Next, the negation is expressed as $f(0, 1, z) = 0 \vee (1 \wedge \neg z)$, where constants 0 and 1 are defined by the usual language equations, employing the disjunction to describe constant 1.

$$\begin{aligned} X &= \{a\}X \\ Y &= \{a_1\}Y \cup \ldots \cup \{a_k\}Y \cup \{\varepsilon\} \qquad\qquad (\Sigma = \{a_1, \ldots, a_k\}) \end{aligned}$$

Finally, f is expressed in the given basis $\mathcal{F}$. According to Proposition 1, the resulting system has a unique solution, with L as one of its components.

$\boxed{I_1^\infty}$ This time, let $g(x, y, z) = x \wedge (y \vee \neg z)$ be representable in $\mathcal{F}$, and consider a system with conjunction, disjunction and negation defining a recursive set $L \subseteq \Sigma^*$ by its unique solution. The first step is to obtain the zero by a language equation for a new variable V, and to substitute it into f as $g(x, 0, z) = x \wedge \neg z$. Using the latter function, constant 1 is expressed through a specially constructed system of language equations. If the alphabet is $\Sigma = \{a_1, \ldots, a_k\}$, the system is comprised of the following $2n + 3$

equations.

$$
\begin{aligned}
X &= X \\
Y_i &= \underbrace{X \cap \overline{X\{a_i\}}}_{g(X,V,X\{a_i\})} && (1 \leqslant i \leqslant k) \\
T_i &= \underbrace{Y_i \cap \overline{T_i}}_{g(Y_i,V,T_i)} && (1 \leqslant i \leqslant k) \\
Z &= \underbrace{\varepsilon \cap \overline{X}}_{g(\varepsilon,V,X)} \\
U &= \underbrace{Z \cap \overline{U}}_{g(Z,V,U)}
\end{aligned}
$$

It is claimed that the unique solution of this system is $X = \Sigma^*$, $Y_i = T_i = \varnothing$, $Z = U = \varnothing$. The task is to show that every string $w \in \Sigma^*$ must be in X, which is proved by induction on the length of w.

Base case, $w = \varepsilon$. Assume that ε is not in X. Then, it belongs to Z, which turns the equation for U into a contradiction of the form "the empty string is in U if and only if it not in U".

Induction step: w to wa_i. Let a string $w \in \Sigma^*$ be in X, and let $a_i \in \Sigma$ be the next symbol. To see that wa_i must be in X as well, suppose it is not. Then, wa_i belongs to the intersection $X \cap \overline{X\{a_i\}}$, and therefore to Y_i. Like in the base case, the equation for T_i becomes a contradiction on the string wa_i.

If $X = \Sigma^*$, then all remaining variables must be equal to the empty set.

With both zero and one defined, the negation can be expressed as $g(1, 0, z) = \neg z$, and the conjunction as $g(x, y, 1) = x \wedge y$. Finally, g is expressed in the given basis $\mathcal{F}$. Thus, the original system is transformed into a new one using g as the only Boolean operation, which describes the desired set L.

$\boxed{L_{01}}$ Let $h(x, y, z) = x \oplus y \oplus z$ be expressible in the basis $\mathcal{F}$. Then one can obtain the zero by an equation $Z = aZ$, and substitute it into h to obtain $h(x, y, 0) = x \oplus y$, that is, the symmetric difference of two languages. This allows a system provided by Theorem A to be transformed to the given basis. □

N: Negation only

Language equations (*) using concatenation and complementation, but no other Boolean operations, have first been considered by Leiss [24], who constructed an example of an equation over a unary alphabet with a non-regular unique solution.

Example 1 (Leiss [24]). *Let φ^2 abbreviate a concatenation $\varphi \cdot \varphi$. Then the following equation has a unique solution $\{a^n \mid \exists k \geqslant 0 : \ 2^{3k} \leqslant n < 2^{3k+2}\}$.*

$$X = \{a\} \cdot \overline{\overline{\overline{X}^2}}^2$$

Later, such equations over arbitrary alphabets were studied by Okhotin and Yakimova [39, 40], who determined that, even though negation is not monotone, these equations share the important property of equations with monotone Boolean operations, that the membership of longer strings in a solution cannot influence the membership of shorter strings [39, Lemma 3.4]. This property restores these equations back to the world of formal grammars as a special case of Boolean grammars [34].

Although these equations can describe such a non-trivial language as the one in Example 1, their limitations are substantial, and some simple languages cannot be defined. It was shown that the regular language $a\Sigma^*b \cup b\Sigma^*a \cup \{\varepsilon\}$ cannot be represented by such equations [40, Ex. 6.3]. Even if all regular languages are allowed in equations as constants, the language $(a\Sigma^*b \cup b\Sigma^*a \cup \{\varepsilon\}) \setminus \{a^n b^n \mid n > 1\}$ is still not representable [40, Ex. 7.2]. For unary languages, a language defined as a symmetric difference $L_1 \triangle L_2 \triangle L_3$, where $L_1 = \{a^n \mid \exists k \geqslant 0 : \ 2^{3k} \leqslant n < 2^{3k+2}\}$, $L_2 = a(a^2)^*$ and $L_3 = \{a^n, a^{n+1} \mid n = 2^{3k+1}, \text{ for some } k \geqslant 0\}$, cannot be defined by these equations [40, Ex. 7.2], even though it can be described by a conjunctive grammar [40, Prop. 7.4].

The complementation operation is a basis for Post's class SU of all self-dual unary functions. The following lemma states that the class of all unary functions $U = [0, \neg x]$, when used in language equations, induces the same family of languages. These two classes are marked in Figure 2 with the letter N, which stands for "negation".

Lemma 4. *Let $\mathcal{F}$ be such a class of Boolean functions, that $[\mathcal{F}] = [\neg x]$ or $[\mathcal{F}] = [0, \neg x]$. Then unique solutions of systems (*) with Boolean operations from $\mathcal{F}$, concatenation and singleton constants define the same single class of languages.*

The proof is by the same constructions as in the previous results, based on the obvious representation of the zero by an equation $X = aX$.

O: No Boolean operations

The remaining Post's classes induce three families of language equations that can be regarded as trivial, as they define small subclasses of regular languages. Nevertheless, they need to be investigated—first of all, to make sure that there is nothing of interest there.

The first trivial family O corresponds to equations in which no Boolean operations are allowed. It is easy to see that their unique solutions contain

only singleton languages and empty sets. As in the previous cases, nothing more can be generated using operations from a certain larger Post's class.

Lemma 5. *Let $\mathcal{F}$ be a class of Boolean functions contained within Post's class MI^2.*

$$\mathcal{F} \subseteq [0,\ \underbrace{(x \vee y) \wedge (y \vee z) \wedge (x \vee z)}_{d_3^*(x,y,z)}]$$

Then, unique solutions of systems (*) over an alphabet Σ, with Boolean operations from $\mathcal{F}$, concatenation and singleton constants define only singleton languages and the empty set.

As indicated by Post's lattice, MI^2 is the largest class of monotone Boolean functions in which neither disjunction nor constant 1 are expressible. This class properly contains two of the infinite hierarchies in the lattice, which are marked in Figure 2 by the letter O.

In order to prove that no other languages can be defined, consider the standard representation of the least solution of a system of equations (*) with only monotone operations in its right-hand sides [1, 27]. Least solutions are defined with respect to the partial order by componentwise inclusion: $(K_1, \ldots, K_n) \sqsubseteq (L_1, \ldots, L_n)$ if $K_i \subseteq L_i$ for all i. Then the least solution is obtained by a so-called *fixpoint iteration*, that is, by taking a vector of empty sets and iteratively transforming it by applying the right-hand sides of the system as a vector function. The least upper bound of the resulting sequence is the least solution.

Let φ be the right-hand sides of the system, regarded as a function mapping vectors of n languages to vectors of n languages. Then the least solution has the following form.

$$\bigsqcup_{k \geqslant 0} \varphi^k(\varnothing, \ldots, \varnothing) \tag{1}$$

Proof of Lemma 5. Consider a resolved system with concatenation and operations from MI^2 that defines a language L by its unique solution. First, all Boolean operations are expressed through d_3^*, and, for simplicity, all equations are decomposed to individual operations, so that each equation is of one of the following forms.

$$\begin{aligned} &U = XY \\ &U = (X \cup Y) \cap (Y \cup Z) \cap (X \cap Z) \\ &U = \{w\} && (w \in \Sigma^*) \end{aligned}$$

The resulting system has a unique solution with L among its components. Note that a unique solution is also the least. The following analysis of fixpoint iteration shows that all components of the least solution, including L, may contain at most one element.

Claim 5.1. *Consider a system of equations (*), where the equation for each variable U is of one of the above forms. Then, for every $k \geqslant 0$, each component of the vector $\varphi^k(\varnothing, \ldots, \varnothing)$ has cardinality at most one.*

The claim is proved by induction on k. In the base case, $k = 0$, the claim holds for the vector of empty sets. For the induction step, assume that at the k-th step, each U-component, denoted by $U^{(k)}$, satisfies $|U^{(k)}| \leqslant 1$, and consider the cardinality of each U at the next step.

If the equation is $U = \{w\}$, the claim holds true.

In the case of an equation $U = XY$, the value of U at the $(k+1)$-th step is obtained by concatenating the values of X and Y at the previous step.

$$|U^{(k+1)}| = |X^{(k)}Y^{(k)}| = |X^{(k)}| \cdot |Y^{(k)}| \leqslant 1 \cdot 1 = 1$$

Let the equation be $U = (X \cup Y) \cap (Y \cup Z) \cap (X \cap Z)$, so that $U^{(k+1)} = (X^{(k)} \cup Y^{(k)}) \cap (Y^{(k)} \cup Z^{(k)}) \cap (X^{(k)} \cup Z^{(k)})$. Since, by the induction hypothesis, each of the sets $X^{(k)}$, $Y^{(k)}$ and $Z^{(k)}$ contains at most one element, assume that $X^{(k)} = \{u\}$, $Y^{(k)} = \{v\}$ and $Z^{(k)} = \{w\}$, for some strings $u, v, w \in \Sigma^*$. If these strings are not pairwise distinct, then at least one of the unions $X^{(k)} \cup Y^{(k)}$, $Y^{(k)} \cup Z^{(k)}$ and $X^{(k)} \cup Z^{(k)}$ must be a singleton, in which case their intersection cannot contain more than one element. Otherwise, if u, v and w are pairwise distinct, then each of the three unions is of cardinality 2, but their intersection is empty. If any of $X^{(k)}$, $Y^{(k)}$ or $Z^{(k)}$ are empty sets rather than singletons, the value of the entire expression can only be reduced, and therefore is at most a singleton.

This completes the proof of Claim 5.1, as well as of Lemma 5. □

I: Constant 1 only

The second trivial case of language equations is given by a basis containing constant 1 and no other Boolean functions: the corresponding Post's class is C_1. Constant 1 can be used to express the language Σ^*, which results in a somewhat larger family of languages than O. The new family has the following characterization.

Lemma 6. *Let $\mathcal{F}$ be a class of Boolean functions contained within the following bounds.*

$$[1] \subseteq \mathcal{F} \subseteq [0, 1, x]$$

Then, unique solutions of systems (*) over the alphabet Σ, using Boolean operations from $\mathcal{F}$, concatenation and singleton constants, define languages of the form $\varnothing$ and $w_0\Sigma^*w_1\Sigma^* \ldots w_{m-1}\Sigma^*w_m$, with $m \geqslant 0$ and $w_i \in \Sigma^*$.

These bounds cover Post's classes C_1, C, U_1 and MU, as shown in Figure 2. The resulting family of languages is denoted by I, as to resemble the digit "1".

The proof of this lemma is again by analyzing the sequence converging to the least solution. Consider any resolved system of language equations in variables $X_1, \ldots, X_n$, with monotone operations in the right-hand sides. The infinite sequence $\varphi^k(\varnothing, \ldots, \varnothing)$ may be regarded as a computation, which uses n language variables: in the beginning, all variables are initialized to empty sets, and at every k-th step, as φ is applied, the value of each variable may change to any superset of its current value. If the Boolean operations used in the system are limited to constant 1 and intersection (Post's class K), then this process is known to have the *unique assignment property*: whenever a variable X gets assigned some non-empty value, it must maintain the same value at all subsequent steps[2].

Lemma B ([30, Lemma 9])**.** *Let Σ be an alphabet, and consider any resolved system of language equations (*), where the right-hand sides $\varphi = (\varphi_1, \ldots, \varphi_n)$ may use arbitrary constant languages and the operations of intersection and concatenation. Denote by $X^{(k)}$ the value of a variable X in the vector $\varphi^k(\varnothing, \ldots, \varnothing)$ obtained at the k-th iteration. Then, whenever $X^{(k)} \neq \varnothing$, all subsequent values $X^{(\ell)}$, with $\ell > k$, coincide with $X^{(k)}$.*

Accordingly, every such system of language equations degenerates to a *formula*, in which the values of the variables can be evaluated in the order implied by Lemma B. Using this property, the limitations of language equations stated in Lemma 6 are easy to establish.

Proof of Lemma 6. Given a system of language equations with concatenations and Boolean constants 0 and 1 as the only operations, the system is first transformed by replacing Boolean constants with language constants $\varnothing$ and Σ^*. For the resulting system, Lemma B implies that every component of its unique solution can be evaluated as a formula over constant languages $\varnothing$, Σ^* and $\{w\}$, with $w \in \Sigma^*$. Such formulae can express only languages of the form $w_0\Sigma^* w_1 \Sigma^* \ldots w_{m-1}\Sigma^* w_m$, as well as the empty set. □

Note that the family I is not closed under intersection, because the language $\Sigma^* a \Sigma^* \cap \Sigma^* b \Sigma^*$ is not representable in the form given in Lemma 6. The last family of language equations adds the conjunction operation to allow such languages to be represented.

K: Conjunction and constant 1

As is evident from Lemma 5, conjunction alone is not enough to define anything more than singletons. However, once Σ^* can be expressed, the intersection operation slightly increases the expressive power.

[2]Note that systems with Boolean operations from MI^2, described in Lemma 5, also have the unique assignment property. Indeed, due to the monotonicity of the sequence, the first assigned non-empty value $\{w\}$ can only be reassigned to a set containing at least two elements, which was proved to be impossible.

Lemma 7. *Let $\mathcal{F}$ be a class of Boolean functions contained within the following bounds.*

$$[x \wedge y,\ 1] \subseteq \mathcal{F} \subseteq [x \wedge y,\ 0,\ 1]$$

Then, unique solutions of systems (*) with Boolean operations from $\mathcal{F}$, concatenation and singleton constants define exactly the languages from the intersection and concatenation closure of I.

These are two Post's classes, K and K_1, marked in Figure 2 by the letter K. The proof of Lemma 7 uses Lemma B analogously to the proof of Lemma 6.

5 Summary

As evident from Figure 2, the above Lemmata 1–7 cover the entire Post's lattice. Hence, no families besides these seven families can be generated by language equations of the given kind, which leads to the following final result.

Theorem 1. *Let Σ be any finite alphabet, let $\mathcal{F} \subseteq P_2$ be a class of Boolean functions, and consider the family of languages representable as unique solutions of systems of language equations of the following form, with operations from $\mathcal{F}$, concatenation and singleton constant languages.*

$$\left\{ \begin{array}{lcl} X_1 & = & \varphi_1(X_1, \ldots, X_n) \\ & \vdots & \\ X_n & = & \varphi_n(X_1, \ldots, X_n) \end{array} \right. \qquad (*)$$

Then, depending on the basis $\mathcal{F}$, these equations define one of the following families of formal languages:

- *P. the recursive sets [28, 32, 15, 33], if $O_0^\infty \subseteq [\mathcal{F}]$, or $I_1^\infty \subseteq [\mathcal{F}]$, or $L_{01} \subseteq [\mathcal{F}]$;*
- *M. the languages described by conjunctive grammars [26], if $MO_0^\infty \subseteq [\mathcal{F}] \subseteq M$;*
- *D. the languages described by ordinary ("context-free") grammars [8, 1], if $D_{01} \subseteq [\mathcal{F}] \subseteq D$;*
- *N. a certain special subclass of Boolean grammars using negation only [24, 39, 40], if $SU \subseteq [\mathcal{F}] \subseteq U$;*
- *K. the intersection and concatenation closure of the class of languages of the form $w_0\Sigma^*w_1\Sigma^* \ldots w_{m-1}\Sigma^*w_m$, if $K_1 \subseteq [\mathcal{F}] \subseteq K$;*
- *I. all languages $w_0\Sigma^*w_1\Sigma^* \ldots w_{m-1}\Sigma^*w_m$, if $C_1 \subseteq [\mathcal{F}] \subseteq MU$;*

O. all singletons and the empty set, if $\mathcal{F} \subseteq MI^2$.

From Post's lattice, one can infer the following equivalent conditions on $\mathcal{F}$ that delimit the seven classes of language equations. First, equations (*) describe all *recursive sets* (P) if and only if the basis $\mathcal{F}$ contains a non-monotone function and a non-unary function (which may be the same function). Otherwise, there are two possibilities: either all functions in $\mathcal{F}$ are monotone, or all of them are unary.

If $\mathcal{F}$ contains only monotone functions, and as long as it generates the disjunction, these are formal grammars; they are separated into *conjunctive grammars* (M) and *ordinary grammars* (D) by the condition of whether the function $x \vee (y \wedge z)$ is expressible in $\mathcal{F}$. If there are only monotone functions in $\mathcal{F}$ and the disjunction cannot be expressed, this is one of the three subregular cases (O, I, K): their form given in the theorem depends on using only singleton constant languages[3].

In the remaining case, when $\mathcal{F}$ contains only unary functions and the negation is among them, the resulting language equations can be simulated by Boolean grammars [39]. Allowing all regular constants leads to a slightly larger family with similar properties [40].

Thus, of the seven classes of languages, six (M, D, N, K, I, O) are defined by special cases of Boolean grammars, and therefore can be recognized in polynomial time by the corresponding parsing algorithms [34]. More precisely, all these languages can be recognized in time $O(n^\omega)$ [35], with $\omega < 2.4$, that is, the complexity of multiplying a pair of $n \times n$ Boolean matrices.

The hierarchy formed by these seven families is illustrated in Figure 3, and formally established in the following theorem.

Theorem 2. *For every alphabet* Σ *containing at least two symbols, the seven classes of languages described in Theorem 1 are pairwise distinct, and are organized into the following two chains of proper inclusions; the regular languages (*Reg_Σ*) are inserted for reference.*

$$\boldsymbol{O_\Sigma} \subset \boldsymbol{I_\Sigma} \subset \boldsymbol{K_\Sigma} \subset Reg_\Sigma \subset \boldsymbol{D_\Sigma} \subset \boldsymbol{M_\Sigma} \subset \boldsymbol{P_\Sigma}$$
$$\boldsymbol{I_\Sigma} \subset \boldsymbol{N_\Sigma} \subset \boldsymbol{P_\Sigma}$$

Furthermore, D_Σ *is incomparable with* N_Σ*, and* M_Σ *is not contained in* N_Σ.

Proof. Let a and b be two distinct symbols in Σ. The strictness of these inclusions is witnessed by the following languages.

$$\Sigma^* \in I_\Sigma \setminus O_\Sigma$$
$$\Sigma^* a \Sigma^* \cap \Sigma^* b \Sigma^* \in K_\Sigma \setminus I_\Sigma$$

[3]For example, if all regular constant languages are allowed, then everything up to Post's class K will generate exactly the regular languages, according to Lemma B. The upper part of the O-area in Figure 2 will then collapse up to the conjunctive grammars.

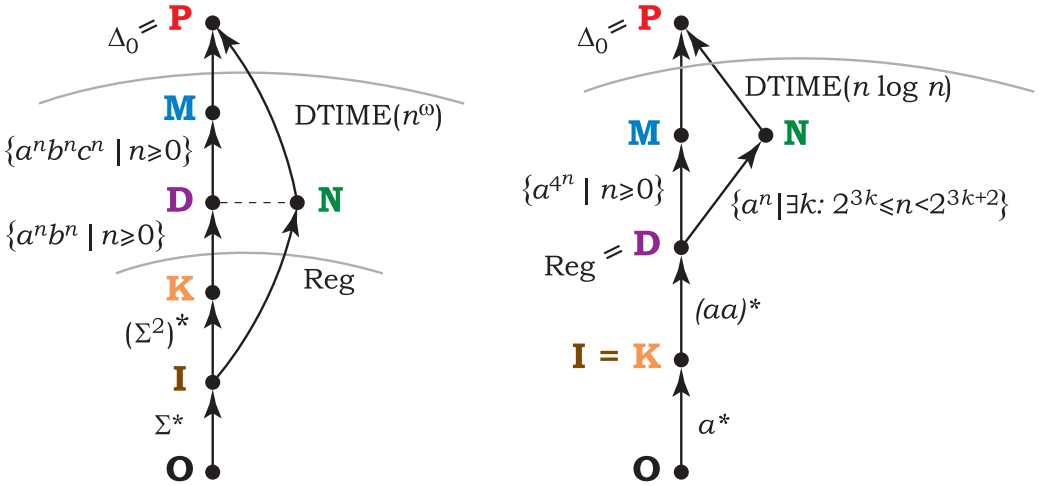

Figure 3: Hierarchy of language families defined by language equations over an alphabet Σ: (left) seven families for $|\Sigma| \geqslant 2$; (right) six families for $|\Sigma| = 1$.

The next separation uses the fact that K_Σ is contained in a special subclass of regular languages called *star-free languages*, which does not include the set of all strings of even length.

$$(\Sigma^2)^* \in Reg_\Sigma \setminus K_\Sigma$$

Separations between regular languages, ordinary grammars and conjunctive grammars are well-known.

$$\{a^n b^n \mid n \geqslant 0\} \in D_\Sigma \setminus Reg_\Sigma$$
$$\{a^n b^n c^n \mid n \geqslant 0\} \in M_\Sigma \setminus D_\Sigma$$

Even though no methods for proving languages to be non-representable by conjunctive grammars are currently known [34], they can be separated from all recursive sets by using the time hierarchy theorem from the complexity theory. Let L_3 be any set that can be recognized in time $O(n^3)$, but not in time $O(n^{2.9})$. Then, this set has no conjunctive grammar, and no Boolean grammar either [35], and therefore it separates both M_Σ and N_Σ from the recursive sets.

$$L_3 \in P_\Sigma \setminus M_\Sigma$$
$$L_3 \in P_\Sigma \setminus N_\Sigma$$

To show that N_Σ is a proper subset of I_Σ, it is sufficient to generate any non-regular language, such as the following one [40, Ex. 4.3].

$$\{a^n w b^n \mid w = \varepsilon \text{ or } w \in \{b, c\}\Sigma^*\} \in N_\Sigma \setminus Reg_\Sigma$$

Finally, turning to the incomparability of D_Σ and N_Σ, the latter class is known not to contain the following regular language [40, Ex. 6.3].

$$a\Sigma^*b \cup b\Sigma^*a \cup \{\varepsilon\} \in Reg \setminus N_\Sigma$$

A language in N_Σ that is not in D_Σ can be obtained by taking Example 1 as a system of equations over the alphabet Σ: the resulting language will have a non-regular intersection with a^*, and therefore is not in D_Σ. □

Note that Theorem 2 does not completely describe the structure of inclusions between these seven classes. For instance, K could be a subset of N, whereas N could be a subset of M.

A similar hierarchy can be established for a one-symbol alphabet.

Theorem 3. *Let $\Sigma = \{a\}$ be a unary alphabet. Then, $D_{\{a\}}$ is the class of all regular languages, whereas $I_{\{a\}}$ coincides with $K_{\{a\}}$ and contains all languages $\varnothing$, $\{a^n\}$ and a^na^*, with $n \geqslant 0$. These families are pairwise distinct and form the following proper inclusions.*

$$\boldsymbol{O_{\{a\}}} \subset \boldsymbol{I_{\{a\}}} = \boldsymbol{K_{\{a\}}} \subset Reg_{\{a\}} = \boldsymbol{D_{\{a\}}} \subset \boldsymbol{M_{\{a\}}} \subset \boldsymbol{P_{\{a\}}}$$
$$\boldsymbol{D_{\{a\}}} \subset \boldsymbol{N_{\{a\}}} \subset \boldsymbol{P_{\{a\}}}$$

In addition, $M_{\{a\}}$ is not a subset of $N_{\{a\}}$.

Proof. The characterization of $I_{\{a\}}$ is given by Lemma 6, with $\Sigma = \{a\}$. This class of languages is closed under intersection and concatenation, and therefore, by Lemma 7, it is the same as $K_{\{a\}}$. The equality of $D_{\{a\}}$ to the regular languages is a classical result by Ginsburg and Rice [8]. As proved by Okhotin and Yakimova [40, Thm. 5.2], all regular languages are in $N_{\{a\}}$; this inclusion is proper, because $N_{\{a\}}$ contains the non-regular language given in Example 1.

$$\{a^n \mid \exists k \geqslant 0 :\ 2^{3k} \leqslant n < 2^{3k+2}\} \in N_{\{a\}} \setminus D_{\{a\}}$$

The separation of $M_{\{a\}}$ from $D_{\{a\}}$ is a result by Jeż [9].

$$\{a^{4^n} \mid n \geqslant 0\} \in M_{\{a\}} \setminus D_{\{a\}}$$

The sets $M_{\{a\}}$ and $N_{\{a\}}$ are separated from $P_{\{a\}}$ (the recursive sets) by any unary language with a sufficiently high computational complexity.

A language in $M_{\{a\}}$ but not in $N_{\{a\}}$ was given by Okhotin and Yakimova [40, Example 7.2, Prop. 7.4]. □

It remains unknown whether $N_{\{a\}}$ a subset of $M_{\{a\}}$. Given the formidable expressive power of conjunctive grammars over a one-symbol alphabet [9, 11, 13], this does not look impossible.

6 Further work

The general form of equations studied in this paper was defined by several fixed parameters: the unknowns are *formal languages*, the equations are in *resolved form* $X_i = \varphi_i(X_1, \ldots, X_n)$, languages are defined by *unique solutions*, strings are combined using *concatenation*, and constant languages are *singletons*. On top of these, there can be an arbitrary set of Boolean operations, and this paper has investigated all possibilities here. Although, to a critical eye, the results might look as if "nothing of interest besides the previously studied cases has been found", in fact, early sketches of this paper (with the earliest one dating back to 2002) actually motivated the investigation of equations with complementation [39, 40] and with symmetric difference [33]. Applying the same method of study based upon Post's lattice to other types of language equations could similarly allow their interesting cases to be identified.

First, one could consider some special cases of the equations studied in this paper, such as those with concatenation restricted to be linear, so that one of its arguments is always a constant language. With union only, these equations correspond to the well-known *linear grammars*, and with disjunction and conjunction, they define *linear conjunctive grammars*, which are notable, in particular, for being equivalent to one-way real-time cellular automata [34, Sect. 4]. Using all Boolean operations, every recursive set can be represented if the alphabet contains at least two symbols [32]. For the symmetric difference operation, a computational universality construction in known only for regular constant languages, whereas nothing is known if all constant languages are singletons [33]; this might be a non-trivial and non-universal class. For complementation only, there are some unsystematic results on the expressive power [39, 40]. It remains to apply Post's lattice to systematize these cases and to obtain a hierarchy similar to the one presented in this paper.

A further restriction is to limit concatenation to *one-sided*, in which case unique solutions (as well as least and greatest solutions) are always regular, even if equations of the general ("unresolved") form $\varphi(X_1, \ldots X_n) = \psi(X_1, \ldots X_n)$ are allowed. This follows from Rabin's [43] regularity result for MSO logic. Since everything is regular, there is not much to study in terms of expressive power (besides determining which Post's classes are necessary to describe all regular sets). However, there are interesting computational complexity questions, such as what is the complexity of testing whether a given system has any solution, has a unique, least or greatest solution, etc. For several types of language equations with fixed sets of Boolean operations, such problems were researched by Baader and his colleagues [2, 3, 4, 5], and their results could be refined using Post's lattice.

Concerning least and greatest solutions, one can investigate their power in

the equations of the form studied in this paper. Equations with all Boolean operations are known to define exactly the recursively enumerable sets by least solutions, and their complements by greatest solutions [28, 32, 15, 33]. It remains to check the rest of the hierarchy, which will mostly be like the one in this paper, although there could be some variations.

Turning to more general models, language equations of the general form $\varphi = \psi$ have received much attention in the literature. Such equations are known to be computationally complete even without any Boolean operations, which was first shown by Kunc [19] for the equation $LX = XL$, where L is a finite constant language over a two-symbol alphabet. For a unary alphabet, Jeż and Okhotin [15, 10] established computational completeness of unresolved equations with only concatenation; this result was improved by Lehtinen and Okhotin [22, 23] using systems of two equations, $XXK = XXL$ and $XM = N$, with regular constant languages $K, L, M, N \subseteq a^*$. There were also some computational completeness results for inequalities $\varphi \subseteq \psi$ [18] and for inequations $\varphi \neq \psi$ [31]. Overall, for equations of this kind, a study of Boolean operations does not appear worthwhile.

On the other hand, language equations of the form $\varphi(X_1, \ldots, X_n) = C$, the prospects of applying Post's lattice look more promising. The computational complexity of some decision problems for such equations was determined by Bala [6] for the case of concatenation and union, and by Martens et al. [25] for equations with concatenation only. On the other hand, if the symmetric difference is allowed, then one can already express arbitrary equalities, and thus attain computational completeness. An analysis of Post's lattice is needed to enumerate all possibilities.

Another type of equations are those using other operations on strings instead of the concatenation. Equations of this kind were studied, in particular, by Kari [16] and by Domaratzki and Salomaa [7]. One possible subject for future work is to consider resolved systems exactly like in this paper, but with the concatenation replaced with the *shuffle operation.* Then, all computational completeness results for the unary case are directly inherited from the case of concatenation (because shuffle and concatenation are the same in the unary case), whereas for multiple-symbol alphabets, these equations will likely define some entirely different families of languages.

Using *erasing operations* on strings, such as quotients and homomorphisms, completely changes the expressive power of language equations. Jeż and Okhotin [12] investigated resolved equations with concatenation and quotient over a unary alphabet, characterizing their least and greatest solutions, showing that least solutions are computationally universal, whereas greatest solutions can represent complete sets for the bottom level of the analytical hierarchy. Unresolved equations characterize the hyper-arithmetical sets by their unique solutions [14, 21]. The effect of different Boolean operations in these equations, especially in the resolved ones, remains to be analyzed.

References

[1] J. Autebert, J. Berstel, L. Boasson, "Context-free languages and pushdown automata", in: Rozenberg, Salomaa (Eds.), *Handbook of Formal Languages*, Vol. 1, Springer-Verlag, 1997, 111–174.

[2] F. Baader, R. Küsters, "Unification in a description logic with transitive closure of roles", *Logic for Programming, Artificial Intelligence, and Reasoning* (LPAR 2001, Havana, Cuba, December 3–7, 2001), LNCS 2250, 217–232.

[3] F. Baader, P. Narendran, "Unification of concept terms in description logic", *Journal of Symbolic Computation*, 31:3 (2001), 277–305.

[4] F. Baader, A. Okhotin, "On language equations with one-sided concatenation", *Fundamenta Informaticae*, 126:1 (2013), 1–35.

[5] F. Baader, A. Okhotin, ‘‘Solving language equations and disequations with applications to disunification in description logics and monadic set constraints", *Logic for Programming, Artificial Intelligence, and Reasoning* (LPAR 2012, Mérida, Venezuela, March 11–15, 2012), LNCS 7180, 107–121.

[6] S. Bala, "Complexity of regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms", *Theory of Computing Systems*, 39:1 (2006), 137–163.

[7] M. Domaratzki, K. Salomaa, "Decidability of trajectory-based equations", *Theoretical Computer Science*, 345:2–3 (2005), 304–330.

[8] S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.

[9] A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615.

[10] A. Jeż, A. Okhotin, "Equations over sets of natural numbers with addition only", *26th Annual Symposium on Theoretical Aspects of Computer Science* (STACS 2009, Freiburg, Germany, 26–28 February 2009), Dagstuhl Seminar Proceedings 09001, 577–588.

[11] A. Jeż, A. Okhotin, "Conjunctive grammars over a unary alphabet: undecidability and unbounded growth", *Theory of Computing Systems*, 46:1 (2010), 27–58.

[12] A. Jeż, A. Okhotin, "Least and greatest solutions of equations over sets of integers", *Mathematical Foundations of Computer Science* (MFCS 2010, Brno, Czech Republic, 23–27 August 2010), LNCS 6281, 441–452.

[13] A. Jeż, A. Okhotin, "Complexity of equations over sets of natural numbers", *Theory of Computing Systems*, 48:2 (2011), 319–342.

[14] A. Jeż, A. Okhotin, "Representing hyper-arithmetical sets by equations over sets of integers", *Theory of Computing Systems*, 51:2 (2012), 196–228.

[15] A. Jeż, A. Okhotin, "Computational completeness of equations over sets of natural numbers", *Information and Computation*, 237 (2014), 56–94.

[16] L. Kari, "On language equations with invertible operations", *Theoretical Computer Science*, 132 (1994), 129–150.

[17] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, "Well-founded semantics for Boolean grammars", *Information and Computation*, 207:9 (2009), 945–967.

[18] M. Kunc, "On language inequalities $XK \subseteq LX$", *Developments in Language Theory* (DLT 2005, Palermo, Italy, 4–8 July 2005), LNCS 3572, 327–337.

[19] M. Kunc, "The power of commuting with finite sets of words", *Theory of Computing Systems*, 40:4 (2007), 521–551.

[20] D. Lau, *Function Algebras on Finite Sets: Basic Course on Many-Valued Logic and Clone Theory*, Springer, 2006.

[21] T. Lehtinen, "Equations $X + A = B$ and $(X + X) + C = (X - X) + D$ over sets of natural numbers", *Mathematical Foundations of Computer Science* (MFCS 2012, Bratislava, Slovakia, 27–31 August 2012), LNCS 7464, 615–629.

[22] 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.

[23] 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.

[24] E. L. Leiss, "Unrestricted complementation in language equations over a one-letter alphabet", *Theoretical Computer Science*, 132 (1994), 71–93.

[25] W. Martens, M. Niewerth, T. Schwentick, "Schema design for XML repositories: complexity and tractability", *PODS 2010* (Indianapolis, USA, 6–11 June 2010), 239–250.

[26] A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.

[27] A. Okhotin, "Conjunctive grammars and systems of language equations", *Programming and Computer Software*, 28 (2002), 243–249.

[28] A. Okhotin, "Decision problems for language equations with Boolean operations", *Automata, Languages and Programming* (ICALP 2003, Eindhoven, The Netherlands, June 30–July 4, 2003), LNCS 2719, 239–251.

[29] A. Okhotin, "Boolean grammars", *Information and Computation*, 194:1 (2004), 19–48.

[30] A. Okhotin, "Unresolved systems of language equations: expressive power and decision problems", *Theoretical Computer Science*, 349:3 (2005), 283–308.

[31] A. Okhotin, "Strict language inequalities and their decision problems", *Mathematical Foundations of Computer Science* (MFCS 2005, Gdańsk, Poland, August 29–September 2, 2005), LNCS 3618, 708–719.

[32] A. Okhotin, "Decision problems for language equations", *Journal of Computer and System Sciences*, 76:3–4 (2010), 251–266.

[33] A. Okhotin, "Language equations with symmetric difference", *Fundamenta Informaticae*, 116:1–4 (2012), 205–222.

[34] A. Okhotin, "Conjunctive and Boolean grammars: the true general case of the context-free grammars", *Computer Science Review*, 9 (2013), 27–59.

[35] A. Okhotin, "Parsing by matrix multiplication generalized to Boolean grammars", *Theoretical Computer Science*, 516 (2014), 101–120.

[36] A. Okhotin, C. Reitwießner, "Conjunctive grammars with restricted disjunction", *Theoretical Computer Science*, 411:26–28 (2010), 2559–2571.

[37] A. Okhotin, C. Reitwießner, "Parsing Boolean grammars over a one-letter alphabet using online convolution", *Theoretical Computer Science*, 457 (2012), 149–157.

[38] A. Okhotin, P. Rondogiannis, "On the expressive power of univariate equations over sets of natural numbers", *Information and Computation*, 212 (2012), 1–14.

[39] A. Okhotin, O. Yakimova, "Language equations with complementation: decision problems", *Theoretical Computer Science*, 376:1–2 (2007), 112–126.

[40] A. Okhotin, O. Yakimova, "Language equations with complementation: Expressive power", *Theoretical Computer Science*, 416 (2012), 71–86.

[41] E. L. Post, "Introduction to a general theory of elementary propositions", *American Journal of Mathematics*, 43:3 (1921), 163–185.

[42] E. L. Post, *The Two-Valued Iterative Systems of Mathematical Logic*, Princeton University Press, 1941.

[43] M. O. Rabin, "Decidability of second-order theories and automata on infinite trees", *Transactions of the American Mathematical Society*, 141 (1969), 1–35.

[44] W. C. Rounds, "LFP: A logic for linguistic descriptions and an analysis of its complexity", *Computational Linguistics*, 14:4 (1988), 1–9.

[45] S. V. Yablonski, G. P. Gavrilov, V. B. Kudryavtsev, *Funktsii algebry logiki i klassy Posta* (Functions of the logic algebra and the classes of Post), Nauka, Moscow, 1966, in Russian.

- German translation: *Boolesche Funktionen und Postsche Klassen*, Akademie-Verlag, Berlin, 1970.

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN 978-952-12-1963-4
ISSN 1239-1891