On the expressive power of GF(2)-grammars*

Vladislav Makarov and Alexander Okhotin

St. Petersburg State University, 7/9 Universitetskaya nab.,
Saint Petersburg 199034, Russia
vm450@yandex.ru, alexander.okhotin@spbu.ru

Abstract. GF(2)-grammars, recently introduced by Bakinova et al. ("Formal languages over GF(2)", LATA 2018), are a variant of ordinary context-free grammars, in which the disjunction is replaced by exclusive OR, whereas the classical concatenation is replaced by a new operation called GF(2)-concatenation: $K \odot L$ is the set of all strings with an odd number of partitions into a concatenation of a string in K and a string in L. This paper establishes several results on the family of languages defined by these grammars. Over the unary alphabet, GF(2)-grammars define exactly the 2-automatic sets. No language of the form $\{a^n b^{f(n)} \mid n \geqslant 1\}$, with uniformly superlinear f, can be described by any GF(2)-grammar. The family is not closed under union, intersection, classical concatenation and Kleene star, non-erasing homomorphisms. On the other hand, this family is closed under injective nondeterministic finite transductions, and contains a hardest language under reductions by homomorphisms.

1 Introduction

A new family of formal grammars, the *GF(2)-grammars*, was recently introduced by Bakinova et al. [2]. These grammars differ from the ordinary grammars (Chomsky's "context-free") as follows. In ordinary grammars, the operations are: the *disjunction* of syntactical conditions, expressed by multiple rules for the same nonterminal symbol, and the *concatenation* of languages, which is defined through conjunction and disjunction [17]. In GF(2)-grammars, these operations are modified by replacing the underlying Boolean logic with the GF(2) field. Accordingly, instead of set-theoretic union of languages, GF(2)-grammars feature *symmetric difference*, whereas concatenation of languages is replaced with a new operation called *GF(2)-concatenation*, defined as follows.

$$K \odot L = \{\, w \mid \text{the number of partitions } w = uv, \text{ with } u \in K \text{ and } v \in L, \text{ is odd} \,\}$$

GF(2)-grammars were introduced as a part of a general study of GF(2)-concatenation as an operation on formal languages. Their formal definition is based on parse trees in the corresponding ordinary grammar with classical concatenation and union: assuming that every string has a finite number of parse

* Supported by Russian Science Foundation, project 18-11-00100.

trees, the GF(2)-grammar defines all strings with an odd number of parse trees. The intuitive correctness of this definition is confirmed by a result that if a grammar is represented by a system of *language equations*, similar to the equations of Ginsburg and Rice [8], but using the operations of GF(2)-concatenation and symmetric difference, then the language defined by this GF(2)-grammar satisfies the system.

A few related, more general grammar models were studied before. Knuth [13] investigated specification of *multisets* by grammars, with every parse tree contributing an element to a multiset. A more general extension are formal languages over multiplicities, that is, mappings from the set of strings to a semiring. Under certain monotonicity assumptions on the semiring, equations in formal power series over a semiring behave similarly to ordinary grammars, and a substantial theory has been developed around them, see the survey by Petre and Salomaa [18]. However, the two-element field does not have the required monotonicity properties, and this general theory does not apply to GF(2)-grammars. Another matter is that languages over multiplicities are, after all, functions, and not languages as such. There are just two cases when actual languages are defined: this is when the semiring is either the Boolean semiring or the GF(2) field. Whereas the former case is classical, the other case deserves investigation.

The study of GF(2)-grammars is a part of the research on formal grammars with different sets of operations [17]. All these grammars are variants of Chomsky's "context-free" model, and particular models include *conjunctive grammars* equipped with a conjunction operator in the rules [15]; *multi-component grammars* [19] that allow substrings with gaps as basic constituents; *grammars with context operators* [3], and a few other models.

Every *unambiguous grammar* is a GF(2)-grammar, and it still defines the same language. In the presence of ambiguity, ordinary grammars assert the *existence* of a parse tree, whereas GF(2)-grammars check the *parity*. For this reason, ordinary grammars and GF(2)-grammars are two different generalizations of unambiguous grammars. These two generalizations share the same complexity upper bound: there is a basic cubic-time parsing algorithm, more efficient parsing by matrix multiplication, and parallel parsing in NC^2 [2]. These practically valuable properties make the class of GF(2)-grammars potentially useful and accordingly deserving further study.

There is some evidence that the formal properties of ordinary and GF(2)-grammars are not symmetric. First, unlike the classical concatenation, the GF(2)-concatenation is *invertible*: to be precise, for every language L containing the empty string, there exists a language L^{-1}, for which $L \odot L^{-1} = L^{-1} \odot L = \{\varepsilon\}$ [2]. How this property affects language specification, remains to be investigated. Second, over a unary alphabet, GF(2)-grammars can describe some non-regular sets, such as $\{\, a^{2^n} \mid n \geqslant 0 \,\}$. These differences make this family an interesting subject for theoretical research.

The goal of this paper is to investigate the family of GF(2)-grammars and to determine, which languages they can describe and which they cannot. These results shall be used to establish the basic closure properties of GF(2)-grammars.

as well as to compare their expressive power with that of the main families of formal grammars.

2 GF(2)-grammars

Syntactically, a GF(2)-grammar is defined exactly as an ordinary grammar, with a finite sequence of symbols and nonterminal symbols on the right-hand side of each rule. However, every such sequence has semantics of GF(2)-concatenation, whereas multiple rules for the same nonterminal symbol implicitly denote symmetric difference of the given conditions.

Definition 1 ([2]). *A GF(2)-grammar is a quadruple $G = (\Sigma, N, R, S)$, where:*

- *Σ is the alphabet of the language;*
- *N is the set of nonterminal symbols;*
- *every rule in R is of the form $A \to X_1 \odot \ldots \odot X_\ell$, with $\ell \geqslant 0$ and $X_1, \ldots X_\ell \in \Sigma \cup N$, which represents all strings that have an odd number of partitions into $w_1 \ldots w_\ell$, with each w_i representable as X_i;*
- *$S \in N$ is the initial symbol.*

The grammar must satisfy the following condition. Let $\widehat{G} = (\Sigma, N, \widehat{R}, S)$ be the corresponding ordinary grammar, with $\widehat{R} = \{\, A \to X_1 \ldots X_\ell \mid A \to X_1 \odot \ldots \odot X_\ell \in R \,\}$. It is assumed that, for every string $w \in \Sigma^*$, the number of parse trees of w in $\widehat{G}$ is finite; if this is not the case, then G is considered ill-formed.

Then, for each $A \in N$, the language $L_G(A)$ is defined as the set of all strings with an odd number of parse trees as A in $\widehat{G}$.

A grammar is GF(2)-linear*, if, in each rule, at most one of $X_1, \ldots, X_\ell$ is a nonterminal symbol.*

Theorem A ([2]) *Let $G = (\Sigma, N, R, S)$ be a GF(2)-grammar. Then the substitution $A = L_G(A)$ for all $A \in N$ is a solution of the following system of language equations.*

$$A = \bigwedge_{A \to X_1 \odot \ldots \odot X_\ell \in R} X_1 \odot \ldots \odot X_\ell \qquad (A \in N)$$

Multiple rules for the same nonterminal symbol can be denoted by separating the alternatives with the "sum modulo two" symbol ($\oplus$), as in the following example.

Example 2 ([2]). The following GF(2)-linear grammar defines the language $\{\, a^\ell b^m c^n \mid \ell = m \text{ or } m = n, \text{ but not both} \,\}$.

$$\begin{aligned} S &\to A \oplus C \\ A &\to aA \oplus B \\ B &\to bBc \oplus \varepsilon \\ C &\to Cc \oplus D \\ D &\to aDb \oplus \varepsilon \end{aligned}$$

Indeed, each string $a^\ell b^m c^n$ with $\ell = m$ or with $m = n$ has a parse tree, and if both equalities hold, then there are accordingly two parse trees, which cancel each other.

Since GF(2)-concatenation with a singleton language is the same as classical concatenation, GF(2)-linear grammars are a special case of *linear Boolean grammars*, in which the allowed operations are all Boolean operations and concatenation with singletons. In the latter grammars, negation can be eliminated, resulting in a *linear conjunctive grammar* [14].

Since linear conjunctive grammar over a unary alphabet define only regular languages, so do the GF(2)-linear grammars. On the other hand, GF(2)-grammars of the general form can define some non-regular unary languages.

Example 3 ([2]). The following grammar describes the language $\{ a^{2^n} \mid n \geqslant 0 \}$.

$$S \to (S \odot S) \oplus a$$

The main idea behind this grammar is that the GF(2)-square $S \odot S$ over a unary alphabet doubles the length of each string: $L \odot L = \{ a^{2\ell} \mid a^\ell \in L \}$. The grammar iterates this doubling to produce all powers of two.

3 GF(2)-grammars over the unary alphabet

Ordinary grammars over the unary alphabet $\Sigma = \{a\}$ define only regular languages [8]. On the other hand, as demonstrated by Example 3, GF(2)-grammars can define some non-regular languages. The question is, which unary languages can be defined? The answer follows from the famous Christol's theorem [5].

A few definitions are necessary.

Definition 4. *A set of natural numbers $S \subseteq \mathbb{N}$ is called* k-automatic *[1], if there is a finite automaton over the alphabet $\Sigma_k = \{0, 1, \ldots, k-1\}$ recognizing base-k representations of these numbers.*

Let $\mathbb{F}_k[t]$ be the ring of polynomials over the k-element field $\mathrm{GF}(k)$, and let $\mathbb{F}_k[[t]]$ denote the ring of formal power series over the same field.

Definition 5. *A formal power series $f \in \mathbb{F}_k[[t]]$ is said to be algebraic, if there exists a non-zero polynomial P with coefficients from $\mathbb{F}_k[t]$, such that $P(f) = 0$.*

Theorem B (Christol's theorem for GF(2)) *A formal power series $\sum_{n=0}^{\infty} f_n t^n \in \mathbb{F}_2[[t]]$ is algebraic if and only if the set $\{ n \in \mathbb{N}_0 \mid f_n = 1 \}$ is 2-automatic.*

For a unary alphabet, solutions of language equations corresponding to a GF(2)-grammar, as in Theorem A, are algebraic formal power series in $\mathbb{F}_2[[t]]$, which has the following consequence.

Corollary 6. *Every unary language defined by GF(2)-grammar is 2-automatic.*

Inferring the converse characterization from Christol's theorem is not trivial, it is easier to give a direct proof.

Theorem 7. *Every 2-automatic unary language is described by a GF(2)-grammar.*

Proof. Let $\mathcal{A} = (\{\mathtt{0},\mathtt{1}\}, Q, q_0, \delta, F)$ be a DFA that recognizes binary representations of natural numbers without leading zeroes. The corresponding GF(2)-grammar is defined as $G = (\{a\}, \{\, A_q \mid q \in Q \,\} \cup \{S\}, R, S)$, with the following set of rules.

$$\begin{aligned}
A_q &\to A_p \odot A_p && (p \in Q,\ \delta(p,\mathtt{0}) = q)\\
A_q &\to a \odot A_p \odot A_p && (p \in Q,\ \delta(p,\mathtt{1}) = q)\\
A_q &\to a && (\delta(q_0,\mathtt{1}) = q)\\
S &\to A_q && (q \in F)\\
S &\to \varepsilon && (\text{if } \mathtt{0} \in L(\mathcal{A}))
\end{aligned}$$

Here, as in Example 3 the rule for $A_q \to A_p \odot A_p$ produces all strings $a^{2\ell}$, with a^ℓ defined by A_p: this effectively appends zero to the binary representation. The rule $A_q \to A_p \odot A_p \odot a$ doubles the length and adds one, thus appending digit $\mathtt{1}$.

Then, $L(A_q)$ consists of all strings $a^{(\mathtt{1}w)_2}$, with $\delta(q_0, \mathtt{1}w) = q$. □

By the above, the unary languages defined by GF(2)-grammars are exactly the 2-automatic languages. This characterization also gives a tool for proving that a given language over a non-unary alphabet cannot be defined by any GF(2)-grammar.

Theorem 8 (Method of unary image). *Let a language L over an alphabet Σ be defined by a GF(2)-grammar, and let $h\colon \Sigma \to \{t\}^*$ be a non-erasing homomorphism that is injective on L, in the sense that $h(u) \neq h(v)$ for any distinct $u, v \in L$. Then, $h(L)$ is a 2-automatic language over the unary alphabet $\{t\}$.*

In the GF(2)-grammar for L, it is sufficient to replace every occurrence of every symbol $a \in \Sigma$ in the rules with $h(a)$. The resulting GF(2)-grammar defines the language $h(L)$, which is then 2-automatic by Corollary 6.

4 Representability of subsets of a^*b^*

Defining languages of the form $L \subseteq a^*b^*$ in a certain formalism represents its ability to *count*. A particular special case are languages of the form $L_f = \{\, a^n b^{f(n)} \mid n \geqslant 1 \,\}$, where f is a function $f\colon \mathbb{N} \to \mathbb{N}$.

Finite automata *cannot keep count*, in the sense that L_f is regular only if there is a partition of $\mathbb{N}$ into finitely many pairwise disjoint arithmetic progressions, including singletons, and for each arithmetic progression $\{\, m_0 + ip \mid i \geqslant 0 \,\}$, the language contains a subset $\{\, a^{m_0+ip} b^{n_0} \mid i \geqslant 0 \,\}$, for a fixed number n_0.

For ordinary grammars, the subset may be *linear*, that is, of the form $\{ a^{m_0+ip}b^{n_0+iq} \mid i \geqslant 0 \}$. Using linear conjunctive grammars, more sophisticated languages with exponential growth can be expressed.

Example 9 (Ibarra and Kim [11]). The language $\{ a^n b^{2^n} \mid n \geqslant 1 \}$ is recognized by a one-way real-time cellular automaton, and, equivalently, is described by a linear conjunctive grammar.

At the same time, there is the following bound on the growth of f.

Theorem C (Buchholz and Kutrib [4]) *For every function $f\colon \mathbb{N} \to \mathbb{N}$, if the language $\{ a^n b^{f(n)} \mid n \geqslant 1 \}$ is linear conjunctive, then f is bounded by an exponential function.*

Buchholz and Kutrib [4] further examined the ability to count for several classes of cellular automata. The question investigated in this paper is, what kind of languages of the form L_f can be expressed using GF(2)-grammars? The starting point is the following class of obviously representable languages.

Theorem 10. *Let $\mathbb{N}$ be represented as a disjoint union of finitely many 2-automatic sets: $\mathbb{N} = S_1 \uplus \ldots \uplus S_k$. For each of these sets, S_j, let L_j be a language of the following form: either $L_j = \{ a^m b^{n_0} \mid m \in S_j \}$, for some $n_0 \geqslant 0$, or, as long as S_j is an arithmetic progression $\{ m_0 + ip \mid i \geqslant 0 \}$ with $m_0 \geqslant 0$ and $p \geqslant 1$, a language $L_j = \{ a^{m_0+ip} b^{n_0+iq} \mid i \geqslant 0 \}$ with $n_0 \geqslant 0$ and $q \geqslant 1$. Then, the languages $L_1, \ldots, L_k$ are pairwise disjoint, their union is a language of the form L_f, and it can be described by a GF(2)-grammar.*

The current conjecture is that no other languages of the form L_f can be represented. The next theorem identifies a class of non-representable languages, which are all those with a superlinearly growing function f, under the following uniformness restriction.

Definition 11. *A function $f\colon \mathbb{N} \to \mathbb{N}$ is called* uniformly superlinear*, if, for every $c > 0$, there exists $N \in \mathbb{N}$, such that $f(n+1) - f(n) > c$ for all $n > N$; in other words, $f(n+1) - f(n)$ is eventually larger than any constant, or $\liminf\limits_{n \to +\infty} f(n+1) - f(n) = +\infty$.*

Theorem 12. *Let $f\colon \mathbb{N} \to \mathbb{N}$ be a monotonically increasing uniformly superlinear function. Then the language $\{ a^n b^{f(n)} \mid n \in \mathbb{N} \}$ is not described by any GF(2)-grammar.*

Proof. Proof by contradiction. Suppose that $L := \{ a^n b^{f(n)} \mid n \in \mathbb{N} \}$ is described by some GF(2)-grammar. Then $S_1 := \{ n + f(n) \mid n \in \mathbb{N} \}$ and $S_2 := \{ 2n + f(n) \mid n \in \mathbb{N} \}$ are both 2-automatic by virtue of being unary images of L under homomorphisms $a \to t, b \to t$ and $a \to t^2, b \to t$, respectively.

Let ℓ be an integer large enough, so that $2^{\ell/2}$ is greater than the number of states in the minimal NFAs recognizing both S_1 and S_2 in binary notation. By the uniform superlinearity of f, there exists a number M, such that $f(n+1) - f(n) > 2^\ell$ for all $n \geqslant M$. Consider the integers $M+1, M+2, \ldots, M+2^\ell$. Clearly, they all have different remainders modulo 2^ℓ.

Claim. For any function $f\colon \mathbb{N} \to \mathbb{N}$ and for any two numbers $\ell, M \in \mathbb{N}$, there exists a factor $k \in \{1,2\}$ and a set $X \subseteq \{M+1, M+2, \ldots, M+2^\ell\}$, such that $|X| = \lceil 2^{\ell/2} \rceil$ and all residues $kn + f(n)$ modulo 2^ℓ for $n \in X$ are distinct.

Proof (of the claim). The first observation is that the mapping $n \mapsto (n + f(n), 2n + f(n)) \pmod{2^\ell}$ is injective on $\{M+1, \ldots, M+2^\ell\}$. Indeed, if, for any two arguments $n, n' \in \{M+1, \ldots, M+2^\ell\}$ the values coincide, that is, $n + f(n) \equiv n' + f(n') \pmod{2^\ell}$ and $2n + f(n) \equiv 2n' + f(n') \pmod{2^\ell}$, then, subtracting the former equality from the latter yields $n \equiv n' \pmod{2^\ell}$, which implies that the arguments must be the same.

Now the statement is proved by contradiction. For $k = 1$, the assumption that no such set X exists means that there are fewer than $2^{\ell/2}$ distinct values $n + f(n)$ modulo 2^ℓ, for $n \in \{M+1, M+2, \ldots, M+2^\ell\}$. Similarly, for $k = 2$, by assumption, there are fewer than $2^{\ell/2}$ distinct values $2n + f(n)$ modulo 2^ℓ, for all $n \in \{M+1, M+2, \ldots, M+2^\ell\}$. Therefore, the number of distinct pairs $(n + f(n),\ 2n + f(n))$ modulo 2^ℓ, obtained for different n, is strictly less than $2^{\ell/2} \cdot 2^{\ell/2} = 2^\ell$. Since there are 2^ℓ different arguments n, the mapping $n \mapsto (n + f(n), 2n + f(n)) \pmod{2^\ell}$ cannot be injective, which contradicts the above observation. □

Resuming the proof of the theorem, by the lemma, there exist $k \in \{1,2\}$ and $X \subseteq \{M+1, M+2, \ldots, M+2^\ell\}$, such that all numbers $(kn + f(n))$, with $n \in X$, are pairwise distinct modulo 2^ℓ. Let $X := \{n_1, n_2, \ldots, n_{|X|}\}$. For each number n_i, the least significant ℓ digits in its binary representation are denoted by $v_i \in \{\mathtt{0},\mathtt{1}\}^\ell$; if n_i is less than $2^{\ell-1}$, the string is accordingly padded by zeroes. Similarly, let $u_i \in \{\mathtt{0},\mathtt{1}\}^*$ be the string of all remaining digits, so that $(u_i v_i)_2 = kn_i + f(n_i)$. By the choice of X, all v_i are different.

It is claimed that the set of $|X|$ pairs $(u_1, v_1), \ldots, (u_{|X|}, v_{|X|})$ forms a fooling set for the language of binary representations of S_k. Indeed, $(u_i v_i)_2 = (kn_i + f(n_i)) \in S_k$ for $i = 1, 2, \ldots, |X|$. On the other hand, for any i and j from $\{1, 2, \ldots, |X|\}$, such that $i \neq j$, at least one of the numbers $(u_i v_j)_2$ and $(u_j v_i)_2$ is not in S_k. By choice of X, $v_i \neq v_j$. Without loss of generality, let $(v_i)_2 < (v_j)_2$. For the sake of a contradiction, suppose that $(u_j v_i)_2 \in S_k$, that is, $(u_j v_i)_2 = km + f(m)$ for some $m \in \mathbb{N}$. On the one hand, $(u_j v_i)_2 < (u_j v_j)_2 < (u_j v_i)_2 + 2^\ell$, because $(u_j v_j)_2$ and $(u_j v_i)_2$ differ only in ℓ lowest bits, and $(v_i)_2 < (v_j)_2$. On the other hand, since $km + f(m) = (u_j v_i)_2 < (u_j v_j)_2 = kn_j + f(n_j)$ and f is non-decreasing, one can conclude that $m < n_j$ and $(u_j v_j)_2 - (u_j v_i)_2 = (kn_j + f(n_j)) - (km + f(m)) > f(n_j) - f(m) \geqslant f(n_j) - f(n_j - 1) > 2^\ell$, because $n_j - 1 \geqslant M$ and $f(n+1) - f(n) > 2^\ell$ for $n \geqslant M$. Together these facts yield $2^\ell > (u_j v_j)_2 - (u_j v_i)_2 > 2^\ell$, contradiction.

It has thus been proved that the language of binary representations of S_k. has a fooling set of size $2^{\ell/2}$, and therefore every NFA recognizing this language must have at least this many states. This contradicts the assumption that there is a smaller NFA for this language. □

Example 13 (cf. Example 9). The language $\{ a^n b^{2^n} \mid n \geqslant 1 \}$ is not described by any GF(2)-grammar, because the function $f(n) = 2^n$ is increasing and uniformly superlinear.

5 A separating example and the hierarchy

In order to compare the expressive power of GF(2)-grammars to other grammar families, it is essential to find a simple language which they could not represent, but other kinds of grammars could. Most of the results presented later on are based on the following language over the alphabet $\{a, b\}$.

$$L = \{ ba^{2 \cdot 3^n - 1} \dots ba^{17}\, ba^5\, ba\, bbb\, a^3 b\, a^{11} b\, a^{35} b \dots a^{4 \cdot 3^n - 1} b \mid n \geqslant 0 \}$$

Lemma 14. *The language L is representable as $L = L_1 \cap L_2$, where both L_1 and L_2 are described by unambiguous linear grammars.*

Furthermore, their complements $\overline{L_1}$ and $\overline{L_2}$ are described by linear grammars, and therefore so is the complement of L.

Proof (a sketch). This is a standard construction, inspired by a proof by Ginsburg and Spanier [9]. Each string in L encodes two sequences of numbers: 1, 5, 17, ..., $2 \cdot 3^n - 1$ on the left, and 3, 11, 35, ..., $4 \cdot 3^n - 1$ on the right. The language L_1 ensures that for each i-th element m on the left-hand side, the i-th element on the right-hand side must be $2m + 1$; the language L_2 similarly ensures that for each i-th element $2m - 1$ on the right-hand side, the $(i + 1)$-th element on the left-hand side must be $3m - 1$, and also that the first element of the left-hand-side sequence is 1.

Linear grammars for the complements of L_1 and of L_2 simply check that there is at least one error in the above correspondence. □

Lemma 15. *Neither L nor its complement are described by any GF(2)-grammars.*

Proof. Indeed, the unary image of L is $\{ a^{3^n} \mid n \geqslant 2 \}$, and the latter language is not described by any GF(2)-grammar by Christol's theorem. Since complementation is representable in GF(2)-grammars, there cannot be a grammar for the complement of L either. □

With this last example, the position of GF(2)-grammars in the hierarchy of grammars with different sets of operations can be determined as follows. The hierarchy in Figure 1 includes the following grammar families: ordinary grammars or Chomsky's context-free (union and concatenation: Ordinary); unambiguous grammars (disjoint union, unambiguous concatenation: Unamb); linear grammars (union, concatenation with symbols: Lin); unambiguous linear grammars (disjoint union, concatenation with symbols: UnambLin); linear conjunctive grammars (union, intersection, concatenation with symbols: LinConj); conjunctive grammars (union, intersection, concatenation: Conj).

The families are separated by the following examples.

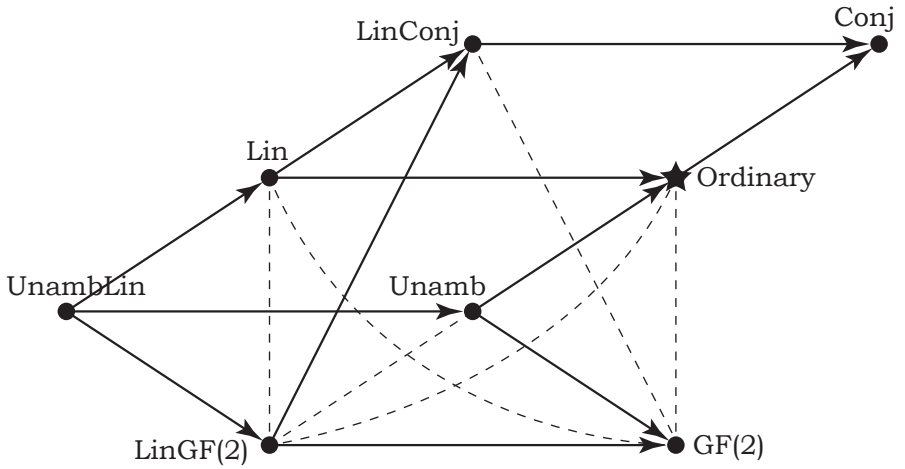

Fig. 1. The hierarchy of grammars: solid lines indicate proper inclusions, dashed lines mark incomparable families (shown only for GF(2)-families).

- GF(2)-linear, but not ordinary (and thus not linear and not unambiguous): $\{\, a^\ell b^m c^n \mid \ell = m \text{ or } m = n, \text{ but not both} \,\}$ (Example 2).
- GF(2), but not ordinary (and thus not unambiguous) and not linear conjunctive (and thus neither linear nor GF(2)-linear): $\{\, a^{2^n} \mid n \geqslant 0 \,\}$ (Example 3).
- Linear conjunctive but not GF(2) (and thus not GF(2)-linear): $\{\, a^n b^{2^n} \mid n \geqslant 1 \,\}$ (Example 13).
- Linear (and also ordinary), but not GF(2) (and thus not GF(2)-linear): the complement of L (Lemmata 14–15).
- Conjunctive, but not GF(2): the complement of L applies as well. Furthermore, non-containment is witnessed by the unary language $\{\, a^{3^n} \mid n \geqslant 0 \,\}$, which has a conjunctive grammar [12], but not a GF(2)-grammar (Corollary 6).
- Unambiguous, but not GF(2)-linear: a language defined by an unambiguous grammar, but not a linear conjunctive grammar, was constructed by Okhotin [16, Lemma 4] using a method of Terrier [20].

The comparison between GF(2)-grammars and conjunctive grammars remains incomplete, because no example of a language defined by a GF(2)-grammar, but not by any conjunctive grammar, is known. The conjectured example is $\{\, uv \mid u, v \in \{a, b\}^*, |u| = |v|, u \text{ and } v \text{ differ in an odd number of positions} \,\}$ [2]. No way of constructing a conjunctive grammar for this language is known; however, no proof of this could be given due to the general lack of knowledge on conjunctive grammars [15].

6 Closure properties

Some closure properties of GF(2)-grammars are quite expected, and follow by well-known arguments. Such is the closure under intersection with regular languages: the classical construction by Bar-Hillel et al. applies verbatim, because it preserves multiplicities of parse trees. It makes sense to prove this result in its most general form, for all mappings computed by injective nondeterministic finite tranducers (NFT). First, it is established under the following technical assumption.

Lemma 16. *Let G be a GF(2)-grammar over an alphabet Σ, and let a mapping $T\colon \Sigma^* \to 2^{\Omega^*}$ be computed by an injective NFT, which has the following property: for every pair $(w, x) \in \Sigma^* \times \Omega^*$, there is at most one computation on w that emits x. Then the language $T(L(G))$ is defined by a GF(2)-grammar G'. Furthermore, if G is linear, then so is G'.*

Proof (a sketch). The construction is standard, and the assumptions of injectivity and of the uniqueness of a computation ensure that, whenever the original grammar G defines w and $x \in T(w)$, the number of parse trees of x in the constructed grammar G' is the same as the number of parse trees of w in G. For that reason, $L(G') = T(L(G))$, as desired. □

Using this result, the desired closure property is proved as follows.

Theorem 17. *Let a mapping $T\colon \Sigma^* \to 2^{\Omega^*}$ be computed by an injective NFT. Then the language families defined by GF(2)-grammars and GF(2)-linear grammars are closed under T.*

Proof (a sketch). Since the definition of NFT is symmetric with respect to its input and its output, there is a single-valued NFT implementing a partial mapping $T'\colon \Omega^* \to \Sigma^*$, with $w = T'(x)$ if and only if $x \in T(w)$. As proved by Eilenberg [6, p. 186] a single-valued NFT can be transformed to an *unambiguous NFT*, that is, with at most one accepting computation on every input. Swapping the input and the output again yields an NFT implementing the original mapping T that satisfies the conditions of Lemma 16. Therefore, by Lemma 16, both language families are closed under T. □

Corollary 18. *The families defined by GF(2) and GF(2)-linear grammars are closed under intersection with regular languages, as well as under union with regular languages.*

For all other standard operations on languages, GF(2)-grammars demonstrate non-closure.

Theorem 19. *The family of languages described by GF(2)-grammars is not closed under (a) union, (b) intersection, (c) concatenation, (d) Kleene star, (e) left- and right-quotient with a two-element set, and (f) non-erasing homomorphisms. The same results hold for GF(2)-linear grammars.*

Intuitively, all these operations essentially use conjunction or disjunction in their definitions, and those Boolean operations are not expressible in GF(2).

Proof. The proofs of all cases are based on the languages L, L_1 and L_2 given in Lemmata 14–15. By Lemma 14, L_1 and L_2 are described by unambiguous linear grammars, and hence both these languages and their complements are described by GF(2)-linear grammars.

(a) Both $\overline{L_1}$ and $\overline{L_2}$ are described by GF(2)-linear grammars, but their union $\overline{L_1} \cup \overline{L_2} = \overline{L}$ is not.

(b) Similarly, $L_1 \cap L_2 = L$, where L_1 and L_2 are described by GF(2)-grammars, but L is not.

(c) Let c be a new symbol. By the assumptions, $\{\varepsilon, c\}$ and $c\overline{L_1} \triangle \overline{L_2} = c\overline{L_1} \cup \overline{L_2}$ are described by some GF(2)-grammars. Their concatenation is the following language.

$$(\{\varepsilon, c\} \cdot (c\overline{L_1} \cup \overline{L_2})) = \overline{L_2} \cup c(\overline{L_1} \cup \overline{L_2}) \cup cc\overline{L_1}$$

If it is represented by some GF(2)-grammar, then, by Theorem 17, so is its image under a finite transduction T defined by $T(cw) = w$ for all $w \in \Sigma^*$, and undefined on all other strings. This image is the language $\overline{L}$, which is not described by any GF(2) grammar, contradiction.

(d) By Corollary 18, $(ccc\overline{L_1} \triangle cc\overline{L_2} \triangle c)^* \cap (c^3\{a,b\}^*) = (ccc\overline{L_1} \cup cc\overline{L_2} \cup c)^* \cap (c^3\{a,b\}^*) = c^3(\overline{L_1} \cup \overline{L_2} \cup \{\varepsilon\}) = c^3(\overline{L} \cup \{\varepsilon\}) = c^3\overline{L}$. Simillarly to (c), this language is not described by any GF(2)-grammar.

(e) For symmetry reasons it suffices to prove only the left-quotient result. Denote $K \backslash M = \{\, v \mid \exists u \in K : \; uv \in M \,\}$. Indeed, $(\{\varepsilon, c\} \backslash (c\overline{L_1} \triangle \overline{L_2})) \triangle c\overline{L_1} = (\{\varepsilon, c\} \backslash (c\overline{L_1} \cup \overline{L_2})) \triangle c\overline{L_1} = (c\overline{L_1} \cup \overline{L_2} \cup \overline{L_1} \cup \varnothing) \triangle c\overline{L_1} = (\overline{L_1} \cup \overline{L_2}) = \overline{L}$.

(f) $h(c\overline{L_1} \triangle d\overline{L_2}) = h(c\overline{L_1} \cup d\overline{L_2}) = c(\overline{L_1} \cup \overline{L_2}) = c\overline{L}$, where the images of the letters under homomorphism $h\colon \{a,b,c,d\} \to \{a,b,c\}^*$ are $h(a) = a, h(b) = b, h(c) = h(d) = c$ respectively. □

In Table 1, the closure properties of GF(2)-grammars and of their linear subclass are summarized and compared with other grammar families. The operations featured in the table are: intersection with regular languages ($\cap$Reg), union ($\cup$), intersection ($\cap$), complementation ($\sim$), concatenation ($\cdot$), Kleene star ($*$), GF(2)-concatenation ($\odot$), GF(2)-inverse ($^{-1}$), quotient with regular languages (/Reg), homomorphisms (h), injective homomorphisms (h_{inj}), inverse homomorphisms (h^{-1}). All closure properties of GF(2) and GF(2)-linear grammars are proved in this paper. Non-closure of the classical families under GF(2)-concatenation and GF(2)-inverse is known [2]; most likely, this non-closure extends to linear conjunctive grammars and could be proved by the method of Terrier [20].

7 Hardest language

Some formal properties of GF(2)-grammars are the same as for ordinary grammars and are established by the same argument. One such property is Greibach's *hardest language theorem* [10], which has the same statement in the case of GF(2)-grammars.

	$\cap$	Reg	$\cup$	$\cap$	$\sim$	$\cdot$	$*$	$\odot$	$^{-1}$	/Reg	h	h_{inj}	h^{-1}
GF(2)-linear ($\triangle$, LIN$\cdot$)	+		−	−	+	−	−	−	−	−	−	+	+
Unambiguous ($\uplus$, UNAMB$\cdot$)	+		−	−	−	−	−	−	−	−	−	+	+
Ordinary ($\cup, \cdot$)	+		+	−	−	+	+	−	−	+	+	+	+
GF(2) ($\triangle, \odot$)	+		−	−	+	−	−	+	+	−	−	+	+
Linear conjunctive ($\cup, \cap$, LIN$\cdot$)	+		+	+	+	−	−	?	?	−	−	+	+
Conjunctive ($\cup, \cap, \cdot$)	+		+	+	?	+	+	?	?	−	−	+	+

Table 1. Closure properties of grammars under classical and under GF(2)-operations.

Theorem 20. *There exist an alphabet Σ_0 and a GF(2)-grammar $G_0 = (\Sigma_0, N_0, R_0, S_0)$, such that for every GF(2)-grammar over any alphabet Σ, there exists a homomorphism $h\colon \Sigma \to \Sigma_0^*$, such that a non-empty string w over Σ is in $L(G)$ if and only if $h(w)$ is in $L(G_0)$.*

The proof requires a Greibach normal form.

Definition 21. *A GF(2)-grammar is said to be in* Greibach normal form *(GNF), if all its rules are of the form $A \to a \odot B_1 \odot \ldots \odot B_\ell$, with $a \in \Sigma$, $\ell \geqslant 0$ and $B_1, \ldots, B_\ell \in N$.*

Proposition 22. *For every GF(2)-grammar G with $\varepsilon \notin L(G)$, there exists a GF(2)-grammar in the Greibach normal form that describes the same language.*

It is known that the transformation to the Greibach normal form preserves the number of parse trees [7, Lemma 4]. Taking this modulo two yields Proposition 22.

Proof (of Theorem 20). Greibach's classical construction for ordinary grammars applies here, because it is known to *preserve multiplicities*: for a non-empty string $w \in \Sigma^*$, its image $h(w)$ has the same number of parse trees in G_0 as w has in G [10, p. 307]. Therefore, the number of trees modulo 2 is preserved as well. □

8 Conclusion

The new negative results for GF(2)-grammars were sufficient to establish their position in the hierarchy and their basic closure properties. However, these methods are still quite limited, and the existence of GF(2)-grammars remains unknown even for some very simple languages. For instance, can the language $\{ a^n b^n c^n \mid n \geqslant 0 \}$ be defined by these grammars? Exactly which subsets of a^*b^* can be defined? In particular, what is the exact class of functions f, for which the language $L_f = \{ a^n b^{f(n)} \mid n \geqslant 0 \}$ can be defined—is it any larger than the class in Theorem 10?

References

1. J.-P. Allouche, J. Shallit, *Automatic Sequences: Theory, Applications, Generalizations*, Cambridge University Press, 2003.
2. E. Bakinova, A. Basharin, I. Batmanov, K. Lyubort, A. Okhotin, E. Sazhneva, "Formal languages over GF(2)", *Language and Automata Theory and Applications* (LATA 2018, Bar-Ilan near Tel Aviv, Israel, 9–11 April 2018), LNCS 10792, 68–79.
3. M. Barash, A. Okhotin, "An extension of context-free grammars with one-sided context specifications", *Information and Computation*, 237 (2014) 268–293.
4. T. Buchholz, M. Kutrib, "On time computability of functions in one-way cellular automata", *Acta Informatica*, 35:4 (1998), 329–352.
5. G. Christol, "Ensembles presque periodiques k-reconnaissables", *Theoretical Computer Science*, 9 (1979), 141–145.
6. S. Eilenberg, *Automata, Languages and Machines, Volume A*, Academic Press, 1974.
7. V. Forejt, P. Jančar, S. Kiefer, J. Worrell, "Language equivalence of probabilistic pushdown automata", *Information and Computation*, 237 (2014), 1–11.
8. S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.
9. S. Ginsburg, E. H. Spanier, "Quotients of context-free languages", *Journal of the ACM*, 10:4 (1963), 487–492.
10. S. A. Greibach, "The hardest context-free language", *SIAM Journal on Computing*, 2:4 (1973), 304–310.
11. O. H. Ibarra, S. M. Kim, "Characterizations and computational complexity of systolic trellis automata", *Theoretical Computer Science*, 29 (1984), 123–153.
12. A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615.
13. D. E. Knuth, "Context-free multilanguages", *Theoretical Studies in Computer Science*, Academic Press, 1992, 1–13.
14. A. Okhotin, "On the equivalence of linear conjunctive grammars to trellis automata", *RAIRO Informatique Théorique et Applications*, 38:1 (2004), 69–88.
15. A. Okhotin, "Conjunctive and Boolean grammars: the true general case of the context-free grammars", *Computer Science Review*, 9 (2013), 27–59.
16. A. Okhotin, "Input-driven languages are linear conjunctive", *Theoretical Computer Science*, 618 (2016), 52–71.
17. A. Okhotin, "Underlying principles and recurring ideas of formal grammars", *Language and Automata Theory and Applications* (LATA 2018, Bar-Ilan near Tel Aviv, Israel, 9–11 April 2018), LNCS 10792, 36–59.
18. I. Petre, A. Salomaa, "Algebraic systems and pushdown automata", in: Droste, Kuich, Vogler (Eds.), *Handbook of Weighted Automata*, Springer, 2009, 257–289.
19. H. Seki, T. Matsumura, M. Fujii, T. Kasami, "On multiple context-free grammars", *Theoretical Computer Science*, 88:2 (1991), 191–229.
20. V. Terrier, "On real-time one-way cellular array", *Theoretical Computer Science*, 141:1–2 (1995), 331–335.