

On the Complexity of Language Membership for Probabilistic Words

Antoine Amarilli   

Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRISTAL, France

Mikaël Monet   

Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRISTAL, France

Paul Raphaël   

École normale supérieure, Paris, France

Sylvain Salvati   

Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRISTAL, France

Abstract

We study the membership problem to context-free languages L (CFLs) on *probabilistic words*, that specify for each position a probability distribution on the letters (assuming independence across positions). Our task is to compute, given a probabilistic word, what is the probability that a word drawn according to the distribution belongs to L . This problem generalizes the problem of counting how many words of length n belong to L , or of counting how many completions of a partial word belong to L .

We show that this problem is in polynomial time for unambiguous context-free languages (uCFLs), but can be $\#P$ -hard already for unions of two linear uCFLs. More generally, we show that the problem is in polynomial time for so-called *poly-slicewise-unambiguous languages*, where given a length n we can tractably compute an uCFL for the words of length n in the language. This class includes some inherently ambiguous languages, and implies the tractability of *bounded CFLs* and of languages recognized by *unambiguous polynomial-time counter automata*; but we show that the problem can be $\#P$ -hard for nondeterministic counter automata, even for Parikh automata with a single counter. We then introduce classes of circuits from knowledge compilation which we use for tractable counting, and show that this covers the tractability of poly-slicewise-unambiguous languages and of some CFLs that are not poly-slicewise-unambiguous. Extending these circuits with negation further allows us to show tractability for the language of primitive words, and for the language of concatenations of two palindromes. We finally show the conditional undecidability of the meta-problem that asks, given a CFG, whether the probabilistic membership problem for that CFG is tractable or $\#P$ -hard.

2012 ACM Subject Classification Theory of computation \rightarrow Regular languages

Keywords and phrases Automaton, probabilistic words, context-free grammar, membership problem

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.5

Related Version *Full Version*: <https://arxiv.org/abs/2510.08127> [6]

Funding Partially supported by the ANR project ANR-25-CE23-1215 (“Expand”).

Acknowledgements We are grateful to Arthur Ledaguenel for initial discussions leading to the formulation of the problem, to Alexis de Colnet for early discussions, to Louis Jachiet for remarking that the problem is tractable for uCFLs, and to Charles Paperman and Michaël Cadilhac for advice on automata models.



© Antoine Amarilli, Mikaël Monet, Paul Raphaël, and Sylvain Salvati;
licensed under Creative Commons License CC-BY 4.0
43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).
Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng
Article No. 5; pp. 5:1–5:21



Leibniz International Proceedings in Informatics
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

One of the most fundamental problems in formal languages is the *membership problem* of determining whether an input word belongs to a language of interest. The problem is in linear time for regular languages when given as deterministic finite automata, but complexity increases for the class of *context-free languages* (CFLs): the best algorithms on an input word of length n run in time $O(n^\omega)$ where ω is the exponent of Boolean matrix multiplication [1]. Because of the practical importance of CFL parsing, several classes of CFLs have been identified with a better complexity, e.g., *unambiguous CFLs* (uCFLs), which can be recognized by *unambiguous context-free grammars* (uCFGs), can be parsed in $O(n^2)$ [33]; further, *deterministic CFLs* (DCFLs) can be parsed in linear time.

Beyond the question of membership testing for a single input word, subsequent works have investigated more general tasks. For instance, one problem is *counting*: given a CFL L and a length $n \in \mathbb{N}$, determine how many words of length n are in L . (Counting problems that do not fix the length can also be defined, e.g., the *coin-flip measure* of [22].) An FPRAS to *approximately* count $|L \cap \Sigma^n|$ when given n and a CFG for L was recently claimed in [67], but in the exact setting the task is intractable already for regular languages presented as NFAs [2]. This counting problem was also studied for CFLs in [13] when the target language is fixed (and the input only specifies the length); we will review this work in more detail later. Another task is to *sample* a word of the language of L uniformly at random: a polynomial algorithm for uCFGs was shown in [46, 64], and a generalization to finitely ambiguous CFGs is given in [14], along with a quasi-polynomial time algorithm for almost uniform sampling [39]. Other tasks include *enumeration*, i.e., producing the sequence of all words in a CFL L according to some order (e.g., the lexicographic order): see [82, 65, 32, 25]; and *ranking*, i.e., determining how many words in L are smaller than an input word [48, 49, 12].

In this paper, we introduce the problem of *language membership for probabilistic words*, for short *probabilistic membership*. To our knowledge this problem has not been previously investigated, and in some sense it generalizes the problems above, as we will explain. In our setting, a *probabilistic word* on an alphabet Σ is a sequence $p = p_1 \cdots p_n$ of length n , where each p_i is a probability distribution on Σ . The probabilistic word p is a concise representation of a probability distribution of words (each of length n) obtained by picking the i -th letter according to the distribution p_i , and assuming independence across letters. Probabilistic words are also called *weighted sequences* or *weighted strings* in the literature, with possible applications to bioinformatics, pattern recognition or data mining [20, 76, 21]. For a fixed language L , the *probabilistic membership* problem asks us to determine, given a probabilistic word p , the total probability of the outcomes that belong to L , i.e., the probability that a word drawn according to p belongs to L . The problem can be solved naively by going over all possible outcomes (of which there can be up to $|\Sigma|^n$); our question is to understand for which languages L we can solve this problem more efficiently, e.g., in polynomial time in n .

The probabilistic membership problem of course generalizes the non-probabilistic membership problem, and it can be seen as a weighted generalization of the counting problem of [13]; the latter corresponds to the case of probabilistic words that can only use the uniform distribution over Σ at every position. Moreover, it is not hard to see that the tractability of probabilistic membership can be used as a subroutine to allow efficient sampling, enumeration, and ranking in the radix order (up to polynomial-time overhead). Indeed, probabilistic membership generalizes the question of counting how many words of length n start with a given prefix, so that sampling, enumeration, and ranking can be performed using self-reducibility. Furthermore, we can also see probabilistic words as a weighted generalization of

so-called *partial words* [11]: these are words on an alphabet Σ augmented with a wildcard $??$ that can be replaced by any letter of Σ . We can see partial words as the special case of probabilistic words where each distribution is either uniform or Dirac (i.e., with only one possible outcome). Probabilistic membership to a language L is thus a generalization of the problem of counting the completions of partial words that belong to L ; to our knowledge this problem on partial words was not studied in relevant counting works [66, 15].

This paper focuses on the probabilistic membership problem in its own right, and aims at understanding for which CFLs it can be solved in polynomial time. Borrowing terminology from database theory [90], we mostly study the problem in *data complexity*, i.e., we assume that the target language is fixed and measure complexity only as a function of the input word. All our hardness results will be given in this sense, but some of our tractability results also hold in *combined complexity*, i.e., when the input contains both the word and some representation of the target language (e.g., as a CFG). Our work does not achieve a complete characterization of the CFLs that enjoy tractable probabilistic membership, but we give a hierarchy of sufficient conditions for tractability of the task (even beyond CFLs), complemented by lower bounds. We summarize our contributions below.

Contributions and outline. We give preliminaries and formally introduce the problem in Section 2. We then give some first results in Section 3, which are essentially generalizations of the unweighted counting results of [13]. Namely, we show that probabilistic membership is tractable for uCFLs (even in combined complexity), because we can count the total probability of derivation trees with a variant of the Cocke-Younger-Kasami (CYK) algorithm. However, we show that the task can be intractable (namely, $\#P$ -hard) for some inherently ambiguous CFLs: we give a simple example of such a CFL, and generalize and extend techniques from [13] to show hardness even in the case of the union of two unambiguous *linear CFLs*, i.e., the right-hand-side of each production rule contains at most one nonterminal.

The next section (Section 4) shows that there are CFLs that are inherently ambiguous but for which probabilistic membership is nevertheless tractable. These include, for instance, all *bounded CFLs* [38]: these are known to coincide with the class of *polyslender CFLs* [50], i.e., those CFLs containing only polynomially many words of length n . Thus, we can tractably solve probabilistic membership for bounded CFLs L (even in combined complexity) by simply listing all words of length n of L explicitly and summing their probabilities. Bounded CFLs are incomparable to uCFLs, as there are non-bounded uCFLs (e.g., palindromes) and also inherently ambiguous bounded CFLs (e.g., $\{a^n b^m c^m \mid n, m \in \mathbb{N}\} \cup \{a^n b^n c^m \mid n, m \in \mathbb{N}\}$). We give a unifying explanation to the tractability of uCFLs and bounded CFLs by introducing the class of languages L that are *poly-slicewise-unambiguous* in the sense that, given a length $n \in \mathbb{N}$, we can compute in polynomial time in n a uCFG G recognizing the words of length n of L , i.e., recognizing a language L' with $L \cap \Sigma^n = L' \cap \Sigma^n$. For such languages, probabilistic membership is tractable simply by reducing to uCFGs. We last introduce a formalism of *unambiguous polynomial-time counter automata* which are poly-slicewise-unambiguous: these recognize in particular some CFLs that are neither bounded nor unambiguous. We show that probabilistic membership is tractable for such automata, but that it can be $\#P$ -hard for some CFLs accepted by very restricted nondeterministic counter automata.

The natural question is then whether poly-slicewise-unambiguity is the unifying explanation for the tractability of probabilistic membership for CFLs. We answer in the negative in Section 5 by introducing a framework of *tractable circuits* from the field of *knowledge compilation* [29], inspired by database theory [4]. Intuitively, tractable circuits are circuits that give a factorized representation of a set of partial functions (intuitively corresponding

to outcomes of a probabilistic word) using some tractable operators: *Cartesian product* (called *decomposable AND* in the setting of Boolean knowledge compilation circuit classes), and *disjoint union* (called *deterministic OR* in the Boolean setting). These circuits are a multivalued analogue to so-called *sd-DNNF circuits* [29]. The choice of operators ensures that we can compute in linear time (up to the cost of arithmetic operations) the total probability of the assignments captured by a tractable circuit. We show that these circuits can be used to recapture the tractability of all poly-slicewise-unambiguous languages. What is more, we show that there are CFLs that are not poly-slicewise-unambiguous but admit tractable circuits: we use for this the language very recently studied in [57] whose length- n slices were recently shown [68] not to admit uCFLs of polynomial size in n .

We then continue our study of tractable circuits in Section 6 and extend them by adding a complementation operator (corresponding to negation in the Boolean sense). This follows the observation from probabilistic databases [69] that counting the satisfying valuations of d-DNNF circuits can be tractably achieved even in the presence of negation. Further, negation is very useful to encode some inclusion-exclusion-based counting arguments (through the full extent to which this is true is not yet understood [7]). Tractable circuits with negation imply the obvious fact that probabilistic membership is tractable for the complements of all tractable languages studied so far (which are generally not CFLs). We use those circuits to further show tractability for some other languages: we introduce the technique with the language of *primitive words* (whose context-freeness is a long-standing open problem [51]), and further apply it to the language of the *concatenation of palindromes*, a non-linear CFL of unbounded degree of ambiguity [26]. Our tractability result for probabilistic membership for this language generalizes an earlier result on unweighted counting [56].

We then conclude the paper in Section 7 and survey directions for future work; we also mention there the undecidability of finding out, given a CFG, whether it admits PTIME probabilistic membership. For reasons of space, most proofs are deferred to the full version of the article [6].

2 Preliminaries and Problem Statement

Words and languages. We fix a nonempty finite set of symbols Σ called an *alphabet* and whose elements are called *letters*. We let Σ^* be the set of all *words* over Σ , which are finite sequences of letters of Σ . We write ϵ for the empty word. The *mirror image* of $w = a_1 \cdots a_n \in \Sigma^*$ is $w^R := a_n \cdots a_1$; we call w a *palindrome* if $w = w^R$. A *language* over Σ is a subset of Σ^* . We omit the definition of standard language operations, e.g., concatenation, Kleene star, etc.

Grammars. A *context-free grammar* (CFG) over Σ is a tuple $\Gamma = (N, S, P)$ where N is a nonempty finite set of symbols called *nonterminals*, $S \in N$ is the *axiom*, and $P \subseteq N \times (\Sigma \cup N)^*$ is a set of *production rules*, with each production $(U, \alpha) \in P$ written as $U \rightarrow \alpha$. We call the letters of Σ *terminals*, and write them in lowercase. We say that Γ is a *linear CFG* if the right-hand-side of every production contains at most one nonterminal. The *size* of Γ , written $|\Gamma|$, is simply $|N|$ plus the total size of the production rules.

We omit the standard formal definitions of *derivations* and *parse trees* for CFGs. A CFG Γ *recognizes* the language $L(\Gamma)$ formed of the words $w \in \Sigma^*$ on which Γ admits a parse tree. We say that Γ is *unambiguous* (or a *uCFG*) if on every word it admits at most one parse tree; otherwise Γ is *ambiguous*. We call Γ *k-ambiguous* for $k \in \mathbb{N}$ if on every word it admits at most k parse trees. A language L is a *context-free language* (CFL) if there is a

CFG Γ with $L(\Gamma) = L$. It is an *unambiguous CFL* (uCFL) if Γ can be taken to be a uCFG (but note that there may also be ambiguous CFGs recognizing L). We define *unambiguous linear CFLs* and *k-ambiguous linear CFLs* in the expected way. A CFL L is called *inherently ambiguous* if it is not a uCFL, meaning that every CFG recognizing L is ambiguous.

Probabilistic words. We now define the notion of *probabilistic words* on an alphabet Σ :

► **Definition 2.1.** A probability distribution p on a finite set S is a function from S to $[0, 1]$ such that $\sum_{s \in S} p(s) = 1$. We call p uniform if we have $p(s) := 1/|S|$ for all $s \in S$, and call p the Dirac distribution on $s \in S$ if we have $p(s) := 1$ (hence $p(s') = 0$ for all $s' \neq s$).

A probabilistic word is a tuple $p = (p_1, \dots, p_n)$ of probability distributions on Σ , with n being the length of p . The semantics of p is that it represents a probability distribution $p: \Sigma^* \rightarrow [0, 1]$, also written p by abuse of notation. Intuitively, p gives probability 0 to words of length different from n , and otherwise the probability is obtained by multiplying the probabilities given by the distributions to the letters that occur at each position. Formally, for $w = w_1 \dots w_m$ in Σ^* , we have: $p(w) := 0$ if $m \neq n$, and $p(w) := \prod_{k=1}^n (p_k(w_k))$ if $m = n$.

Concatenating probabilistic words means concatenating the tuples: this is associative and the neutral element is the empty probabilistic word (the empty tuple). Note that probabilistic words indeed define probability distributions on Σ^* , i.e., the probabilities sum to 1.

Probabilistic membership problem. We now define the problem that we study, called *probabilistic membership*. Given a language L on an alphabet Σ , and a probabilistic word p , we abuse notation and write $p(L)$ to mean the probability that a random word of Σ^* drawn according to p will belong to the language L . Formally, we have: $p(L) := \sum_{w \in L} p(w)$. The *probabilistic membership problem* then asks: given a language L and a probabilistic word p , compute $p(L)$. Here, probabilities of the input word are given as rational numbers $\frac{r}{q}$ for $(r, q) \in \mathbb{N} \times (\mathbb{N} \setminus \{0\})$, with r and q encoded in binary. We focus on the *data complexity* perspective, where the target L is fixed: this defines a problem $\#pM(L)$, with complexity measured as a function of p . However, some of our tractability results hold in *combined complexity*, where the input is p along with a CFG Γ with $L(\Gamma) = L$. We illustrate special cases of probabilistic membership below:

- For each $u \in \Sigma^*$, the *Dirac probabilistic word* $u' = (p_1, \dots, p_{|u|})$ is the one that ensures that $u'(u) = 1$, namely, p_i is the Dirac distribution on u_i for each $1 \leq i \leq |u|$. We may abuse notation and identify u' with u . Probabilistic membership to a language L thus generalizes the standard (non-probabilistic) membership problem to L .
- For each $n \in \mathbb{N}$ the *uniform probabilistic word* of length n is $p_n = (p_1, \dots, p_n)$ where each p_i is the uniform distribution on Σ . Its outcomes are precisely the words of Σ^n , each with probability $1/|\Sigma|^n$. Probabilistic membership to L thus generalizes (up to renormalization by an easily-computable factor of $|\Sigma|^n$) the unweighted counting problem studied in [13]: given an integer $n \in \mathbb{N}$, compute the cardinality $|L \cap \Sigma^n|$.
- Probabilistic words are also a weighted generalization of *partial words* [11]. A *partial word* is a word p over the alphabet $\Sigma \cup \{?\}$, for $?$ a fresh wildcard. The semantics of p is that it represents a set of possible words over Σ^* obtained by considering all possible *completions* of p , i.e., all possible ways to replace each occurrence of $?$ in p by a letter of Σ . Note that all completions have length $|p|$. A partial word p can be translated to a probabilistic word $q = (q_1, \dots, q_{|p|})$ where for each $1 \leq i \leq |p|$ we set q_i as the Dirac on p_i if $p_i \in \Sigma$ or as the uniform distribution if $p_i = ?$. Probabilistic membership to L thus generalizes the problem of counting the number of completions of p that are in L . In fact, all our hardness results already hold for a variant of this problem (see [6, Appendix A]).

Model and problem statement. Recall that FP is the class of function problems that can be solved by a deterministic polynomial-time Turing machine, and that $\#P$ is the class of counting problems that can be expressed as the number of accepting runs of a nondeterministic polynomial-time Turing machine. When seeing the output of FP problems as an integer, we have $FP \subseteq \#P$. Further, $FP^{\#P}$ is the class of function problems that can be solved in deterministic polynomial time with access to a $\#P$ oracle: we use this class simply because the answers to our problems are formally probabilities rather than integers. In the data complexity perspective, it is easy to see that the probabilistic membership problem $\#pM(L)$ is in $FP^{\#P}$ for any language L which is in the class $PTIME$, i.e., where there is a deterministic polynomial-time algorithm testing membership of an input (non-probabilistic) word to the language L : see, e.g., [40, Theorem 4.2] or [70, p19]. All languages studied in the sequel will be $PTIME$ languages in this sense (as is the case of all CFLs).

We will be interested in languages L for which $\#pM(L)$ is in FP (we will often abuse terminology and say in that case that $\#pM(L)$ is in $PTIME$), and languages L for which $\#pM(L)$ is $\#P$ -hard, i.e., any problem in $\#P$ can be reduced by Turing reductions to $\#pM(L)$: note that we always use Turing reductions (and not many-one reductions). When giving complexity bounds with an explicit polynomial degree, to avoid specifying the details of the computational model we will state them *up to the cost of arithmetic operations*, i.e., assuming that all arithmetic operations take unit time.

Our goal in this paper is to determine for which languages L the problem $\#pM(L)$ can be solved in polynomial time data complexity, and for which it is $\#P$ -hard.

3 CFLs and Unambiguity

To start our study of tractable cases of the $\#pM$ problem for CFLs, we study the effect of *unambiguity*. On the one hand, we show that $\#pM$ is tractable for *uCFLs*, in data complexity and even in combined complexity. This implies in particular that $\#pM$ is also tractable for all regular languages, and gives a weighted analogue to the tractability result of [13] (with similar techniques).

On the other hand, we show that $\#pM$ can be $\#P$ -hard for inherently ambiguous CFLs: we give an explicit hard linear CFL. We further show that $\#pM$ is already $\#P$ -hard for unions of two linear uCFLs, though their language is more complicated. This extends the intractability results of [13] to the weighted setting, noting that their work instead shows $\#P_1$ -completeness [88] because their input instances are unweighted hence fully defined by their length. Our techniques are similar but we encode Turing machine computations with the encoding of [9] (whereas [13] uses [44]), so our result applies to linear CFGs.

Unambiguous CFLs. We focus on the case of unambiguous CFLs and show:

► **Proposition 3.1.** *Given Γ a uCFG and p a probabilistic word, we can solve $\#pM$ in $O(|p|^3|\Gamma|)$, assuming unit cost for arithmetic operations.*

We do not claim that the cubic dependency in the probabilistic word is optimal, and leave to future work the investigation of whether the complexity can be lowered, e.g., to match the complexity of Valiant's parser [87], or the quadratic upper bound given by the Earley parser for unambiguous CFGs [33].

This result implies that $\#pM$ is tractable for many CFGs, e.g., the language of palindromes, the language $\{a^n b^n \mid n \in \mathbb{N}\}$, or the language of Dyck words. It also implies the tractability in data complexity of subclasses of uCFGs, for instance *deterministic CFGs* [37], or indeed all

regular languages. Our result generalizes the tractability in combined complexity of (weighted or unweighted) counting for regular languages given as *unambiguous automata*, see, e.g., [24]. We note that, by contrast, hardness holds in combined complexity with nondeterministic automata as input: the problem is then #P-complete [2] but admits an FPRAS [8].

Proof sketch. We convert the input uCFG Γ in linear time into *arity-2 normal form* (2NF) [63], which preserves unambiguity [5, Appendix A.2]. We then generalize the textbook CYK parsing algorithm that computes inductively, for each nonterminal X and each interval $1 \leq i \leq j \leq n$, whether the factor of the input word with endpoints $[i, j]$ can be derived from X . Instead, we compute the *total probability* of the words that can be derived from X in the distribution represented by the factor (p_i, \dots, p_j) . As Γ is unambiguous, every word admits at most one derivation tree, so there are no double counts. ◀

However, for arbitrary CFGs, we cannot solve #pM simply by counting the probability of derivation trees in this fashion – because the probability of words with multiple derivation trees is counted as many times as there are trees¹.

An intractable CFL. The previous result shows that #pM is tractable for uCFLs, leaving open the complexity of inherently ambiguous CFLs. We will now show intractability in this case by first giving a simple explicit CFL for which #pM is hard, before extending the result to a 2-ambiguous CFL in the sequel.

Our #P-hardness result is by reducing from the problem of counting satisfying assignments to a certain class of Boolean formulas, called *PP2DNFs*, and defined below:

► **Definition 3.2** ([86]). *A positive partitioned 2-DNF formula, for short PP2DNF, is an expression on Boolean variables $V = \{x_1, \dots, x_n, y_1, \dots, y_n\}$ of the form $\bigvee_{1 \leq k \leq m} (x_{i_k} \wedge y_{j_k})$ where $i_1, \dots, i_m, j_1, \dots, j_m \in \{1, \dots, n\}$. Given a PP2DNF formula Φ on variables V , a valuation v is a mapping ν from V to $\{0, 1\}$. It is said to be a satisfying valuation if, when substituting the variables of Φ according to ν , the formula Φ evaluates to true.*

Satisfiability for PP2DNFs is trivial as there are no negations; by contrast:

► **Theorem 3.3** ([88, 86]). *The following problem, written #PP2DNF, is #P-hard: given a PP2DNF formula Φ , count the number of satisfying valuations of Φ .*

We will now define our hard CFL L_0 on alphabet $\Sigma := \{0, 1, \#\}$:

$$L_0 := \{u\#\Sigma^*v^R\Sigma^* \mid u, v \in \{0, 1\}^*, v \leq u\}$$

where we abuse notation and write Σ^* to denote arbitrary factors in Σ^* , and where we denote by \leq the order relation defined on $\{0, 1\}$ by $0 \leq 1$ and extended in a pointwise manner on binary words of the same length, i.e., we have $v \leq u$ if $|u| = |v|$ and for each $1 \leq i \leq |u|$ we have $v_i \leq u_i$. Equivalently, for each $1 \leq i \leq |u|$ such that $v_i = 1$ we also have $u_i = 1$.

It is not hard to see that L_0 is a linear CFL, e.g., it can be recognized by $\Gamma_0 = (\{S, S_1, S_2\}, S, P)$ with productions:

$$S \rightarrow S0 \mid S1 \mid S\# \mid S_1 \quad S_1 \rightarrow 0S_10 \mid 1S_10 \mid 1S_11 \mid \#S_2 \quad S_2 \rightarrow 0S_2 \mid 1S_2 \mid \#S_2 \mid \epsilon$$

Our first hardness result is then:

¹ However, the algorithm as presented could apply to those CFGs in which all accepted words have precisely the same number of derivation trees. We do not know if this strictly generalizes uCFLs: see [3].

► **Proposition 3.4.** *The problem $\#pM(L_0)$ is $\#P$ -hard.*

Proof sketch. We code a PP2DNF Φ into a probabilistic word p defined as follows. It starts with $2n$ times the uniform distribution on $\{0, 1\}$ (representing all possible Boolean valuations on $\{0, 1\}$), followed by $\#$, then followed by $\#$ -separated codings of each clause C_i as a non-probabilistic (mirrored) word u_i^R of length $2n$. The word u_i carries a 1 at the position of the variables occurring in C_i . One then sees that the outcomes of p satisfying L_0 are precisely those for which there is a clause C_i such that the outcome of u contains a 1 at all positions where u_i contains as 1, i.e., precisely those where the corresponding Boolean valuation satisfies Φ . ◀

We remark that the hardness proof easily adapts to a more symmetric language:

► **Proposition 3.5.** *The problem $\#pM(L'_0)$ is $\#P$ -hard for the following CFL on $\Sigma = \{0, 1, \#\}$:*

$$L'_0 := \{\Sigma^* \# u \# \Sigma^* \# v^R \# \Sigma^* \mid u, v \in \{0, 1\}^*, u \leq v\}.$$

Intractability for 2-ambiguous linear CFLs. We now move to our second hardness result:

► **Proposition 3.6.** *Let $\Sigma = \{0, 1, \#\}$ and $\Sigma' = \{0, 1\}$. There are two linear uCFLs L_L and L_R on Σ such that $\#pM(L_L \cup L_R)$, is $\#P$ -complete.*

This result above implies in particular² that $\#pM$ is hard already for 2-ambiguous linear CFLs.

Proof sketch. We show instead hardness of $\#pM(L_L \cap L_R)$, which is PTIME-equivalent by inclusion-exclusion, using the fact that $\#pM(L_L)$ and $\#pM(L_R)$ are in PTIME by Proposition 3.1. We fix a nondeterministic polynomial-time Turing machine for which it is $\#P$ -hard to count the accepting runs, and encode its runs as in [9], intuitively as a probabilistic word of the form $p = w_1 \# w_3 \# \dots \# w_4^R \# w_2^R$, with all w_i having the same length and each of them being uniform probabilistic words that will encode the configuration of the machine at time step i . (Note that the word p hardcodes a polynomial-sized initial tape and a polynomial duration for the computation.) The language L_L checks the correctness of even transitions (i.e., it reads w_{2k+1} and w_{2k+2} in lockstep for each k and check that the transition of the machine is correct), and L_R does the same for odd transitions (it skips until the first $\#$ and then proceeds accordingly). This ensures that $L_L \cap L_R$ precisely accepts those outcomes of p where each pair of configuration is a valid transition, i.e., $\#pM(L_L \cap L_R)$ counts the accepting runs of the machine. ◀

4 Poly-Slicewise Unambiguity

We have seen in the previous section that $\#pM$ is tractable for uCFLs, and can be intractable already for 2-ambiguous linear CFLs. This leaves open the question of whether unambiguity is the tractability boundary for $\#pM$ on CFLs. In this section, we show that this is not the case: there are inherently ambiguous CFLs for which $\#pM$ is tractable. We do so by introducing the notion of *poly-slicewise-unambiguous* languages, which are immediately seen to generalize uCFLs and to enjoy tractable $\#pM$. Then, we state some consequences of this result: first on *polyslender languages*, then on *unambiguous polynomial-time counter automata*. Third, we show that, by contrast, $\#pM$ can be hard already on very restricted counter automata if we forego the unambiguity requirement.

² We do not know whether this is a generalization, because it is not known whether all 2-ambiguous CFLs can be written as the union of two uCFLs, cf [92, Section 9.3, point (iv)].

Poly-slicewise-unambiguity. We introduce our more general class of tractable languages:

► **Definition 4.1.** *A language L is poly-slicewise-unambiguous if there exists a polynomial-time algorithm \mathcal{A}_L for the following task: given as input an integer $n \in \mathbb{N}$ (written in unary), compute a uCFG $\Gamma_{L,n}$ such that $L(\Gamma_{L,n}) \cap \Sigma^n = L \cap \Sigma^n$.*

Poly-slicewise-unambiguous languages include languages that are not CFLs, hence tractability will also hold for such languages. Note that the complexity of \mathcal{A}_L is measured as a function of n only (as L is fixed). Further note that a similar notion has been studied in [79, 93] under the name “1U”. These are families of sequences of languages $(L_i)_{i \in \mathbb{N}}$ where each L_i has a unambiguous NFA of size $p(i)$ for some fixed polynomial p . However, unlike us, the definition of 1U does not require the NFAs to be computable in PTIME, i.e., these works use a non-uniform setting.

Of course, any uCFL L is immediately poly-slicewise-unambiguous (taking \mathcal{A}_L that always returns an unambiguous grammar Γ for L), but we will later show that some inherently ambiguous CFLs are poly-slicewise-unambiguous. The following tractability result in data complexity immediately follows from Proposition 3.1:

► **Proposition 4.2.** *Let L be poly-slicewise-unambiguous. Then $\#pM(L)$ is in polynomial time.*

Indeed, to solve $\#pM(L)$ on an input probabilistic word p of length n , we use \mathcal{A}_L to compute in polynomial time the uCFG $\Gamma_{L,n}$, and then invoke Proposition 3.1. The overall complexity (up to arithmetics) is in $O(n^{k+3})$ if \mathcal{A}_L has running time bounded by $O(n^k)$.

We will now show how Proposition 4.2 specializes to so-called *bounded CFLs* and to counter automata, and remark in particular that it strictly generalizes Proposition 3.1.

Bounded CFLs. In this section we study the class of *bounded CFLs*, namely, those CFLs L for which there exist words u_1, \dots, u_k such that $L \subseteq u_1^* \cdots u_k^*$. This class has been studied for a long time [38] and was equivalently characterized as those CFLs that are *polyslender*, i.e., there is $k \in \mathbb{N}$ such that for all $n \in \mathbb{N}$, we have $|L \cap \Sigma^n|$ is in $O(n^k)$. It is immediate that bounded CFLs are polyslender, and the reverse inclusion is known to hold [50]. The following result is easy to show by a variant of CYK:

► **Proposition 4.3.** *Given a CFG Γ for a bounded CFL and an integer $n \in \mathbb{N}$, we can compute the list of words of $L(\Gamma) \cap \Sigma^n$ in time polynomial in Γ and n .*

Thus, bounded CFLs are poly-slicewise-unambiguous, even in “combined complexity”:

► **Claim 4.4.** *Any polyslender CFL L is poly-slicewise-unambiguous, and the algorithm \mathcal{A}_L can be made to run in PTIME in the input length and in an input CFG representing L .*

This implies the following combined tractability result of $\#pM$ on bounded CFLs:

► **Corollary 4.5.** *The $\#pM$ problem for polyslender CFLs is tractable in combined complexity.*

Importantly, some polyslender CFLs are inherently ambiguous, such as $\{a^n b^m c^m \mid n, m \in \mathbb{N}\} \cup \{a^n b^n c^m \mid n, m \in \mathbb{N}\}$. Hence, this result covers some languages not covered by Proposition 3.1. Also note that the results above adapt to any language which is “constructively polyslender” in the sense of Proposition 4.3 (even beyond CFLs).

Unambiguous counter automata. We now present another language formalism enjoying tractable #pM via Proposition 4.2, namely, *polynomial-time counter automata*:

► **Definition 4.6.** A counter automaton is a machine $\mathcal{A} = (Q, \Sigma, q_0, k, F, \delta)$ where Q is a finite set of states, Σ is the alphabet, $q_0 \in Q$ is the initial state, $k \in \mathbb{N}$ is the number of counters, $F \subseteq Q \times \mathbb{Z}^k$ is the acceptance relation, and $\delta \subseteq Q \times \mathbb{Z}^k \times \Sigma \times Q \times \mathbb{Z}^k$ is the transition relation. A configuration of \mathcal{A} is a tuple (q, \vec{v}) with $q \in Q$ and $\vec{v} \in \mathbb{Z}^k$, the initial configuration is $(q_0, \vec{0})$, and a configuration is final if it is in the relation F . A configuration (q', \vec{v}') is a successor of (q, \vec{v}) for the letter $a \in \Sigma$ if the tuple $(q, \vec{v}, a, q', \vec{v}')$ is in the relation δ . A run of \mathcal{A} on a word $w = a_1 \cdots a_n$ is a sequence of configurations c_0, \dots, c_n with c_0 the initial configuration and c_{i+1} a successor of c_i for a_i for each $1 \leq i \leq |w|$. The run is accepting if the last configuration is final. The language $L(\mathcal{A})$ is the set of words on which \mathcal{A} has an accepting run. We say that \mathcal{A} is unambiguous if it has at most one accepting run on every word.

We say that a counter automaton is polynomial-time if there is a polynomial P satisfying two requirements. First, for every $n \in \mathbb{Z}$ and for every accepting run $(q_0, \vec{0}), (q_1, \vec{v}_1), \dots, (q_n, \vec{v}_n)$ of \mathcal{A} on a word w of length n , we have $\|\vec{v}_i\|_\infty \leq P(n)$ for all $1 \leq i \leq n$, where $\|\vec{v}\|_\infty$ denotes the max of the absolute values. Second, there is an algorithm that decides whether $F(q, \vec{v})$ holds on an input $(q, \vec{v}) \in Q \times \mathbb{Z}^k$ in time bounded by $P(\|\vec{v}\|_\infty)$, and likewise there is an algorithm deciding whether $\delta(q, \vec{v}, a, q', \vec{v}')$ holds in time bounded by $P(\|\vec{v}\|_\infty + \|\vec{v}'\|_\infty)$.

The model described here is very general: it allows arbitrary polynomial-time computations at each step, and it can freely use the value of counters during the run (i.e., it is not *blind* or *partially blind* [42]). Thus, it generalizes the model of counter machines from Fischer et al. [35] when those machines are required to be *real-time* (i.e., they read one input symbol at each transition), or more generally when they operate in polynomial-time (so they can only do a polynomial number of autonomous transitions between input symbols). Further, the model generalizes *Vector Addition Systems with States* or *VASSes* [55]; *Parikh automata* [58] (also called \mathbb{Z} -*VASSes* [27] or *integer VASSes* [23]); pushdown automata with a unary stack alphabet aka *one-counter automata* [89]; and extensions such as *Parikh one-counter automata* [19]. The main restriction is that our counter automata must read the word in a one-way fashion and in particular they only see each input symbol once.

The non-probabilistic membership problem to polynomial-time counter automata \mathcal{A} is in PTIME, and for unambiguous such automata #pM is in PTIME for similar reasons:

► **Proposition 4.7.** Let \mathcal{A} be an unambiguous polynomial-time counter automata. Then $L(\mathcal{A})$ is poly-slice-wise-unambiguous. (Hence, #pM($L(\mathcal{A})$) can be solved in polynomial time.)

Proof sketch. We use a product construction to construct a nondeterministic finite automaton (NFA) A_n with $L(A_n) = L(\mathcal{A}) \cap \Sigma^n$, by considering the polynomial number of potentially reachable configurations, and testing the acceptance and transition relations in polynomial time. As \mathcal{A} is unambiguous, we know that A_n is also an unambiguous NFA, and A_n can directly be converted in linear time to a uCFG, which concludes. ◀

By Proposition 4.2, this immediately implies that #pM(L) is tractable for any language recognized by an unambiguous polynomial-time counter automata: so the same is true for languages recognized by unambiguous Parikh automata, unambiguous VASSes, etc.

To contrast with our previous results, note that unambiguous polynomial-time counter automata and uCFLs are incomparable. These automata cannot recognize some uCFLs such as the language of palindromes, intuitively because this requires exponentially many

configurations: see [71, Theorem 5]. Conversely, these automata can recognize some languages that are not even CFLs (e.g., $\{a^n b^n c^n \mid n \in \mathbb{N}\}$), and they can also recognize some CFLs such as $\{u \in \Sigma^* \mid |u|_a \neq |u|_b \vee |u|_a \neq |u|_c\}$ that are not polyslender and inherently ambiguous [10, Proposition 1.10], so their tractability for #pM did not follow from our previous results.

Nondeterministic counter automata. Is the unambiguity requirement in Proposition 4.7 necessary? In this section, we show that #pM is intractable on nondeterministic counter automata, even for CFLs accepted by very restricted such automata:

► **Proposition 4.8.** *The problem #pM is #P-hard for L_1 and L_2 on $\Sigma = \{a, \$, \#\}$, with:*

$$L_1 := \{\Sigma^* \# \# a^i \# \Sigma^* \# a^i \# a^j \# \Sigma^* \# a^j \# \# \Sigma^* \mid i, j > 0\}$$

$$L_2 := \{\Sigma^* \# \# a^i \# \Sigma^* \# a^{i'} \# a^j \# \Sigma^* \# a^{j'} \# \# \Sigma^* \mid i, i', j, j' > 0, i + j = i' + j'\}$$

Proof sketch. We reduce from #PP2DNF. The left and right parts of the words contain blocks of each possible length (corresponding to the variables x_i and y_j respectively), with a 1/2 chance of being set to true, and the middle part contains a block for each clause. ◀

Both L_1 and L_2 are CFLs because they can be recognized by nondeterministic pushdown automata with unary stack alphabet: this is a restriction of our counter automaton formalism, with only one counter and very simple transition and acceptance relations. The language L_2 can even be recognized by a (nondeterministic) *1-dimensional Parikh automaton*, i.e., a counter automaton with a single blind counter which is only tested at the end of the run. Hence, hardness for #pM can hold even for very restricted ambiguous counter automata.

5 Tractable Circuits

We have seen that #pM is tractable for any uCFL, and also for any poly-slicewise-unambiguous language (yielding more tractable CFLs). In this section, we give a more elaborate tractability approach for #pM, using a notion of *tractable multivalued circuits*, that we will follow in the rest of the paper. We will show that this formalism ensures the tractability of #pM, and that it subsumes uCFLs and poly-slicewise-unambiguous languages.

The section is structured as follows. We first introduce our circuits and show that counting is tractable for them. Then, we show that tractable circuits generalize the tractability of poly-slicewise-unambiguous languages, and further show that the generalization is strict.

Tractable circuits. We will define (*smooth*) \times, \uplus -circuits following [4], which are a multivalued analogue to the *sd-DNNF* circuits used in knowledge compilation [29]. A *circuit* C is a directed acyclic graph with vertices called *gates*. An *input* to a gate g is a gate g' having a directed edge to g , the *fan-in* of g is the number of inputs that g has, and the *fan-out* of g is the number of gates of which g is an input. An *input gate* is a gate with fan-in zero. We assume that C has a distinguished gate with fan-out zero, called the *output gate*.

Let Σ be the alphabet. A \times, \uplus -circuit on Σ is a circuit whose input gates are labeled by pairs $n : a$ for $n > 0$ and $a \in \Sigma$, and whose internal gates are labeled with \times or \uplus and satisfy the three requirements of *decomposability*, *smoothness*, and *disjointness* that we now present. To state the first two requirements, we need to define inductively the *domain* of each gate g of C : the domain of an input gate g labeled by $n : a$ is $\text{dom}(g) := \{n\}$, and the domain $\text{dom}(g)$ of an internal gate g is the union of $\text{dom}(g')$ for all inputs g' of g . The *domain* $\text{dom}(C)$ of C is that of its output gate. We then say that a \times -gate g is *decomposable* if the

domains of the input gates of g are pairwise disjoint, and we require that C is *decomposable*, i.e., every \times -gate of C is. We say that a \uplus -gate g is *smooth* if the domains of the input gates of g are all identical, and we require that C is *smooth*, i.e., every \uplus -gate is.

To state disjointness, we need to define inductively the *sets of assignments* $S(g)$ captured by each gate g of C , as a set of partial functions f from \mathbb{N} to Σ with $\text{dom}(f) = \text{dom}(g)$:

- For an input gate g of the form $n : a$, we have $S(g) := \{n \mapsto a\}$;
 - For an internal \uplus -gate g , letting g_1, \dots, g_k be its input gates, we have $S(g) := \uplus_i S(g_i)$;
 - For an internal \times -gate g , letting g_1, \dots, g_k be its input gates, we have $S(g) := \times_i S(g_i)$.
- The \times operator denotes the Cartesian product, where we identify a tuple of functions $(f_1, \dots, f_k) \in \times_i S(g_i)$ with the function f defined like each f_i on $\text{dom}(f_i) = \text{dom}(g_i)$, noting that by decomposability these domains are disjoint. In particular, if g is a \uplus -gate with no inputs then $S(g) := \emptyset$, and if g is a \times -gate with no inputs then $S(g) := \{\emptyset_\Sigma\}$ for \emptyset_Σ the function to Σ with empty domain. We will write $S(C)$ to mean $S(g_0)$ for g_0 the output gate of C . We then say that an \uplus -gate g is *disjoint* if for any two inputs g' and g'' of g , the sets $S(g')$ and $S(g'')$ are disjoint. We require that C is *disjoint*, i.e., every \uplus -gate is.

Given a probabilistic word $p = (p_1, \dots, p_n)$, and a circuit C with domain $\text{dom}(C) = \{1, \dots, n\}$, the *probability* $p(C)$ of C under p is the sum of the probabilities of the functions of $S(C)$ under p , where the probability of $f: \text{dom}(C) \rightarrow \Sigma$ under p is simply $p(w_f)$ for the word $w_f = f(1) \cdots f(n)$. The point of \times, \uplus -circuits C is that we can tractably compute the probability of C under a probabilistic word p without materializing the (potentially exponential) set $S(C)$ of assignments that C captures. This is the immediate analogue of the tractability of model counting for *sd-DNNFs* [28]:

► **Proposition 5.1** ([28]). *Given a \times, \uplus -circuit C and a probabilistic word $p = (p_1, \dots, p_n)$, we can compute $p(C)$ in linear time (assuming unit cost for arithmetic operations).*

Languages admitting tractable circuits. For $n \in \mathbb{N}$, we say that a \times, \uplus -circuit *captures the n -th slice* of a language L if $\text{dom}(C) = \{1, \dots, n\}$ and $S(C)$ precisely corresponds to the words of $L \cap \Sigma^n$, i.e., a function $f: \{1, \dots, n\} \rightarrow \Sigma$ is in $S(C)$ precisely if the word $f(1) \cdots f(n)$ is in L . We say that a language *admits tractable circuits* if there is an algorithm A running in polynomial time in its input $n \in \mathbb{N}$ which computes a \uplus, \times -circuit $A(n)$ that captures the n -th slice of L . (Note that this is stronger than requiring the mere existence of polynomial-sized circuits: we assume that they can also be computed in polynomial time.) The following is then a direct consequence of Proposition 5.1:

► **Proposition 5.2.** *If a language L admits tractable circuits then $\#pM(L)$ is in PTIME.*

Which languages admit tractable circuits? In fact, all poly-slicewise-unambiguous languages do, because we can build tractable circuits for uCFLs:

► **Proposition 5.3.** *Given a uCFL G and an integer n , we can build in time $O(|G|n^3)$ a \times, \uplus -circuit that captures the n -th slice of $L(G)$.*

This implies the following on poly-slicewise-unambiguous languages (in particular uCFLs):

► **Corollary 5.4.** *Any poly-slicewise-unambiguous language has tractable circuits.*

A tractable non-poly-slicewise-unambiguous CFL. We next show that, even within the class of CFLs, the converse of Corollary 5.4 does not hold. Thus, the tractability of $\#pM$ for languages admitting tractable circuits (Proposition 5.2) strictly generalizes the same result for poly-slicewise-unambiguous languages (Proposition 4.2). For this, consider the language

$L_3 = \{(a+b)^k a(a+b)^n a(a+b)^{n-k-2} \mid k, n \in \mathbb{N}, 0 \leq k < n-1\}$ on $\Sigma = \{a, b\}$, which consists of the words of length $2n$ which contain two occurrences of a at distance exactly n . This language is a CFL which is inherently ambiguous [57, Proposition 5.8]. It is shown in [68, Theorem 1] that this language is not poly-slicewise-unambiguous: they show a lower bound of $2^{\Omega(n)}$ on the size of any uCFG accepting $L \cap \Sigma^n$, so in particular there is no algorithm computing such uCFGs in polynomial time in n . However:

► **Claim 5.5.** *The language L_3 admits tractable circuits.*

Proof sketch. We first test letters a_1 and a_{n+1} and check if they are both equal to a . If not, we move to a_2 and a_{n+2} . We continue until we reach a_n, a_{2n} , and otherwise we fail. This describes an *ordered binary decision diagram* or OBDD [18], which gives a \times, \uplus -circuit. ◀

This implies that $\#pM(L_3)$ is tractable. Intuitively, the proof uses the fact that tractable circuits are not constrained to reading the word in left-to-right fashion, or in any predefined order: this is in contrast with uCFGs and with poly-slicewise-unambiguous languages. This also resembles formalisms such as *Multiple Context Free Grammars* (MCFGs) [81], which generalize CFGs by deriving tuples of strings instead of strings: MCFGs can recognize complex parenthetical structures [53], languages that do not satisfy strong forms of pumping lemmas [54], or images of regular tree languages by MSO tree-to-string transductions [61]. In fact, since MCFGs admit a generalization of the CYK algorithm [81], we could easily generalize Proposition 5.3 to unambiguous MCFGs. This would recapture the tractability of $\#pM(L_3)$, and also show tractability for other languages definable by unambiguous MCFGs.

6 Tractable Circuits with Complementation

We have introduced the notion of a language *having tractable circuits*, and showed that this implies the tractability of $\#pM$. In this section, we show how to extend tractable circuits with a *complementation operator*, intuitively corresponding to Boolean negation, and we show that languages enjoying such circuits are still tractable. We then apply this technique to show the tractability of $\#pM$ for two languages: the context-free language PAL^2 of the concatenation of two palindromes, and the language of primitive words. It is known that neither of these languages admits an unambiguous CFG, so their tractability would not follow from Proposition 3.1; but we do not know whether these languages could be handled with the technique of poly-slicewise-unambiguity, or with tractable circuits without complementation.

The section is structured as follows. We first present the notion of circuits with complementation. Then, we first use such circuits to show the tractability of $\#pM$ on the language of primitive words, because the proof is simpler. We then turn to the language PAL^2 .

Circuits with complement gates. The motivation for extending \times, \uplus -circuits with complement gates is that complements of tractable languages for $\#pM$ are always tractable:

► **Claim 6.1.** *For any language L , the problem $\#pM(\Sigma^* \setminus L)$ reduces in PTIME to $\#pM(L)$.*

Proof. The answer to $\#pM(L')$ on p is simply $1 - A$, for A the answer to $\#pM(L)$ on p . ◀

Thus, all our tractability results on languages immediately extend to their complements (even though the complement of a CFL, or indeed of a uCFL, is generally not a CFL [45]).

However, beyond the use of negation at the top-level, we could also support negation as an arbitrary intermediate gate. This motivates the definition of $\times, \uplus, \complement$ -circuits:

► **Definition 6.2.** A $\times, \uplus, \mathbb{C}$ -circuit on an alphabet Σ is defined like a \times, \uplus -circuit in Section 5 but where we additionally allow \mathbb{C} -gates. Such gates must have only one input, and their semantics is defined as follows: for g a \mathbb{C} -gate with input g' , let S be the set of all functions from $\text{dom}(g) = \text{dom}(g')$ to Σ . Then $S(g) := S \setminus S(g')$.

These gates correspond to negation gates for Boolean circuits, and $\times, \uplus, \mathbb{C}$ -circuits can be seen as a multivalued analogue of the (smooth) d -D circuits recently studied in probabilistic databases [69]. We can immediately generalize Proposition 5.1 to show:

► **Proposition 6.3.** Given a $\times, \uplus, \mathbb{C}$ -circuit C , and a probabilistic word $p = (p_1, \dots, p_n)$, we can compute $p(C)$ in linear time (assuming unit cost for arithmetic operations).

Hence, we can use $\times, \uplus, \mathbb{C}$ -circuits to show the tractability of $\#pM$. We say that such a circuit captures the n -th slice of a language L by the obvious generalization of the definition of Section 5, and that L admits tractable $\times, \uplus, \mathbb{C}$ -circuits if there is an algorithm which given a length n computes in polynomial time in n a $\times, \uplus, \mathbb{C}$ -circuit that captures the n -th slice of L . By the immediate generalization of Proposition 5.2, this implies tractability for $\#pM$.

While we will use complement gates in this section, we do not know whether they are necessary, i.e., whether $\times, \uplus, \mathbb{C}$ -circuits capture the tractability of $\#pM$ for more languages than \times, \uplus -circuits; we come back to this point in the conclusion. We also note that the support of negation in relation with (u)CFLs seems superficially similar to the notions of conjunctive grammars and Boolean grammars [73], in particular to unambiguous such grammars [72]. However, the semantics of Boolean grammars are different, because they allow in particular conjunctions that range over the entire word (and so are not decomposable). Thus, unambiguous conjunctive grammars include, e.g., the intersections of two uCFGs; but $\#pM$ can be intractable for such languages (see the proof of Proposition 3.6). This suggests that the tractability of unambiguous Boolean grammars (e.g., for parsing) does not extend to counting, so there is no obvious connection to our results.

We will now show that the language of primitive words admits tractable $\times, \uplus, \mathbb{C}$ -circuits in the sense above, and will then show the same for the language PAL^2 .

Primitive words. Let us first define the language L_{prim} of primitive words. We fix an arbitrary alphabet Σ . A word $w \in \Sigma^*$ is *composite* if there exist $u \in \Sigma^*$ and $k \geq 2$ such that $w = u^k$. If w is not composite, then we say that w is *primitive*, i.e., a word is primitive if it is not a proper power of another word. In particular, the empty word is not primitive. The language L_{prim} consists of all primitive words over the alphabet Σ .

We note that, over non-unary alphabets, it is not currently known whether L_{prim} is a CFL or not [51]. However, it is known that L_{prim} is not a uCFL [74, 60], and also that L_{prim} is not a linear CFL [47]. We do not know whether L_{prim} is poly-slicewise-unambiguous, or whether it admits tractable circuits without complementation. Our goal is to show the tractability of $\#pM$ for primitive words (and hence for composite words, by Claim 6.1). Namely:

► **Proposition 6.4.** The problem $\#pM(L_{\text{prim}})$ can be solved in data complexity $O(n^2|\Sigma|)$ up to the cost of arithmetic operations, with n the length of the input word.

We prove this result before turning to the language PAL^2 , which will use a similar technique. Our proofs rely on the standard notion of the (primitive) root and of the order of a word. For the uniqueness and well-definedness of these notions, see [83, Theorem 2.3.4]:

► **Definition 6.5.** For any word $u \in \Sigma^* \setminus \epsilon$, the root of u is the unique primitive word w such that we have $u = w^d$ for a certain integer $d \geq 1$, called the order of u . Note that if w is primitive then $w = u$ and $d = 1$.

Our algorithm to solve $\#pM(L_{\text{prim}})$ will proceed by distinguishing words according to their order. For this, we introduce two families of languages:

► **Definition 6.6.** Let $k > 0$. We write L_k for the set of words over Σ^* of order equal to k . In particular, $L_1 = L_{\text{prim}}$. Note that the L_k are a partition of Σ^* .

We write M_k for the set of non-empty words u over Σ^* that can be written as $u = v^k$ for some $v \in \Sigma^*$ (not necessarily primitive). Note that $M_k = \bigcup_{d \geq 1} L_{dk}$.

To solve $\#pM(L_{\text{prim}})$, we first show that M_k admits tractable \times, \uplus -circuits for all $k \in \mathbb{N}$:

► **Lemma 6.7.** Given two integers $1 \leq k \leq n$, we can compute in time $O(n \times |\Sigma|)$ a \times, \uplus -circuit that captures the n -th slice of M_k .

Proof sketch. The only non-trivial case is when k divides n . Then, letting $d := n/k$, a word belongs to M_k precisely when, for each $1 \leq j \leq d$ all positions on $\{j + pd \mid 0 \leq p \leq k - 1\}$ contain the same letter. So we do a decomposable product across the values of j , and for each value we do a deterministic union across the choice of letter, and then do a decomposable product asserting that this letter occurs at the requisite positions. ◀

Now, we can construct circuits capturing $L_{\text{prim}} = L_1$, and in fact L_k for each k , from the circuits capturing the M_k . Namely, we show the following, which allows us to conclude the proof of Proposition 6.4:

► **Claim 6.8.** Assume that we have pairwise disjoint sets L'_1, \dots, L'_n of words of length n , and define for each $1 \leq i \leq n$ the sets $M'_i := \bigcup_{1 \leq d \leq n/i} L'_{di}$. Given \times, \uplus -circuits C_i computing the n -th slice of M'_i for each $1 \leq i \leq n$, we can build in time $O(n^2 + \sum_i |C_i|)$ a $\times, \uplus, \complement$ -circuit C' computing the n -th slice of L'_1 .

Proof sketch. We do a downwards induction, where we compute a circuit for each L'_k from the circuit for M'_k and the disjoint union of the circuits for L'_{dk} for $d > 1$. This requires us to do a *subset-complement* operation, which can be achieved with complementation. ◀

Concatenations of two palindromes. We now move on to the study of the CFL PAL^2 , where PAL is the language of palindromes on our alphabet Σ . Our main result is:

► **Theorem 6.9.** The problem $\#pM(\text{PAL}^2)$ can be solved in data complexity $O(n^3|\Sigma|)$ up to the cost of arithmetic operations.

It is known that PAL^2 is an inherently ambiguous CFL with infinite ambiguity degree [26], so the tractability of $\#pM(\text{PAL}^2)$ does not follow from Proposition 3.1. It is incidentally easy to show that PAL^2 is not a linear CFL (see [6, Appendix E.3]). The language PAL^2 has been abundantly studied, e.g., in [43, Section 3]; its words have been called *palindrome pairs* [16], or *symmetric words* [17, 34]. The efficient recognition of PAL^k for arbitrary k (e.g., in linear time) was studied in [62]; see also [78]. More relevant to our purposes, the problem of counting the number of words of PAL^2 was studied in [56], but they study unweighted counting, depending only on the word length and on the alphabet size. Our proof of Theorem 6.9 reuses the following word combinatorics result from [56] (using [26], with similar results in [30, Proposition 9] or [31, Lemma 1]):

► **Lemma 6.10** ([56], Corollary 1 (rephrased)). For any $w \in \text{PAL}^2$, the primitive root of w decomposes uniquely into two palindromes.

Using this lemma, the proof of Theorem 6.9 deviates from [56] (which uses generating functions). Namely, we follow an inclusion-exclusion-based reasoning which is similar to that of primitive words but more complicated (we partition words based on their order and on also the offset of their decomposition into two palindromes).

7 Conclusion and Future Work

We have studied the membership problem for probabilistic words to CFLs L , which generalizes the problems of counting how many words of a given length are in L , or how many partial word completions are in L . We have shown that the problem is tractable for unambiguous CFLs, for poly-slicewise-unambiguous languages, and for languages admitting tractable circuits with (decomposable) Cartesian product, (deterministic) disjoint union, and negation. We have shown that the problem is intractable already for unions of two linear uCFLs, or for some languages recognized by restricted kinds of counter automata.

Our work does not give a complete dichotomy between tractable and intractable CFLs for $\#pM$. This is not so surprising in hindsight: via the technique of Greibach [41] we can easily show the undecidability of the *meta-problem*. Namely, given a CFG G , it is conditionally undecidable to determine whether $\#pM(L(G))$ is tractable, in fact already for linear CFGs:

► **Proposition 7.1.** *Assume that $FP \neq \#P$. Then the following problem is undecidable: given a linear CFG Γ , determine whether $\#pM(L(\Gamma))$ is in FP .*

In fact, the same proof shows that it is (conditionally) undecidable to decide, given a CFL G , whether G is unambiguous, poly-slicewise-unambiguous, or whether it admits tractable circuits (with or without complement). Indeed, in the proof of the result above, in one case the language to which we reduce is regular so it falls in all these classes, and in the other case the language is intractable for $\#pM$ so conditionally not in these classes. Thus, it is an interesting question whether showing $\#P$ -hardness of $\#pM$ can be a useful method to show (conditionally) that a language is inherently ambiguous (compared to other methods [60]), or that it is not poly-slicewise-unambiguous (compared to [68]), or that it does not have tractable circuits.

Despite Proposition 7.1, one natural question for future work is to classify more CFLs. Does the tractability of PAL^2 extend to greater values of the *palindromic length* [36, 16], e.g., PAL^3 , or more generally *palstars* [59]? We can also see the languages L_1 of Proposition 4.8 or L'_0 of Proposition 3.5 as variants of palindrome concatenations, with “markers” (the $\#$ ’s) and “gaps” (the Σ^* ’s). Does intractability still hold without these features? What about the language on $\{0, 1, \#\}$ defined as $\{\Sigma^* \# u \# \Sigma^* \# u^R \# \Sigma^* \mid u \in \{0, 1\}^*\}$? We also do not know how tractability is affected if we instead study the restricted case of $\#pM$ corresponding to the counting of completions of partial words. Another intriguing example is *Shamir’s language* [60, 84]: $\#pM$ for this language amounts to computing, given two probabilistic words p and p' , the total probability of the outcomes u, u' such that u is a factor of u' .

A broader direction is to understand which CFLs admit tractable circuits in various circuit classes. For instance, when can we have tractable *decision diagrams* (e.g., OBDDs), or tractable *structured circuits* [75]? This relates to a line of work in probabilistic databases that asks which queries admit tractable circuits in various representations [52], and more broadly asks about the relative power of circuit formalisms. One obvious question is whether $\times, \uplus, \complement$ -circuits capture the tractability of $\#pM$ for more languages than \times, \uplus -circuits (i.e., whether they are more concise): this relates to the open problem of whether d-DNNFs are closed under complementation (see [29, 69, 91]). Another question is whether there are CFLs that are tractable for $\#pM$ but do not admit tractable circuits: this relates to the question of whether such circuits can express *inclusion-exclusion* [7].

Yet another direction concerns the study of $\#pM$ in the lens of combined complexity, where the target language is part of the input. Some of our tractability results (e.g., for uCFGs) hold in combined complexity, but other questions remain. For instance, it appears likely that $\#pM$ is also tractable in combined complexity for k -ambiguous automata for any

fixed $k \in \mathbb{N}$ (by adaptation of the methods of [85, 77]), and maybe for tree automata as well [80]; but the problem is intractable in combined complexity for NFAs [2] (though it is approximable [8], like for CFLs [67]).

Finally, a much broader question is to ask about the relationship of $\#pM$ to non-probabilistic membership. Indeed, a central question in CFLs is to characterize the fine-grained (data) complexity of the (non-probabilistic) membership problem for each specific CFL L , i.e., the best achievable complexity in the input word. Lower bounds are known for some grammars [1], and the ability to code hard problems in $\#pM(L)$ for a language L might relate to the ability to code hard problems for parsing; so classifying the tractable CFLs for $\#pM$ may be a way to shed light on the complexity of (non-probabilistic) CFG parsing.

References

- 1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant’s parser. *SIAM Journal on Computing*, 47(6), 2018. doi:10.1137/16M1061771.
- 2 Carme Àlvarez and Birgit Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107(1), 1993. doi:10.1016/0304-3975(93)90252-0.
- 3 Antoine Amarilli. Context-free grammars where every word has exactly two derivation trees. Theoretical Computer Science Stack Exchange, 2025. URL: <https://cstheory.stackexchange.com/q/55726>.
- 4 Antoine Amarilli and Florent Capelli. Tractable circuits in database theory. *ACM SIGMOD Record*, 53(2), 2024. doi:10.1145/3685980.3685982.
- 5 Antoine Amarilli, Louis Jachiet, Martín Muñoz, and Cristian Riveros. Efficient enumeration algorithms for annotated grammars. In *PODS*, 2022. doi:10.1145/3517804.3526232.
- 6 Antoine Amarilli, Mikaël Monet, Paul Raphaël, and Sylvain Salvati. On the complexity of language membership for probabilistic words. *arXiv preprint arXiv:2510.08127*, 2025. Full version of this article. doi:10.48550/arXiv.2510.08127.
- 7 Antoine Amarilli, Mikaël Monet, and Dan Suciu. The non-cancelling intersections conjecture. Preprint: <https://arxiv.org/abs/2401.16210>, 2024. doi:10.48550/arXiv.2401.16210.
- 8 Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. $\#NFA$ admits an FPRAS: Efficient enumeration, counting, and uniform generation for logspace classes. *Journal of the ACM*, 68(6), 2021. doi:10.1145/3477045.
- 9 Brenda S. Baker and Ronald V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8(3), 1974. doi:10.1016/S0022-0000(74)80027-9.
- 10 J. Berstel and L. Boasson. Context-free languages. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol B*, chapter 2. MIT Press, 1990. URL: <https://www-igm.univ-mlv.fr/~berstel/Articles/1990HandbookCf1.pdf>.
- 11 Jean Berstel and Luc Boasson. Partial words and a theorem of Fine and Wilf. *Theoretical Computer Science*, 218(1), 1999. doi:10.1016/S0304-3975(98)00255-2.
- 12 Alberto Bertoni, Danilo Bruschi, and Massimiliano Goldwurm. Ranking and formal power series. *Theoretical computer science*, 79(1), 1991. doi:10.1016/0304-3975(91)90144-Q.
- 13 Alberto Bertoni, Massimiliano Goldwurm, and Nicoletta Sabadini. The complexity of computing the number of strings of given length in context-free languages. *Theoretical Computer Science*, 86(2), 1991. doi:10.1016/0304-3975(91)90023-U.
- 14 Alberto Bertoni, Massimiliano Goldwurm, and Massimo Santini. Random generation for finitely ambiguous context-free languages. *RAIRO-Theoretical Informatics and Applications*, 35(6), 2001. doi:10.1051/ITA:2001128.
- 15 Francine Blanchet-Sadri, Robert Mercas, and Geoffrey Scott. Counting distinct squares in partial words. *Acta Cybernetica*, 19(2), 2009. URL: <https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3777>.

- 16 Adam Borchert and Narad Rampersad. Words with many palindrome pair factors. *The Electronic Journal of Combinatorics*, 2015.
- 17 Srečko Brlek, Sylvie Hamel, Maurice Nivat, and Christophe Reutenauer. On the palindromic complexity of infinite words. *International Journal of Foundations of Computer Science*, 15(02), 2004. doi:10.1142/S012905410400242X.
- 18 Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8), 1986. doi:10.1109/TC.1986.1676819.
- 19 Michaël Cadilhac, Arka Ghosh, Guillermo A. Pérez, and Ritam Raha. Parikh one-counter automata. *Information and Computation*, 2025. doi:10.1016/J.IC.2025.105322.
- 20 Panagiotis Charalampopoulos, Costas S. Iliopoulos, Chang Liu, and Solon P. Pissis. Property suffix array with applications in indexing weighted sequences. *Journal of Experimental Algorithmics (JEA)*, 25, 2020. doi:10.1145/3385898.
- 21 Manolis Christodoulakis, Costas S. Iliopoulos, Laurent Mouchard, and Kostas Tsichlas. Pattern matching on weighted sequences. In *CompBioNets 2004*, 2004.
- 22 Lorenzo Clemente. On the complexity of the universality and inclusion problems for unambiguous context-free grammars, 2020. Preprint: <https://arxiv.org/abs/2008.04667>.
- 23 Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Separability of reachability sets of vector addition systems. In *STACS*, 2017.
- 24 Thomas Colcombet. Unambiguity in automata theory. In *International Workshop on Descriptive Complexity of Formal Systems*, 2015. doi:10.1007/978-3-319-19225-3_1.
- 25 Christophe Costa Florencio, Jonny Daenen, Jan Ramon, Jan Van den Bussche, and Dries Van Dyck. Naive infinite enumeration of context-free languages in incremental polynomial time. *Journal of Universal Computer Science*, 21(7), 2015. URL: http://www.jucs.org/jucs_21_7/naive_infinite_enumeration_of.
- 26 J.P. Crestin. Un langage non ambigu dont le carré est d'ambiguïté non-bornée. In *ICALP*, 1972.
- 27 Wojciech Czerwiński and Georg Zetsche. An approach to regular separability in vector addition systems. In *LICS*, 2020.
- 28 Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2), 2001. doi:10.3166/JANCL.11.11-34.
- 29 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17, 2002. doi:10.1613/JAIR.989.
- 30 Aldo de Luca. On some combinatorial problems in free monoids. *Discrete Mathematics*, 38(2-3), 1982. doi:10.1016/0012-365X(82)90290-4.
- 31 Aldo de Luca and Filippo Mignosi. Some combinatorial properties of sturmian words. *Theoretical Computer Science*, 136(2), 1994.
- 32 Pál Dömösi. Unusual algorithms for lexicographical enumeration. *Acta Cybernetica*, 14(3), 2000. URL: <https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3539>.
- 33 Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2), 1970. doi:10.1145/362007.362035.
- 34 Gabriele Fici, Jeffrey Shallit, and Jamie Simpson. Some remarks on palindromic periodicities. Preprint: <https://arxiv.org/pdf/2407.10567>, 2024.
- 35 Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. Counter machines and counter languages. *Mathematical systems theory*, 2(3), 1968. doi:10.1007/BF01694011.
- 36 Anna E. Frid, Svetlana Puzynina, and Luca Q. Zamboni. On palindromic factorization of words. *Advances in Applied Mathematics*, 50(5), 2013. doi:10.1016/J.AAM.2013.01.002.
- 37 Seymour Ginsburg and Sheila Greibach. Deterministic context free languages. In *SWCT*, 1965. doi:10.1109/FOCS.1965.7.
- 38 Seymour Ginsburg and Edwin H Spanier. Bounded algol-like languages. *Transactions of the American Mathematical Society*, 113(2), 1964.

- 39 Vivek Gore, Mark Jerrum, Sampath Kannan, Z. Sweedyk, and Steve Mahaney. A quasi-polynomial-time algorithm for sampling words from a context-free language. *Information and Computation*, 134(1), 1997. doi:10.1006/INCO.1997.2621.
- 40 Erich Grädel, Yuri Gurevich, and Colin Hirsch. The complexity of query reliability. In *PODS*, 1998. doi:10.1145/275487.295124.
- 41 Sheila Greibach. A note on undecidable properties of formal languages. *Mathematical systems theory*, 2, 1968. doi:10.1007/BF01691341.
- 42 Sheila A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7(3), 1978. doi:10.1016/0304-3975(78)90020-8.
- 43 Chuan Guo, Jeffrey Shallit, and Arseny M. Shur. On the combinatorics of palindromes and antipalindromes. Preprint: <https://arxiv.org/abs/1503.09112>, 2015. arXiv:1503.09112.
- 44 J. Hartmanis. Context-free languages and turing machine computations. In *Proceedings of Symposia in Applied Mathematics*, 1967.
- 45 Thomas N. Hibbard and Joseph Ullian. The independence of inherent ambiguity from complementedness among context-free languages. *Journal of the ACM*, 13(4), 1966. doi:10.1145/321356.321366.
- 46 Timothy Hickey and Jacques Cohen. Uniform random generation of strings in a context-free language. *SIAM Journal on Computing*, 12(4), 1983. doi:10.1137/0212044.
- 47 Sándor Horváth. Strong interchangeability and nonlinearity of primitive words. *Algebraic Methods in Language Processing*, 1995.
- 48 Dung T. Huynh. The complexity of ranking. In *Structure in Complexity Theory*, 1988.
- 49 Dung T. Huynh. The complexity of ranking simple languages. *Mathematical Systems Theory*, 23(1), 1990. doi:10.1007/BF02090763.
- 50 Lucian Ilie, Grzegorz Rozenberg, and Arto Salomaa. A characterization of poly-slender context-free languages. *RAIRO-Theoretical Informatics and Applications*, 34(1), 2000. doi:10.1051/ITA:2000100.
- 51 Masami Ito and Pal Domosi. *Context-free languages and primitive words*. World Scientific, 2014.
- 52 Abhay Jha and Dan Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. *Theory of Computing Systems*, 2013.
- 53 Makoto Kanazawa. Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. *J. Log. Comput.*, 26(5), 2016. doi:10.1093/LOGCOM/EXU043.
- 54 Makoto Kanazawa, Gregory M. Kobele, Jens Michaelis, Sylvain Salvati, and Ryo Yoshinaka. The failure of the strong pumping lemma for multiple context-free languages. *Theory Comput. Syst.*, 55(1), 2014. doi:10.1007/S00224-014-9534-Z.
- 55 Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and system Sciences*, 3(2), 1969. doi:10.1016/S0022-0000(69)80011-5.
- 56 R. Kemp. On the number of words in the language $\{w \in \Sigma^* \mid w = w^R\}^2$. *Discrete Mathematics*, 40(2), 1982.
- 57 Benny Kimelfeld, Wim Martens, and Matthias Niewerth. A formal language perspective on factorized representations. In *ICDT*, 2025. doi:10.4230/LIPIcs.ICDT.2025.20.
- 58 Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *International Colloquium on Automata, Languages, and Programming*, 2003. doi:10.1007/3-540-45061-0_54.
- 59 Donald E. Knuth, James H. Morris, Jr, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2), 1977. doi:10.1137/0206024.
- 60 Florent Koechlin. New analytic techniques for proving the inherent ambiguity of context-free languages. In *FSTTCS*, volume 250, 2022.
- 61 Hans-Peter Kolb, Jens Michaelis, Uwe Mönnich, and Frank Morawietz. An operational and denotational approach to non-context-freeness. *Theoretical Computer Science*, 293(2), 2003. doi:10.1016/S0304-3975(01)00348-6.

- 62 Dmitry Kosolobov, Mikhail Rubinchik, and Arseny M. Shur. Pal^k is linear recognizable online. In *SOFSEM*, 2015. doi:10.1007/978-3-662-46078-8_24.
- 63 Martin Lange and Hans Leiß. To CNF or not to CNF? an efficient yet presentable version of the CYK algorithm. *Informatica Didactica*, 8(2009), 2009. URL: <http://www.informatica-didactica.de/cmsmadesimple/index.php?page=LangeLeiss2009>.
- 64 Harry G. Mairson. Generating words in a context-free language uniformly at random. *Information Processing Letters*, 49(2), 1994. doi:10.1016/0020-0190(94)90033-7.
- 65 Erkki Mäkinen. On lexicographic enumeration of regular and context-free languages. *Acta Cybernetica*, 13(1), 1997. URL: <https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3479>.
- 66 Florin Manea and Cătălin Tisceanu. Hard counting problems for partial words. In *International Conference on Language and Automata Theory and Applications*, 2010.
- 67 Kuldeep S. Meel and Alexis de Colnet. #CFG and #DNNF admit FPRAS. Preprint: <https://arxiv.org/abs/2406.18224>, 2024. doi:10.48550/arXiv.2406.18224.
- 68 Stefan Mengel and Harry Vinnal-Smeeth. A lower bound on unambiguous context free grammars via communication complexity. *PACMOD*, 3(2), 2025. doi:10.1145/3725225.
- 69 Mikaël Monet. Solving a special case of the intensional vs extensional conjecture in probabilistic databases. In *PODS*, 2020. doi:10.1145/3375395.3387642.
- 70 Mikaël Monet. *Combined Complexity of Probabilistic Query Evaluation*. PhD thesis, Télécom ParisTech, 2018.
- 71 Cristopher Moore. Lecture notes on automata, languages, and grammars, 2012. URL: <https://sites.santafe.edu/~moore/500/automata-notes.pdf>.
- 72 Alexander Okhotin. Unambiguous boolean grammars. *Information and Computation*, 206(9-10), 2008. doi:10.1016/J.IC.2008.03.023.
- 73 Alexander Okhotin. Conjunctive and boolean grammars: The true general case of the context-free grammars. *Computer Science Review*, 9, 2013. doi:10.1016/J.COSREV.2013.06.001.
- 74 Holger Petersen. The ambiguity of primitive words. In *STACS*, 1994. doi:10.1007/3-540-57785-8_181.
- 75 Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, volume 8, 2008. URL: <http://www.aaai.org/Library/AAAI/2008/aaai08-082.php>.
- 76 Jakub Radoszewski and Tatiana Starikovskaya. Streaming k-mismatch with error correcting and applications. *Information and Computation*, 271, 2020. doi:10.1016/J.IC.2019.104513.
- 77 Ritam Raha and Nathanaël Fijalkow. The universality problem for automata with bounded ambiguity. “Games Automata Play” blog, 2018. . Accessed: September 12, 2025. URL: https://games-automata-play.github.io/blog/universality_finitely_ambiguous/.
- 78 Mikhail Rubinchik and Arseny M. Shur. Palindromic k-factorization in pure linear time. In *MFCS*, 2020. doi:10.4230/LIPIcs.MFCS.2020.81.
- 79 William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *STOC*, 1978. doi:10.1145/800133.804357.
- 80 Helmut Seidl. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3), 1990. doi:10.1137/0219027.
- 81 Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2), 1991. doi:10.1016/0304-3975(91)90374-B.
- 82 Ichiro Semba. Generation of all the balanced parenthesis strings in lexicographical order. *Information Processing Letters*, 12(4), 1981. doi:10.1016/0020-0190(81)90098-3.
- 83 Jeffrey Shallit. *A second course in formal languages and automata theory*. Cambridge University Press, 2008.
- 84 Eliahu Shamir. Some inherently ambiguous context-free languages. *Information and Control*, 18(4), 1971. doi:10.1016/S0019-9958(71)90455-4.

- 85 Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3), 1985. doi:10.1137/0214044.
- 86 D. Suci, D. Olteanu, and C. Koch. *Probabilistic Databases*. Synthesis digital library of engineering and computer science. Morgan & Claypool Publishers, 2011.
- 87 Leslie G. Valiant. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, 10(2), 1975. doi:10.1016/S0022-0000(75)80046-8.
- 88 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3), 1979. doi:10.1137/0208032.
- 89 Leslie G. Valiant and Michael S. Paterson. Deterministic one-counter automata. *Journal of Computer and System Sciences*, 10(3), 1975. doi:10.1016/S0022-0000(75)80005-5.
- 90 Moshe Y. Vardi. The complexity of relational query languages. In *STOC*, 1982.
- 91 Harry Vinall-Smeeth. Structured d-DNNF is not closed under negation. In *IJCAI*, 2024. URL: <https://www.ijcai.org/proceedings/2024/398>.
- 92 Klaus Wich. *Ambiguity functions of context-free grammars and languages*. PhD thesis, Universität Stuttgart, 2005.
- 93 Tomoyuki Yamakami. Power of counting by nonuniform families of polynomial-size finite automata. *Information and Computation*, 2025. doi:10.1016/J.IC.2025.105372.