A Lower Bound on Unambiguous Context Free Grammars via Communication Complexity

STEFAN MENGEL, Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), France
HARRY VINALL-SMEETH, Technische Universität Ilmenau, Germany

Motivated by recent connections to factorised databases, we analyse the efficiency of representations by context free grammars (CFGs). Concretely, we prove a recent conjecture by Kimelfeld, Martens, and Niewerth (ICDT 2025), that for finite languages representations by general CFGs can be doubly-exponentially smaller than those by unambiguous CFGs. To do so, we show the first exponential lower bounds for representation by unambiguous CFGs of a finite language that can efficiently be represented by ambiguous CFGs. Our proof first reduces the problem to proving a lower bound in a non-standard model of communication complexity. Then, we argue similarly in spirit to a recent discrepancy argument to show the required communication complexity lower bound. Our result also implies that a finite language may admit an exponentially smaller representation as a nondeterministic finite automaton than as an unambiguous CFG.

CCS Concepts: • **Theory of computation → Grammars and context-free languages**; **Database theory**.

Additional Key Words and Phrases: formal languages, context free grammar, ambiguity, communication complexity, factorised representations

ACM Reference Format:
Stefan Mengel and Harry Vinall-Smeeth. 2025. A Lower Bound on Unambiguous Context Free Grammars via Communication Complexity. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 88 (May 2025), 19 pages. https://doi.org/10.1145/3725225

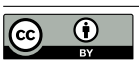
1 Introduction

Over recent years, the succinct representation of query results has become a prominent topic within database theory. This raises a fundamental question: how do different representation formats compare to one another? In this paper, we compare context free grammars (CFG) to their unambiguous variant (uCFG). In a nutshell, we show that enforcing unambiguity—which is useful for applications like counting and enumeration—comes at a heavy price in terms of succinctness.

At this point some readers may wonder what CFGs have to do with representing query answers. In fact, recent work by Kimelfeld, Martens and Niewerth [20] observed a close connection between CFGs and *d-representations*—a form of factorised representations introduced over a decade ago by Olteanu and Závodný in a landmark paper [29]. To be precise, CFGs representing finite languages are isomorphic to d-representations in the *unnamed perspective*, see [20] for details.

Let us highlight two nice features of d-representations. Firstly, given a query and a database we can often directly compute a d-representation which moreover, may be exponentially smaller than the materialisation of all query answers [29]. And secondly, we can carry out many algorithms directly on d-representations [4]. These representations, and close variants thereof, have been used in various settings e.g. databases under updates [5, 27], representing provenance [28] and machine

Authors' Contact Information: Stefan Mengel, Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), Lens, France, mengel@cril.fr; Harry Vinall-Smeeth, Technische Universität Ilmenau, Ilmenau, Germany, harry.vinall-smeeth@tu-ilmenau.de.



ACM 2836-6573/2025/5-ART88
https://doi.org/10.1145/3725225

learning [34]. More broadly, d-representations can be placed within the framework of circuits originating in *knowledge compilation*, a sub-field of artificial intelligence concerning the efficient representation of various forms of knowledge and data [13, 26]. Techniques from knowledge compilation have been widely deployed in database theory in recent years, see [2] for a survey. Moreover, the circuits corresponding to uCFGs have been used in the context of enumeration [1].

From an algorithmic perspective, uCFGs offer several advantages over CFGs. For example, it is well know that while counting is in polynomial time for uCFGs, it is #P-complete for CFGs. Similarly, uCFGs allow for more efficient enumeration, e.g. via [1, 24]. Different variants of CFGs have also been used in information extraction [3, 30] and unambiguity is also crucual there. But are there finite languages which are accepted by a small CFG and not by any small uCFG? This is a question left open by Kimelfeld, Martens and Niewerth [20]. They conjecture a double exponential separation for CFGs for a concrete language. However, they say '[t]o the best of our knowledge, the literature does not yet have well-developed methods for proving size lower bounds for uCFGs.' In this paper, we develop such methods and prove the conjecture from [20] via the following theorem.

Theorem 1. *For every $n \in \mathbb{N}$, there is finite language L_n over a binary alphabet in which all words have length $2n$ such that:*

(1) *L_n is accepted by a CFG of size $\Theta(\log(n))$,*
(2) *L_n is accepted by a non-deterministic finite automaton of size $\Theta(n)$ and*
(3) *every uCFG accepting L_n has size $2^{\Omega(n)}$.*

Moreover, we may take L_n to be exactly the language from the conjecture of Kimelfeld, Martens and Niewerth [20]. This language is very natural: it consists exactly of those strings of length $2n$ over the alphabet $\{a, b\}$ such that there are two a symbols at distance exactly n from one-another.

Conceptually, our techniques not only answer the call for size lower bound methods for uCFGs but contribute more generally to our understanding of the role of unambiguity within computation. This is an area which has received quite some attention within the formal language community. For example, unambiguity within automata theory was the subject of a recent Dagstuhl seminar [12], see [11] for a survey on this topic. It is not only in the context of CFGs that unambiguity has proved difficult to deal with. One reason for this is probably that unambiguity is a semantic property, not a syntactic one, making it hard to 'get your hands on' in a proof.

As a result, basic questions about unambiguity remain open and others have only recently been resolved. For example, whether unambiguous finite automata admit polynomial time complementation was only relatively recently answered in the negative [16, 32]. Similarly, within knowledge compilation, whether d-DNNF circuits admit polynomial time negation has been open for over twenty years and even the special case of structured d-DNNF has only recently been resolved [42].

On the Naturalness of L_n. Let us quickly comment on the naturalness of the language L_n that we consider. Of course, the lower bound on L_n is not directly a result in a practical setting. However, it encapsulates in a prototypical fashion a problem of unambiguous representations: how can one unambiguously deal with unions of highly non-disjoint sets? This is the underlying difficulty for many problems, e.g. the hardness of counting for NFAs and CFGs, and here we show that it is also hard for a type of succinct representation.

An advantage of studying L_n as a very bare-bones version of the problem is that one can hopefully then use it to show hardness of other, more practical problems. This is the reason why different versions of Set Disjointness (essentially a complement of L_n) have been studied extensively in the communication complexity literature and then used in many lower bounds, see e.g. the applications for lower bounds on data structures, streaming algorithms, distributed computing, and the extension complexity of polytopes in [31] and the survey [39].

One somewhat natural concrete setting for the use of L_n in information extraction might be the following: consider data in a CSV file with fixed columns from which we want to extract all pairs of lines that have identical entries in at least one column from a column set S. This can easily be modelled with the CFG formalisms proposed for information extraction in [3, 30], but if the algorithm requires unambiguous CFGs (as those in [3, 30] do), then an easy reduction from L_n shows that any such grammar must be of exponential size in the number of considered columns in S. This lower bound remains true if instead of equality we require other natural comparison of the columns, say lexicographic order, similarity measures, and so on.

Thus, while L_n is not itself natural from a data management perspective, we still feel that it might have practical applications. We will discuss these aspects in future versions of our paper.

Proof Outline. As mentioned, we consider the language L_n from [20]. There, it was already shown that for some n the language L_n admits a CFG of size $\Theta(\log(n))$; for completeness we give such a grammar for all n in Appendix A (see also the discussion in Section 2). It was also remarked in [20] that L_n admits a nondeterministic finite automaton of size $\Theta(n)$; the idea is that the automaton first nondeterministically 'guesses' the positions of the matching a symbols and then verifies this guess. Therefore, to prove Theorem 1 it suffices to prove the lower bound (3) for L_n.

Thus far the best tools to analyse unambiguity seem to come from communication complexity, see [31] for an introduction. Our paper is no different. We make a connection to communication complexity, motivated by work in knowledge compilation [6], by relating the size of uCFGs and so-called *rectangles* (see Section 3). To be precise, we show that if we cannot split up a language L into a small number of disjoint rectangles, then L does not admit a small uCFG. Since rectangles have been studied extensively in communication complexity, one may hope to use results from that area as a 'black box.' Unfortunately, this cannot be done since our setting is more general than the one normally studied in communication complexity. The main technical contribution of this paper is that we overcome this difference in settings.

To do this we build on an approach from [40], by defining disjoint $A, B \subset L_n$, such that any rectangle contains roughly the same number of words from A and B. On the other hand, L_n contains many more words from A than B, so we need many disjoint rectangles to cover the whole language. This type of argument is known as a *discrepancy argument* in the communication complexity literature, see e.g. the textbook description [31, Chapter 5].

Related Work. It has long been known that for *infinite* languages CFGs are far more succinct than uCFGs. More than 40 years ago, Schmidt and Szymanski [38] showed that in general the size of a uCFG representing a language L cannot be bounded by any computable function in the size of a CFG for the same language L. However, this result is not true for *finite* languages, which are more interesting in the database setting that we consider. There, our doubly exponential separation is optimal, in the sense that every CFG accepting a finite language can be transformed into an equivalent uCFG with at most a double-exponential blow-up [20].

This is not the first time that different representations of finite languages have been compared qualitatively. In fact, since the influential paper [7], the representation of finite languages has been an active research field, see e.g. [8, 9, 17]. Actually, in [7] uCFGs and CFGs are also compared for succinctness. However, they measure the size of a grammar in terms of the total number of rules whereas we measure the sum of the sizes of all rules. Our size measure corresponds to the size of factorised representations while the measure in [7] is far smaller: a constant number of rules can create languages that need linear size factorised representations. Consequently, our results, are quite different. Perhaps the paper closest in flavour to ours is by Filmus [15], where lower bounds on the sizes of CFGs for various finite languages are shown, with respect to our size model.

Another influential research strand considering representation by CFGs is that of *grammar based compression* [18, 19, 43]. There one aims to find a small CFG representing a single word w; we may think of this process as compressing a long document.[1] Algorithms can then be applied directly to the compressed document [21]. This paradigm has become increasingly influential in database theory in recent years, see e.g. [22, 25, 35–37]. Since we are interested in CFGs that represent many strings our results are quite different from this line of work.

2 Preliminaries

We write $[n] := \{1, 2, \dots, n\}$ and for integers i, j with $i \leq j$, we write $[i, j] := \{\ell \in \mathbb{Z} \mid i \leq \ell \leq j\}$. For a set U we write $\mathcal{P}(U)$ to denote its power set, i.e., the set containing every subset of U. Moreover, let U and V be sets of sets, such that $A \cap B = \varnothing$ for all $A \in U$, $B \in V$. Then we may identify each pair (A, B) with the set $A \cup B$ and so reflecting this we will write $U \times V := \{A \cup B \mid A \in U, B \in V\}$.

We assume that the reader is familiar with some basics of context free grammars, see e.g. [41]. Let Σ be a set of symbols which we call an *alphabet*. A *word* over σ is a finite string of symbols from Σ. For a word w we write $|w|$ to denote its length. We write Σ^* for the set of all words over Σ. Then a *language over* Σ is defined to be a set $L \subseteq \Sigma^*$. This paper is only concerned with *finite languages* which means that L is a finite set.

Definition 2. A *context free grammar* (CFG) is a four-tuple $G = (\Sigma, N, R, S)$ where

- Σ is a finite set of symbols called the *terminals*,
- N is a finite set called the *non-terminals*,
- R is a set of *rules* of the form $A \to W$ where $A \in N$ and $W \in (\Sigma \cup N)^*$, and
- $S \in N$ is the *start symbol*.

When we have two rules $A \to W$ and $A \to W'$ with $W \neq W'$, we sometimes write them more compactly as $A \to W \mid W'$. This notation always has to be interpreted as two rules and not as a single, more complex rule.

CFGs generate a language over the alphabet Σ as follows. Let $A \to W$ be a rule in R and let $u, v \in \Sigma^*$. Then we say that uAv *yields* uWv, in symbols $uAv \to uWv$. For $W, W' \in (\Sigma \cup N)^*$, we say that W derives W' if there is a finite sequence $(w_i)_{i \in [k]}$ such that $w_1 = W$, $w_k = W'$ and $w_i \to w_{i+1}$ for all $i \in [k-1]$. We write $W \to^* W'$. Then the language of a grammar G is defined as $L(G) := \{w \in \Sigma^* \mid S \to^* w\}$. We also say that the CFG *accepts* $L(G)$. We define the size of a CFG to be $|G| := \sum_{(A \to W) \in R} |W|$. Note that—in contrast to infinite languages—every finite language is accepted by some context free grammar, so the concern of our work is not expressivity of CFGs but succinctness of representation by CFGs.

We now give an example that we will use for our main result.

Example 3. Define the language

$$L_n := \{(a+b)^k a (a+b)^{n-1} a (a+b)^{n-1-k} \mid k \leq n-1\}$$

consisting of words of length $2n$ that contain two a symbols with a word of length exactly $n-1$ between them. In [20] the following small CFGs are defined which accept L_{2^n+1}.

[1] Note, that in this context CFGs are often called *straight-line programs*.

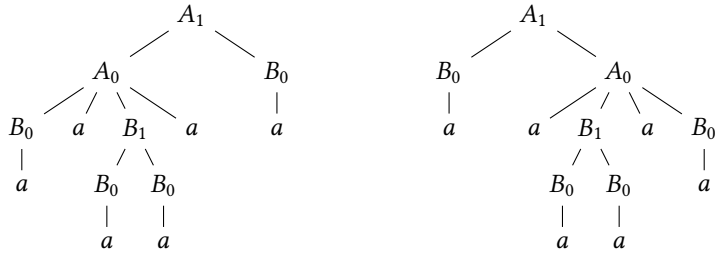


Fig. 1. Two different parse trees for the word *aaaaaa* for the grammar of Example 3.

Let G_n be the grammar with terminals $\{a, b\}$, non-terminals $\{A_i, B_i\}_{0 \le i \le n}$, start symbol A_n and rules,

$$\begin{aligned} A_i &\to B_{i-1}A_{i-1} \mid A_{i-1}B_{i-1} && \text{for } i \in [n] \\ A_0 &\to B_0 a B_n a \mid a B_n a B_0 \\ B_i &\to B_{i-1}B_{i-1} && \text{for } i \in [n] \\ B_0 &\to a \mid b. \end{aligned}$$

Then G_n has size $\Theta(n)$ and accepts L_{2^n+1} see [20] for details. ◁

In fact the above grammar can be adapted to give a $\Theta(\log(n))$ sized CFG accepting L_n for every n. For completeness we give such a grammar in Appendix A.

Each derivation in a context free-grammar may be associated with a tree, called a parse-tree, in the natural way (see Figure 1). We say that a CFG is *unambiguous* if every $w \in L(G)$ has a unique parse tree; equivalently this means that every word in $L(G)$ has a unique derivation in G. We call such a grammar a uCFG.

Example 4. The grammar from Example 3 is not unambiguous; see Figure 1 for two different parse trees of the same word. So let us construct an unambiguous grammar for L_n.

In a first step, it will be useful to generate all possible words of a given length i. To this end, we introduce non-terminals $C_1, \ldots, C_n$ and rules

$$\begin{aligned} C_i &\to aC_{i-1} \mid bC_{i-1} && \text{for } i \in [n], i > 1 \\ C_1 &\to a \mid b. \end{aligned}$$

Clearly, any derivation from any C_i is unambiguous. Next, for every word $w \in \Sigma^{\le n}$, we introduce a non-terminal A_w and a rule

$$A_w \to w.$$

One way to enforce unambiguity for L_n is to ensure that each derivation of a word w in L_n determines the first pair of positions containing a at distance n in w. To this end, we introduce for every $i \in [n]$ a way to generate words u such that i is the smallest integer for which the letters at position i and $i + n$ in u are both a. For every $i < n$, we thus introduce a non-terminal A_i and for every $w \in \Sigma^{i-1}$ the rule

$$A_i \to A_w a C_{n-i} A_{\bar{w}} a C_{n-i},$$

where $\bar{w}$ is the word we get from w by complementing all its letters, i.e., by flipping all occurrences of a to b and vice-versa. For $i = n$, we introduce A_n and the rules

$$A_n \to A_w a A_{\bar{w}} a \tag{1}$$

for all $w \in \Sigma^{n-1}$. Clearly, all derivations from any A_i are unambiguous. It only remains to add a starting symbol S and the rules

$$S \to A_1 \mid \ldots \mid A_n.$$

Note that the grammar we have just constructed accepts L_n but it has exponential size in n; in particular, there are exponentially many rules in (1). So the unambiguous grammar here is of double exponential size with respect to the ambiguous one from Example 3. Our main result will show that this is unavoidable because every grammar for L_n has exponential size in n. ◁

We say that a CFG is in *Chomsky normal form* if all rules are of the form $A \to BC$ or $A \to a$ where A, B, C are non-terminals and a is a terminal [10]. It is well-known that any CFG G can be transformed into an equivalent one G' in Chomsky normal form, such that $|G'| \leq |G|^2$. Therefore, from now on we always assume that CFGs are in Chomsky normal form. Moreover, we assume that we don't have any redundant non-terminals, i.e. that every non-terminal A appears in at least one parse tree generated by the grammar. If this is not the case, we get a smaller grammar accepting the same language by deleting all rules containing A.

3 From Unambiguous CFGs to Rectangle Covers

In this section, we show how one can prove lower bounds for unambiguous CFGs by means of communication complexity. In fact, we focus on CFGs whose languages are finite and, in particular, whose words are all of the same length. The following definition is crucial.

Definition 5 (Rectangle)**.** We say that a language L with words of length n is a *rectangle* if there are numbers n_1, n_2, n_3 such that:

$$L = \bigcup_{\substack{w_1 w_3 \in L_1: \\ |w_1|=n_1, |w_3|=n_3}} \{w_1\} \times L_2 \times \{w_3\} \text{ where } L_1 \subseteq \Sigma^{n_1+n_3}, L_2 \subseteq \Sigma^{n_2}.$$

We also say that L is a rectangle with *parameters* $(L_1, L_2, n_1, n_2, n_3)$. We say that a rectangle is *balanced* if and only if $\frac{n}{3} \leq n_2 \leq \frac{2n}{3}$.

Example 6. Consider the language

$$L_n^* := a^{n/2}(a+b)^n a^{n/2}$$

of all words of length exactly $2n$ which begin and end with $n/2$ consecutive a symbols. Then L_n^* is a balanced rectangle. This follows by setting $n_1 = n_3 = n/2$, $n_2 = n$, $L_1 = \{a^n\}$ and $L_2 = (a+b)^n$. ◁

It is easy to see that every language can be written as a union of balanced rectangles since any language containing a single word is a balanced rectangle. The aim of this section is to show that the number of rectangles needed to cover a language is closely related to the size of CFGs accepting the language. This is formalised by the following proposition.

Proposition 7. *Let L be a language in which all words have length n and which is accepted by a CFG G. Then L can be written as*

$$L := \bigcup_{i \in [\ell]} L_i \tag{2}$$

where every L_i is a balanced rectangle and $\ell \leq n|G|$. Moreover, if G is unambiguous, then the union in (2) is disjoint.

With Proposition 7 in hand, to prove that a language does not admit a small uCFG it suffices to show that whenever L is equal to a disjoint union of rectangles, the number of rectangles in

the union must be big. This gives a lower bound technique analogous to those which have been successfully deployed in *knowledge compilation*, see for example [6].

Example 8. L_n can be written as a union of n balanced rectangles. To see this observe that for all fixed $0 \leq k \leq n-1$ the language

$$L_n^k := (a+b)^k a (a+b)^{n-1} a (a+b)^{n-1-k}$$

is a balanced rectangle, setting $n_1 = k$, $n_2 = n+1$, $n_3 = n-1-k$, $L_1 = (a+b)^{n-1}$ and $L_2 = a(a+b)^{n-1}a$. Moreover L_n is by definition the union of all these languages. However, note that the L_n^k languages are not disjoint. In fact, in Section 4 we will see that any union of balanced rectangles which equals L must have size $2^{\Omega(n)}$, which by Proposition 7 proves our main lower bound. ◁

In the remainder of this section, we will prove Proposition 7. So fix a language L in which all words have the same length n and a context free grammar G accepting L. We start with the following simple observation.

Observation 9. *For every non-terminal A of G, the words that can be generated by a derivation in G starting from A all have the same length.*

Proof. By way of contradiction, assume that this were not the case, so there is some A from which we can generate words of different lengths. Then, by assumption, we know that A appears in a parse tree of G. But in that parse tree we can substitute the derivation below A by derivations for words of different lengths, and we get that L contains words of different lengths, which contradicts the assumption on L. □

With the help of Observation 9, we can rewrite G into an equivalent grammar in the following way: for every non-terminal A of G introduce new non-terminals $A_1, \dots A_n$. For every rule $A \to BC$ let ℓ be the length of the words generated by B and $i \in [n]$. Then if $i + \ell \leq n$ the grammar G' has rules $A_i \to B_i C_{i+\ell}$. Moreover, for every rule $A \to a$, the grammar G' has the rules $A_i \to a$ for $i \in [n]$. Finally, the start symbol of G' is S_1.

Lemma 10. *The grammars G and G' generate the same language and $|G'| \leq n|G|$. Moreover, if G is unambiguous then so is G'.*

Proof. We build on the observation that in G' the index i of the non-terminals A_i determines the position of the first letter of the word generated from A_i in every word generated by G. We use this to show that there is a bijection between the parse trees for the grammar G and G'. So fix a parse tree for a word w in G. Let A be a non-terminal symbol in this parse tree and say that A generates the word w'. Let i be the position of the first letter of w' in w. Then we substitute A by A_i in the parse tree. Doing this for every non-terminal yields a new tree which is easily seen to be a parse tree of G' for w. The inverse construction is simply deleting the indices that we introduced in the construction of G'. Overall, this shows that G and G' generate the same languages. Moreover, since we have a bijection of parse trees, if G is unambiguous then so is G'.

For the size bound, observe that for every $i \in [n]$ and every rule of G, we introduced a new rule of the same size. Thus, the size grows by a factor of n. □

We now make a connection between non-terminals of G' and rectangles.

Observation 11. *Let A_i be a non-terminal of G'. Let L' consist of all the words in the language of G' with a parse tree containing A_i. Then L' is a rectangle for some parameters $(L_1, L_2, n_1, n_2, n_3)$ where the words in L_2 are exactly those generated by A_i.*

Proof. Consider any word w in L'. Then $w = w_1w_2w_3$, where w_2 is generated from A_i. Then we can exchange w_2 by any other word $\bar{w}_2$ generated from A_i to get $\bar{w} = w_1\bar{w}_2w_3$ which is also in L', so $\{w_1\} \times L(A_i) \times \{w_3\} \subseteq L'$ where $L(A_i)$ consists of all words generated from A_i.

By Observation 9, we know that all words in $L(A_i)$ have the same length, that $|w_1| = i - 1$ and $|w_3| = n - (i-1) - |w_2|$. Therefore, for all $w' = w'_1w'_2w'_3 \in L'$ with $w'_2 \in L(A_i)$ we have $|w_1| = |w'_1|$, $|w_2| = |w'_2|$ and $|w_3| = |w'_3|$. So we can write

$$L' = \bigcup_{\substack{w_1w_2w_3 \in L': \\ w_2 \in L(A_i)}} \{w_1\} \times L(A_i) \times \{w_3\}$$

which proves the observation. □

We can now construct the set of rectangles in Proposition 7 iteratively via the following process. While the language of G' is not empty, choose a word w in it and a parse tree for w. We claim that this parse tree must contain a non-terminal A_i that generates words of length between $\frac{n}{3}$ and $\frac{2n}{3}$. To find this non-terminal, we use the following standard procedure: we descend in the parse tree from the root, iteratively always following the edge to the child whose subtree contains more leaves. We stop when the current node v has less than $\frac{2n}{3}$ leaves in its subtree and claim that we have found the desired non-terminal which is the label of v. To see this, note that, since we have not stopped before, the subtree of the parent of v contains more than $\frac{2n}{3}$ leaves, and since we go to the child with more children, v must have at least $\frac{n}{3}$ leaves in its subtree. Observing that the number of leaves below v is the same as the length of the generated word proves the claim.

We put L', defined as in Observation 11, into the set of rectangles we are constructing. Then we delete A_i from G' as well as all other non-terminals that no longer appear in any parse tree and continue the iteration with the resulting language.

This process terminates after at most $|G'| \leq n|G|$ iterations and so produces some set of rectangles $\{R_i\}_{i\in[\ell]}$ with $\ell \leq n|G|$. Observe that, by construction, $\bigcup_{i\in[\ell]} R_i = L$. Moreover, if G is unambiguous then, by Lemma 10, G' is also unambiguous. This means that every word in L appears in a unique parse tree in G' and so the union $\bigcup_{i\in[\ell]} R_i$ is disjoint. Therefore, this process really does produce a set of rectangles witnessing Proposition 7.

4 A Lower Bound for Disjoint Rectangle Covers

In this section, we show our lower bound for representations by uCFGs via an exponential lower bound for the language L_n from Example 3. This will provide the missing lower bound of Theorem 1 and thus prove our main result. Recall, that L_n is defined as follows:

$$L_n := \{(a+b)^k a(a+b)^{n-1}a(a+b)^{n-k-1} \mid k \leq n-1\}.$$

We will prove the following, confirming a conjecture from [20].

Theorem 12. *For every $n \in \mathbb{N}$, every uCFG accepting L has size $2^{\Omega(n)}$.*

In the rest of this section, we prove Theorem 12. Observe that every word in L_n has length $2n$ and that L_n consists of all those words such that there are two a symbols with distance exactly n between them. To prove Theorem 12, we will use the rectangle based method presented in Section 3.

Note that we may assume that n is sufficiently big by choosing the constant hidden in the Ω-notation to be sufficiently small. To simplify some of our arguments, for now we only consider L_n for values of n which are divisible by 4; it will be easy to remove this condition at the end of this section.

4.1 A Set Perspective

Before diving into our proof we first introduce a change of perspective. It will be helpful to think of each word over $\{a, b\}$ of length $2n$ as defining a pair of subsets of $[n]$. Formally, we define a bijection which maps each $w = w_1 \dots w_{2n}$, where $w_i \in \{a, b\}$ for each i to a pair $(X_w, Y_w) \subseteq \{x_1, \dots, x_n\} \times \{y_1, \dots, y_n\}$ where $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$ are both sets of elements and X_w contains every x_i such that $w_i = a$ and Y_w contains every y_i such that $w_{i+n} = a$. We write $X := \{x_1, \dots, x_n\}$ and $Y := \{y_1, \dots, y_n\}$. Sometimes it will be useful to refer to the elements of $X \cup Y$ in a unified way, hence we define

$$z_i := \begin{cases} x_i & i \in [n] \\ y_{i-n} & i \in [n+1, 2n]. \end{cases}$$

We write $Z[i, j] := \{z_\ell \mid i \le \ell \le j\}$ and $Z := Z[1, 2n]$. We similarly define $X[i, j]$ and $Y[i, j]$. Observe that each interval also induces a partition of Z as follows.

Definition 13 (Ordered Partition)**.** We say that a partition (Π_0, Π_1) of Z is *induced by the interval* $[i, j]$ if for some $\ell \in \{0, 1\}$, we have $\Pi_\ell = Z[i, j]$. We call a partition which is induced by some interval an *ordered* partition. Moreover, we call an ordered partition (Π_0, Π_1) *balanced* if

$$\frac{2n}{3} \le |\Pi_0|, |\Pi_1| \le \frac{4n}{3}$$

Having these dual notions of intervals and the partitions they induce on Z will be useful in our proofs. We next translate Definition 5 to fit the set perspective.

Definition 14 (Set Rectangle)**.** Let (Π_0, Π_1) be an ordered partition of Z. We say that $R \subseteq \mathcal{P}(Z)$ is an ordered (Π_0, Π_1)*-set rectangle* if there are sets $S \subseteq \mathcal{P}(\Pi_0)$ and $T \subseteq \mathcal{P}(\Pi_1)$ such that $R = S \times T$.[2] We say that R is a *balanced set-rectangle* if (Π_0, Π_1) is balanced. Moreover, if (Π_0, Π_1) is induced by $[i, j]$ we also call R an $[i, j]$-set rectangle.

The following lemma justifies Definition 14 and therefore our use of the set perspective in the rest of the proof.

Lemma 15. *Let L be a language over the alphabet $\{a, b\}$ with words of length $2n$, such that L is a rectangle in the sense of Definition 5 with parameters $(L_1, L_2, n_1, n_2, n_3)$. Then $\{(X_w, Y_w) \mid w \in L\}$ is a $[n_1 + 1, n_1 + n_2]$-set rectangle.*

Conversely, for any language L over the alphabet $\{a, b\}$ with words of length $2n$ such that $\{(X_w, Y_w) \mid w \in L\}$ is an $[i, j]$-set rectangle, L is a rectangle with $n_1 = i - 1$, $n_2 = j - i + 1$ and $n_3 = 2n - j$.

Proof. Let

$$L = \bigcup_{\substack{w_1 w_3 \in L_1: \\ |w_1| = n_1, |w_3| = n_3}} \{w_1\} \times L_2 \times \{w_3\} \text{ where } L_1 \subseteq \Sigma^{n_1 + n_3}, L_2 \subseteq \Sigma^{n_2}.$$

be a rectangle with parameters $(L_1, L_2, n_1, n_2, n_3)$. Define

$$S := \{(X_w, Y_w) \mid w = w_1 b^{n_2} w_3 \text{ for some } w_1 w_3 \in L_1\} \subseteq Z[1, n_1] \cup Z[n_1 + n_2 + 1, 2n] \text{ and}$$
$$T := \{(X_w, Y_w) \mid w = b^{n_1} w_2 b^{n_3} \text{ for some } w_2 \in L_2\} \subseteq Z[n_1 + 1, n_1 + n_2].$$

By construction $U \cap V = \varnothing$ for all $U \in S$, $V \in T$, so it is easy to verify that $S \times T = \{U \cup V \mid U \in S, V \in T\} = \{(X_w, Y_w) \mid w \in L\}$ and that this is a $[n_1 + 1, n_1 + n_2]$ rectangle. The converse direction is similar. □

[2] Remember that to simplify notation we read $S \times T$ as $\{U \cup V \mid U \in S, V \in T\}$.

Slightly abusing notation, we will from now on write L_n to mean $\{(X_w, Y_w) \mid w \in L_n\}$. Also from now on we refer to ordered set-rectangles simply as ordered rectangles; that is, we will now completely switch to the set perspective. Seen this way, the language L_n consists exactly of those pairs (X_w, Y_w) of sets for which there is an i such that $x_i \in X_w$ and $y_i \in Y_w$ are in the respective sets. Going one step further, if we identify the sets X_w, Y_w with the corresponding index sets, L_n consists of intersecting pairs of sets, so L_n is essentially the complement of the famous set disjointness problem, which is widely regarded as the flagship problem of communication complexity, see e.g. the survey [39].

From the above discussion along with Proposition 7, it follows that to prove Theorem 12 it suffices to prove the following proposition.

Proposition 16. *If $R_1, \dots, R_\ell$ is a set of ordered balanced rectangles such that R_i and R_j are disjoint for every $i \neq j$ and such that*

$$\bigcup_{i \in [\ell]} R_i = L_n,$$

then $\ell = 2^{\Omega(n)}$.

We call a set of rectangles as in Proposition 16 a *disjoint rectangle cover* for L_n. We will prove Proposition 16 in the next two subsections.

4.2 A Lower Bound for Restricted Rectangles

Before showing Proposition 16 in full generality, it will be useful to first consider a special case in which all rectangles share a particular partition.

Theorem 17. *Any disjoint rectangle cover of L_n by $[1, n]$-rectangles has size $2^{\Omega(n)}$.*

Theorem 17 is an immediate consequence of the so-called rank bound from communication complexity pioneered in [23], see e.g. [31, Chapter 2] for a textbook introduction. However, to prove Proposition 16 we need to generalise Theorem 17, which unfortunately cannot directly be done using known results from communication complexity: in contrast to most of the literature, Proposition 16 is not in the standard setting in which the partitions for all rectangles are fixed. Rather, different rectangles may use different partitions. This setting is called *multi-partition communication complexity* [14] and is far less studied. In particular, there is only one known lower bound that explicitly works for disjoint rectangle covers [33], which we cannot directly apply here.

As a first step to proving Proposition 16 we thus give a new proof of Theorem 17. This proof will be more involved then the usual approach, but it has the advantage that we can later generalise it to Proposition 16. We use a discrepancy argument in the spirit of that given by Sherstov in the context of multiparty communication complexity[3] [40]. The idea is to define disjoint $A, B \subset L_n$, such that (1) any $[1, n]$-rectangle contains roughly the same number of sets from A as from B and (2) L_n contains many more words from A than B. Therefore, the size of any disjoint cover for L_n by $[1, n]$-rectangles must be large.

Informally, we split $X \cup Y$ into intervals of size four and then define $\mathcal{L}$ to be those sets (U, V) containing exactly one element from each interval. Then A is defined to be the set of $(U, V) \in \mathcal{L}$ such that the number of i with $x_i \in U$ and $y_i \in V$ is odd and $B := \mathcal{L} \setminus A$. We next give the formal definition.

[3]We warn the reader that despite superficially similar names *multiparty* and *multipartition* communication complexity are very different concepts!

So set $m := n/4$; note that m is an integer since by assumption n is divisible by four. Then for $i \in [m]$ we define:

$$\begin{aligned} I_i^X &:= X[4(i-1)+1, 4i] \\ \mathcal{L}_i^X &:= \{U \subset I_i^X \mid |U| = 1\} \\ \mathcal{L}^X &:= \{U \subset X \mid |U \cap I_i| = 1 \text{ for every } i \in [m]\}. \end{aligned}$$

We analogously define I_i^Y, $\mathcal{L}_i^Y$ and $\mathcal{L}^Y$ exactly as above but with Y playing the role of X. Then we set $\mathcal{L} := \mathcal{L}^X \times \mathcal{L}^Y$ and $\mathcal{L}_i = \mathcal{L}_i^X \times \mathcal{L}_i^Y$. Since we often want to take a unified perspective we define for $i \in [m]$, $I_i := I_i^X$ and $I_{i+m} = I_i^Y$. We call every I_j an *interval*.

We next show that $\mathcal{L}$ is big and that L_n contains many more words from A than from B. Recall, that $A \subset \mathcal{L}$ contains those $(U, V) \in \mathcal{L}$ such that the number of i with $x_i \in U$ and $y_i \in V$ is odd and $B := \mathcal{L} \setminus A$.

Lemma 18. *The following holds:*

(1) $|\mathcal{L}| = 2^{4m}$.

(2) $|A \cap L_n| - |B \cap L_n| = |A| - |B \cap L_n| > 2^{\frac{7m}{2}}$.

Proof. For (1) observe that in total there are $2m$ intervals each of length four and that for every such interval exactly one element is chosen for every $(U, V) \in \mathcal{L}$, so $|\mathcal{L}| = 4^{2m} = 2^{4m}$.

The equality in (2) is trivial since $A \subset L_n$. To prove the inequality it suffices to show that $|B \setminus L_n| = 12^m$ and that $|B| - |A| = 2^{3m}$. To see this assume the aforementioned equalities, then

$$\begin{aligned} |A| - |B \cap L_n| &= |A| - (|B| - |B \setminus L_n|) \\ &= |B \setminus L_n| - (|B| - |A|) \\ &= 12^m - 2^{3m} > 2^{\frac{7m}{2}}, \end{aligned}$$

where in the last step we use that n (and thus m) is sufficiently big.

To show that $|B \setminus L_n| = 12^m$, consider for each $i \in [m]$ the 16 sets $(U_i, V_i) \in \mathcal{L}_i$. By the definition of $\mathcal{L}_i$, the sets U_i and V_i contain a single element, respectively, so $U_i = \{x_j\}$, $V_i = \{y_k\}$, for some appropriate j, k. There are exactly four combinations such that $j = k$ and 12 with $j \neq k$. Combining the latter for all $i \in [m]$ gives exactly $B \setminus L_n$, so $|B \setminus L_n| = 12^m$.

Finally, $|B| - |A| = 2^{3m}$ is true since

$$\begin{aligned} 2^{3m} = (12-4)^m &= \sum_{i=0}^{m} \binom{m}{i} 12^{m-i} (-4)^i \\ &= \sum_{i=0}^{\lfloor m/2 \rfloor} \left(\binom{m}{2i} 12^{m-2i} 4^{2i} \right) - \sum_{i=1}^{\lceil m/2 \rceil} \left(\binom{m}{2i-1} 12^{m-(2i-1)} 4^{2i-1} \right) \\ &= |B| - |A|. \end{aligned}$$

□

The key to proving Theorem 17 is the following: in Lemma 18, we have seen that there are many more sets in $A \cap L_n$ than in $B \cap L_n$. However, we will next show that each $[1, n]$-rectangle cannot contain many more sets of A than B. Therefore, any disjoint cover of L_n by $[1, n]$-rectangles must by big.

Lemma 19. *Let $R = S \times T$ be a $[1, n]$-rectangle. Then*

$$\big||R \cap A| - |R \cap B|\big| \leq 2^{3m}.$$

Proof. The key idea is to express $\big||R \cap A| - |R \cap B|\big|$ in terms of an appropriate expectation calculation. To set this up we need some definitions.

Define for each $i \in [m]$ random variables X_i, Y_i that take a value uniformly at random from $\mathcal{L}_i^X$ and $\mathcal{L}_i^Y$ respectively. We let all these random variables be independent and define new random variables $X := \bigcup_{i=1}^m X_i$, $Y = \bigcup_{i=1}^m Y_i$. Note that X has the uniform distribution over $\mathcal{L}^X$ and Y the uniform distribution on $\mathcal{L}^Y$. Moreover, X and Y are independent. Let Y' be an independent copy of Y.

For $L \subseteq \mathcal{P}(Z)$ we write χ_L for the indicator function of L, i.e. the function $\chi_L : \mathcal{P}(Z) \to \{0,1\}$ that evaluates to one exactly on inputs contained in L. For $U \in \mathcal{L}^X$ and $V \in \mathcal{L}^Y$ let $\mathsf{Int}(U,V) := 1$ if there is some i such that $x_i \in U$ and $y_i \in V$, otherwise $\mathsf{Int}(U,V) := 0$. Note that if $U \in \mathcal{L}^X$ and $V \in \mathcal{L}^Y$ then $(-1)^{\chi_A(U,V)} = (-1)^{\sum_{i=1}^m \mathsf{Int}(U_i,V_i)}$, where $U_i := U \cap I_i^X$ and $V_i := V \cap I_i^Y$ for $i \in [m]$. The following calculation borrows heavily from [31, Lemma 5.9] which in turn is inspired by an approach pioneered in [40].

$$\begin{aligned}
\left(\frac{|R \cap A| - |R \cap B|}{2^{4m}}\right)^2 &= \left(\mathop{\mathbb{E}}_{X,Y}\left[\chi_R(X,Y) \cdot (-1)^{\chi_A(X,Y)}\right]\right)^2 \\
&= \left(\mathop{\mathbb{E}}_{X,Y}\left[\chi_S(X) \cdot \chi_T(Y) \cdot (-1)^{\chi_A(X,Y)}\right]\right)^2 \\
&\leq \mathop{\mathbb{E}}_{X}\left[\chi_S(X)^2 \left(\mathop{\mathbb{E}}_{Y}\left[\chi_T(Y) \cdot (-1)^{\chi_A(X,Y)}\right]\right)^2\right] \\
&\leq \mathop{\mathbb{E}}_{X,Y,Y'}\left[\chi_T(Y) \cdot \chi_T(Y') \cdot (-1)^{\chi_A(X,Y)+\chi_A(X,Y')}\right] \\
&\leq \mathop{\mathbb{E}}_{Y,Y'}\left[\left|\mathop{\mathbb{E}}_{X}\left[(-1)^{\chi_A(X,Y)+\chi_A(X,Y')}\right]\right|\right] \\
&= \mathop{\mathbb{E}}_{Y,Y'}\left[\left|\mathop{\mathbb{E}}_{X}\left[(-1)^{\sum_{i=1}^m \mathsf{Int}(X_i,Y_i)+\mathsf{Int}(X_i,Y_i')}\right]\right|\right]
\end{aligned}$$

Now fix Y, Y'. We claim that if $Y \neq Y'$ the inner expectation must evaluate to zero. First note that the probability that $Y = Y'$ is $1/|\mathcal{L}^Y| = 2^{-2m}$. Therefore, the claim implies the result via a simple calculation. We now prove the claim.

So fix $Y \neq Y'$ while X remains random as before. Let C be the event that $\sum_{i=1}^m \mathsf{Int}(X_i,Y_i) + \mathsf{Int}(X_i,Y_i')$ is even and C_i be the event that $\mathsf{Int}(X_i,Y_i) + \mathsf{Int}(X_i,Y_i') = 1$. Then C is true if and only if the number of $i \in [m]$ such that C_i is true is even. Note that if $Y_i = Y_i'$ then C_i is false. So suppose $Y_i \neq Y_i'$. Since all intervals are of size four and $|X_i| = |Y_i| = |Y_i'| = 1$, we have $\mathbb{P}(C_i) = \frac{1}{2}$. Let $\mathcal{I} := \{i \in [m] \mid Y_i \neq Y_i'\}$ and $\alpha := |\mathcal{I}|$. Then

$$\mathbb{P}(C) = \sum_{i=0}^{\lfloor \frac{\alpha}{2} \rfloor} \binom{\alpha}{2i} 2^{-2i} 2^{-(\alpha-2i)} = 2^{-\alpha} \sum_{i=0}^{\lfloor \frac{\alpha}{2} \rfloor} \binom{\alpha}{2i} = 2^{-\alpha} 2^{\alpha-1} = \frac{1}{2},$$

where we use the fact that the sum of $\binom{\alpha}{i}$ over all even values of i is $2^{\alpha-1}$. The claim and therefore the result follows. □

From Lemma 19, we can deduce Theorem 17. In detail, let $R_1, \dots, R_\ell$ be a disjoint set of $[1, n]$-rectangles whose union is L_n. Then

$$\begin{aligned} 2^{\frac{7m}{2}} &< |A \cap L_n| - |B \cap L_n| \\ &= \sum_{i=1}^{\ell} |A \cap R_i| - |B \cap R_i| \\ &\leq \ell \cdot 2^{3m}, \end{aligned}$$

where the first inequality is Lemma 18(2), the equality uses the fact the rectangles form a cover and are disjoint, and the second inequality follows by Lemma 19. It follows that $\ell = 2^{\Omega(m)} = 2^{\Omega(n)}$ which completes the proof of Theorem 17.

4.3 Proof of Theorem 12

To prove Proposition 16, and therefore Theorem 12, it suffices to prove an analogue of Lemma 19 that applies to all balanced ordered rectangles. In fact we have already done much of the work. The main point is that for every balanced ordered partition (Π_0, Π_1) and 'most' values of ℓ, we have that x_ℓ and y_ℓ are on different sides of the partition. We now formalise the above idea, beginning with the nice case where the partition splits x_ℓ and y_ℓ for every ℓ. Here using symmetry we may deduce the following corollary to the proof of Lemma 19.

Corollary 20. *Let $R = S \times T$ be an $[i, j]$-rectangle such that $j - i = n - 1$. Then*

$$\big||R \cap A| - |R \cap B|\big| \leq 2^{3m}$$

To move from this to our more general case, we first elucidate some properties of balanced ordered partitions. So fix such a (Π_0, Π_1). Let G be the set containing those $i \in [n]$ such that x_i and y_i are in different parts of the partition, i.e. G contains those $i \in [n]$ such that exactly one of x_i and y_i lies in Π_0. Let V_G contain all those x_i and y_i such that $i \in G$. Moreover, let I_G contain all intervals I_i such that $I_i \subseteq V_G$.

It will be convenient to assume that (Π_0, Π_1) is well-behaved with respect to the intervals I_ℓ in the following sense: we call (Π_0, Π_1) *neat* if for every interval I_ℓ we have that $I_\ell \subseteq \Pi_0$ or $I_\ell \subseteq \Pi_1$. In the remainder, we can largely restrict ourselves to neat partitions due to the following result.

Lemma 21. *Let R be an ordered balanced (Π_0, Π_1)-rectangle. Then there there is a neat ordered balanced partition (Γ_0, Γ_1) and a set of disjoint (Γ_0, Γ_1)-rectangles $R_1, R_2, \dots, R_k$ with $k \leq 256$ such that $R = \bigcup_{\ell \in [k]} R_\ell$.*

Proof. Suppose w.l.o.g. that $|\Pi_0| \leq |\Pi_1|$. There are at most two intervals I_i, I_j which violate neatness, i.e. that contain elements from Π_0 and Π_1. Let (Γ_0, Γ_1) be the partition of Z that we get by putting all elements from I_i and I_j into Π_0. Then, by construction, (Γ_0, Γ_1) is neat and ordered. Moreover, it is balanced, since Γ_0 gained at most the 8 elements from $I_i \cup I_j$, so $|\Gamma_0| \leq |\Pi_0| + 8 \leq n + 8 \leq \frac{2n}{3}$, where for the last inequality we use that n is sufficiently big.

Now consider $\alpha \subseteq I_i \cup I_j$. Then let R_α contain exactly those sets U from R for which $U \cap (I_i \cup I_j) = \alpha$. Then clearly $R_\alpha \cap R_{\alpha'} = \varnothing$ for $\alpha \neq \alpha'$. Note that R_α is both a (Π_0, Π_1)-rectangle and a (Γ_0, Γ_1)-rectangle since it is fully determined on $I_i \cup I_j$ and thus on the difference between the two partitions. Moreover, we have

$$R = \bigcup_{\alpha \subseteq I_i \cup I_j} R_\alpha.$$

Since $|I_i \cup I_j| = 8$ and thus the disjoint union ranges over $2^8 = 256$ sets, the claim follows. □

Neat ordered partitions have the following useful properties.

Lemma 22. *Let (Π_0, Π_1) be a neat, ordered and balanced partition such that $|\Pi_0| \leq |\Pi_1|$. Then*
(1) *$\Pi_0 \subseteq V_G$ and*
(2) *$|\Pi_0| = |G|$.*

Proof. Let $[i, j]$ be the interval which induces the partition. First, suppose $\Pi_0 = Z[i, j]$. Since $|\Pi_0| \leq n$ it follows that for every two elements z_k, z_ℓ of Π_0 the indices have distance at most $n-1$, i.e., if $z_k, z_\ell \in \Pi_0$ then $|\ell - k| \leq n-1$. Since the distance between x_ℓ and y_ℓ is n for every ℓ, it follows that Π_0 cannot contain both x_ℓ and y_ℓ and thus $\Pi_0 \subseteq V_G$.

If instead $\Pi_0 = Z \setminus Z[i, j]$, then

$$\Pi_0 = \{z_\ell \in Z \mid \ell < i\} \cup \{z_\ell \in Z \mid \ell > j\}.$$

So since $|Z[i, j]| \geq n$, we know that $j - i \geq n-1$ and it is easy to see that no two elements of Π_0 are at distance n. So again $\Pi_0 \subseteq V_G$; we have shown (1).

For (2), observe that for every $\ell \in G$ we must have either x_ℓ or y_ℓ in Π_0, so $|G| \leq |\Pi_0|$. Moreover, from (1) we get that for every $\ell \in G$ the set Π_0 cannot contain x_ℓ and y_ℓ, so $|\Pi_0| \leq |G|$ and thus $|\Pi_0| = |G|$. □

We can now deploy Corollary 20 and Lemma 22 to prove a generalisation of Lemma 19.

Lemma 23. *Let (Π_0, Π_1) be a neat, ordered and balanced partition and let $R = S \times T$ be a (Π_0, Π_1)-rectangle. Then*

$$\big||R \cap A| - |R \cap B|\big| \leq 2^{10m/3}.$$

Proof. Suppose w.l.o.g. that $|\Pi_0| \leq |\Pi_1|$. Note that, since the partition is balanced, we have $|\Pi_0| \geq 2n/3$. Define $\Pi_1^g := \Pi_1 \cap V_G$ and $\Pi_1^b := \Pi_1 \setminus \Pi_1^g$. Observe that, via Lemma 22, we get $|\Pi_0| = |\Pi_1^g|$ and thus $|\Pi_1^b| = 2n - |\Pi_0| - |\Pi_1^g| = 2n - 2|\Pi_0| \leq 2n/3$.

Now let $\alpha \subseteq \Pi_1^b$ be a subset of the 'bad elements', i.e. the elements x_ℓ, y_ℓ that both lie in Π_1. Let $T_\alpha := \{U \in T \mid U \cap \Pi_1^b = \alpha\}$ and $T^\alpha = \{U \setminus \alpha \mid U \in T_\alpha\}$. Define $R^\alpha = S \times T^\alpha$. By construction, every element of $S \times T^\alpha$ is a subset of V_G. By renaming elements we can view $S \times T^\alpha$ as a subset of $L_{|G|}$, since all elements in Π_0 are in V_G by Lemma 22(1). Then for every $\alpha \in U$, since $|G| = |\Pi_0|$—by Lemma 22(2)—and by applying Corollary 20 to R^α with $m' := |G|/4$, we obtain that

$$\big||R^\alpha \cap A| - |R^\alpha \cap B|\big| \leq 2^{3m'} = 2^{\frac{3|G|}{4}}.$$

Here we use that our partition is neat and thus that G is a union of m' many intervals.

Moreover, because (Π_0, Π_1) is neat, Π_1^b is a union of intervals. Suppose there is some ℓ such that $I_\ell \subseteq \Pi_1^b$ and $|\alpha \cap I_\ell| \neq 1$. Then $\alpha \notin \mathcal{L}$ and thus $T_\alpha \cap \mathcal{L} = \varnothing$. Let $U \subseteq \mathcal{L}$ be the set of $\alpha \subseteq \Pi_1^b$ such that $T_\alpha \cap \mathcal{L} \neq \varnothing$. Then for all $\alpha \in U$ we have that $|\alpha \cap I_\ell| = 1$, for every ℓ such that $I_\ell \subseteq \Pi_1^b$. We obtain that

$$|U| \leq 4^{\frac{|\Pi_1^b|}{4}} = 4^{\frac{2n-2|G|}{4}} = 2^{n-|G|}.$$

It follows that

$$\begin{aligned}\big||R \cap A| - |R \cap B|\big| &\leq \sum_{\alpha \in U} \big||R^\alpha \cap A| - |R^\alpha \cap B|\big| \\ &\leq 2^{n-|G|} \cdot 2^{\frac{3|G|}{4}} = 2^{n-\frac{|G|}{4}} \leq 2^{\frac{5n}{6}} = 2^{\frac{10m}{3}}\end{aligned}$$

as required, where we use that $|G| = |\Pi_0| \geq 2n/3$. □

Proposition 16 now follows by a routine calculation similar to what we have already seen.

Proof of Proposition 16. We assume first, as before, that n is divisible by four, so we can apply all results from Section 4. Let $R_1, \ldots, R_\ell$ be an ordered disjoint cover of L_n. With Lemma 21, we can assume that the partitions for all rectangles are neat, since this only changes the number of rectangles by a constant factor. Then

$$\begin{aligned} 2^{\frac{7m}{2}} &< |A \cap L_n| - |B \cap L_n| \\ &= \sum_{i=1}^{\ell} |A \cap R_i| - |B \cap R_i| \\ &\leq \ell \cdot 2^{\frac{10m}{3}}. \end{aligned}$$

Where the first equality is Lemma 18(2), the second uses the fact the $\{R_i\}_{i \in [\ell]}$ is a disjoint cover of L_n and the inequality follows by Lemma 23. Since $\frac{7}{2} > \frac{10}{3}$, we obtain that $\ell = 2^{\Omega(m)} = 2^{\Omega(n)}$.

Now assume that n is not divisible by four, so $n = 4t + a$ for some $a \in [3]$. We call the elements of the form x_{4t+b}, y_{4t+b}, where $b \in [a]$ the *spare elements*. For each $i \in [\ell]$, let R'_i contain exactly those sets U from R_i, that do not contain any spare elements. Then clearly $\bigcup_{i \in \ell} R'_i = L_{4t}$ and moreover, the union is disjoint. Suppose R_i is a (Π_0, Π_1)-rectangle. Then for $i \in \{0, 1\}$ define Π'_i to be the set obtained from Π_i by removing all the spare elements. It follows that R'_i is also a (Π'_0, Π'_1) rectangle. The only remaining problem is that R'_i might not be balanced. But, by essentially the same argument as that used in Lemma 21, there is an ordered balanced partition (Γ_0, Γ_1) and a set of disjoint (Γ_0, Γ_1)-rectangles $T_1, \ldots, T_k$ with $k \leq 2^6 = 64$ such that $R'_i = \bigcup_{j \in [k]} T_j$. We thus obtain an ordered disjoint cover of L_{4t} of size at most 64ℓ. It follows that $\ell = 2^{\Omega(4t)} = 2^{\Omega(n)}$. □

Combining Proposition 16 and Proposition 7 directly yields Theorem 12 and thus Theorem 1.

5 Conclusions

We have shown an optimal double-exponential separation in terms of succinctness of CFGs from their unambiguous variant. Notably, this separation holds for a natural language over a binary alphabet. To obtain our separation we introduced a lower bound technique based on rectangles (Proposition 7), inspired by methods from knowledge compilation [6]. However, our argument has a different flavour to these lower bounds and so opens up interesting prospects for future work.

In particular, our lower bound applies to an *unambiguous* representation format. Dealing with such models poses a challenge and perhaps our techniques can be applied more widely to gain a better understanding of unambiguity. For example, in the context of *document spanners* it has been shown that *rigid* grammars can be made unambiguous in exponential time [3, Theorem 5]; can we show that this is optimal? Closer to the topic of our paper, can we prove that complementation is hard for uCFGs over finite languages? We should note that studying the complement operation for unambiguous representation formats can be challenging. The case of unambiguous finite automata [16, 32] and structured d-DNNF circuits [42] have only recently been solved, and even then the lower bounds are only quasi-polynomial. Moreover, the case of d-DNNF circuits has been open for over two decades. We hope that this paper marks a step towards a better understanding of unambiguity.

Acknowledgments

The authors thank Florent Capelli for helpful discussions. The first author would also like to thank the Simons Institute Fall 2023 program 'Logic and Algorithms in Database Theory and AI'. Here, via discussions with Wim Martens, he learned about the conjecture from [20] that is solved in this paper.

The first author was partially supported by the Agence nationale de la recherche (ANR) project EQUUS ANR-19-CE48-0019. The second author was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) project number 414325841.

References

[1] Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. A circuit-based approach to efficient enumeration. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 111:1–111:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.ICALP.2017.111, doi:10.4230/LIPICS.ICALP.2017.111.

[2] Antoine Amarilli and Florent Capelli. Tractable circuits in database theory. *SIGMOD Rec.*, 53(2):6–20, 2024. doi:10.1145/3685980.3685982.

[3] Antoine Amarilli, Louis Jachiet, Martin Muñoz, and Cristian Riveros. Efficient enumeration for annotated grammars. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 291–300. ACM, 2022. doi:10.1145/3517804.3526232.

[4] Nurzhan Bakibayev, Tomás Kociský, Dan Olteanu, and Jakub Zavodny. Aggregation and ordering in factorised databases. *Proc. VLDB Endow.*, 6(14):1990–2001, 2013. URL: http://www.vldb.org/pvldb/vol6/p1990-zavodny.pdf, doi:10.14778/2556549.2556579.

[5] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. Answering conjunctive queries under updates. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 303–318. ACM, 2017. doi:10.1145/3034786.3034789.

[6] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Knowledge compilation meets communication complexity. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1008–1014. IJCAI/AAAI Press, 2016. URL: http://www.ijcai.org/Abstract/16/147.

[7] Walter Bucher, Hermann A. Maurer, Karel Culík II, and Detlef Wotschke. Concise description of finite languages. *Theor. Comput. Sci.*, 14:227–246, 1981. doi:10.1016/0304-3975(81)90044-X.

[8] Cezar Câmpeanu and Wing Hong Ho. The maximum state complexity for finite languages. *J. Autom. Lang. Comb.*, 9(2/3):189–202, 2004. URL: https://doi.org/10.25596/jalc-2004-189, doi:10.25596/JALC-2004-189.

[9] Katrin Casel, Henning Fernau, Serge Gaspers, Benjamin Gras, and Markus L. Schmid. On the complexity of the smallest grammar problem over fixed alphabets. *Theory Comput. Syst.*, 65(2):344–409, 2021. URL: https://doi.org/10.1007/s00224-020-10013-w, doi:10.1007/S00224-020-10013-W.

[10] Noam Chomsky. On certain formal properties of grammars. *Inf. Control.*, 2(2):137–167, 1959. doi:10.1016/S0019-9958(59)90362-6.

[11] Thomas Colcombet. Unambiguity in automata theory. In Jeffrey O. Shallit and Alexander Okhotin, editors, *Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings*, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015. doi:10.1007/978-3-319-19225-3_1.

[12] Thomas Colcombet, Karin Quaas, and Michal Skrzypczak. Unambiguity in automata theory (dagstuhl seminar 21452). *Dagstuhl Reports*, 11(10):57–71, 2021. URL: https://doi.org/10.4230/DagRep.11.10.57, doi:10.4230/DAGREP.11.10.57.

[13] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002. URL: https://doi.org/10.1613/jair.989, doi:10.1613/JAIR.989.

[14] Pavol Duris, Juraj Hromkovic, Stasys Jukna, Martin Sauerhoff, and Georg Schnitger. On multi-partition communication complexity. *Inf. Comput.*, 194(1):49–75, 2004. URL: https://doi.org/10.1016/j.ic.2004.05.002, doi:10.1016/J.IC.2004.05.002.

[15] Yuval Filmus. Lower bounds for context-free grammars. *Inf. Process. Lett.*, 111(18):895–898, 2011. URL: https://doi.org/10.1016/j.ipl.2011.06.006, doi:10.1016/J.IPL.2011.06.006.

[16] Mika Göös, Stefan Kiefer, and Weiqiang Yuan. Lower bounds for unambiguous automata via communication complexity. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 126:1–126:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.ICALP.2022.126, doi:10.4230/LIPICS.ICALP.2022.126.

[17] Markus Holzer and Simon Wolfsteiner. On the grammatical complexity of finite languages. In Stavros Konstantinidis and Giovanni Pighizzini, editors, *Descriptional Complexity of Formal Systems - 20th IFIP WG 1.02 International Conference, DCFS 2018, Halifax, NS, Canada, July 25-27, 2018, Proceedings*, volume 10952 of *Lecture Notes in Computer Science*, pages

151–162. Springer, 2018. doi:10.1007/978-3-319-94631-3_13.

[18] John C. Kieffer and En-Hui Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory*, 46(3):737–754, 2000. doi:10.1109/18.841160.

[19] John C. Kieffer, En-Hui Yang, Gregory J. Nelson, and Pamela C. Cosman. Universal lossless compression via multilevel pattern matching. *IEEE Trans. Inf. Theory*, 46(4):1227–1245, 2000. doi:10.1109/18.850665.

[20] Benny Kimelfeld, Wim Martens, and Matthias Niewerth. A formal language perspective on factorized representations. *arXiv*, 2025. Accepted for publication in the *Proceedings of the 28th International Conference on Database Theory (ICDT 2025)*. URL: https://doi.org/10.48550/arXiv.2309.11663, arXiv:2309.11663, doi:10.48550/ARXIV.2309.11663.

[21] Markus Lohrey. Algorithmics on slp-compressed strings: A survey. *Groups Complex. Cryptol.*, 4(2):241–299, 2012. URL: https://doi.org/10.1515/gcc-2012-0016, doi:10.1515/GCC-2012-0016.

[22] Markus Lohrey and Markus L. Schmid. Enumeration for mso-queries on compressed trees. *Proc. ACM Manag. Data*, 2(2):78, 2024. doi:10.1145/3651141.

[23] Kurt Mehlhorn and Erik Meineche Schmidt. Las vegas is better than determinism in VLSI and distributed computing (extended abstract). In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 330–337. ACM, 1982. doi:10.1145/800070.802208.

[24] Martin Muñoz and Cristian Riveros. Streaming enumeration on nested documents. In Dan Olteanu and Nils Vortmeier, editors, *25th International Conference on Database Theory, ICDT 2022, March 29 to April 1, 2022, Edinburgh, UK (Virtual Conference)*, volume 220 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.ICDT.2022.19, doi:10.4230/LIPICS.ICDT.2022.19.

[25] Martin Muñoz and Cristian Riveros. Constant-delay enumeration for slp-compressed documents. In Floris Geerts and Brecht Vandevoort, editors, *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, volume 255 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.ICDT.2023.7, doi:10.4230/LIPICS.ICDT.2023.7.

[26] Dan Olteanu. Factorized databases: A knowledge compilation perspective. In Adnan Darwiche, editor, *Beyond NP, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016*, volume WS-16-05 of *AAAI Technical Report*. AAAI Press, 2016. URL: http://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12638.

[27] Dan Olteanu. Recent increments in incremental view maintenance. In Floris Geerts and Wim Martens, editors, *Companion of the 43rd Symposium on Principles of Database Systems, PODS 2024, Santiago, Chile, June 9-15, 2024*, pages 8–17. ACM, 2024. doi:10.1145/3635138.3654763.

[28] Dan Olteanu and Jakub Zavodny. Factorised representations of query results: size bounds and readability. In Alin Deutsch, editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 285–298. ACM, 2012. doi:10.1145/2274576.2274607.

[29] Dan Olteanu and Jakub Závodný. Size bounds for factorised representations of query results. *ACM Trans. Database Syst.*, 40(1):2:1–2:44, 2015. doi:10.1145/2656335.

[30] Liat Peterfreund. Grammars for document spanners. In Ke Yi and Zhewei Wei, editors, *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, volume 186 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: https://doi.org/10.4230/LIPIcs.ICDT.2021.7, doi:10.4230/LIPICS.ICDT.2021.7.

[31] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.

[32] Mikhail A. Raskin. A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 138:1–138:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. URL: https://doi.org/10.4230/LIPIcs.ICALP.2018.138, doi:10.4230/LIPICS.ICALP.2018.138.

[33] Martin Sauerhoff. Approximation of boolean functions by combinatorial rectangles. *Theor. Comput. Sci.*, 301(1-3):45–78, 2003. doi:10.1016/S0304-3975(02)00568-6.

[34] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. Learning linear regression models over factorized joins. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 3–18. ACM, 2016. doi:10.1145/2882903.2882939.

[35] Markus L. Schmid and Nicole Schweikardt. Spanner evaluation over slp-compressed documents. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 153–165. ACM, 2021. doi:10.1145/3452021.3458325.

[36] Markus L. Schmid and Nicole Schweikardt. Document spanners - A brief overview of concepts, results, and recent developments. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data,*

Philadelphia, PA, USA, June 12 - 17, 2022, pages 139–150. ACM, 2022. doi:10.1145/3517804.3526069.

[37] Markus L. Schmid and Nicole Schweikardt. Query evaluation over slp-represented document databases with complex document editing. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 79–89. ACM, 2022. doi:10.1145/3517804.3524158.

[38] Erik Meineche Schmidt and Thomas G. Szymanski. Succinctness of descriptions of unambiguous context-free languages. *SIAM J. Comput.*, 6(3):547–553, 1977. doi:10.1137/0206039.

[39] Alexander A. Sherstov. Communication complexity theory: Thirty-five years of set disjointness. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2014. doi:10.1007/978-3-662-44522-8_3.

[40] Alexander A. Sherstov. The multiparty communication complexity of set disjointness. *SIAM J. Comput.*, 45(4):1450–1489, 2016. doi:10.1137/120891587.

[41] Michael Sipser. *Introduction to the theory of computation.* PWS Publishing Company, 1997.

[42] Harry Vinall-Smeeth. Structured d-dnnf is not closed under negation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 3593–3601. ijcai.org, 2024. URL: https://www.ijcai.org/proceedings/2024/398.

[43] En-Hui Yang and John C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform - part one: Without context models. *IEEE Trans. Inf. Theory*, 46(3):755–777, 2000. doi:10.1109/18.841161.

A A small CFG for L_n

Fix an integer $n \in \mathbb{N}$. The idea will be to construct a word w of length $n-1$ and then add another word $aw'a$ where w' also has length $n-1$ at some position into w.

First, it will be useful to create words of specific lengths. To this end, for all i with $2^i < n$, we introduce a non-terminal B_i that generates all possible words of length 2^i by the rules

$$B_i \to B_{i-1}B_{i-1} \qquad \text{for } i \text{ with } 1 \leq 2^i < n$$
$$B_0 \to a \mid b$$

From the binary representation of $n-1$, we can compute a set $I = \{i_1, \dots, i_\ell\}$ of integers of size $O(\log(n))$ such that $n - 1 = \sum_{i \in I} 2^i$. Then we can imagine that w is decomposed into blocks of length 2^i for $i \in I$, that is $w \in \{a,b\}^{2^{i_1}} \times \dots \times \{a,b\}^{2^{i_\ell}}$. Creating the blocks of w is easy with the B_i, but we still have to choose in which block we want to add $aw'a$. To this end, we introduce a binary tree $T = (V, E)$ whose leaves are bijectively labelled with the elements from I. For every node v of T we introduce two non-terminals C_v, D_v where intuitively in a parse tree we will use C_v if $aw'a$ is added in a subword corresponding to the blocks of the leaf labels below v and D_v otherwise. Concretely, for v with children u, w, we add the rules

$$C_v \to C_u D_w \mid D_u C_w$$
$$D_v \to D_u D_w.$$

For the leaves v, there are two cases: first we add for all leaves v and all i_j such that the label of v is i_j the rule

$$D_v \to B_{i_j}$$

to generate all words of length 2^{i_j} from symbol D_v (since there we do not want to add $aw'a$).

For the C_v, we want to create all words of length 2^{i_j} and insert $aw'a$ at some position. We proceed similarly to Example 3. For all leaves v and all i_j such that the label of v is i_j we add the rule

$$C_v \to A_{I_j}.$$

Then, we introduce symbols A_i for all i with $2^i < n$ and rules

$$A_i \to B_{i-1}A_{i-1} \qquad \text{for } i \text{ with } 2^i < n$$
$$A_0 \to B_0 aSa \mid aSaB_0,$$

where S is a new symbol from which we create w' by the rule

$$S \to B_{i_1} \dots B_{i_\ell}.$$

Taking C_r, where r is the root of T, as the starting symbol completes the construction of the grammar. By the discussion above, it generates L_n. So it only remains to show that the size of the grammar is $O(\log(n))$.

First, observe that all rules but the last—which is of size $|I| = O(\log(n))$—have constant size. So it suffices to bound the number of rules. Noting further, that for every non-terminal T we have only a constant number of rules $T \to W$, we can in fact just bound the number of non-terminals. By construction, we have $O(\log(n))$ non-terminals A_i and B_i. Moerover, we have $O(|V(T)|) = O(|I|) = O(\log(n))$ non-terminals C_v and D_v. The only additional non-terminal is the single S, which completes the proof.

Received December 2024; revised February 2025; accepted March 2025