

# On the complexity of the universality and inclusion problems for unambiguous context-free grammars (Invited Paper)

Lorenzo Clemente

Department of Mathematics, Informatics, and Mechanics (MIMUW)

University of Warsaw (Warsaw, Poland)\*

clementelorenzo@gmail.com

We study the computational complexity of universality and inclusion problems for unambiguous finite automata and context-free grammars. We observe that several such problems can be reduced to the universality problem for unambiguous context-free grammars. The latter problem has long been known to be decidable and we propose a PSPACE algorithm that works by reduction to the zeroness problem of recurrence equations with convolution. We are not aware of any non-trivial complexity lower bounds. However, we show that computing the coin-flip measure of an unambiguous context-free language, a quantitative generalisation of universality, is hard for the long-standing open problem SQRTSUM.

## 1 Introduction

The purpose of this note is to attract attention to a long-standing open problem in formal language theory. The problem in question is the exact complexity of deciding universality of unambiguous context-free grammars (UUCFG). A context-free grammar is *unambiguous* if every accepted word admits a unique parse tree, and the universality problem asks, for a given grammar  $G$  over a finite set of terminals  $\Sigma$  (alphabet), whether  $G$  accepts every word  $L(G) = \Sigma^*$ . While the universality problem for context-free grammars is undecidable [19], the same problem for unambiguous grammars is long-known to be decidable (a corollary of [29, Theorem 5.5]), e.g., by reducing to the first-order theory of the reals with one quantifier alternation [29, eq. (3), page 149]. Since the latter fragment is decidable in EXPTIME [16], this yields an EXPTIME upper bound for UUCFG. No non-trivial lower bound for UUCFG seems to be known in the literature.

The typical way to solve a containment problem of the form  $L \subseteq M$  is to complement  $M$  and solve  $L \cap (\Sigma^* \setminus M) = \emptyset$ . For instance, when  $L$  is regular and  $M$  is deterministic context-free (DCFG), this gives a PTIME procedure since DCFG languages are efficiently closed under complement and intersection with regular languages, and their emptiness problem is in PTIME. However, UCFG languages are not closed under complement (the complement is not even context-free in general [18]), so the language-theoretic approach is not available. As Salomaa and Soittola remark in their book from 1978, “no proof is known for Theorem 5.5 which uses only standard formal language theory”. To this day, we are not aware of a proof of decidability for UUCFG using different techniques<sup>1</sup>. The UUCFG problem is not isolated in this respect.

---

\*This work has been partially supported by the Polish NCN grant 2017/26/D/ST6/00201. This is a technical report of an invited paper of the same title to appear in the proceedings of VPT 2020.

<sup>1</sup> In a later book, Kuich and Salomaa reprove decidability [23, Corollary 16.25] by using variable elimination, which is arguably closer to algebraic geometry than formal languages.

$\subseteq$	DFA	UFA	NFA	DCFG	UCFG	CFG
DFA	PTIME	PTIME	PSPACE-c. [26]	PTIME	=UUCFG (Th. 8)	undec.
UFA	PTIME	PTIME [33]	PSPACE-c. [26]	PTIME	=UUCFG (Th. 8)	undec.
NFA	PTIME	PTIME (Th. 7)	PSPACE-c. [26]	PTIME	=UUCFG (Th. 8)	undec.
DCFG	PTIME	$\leq$ UUCFG (Th. 9)	EXPTIME-c. [20]	undec.	undec.	undec.
UCFG	PTIME	$\leq$ UUCFG (Th. 9)	EXPTIME-c. [20]	undec.	undec.	undec.
CFG	PTIME	$\leq$ UUCFG (Th. 9)	EXPTIME-c. [20]	undec.	undec.	undec.

“ $\leq$ UUCFG”: the problem reduces in PTIME to UUCFG.

“=UUCFG”: the problem is PTIME inter-reducible with UUCFG.

Figure 1: Inclusion problems for various classes of regular and context-free languages.

**State of the art.** Let  $\mathcal{A}, \mathcal{B}$  be two classes of language acceptors. Examples include deterministic (DFA), unambiguous (UFA), and nondeterministic finite automata (NFA), and similarly for context-free grammars we have the classes DCFG, UCFG, and CFG. The “ $\mathcal{A} \subseteq \mathcal{B}$ ” *inclusion problem* asks, given a language acceptor  $A$  from  $\mathcal{A}$  and  $B$  from  $\mathcal{B}$ , whether the languages they recognise satisfy  $L(A) \subseteq L(B)$ . A summary of decidability and complexity result for inclusion problems involving finite automata and grammars is presented in Fig. 1. Many entries in the table are well-known. The problem  $\text{NFA} \subseteq \text{NFA}$  is a classic PSPACE-complete problem [26]. The problem  $\text{UFA} \subseteq \text{UFA}$  was shown in PTIME by Stearns and Hunt in their seminal paper [33]<sup>2</sup>. The fact that  $\text{CFG} \subseteq \text{NFA}$  is EXPTIME-complete is somewhat less known [20, Theorem 2.1]. The inclusion problems  $\mathcal{A} \subseteq \text{UFA}$  when  $\mathcal{B}$  is DCFG, UCFG, or CFG do not appear to have been studied before. The  $\mathcal{A} \subseteq \mathcal{B}$  problem is undecidable as soon as both  $\mathcal{A}, \mathcal{B}$  are context-free grammars, since  $\text{DCFG} \subseteq \text{DCFG}$  is well-known to be undecidable [19, Theorem 10.7, Point 2]. We have already observed that  $\text{NFA} \subseteq \text{DCFG}$  is in PTIME. The equivalence problem  $\text{NFA} = \text{UCFG}$  is shown to be decidable in [29, Theorem 5.5], although no complexity bound is given. The more general inclusion  $\text{NFA} \subseteq \text{UCFG}$  does not seem to have been studied before.

**Contributions.** We establish several connections between inclusion problems  $\mathcal{A} \subseteq \mathcal{B}$  when  $\mathcal{B}$  is UFA or UCFG with the UUCFG problem. Our contributions are as follows.

1. We observe that in many cases the inclusion problem  $L \subseteq M$  reduces in polynomial time to the sub-case where  $L$  is deterministic (Section 3.1.1). One application is lower bounds: Once we know that  $\text{CFG} \subseteq \text{NFA}$  is EXPTIME-hard [20, Theorem 2.1], we can immediately deduce that the same lower bound carries over to  $\text{DCFG} \subseteq \text{NFA}$  [20, Theorem 3.1].
2. We observe that in many cases the inclusion problem  $L \subseteq M$  with  $L$  deterministic reduces in polynomial time to the universality problem (Section 3.1.2). One application is upper bounds (combined with the previous point): For instance, from the fact that  $\text{UFA} = \Sigma^*$  is in PTIME we can deduce that the more general problem  $\text{NFA} \subseteq \text{UFA}$  is also in PTIME (Theorem 7), which seems to be a new observation.
3. We apply the last two points to show that the following inclusion problems  $\mathcal{A} \subseteq \mathcal{B}$  reduce to UUCFG:  $\mathcal{A} \in \{\text{DCFG}, \text{UCFG}, \text{CFG}\}$  and  $\mathcal{B} = \text{UFA}$  (Theorem 9);  $\mathcal{A} \in \{\text{DFA}, \text{UFA}, \text{NFA}\}$  and  $\mathcal{B} = \text{UCFG}$  (Theorem 8). Since UUCFG is a special instance of the latter set of problems, they are PTIME inter-reducible with UUCFG.

<sup>2</sup> An incomparable  $\text{NC}^2$  upper bound for this problem is also known [25, Fact 4.5] (c.f. [35, Theorem 2]).

4. We show that UUCFG is in PSPACE (Theorem 10), which improves the EXPTIME upper bound that can be extracted from [29]. A PSPACE upper bound for the same problem has also been shown by S. Purgal in his master thesis [28, Section 3.7].
5. We complement the upper bound in the previous point by showing that computing the so-called *coin-flip measure* of a UCFG (a quantitative problem generalising universality; c.f. Section 4) is SQRT-SUM-hard (Theorem 11). The latter is a well-known problem in the theory of numerical computation, which is not known to be in NP or NP-hard [1, 13].

The generic and simple polynomial time reductions of points 1. and 2. above do not seem to be known in the literature. Beyond the seminal work on UFA [33], they also apply to very recent contributions on expressive models such as unambiguous register automata (c.f. [27] for equality atoms) and unambiguous finite and pushdown Parikh automata [4]. In each of the cases above, one can reduce from inclusion to universality. A non-example where the reduction cannot be applied is unambiguous Petri-nets with coverability semantics [8].

The PSPACE upper bound on UUCFG is obtained by reduction to a more general counting problem interesting on its own. We introduce a natural class of number sequences  $f : \mathbb{N} \rightarrow \mathbb{N}$  which we call *convolution recursive* (conv-rec). Examples include the Fibonacci  $F(n+1) = F(n) + F(n-1)$  and Catalan numbers  $C(n+1) = (C * C)(n)$ , where “ $*$ ” denotes the convolution product. We show that the function counting the number of words in  $L(G)$  of a given length is conv-rec if  $G$  is UCFG. (This result is analogous to the well-known fact that UCFG have algebraic generating functions [7].) The *zeroness problem* asks whether such a sequence is identically zero. Our last contribution is a complexity upper-bound for the zeroness problem of conv-rec sequences.

6. We show that the zeroness problem of conv-rec sequences is in PSPACE (Theorem 4). We express this problem with a formula in the existential fragment for first-order logic over the reals, which can be decided in PSPACE [6].

## 2 Convolution recursive sequences and their zeroness problem

**Convolution recursive sequences.** Let  $\mathbb{N}$ ,  $\mathbb{Z}$ ,  $\mathbb{Q}$ , and  $\mathbb{R}$  be the sets of natural, resp., integer, rational, and real numbers. Let  $\mathbb{Q}[x_1, \dots, x_k]$  denote the ring of polynomials with coefficients from  $\mathbb{Q}$  and variables  $x_1, \dots, x_k$ . For two sequences indexed by natural numbers  $f, g : \mathbb{N} \rightarrow \mathbb{R}$ , their *sum*  $f + g$  is the sequence  $(f + g)(n) = f(n) + g(n)$ , and their *convolution* is the sequence  $(f * g)(n) = \sum_{k=0}^n f(k) \cdot g(n-k)$ . The convolution operation is associative  $f * (g * h) = (f * g) * h$ , commutative  $f * g = g * f$ , has as (left and right) identity the sequence  $1, 0, 0, \dots$ , and distributes over the sum operation  $(f + g) * h = f * g + g * h$ . Thus, sequences with the operations “ $+$ ” and “ $*$ ” form a semiring. Let  $\sigma : (\mathbb{N} \rightarrow \mathbb{R}) \rightarrow (\mathbb{N} \rightarrow \mathbb{R})$  be the (*forward*) *shift operator* on sequences, which is defined as  $(\sigma f)(n) = f(n+1)$ . The *zeroness problem* for a sequence  $f : \mathbb{N} \rightarrow \mathbb{R}$  amounts to decide whether  $f(n) = 0$  for every  $n \in \mathbb{N}$ .

A *convolution polynomial*  $p(x_1, \dots, x_k)$  is a polynomial where the multiplication operation is interpreted as convolution and a constant  $k \in \mathbb{Q}$  is interpreted as the sequence  $k, 0, 0, \dots$ . For example,  $4 * (x_1 * x_2) + 3 * (x_2 * x_2)$  is a convolution polynomial of two variables  $x_1, x_2$ . Let  $\mathbb{Q}_*[x_1, \dots, x_k]$  denote the ring of convolution polynomials with variables  $x_1, \dots, x_k$ . A sequence  $f : \mathbb{N} \rightarrow \mathbb{R}$  is *convolution recursive* (conv-rec) if there are  $k$  auxiliary sequences  $f_1, \dots, f_k : \mathbb{N} \rightarrow \mathbb{R}$  with  $f_1 = f$  and  $k$  convolution

polynomials  $p_1, \dots, p_k \in \mathbb{Q}_*[x_1, \dots, x_k]$  s.t.,

$$\begin{cases} \sigma f_1 &= p_1(f_1, \dots, f_k), \\ &\vdots \\ \sigma f_k &= p_k(f_1, \dots, f_k). \end{cases} \quad (1)$$

The *combined degree* of the representation above is the sum of the degrees of  $p_1, \dots, p_k$ . For example, the Catalan numbers  $C : \mathbb{N} \rightarrow \mathbb{N}$  are conv-rec (of combined degree two) since  $(\sigma C)(n) = (C * C)(n)$ .

**Lemma 1.** *Let  $f : \mathbb{N} \rightarrow \mathbb{R}$  be a conv-rec sequence of combined degree  $\leq d$ . Then  $\lim_{n \rightarrow \infty} \frac{f(n+1)}{f(n)} = O(d)$ .*

*Proof.* The maximal relative growth  $\frac{f(n+1)}{f(n)}$  of a conv-rec sequence is achieved when  $f$  satisfies a recurrence of the form  $\sigma f = f * \dots * f$  ( $d$  times) for some degree  $d \in \mathbb{N}$ . If  $f(0) = 1$ , then the resulting sequence is known as the Fuss-Catalan numbers [15] and it equals  $f(n) = \binom{d \cdot n + 1}{n} \frac{1}{d \cdot n + 1}$ . It can be checked by using Stirling's approximation  $n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$  that  $\lim_{n \rightarrow \infty} \frac{f(n+1)}{f(n)} = \frac{d^d}{(d-1)^{d-1}} = d \cdot \left(1 + \frac{1}{d-1}\right)^{d-1}$ . The latter quantity is upper bounded by  $d \cdot e$  for every  $d \geq 1$ .  $\square$

**Generatingfunctionology.** The *formal power series* (a.k.a. *ordinary generating function*) associated with a number sequence  $a : \mathbb{N} \rightarrow \mathbb{R}$  is the infinite polynomial  $g_a(x) = \sum_{n=0}^{\infty} a(n) \cdot x^n$ . Let  $[x_n]g_a$  denote the coefficient  $a(n)$  of  $x^n$  in  $g_a$ . Let  $f, f_1, f_2 : \mathbb{N} \rightarrow \mathbb{R}$  be sequences. It is well known that  $g_k(x) = k$  for  $k \in \mathbb{R}$ ,  $g_{f_1+f_2} = g_{f_1} + g_{f_2}$ ,  $g_{f_1 * f_2} = g_{f_1} \cdot g_{f_2}$ , and  $g_f(x) = f(0) + x \cdot g_{\sigma f}(x)$ . Consequently, if  $f_1$  is conv-recursive with auxiliary sequences  $f_1, \dots, f_k$ , then their generating functions  $g_{f_1}, \dots, g_{f_k}$  satisfy the following system of polynomial equations

$$\begin{cases} g_{f_1}(x) &= f_1(0) + x \cdot \hat{p}_1(g_{f_1}(x), \dots, g_{f_k}(x)), \\ &\vdots \\ g_{f_k}(x) &= f_k(0) + x \cdot \hat{p}_k(g_{f_1}(x), \dots, g_{f_k}(x)). \end{cases} \quad (2)$$

where  $\hat{p}_i$  is the polynomial obtained from the convolution polynomial  $p_i$  by replacing the convolution operation “ $*$ ” on sequences by the product operation “ $\cdot$ ” on real numbers. Thus, the generating function  $g_f$  of a conv-rec sequence  $f$  is *algebraic*.

**Lemma 2.** *The system of equations (2) has a unique formal power series solution.*

*Proof.* By construction,  $g_f = (g_{f_1}, \dots, g_{f_k})$  is a formal power series solution of (2). We now argue that there is no other solution. Assume that  $g = (g_1, \dots, g_k)$  is a solution of (2). We prove that, for every  $n \in \mathbb{N}$ ,  $[x^n]g = ([x^n]g_1, \dots, [x^n]g_k)$  equals  $[x^n]g_f = ([x^n]g_{f_1}, \dots, [x^n]g_{f_k})$ . The base case follows immediately from (2), since  $[x^0]g_i = f_i(0)$  by definition. For the inductive step  $n > 0$ , notice that 1) from (2) we have  $[x^n]g_i = [x^n](x \cdot \hat{p}_i(g)) = [x^{n-1}]\hat{p}_i(g)$ , and 2) the latter quantity is a (polynomial) function of the coefficients  $[x^i]g$  for  $0 \leq i \leq n-1$ . By inductive assumption,  $[x^i]g = [x^i]g_f$  for every  $0 \leq i \leq n-1$ , and thus by the two observations above  $[x^n]g = [x^n]g_f$ .  $\square$

**Lemma 3.** *Let  $d$  be the combined degree of  $f = (f_1, \dots, f_k)$ . The system (2) has a unique solution  $g_f(x^*) = (g_{f_1}(x^*), \dots, g_{f_k}(x^*)) \in \mathbb{R}^k$  for every  $0 \leq x^* < \frac{1}{d}$ .*

*Proof.* Let  $g_f = (g_{f_1}, \dots, g_{f_k})$  be the tuple of formal power series of the sequences  $f_1, \dots, f_k$ . By Lemma 1,  $\lim_{n \rightarrow \infty} \frac{f_i(n+1)}{f_i(n)} = O(d)$ . Thus,  $g_f(x^*) = (g_{f_1}(x^*), \dots, g_{f_k}(x^*)) \in \mathbb{R}^k$  converges for every  $0 \leq x^* < \frac{1}{d}$ . By Lemma 2,  $g_f$  is the unique formal power series solution of (2).  $\square$

**Theorem 4.** *The zeroness problem for conv-rec sequences is in PSPACE.*

*Proof.* Let  $f_1$  be a conv-rec sequence of combined degree  $d$  with auxiliary sequences  $f_2, \dots, f_k$  satisfying (1). Consider the associated generating functions  $g = (g_{f_1}, \dots, g_{f_k})$ . Clearly,  $f_1(n) = 0$  for every  $n \in \mathbb{N}$  if, and only if,  $g_{f_1}(x) = 0$  for every  $x$  sufficiently small. By Lemma 3,  $g(x^*)$  is the unique solution of (2) for every  $0 \leq x^* < \frac{1}{d}$ . It thus suffices to say that, for every  $0 \leq x^* < \frac{1}{d}$ , all solutions  $g(x^*)$  of the system (2) satisfy  $g_{f_1}(x^*) = 0$ . This can be expressed by the following universal first-order sentence over the reals (where  $\bar{y} = (y_1, \dots, y_k)$ )

$$\forall \left( 0 \leq x < \frac{1}{d} \right) . \forall \bar{y} . \bar{y} = f(0) + x \cdot \hat{p}(\bar{y}) \rightarrow y_1 = 0.$$

The sentence above can be decided in PSPACE by appealing to the existential theory of the reals [6, Theorem 3.3].  $\square$

### 3 Universality of unambiguous grammars

Let  $\Sigma$  be a finite alphabet. We denote by  $\Sigma^*$  the set of all finite words over  $\Sigma$ , including the empty word  $\varepsilon$ . A *language* is a subset  $L \subseteq \Sigma^*$ . The concatenation of two languages  $L, M \subseteq \Sigma^*$  is *unambiguous* if  $w \in L \cdot M$  implies that  $w$  factors uniquely as  $w = u \cdot v$  with  $u \in L$  and  $M \in v$ . A *context-free grammar* (CFG) is a tuple  $G = (\Sigma, N, S, \leftarrow)$  where  $\Sigma$  is a finite alphabet of *terminal symbols*,  $N$  is a finite set of *nonterminal symbols*, of which  $S \in N$  is the *starting nonterminal symbol*, and  $\leftarrow \subseteq N \times (N \cup \Sigma)^*$  is a set of productions. A CFG is in *short Greibach normal form* if productions are of the form either  $X \leftarrow \varepsilon$ . or  $X \leftarrow aYZ$ . An *X-derivation tree* is a tree satisfying the following conditions: 1) the root node  $\varepsilon$  is labelled by the nonterminal  $X \in X$ , 2) every internal node is labelled by a nonterminal from  $N$ , 3) whenever a node  $u$  has children  $u \cdot 1, \dots, u \cdot k$  there exists a rule  $Y \leftarrow w_1 \dots w_k$  with  $w_i \in N \cup \Sigma$  s.t.  $Y$  is the label of  $u$  and  $w_i$  is the label of  $u \cdot i$ , and 4) leaves are labelled with terminal symbols from  $\Sigma$ . The *language recognised* by a nonterminal  $X$  is the set  $L(X)$  of words  $w = a_1 \dots a_n \in \Sigma$  s.t. there exists an  $X$ -derivation tree with leaves labelled by (left-to-right)  $a_1, \dots, a_n$ ; the language recognised by  $G$  is the language recognised by the starting nonterminal  $L(G) = L(S)$ . A CFG  $G$  is *unambiguous* (UCFG) if for every accepted word  $w \in L(G)$  there exists exactly one derivation tree witnessing its acceptance. The universality problem (UUCFG) asks, given a UCFG  $G$ , whether  $L(G) = \Sigma^*$ .

#### 3.1 Reductions

In this section present PTIME reductions from inclusion problems for NFA and UCFG to UUCFG. This serves us as a motivation to study the complexity of UUCFG in Section 3.2. We proceed in two steps. In the first step, we present a general l.h.s. determinisation procedure for inclusion problems (Section 3.1.1) which is widely applicable to essentially any machine-based model of computation. In the second step, assuming a deterministic l.h.s., we show a reduction from inclusion to universality (Section 3.1.2). We apply these two reductions in Section 3.1.3.

##### 3.1.1 L.h.s. determinisation for inclusion problems

It is an empirical observation that in many inclusion problems of the form  $L \subseteq M$  the major source of difficulty is with  $M$  and not with  $L$ . For example, for finite automata the inclusion problem is PSPACE-complete when  $M$  is presented by a NFA and in NLOGSPACE when it is presented by a DFA. In either

case, it is folklore that whether  $L$  is presented as a NFA or DFA does not matter. A more dramatic example is given when  $L$  is regular and  $M$  context-free, since the inclusion above is undecidable when  $M$  is presented by a CFG and in PTIME when it is presented by a DCFG.

In this section we give a formal explanation of this phenomenon by providing a generic reduction of an inclusion problem as above to one where the l.h.s.  $L$  is a deterministic language. The reduction will be applicable under mild assumptions which are satisfied by most machine-based models of language acceptors such as finite automata, Büchi automata, context-free grammars/pushdown automata, Petri-nets, register automata, timed automata, etc. For the language class of the r.h.s.  $M$  it suffices to have closure under inverse homomorphic images, and for the l.h.s.  $L$  it suffices that we can rename the input symbols read by transitions in a suitable machine model<sup>3</sup>. Moreover, we argue that such transformation preserves whether  $M$  is recognised by a deterministic or an unambiguous machine.

Let  $\Sigma$  be a finite alphabet<sup>4</sup>. Assume that  $L = L(A) \subseteq \Sigma^*$  is recognised<sup>5</sup> by a nondeterministic machine  $A$  with transitions of the form  $\delta = p \xrightarrow{a, \text{op}} q \in \Delta_A$ , where  $\text{op}$  is an optional operation that manipulates a local data structure (a stack, queue, a tape of a Turing machine, etc...). The construction below does not depend on what  $\text{op}$  does. We assume w.l.o.g. that  $A$  is *total*, i.e., for every control location  $p$  and input symbol  $a \in \Sigma$  there exists a transition of the form  $p \xrightarrow{a, \text{op}} \_ \in \Delta_A$ . Consider a new alphabet  $\Sigma' = \Delta_A$ , together with the projection homomorphism  $h : \Sigma' \rightarrow \Sigma$  that maps a transition  $\delta = p \xrightarrow{a, \text{op}} q \in \Delta_A$  to its label  $h(\delta) = a \in \Sigma$ . We modify  $A$  into a new machine  $A'$  by replacing each transition  $\delta$  above with  $p \xrightarrow{\delta, \text{op}} q \in \Delta_{A'}$ . Intuitively,  $A'$  behaves like  $A$  except that it needs to declare which transition  $\delta$  it is actually taking in order to read  $a = h(\delta)$ . By construction,  $A'$  is deterministic (in fact, every transition has a unique label across the entire machine) and  $L(A) = h(L(A'))$  is the homomorphic image of  $L(A')$ .

We need to adapt the machine  $B$  recognising  $M = L(B)$  in order to preserve inclusion. For every transition  $r \xrightarrow{a, \text{op}} s \in \Delta_B$  and for every  $\delta = p \xrightarrow{b, \text{op}'} q \in \Delta_A$  with  $b = a$ , we have in  $B'$  a transition  $r \xrightarrow{\delta, \text{op}} s \in \Delta_{B'}$ . Intuitively,  $B'$  behaves like  $B$  except that it reads additional information on the transition taken by  $A'$ . This information is not actually used by  $B'$  during its execution but it is merely added in order to lift the alphabet from  $\Sigma$  to  $\Sigma'$ . We have  $L(B') = h^{-1}(L(B))$  is the inverse homomorphic image of  $L(B)$ . The following lemma states the correctness of the reduction.

**Lemma 5.** *We have the following equivalence:  $L(A) \subseteq L(B)$  if, and only if,  $L(A') \subseteq L(B')$ .*

*Proof.* By generic properties of images and inverse images we have the following two inclusions:

$$L(A') \subseteq h^{-1}(h(L(A'))) \quad \text{and} \quad h(h^{-1}(L(B))) \subseteq L(B). \quad (3)$$

For the “only if” direction, if  $L(A) \subseteq L(B)$  holds, then  $h^{-1}(L(A)) \subseteq h^{-1}(L(B))$ , which, by the definition of  $A'$  and  $B'$ , is the same as  $h^{-1}(h(L(A))) \subseteq L(B')$ . By (3),  $L(A') \subseteq h^{-1}(h(A')) \subseteq L(B')$ , as required. For the “if” direction, if  $L(A') \subseteq L(B')$  holds, then also  $h(L(A')) \subseteq h(L(B'))$  holds. Similarly as above, we have  $L(A) = h(L(A')) \subseteq h(L(B')) = h(h^{-1}(L(B))) \subseteq L(B)$ , as required.  $\square$

The following lemma states that the reduction above preserves whether  $B$  is deterministic or unambiguous. We mean here the following generic semantic notion of unambiguity:  $B$  is *unambiguous* if for every  $w \in \Sigma^*$ , there exists at most one accepting run of  $B$  over  $w$ . (This notion specialises to the classical notion of unambiguity of finite automata, pushdown automata, Parikh automata, etc.)

<sup>3</sup>The reduction applies also to undecidable instances of the language inclusion problem such as  $\text{CFG} \subseteq \text{DCFG}$ , however in this case it is of no use since  $\text{DCFG} \subseteq \text{DCFG}$  is known to be undecidable [19, Theorem 10.7, Point 2].

<sup>4</sup>The construction below can easily be adapted to infinite alphabets of the form  $\Sigma \times \mathbb{A}$ , where  $\Sigma$  is finite and  $\mathbb{A}$  is an infinite set of data values [3].

<sup>5</sup>Languages of infinite words can be handled similarly.

**Lemma 6.** *If  $B$  is deterministic, then so is  $B'$ . If  $B$  is unambiguous, then so is  $B'$ .*

*Proof.* A transition  $p \xrightarrow{\delta, \text{op}} q \in \Delta_{B'}$  in  $B'$  is obtained taking several distinct copies of a transition  $p \xrightarrow{a, \text{op}} q \in \Delta_B$  in  $B$  w.r.t. every possible transition  $\delta \in \Delta_A$  over the same input symbol  $h(\delta) = a$ . By way of contradiction, assume that  $B$  is deterministic and that  $B'$  is not deterministic. There are two distinct transitions  $p \xrightarrow{\delta, \text{op}_1} q_1, p \xrightarrow{\delta, \text{op}_2} q_2 \in \Delta_{B'}$  in  $B'$  from the same control location  $p$  and input  $\delta \in \Sigma'$ . If  $\delta$  is labelled by  $h(\delta) = a \in \Sigma$ , then by construction there are two distinct transitions  $p \xrightarrow{a, \text{op}_1} q_1, p \xrightarrow{a, \text{op}_2} q_2 \in \Delta_B$  in  $B$  over the same input symbol  $a$ . This contradicts the fact that  $B$  was assumed to be deterministic, and thus  $B'$  must be deterministic as well. An analogous argument shows that also unambiguity is preserved.  $\square$

### 3.1.2 From inclusion to universality

Let  $\mathcal{L}$  and  $\mathcal{M}$  be two classes of languages and let  $L \in \mathcal{L}$  and  $M \in \mathcal{M}$ . A naive approach to decide the inclusion problem (and the most common) is to use the following equivalence:

$$L \subseteq M \quad \text{if, and only if,} \quad L \cap (\Sigma^* \setminus M) = \emptyset. \quad (4)$$

However, this requires complementation of  $M$ , which is either expensive (exponential complexity for NFA) or just impossible (context-free languages are not closed under complementation, even for the unambiguous subclass [18]). However, we observe the following related reduction which works much better in our setting:

$$L \subseteq M \quad \text{if, and only if,} \quad (M \cap L) \cup (\Sigma^* \setminus L) = \Sigma^*. \quad (5)$$

On the face of it, this looks more complicated than (4) because we now have to perform a complementation (of  $L$ ), an intersection, a union, and finally we reduce to the universality problem instead of the nonemptiness, which is still difficult in general. However, in our setting there are gains. First of all, thanks to Section 3.1.1 we can assume that  $L$  is a deterministic language, and thus complementation is usually available (and cheap). Second, while universality is still a difficult problem, it can be easier than inclusion, e.g., DCFG inclusion is undecidable while DCFG universality is decidable (even in PTIME).

In order to apply (5) we require that  $\mathcal{L}$  is a deterministic class efficiently closed under complement (i.e., a representation for the complement is constructible in PTIME) and that the class  $\mathcal{M}$  is closed under disjoint unions and intersections with languages from  $\mathcal{L}$ . Most deterministic languages classes, such as those recognised by deterministic finite automata, deterministic context-free grammars, deterministic Parikh automata, deterministic register automata, etc., satisfy the first requirement<sup>6</sup>. The second requirement is satisfied for classes of languages for which the underlying machine models admit a product construction<sup>7</sup>.

### 3.1.3 Applications

In this section we apply the reductions of Section 3.1.1 and Section 3.1.2 in order to reduce certain inclusion problems to their respective universality variant.

<sup>6</sup> A notable exception is deterministic Petri-net languages under coverability semantics, since the complement of such languages intuitively requires checking whether some counter is negative, which is impossible without zero tests. In fact, if both a language and its complement are deterministic Petri-net recognisable under coverability semantics, then they are both regular [9].

<sup>7</sup> As an example not satisfying this requirement, one can take  $\mathcal{L} = \mathcal{M}$  to be the class of DCFL, since they are not closed under intersection. In fact, while we show in this paper that UUCFG is decidable, the equivalence problem for UCFG is open.

**Theorem 7.** “ $NFA \subseteq UFA$ ” is in PTIME.

While equivalence and inclusion of UFA is well-known to be in PTIME [33, Corollary 4.7], the same complexity for the more general problem “ $NFA \subseteq UFA$ ” does not seem to have been observed before.

*Proof.* By Section 3.1.1, the problem reduces to “ $DFA \subseteq UFA$ ”. By (5),  $L \subseteq M$  is equivalent to  $N := M \cap L \cup (\Sigma^* \setminus L) = \Sigma^*$ . Notice that  $N$  is effectively UFA, since the DFA language  $L$  can be complemented in PTIME, the intersection  $M \cap L$  is also UFA and computable in quadratic time, and the disjoint union of a UFA and a DFA is also a UFA computable in linear time. Since the universality problem for unambiguous automata can be solved in PTIME, also “ $DFA \subseteq UFA$ ”, and thus “ $NFA \subseteq UFA$ ”, is in PTIME as well.  $\square$

**Theorem 8.** “ $NFA \subseteq UCFG$ ” is PTIME inter-reducible with UUCFG.

*Proof.* By Section 3.1.1, the problem reduces to “ $DFA \subseteq UCFG$ ”. Thanks to Section 3.1.2, the latter problem reduces to UUCFG since 1) DFA languages are efficiently closed under complement (in PTIME), 2) UCFG languages are efficiently closed under intersection with DFA languages (in PTIME), and 3) the disjoint union of a UCFG language and a DFA language is a UCFG language. Thus, “ $NFA \subseteq UCFG$ ” reduces to UUCFG, and since UUCFG is a special case of the former problem, “ $NFA \subseteq UCFG$ ” is PTIME inter-reducible with UUCFG.  $\square$

**Theorem 9.** “ $CFG \subseteq UFA$ ” reduces to UUCFG.

*Proof.* By Section 3.1.1, “ $CFG \subseteq UFA$ ” reduces to “ $DCFG \subseteq UFA$ ”, which in turn reduces to UUCFG thanks to Section 3.1.2 because 1) DCFG languages are efficiently closed under complement, 2) the intersection of a UFA and a DCFG language is efficiently DCFG, and 3) the disjoint union of two DCFG languages is efficiently UCFG. (The latter problem reduces to universality of two disjoint DCFG languages, which in principle may be easier than UUCFG.)  $\square$

### 3.2 UUCFG in PSPACE

In this section we show that UUCFG is in PSPACE by reducing to the zeroness problem for conv-rec sequences. This complexity upper bound appears also in [28], albeit with a more direct argument reducing to systems of monotone polynomial equations.

Let  $\Sigma = \{a, b\}$  be a finite alphabet and let  $L \subseteq \Sigma^*$  be a language of finite words over  $\Sigma$ . The *counting function* of  $L$  is the sequence  $f_L : \mathbb{N} \rightarrow \mathbb{N}$  s.t. for every  $n \in \mathbb{N}$ ,  $f_L(n) = |L \cap \Sigma^n|$  counts the number of words of length  $n$  in  $L$ . Given a unambiguous context-free grammar  $G = (\Sigma, N, S, \leftarrow)$  in short Greibach normal form, let  $f_X := f_{L(X)} : \mathbb{N} \rightarrow \mathbb{N}$  be the counting function of the language  $L(X)$  recognised by the nonterminal  $X \in N$ . It is well-known that the  $f_X$ ’s satisfy the following system of equations with convolution:

$$f_X(n+1) = \sum_{X \leftarrow aYZ} (f_Y * f_Z)(n). \quad (6)$$

The initial condition is  $f_X(0) = 1$  if  $X \leftarrow \varepsilon$  and  $f_X(0) = 0$  otherwise. In other words,  $f_S$ , which is the counting function of the language  $L(G)$  recognised by  $G$ , is conv-rec. Unambiguity is used crucially to show that any word  $w$  in  $L(Y \cdot Z)$  factorises uniquely as  $w = u \cdot v$  with  $u \in L(Y)$  and  $v \in L(Z)$ , which allows us to obtain  $f_{L(Y \cdot Z)} = f_{L(Y)} * f_{L(Z)}$ .



Clearly,  $G$  is universal if, and only if,  $f_S$  is identically equal to the sequence  $g(n) = 2^n$ . The latter sequence is conv-rec since it satisfies  $g(n+1) = (g * g)(n)$ , with the initial condition  $g(0) = 1$ . Thus  $G$  is universal if, and only if,  $f(n) = g(n) - f_S(n)$  is identically zero. Since conv-rec sequences are closed under subtraction,  $f(n)$  is also conv-rec. By Theorem 4, we can decide zeroness of  $f$  in PSPACE, and thus the same upper bound holds for UUCFG.

**Theorem 10.** *The universality problem for unambiguous context-free grammars UUCFG is in PSPACE.*

## 4 SQRTSUM-hardness of coin-flip measure

In this section we show that a quantitative generalisation of UUCFG is hard for a well-known problem in numerical computing. Let  $\Sigma_n = \{a_1, \dots, a_n\}$  be a finite alphabet of  $n$  distinct letters. Consider the following random process to generate a finite word in  $\Sigma^*$ . At step  $k$  we select one option  $a_k \in \Sigma_\varepsilon = \Sigma_n \cup \{\varepsilon\}$  uniformly at random. If  $a_k = \varepsilon$ , then we terminate and we produce in output  $a_0 \dots a_{k-1}$ . Otherwise, we continue to the next step  $k+1$ . It is easy to see that the probability to generate a word depends only on its length and equals  $\mu_{\text{coin}}(w) = \left(\frac{1}{|\Sigma|+1}\right)^{|w|+1}$ . The *coin-flip measure* of a language of finite words  $L \subseteq \Sigma^*$  is  $\mu_{\text{coin}}(L) = \sum_{w \in L} \mu_{\text{coin}}(w)$ . Clearly,  $0 \leq \mu_{\text{coin}}(L) \leq 1$ ,  $\mu_{\text{coin}}(L) = 0$  iff  $L = \emptyset$ , and  $\mu_{\text{coin}}(L) = 1$  iff  $L = \Sigma^*$ .

Since  $\mu_{\text{coin}}(w)$  depends just on  $|w|$ , we can write  $\mu_{\text{coin}}(L) = \sum_{k=0}^{\infty} f_L(k) \cdot \left(\frac{1}{n+1}\right)^{k+1}$ , where  $f_L(k) = |L \cap \Sigma^k|$  is the counting function of  $L$ . In other words, one possible way of computing the coin-flip measure is by evaluating the generating function  $g_{f_L}(x)$  at  $x = \frac{1}{n+1}$  (modulo a correction factor):  $\mu_{\text{coin}}(L) = \frac{1}{n+1} \cdot g_{f_L}\left(\frac{1}{n+1}\right)$ . Consequently, the coin-flip measure of a regular language is rational, and that of an unambiguous context-free language is algebraic (following from the analogous, and more general, facts about the respective generating functions [7]). Let  $L, M \subseteq \Sigma_n^*$  be two languages with unambiguous concatenation  $L \cdot M$ . Then

$$\mu_{\text{coin}}(L \cdot M) = (n+1) \cdot \mu_{\text{coin}}(L) \cdot \mu(M). \quad (7)$$

The *coin-flip comparison problem* asks, given a language  $L \subseteq \Sigma^*$ , a rational threshold  $0 \leq \varepsilon \leq 1$  encoded in binary, and a comparison operator  $\sim \in \{\leq, <, >, \geq\}$ , whether  $\mu_{\text{coin}}(L) \sim \varepsilon$  holds. The universality problem for  $L$  is the special case when  $\varepsilon = 1$ . We now relate the coin-flip comparison problem to an open problem in numerical computing. The *SQRTSUM problem* asks, given  $d_0, \dots, d_n \in \mathbb{N}$  encoded in binary and a comparison operator  $\sim \in \{\leq, <, >, \geq\}$ , whether <sup>8</sup>:

$$\sum_{i=1}^n \sqrt{d_i} \sim d_0. \quad (8)$$

This problem can be shown to be in PSPACE by deciding the existential formula  $\exists x_1, \dots, x_n. x_1^2 = d_1 \wedge \dots \wedge x_n^2 = d_n \wedge x_1 + \dots + x_n \sim d_0$  over the reals [6]. It is a long-standing open problem in the theory of numerical computation whether SQRTSUM is in NP, or whether it is NP-hard [1, 13].

**Theorem 11.** *The coin-flip measure comparison problem is SQRTSUM-hard for UCFG.*

<sup>8</sup> In fact, the problem reduces to the case when  $\sim = \geq$  is fixed. By doing binary search in the interval  $\{0, 1, \dots, n \cdot d\}$ , with only  $O(\log(n \cdot d))$  queries to (8) we can find the unique  $\hat{d}_0 \in \mathbb{N}$  s.t.  $\hat{d}_0 \leq \sum_{i=1}^n \sqrt{d_i} \leq \hat{d}_0 + 1$ . We can then solve  $\sum_{i=1}^n \sqrt{d_i} \leq d_0$  by checking  $d_0 \leq \hat{d}_0 + 1$ , and similarly for the other comparison operators.

In the rest of the section we prove the theorem above. Let  $d_0, \dots, d_n \in \mathbb{N}$  be the input to Sqrt-SUM. We assume w.l.o.g. that  $n$  is an odd number  $\geq 3$ . We construct a rational constant  $\varepsilon \in \mathbb{Q}$  and a UCFG  $G = (\Sigma_n, N, X_0, \leftarrow)$  over a  $n$ -ary alphabet  $\Sigma_n = \{a_1, \dots, a_n\}$  and nonterminals  $N$  containing  $\{X_0, \dots, X_n, C_1, \dots, C_n, A\}$  plus some auxiliary nonterminals (omitted for readability) s.t.  $\mu_{\text{coin}}(L(G)) \sim \varepsilon$  if, and only if, (8) holds. The principal productions of the grammar are:

$$\begin{aligned} X_0 &\leftarrow a_1 \cdot X_1 \mid \dots \mid a_n \cdot X_n, \\ X_1 &\leftarrow C_1 \mid A \cdot X_1 \cdot a_n \cdot X_1, \\ &\vdots \\ X_n &\leftarrow C_n \mid A \cdot X_n \cdot a_n \cdot X_n. \end{aligned}$$

The remaining nonterminals  $C_i$ 's and  $A$  will generate certain regular languages to be determined below. Let  $d = \max_{i=1}^n d_i$ . For every  $1 \leq i \leq n$ , let  $x_i = 1 - \frac{\sqrt{d_i}}{d}$ . It is easy to check that  $x_i$  is the least non-negative solution of

$$x_i = c_i + a \cdot x_i^2 \quad \text{where } c_i := \frac{1}{2} \left(1 - \frac{d_i}{d^2}\right) \text{ and } a := \frac{1}{2}. \quad (9)$$

In the following, we write  $\mu(X)$  for a non-terminal  $X \in N$  as a shorthand for  $\mu_{\text{coin}}(L(X))$ . Since  $\mu(a_1) = \dots = \mu(a_n) = \frac{1}{(n+1)^2}$ , by (7) we have

$$\mu(X_0) = \frac{1}{n+1}(\mu(X_1) + \dots + \mu(X_n)) \text{ and } \mu(X_i) = \mu(C_i) + (n+1) \cdot \mu(A) \cdot \mu(X_i)^2, i \in \{1, \dots, n\}. \quad (10)$$

We aim at obtaining  $\mu(X_i) = x_i$ . By comparing (10) with (9) we deduce that the nonterminals  $C_i$  and  $A$  must generate languages of measure  $\mu(C_i) = c_i$ , resp.,  $\mu(A) = \frac{a}{n+1} = \frac{1}{2(n+1)}$ . Since the measures  $a, c_i$  are rational, it suffices to find regular languages  $L(A), L(C_i)$ . The main difficulty is to define these language as to ensure that  $G$  is unambiguous and of polynomial size. In order to achieve this we further require that 1)  $L(A) \subseteq \Sigma_{n-1}$  is a finite set of words of length 1 (single letters) not containing letter  $a_n$ , and 2)  $L(C_i) \subseteq \Sigma_{n-1}^*$  is a set of words not containing letter  $a_n$ .

We first define  $L(A)$ . Let

$$A \leftarrow a_1 \mid \dots \mid a_{\frac{n+1}{2}}.$$

In order to avoid letter  $a_n$ , we require  $\frac{(n+1)}{2} \leq n-1$ . The latter condition is satisfied since we assumed  $n \geq 3$ . Thus,  $L(A) \subseteq \Sigma_{n-1}$  is finite, contains only words of length 1, and has measure  $\mu(A) = \frac{n+1}{2} \cdot \frac{1}{(n+1)^2} = \frac{a}{n+1}$ , as required.

The definition of  $L(C_i)$  of measure  $\mu(C_i) = c_i$  is more involved. In general, it is easy to construct a regular expression (or a finite automaton) recognising a language of measure equal to a given rational number. However, we have two constraints to respect: 1) we can use only letters from  $\Sigma_{n-1}$ , and 2) the regular expression must have size polynomial in the bit encoding of  $c_i$ . The first constraint entails an upper bound  $\mu(\Sigma_{n-1}^*) = \frac{1}{2}$  on the maximal measure that  $L(C_i)$  can have. However, this is not a problem in our case since  $c_i < \frac{1}{2}$  by definition. The second constraint is handled by the following lemma.

**Lemma 12** (Representation lemma). *Let  $n+1 \in \mathbb{N}$  with  $n \geq 2$  be a base, let  $m \in \mathbb{N}$  s.t.  $1 \leq m \leq n$ , and let  $c \in \mathbb{R}$  with  $0 \leq c \leq \frac{1}{n-m+1}$  be a target rational measure written in reduced form as  $c = \frac{p}{q}$ , with  $p, q \in \mathbb{N}$ ,  $p \leq q$ . There exists an unambiguous regular expression  $e$  using only letters from  $\Sigma_m \subseteq \Sigma_n$  recognising a language of measure  $\mu(L(e)) = c$ . Moreover, if there exists  $\ell \in \mathbb{N}$  s.t.  $q \mid (n+1)^\ell$ , then  $e$  can be taken of size polynomial in  $\log q$ ,  $n$ , and  $\ell$ .*

We apply Lemma 12 with  $m := n - 1$  and  $c := c_i$  and obtain an unambiguous regular expression  $e$  recognising a language  $L(e) \subseteq \Sigma_{n-1}^*$  of measure  $c_i$ . We now argue that  $e$  can be taken of polynomial size. In order to achieve this, we assume w.l.o.g. that  $d = (n+1)^{2h}$  for some  $h$ . (This can be ensured by adding a new integer  $d_{n+1} = (n+2)^{2h}$  for some  $h$  large enough, and by replacing  $d_0$  with  $d_0 + \sqrt{d_{n+1}} = d_0 + (n+2)^h$ .) Consequently,  $c_i = \frac{d(d^2 - d_i)}{(n+1)^{4h}} = \frac{p}{q}$  with  $p, q \in \mathbb{N}$  relatively prime and  $q \mid (n+1)^{4h}$ , and thus  $e$  has polynomial size by taking  $\ell = 4h$  in the lemma. The set of polynomially many production rules for nonterminal  $C_i$  is derived immediately from the regular expression  $e$  by adding some auxiliary nonterminals. Moreover, since  $e$  is unambiguous, the same applies to the rules for  $C_i$ . This completes the description of the grammar  $G$ .

**Lemma 13.** *The grammar  $G$  is unambiguous.*

*Proof.* Since  $L(G) = L(X_0)$  is the union of languages  $L(a_1 \cdot X_1), \dots, L(a_n \cdot X_n)$ , and the latter are disjoint, it suffices to show that the  $L(X_i)$ 's are recognised unambiguously. Let  $w \in L(X_i)$ . If  $w$  does not contain any  $a_n$ , then necessarily  $w \in L(C_i)$ . Otherwise, let  $w = ua_nv$  where  $v$  does not contain any  $a_n$ . Thus  $v \in L(C_i)$  and  $u \in L(A \cdot X_i)$ . Since  $A$  produces only words of fixed length,  $u = xw'$  unambiguously with  $x \in A$  and  $w' \in L(X_i)$ . This argument shows that for any  $w \in L(X_i)$  if we let  $s$  be the number of  $a_n$  in  $w$ , then  $w \in L(A^s \cdot C_i \cdot (a_n \cdot C_i)^s)$ . Since  $A$  produces words of fixed length and  $C_i$  does not produce any word containing  $a_n$ , the latter concatenation is unambiguous and thus  $w$  is produced unambiguously by  $X_i$ .  $\square$

Let  $\varepsilon := \frac{1}{n+1} \left( n - \frac{d_0}{d} \right)$ . The following lemma states the correctness of the reduction.

**Lemma 14.** *We have  $\mu(L(G)) \sim \varepsilon$  if, and only if, (8) holds.*

*Proof.* Since  $x_i = 1 - \frac{\sqrt{d_i}}{d}$ , we have  $\mu(X_0) = \mu(a_1 \cdot X_1) + \dots + \mu(a_n \cdot X_n) = (n+1)(\mu(a_1) \cdot \mu(X_1) + \dots + \mu(a_n) \cdot \mu(X_n)) = \frac{1}{n+1}(\mu(X_1) + \dots + \mu(X_n)) = \frac{1}{n+1}(x_1 + \dots + x_n) = \frac{1}{n+1} \left( \left( 1 - \frac{\sqrt{d_1}}{d} \right) + \dots + \left( 1 - \frac{\sqrt{d_n}}{d} \right) \right) = \frac{1}{n+1} \left( n - \frac{\sqrt{d_1} + \dots + \sqrt{d_n}}{d} \right)$ , and thus  $\sum_{i=1}^n \sqrt{d_i} \sim d_0$  if, and only if,  $\mu(L(X)) \sim \varepsilon$ , as required.  $\square$

## 5 Discussion

We have shown novel PSPACE upper bounds for several inclusion problems on UCFG and finite automata. We did not address language equivalence problems  $L = M$ , which in principle can be easier to decide than the corresponding inclusions. For instance, while  $\text{DCFG} \subseteq \text{DCFG}$  is undecidable [19, Theorem 10.7, Point 2],  $\text{DCFG} = \text{DCFG}$  is decidable by the result of G. Sénizergues [30]. It is worth remarking that decidability of the equivalence problem  $\text{UCFG} = \text{UCFG}$  is not known. In fact, this is a special case of the *multiplicity equivalence problem* for CFG, which asks whether two CFGs have the same number of derivations for every word they accept. Decidability of the latter problem is open as well [22] and inter-reducible with the language equivalence for probabilistic pushdown automata [14]. The restriction of the  $\text{UCFG} = \text{UCFG}$  equivalence problem to words of a given length has been studied in [24].

**Number sequences and the zeroness problem.** We obtained the PSPACE upper bound for UUCFG by reducing to the zeroness problem for conv-rec sequences. Conv-rec sequences generalise linear difference recurrence with constant coefficients (a.k.a. *constant-recursive* or *C-finite* [21]; c.f. also [2] and citations therein) by allowing the convolution product in the recurrence. They are a special case of more expressive classes such as P-recursive [21, Ch. 7] (a.k.a. *holonomic*) and polynomial recursive sequences

[5]. The zeroness problem for P-recursive sequences is decidable [36] and the same holds for polynomial recursive sequences (as a corollary of the existence of cancelling polynomials [5, Theorem 11]). However, no complexity upper bounds are known for those more general classes.

**Coin-flip measure.** As a complement to the PSPACE upper bound for UUCFG, we have shown that the coin-flip measure comparison problem  $\mu_{\text{coin}}(L(G)) \sim \varepsilon$  of a UCFG  $G$  with  $\sim \in \{\leq, <, \geq, >\}$  and  $0 \leq \varepsilon \leq 1$  is SQRTSUM-hard. The main difficulty is that the measure is generated according to a fixed stochastic process. If we relax this constraint and generate the measure according to an arbitrary finite Markov process, then one can obtain SQRTSUM-hardness already for DCFG.

It is known that the quantitative decision problem for  $\mu_G(\Sigma^*)$  where  $G$  is a stochastic context-free grammar (SCFG) is SQRTSUM-hard [13]. Our setting is incomparable: On the one hand we fix a particular measure, namely the coin-flip measure  $\mu_{\text{coin}}$  (which corresponds to a fixed SCFG with rules  $X \leftarrow \varepsilon \mid a_1 \cdot X \mid \dots \mid a_n \cdot X$ ). On the other hand, we are interested in the quantity  $\mu_{\text{coin}}(L(G))$  where  $G$  is an arbitrary UCFG (and thus not necessarily universal).

We leave it as an open problem to establish the exact complexity of the universality problem for UCFG and the coin-flip measure 1 problem. When the system of polynomial equations obtained from the grammar is *probabilistic* (PPS<sup>9</sup>) the measure 1 problem is in PTIME [13] (and even in strongly polynomial time [11]). However, the equations obtained from UCFG are monotone (MPS) but not PPS in general. As an example, consider a singleton alphabet  $\Sigma = \{a\}$  and productions of the form  $X_0 \leftarrow a$  and, for  $n \geq 0$ ,  $X_{n+1} \leftarrow X_n \cdot X_n$ . The corresponding MPS system is  $x_0 = \frac{1}{2^2}$  and  $x_{n+1} = 2 \cdot x_n^2$ . The former system is not a PPS, since in the second equation the coefficients sum up to 2. It may be argued that by the change of variable  $z_n := 2 \cdot x_n$  we obtain the system  $z_0 = \frac{1}{2}$  and  $z_{n+1} = z_n^2$  which is PPS. However, this transformation reduces the value 1 problem on the original MPS to the value 1/2 problem in the new PPS, and the latter problem is not known to be in PTIME.

One source of difficulty in the UUCFG problem is that witnesses of non-universality can have exponential length. Extending the previous example, consider the additional rules  $Y_0 \leftarrow \varepsilon$  and  $Y_{n+1} \leftarrow Y_n \mid X_n \cdot Y_n$ . The nonterminal  $X_n$  generates a single word  $L(X_n) = \{a^{2^n}\}$  of length  $2^n$ . It can be verified by induction that  $Y_n$  generates all words  $L(Y_n) = \{a^0, a^1, \dots, a^{2^n-1}\}$  of length  $\leq 2^n - 1$ , and consequently the grammar is unambiguous. Thus  $L(Y_n)$  is not universal, however the shortest witness has length  $2^n$ . In terms of measures,  $\mu_{\text{coin}}(L(X_n)) = \frac{1}{2^{2^n+1}}$  and  $\mu_{\text{coin}}(L(Y_n)) = 1 - \frac{1}{2^{2^n+1}}$ , and thus UCFG have measures that can be exponentially close to 0, resp., to 1. Since a word of length  $n$  over a unary alphabet has measure  $\frac{1}{2^{n+1}} = 2^{-O(n)}$ , if language  $L$  is not universal  $\mu_{\text{coin}}(L) < 1$ , then there is a non-universality witness of length at most  $\log(1 - \mu_{\text{coin}}(L))$ . Thus upper bounds on  $1 - \mu_{\text{coin}}(L)$  yield upper bounds on the shortest non-universality witness.

**The “CFG  $\subseteq$  UFA” problem.** We have shown that CFG  $\subseteq$  UFA reduces to DCFG  $\subseteq$  UFA and, in turn, the latter reduces to UUCFG and thus can be solved in PSPACE. This needs not be optimal and there are reasons to suspect that better algorithms may be obtained. If we interpret a DCFG  $G$  as a stochastic context-free grammar (SCFG), then query  $L(G) \subseteq L(A)$  is equivalent to  $\mu_G(L(A)) = 1$  when  $A$  is unambiguous, where  $\mu_G$  is the measure generated by  $G$  (a generalisation of the coin-flip measure). When  $A$  is DFA,  $\mu_G(L(A))$  can be approximated in PTIME [12]. Generalising this result for  $A$  being UFA would put DCFG  $\subseteq$  UFA in PTIME.

---

<sup>9</sup>The sum of all coefficients is at most 1.

**The regularity problem for UCFG.** There are other problems which are known to be undecidable for CFG but decidable for DCFG, such as the regularity problem [32, 34, 31]. An interesting open problem [10] is whether the regularity problem is decidable for UCFG.

**Acknowledgements.** I warmly thank Alberto Pettorossi for his encouragement and guidance during my first steps in doing research. I also thank an anonymous reviewer for his helpful comments on a preliminary version of this draft.

## References

- [1] Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen & Peter Bro Miltersen (2009): *On the Complexity of Numerical Analysis*. *SIAM J. Comput.* 38(5), pp. 1987–2006, doi:10.1137/070697926. Available at <http://dx.doi.org/10.1137/070697926>.
- [2] Corentin Barloy, Nathanaël Fijalkow, Nathan Lhote & Filip Mazowiecki (2020): *A Robust Class of Linear Recurrence Sequences*. In Maribel Fernández & Anca Muscholl, editors: *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, *LIPIcs* 152, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 9:1–9:16.
- [3] Mikołaj Bojańczyk, Bartek Klin & Sławomir Lasota (2014): *Automata theory in nominal sets*. *Logical Methods in Computer Science* Volume 10, Issue 3, doi:10.2168/LMCS-10(3:4)2014. Available at <https://lmcs.episciences.org/1157>.
- [4] Alin Bostan, Arnaud Carayol, Florent Koechlin & Cyril Nicaud (2020): *Weakly-unambiguous Parikh automata and their link to holonomic series*. In: *To appear in ICALP’20*.
- [5] Michaël Cadilhac, Filip Mazowiecki, Charles Paperman, Michał Pilipczuk & Géraud Sénizergues (2020): *On polynomial recursive sequences*. In: *To appear in ICALP’20*.
- [6] John Canny (1988): *Some Algebraic and Geometric Computations in PSPACE*. In: *Proc. of STOC ’88*, ACM, New York, NY, USA, pp. 460–467, doi:10.1145/62212.62257. Available at <http://doi.acm.org/10.1145/62212.62257>.
- [7] N. Chomsky & M. P. Schützenberger (1963): *The Algebraic Theory of Context-Free Languages*. In P. Braffort & D. Hirschberg, editors: *Computer Programming and Formal Systems, Studies in Logic and the Foundations of Mathematics* 35, Elsevier, pp. 118 – 161, doi:[https://doi.org/10.1016/S0049-237X\(08\)72023-8](https://doi.org/10.1016/S0049-237X(08)72023-8). Available at <http://www.sciencedirect.com/science/article/pii/S0049237X08720238>.
- [8] Wojciech Czerwiński, Diego Figueira & Piotr Hofman (2020): *Universality Problem for Unambiguous VASS*. Available at <https://hal.archives-ouvertes.fr/hal-02483495>. Working paper or preprint.
- [9] Wojciech Czerwinski, Sławomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar & Prakash Saivasan (2018): *Regular Separability of Well-Structured Transition Systems*. In Sven Schewe & Lijun Zhang, editors: *Proc. of CONCUR’18*, *LIPIcs* 118, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 35:1–35:18, doi:10.4230/LIPIcs.CONCUR.2018.35. Available at <http://drops.dagstuhl.de/opus/volltexte/2018/9573>.
- [10] Volker Diekert & Steffen Kopecki (2010): *Complexity Results and the Growths of Hairpin Completions of Regular Languages*. In: *Proc. of CIAA’10*, *CIAA’10*, Springer-Verlag, Berlin, Heidelberg, pp. 105–114.
- [11] Javier Esparza, Andreas Gaiser & Stefan Kiefer (2010): *Computing Least Fixed Points of Probabilistic Systems of Polynomials*. In Jean-Yves Marion & Thomas Schwentick, editors: *STACS’10*, *LIPIcs* 5, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 359–370, doi:10.4230/LIPIcs.STACS.2010.2468. Available at <http://drops.dagstuhl.de/opus/volltexte/2010/2468>.
- [12] Kousha Etessami, Alistair Stewart & Mihalis Yannakakis (2013): *Stochastic Context-free Grammars, Regular Languages, and Newton’s Method*. In: *Proc. of ICALP’13*,

- Springer-Verlag, pp. 199–211, doi:10.1007/978-3-642-39212-2\_20. Available at [http://dx.doi.org/10.1007/978-3-642-39212-2\\_20](http://dx.doi.org/10.1007/978-3-642-39212-2_20).
- [13] Kousha Etessami & Mihalis Yannakakis (2009): *Recursive Markov Chains, Stochastic Grammars, and Monotone Systems of Nonlinear Equations*. *J. ACM* 56(1), pp. 1:1–1:66, doi:10.1145/1462153.1462154. Available at <http://doi.acm.org/10.1145/1462153.1462154>.
- [14] Vojtěch Forejt, Petr Jančar, Stefan Kiefer & James Worrell (2014): *Language equivalence of probabilistic pushdown automata*. *Information and Computation* 237, pp. 1–11, doi:<https://doi.org/10.1016/j.ic.2014.04.003>. Available at <http://www.sciencedirect.com/science/article/pii/S0890540114000625>.
- [15] Ronald Graham, Donald Knuth & Oren Patashnik (1994): *Concrete Mathematics*, 2nd edition. Addison Wesley.
- [16] D. Yu Grigor’ev (1988): *Complexity of deciding Tarski algebra*. *Journal of Symbolic Computation* 5(1), pp. 65–108, doi:[https://doi.org/10.1016/S0747-7171\(88\)80006-3](https://doi.org/10.1016/S0747-7171(88)80006-3). Available at <http://www.sciencedirect.com/science/article/pii/S0747717188800063>.
- [17] Wright E.M. Hardy G.H. (2008): *An introduction to the theory of numbers*, 6ed. edition. OUP.
- [18] Thomas N. Hibbard & Joseph Ullian (1966): *The Independence of Inherent Ambiguity From Complementedness Among Context-Free Languages*. *J. ACM* 13(4), pp. 588–593.
- [19] John Hopcroft & Jeffrey Ullman (1979): *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.
- [20] Takumi Kasai & Shigeki Iwata (1992): *Some Problems in Formal Language Theory Known as Decidable are Proved EXPTIME Complete*. 数理解析研究所講究録 796.
- [21] Manuel Kauers & Peter Paule (2011): *C-Finite Sequences*, pp. 63–86. Springer Vienna, Vienna, doi:10.1007/978-3-7091-0445-3\_4. Available at [https://doi.org/10.1007/978-3-7091-0445-3\\_4](https://doi.org/10.1007/978-3-7091-0445-3_4).
- [22] Werner Kuich (1994): *On the Multiplicity Equivalence Problem for Context-Free Grammars*. In: *Proceedings of the Colloquium in Honor of Arto Salomaa on Results and Trends in Theoretical Computer Science*, Springer-Verlag, Berlin, Heidelberg, pp. 232–250.
- [23] Werner Kuich & Arto Salomaa (1986): *Semirings, Automata, Languages*. EATCS Monographs on Theoretical Computer Science 5, Springer. Available at <http://gen.lib.rus.ec/book/index.php?md5=c22e327ec8a3a010137c7809f7a5abe1>.
- [24] B. Litow (1996): *Bounded length UCFG equivalence*. In Tetsuo Asano, Yoshihide Igarashi, Hiroshi Nagamochi, Satoru Miyano & Subhash Suri, editors: *Algorithms and Computation*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 239–246.
- [25] P. Massazza & N. Sabadini (1989): *Some applications and techniques for generating functions*. In Josep Díaz & Fernando Orejas, editors: *TAPSOFT ’89*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 321–336.
- [26] A. R. Meyer & L. J. Stockmeyer (1972): *The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space*. In: *Proceedings of the 13th Annual Symposium on Switching and Automata Theory, SWAT ’72*, IEEE Computer Society, Washington, DC, USA, pp. 125–129.
- [27] Antoine Mottet & Karin Quaas (2019): *The Containment Problem for Unambiguous Register Automata*. In Rolf Niedermeier & Christophe Paul, editors: *Proc. of STACS’19, LIPIcs* 126, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 53:1–53:15, doi:10.4230/LIPIcs.STACS.2019.53. Available at <http://drops.dagstuhl.de/opus/volltexte/2019/10292>.
- [28] Stanisław Purgał (2018): *Learning regular languages offline from a positive sample*. Master’s thesis, University of Warsaw.
- [29] Arto Salomaa & Marti Soittola (1978): *Automata-theoretic aspects of formal power series*. Texts and Monographs in Computer Science, Springer. Available at <http://gen.lib.rus.ec/book/index.php?md5=1E7B3DCFFFC7F6146062637382E5067D>.

- [30] Géraud Sénizergues (1997): *The equivalence problem for deterministic pushdown automata is decidable*. In Pierpaolo Degano, Roberto Gorrieri & Alberto Marchetti-Spaccamela, editors: *Proc. of ICALP'97*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 671–681.
- [31] Priti Shankar & B. S. Adiga (1991): *A Graph-based Regularity Test for Deterministic Context-free Languages*. *Theor. Comput. Sci.* 88(1), pp. 117–125, doi:10.1016/0304-3975(91)90076-E. Available at [http://dx.doi.org/10.1016/0304-3975\(91\)90076-E](http://dx.doi.org/10.1016/0304-3975(91)90076-E).
- [32] R.E. Stearns (1967): *A regularity test for pushdown machines*. *Information and Control* 11(3), pp. 323 – 340, doi:http://dx.doi.org/10.1016/S0019-9958(67)90591-8. Available at <http://www.sciencedirect.com/science/article/pii/S0019995867905918>.
- [33] Richard E. Stearns & Harry B. Hunt (1981): *On the equivalence and containment problems for unambiguous regular expressions, grammars, and automata*. In: *Proc. of SFCS'81*, IEEE Computer Society, Washington, DC, USA, pp. 74–81, doi:10.1109/SFCS.1981.29. Available at <http://dx.doi.org/10.1109/SFCS.1981.29>.
- [34] Leslie G. Valiant (1975): *Regularity and Related Problems for Deterministic Push-down Automata*. *J. ACM* 22(1), pp. 1–10, doi:10.1145/321864.321865. Available at <http://doi.acm.org/10.1145/321864.321865>.
- [35] Tzeng Wen-Guey (1996): *On path equivalence of nondeterministic finite automata*. *Information Processing Letters* 58(1), pp. 43–46.
- [36] Doron Zeilberger (1990): *A holonomic systems approach to special functions identities*. *Journal of Computational and Applied Mathematics* 32(3), pp. 321–368, doi:https://doi.org/10.1016/0377-0427(90)90042-X. Available at <http://www.sciencedirect.com/science/article/pii/037704279090042X>.

## A Appendix

**Lemma 12** (Representation lemma). *Let  $n + 1 \in \mathbb{N}$  with  $n \geq 2$  be a base, let  $m \in \mathbb{N}$  s.t.  $1 \leq m \leq n$ , and let  $c \in \mathbb{R}$  with  $0 \leq c \leq \frac{1}{n-m+1}$  be a target rational measure written in reduced form as  $c = \frac{p}{q}$ , with  $p, q \in \mathbb{N}$ ,  $p \leq q$ . There exists an unambiguous regular expression  $e$  using only letters from  $\Sigma_m \subseteq \Sigma_n$  recognising a language of measure  $\mu(L(e)) = c$ . Moreover, if there exists  $\ell \in \mathbb{N}$  s.t.  $q \mid (n+1)^\ell$ , then  $e$  can be taken of size polynomial in  $\log q$ ,  $n$ , and  $\ell$ .*

*Proof.* Fix an alphabet  $\Sigma_n$  and let  $m \leq n$  as in the statement of the lemma. If  $m = n$  then there is no difficulty since we can just look at the periodic expansion  $c = \sum_{i=0}^{\infty} \frac{c_i}{(n+1)^{i+1}}$  of  $c$  in base  $n+1$ , where crucially  $0 \leq c_i \leq n$ , and one can just interpret  $c_i$  as the number of words of length  $i$  that the regular expression must accept (since there are  $n^i$  such words of length  $i$ , this can always be done, perhaps with the exception of  $i = 0$ ). However, we work under the more general constraint  $m \leq n$ , which requires a refinement of the analysis above.

There are  $m^k$  words of length  $k$  over  $\Sigma_m \subseteq \Sigma_n$ , and thus the measure  $\mu_{\text{coin}}(\Sigma_m^*)$  (which is always computed w.r.t. alphabet  $\Sigma_n$ ) satisfies

$$\mu_{\text{coin}}(\Sigma_m^*) = \sum_{k=0}^{\infty} \frac{m^k}{(n+1)^{k+1}} = \frac{1}{n+1} \cdot \sum_{k=0}^{\infty} \left( \frac{m}{n+1} \right)^k = \frac{1}{n+1} \cdot \frac{1}{1 - \frac{m}{n+1}} = \frac{1}{n-m+1}. \quad (11)$$

We consider two cases. If  $c = \frac{1}{n-m+1}$ , then just let  $e = \Sigma_m^*$ , and we are done. Otherwise, assume  $c < \frac{1}{n-m+1}$ . Our aim is to find some  $k \in \mathbb{N}$  and write  $c$  as

$$c = \sum_{i=0}^k \frac{c_i}{(n+1)^{i+1}} + \sum_{j=1}^{\infty} \frac{d_j}{(n+1)^{k+j+1}}, \quad \text{with } 0 \leq c_i \leq m^i \text{ and } 0 \leq d_j \leq n. \quad (12)$$

Intuitively,  $c_i$  counts the number of words of length  $i$  in  $L(e)$  for  $0 \leq i \leq k$ , and  $d_j$  counts the number of words of length  $k+j$  in  $L(e)$  for  $j \geq 1$ . Moreover, in the first part the  $c_i$ 's are allowed to be very big (up to  $m^i$ ) and in the second part the  $d_j$ 's must be small (up to  $n$ ). Thus we will be able to interpret the  $d_j$ 's as digits in the base  $n+1$  expansion of the second quantity. We now find the required value for  $k$ . Generalising (11), the measure of all words of length at most  $k \in \mathbb{N}$  over the sub-alphabet  $\Sigma_m$  is (always computed w.r.t.  $\Sigma_n$ )

$$\mu_{\text{coin}}(\Sigma_m^{\leq k}) = \sum_{i=0}^k \frac{m^i}{(n+1)^{i+1}} = \frac{1}{n-m+1} \left( 1 - \left( \frac{m}{n+1} \right)^{k+1} \right). \quad (13)$$

Since  $c < \frac{1}{n-m+1} = \lim_{k \rightarrow \infty} \mu_{\text{coin}}(\Sigma_m^{\leq k})$ , there is some  $k$  s.t.  $c < \mu_{\text{coin}}(\Sigma_m^{\leq k})$ . Let  $k$  be such a minimal number, i.e.,  $\mu_{\text{coin}}(\Sigma_m^{\leq k-1}) \leq c < \mu_{\text{coin}}(\Sigma_m^{\leq k})$ . Consequently, for this choice of  $k$  we have  $c_0 = m^0, c_1 = m^1, \dots, c_{k-1} = m^{k-1}$ , and  $0 \leq c_k < m^k$  where

$$\mu_{\Sigma_n}(\Sigma_m^{\leq k-1}) + \frac{c_k}{(n+1)^{k+1}} \leq c < \mu_{\Sigma_n}(\Sigma_m^{\leq k}) + \frac{c_k + 1}{(n+1)^{k+1}}.$$

For complexity considerations, we note that  $k$  satisfies  $c < \mu_{\text{coin}}(\Sigma_m^{\leq k})$ , and thus  $k > \frac{\log(1-(n-m+1)c)}{\log m - \log(n+1)} - 1 = \frac{-\log(1-(n-m+1)c)}{\log(n+1) - \log m}$ . The minimal denominator is achieved by  $m = n$ , and the maximal numerator by  $m = 1$ . Performing this substitution, we can see that one can take  $k > \frac{-\log(1-nc)}{\log(n+1) - \log n}$ . Since  $-\log(1-nc) = O(\log q)$  and  $\log(n+1) - \log n = \log \frac{n+1}{n} = \log(1 + \frac{1}{n}) = O(\frac{1}{n})$ , we obtain

$$k = O(n \log q). \quad (14)$$

This choice of  $k$  also guarantees  $n \leq m^k$ . Let  $d := c - \left( \mu_{\Sigma_n}(\Sigma_m^{\leq k-1}) + \frac{c_k}{(n+1)^{k+1}} \right)$  be the remaining part of  $c$  after all terms up to  $k$  have been considered: Thus,  $d$  is small in the sense that  $0 \leq d < \frac{1}{(n+1)^{k+1}}$ . Since  $\sum_{j=1}^{\infty} \frac{n}{(n+1)^{j+1}} = \frac{1}{n+1}$ , we can write  $d$  in base  $n+1$  as required by (12):

$$d = \frac{1}{(n+1)^k} \sum_{j=1}^{\infty} \frac{d_j}{(n+1)^{j+1}}, \quad \text{with } 0 \leq d_j \leq n \leq m^k.$$

The coefficients are small numbers  $d_j$  at most  $n$ , which can thus be interpreted as digits in base  $n+1$ . Since  $d$  is a rational number, the sequence  $d_1, d_2, \dots$  is ultimately periodic [17], i.e., there exists a threshold  $j_1 \in \mathbb{N}$  and a period  $l \in \mathbb{N}$  s.t., for every  $j \geq j_1$ ,  $d_{j+l} = d_j$ . Let  $\gamma_1 = d_{j_1}, \gamma_2 = d_{j_1+1}, \dots, \gamma_l = d_{j_1+l-1}$  be the coefficients comprising the period. Consequently,

$$\begin{aligned} d &= \frac{1}{(n+1)^k} \left( \sum_{j=1}^{j_1-1} \frac{d_j}{(n+1)^{j+1}} + \sum_{s=0}^{\infty} \left( \frac{\gamma_1}{(n+1)^{j_1+s l+1}} + \dots + \frac{\gamma_l}{(n+1)^{j_1+s l+l}} \right) \right) \\ &= \frac{1}{(n+1)^k} \left( \sum_{j=1}^{j_1-1} \frac{d_j}{(n+1)^{j+1}} + \frac{1}{(n+1)^{j_1-1}} \sum_{s=0}^{\infty} \frac{\gamma}{(n+1)^{(s+1)l+1}} \right) = \end{aligned} \quad (15)$$

$$= \frac{1}{(n+1)^k} \left( \sum_{j=1}^{j_1-1} \frac{d_j}{(n+1)^{j+1}} + \frac{\gamma}{(n+1)^{j_1}} \frac{1}{(n+1)^l - 1} \right), \quad (16)$$

where  $\gamma := \gamma_1(n+1)^{l-1} + \dots + \gamma_l(n+1)^0$  is the number of words of length  $k+j_1-1+(s+1)l$ , for every  $s \in \mathbb{N}$ .



We are now ready to build the regular expression  $e$ . We begin with a basic building block. For a given length  $k$  and a cardinality  $0 \leq h \leq m^k$  written in base  $m$  as  $h = \sum_{i=0}^k h_i \cdot m^i$  ( $0 \leq h_i < m$ ), consider the regular expression  $e_{h,k}$  s.t.  $e_{h,k} = \Sigma_m^k$  if  $h = m^k$ , and otherwise:

$$e_{h,k} = a_m^{k-1} \cdot (a_1 + \dots + a_{h_0}) \cdot \Sigma_m^0 + a_m^{k-2} \cdot (a_1 + \dots + a_{h_1}) \cdot \Sigma_m^1 + \dots + a_m^0 \cdot (a_1 + \dots + a_{h_{k-1}}) \cdot \Sigma_m^{k-1}.$$

*Claim 15.* The expression  $e_{h,k}$  is unambiguous and  $L(e_{h,k})$  contains precisely  $h$  words of length  $k$ . Consequently,  $\mu_{\text{coin}}(L(e_{h,k})) = \frac{h}{(n+1)^{k+1}}$ .

*Proof (of the claim).* All words recognised by  $e_{h,k}$  have length  $k$ . The  $i$ -th block  $a_m^{k-i-1} \cdot (a_1 + \dots + a_{h_i}) \cdot \Sigma_m^i$  contains  $h_i \cdot m^i$  words since there are  $h_i$  distinct options to choose a letter from  $a_1 + \dots + a_{h_i}$  and  $m^i$  distinct options to choose a word from  $\Sigma_m^i$ . The languages of different blocks are disjoint, since the prefix  $a_m^{k-i-1}$  uniquely determines the block (since  $h_i < m$  and thus  $a_1 + \dots + a_{h_i}$  does not contain  $a_m$ ). Similarly, the two concatenations in a block are unambiguous, and thus  $e_{h,k}$  is unambiguous.  $\square$

The sought expression of measure  $c$  is

$$e = \Sigma_m^{\leq k-1} + e_{c_k,k} + e_{d_1,k+1} + \dots + e_{d_{j_1-1},k+j_1-1} + e_{\gamma,k+j_1-1+l} \cdot e_{1,l}^*. \quad (17)$$

*Claim 16.* The expression  $e$  is well defined provided  $k \geq l \cdot \frac{\log(n+1)}{\log m} - 1 + l$ .

*Proof (of the claim).* The  $e_{d_j,k+j}$ 's are well defined since  $d_j \leq n \leq m^k \leq m^{k+j}$  and thus there are  $d_j$  words of length  $k+j$  over alphabet  $\Sigma_m$ . In order for  $e_{\gamma,k+j_1-1+l}$  to be well defined we need  $\gamma \leq m^{k+j_1-1+l}$ . Since  $\gamma \leq (n+1)^l$  and  $n \leq m^k$ , it suffice to require  $(n+1)^l \leq m^{k-1+l}$ . The latter condition follows from the assumption.  $\square$

*Claim 17.* The expression  $e$  is unambiguous and  $\mu_{\text{coin}}(L(e)) = c$ .

*Proof (of the claim).* Unambiguity follows from the fact that 1) all disjuncts of  $e$  recognise disjoint languages, since  $\Sigma_m^{\leq k-1}$  recognises only words of length  $\leq k-1$ ,  $e_{c_k,k}$  only words of length  $k$ ,  $e_{d_1,k+1}$  only words of length  $k+1$ ,  $\dots$ ,  $e_{d_{j_1-1},k+j_1-1}$  only words of length  $k+j_1-1$ , and  $e_{\gamma,k+j_1-1+l} \cdot e_{1,l}^*$  only words of length  $\geq k+j_1-1+l$ , and 2) all disjuncts  $e_{c_k,k}, e_{d_1,k+1}, \dots, e_{d_{j_1-1},k+j_1-1}$  are unambiguous by Claim 15, and  $e_{\gamma,k+j_1-1+l} \cdot e_{1,l}^*$  is unambiguous since  $e_{1,l}^*$  contains only one word of length  $l$ . In general  $\mu_{\text{coin}}(L^k) = (n+1)^{k-1} \mu_{\text{coin}}(L)$  (by (7), when such power is unambiguous) and thus  $\mu_{\text{coin}}(L^*) = \mu_{\text{coin}}(L^0 \cup L^1 \cup \dots) = \mu_{\text{coin}}(L^0) + \mu_{\text{coin}}(L^1) + \dots = \frac{1}{n+1}(((n+1)\mu_{\text{coin}}(L))^0 + ((n+1)\mu_{\text{coin}}(L))^1 + \dots) = \frac{1}{n+1} \cdot \frac{1}{1-(n+1)\mu_{\text{coin}}(L)}$  when the Kleene iteration is unambiguous. Since  $\mu_{\text{coin}}(L(e_{1,l})) = \frac{1}{(n+1)^{l+1}}$ , we obtain  $\mu_{\text{coin}}(L(e_{1,l}^*)) = \frac{1}{n+1} \cdot \frac{1}{1-(n+1)\frac{1}{(n+1)^{l+1}}} = \frac{(n+1)^{l-1}}{(n+1)^l - 1}$ . Consequently,

$$\begin{aligned} \mu_{\text{coin}}(L(e_{\gamma,k+j_1-1+l} \cdot e_{1,l}^*)) &= (n+1) \cdot \mu_{\text{coin}}(L(e_{\gamma,k+j_1-1+l})) \cdot \mu_{\text{coin}}(L(e_{1,l}^*)) = \\ &= (n+1) \cdot \frac{\gamma}{(n+1)^{k+j_1+l}} \cdot \frac{(n+1)^{l-1}}{(n+1)^l - 1} = \\ &= \frac{\gamma}{(n+1)^{k+j_1}} \cdot \frac{1}{(n+1)^l - 1}. \end{aligned}$$

By (16),  $\mu_{\text{coin}}(L(e)) = \mu_{\text{coin}}(\Sigma_m^{\leq k-1}) + \mu_{\text{coin}}(L(e_{c_k,k})) + \mu_{\text{coin}}(L(e_{d_1,k+1})) + \dots + \mu_{\text{coin}}(L(e_{d_{j_1-1},k+j_1-1})) + \mu_{\text{coin}}(L(e_{\gamma,k+j_1-1+l} \cdot e_{1,l}^*)) = c$ , as required.  $\square$

*Claim 18.* The expression  $e$  has size  $O((n \log q + j_1 + l + 1)^3)$ .

*Proof (of the claim).* The size of  $e_{h,k}$  is  $|e_{h,k}| = O(k \cdot (m + k)) = O(k \cdot (n + k))$ , since  $m \leq n$ . Thus the size of  $e$  is  $|e| \leq O((k + n + j_1 + l + 1)^3)$ . By (14),  $|e| \leq O((n \log q + j_1 + l + 1)^3)$ , as required.  $\square$

If we further assume that  $q \mid (n + 1)^\ell$  for some  $\ell \in \mathbb{N}$ , then we argue that we can take  $e$  to have size polynomial in  $\log q$ ,  $n$ , and  $\ell$ . By (14) and the additional assumption on  $q$ ,

$$k = O(n\ell \log(n + 1)). \quad (18)$$

We can write  $d = \frac{r}{s}$  with  $r, s \in \mathbb{N}$  and  $r \leq s$  relatively prime, where  $s \mid (n + 1)^{k+1}$ . If we decompose the base  $n + 1$  in prime factors as  $n + 1 = p_1^{z_1} p_2^{z_2} \cdots p_m^{z_m}$  with  $z_1, \dots, z_m \in \mathbb{N}$ , then  $s$  is of the form  $s = p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m}$  with  $t_i \leq (k + 1)z_i$ . By [17, Theorem 136]<sup>10</sup>,  $d$  can be written in base  $(n + 1)$  with a *finite* expansion of length  $j_1 = \max \left\{ \frac{t_1}{z_1}, \dots, \frac{t_m}{z_m} \right\} = O(k + 1)$ . In particular, the period has zero length  $l = 0$ . By (18),  $j_1 = O(n\ell \log(n + 1))$ . By applying Claim 18 to this case, we obtain a regular expression  $e$  of size  $O((n \log q + j_1 + l + 1)^3) \leq O((n \log q + n\ell \log(n + 1) + 1)^3)$  which is polynomial in  $\log q = O(\ell \log n)$ ,  $\ell$ , and  $n$ , as required.  $\square$

---

<sup>10</sup>Strictly speaking, [17, Theorem 136] requires that the base  $n + 1$  be a product of primes. However, as explained in the paragraph preceding the statement of the theorem, no difficulty arises if  $n + 1$  has square divisors; c.f. also the first footnote on page 145.