Theoretical Computer Science 613 (2016) 1–27

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Pseudorandom generators against advised context-free languages

Tomoyuki Yamakami [1]

Department of Information Science, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan

ARTICLE INFO

Article history:
Received 30 August 2012
Received in revised form 18 August 2015
Accepted 21 October 2015
Available online 23 October 2015
Communicated by G. Ausiello

Keywords:
Context-free language
Advice
Pseudorandom generator
Pushdown automaton
Pseudorandom language
Swapping property

ABSTRACT

Pseudorandomness has played a central role in modern cryptography, finding theoretical and practical applications to various fields of computer science. A function that generates pseudorandom strings from shorter but truly random seeds is known as a pseudorandom generator. Our generators are designed to fool languages (or equivalently, Boolean-valued functions). In particular, our generator fools advised context-free languages, namely, context-free languages assisted by external information known as advice, and moreover our generator is made almost one-to-one, stretching n-bit seeds to $n+1$ bits. We explicitly construct such a pseudorandom generator, which is computed by a deterministic Turing machine using logarithmic space and also belongs to CFLMV(2)/n—a functional extension of the 2-conjunctive closure of CFL with the help of appropriate deterministic advice. In contrast, we show that there is no almost one-to-one pseudorandom generator against context-free languages if we demand that it should be computed by a nondeterministic pushdown automaton equipped with a write-only output tape. Our generator naturally extends known pseudorandom generators against advised regular languages. Our proof of the CFL/n-pseudorandomness of the generator is quite elementary and, in particular, one part of the proof utilizes a special feature of the behaviors of nondeterministic pushdown automata, called a swapping property, which is interesting in its own right, generalizing the swapping lemma for context-free languages.

1. Our challenges and contributions

Regular and context-free languages are unarguably considered as the most fundamental notions in formal language and automata theory. Those special languages have been extensively studied since the 1950s and a large volume of work has been devoted to unearthing quite intriguing features of their behaviors and powers. Underlying finite(-state) automata that recognize those languages can be further assisted by external information, called *(deterministic) advice*, which is given besides input instances in order to enhance the computational power of the automata. One-way deterministic finite automata (or dfa's, in short) and their associated regular languages that are appropriately supplemented by advice strings of size n in parallel to input instances of length n naturally form an advised language family, which is dubbed as REG/n in [11] and further studied in [12–15]. In a similar fashion, one-way nondeterministic pushdown automata (or npda's) and their corresponding context-free languages with appropriate advice naturally induce another advised language family CFL/n [12,13]. The notion of advice endows the underlying machines with a non-uniform nature of computation; for instance, advised

E-mail address: TomoyukiYamakami@gmail.com.
[1] Tel.: +81 776 27 8587; fax: +81 776 27 8751.

http://dx.doi.org/10.1016/j.tcs.2015.10.026
0304-3975/

regular languages are recognized by non-uniform series of length-dependent dfa's and also characterized in [13] in terms of length-dependent *non-regularity*. Beyond the above-mentioned advice, recent studies further dealt with its important variants: *randomized advice* [13,15] and *quantum advice* [15].

In an analysis of the behaviors of languages, their corresponding *functions* defined on finite strings over certain alphabets have sometimes played a supporting role. Types of those functions vary considerably from an early example of functions computed by Mealy machines [7] and Moore machines [8] to more recent examples of acceptance probability functions (e.g., [6]) and counting functions [11] and to an example of functions computed by npda's equipped with write-only output tapes [14,16,17]. Nonetheless, a field of such functions has been largely unexplored in formal language and automata theory, and our goal to the full understandings of structural properties of those functions still awaits to be fulfilled. Our particular interest in this paper rests in one of those structural properties, known as *pseudorandomness* against advised language families [14], and its theoretical application to *pseudorandom generators*.

The notion of pseudorandom generator dates back to early 1980s and it has since then become a key ingredient in modern cryptography and also it has made a significant impact on the development of computational complexity theory. An early generator that Blum and Micali [2] proposed is designed to produce a sequence in which any reasonably powerful adversary hardly predicts the sequence's next bit. Yao's [18] generator, on the contrary, produces a sequence that no adversary distinguishes from a uniformly random sequence with a small margin of error. Those two formulations—unpredictability and indistinguishability—are essentially equivalent and the generators that are formulated accordingly are now known as *pseudorandom generators*. Since their introduction, the pseudorandom generators have played key roles in constructing various secure protocols as an important cryptographic primitive. However, the existence of a (polynomial-time computable) pseudorandom generator is still unknown unless we impose certain unproven complexity-theoretical assumptions, such as $\mathrm{NP} \not\subseteq \mathrm{BPP}$ or the existence of polynomial-time one-way functions (see, e.g., [4]).

Within a framework of formal language and automata theory, a recent study [14] was focused on a specific type of pseudorandom generator, whose adversaries are represented in a form of languages (or equivalently, $\{0,1\}$-valued functions), compared to standard "probabilistic algorithms." Such a generator also appears when the generator's adversaries are "Boolean circuits" that produce one-bit outputs. Intuitively, given an arbitrary alphabet Σ, a (single-valued total) function $G : \Sigma^* \to \Sigma^*$, which stretches n-symbol seeds to $s(n)$-symbol strings, is said to *fool* language A over Σ if the characteristic function[2] χ_A of A cannot distinguish between the output distribution of $\{G(x)\}_{x\in\Sigma^n}$ and a truly random distribution of $\{y\}_{y\in\Sigma^{s(n)}}$ with *non-negligible* success probability. We call G a *pseudorandom generator* against language family $\mathcal{C}$ if G fools every language A over Σ in $\mathcal{C}$. As our limited adversaries, we intend to take regular languages and context-free languages assisted further by advice. An immediate advantage of dealing with such weak adversaries is that we can actually construct corresponding pseudorandom generators *without any unproven assumption*.

A fundamental question that naturally arises from the above definition is whether there exists an *efficiently computable* pseudorandom generator against a "low-complexity" family of languages. In an early study [14], a single-valued total function computed by an appropriate npda equipped with a write-only output tape (where the set of those functions is briefly denoted $\mathrm{CFLSV_t}$, an automaton-analogue of $\mathrm{NPSV_t}$ [3]) was proven to be a pseudorandom generator against REG/n. This pseudorandom generator actually stretches truly random seeds of n bits to strings of $n+1$ bits and, moreover, it is made one-to-one for all but a negligible fraction of their domain instances (called *almost one-to-one*, or almost 1-1). The existence of such a restricted pseudorandom generator is closely linked to the REG/n-*pseudorandomness* of languages in CFL (context-free language family) [14]. Regarding the computational complexity of the generator, one may wonder if such a generator can be computed much more efficiently. Unfortunately, as shown in [14], no pseudorandom generator against REG (regular language family) can be computed by single-tape linear-time Turing machines as long as the generator is almost 1-1 and stretches n-bit seeds to $(n+1)$-bit strings. Notice that almost one-to-oneness and a small stretch factor are a key to establish those results, because any generator satisfying those properties become pseudorandom if and only if its range (viewed as a language) is pseudorandom [14] (see also Lemma 3.5).

A critical question left unsolved in [14] is whether an efficient pseudorandom generator of small stretch factor actually exists against CFL/n. A simple and natural way to construct such a specific generator is to apply a so-called *diagonalization technique*: first enumerate all advised languages in CFL/n and then diagonalize them one by one to determine an outcome of the generator. Such a technique gives a generator that can be computed deterministically in exponential time. For each language in CFL/n, since it can be expressed as a family of polynomial-size Boolean circuits, a design-theoretic method of Nisan and Wigderson [9] can be used to construct a pseudorandom generator against those polynomial-size circuits, however, at a cost of super-polynomial running time. With a much harder effort in this paper, we intend to give an explicit construction of a pseudorandom generator against CFL/n whose computational complexity is simultaneously in FL (logarithmic-space function class) and in $\mathrm{CFLMV}(2)/n$—a functional analogue of $\mathrm{CFL}(2)/n$ (which coincides with the *2-conjunctive closure* of CFL/n by Claim 2) as well as a natural extension of CFLMV (multiple-valued partial CFL-function class) given in [14].

First Main Theorem. *A pseudorandom generator G against all advised context-free languages exists in* $\mathrm{FL} \cap \mathrm{CFLMV}(2)/n$. *More strongly, G can be made almost 1-1 with stretch factor $n+1$. (Theorem 3.2.)*

[2] The characteristic function χ_A of a language A is defined as $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ otherwise, for every input string x.