Very Sparse Leaf Languages[1]

Lance Fortnow[2]
Department of Computer Science
University of Chicago

Mitsunori Ogihara[3]
Department of Computer Science
University of Rochester

Technical Report 899
University of Rochester
Department of Computer

June, 2006

[1]This work is support in part by XEROX/NYSTAR Grant C040130 and NSF-EIA Grant 0205061.
[2]Department of Computer Science, University of Chicago, 1100 E. 58th St., Chicago, IL 60637. email: fortnow@cs.uchicago.edu.
[3]Box 270226, Department of Computer Science, University of Rochester, Rochester, NY 14627-0226. email: ogihara@cs.rochester.edu.

Abstract

Unger studied the balanced leaf languages defined via poly-logarithmically sparse leaf pattern sets. Unger shows that NP-complete sets are not polynomial-time many-one reducible to such balanced leaf language unless the polynomial hierarchy collapses to Θ_2^p and that Σ_2^p-complete sets are not polynomial-time bounded-truth-table reducible (respectively, polynomial-time Turing reducible) to any such balanced leaf language unless the polynomial hierarchy collapses to Δ_2^p (respectively, Σ_4^p).

This paper studies the complexity of the class of such balanced leaf languages, which will be denoted by VSLL. In particular, the following tight upper and lower bounds of VSLL are shown:

1. coNP $\subseteq$ VSLL $\subseteq$ coNP/poly (the former inclusion is already shown by Unger).
2. coNP/1 $\not\subseteq$ VSLL unless PH $= \Theta_2^p$.
3. For all constant $c > 0$, VSLL $\not\subseteq$ coNP/n^c.
4. P/$(\log\log(n) + O(1)) \subseteq$ VSLL.
5. For all $h(n) = \log\log(n) + \omega(1)$, P/$h \not\subseteq$ VSLL.

1 Introduction

Bovet, Crescenzi, and Silvestri [2] introduced the concept of leaf languages — the languages defined in terms of the pattern appearing at the leaf-level a polynomial-time nondeterministic Turing machine that outputs a symbol along each computation path. The concept of using outputs of nondeterministic Turing machines for defining complexity classes appears earlier, in a paper by Goldschlager and Parberry [7], but it is in this work of Bovet, Crescenzi, and Silvestri that the concept was formulated and fully explored. Given a polynomial-time nondeterministic Turing machine M that accepts all inputs on all computations and that outputs a symbol from an alphabet Γ, leafstring_M is the function that maps each input of M to the word produced by reading the output symbols in the computation tree of M on input x according to the dictionary order over the computation paths of M on x. Given a language K over the alphabet Γ, the leaf language with respect to M and K, is the set of all inputs x such that $\text{leafstring}_M(x) \in K$.

Bovet, Crescenzi, and Silvestri showed that many well-known complexity classes can be characterized this way using some simple leaf languages, including NP, coNP, and PP. The leaf languages offer a rich theory of complexity classes and are very strongly connected with branching programs [1] and bottleneck Turing machines [5].

Recently, Unger [14] studied the leaf languages defined with respect to poly-logarithmically sparse leaf pattern sets. We call this complexity class VSLL (Very Sparse Leaf Languages) as Unger did not give a name to this class. Unger showed that if SAT is polynomial-time many-one reducible to a language in VSLL then $\text{PH} = \Theta_2^p$ and that if a Σ_2^p-complete set is polynomial-time bounded-truth-table reducible to a language in VSLL then $\text{PH} = \Delta_2^p$.

The leaf languages defined with respect to such very sparse leaf pattern sets are related to polynomially sparse sets. Although he did not make it an explicit claim, Unger's proof of the former result essentially uses the fact that VSLL is included coNP/poly. Here coNP/poly is the "nonuniform-coNP" [10], in the sense that each language in the class is decidable in coNP with the aid of an "advice" string that is polynomially long and that is dependent solely on the input length. As we will show explicitly in this paper, for each language $L \in \text{VSLL}$, the advice function that puts L in coNP/poly maps each input length to a polynomial number of members of L and the set of the words appearing in advice strings is a sparse subset of L.

We thus naturally question the connection between VSLL and sparse sets. Indeed, the proof of the former of the two results of Unger is reminiscent of the technique that uses the census function in NP exploited in the result of Kadin [9] that showed the existence of sparse Turing-complete sets for NP collapses the polynomial hierarchy to Θ_2^p; and the proof of the latter directly uses the left-set technique of Ogihara and Watanabe [12]. However, it is unclear whether the assumption $\text{NP} \subseteq \text{VSLL}$ for the first result implies the existence of sparse Turing-complete sets for NP. This observation motivates us to explore the complexity of VSLL.

It is easy to see that $\text{coNP} \subseteq \text{VSLL}$, via the poly-logarithmically sparse leaf pattern language $\{0^{2^n} \mid n \geq 0\}$ (see [14]). We show that this is in fact a tight lower bound of VSLL. The class coNP/1, i.e., coNP with a single-bit of advice, is not included in VSLL unless NP is already included in VSLL, which, according to the result of Unger, collapses the polynomial hierarchy to Θ_2^p. In fact, we show that for any recursive complexity class $\mathcal{C}$, if $\mathcal{C}/1 \subseteq \text{VSLL}$ then $\mathcal{C} \subseteq \text{P/poly}$. We also show a "provably" tight lower bound for VSLL with respect to polynomial-time decision with advice. While $\text{P}/\log\log \subseteq \text{VSLL}$, $\text{P}/(\log\log(n) + \omega(1)) \not\subseteq \text{VSLL}$.

Along with the above lower bound results, we show two tight upper bounds. First, if $\text{VSLL} \subseteq \text{P/poly}$ then PH collapses to Σ_2^p, more precisely, to class S_2^p [6, 13]. Second, for an arbitrary constant $c > 0$, $\text{VSLL} \not\subseteq \text{coNP}/n^c$ holds with no assumption.

2 Preliminaries

Let Σ be the alphabet $\{0,1\}$. As usual, Σ^* denotes the set of all words over Σ and $\mathcal{N}$ denotes the set of all natural numbers. For each $n \in \mathcal{N}$, Σ^n denotes the set of all words over Σ having length n. For each $n \in \mathcal{N}$ and for each language L, $L^{=n}$ denotes $\{x \mid x \in L \wedge |x| = n\}$ and $L^{\leq n}$ denotes $\{x \mid x \in L \wedge |x| \leq n\}$. For a language A and for a natural number n, $\mathrm{census}_A(n) = \parallel L^{\leq n} \parallel$. The function census_A is called the *census function* of A.

We assume the reader's familiarity in basic complexity classes and reducibility notions, including classes P, NP, coNP, PH, $\{\Sigma_k^p, \Pi_k^p, \Delta_k^p, \Theta_k^p\}_{k\geq 0}$, L, and NL and reducibility notions $\leq_m^p$, $\leq_T^p$, and $\leq_{btt}^p$.

Let M be a polynomial time-bounded nondeterministic Turing machine transducer such that M accepts on all inputs and along all computation paths and such that on each computation path M outputs a symbol from an alphabet Γ. We call such a machine M an NP *character transducer*. For each input x and for each computation path π on M on x, let $M_\pi(x)$ be the output of M on x along path π. For each input x, define

$$\mathrm{leafstring}_M(x) = M_{\pi_1}(x) \cdots M_{\pi_r}(x),$$

where $\pi_1, \ldots, \pi_r$ is the enumeration in the dictionary order of all computation paths of M on input x. By abuse of notation, for a set of inputs S, we define

$$\mathrm{leafstring}_M(S) = \{\mathrm{leafstring}_M(x) \mid x \in S\}.$$

For $A \subseteq \Gamma^*$, let $\mathrm{Leaf}_M^{\mathrm{P}}(A)$ be the language consisting of all inputs x such that $\mathrm{leafstring}_M(x) \in A$.

Definition 2.1 *For a language $A \subseteq \Gamma^*$, the language class $\mathrm{Leaf}^{\mathrm{P}}(A)$ is the set of all languages $\mathrm{Leaf}_M^{\mathrm{P}}(A)$, where M is an* NP *character transducer.*

For a language class $\mathcal{C}$, the language class $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{C})$ is $\{\mathrm{Leaf}^{\mathrm{P}}(A) \mid A \in \mathcal{C}\}$.

We pay a special attention to the leaf language defined via a Turing machine whose computation tree is *balanced* in the following sense: there exists a polynomial p such that for all inputs x, the computation tree of the Turing machine on input x is the full complete binary tree of height $p(|x|)$. We call such a machine a *balanced-tree* NP *character transducer*. We will use $\mathrm{BalancedLeaf}^{\mathrm{P}}(A)$ and $\mathrm{BalancedLeaf}^{\mathrm{P}}(\mathcal{C})$, respectively, to denote $\mathrm{Leaf}^{\mathrm{P}}(A)$ and $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{C})$ in which the NP character transducers are restricted to be balanced-tree character transducers.

2.1 Sparse sets

A language S is *sparse* if there exists a polynomial p such that for all n $\mathrm{census}_S(n) \leq p(n)$. By SPARSE we denote the set of all sparse sets.

A language S is *poly-logarithmically sparse* (or *polylog sparse*, for short) if there exist positive constants c and d such that for all n $\mathrm{census}_S(n) \leq c(\log n)^d$. By PLOGSPARSE we denote the set of all poly-logarithmically sparse sets.

2.2 Advice classes

Let f be a function from $\mathcal{N}$ to itself and let $\mathcal{C}$ be a complexity class. Then $\mathcal{C}/f$ is the set of all languages L for which there exist $A \in \mathcal{C}$ and a function h such that

- for all x, $|h(x)| \leq f(|x|)$, and

- for all x, $x \in L$ if and only if $\langle x, h(0^{|x|}) \rangle \in A$.

Complexity classes defined this way are called "advice" classes and the function h is called an "advice" function. By poly we denote the set of all polynomials. It is a well-known fact that $\mathrm{P/poly} = \mathrm{P}^{\mathrm{SPARSE}}$ [10].

2.3 Very sparse leaf languages

We define VSLL (Very Sparse Leaf Languages) to be the set of all leaf languages whose leaf pattern sets are poly-logarithmically sparse.

Definition 2.2 $\mathrm{VSLL} = \mathrm{BalancedLeaf}^{\mathrm{P}}(\mathrm{PLOGSPARSE})$.

Unger [14] showed the following:

Theorem 2.3 (Unger)

1. *If* $\mathrm{NP} \leq_m^p \mathrm{VSLL}$ *then* $\mathrm{PH} = \Theta_2^p$.
2. *If* $\Sigma_2^p \leq_{btt}^p \mathrm{VSLL}$ *then* $\mathrm{PH} = \Delta_2^p$.
3. *If* $\Sigma_2^p \leq_T^p \mathrm{VSLL}$ *then* $\mathrm{PH} = \Sigma_4^p$.

3 Fundamental Properties of VSLL

We show below a number of properties of the class VSLL.

First, the class VSLL is closed under $\leq_m^p$-reductions; that is, if $A \leq_m^p B$ and $B \in \mathrm{VSLL}$ then $A \in \mathrm{VSLL}$.

Proposition 3.1 VSLL *is closed under* $\leq_m^p$*-reductions.*

Proof Let $A \leq_m^p B$ and $B \in \mathrm{VSLL}$. Let f be a polynomial-time computable function that maps A to B. Let $B \in \mathrm{VSLL}$ via a machine M and a polylog sparse $S \subseteq \Gamma^*$. Let a and b be two symbols not in Γ and let $\Upsilon = \Gamma \cup \{a, n\}$. Let p be a polynomial such that the computation tree of M on each input x has height $p(|x|)$. Let q be a polynomial such that for all x $|f(x)| \leq q(|x|)$. Define N to be a Turing machine that on input x behaves as follows:

- N nondeterministically chooses a bit b.
- If $b = 0$ then N executes the following:
  - N nondeterministically selects $w \in \Sigma^{p(q(n))}$.
  - If $w \in 0^*$ then N outputs a.
  - If $w \notin 0^*$ then N outputs b.
- If $b = 1$ then N executes the following:
  - N computes $f(x)$ and sets y to $f(x)$.
  - N sets k to $p(|y|)$ and l to $p(q(|x|)) - k$.
  - N nondeterministically selects $u \in \Sigma^l$.
  - If $u \notin 0^*$, then N nondeterministically selects $v \in \Sigma^k$ and outputs a.

– If $u \in 0^*$, then N nondeterministically simulates M on y and outputs the symbol that M outputs.

It is easy to see that the machine N can be made to run in polynomial time. For every x, N on input x uses exactly $p(q(|x|) + 1$ nondeterministic steps along each computation path. So, N is a balanced-tree nondeterministic Turing machine character transducer. For each x, we have

$$\text{leafstring}_N(x) = ab^{(2^{p(q(|x|))}-1)}\text{leafstring}_M(f(x))a^{(2^{p(q(|x|))}-2^{p(|f(x)|)})}.$$

Define T to be the set of all words of the form

$$ab^{(2^{p(m)}-1)}wa^{(2^{p(m)}-|w|)}$$

such that $m \in \mathcal{N}$ and $w \in S^{\leq 2^{p(m)}}$. Since $a, b \notin \Gamma$ and $S \subseteq \Gamma^*$, w does not contain a or b. So, N and T witness that $A \in \text{BalancedLeaf}^{\text{P}}(T)$.

It remains to show that $T \in$ PLOGSPARSE. For each $n \in \mathcal{N}$, if z is a member of $T^{\leq n}$ then $|z| = 2^r$ for some $r \in \mathcal{N}$ and z is of the form ab^iwa^j such that $1 + i = 2^{r-1}$ and $w \in S^{\leq n}$. So, for each $w \in S^{\leq n}$ there are at most $\log(n)$ words in $T^{\leq n}$ that contain w as a substring. So, $\text{census}_T(n) \leq \log(n)\text{census}_S(n)$. Thus, $T \in$ PLOGSPARSE. This proves the proposition. ■

Next, we present a property that is explicitly shown by Unger [14].

Proposition 3.2 (Unger) $L \in$ VSLL *if and only if there exist a sparse set* S *and a balanced-tree* NP *character transducer* M *such that for all* x*,*

$$x \in L \iff (\exists y \in S^{=|x|})[\text{leafstring}_M(x) = \text{leafstring}_M(y)].$$

Proof Suppose $L \in$ VSLL and this is witnessed by a balanced-tree NP character transducer M and $T \in$ PLOGSPARSE. Let p the polynomial such that $p(n)$ is the height of the computation tree of M on any input of length n. Let $c, d > 0$ be constants such that $\text{census}_T(n) \leq (\log(n))^d$.

Define $S = \{x \mid x \in L \wedge (\forall y \in \Sigma^{|x|}, y < x)[\text{leafstring}_M(x) \neq \text{leafstring}_M(y)]\}$. Let n be fixed. Let $x_1, \cdots, x_m$ be the enumeration of the words in $S^{=n}$ in the lexicographic increasing order. Then $\text{leafstring}_M(x_1), \ldots, \text{leafstring}_M(x_m)$ are pairwise distinct elements in $T^{=2^{p(n)}}$. By our supposition about c and d, $m \leq c(\log(2^{p(n)}))^d = c(p(n))^d$. This implies that S is a sparse set. Also, for each $z \in \Sigma^n$, the following properties hold:

- If $z \in S_n$, then for some i, $1 \leq i \leq m$, $\text{leafstring}_M(z) = \text{leafstring}_M(y_i)$ (otherwise, z would have been chosen to be in S_n).
- If $z \notin S_n$, then for all i, $1 \leq i \leq m$, $\text{leafstring}_M(z) \neq \text{leafstring}_M(y_i)$.

Thus, for all x,

$$x \in L \iff (\exists y \in S^{=|x|})[\text{leafstring}_M(x) = \text{leafstring}_M(y)]$$

as desired.

On the other hand, suppose there exist a sparse S and a balanced-tree NP character transducer M such that for all x,

$$x \in L \iff (\exists y \in S^{=|x|})[\text{leafstring}_M(x) = \text{leafstring}_M(y)].$$

Let p be a polynomial such that for all x the computation tree of M on input x has height $p(|x|)$. Let q be a polynomial that bounds census_S. Define $T = \text{leafstring}_M(S)$. Then for all x, $x \in L \iff \text{leafstring}_M(x) \in T$. So, $L \in \text{BalancedLeaf}^{\text{P}}(T)$. For all n, $\text{census}_T(n) \leq \text{census}_S(\log(n)) \leq q(\log(n))$. So, $T \in$ PLOGSPARSE. This implies that $L \in$ VSLL. ■

A language A is $\leq^{\text{NP}}_{ctt}$-reducible to a language B if there exists a polynomial time-bounded nondeterministic Turing machine transducer N with the property that, on each input x,

- along each computation path of N on x, N produces a nonempty collection of words, $(y_1, \ldots, y_k)$; and
- $x \in A$ if and only if there exists a computation path π of N on x such that every word in the collection produced along π is a member of B.

Theorem 3.3 *If* $L \in$ VSLL *then* $L \in$ coNP/poly *with an "advice" function that is computable in polynomial time using an oracle that is* $\leq_{ctt}^{\mathrm{NP}}$*-reducible to* L*.*

Proof Suppose $L \in$ VSLL. By Proposition 3.2, there exist a sparse set S and a balanced-tree NP character transducer M such that for all x,

$$x \in L \iff (\exists y \in S^{=|x|})[\mathrm{leafstring}_M(x) = \mathrm{leafstring}_M(y)]$$

Here $S = \{x \mid x \in L \wedge (\forall y \in \Sigma^{|x|}, y < x)[\mathrm{leafstring}_M(x) \neq \mathrm{leafstring}_M(y)]\}$.

Suppose that for each n $S^{=n}$ is encoded as

$$e_n = \#x_1\#x_2\#\cdots\#x_m\#,$$

where $\{x_1, \cdots, x_m\} = S^{=n}$ and $x_1 < \cdots < x_m$. Define the prefix set of S, denoted by $\mathrm{pref}(S)$, as follows

$$\mathrm{pref}(S) = \{0^n w \mid (\exists y)[wy = s_n]\}.$$

This set is $\leq_{ctt}^{\mathrm{NP}}$-reducible to L. This is because $0^n\#w \in \mathrm{pref}(S)$ if and only if there exists some y such that wy is of the form $\#x_1\#\cdots\#x_m\#$ satisfying the following conditions:

- $x_1 < \cdots < x_m$ and $|x_1| = \cdots = |x_m| = n$,
- for all i, $1 \leq i \leq m$, $x_i \in L$, and
- for all i and j, $1 \leq i < j \leq m$, there exists some path π such that $M_\pi(x_i) \neq M_\pi(x_j)$

For each n, e_n can be computed using prefix search as follows:

- Set the initial value of the prefix E to $\#$.
- Execute the following:
  - If E ends with a $\#$, test whether $0^nE0 \in \mathrm{pref}(S)$ and whether $0^nE1 \in \mathrm{pref}(S)$. Then,
    - if the former holds then set E to $E0$;
    - if only the latter holds then set E to $E1$;
    - if neither hold then terminate the prefix search.
  - If E is of the form $u\#v$ such that $v \in \Sigma^*$ and $|v| = n$, then set E to $E\#$.
  - If E is of the form $u\#v$ such that $v \in \Sigma^*$ and $1 \leq |v| \leq n-1$, test whether $0^nE0 \in \mathrm{pref}(S)$ and whether $0^nE1 \in \mathrm{pref}(S)$. Then,
    - if the former holds then set E to $E0$;
    - otherwise, set E to $E1$.

In the above algorithm, when E is extendible both by appending a 0 and by appending a 1 the priority is given to appending a 0. So, s_n is correctly computed in polynomial time using $\mathrm{pref}(S)$ as the oracle.

Define

$$A = \{\langle u, \#x_1\#\cdots\#x_m\rangle \mid |u| = |x_1| = \cdots = |x_m| \wedge (\exists i, 1 \le i \le m)[\mathrm{leafstring}_M(u) = \mathrm{leafstring}_M(x_i)]\}.$$

Let p be a polynomial such that for each x the height of the computation tree of M on input x is $p(|x|)$. For every $\langle u, v\rangle$ such that $|u| = |v|$, $\mathrm{leafstring}_M(u) = \mathrm{leafstring}_M(v)$ if and only if for all computation paths π of length $p(|u|)$ it holds that $M_\pi(u) = M_\pi(v)$. This property certainly can be tested coNP, and so, $A \in$ coNP. Also, for all u, $u \in L \iff \langle u, e_{|u|}\rangle \in A$. Thus, $L \in$ coNP/poly. This proves the proposition. ■

The corollary follows immediately from the above theorem.

Corollary 3.4 *If* NP $\subseteq$ VSLL *then* NP $\subseteq$ coNP/poly *with an "advice" function computable in* Δ_2^p.

4 Lower Bounds of VSLL

Next we explore lower bounds of VSLL.

Proposition 4.1 SPARSE $\subseteq$ VSLL.

Proof Let S be an arbitrary sparse set. Define M to be a nondeterministic Turing machine that on input x behaves as follows: M nondeterministically selects $\pi \in \Sigma^{|x|}$, and then outputs 1 if $\pi = x$ and 0 otherwise. For each x, let r_x denote the lexicographic order of x in $\Sigma^{|x|}$. Then, for every x, $\mathrm{leafstring}_M(x) = 0^{r_x-1}10^{2^{|x|}-r_x}$. Define $K = \{0^{r_x-1}10^{2^{|x|}-r_x} \mid x \in S\}$. Then $K \in$ PLOGSPARSE. Thus, (M, K) witnesses that $S \in$ VSLL. ■

Proposition 4.2 coNP $\subseteq$ VSLL $\subseteq$ coNP/poly.

Proof The inclusion VSLL $\subseteq$ coNP/poly follows from Theorem 3.3. The inclusion coNP $\subseteq$ VSLL is shownby by Unger [14]. ■

It immediately follows from the above proposition that: if A is Turing-reducible to a language in VSLL then $A \in \Delta_2^p$/poly. In the case where A is Σ_2^p-complete, the set A being Turing-reducible to a language in VSLL implies that $\Sigma_2^p \subseteq \Delta_2^p$/poly, Then, by invoking Yap's Theorem [15] as presented in [4], we have the collapse $\mathrm{PH} = (\mathrm{S}_2^p)^{\mathrm{NP}}$, where S_2^p is the symmetric-Σ_2^p class [6, 13] and is known to reside between NP $\cup$ coNP and $\mathrm{ZPP}^{\mathrm{NP}}$ [3].

Proposition 4.3 *If* $\Sigma_2^p \subseteq \mathrm{P}^{\mathrm{VSLL}}$ *then* $\mathrm{PH} = (\mathrm{S}_2^p)^{\mathrm{NP}}$.

The above result improves part 3 of Theorem 2.3.

Theorem 4.4 *Suppose* coNP/1 $\subseteq$ VSLL. *Then* NP $\subseteq$ VSLL *and* NP $\subseteq$ P/poly.

Proof Suppose coNP/1 $\subseteq$ VSLL. For each infinite bit sequence $\vec{a} = (a_0, a_1, a_2, \cdots)$, define a language $Q[\vec{a}]$ as follows: For each x,

$$x \in Q[\vec{a}] \iff (a_{|x|} = 1 \vee x \notin \mathrm{SAT}).$$

Then, for every $\vec{a}$, $Q[\vec{a}] \in$ coNP/1, and thus, $Q[\vec{a}] \in$ VSLL by our supposition.

Take $\vec{a}$ to be the characteristic function of a bi-immune set. Then, no recursive function can compute infinitely many bits of $\vec{a}$. This bi-immune requirement is met by choosing $\vec{a}$ be Kolmogorov random infinite sequence [11]. Suppose $Q[\vec{a}] \in$ VSLL is witnessed by a balanced-tree NP character transducer M and $K \in$ PLOGSPARSE. Let p be a polynomial such that $p(n)$ is the height of the computation tree of M. Let $c, d > 0$ be constants such that K is $c(\log n)^d$ sparse. Define $q(n)$ to be the polynomial $c(\log(2^{p(n)}))^d = c(p(n))^d$.

For each n, let

$$\begin{aligned} W_1(n) &= \{\, \mathrm{leafstring}_M(x) \mid |x| = n \wedge x \in \mathrm{SAT} \,\} \text{ and} \\ W_0(n) &= \{\, \mathrm{leafstring}_M(x) \mid |x| = n \wedge x \notin \mathrm{SAT} \,\}. \end{aligned}$$

Note that, for all n,

1. if $a_n = 0$ then $W_1(n) \cap W_0(n) = \emptyset$ and $\| W_0(n) \| \leq q(n)$, and
2. if $a_n = 1$ then $\| W_0(n) \cup W_1(n) \| \leq q(n)$.

Assume that there are infinitely many n for which

$$\text{either } W_1(n) \cap W_0(n) \neq \emptyset \text{ or } \| W_0(n) \cup W_1(n) \| > q(n).$$

Let D be a deterministic Turing machine that, on input n,

- deterministically simulates M on all inputs of length n to check whether one of the above two conditions holds, and then
- outputs 1 if the former condition holds (since, by (1) in the above, it must be the case that $a_n = 1$), 0 if the latter condition holds (since, by (2) in the above, it must be the case that $a_n = 0$), and "?" otherwise.

This machine D correctly computes a_n for infinitely many n. This contradicts the assumption that $\vec{a}$ is Kolmogorov random. So, for all but finitely many n,

- $W_1(n) \cap W_0(n) = \emptyset$ and
- $\| W_0(n) \cup W_1(n) \| \leq q(n)$.

By making changes for a finite number of input lengths, M can be made to satisfy these conditions for all n. Define $K' = \cup_{n \geq 0} W_1(n)$. Then, K' is polylog sparse and (M, K') witnesses that SAT $\in$ VSLL.

To prove that the same assumption implies that NP is in P/poly, note that, for each n and for each pair (u, v) of distinct words in $W_0(n) \cup W_1(n)$, there is a bit position j at which u and v disagree. Such a position corresponds to a computation path of M on inputs of length n. Select such a position for each distinct pair from $W_0(n) \cup W_1(n)$. For each $u \in W_0(n) \cup W_1(n)$, by examining the output of $M(u)$ for the selected paths, $\mathrm{leafstring}_M(u)$ can be uniquely distinguished. The number of such pairs is bounded by $(q(n))^2/2$. Since each position corresponds to a computation path of M, the list of all these positions and how the words $W_0(n) \cup W_1(n)$ can be distinguished using the list can be described using a polynomially long string. This implies that SAT $\in$ P/poly. ■

The following corollary follows from the above theorem by applying Theorem 2.3.

Corollary 4.5 *If* coNP/1 $\subseteq$ VSLL *then* PH $= \Theta_2^p$.

Note that, in the proof of the part that shows NP $\subseteq$ P/poly in the above theorem, the fact that the decision for Q given $\vec{a}$ is in coNP is never used. So, we can generalize the proof and show the following:

Theorem 4.6 *For every reasonable class* $\mathcal{C}$ *of recursive sets,* $\mathcal{C}/1 \subseteq$ VSLL *implies* $\mathcal{C} \subseteq$ P/poly.

Corollary 4.7 *If* (NP $\cap$ coNP)/1 $\subseteq$ VSLL *then* NP $\cap$ coNP $\subseteq$ P/poly.

Corollary 4.8 *If* UP/1 $\subseteq$ VSLL *then* UP $\subseteq$ P/poly.

In the following, define $\log\log(0) = \log\log(1) = 0$.

Proposition 4.9 P/(log log $+O(1)$) $\subseteq$ VSLL.

Proof Let $L \in$ P/(log log $+O(1)$) via $A \in$ P and a function $f(n)$. We can assume that there exists an integer k such that, for all n, $|f(0^n)| \leq \lceil \log\log(n) + k \rceil$. Define $\mu(n) = \sum_{i=0}^{\lceil \log\log(n)+k \rceil} 2^i$. Then, for each n $\mu(n)$ is the number of possibilities for the "advice" string at length n and $\mu(n) \leq 2^{\log\log(n)+k+2}$. Let p be an arbitrary polynomial such that $2^{p(n)} \geq \mu(n)$. Let M be a nondeterministic Turing machine that on input x behaves as follows:

1. M nondeterministically guesses $y \in \Sigma^{p(|x|)}$ and computes its lexicographic order j of y in $\Sigma^{p(|x|)}$.
2. M outputs 0 if $j > \mu(n)$.
3. M computes the word w that has the lexicographic order j in Σ^*.
4. M outputs 1 if $\langle x, w \rangle \in A$ and 0 otherwise.

Clearly, M can be made to run in polynomial time. For each x, leafstring$_M(x)$ has length $2^{p(|x|)}$ and satisfies the following:

- for every j, $1 \leq j \leq \mu(|x|)$, the j-th bit of leafstring$_M(x)$ is 1 if (x is in L in the case where the j-th word is the "advice" string) and 0 otherwise, and
- for every j, $\mu(|x|) + 1 \leq j \leq 2^{p(|x|)}$, the j-th bit of leafstring$_M(x)$ is 0.

For each n, let j_n be the lexicographic order of the "advice" string for length n. Let K_n be the set of all words w having length $2^{p(n)}$ such that the j_n-th bit of w is 1 and such that for every j, $\mu(n) + 1 \leq j \leq 2^{p(n)}$, the j-th bit of w is 0. Define $K = \cup_{n \geq 0} K_n$.

We claim that the membership of L in VSLL is witnessed by M and K. For all x, $x \in L$ if and only if leafstring$_M(x) \in K_{|x|}$. To show that $K \in$ PLOGSPARSE, note that, for each n, $\mu(n) \leq 2^{\log\log(n)+k+2}$, $\| K_n \| \leq 2^{\mu(n)}$, and each word in K_n has length $2^{p(n)}$. Let $N = 2^{p(n)}$. Then, we have $n \leq p(n) = \log(N)$, so $\mu(n) \leq 2^{\log\log\log(N)+k+2} = 2^{k+2}(\log\log(N))$, and thus, $2^{\mu(n)} \leq 2^{2^{k+2}(\log\log(N))} = (\log N)^{2^{k+2}}$, which implies that $K \in$ PLOGSPARSE. ■

Theorem 4.10 *For all functions* $h(n) = \log\log(n) + \omega(1)$, P/$h \not\subseteq$ VSLL.

Proof Let $h(n)$ be a polynomial-time computable function such that $h(n) = \log\log(n)+\omega(1)$. Without loss of generality, we may assume that $h(n)$ is monotonically nondecreasing and $h(n) \leq \log(n)$ for all n.

For each infinite bit sequence $\vec{a} = (a_0, a_1, a_2, \cdots)$, define $Q[\vec{a}]$ as follows. Divide $\vec{a}$ into blocks of consecutive bits having length $h(0), h(1), h(2), \ldots$. More precisely, the first $h(0)$ bits of $\vec{a}$ (that is, $a_0, \ldots, a_{h(0)-1}$) become the first block, the next $h(1)$ bits (that is, $a_{h(1)}, \ldots, a_{h(0)+h(1)-1}$) become the second block, the next $h(2)$ bits (that is, $a_{h(1)+h(2)}, \ldots, a_{h(0)+h(1)+h(2)-1}$) become the third block, and so on. For each n, let B_n be the n-th block. For each n, we define $Q[\vec{a}]^{=n}$, the length-n portion of $Q[\vec{a}]$, as follows: Let $B_n = (d_1, \ldots, d_{h(n)})$ and let $D = 1 + \sum_{i=1}^{h(n)} d_i 2^{i-1}$. Then, $Q[\vec{a}]^{=n}$ is the set of all $x \in \Sigma^n$ whose D-th bit is a 1. For every $\vec{a}$, $Q[\vec{a}] \in \mathrm{P}/h$.

Assume, by way of contradiction, that $\mathrm{P}/h \subseteq \mathrm{VSLL}$. Take $\vec{a}$ to be a sequence with the following property:

(*) For every exponential-time deterministic Turing machine transducer M, if for all but finitely many n M on input 0^n outputs a word of length $h(n)$, then for infinitely many n, B_n is equal to the output of M on input 0^n.

A sequence $\vec{a}$ with this property can be constructed by simple diagonalization, but it suffices to take $\vec{a}$ to be a Kolmogorov random sequence relative to h.

Suppose that $Q[\vec{a}] \in \mathrm{VSLL}$ is witnessed by a balanced-tree NP character transducer M and a polylog sparse K. We then construct a deterministic algorithm that for all but finitely many n computes an $h(n)$-bit pattern ρ_n such that $B_n \neq \rho_n$.

By our assumption, $Q[\vec{a}] \in \mathrm{VSLL}$, so there exists a polynomial p such that for all n it holds that

$$\| \mathrm{leafstring}_M(Q[\vec{a}]^{=n}) \| \leq p(n).$$

Let n be sufficiently large. Let $B_n = (d_1, \ldots, d_{h(n)})$ and let $D = 1 + \sum_{i=1}^{h(n)} d_i 2^{i-1}$. The range of the integer D is 1 to $\Delta = 2^{h(n)}$. For each i, $1 \leq i \leq \Delta$, and for each $b \in \{0,1\}$, let

$$W_b(i) = \{\mathrm{leafstring}_M(x) \mid x \in \Sigma^n \wedge \text{ the } i\text{-th bit of } x \text{ is } b\}.$$

Suppose that there exists an i, $1 \leq i \leq \Delta$, such that $W_0(i) \cap W_1(i) \neq \emptyset$. Let i be such one. There exist $y, z \in \Sigma^n$ such that the i-th bit of y is 0, the i-th bit of z is 1, and $\mathrm{leafstring}_M(y) = \mathrm{leafstring}_M(z)$. This means that either both y and z are members of $Q[\vec{a}]^{=n}$ or both y and z are nonmembers of $Q[\vec{a}]^{=n}$. If D were i, then only z would be a member of $Q[\vec{a}]^{=n}$. So, it must be the case that $D \neq i$. Thus, if there exists an i, $1 \leq i \leq \Delta$, such that $W_0(i) \cap W_1(i) \neq \emptyset$, then it holds that $B_n \neq \beta_i$, where β_i is the word in $\Sigma^{h(n)}$ whose rank is i; that is, $\| \{w \in \Sigma^{h(n)} \mid w \leq \beta_i\} \| = i$. So, we take the smallest such i and set ρ_n to β_i.

We claim that there is always such an i. Assume, by way of contradiction, that for all i, $1 \leq i \leq \Delta$, $W_0(i) \cap W_1(i) = \emptyset$. Divide Σ^n according the first Δ bits. For each $u \in \Sigma^\Delta$, let $w(u) = \mathrm{leafstring}_M(u0^{n-\Delta})$. By our assumption, for all u, u' in Σ^Δ $u \neq u' \Rightarrow w(u) \neq w(u')$. So, regardless of what the value of D is, for exactly half of $u \in \Sigma^\Delta$, $u0^{n-\Delta} \in Q[\vec{a}]^{=n}$. This implies that

$$\| \mathrm{leafstring}_M(Q[\vec{a}]^{=n}) \| \geq 2^{\Delta-1}.$$

Then, by our supposition about $h(n)$, we have

$$\begin{aligned} \| \mathrm{leafstring}_M(Q[\vec{a}]^{=n}) \| &\geq 2^{\Delta-1} \\ &= 2^{2^{h(n)}-1} \\ &= 2^{2^{\log\log(n)+\omega(1)}-1} \\ &= 2^{\log(n)^{\omega(1)}-1}. \end{aligned}$$

The last quantity is super-polynomial, that is, a function that grows faster than any polynomial, in particular, faster than $p(n)$. This contradicts our supposition that $\| \text{leafstring}_M(Q[\vec{a}]^{=n})x) \| \leq p(n)$. Thus, the claim holds.

Define T to be a machine that, on input 0^n, executes the above algorithm and outputs ρ_n. It is not hard to see that M runs in time 2^{cn} for some constant c. For all but finitely many n, M on input 0^n outputs a word having length $h(n)$ not equal to B_n. This contradicts the property (*) of $\vec{a}$. Hence, $Q[\vec{a}] \notin \text{VSLL}$. This proves the theorem. ■

5 Upper Bounds of VSLL

Next we prove upper bounds of VSLL.

Since coNP $\subseteq$ VSLL and P/poly is closed under complementation, by the Karp–Lipton Theorem [10] (see [3, 4]), we have the following:

Proposition 5.1 VSLL $\subseteq$ P/poly *then* PH $= \text{S}_2^p$.

Also, we show that the upper bound coNP/poly $\supseteq$ VSLL is a tight one.

Theorem 5.2 *For any constant c,* VSLL $\not\subseteq$ REC/n^c*, where* REC *is the class of all recursive sets.*

Proof Let $c > 0$ be a fixed integer. Assume, by way of contradiction, that VSLL $\subseteq$ REC/n^c. By Proposition 4.1, SPARSE $\subseteq$ VSLL. We show that there is a sparse set not in REC/n^c.

Let $d = c + 2$. Let $\vec{a}$ be a Kolmogorov random string. We construct a S from $\vec{a}$ as follows.

Fix an integer ν such that $2^\nu \nu^d \leq 2^{2\nu}$. As in the proof of Theorem 4.10, we divide $\vec{a}$ into blocks. This time the first block has length μ^{d+1}, the second block has length $(\mu+1)^{d+1}$, the third block has length $(\mu+2)^{d+1}$, and so. In other words, for each $i \geq 1$, the i-th block consist of the $(\mu+i-1)^{d+1}$ bits of a positioned between $\sum_{j=1}^{i-1}(\mu+j)^{d+1}$ and $(\sum_{j=1}^{i}(\mu+j)^{d+1}) - 1$. Then, for each $i \geq 1$, divide the i-th block, which has length $(\mu+i-1)^{d+1}$, into $(\mu+i-1)$ words of length $(\mu+i-1)^d$.

The $(m-\nu+1)$-st block of $\vec{a}$, which consists of m^d words of length m, defines $S \cap \Sigma^{2m}$ as follows. Let $w_1, \ldots, w_{m^d}$ be the m^d words. Let $b_1, \ldots, b_{m^d}$ be the rank of these words in Σ^m; that is, for each i, $1 \leq i \leq m^d$, $b_i = \| \{y \in \Sigma^m \mid y \leq w_i\} \|$. Then, for each i, $1 \leq i \leq m^d$, define z_i to be the word in Σ^{2m} whose rank is $b_1 + \cdots + b_i$. For all i, $1 \leq i \leq m^d$, $1 \leq b_i \leq 2^m$. Since $2^m m^d \leq 2^{2m}$, $z_1, \ldots, z_{m^d}$ are well defined and $z_1 < \cdots < z_{m^d}$. Also, the $(m-\nu+1)$-st block of $\vec{a}$ can be recovered from $z_1, \cdots, z_{m^d}$.

For all n, $\text{census}_S(n) = \sum_{i:1 \leq i \leq n}(i/2)^d \leq n^{d+1}/2^d$, so S is sparse, and thus, $S \in$ VSLL.

Since VSLL $\subseteq$ REC/n^c by our assumption, we have a halting Turing machine M and an advice function h such that

- for all n, $|h(0^n)| \leq n^c$, and
- for all x, $x \in S$ if and only if M on input $\langle x, h(0^{|x|})\rangle$ accepts.

For each even n, let $H_n = h(1)\#h(2)\#\cdots\#h(n)$. Let N be a machine that, on input w of the form $u_1\#u_2\#\cdots\#u_k$ such that $k \geq 2\nu$, k is even and $u_1, \ldots, u_k$ contain no #, simulate M on all inputs of the form $\langle x, u_{|x|}\rangle$ such that $|x|$ is even and is greater than or equal to 2ν, and then recover the blocks of $\vec{a}$ corresponding to those x's accepted by M. Then, for all even $n \geq 2\nu$, N on input H_n produces the prefix of $\vec{a}$ having length $\sum_{i=\nu}^{n/2} i^d > n^d/2^d$. Since $d = c + 2$, for all but finitely many n, the length of the prefix thus produced is greater than or equal to $n^{c+1.5}$.

On the other hand, the combination of machine N and the advice H_n can be encoded as an input word to a universal Turing machine having length $|H_n| + \alpha$, where α is a constant not depending on n. For all but finitely many n, $|H_n| \leq \sum_{i:1 \leq i \leq n} n^c < n^{c+1}$.

This implies that for all but finitely many n, the prefix of $\vec{a}$ having length at least $n^{c+1.5}$ has Kolmogorov complexity at most n^{c+1}, which contradicts the assumption that $\vec{a}$ is Kolmogorov random. Thus, $S \notin \mathrm{REC}/n^c$. Hence, $\mathrm{VSLL} \not\subseteq \mathrm{REC}/n^c$. ■

6 Logarithmic Space-bounded Character Transducers

We extend the concept of very sparse leaf languages by considering logarithmic space-bounded nondeterministic computation in place of polynomial time-bounded nondeterministic computation. Logarithmic space-bounded nondeterministic machines have polynomially many IDs. We often shrink their computation trees by identifying nodes corresponding to the same ID to one, but for defining leafstring, we will view the computation trees as binary trees without collapsing nodes. Let $\mathrm{Leaf}^{\log}$ and $\mathrm{BalancedLeaf}^{\log}$ respectively denote the $\mathrm{Leaf}^{\mathrm{P}}$ and $\mathrm{BalancedLeaf}^{\mathrm{P}}$ thus defined. Let $\mathrm{VSLL}^{\log}$ be the VSLL defined in terms of $\mathrm{BalancedLeaf}^{\log}$.

For a logarithmic space-bounded balanced-tree nondeterministic Turing machine character transducer M, whether two inputs x and y produce the same $\mathrm{leafstring}_M$ can be tested as follows: Simulating M concurrently on the two inputs x and y choosing the same branch for both of them at each nondeterministic step. If such simulation arrives at an accepting state with two distinct output symbols for x and y, then $\mathrm{leafstring}_M(x) \neq \mathrm{leafstring}_M(y)$. If every simulation arrives at an accepting state with an identical output character for x and y, then $\mathrm{leafstring}_M(x) \neq \mathrm{leafstring}_M(y)$. So, the equality can be tested in coNL. Since NL is closed under complement, this implies that the test can be done in NL.

Theorem 6.1

1. $\mathrm{NL} \subseteq \mathrm{VSLL}^{\log} \subseteq \mathrm{NL/poly}$.
2. *If* $\mathrm{NL}/1 \subseteq \mathrm{VSLL}^{\log}$ *then* $\mathrm{NL} \subseteq \mathrm{L/poly}$.
3. *For all constants* $c > 0$, $\mathrm{VSLL}^{\log} \not\subseteq \mathrm{REC}/n^c$.
4. $\mathrm{VSLL}^{\log} \supseteq \mathrm{L}/(\log\log(n) + O(1))$.
5. $\mathrm{VSLL}^{\log} \not\supseteq \mathrm{L}/(\log\log(n) + \omega(1))$.

7 Conclusion

We would like to find a natural problem in VSLL. Graph Isomorphism is an obvious candidate because Graph Isomorphism is in coAM $\subseteq$ coNP/poly [8] but not known to lie in coNP. However a similar proof to Theorem 2.3 will show that $\mathrm{NP} \cap \mathrm{VSLL}$ is low for Θ_2^p. Since Graph Isomorphism is in NP, Graph Isomorphism in VSLL implies Graph Isomorphism is low for Θ_2^p, which is not known to hold.

Acknowledgments

The authors would like to thank Lane Hemaspaandra and Falk Unger for useful discussions.

References

[1] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, **38**:150–164, 1989.

[2] D. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science,* **104**(2):263–283, 1992.

[3] J. Cai. $S_2^p \subseteq ZPP^{NP}$. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science,* pages 620–629. IEEE Computer Society Press, Los Alamitos, CA, 2001.

[4] J. Cai, V. Chakaravarthy, L. Hemaspaandra, and M. Ogihara. Competing provers yield improved Karp–Lipton collapse results. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, pages 535–546, Springer-Verlag Lecture Notes in Computer Science 2607, 2003.

[5] J. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal on Foundations of Computer Science*, **2**(1):67–76, 1991.

[6] R. Canetti. More on BPP and the polynomial-time hierarchy. *Information Processing Letters* **57**(5):237–241, 1996.

[7] L. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theoretical Computer Science*, **43**:43–58, 1986.

[8] O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, **38**(3):690–728, 1991.

[9] J. Kadin. $P^{NP[\log n]}$ and sparse Turing-complete sets for NP. *SIAM Journal on Computing.* **17**(6):1263–1282, 1988. Erratum, **20**(2):404, 1991.

[10] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 302–309, ACM Press, New York, NY, 1980.

[11] M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its application.* Springer-Verlag, New York, NY, 1993.

[12] M. Ogiwara and O. Watanabe. On polynomial time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal of Computing.* **20**(3):471–483, 1991.

[13] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity* **7**(2):152–162, 1998.

[14] F. Unger. On small hard leaf languages. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, pages 781-792, Lecture Notes in Computer Science 3618, Springer-Verlag, 2005.

[15] C.-K. Yap. Consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science* 26:287–300, 1983.