Finitary Substructure Languages* 
With Application to the Theory of NP-Completeness

Kenneth W. Regan
Cornell Mathematical Sciences Institute
201 Caldwell Hall, Ithaca, NY 14853-2602

Abstract

We consider decision problems which involve the search for a fixed, finite amount of information hidden somewhere in the input. In terms of polynomial complexity these "finitary substructure languages" (k-SL's) are much like tally sets. For instance, every k-SL A p-Turing reduces to a canonically associated tally set $T_k$, and so cannot be $\mathcal{NP}$-hard unless the polynomial hierarchy collapses to its second level. However, we construct a recursive 1-SL $B$ which does not p-m reduce to any sparse set whatsoever. Hence k-SL's have different structural properties.

Many familiar $\mathcal{NP}$-complete sets equal free unions $L_k$ of IC-SL $L_k$. We support an assertion that every such $L$ is p-isomorphic to SAT. This specialization of the Berman-Hartmanis conjecture preserves some of its "natural problem" spirit, and escapes recent technical evidence against the original.

We also show that whether $A$ is p-T equivalent to $T_k$ above is tied to whether k-DNF formulas can be learned deterministically by oracle queries alone in the Valiant model. We give a finite-injury priority construction which highlights obstacles to establishing certain properties for recursive sets. In sum, we promote the concept of a k-SL as simple, appealing, and useful to other areas of research.

1. Introduction

Many oft-studied decision problems take the form of asking whether a given relational structure has a substructure of a certain kind. For instance, the well-known $\mathcal{NP}$-complete CLIQUE problem has the form:

\[ \text{Input: } (G, k), \text{ where } G \text{ is a simple graph and } k \in \mathbb{N}. \]
\[ \text{Query: Is } K_k \text{ (the complete graph on } k \text{ vertices) an induced subgraph of } G? \]

Regarding CLIQUE as a language of pairs, notice that every "cross section" over $k$ is the language

\[ k\text{-CLIQUE} := \{G \mid K_k \text{ occurs somewhere in } G\} \]

Here the size of the desired substructure is fixed over all instances; if one counts edges then the size is $k(k-1)/2$. The Hamilton circuit problem asks whether a given graph $G$ has a circuit $C$ passing through each vertex exactly once. Although here the size of $C$ depends on the instance $G$, a slight generalization recasts HAM in the form of CLIQUE:

\[ \text{Input: } (G, k) \text{ as above.} \]
\[ \text{Query: Does } G \text{ have a cycle } C \text{ on } k \text{ distinct vertices?} \]

This is still $\mathcal{NP}$-complete, owing to the cases $k = \lceil \sqrt{V(G)} \rceil$. Each cross-section $k$-HAM depends only on the presence or absence of a substructure of size $k$.

Observe that both $k$-CLIQUE and $k$-HAM are decidable in time $n^k$. Can this bound be improved, or more important, can it be replaced by a time bound of $n^\beta$ independent of $k$? If not, then $\mathcal{NP} \neq \mathcal{P}$. If so, and if programs realizing the $n^\beta$ time bound are easy to come by in terms of $k$, then $\mathcal{NP} = \mathcal{P}$. Thus the $\mathcal{P} \neq \mathcal{NP}$ question is nearly equivalent to the question of whether these "fixed-size substructure languages," each of which belongs to $\mathcal{P}$, have polynomial solving algorithms whose running times are not sensitive to the size of the object being searched for.

Accordingly we maintain that these languages, which we call k-SLs, merit attention in their own right. Their motivation is to isolate the difficulty of searching for an object from that of recognizing the object itself. When the size of the possible objects is fixed, then the latter is negligible. The information contained in an instance of a k-SL decision problem depends only on whether such a small object is present, and if so, on how well it is hidden.

*This research was supported partly by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University, Contract DAAG 29-85-C-0018, and partly by the Fellows of Merton College, Oxford.

**The other possibility is that $n^\beta$-time algorithms $A_k$ for $k$-CLIQUE exist for all $k$, and yet still $\mathcal{NP} \neq \mathcal{P}$, e.g. because the program-size complexity of $A_k$ increases dramatically with $k$. That such possibilities cannot be neglected is indicated by recent research of Robertson and Seymour.
2. K-SLs in String Encodings

Let Γ be a finite alphabet, which we always suppose contains the the binary alphabet Σ := {0, 1}. For any \( k > 0 \), \( \Gamma^k \) stands for \( \{ z \in \Gamma^* \mid |z| = k \} \), and \( \Gamma^k_s \) stands for \( \{ z \in \Gamma^* \mid |z| \leq k \} \). For any \( z \in \Gamma^* \) and \( j \in \mathbb{N}^* \) with \( \max \{ i \mid i \in I \} \leq |z| \), we write \( z_I \) for the string formed by concatenating the bits indexed by \( I \).

**Definition 2.1.** A string-index relation is a subset \( \rho \subseteq (\mathbb{N}^* \times \Gamma^*) \) such that for all \( (I, w) \in \rho, \ |I| = |w| \). The relation has order \( k \) if \( |I| \leq k \) for all \( (I, w) \in \rho \).

**Definition 2.2.** (a) For any string-index relation \( \rho \), \( L(\rho) \) denotes the set \( \{ z \in \Gamma^* \mid \text{for some } (I, w) \in \rho, \ z_I = w \} \).

(b) For any language \( A \) and \( k \geq 0 \), \( \rho^k_A \) denotes the relation \( \{ (I, w) \mid |I| = |w|, |I| \leq k, \text{ and for all } z_I = w \Rightarrow z \in A \} \).

**Definition 2.3.** A set \( A \subseteq \Gamma^* \) is a finitary substructure language if for some \( k \geq 0 \) we have \( L(\rho^k_A) = A \).

When \( k \) is understood we omit it and write ‘\( \rho_A \)’. We call any \( (I, w) \in \rho_A \) an \( A \)-determining pair. It is possible for a \( k \)-SL \( A \) to equal \( L(\rho) \) for some order-\( k \) string-index relation \( \rho \) different from \( \rho_A \), though always \( \rho \subseteq \rho_A \). In other words, \( \rho_A \) is the maximum relation which “determines” \( A \). Note that there are uncountably many, hence arbitrarily complex, string-index relations for \( A \). However, \( \rho_A \) is always recursive when \( A \) is recursive, as we show in §3.

**Examples:** Let every string \( x \in \Sigma^* \) encode a graph \( G_x \) in the following familiar manner: the edge set \( E(G_x) \) equals \( \{ (v, w) \mid x_v = \text{"1"} \} \), where \( (v, w) \) is the \( j \)th pair in the usual enumeration of 2-subsets of \( \mathbb{N}^* \) [namely \((1, 2), (1, 3), (2, 3), (1, 4), (2, 4), \ldots \)] Then for any \( m \geq 2 \), the language \( A := \text{m-CLIQUE} \) becomes a \( k \)-SL with \( k = m(m-1)/2 \).

Moreover, in any pair \( (I, w) \in \rho_A \), the string \( w \) equals \( 1^k \) and stands for the clique formed from the edges in \( I \). Under the same encoding, \( A \)-CIRCUIT becomes a \( k \)-SL. Let \( C_A \) denote the language of graphs having a chordless circuit of size \( m \). Then \( C_A \) is a \( k \)-SL \( (k = m(m-1)/2) \) whose determining pairs \( (I, w) \) have \( w \neq 1^k \) (except for \( m = 3 \)). Other relational structures besides graphs can be encoded via similar polynomial-time computable numberings of ordered subsets of \( \mathbb{N}^* \).

**Definition 2.4.** For any \( k > 0 \), \( k \)-SL denotes the class of \( k \)-SLs. \( \text{SL} \) stands for \( \bigcup_{k=0}^{\infty} k \)-SL.

**Proposition 2.1.** (a) Every \( k \)-SL \( A \) is a union of basic open languages \( z_1^* \) over \( z \in \Gamma^* \). That is, if \( x \in A \) and \( x \in y \) (read: \( x \) is a prefix of \( y \)), then \( y \in A \).

(b) If a \( k \)-SL \( A \) is nonempty, then \( A \) has exponential density, meaning that for some \( c > 0 \) and almost all \( n \), \( \operatorname{card}(A \cap \Gamma^n) \geq c \cdot |\Gamma|^n \).

(c) The union of any number of \( k \)-SLs is a \( k \)-SL.

(d) The intersection or concatenation of two \( k \)-SLs is a \((2k)\)-SL.

(e) \( \text{SL} \) is not closed under countable unions, nor under \((\lambda\text{-free})\) homomorphism or inverse homomorphism, nor under finite variations of its member languages.

(f) Every cofinite language belongs to \( \text{SL} \), but the only finite language in \( \text{SL} \) is \( \emptyset \).

Part (a) says that every \( k \)-SL is open in the usual topology on \( \Gamma^* \). The elementary proofs are left to the reader.

3. K-SLs, Tally Sets, and NP-Hardness

In one sense, a \( k \)-SL \( A \) contains only a polynomial amount of information. This is because on any input \( z \) of length \( n \), there are only \( O(n^k) \) possible “hiding places” \( I \) for a substructure of size \( k \). The number of strings \( w, |w| \leq k \), which could represent the substructure is finite and independent of \( n \). A language \( B \) which is sparse, meaning that \( \operatorname{card}(z \in B \mid |z| \leq n) = O(n^k) \) for some \( k > 0 \), has polynomial information content in a more familiar sense. A special case of a sparse set is a tally set \( T \), which we define to be a subset of \( 0^* \). Here we have \( k = 1 \).
We connect the two senses by associating to each $k$-SL $A$ a $1$-SL $S_A$ and a tally set $T_A$. The main result of this section estimates the complexity of the transformations among $A$, $S_A$, $T_A$, and a fourth set $R_A$ related to $\rho_A$. Part of this result allows us to conclude that a $k$-SL cannot be $\mathcal{NP}$-complete unless the polynomial hierarchy collapses to its second level.

The chief technical tool is an efficient means of coding members of $(\mathbb{N}^+)_{\leq 2}^k$ (meaning ordered sequences of up to $k$ positive natural numbers) into a single natural number.

**Lemma 3.1.** For any $k \geq 1$, there is an injection $\psi_k$ from $(\mathbb{N}^+)_{\leq 2}^k$ to $\mathbb{N}$ such that both $\psi_k$ and $\psi_k^{-1}$ are computable in linear time, where only the constant factor depends on $k$.

**Proof.** Let $\langle \cdot, \ldots, \cdot \rangle_k$ denote $\langle \cdot, \ldots, \langle \cdot, \ldots, \cdot \rangle \rangle$ ($k$ times), where $\langle \cdot, \ldots, \cdot \rangle$ is the linear-time computable pairing function from [Reg88]. For any $I = \{i_1, \ldots, i_r\}$ define $\psi(I) := c_{i_1, i_2, \ldots, i_r, 0, \ldots, 0}_k$.

This is not a bijection since $I$ is ordered. We suspect a linear-time bijection can be constructed by expanding on [Reg88], but spare the effort since an injection suffices here. Now list the members of $\Gamma^k := 0, 1, \ldots, (\Gamma^k)^{\ast}$. $1$.

**Definition 3.1.** For all $I \in (\mathbb{N}^+)_{\leq 2}^k$ and $w \in \Gamma^{|I|}$, put $\psi_k(I, w) := \psi_k(I) \cdot |I|^{k} + j$, where $\psi_k = w_0^{k-1}j$.

**Lemma 3.2.** For any fixed $k$, $\psi_k$ is 1-1 into $\mathbb{N}$, and is both computable and invertible in linear time.

**Proof.** To compute $\psi_k^{-1}(m)$, first compute $l := \frac{m}{\Gamma^{|I|}k}$ and $j := m - l \Gamma^{|I|}k$. Then take $I := \psi_k^{-1}(l)$, or error if this is not in ascending order, and $w :=$ the string obtained from $\psi_k^j$ by deleting the last $k-1$ ‘$0$’s, or error if the ‘$0$’s are not there. All this takes only linear time since $k$ is fixed, and error happens iff $m \not\in$ Ran($\psi_k$). The other claims for $\psi_k$ itself are clear.

**Definition 3.2.** For any $k$-SL $A$, form the associated tally representation $T_A$, 1-SL representation $S_A$, and determining-pair representation $R_A$ by

$$T_A := \{0^n \mid \text{for some } (I, w) \in \rho_A, \psi_k(I, w) = n\}$$
$$S_A := \{z \in \Sigma^* \mid \exists n \leq |z| : 0^n \in T_A \land z_n = \text{‘}1\text{’}\}$$
$$R_A := \{\text{ unary}(I)z \mid (I, z) \in \rho_A\}.$$ 

Here unary($I$) codes $I = \{i_1, \ldots, i_r\}$ as $10^{i_1}10^{i_2}\cdots10^{i_r}$. $R_A$ is intended to stand for the relation $\rho_A$. Under a unary encoding of the latter, we have $R_A \text{ lin } \rho_A$.

**Lemma 3.3.** For any $k$-SL $A$, $R_A \text{ lin } T_A$.

**Proof.** The map $f : \text{ unary}(I)z \mapsto 0^n$, where $n := \psi_k(I, z_1)$, reduces $R_A$ to $T_A$. For the reverse reduction $g(0^n)$, first compute $\psi_k^{-1}(n)$. If this gives error then $0^n \not\in T_A$, so set $g(0^n)$ to be a fixed string outside $R_A$. Else take $(I, w) := \psi_k^{-1}(n)$ and $g(n) := \text{ unary}(I)z$, where $z$ is such that $|z| = \max(I)$, $z_1 = w$, and $z_i = \text{ ‘}0\text{’}$ for $i \not\in I$. Both $f$ and $g$ are linear-time computable despite the change to unary notation.

In the following, the $\text{ lin }$ reductions do not use $\psi_k$ or $\psi_k^{-1}$, while the exponent in the $\text{ lin }$ reductions does depend on $k$. The subscripts ‘ctt’ and ‘dtt’ stand for conjunctive and disjunctive truth-table reducibility, notions which are weaker than many-one reducibility but stronger than Turing reducibility, while ‘exp’ stands for time $2^\mathcal{O}(n)$.

**Theorem 3.4.** For any $k$-SL $A$, the following "upwards" and "downwards" reductions hold:

(a) $A \text{ ctt } R_A$
(b) $A \text{ dtt } T_A$
(c) $A \text{ ctt } S_A$
(d) $A \text{ dtt } T_A$
(e) $R_A \text{ exp } A$
(f) $T_A \text{ exp } A$
(g) $S_A \text{ lin } T_A$
(h) $S_A \text{ exp } A$.

**Proof.** (a) For any $z$, consider every $I \subseteq \{1, \ldots, |z|\}$ of size $\leq k$. For each such $I$, query the oracle on the string unary($I$)$z$. Then $z \in A$ iff some query produces a ‘yes’ answer, and there are $O(n^k)$ queries.

(b) This follows from (a) and Lemma 3.3.
(c) This reduction has the form $0^n \mapsto 0^{n-1}$.
(d) Given $z$, form a string $y \in \Sigma^*$ of length equal to $|\Gamma^k \cdot \psi_k(\{ |z| + 1-k, \ldots, |z| \})|$ by setting $y_n := \text{ ‘}1\text{’}$ if and only if $0^n = f(I, z)$ for some $I \subseteq \{1, \ldots, |z|\}$ of size $k$, where $f$ is the reduction from $R_A$ to $T_A$ in Lemma 3.3. Then $y \in S_A$ iff for some $I$, unary($I$)$z \in R_A$ (where $I \subseteq \{1, \ldots, |z|\}$, $|I| \leq k$) $\Rightarrow z \in A$. (Note that $|y|$ is $O(n^k)$.)

(e) Given $z$ and $I \subseteq \{1, \ldots, |z|\}$ of size $\leq k$, take $m := \max(I)$, and let $M$ be an OTM which queries its oracle on all strings $y$ such that $|y| = m$ and $y_1 = z_1$. If
all answers are ‘yes’ then $M$ accepts its input unary(y); else $M$ rejects. If unary(y) $\in R_A$, then $y_1 = z_1 \Rightarrow y \in A$ for all y, and so all queries are answered positively with A as the oracle set, giving unary(y) $\in L(M^A)$. If unary(y) $\notin R_A$ then by the definition of $\rho_A$ for a $k$-SL A there is a string z such that $z_1 = z_2$ but $z \notin A$. Take $y := z_2 \cdots z_n$; then also $y_1 = z_2$. Were $y \in A$ then we would have $z \in A$, by Proposition 2.1(a) since $y \in z$. Hence $y \notin A$, and since M queries y during its computation, unary(y) $\notin L(M^A)$. Thus M gives a $2^O(n)$-time ct reduction from $R_A$ to A.

(f) This follows from (e) and Lemma 3.3.

(g) This reduction has the form: Given z, query $0^n$ for each n such that $z_n = '1'$. This follows from (g) and (f).

COROLLARY 3.5. For any $k \geq 0$ and k-SL A, A is recursive if and only if $\rho_A$ is recursive.

COROLLARY 3.6. Any k-SL A has small circuits.

Proof. It is well-known that a language A has small circuits if A $\leq^c L$ for some polynomially sparse set L, so this follows from Theorem 3.4(b).

For the same reason, we can now see that A is unlikely to be $\mathcal{NP}$-hard under polynomial-time Turing reducibility:

COROLLARY 3.7. For any k-SL A, if A is $\mathcal{NP}$-hard under $\leq^c$ then $\mathbb{P} = \Sigma^P_2$. If also A $\in \text{co-}\mathcal{NP}$ then $\mathbb{P} = \Sigma^P_2$.

Proof. R. Karp and R. Lipton [KL80] show that if there is a $\mathcal{NP}$-hard set with small circuits, then $\mathbb{P} = \Sigma^P_2$. The second statement rests on the theorem of T. Long [Long82] that if $\Sigma^P_2$ contains a sparse set which is $\leq^c$-hard for $\mathcal{NP}$, then $\mathbb{P} = \Sigma^P_2$, and the observation that $A \in \mathcal{NP} \Rightarrow R_A, T_A \in \text{co-}\mathcal{NP}$ for any k-SL A. (To certify $(I,w) \notin \rho_A$, guess $z$ such that $z_1 = w$ and certify $z \notin A$.)

We suspect that Long’s proof can be adapted to show the same conclusion if $A \in \mathcal{NP}$. This also leaves open the question of whether techniques directly tailored to k-SLs can improve these “collapse” results. For instance,

Open question: Can one show that the existence of a k-SL which is $\mathcal{NP}$-hard under $\leq^c$ implies $\mathbb{P} = \mathcal{NP}$?

This is plausible because S. Mahaney [Mah82] has shown that if there is a $\leq^c$-hard set A for $\mathcal{NP}$ which is polynomially sparse, then $\mathbb{P} = \mathbb{P}$. With a k-SL A we can begin to mimic Mahaney’s construction. However, the attempt runs up against the complexity of deciding whether two given strings x, y share an I such that $(I,x)$ and $(I,y)$ are both in $R_A$. The corresponding juncures of [Mah82] involve only the trivial task of deciding whether $x = y$. Also, a second technique which [Mah82] uses to convert a sparse $\mathcal{NP}$-hard set into a sparse $\mathcal{NP}$-complete set does not carry over to k-SLs. It would be nice to be able to climb above these technical problems and use [Mah82] directly by showing that every k-SL A $\leq^c$-reduces to some sparse set $L$, as we have done for $\leq^c$ on the shoulders of [KL80] and [Long82]. However, this attempt does not even get one leg up:

THEOREM 3.8. There is a recursive t-SL A which does not $\leq^c$-reduce to any sparse set whatsoever.

In section 6 we prove a stronger result by constructing a recursive t-SL A which does not reduce to any set $B$ whose density is strictly less than exponential.

4. The K-SL Reduction Problem

It is natural to ask whether the exponential-time reductions in Theorem 3.4 can be improved. Viewing the chain of transitivity, we focus on the following

Open question: Is it the case, for all k-SLs A, that $T_A \leq^c A$, or equivalently, that $R_A \leq^c A$?

This question is closely tied to that of whether Boolean formulas in k-DNF can be learned deterministically in polynomial time “by queries alone,” with reference to the model of machine learning offered by L. Valiant [Val84]. The connection applies to k-SLs over $\Sigma = \{0,1\}$. To see it, consider the instance of ‘unary(y) $\in R_A$’ given by $k = 3$, $I = \{2,3,5\}$, and $z_1 = '101'$. This corresponds to asking whether ‘$(z_2 \land z_3 \land z_5)$’ is a clause in the 3-DNF formula $\phi_e$ obtained from $\rho_A$ in this manner, where we are only concerned with the variables $z_1, \ldots, z_5$ and their negations. For any family of k-DNF formulas $\phi_\alpha = \phi(x_1, \ldots, x_n)$ which is consistent in the sense that any clause of $\phi_\alpha$ belongs to $\phi$, for all $r > n$, we can produce the corresponding k-SL A. Then $R_A \leq^c A$ implies that we can learn the “maximum”
\( k \)-DNF formula equivalent to \( \phi_a \) (see remarks following Definition 2.3 and in [Val84]) in polynomial time, by trying all \( O(n^k) \) possible clauses. Queries to \( A \) represent trial assignments to \( x_1, \ldots, x_n \) \( (m \leq n) \).

A single OTM \( M \) giving \( R_A = L(M^A) \) for all \( k \)-SLs \( A \) would yield a uniform algorithm for learning \( k \)-DNF, without using either the randomization and error tolerance of the full Valiant model, or the provision for requesting less of the full Valiant model, or the provision for requesting an appropriate oracle. (Note: In the full Valiant model our restricted sense, and yet this matters in cases such as (random) satisfiability and \( k \)-CLIQUE, the leeway adds no real power other than in coping with the "examples" half of the model alone.) We suspect that constructing a \( k \)-SL \( A \) for \( k \)-DNF can be learned probabilistically using the "examples" half of the model alone.) We suspect that adding an edge to a circuit would give a more meaningful counterexample to deterministic, exact learning than probabilistic learning.

**Theorem 4.1.** With reference to a linear ordering \( \preceq \) on \( \Gamma \), and the induced partial ordering \( \leq \) on \( \Gamma^* \) given by \( z \preceq y \iff (\forall i \leq |z|)[z_i \leq y_i] \), a \( k \)-SL \( A \) is positive if for all \( I \in (\Gamma^*)^k \) and \( v, w \in \Gamma^* \), \( (I,v) \in \rho_a \land v \preceq w \Rightarrow (I,w) \in \rho_a \).

In particular, any \( k \)-SL \( A \in \Xi^* \) such that \( (I,w) \in \rho_a \Rightarrow w \in \Gamma^* \) for all \( I \) is positive, and so \( k \)-CLIQUE and \( k \)-HAM are positive. For any \( k \)-SL \( A \), \( S_a \) is a positive \( k \)-SL. The chordless-circuit \( k \)-SLs \( CC_a \) of Section 2 are not positive, however, because adding an edge to a circuit makes it no longer chordless.

**Theorem 4.2.** If \( A \) is a \( k \)-SL, \( A \subseteq \Gamma^* \), then \( R_A \leq^* A \).

**Proof.** For a \( k \)-SL, the input to \( R_A \) (technically, to \( \rho_a \)) is an integer \( n \) in unary and a single character \( b \in \Gamma \).

Let \( M \) be an OTM which assigns an arbitrary linear order \( \preceq \) to \( \Gamma \), and on input \( (n,b) \) iterates the following process:

First find the least eligible string \( z \) under the partial ordering \( \preceq \) on \( \Gamma^* \) induced by \( \prec \) as in Definition 4.1. At the beginning, the eligible strings \( z \) are those with \( |z| = n \) and \( x_a = b \). (There will be a least eligible string \( z \) under \( \preceq \) at any stage.) Query \( z \). If the answer is 'no', then \( (n,b) \notin \rho_a \).

Now define \( y_i \) to be the string \( z_{i+1} \cdots z_n \), for all \( i \leq n \), and \( j \) to be the least value such that \( M^A \) queried on \( y_i \) answers 'yes'. With \( a := z_j \), we then have that \( (j,a) \in \rho_a \).

If \( j = n \), then \( a = b \), and so \( M^A \) halts and accepts \( (n,b) \).

Else, declare any string \( z \) such that \( z_j = a \) to be ineligible, and proceed to the next stage.

At the end of any stage, \( M^A \) either decides whether \( (n,b) \in \rho_a \) or proves that \( (m,a) \notin \rho_a \) for some \( m < n \) and \( a \in \Gamma \).

Since there are only \( (n-1)|\Gamma| \) such pairs, \( M \) halts after linearly many stages. Since each stage requires at most \( n \) queries, \( M^A \) runs in time \( O(n^2) \). Hence \( M \) reduces \( R_A \) to \( A \), for any \( k \)-SL \( A \).

The method does not by itself engender a polynomial-time backtrack algorithm for the case \( k = 2 \). A natural criterion of "good behavior" for a \( k \)-SL which might help here is:

**Definition 4.2.** A \( k \)-SL \( A \in \Gamma^* \) is nicely branching if there is no \( z \in \Gamma^* \) such that \( z \notin A \) but for all \( a \in \Gamma \), \( za \in A \).

Open question: If \( A \) is a \( 2 \)-SL which is nicely branching, does \( R_A \leq^* A \)?

91
5. NP-Complete Sets Built From K-SLs

In Sections 1 and 4 we observed that several natural \( \mathcal{NP} \)-complete languages \( L \subseteq \Sigma^* \) have cross-sections \( L_A \), each of which is a \( k \)-SL such that all determining pairs \( (I,w) \) have \( w \in 1^* \).

**Definition 5.1.** A \( k \)-SL \( A \subseteq \Gamma^* \) is a top \( k \)-SL if for some \( \alpha \in \Gamma \) and all \( (I,w) \in \rho_A \), \( w \in \alpha^* \).

Clearly every top \( k \)-SL is positive and nicely branching. Every positive \( I \)-SL over a two-letter alphabet is either top or co-finite. For \( k \geq 2 \), positive \( k \)-SLs over \( \Sigma = \{0,1\} \) which are not nicely branching. Hence we emphasize top \( k \)-SLs.

**Definition 5.2.** A language \( A \subseteq \Sigma^* \times \Sigma^* \) is a \( T^* \)SL if for all \( k \), the language \( A_k := \{ z \mid (z,k) \in A \} \) is a top \( k \)-SL.

Thus CLIQUE and HAM (as generalized in §1) are \( \mathcal{NP} \)-complete \( T^* \)SLs. SAT can similarly be transformed into a \( T^* \)SL, at least in its \( 3 \)-CNF form, by the recipe: "The partial assignment \( w \) to the \( 3k \) variables denoted by \( I \) (counting negated variables separately) satisfies \( k \) clauses of the \( 3 \)-CNF formula \( \phi \)." A good many other \( \mathcal{NP} \)-complete problems listed in [GJ79] yield \( T^* \)SLs. We first characterize the \( T^* \)SLs which belong to \( \mathcal{NP} \) before asking which ones are \( \mathcal{NP} \)-complete:

**Definition 5.3.** (a) For any top \( k \)-SL \( A_k \) over \( \Sigma \), (re-)define \( R_A \) to be the set \( \{ \text{unary}(l) \mid (l,l)^k \in \rho_A \} \).

(b) For any \( T^* \)SL \( A \), define \( R_A := \{ \text{unary}(l) \mid \text{unary}(l) \in \rho_{A_{|l|}} \} \).

In the following, we write \( I \) for \( \text{unary}(l) \) when the intent is clear, and similarly regard \( k \) as written in unary.

**Proposition 5.1.** For \( T^* \)SLs \( A \), \( A \in \mathcal{NP} \iff R_A \in \mathcal{NP} \).

**Proof.** (\( \Rightarrow \)) For all \( I \), let \( z(l) \) be the unique string of length \( \max(I) \) such that \( z_1 = '1' \Rightarrow i \in I \). Then \( I \in R_A \iff z(l) \in A_{|l|} \iff (z(l),l,l)^k \in \rho_A \), since \( A_{|l|} \) is a top \( |l| \)-SL.

(\( \Leftarrow \)) For all \( (z,k) \), taking \( n := |l| \), we have: \( (z,k) \in A \iff \text{for some } I \in (\Sigma^*)^k \) with \( \max(I) \leq n \) (hence \( |l| \leq kn \)), \( I \) belongs to \( R_A \) and \( z_1 = '1' \) for all \( i \in I \). If \( R_A \in \mathcal{NP} \), then this is an \( \mathcal{NP} \)-predicate defining \( A \).

In fact, CLIQUE and (generalized) HAM are examples of \( \mathcal{NP} \)-complete \( T^* \)SLs \( L \) which have \( R_A \in \mathcal{NP} \). What interests us is the following measure of how many index places are "critical" to membership in a \( T^* \)SL.

**Definition 5.4.** (a) For any \( T^* \)SL \( A \) and \( k \geq 1 \), define the critical set \( C_A^k := \{ n \in \mathbb{N}^+ \mid (3I \in R_{A_k} \mid n = \max(l)) \} \).

(b) A \( T^* \)SL \( A \) is not critically dense if there is a polynomial-time computable function \( f: (\mathbb{N}^*)^3 \to (\mathbb{N}^*)^k \) such that for all \( a,b,k \in \mathbb{N} \), \( f(a^b,0^k,0^k) \) = \( M_k := \{ m_1,...,m_n \} \), where \( a < m_1 < ... < m_n \) and \( M_k \cap C_A^k = \emptyset \). We also require \( M_k \subseteq M_{k+1} \) for all \( b \).

The language CLIQUE is not critically dense. For instance, under the lexicographic numbering of graph edges used in the standard encoding, any edge involving vertex 5 can only be the highest-numbered edge of a \( k \)-clique for \( k \leq 6 \), so \( (5,1) \not\in C_{\text{CLIQUE}}^k \) for all \( k \leq 6 \) and \( v \in \mathbb{N}^+ \). This yields the desired plenitude of non-critical edges. The known result that an \( \mathcal{NP} \)-complete set is \( \mathcal{NP} \)-isomorphic to \( SAT \) iff it is a polynomial cylinder motivates the following:

**Theorem 5.2.** If \( L \) is a \( T^* \)SL which is not critically dense, then \( L \) is a polynomial cylinder.

**Proof.** It is sufficient (see [BeHa77]) to design a \( 1 \)-1 function \( \text{pad}_k(k,z,y) \): \( \mathbb{N} \times (\mathbb{N}^\ast)^k \times \mathbb{N} \to \Sigma^\ast \) which is both computable and invertible in polynomial time, such that for all \( k,z,y \), \( (z,k,y) \in L \Rightarrow (\text{pad}_k(k,z,y),k) \in L \). An informal description of \( \text{pad} \) runs as follows:

First compute \( \text{pad}(I,z,0^1,0^k) := \{ m_1,...,m_{|z|+1} \} \).

Let \( z \) be the unique string of length \( m_{|z|+1} \) such that \( z \subseteq z \), and for all \( i > |z|, z_i = '1' \Rightarrow i = m_{|z|+1} \) or for some \( j \leq |z|, i = m_j \) and \( y_j = '1' \). The string \( z \) ends in a \( '1' \). Finally define \( \text{pad}_k(k,z,y) := (z,k) \).

If \( z \in L_k \), then clearly \( z \in L_k \) since \( z \subseteq z \). If \( z \not\in L_k \), then there does not exist \( I \in R_{L_k} \) such that \( z \subseteq I \), because \( z_1 = '0' \) by construction \( |I| := \max(I) \). Hence \( z \not\in L_k \).

Last, to decode \( (z,k) \), count the number of \( '0' \)s after the final \( '1' \) to determine \( z \). Then compute \( \text{pad}(I,z,0^1,0^k) \) for \( b := 1,2,... \) in order until reaching the first \( b \) such that \( M_b \not\subseteq M_{b+1} \). (The property \( M_b \subseteq M_{b+1} \) is used here.) Then \( 1|y| = b-1 \), and \( y \) can be read off from \( z \) and \( M_b \). This process requires only polynomial time since \( b \leq |z| \).
Remark: Without the property $M_0 \subseteq M_{n+1}$, we do not see how to do the decoding. That the proof is this sensitive seems surprising, and we ask for a more direct means of seeing how to do the decoding. That the proof is this sensitive.

For all its ungainliness in the abstract, the proof of Theorem 5.2 actually mirrors how NP-complete sets are padded in practice.

The intuition for this concerns the nature of any $\leq^r_\Sigma^*$-reduction $f$ from CLIQUE to $L$. It seems unlikely that $f$ could cause a dramatic shift in the density of the critical set: if it were to take many graphs $G$ which do not nearly have $k$-cliques into strings $y$ which are “almost” members of $L_x$, then $f$ would incorrectly take graphs $G$ which have many $K_{k-1}$'s but no $K_k$ into strings which belong to $L_x$. Or $f$ might need some way of recognizing graphs which come close to having $k$-cliques, but this is still $\mathcal{NP}$-complete.

Also intuitively, having $k$-SLs for cross sections is just the kind of “fine structure” in an $\mathcal{NP}$-complete set $A$ which would be destroyed on moving to the $\mathcal{NP}$-complete set $f(A)$ when $f$ is a one-way function. It is immediate that $f(A)$ cannot be a $T^*$SL (or even have one $k$-SL cross section) if $f$ is a “scrambling function” [KMR89], by the closure of $X^*$ under $f$. Likewise the languages $K^f(f)$ from [JoYg85] are then not $T^*$SL's since they are contained in $\text{Ran}(f)$.

Like the Berman-Hartmanis conjecture, ours has the “defect” of implying $\mathcal{NP} \not\subseteq \mathcal{P}$. It may be more amenable to attack, however. One can focus on $T^*$SLs $L$ such that $L$ as well as $L$ has exponential density. On the other hand, perhaps it is still possible to raise doubt about it by constructing a recursive $T^*$SL $L$ which is not $\mathcal{P}$-isomorphic to its polynomial cylindrification. This may be a tall order, however. To appreciate the likely problems involved, we consider the nature of diagonalization constructions whose object is to produce a language closed under $\geq$.

6. Diagonalization Methods and K-SLs

Working with finitary substructure languages, or within any family of languages $A$ which are closed under extensions, changes the familiar patterns of diagonalization constructions in the literature. This is because adding a single string to $A$ entails adding infinitely — and exponentially — many more. We illustrate the differences by giving a $k$-SL analogue of the construction in [BelHa77] of a recursive set $B \subseteq \Sigma^*$ such that all correct polynomial-time many-one reductions from $B$ are $1$-$1$ a.e. (Such a set $B$ is said to be strongly $\mathcal{P}$-immune [BaS85]). Here we say a function $f: \Sigma^* \rightarrow \Sigma^*$ is correct on $B$ if there do not exist $x,y \in \Sigma^*$ such that $x \in B$, $y \notin B$, and $f(x) = f(y)$, or equivalently, if $f$ reduces $B$ to the set $f(B)$.

**Definition 6.1.** (a) Given a function $f: \Sigma^* \rightarrow \Sigma^*$, define its injectivity to be the function sending $n \in \mathbb{N}$ to $\text{card}(\{z \in \Sigma^n \mid f(z) = f(y)\})$.

(b) The density of a language $L \subseteq \Sigma^*$ is standardly defined by its census function $c_L(n) := \text{card}(L \cap \Sigma^n)$.

An alternate definition of the census function counts the strings in $L$ of length $\leq n$ rather than $= n$, and we could have done the same for the injectivity of $f$. That the difference does not matter to the estimates in our theorems comes out in their proofs.

**Definition 6.2.** (a) A function $h: \mathbb{N} \rightarrow \mathbb{N}$ is fully exponential if $h$ is $\Omega(2^n)$; i.e., if for some $c > 0$ and all but finitely many $n$, $h(n) \geq c \cdot 2^n$.

(b) The function $h$ is strongly subexponential if $h$ is $o(2^n)$ for every $c > 0$.

**Theorem 6.1.** Let $h: \mathbb{N} \rightarrow \mathbb{N}$ be any recursive function which is not fully exponential. Then we can construct a recursive positive $I$-SL $A$ such that (i) the only correct $\leq^r_\Sigma^*$-reductions from $A$ have injectivity $\geq h(n)$ for infinitely many $n$, and (ii) $A$ does not $\leq^r_\Sigma^*$-reduce to any set of strongly subexponential density.

Theorem 3.8 follows immediately, since every polynomial is strongly subexponential. Part (i) does not imply (ii) by itself, because it is possible for a correct reduction from a set $A$ to be fully exponentially injective while $f(A)$ consists of a single string.

**Proof.** We may suppose that $h(n) \geq \frac{2^n}{2^{n^2}}$ for all $n$; else replace $h$ by $h(n) := \max\{h(n), 2^n/2^{n^2}\}$. Since $h$ is not $\Omega(2^n)$, we have that for any $c > 0$ and $m \in \mathbb{N}$ there is $n > m$ such that $h(n)/2^n \leq c$. Let $\{f_i\}_{i=1}^\infty$ be a recursive
enumeration of $\mathcal{FP}$. Last, let $\pi : \mathbb{N} \to \mathbb{N}$ be any recursive function such that $\pi^{-1}(m)$ is infinite for all $m \in \mathbb{N}$.

We construct $A$ in stages $s = 1, 2, 3, \ldots$ by a “wait-and-see” argument, where at the end of each stage $s$ there is a number $n_s > s$ such that the membership or non-membership in $A$ of all strings of length $\leq n_s$ has been determined. We also “lose control” of all strings of length $\geq n_s$ which do not begin with $n_s$ many ‘0’s. At the beginning we have $n_0 := 0$, since we cannot have $\lambda \in A$, and all requirements $R_i := \{z \in A \land y \notin A \land f_i(z) = f_i(y)\}$ are initially uncanceled.

Stage $s$: Take $c := 1/2^{m_s-1}$. Since $h$ is recursive but not $\Omega(2^n)$, we can recursively find an $n > n_{s-1}$ such that $c \cdot 2^n > h(n)$. Set $D_s := \{z \in \Sigma^* \mid |z| = n \}$, then $\text{card}(D_s) \geq h(n)$. List the uncanceled requirements at stage $s$ in ascending order as $(i_1, \ldots, i_k)$, topped off by $i_k := s$. Find the least $j$ such that $f_{i_j}$ is not injective on $D_{s-1}$; if none exists, set $j := +\infty$.

If $\pi(s) \geq j$, then we are allowed to satisfy requirement $R_{i_j}$. Take any distinct $x, y \in D_s$ such that $f_{i_j}(x) = f_{i_j}(y)$, and let $m$ be any place in which $x$ and $y$ differ. Place the string $0^{m_s-1}1$ into $A$, and then only add to $A$ those strings of length $\leq n$ which extend strings previously added to $A$. Finally set $n_s := n$. Doing this satisfies $R_{i_j}$ because if we wlog suppose $x_m = '1'$ and $y_m = '0'$ then $z \in A$, while $y$ is never added to $A$ since future stages consider strings in $0^{m_s-1} \Sigma^*$. Only then cancel $i_j$ and go on to the next stage.

If $\pi(s) < j$, then we say “pumping” has priority. Let $\ell := \pi(s)$. Then $f_{\ell}$ is uncanceled and injective on $D_{\ell}$, so it is injective on the half of $D_{\ell}$ represented by $D^*_s := \{z \in D_s \mid z = 0^{\ell_s-1}1\Sigma^* \frac{n_s}{2}\}$; Place the string $0^{\ell_s-1}1$ into $A$ and set $n_s := n_{s-1} + 1$. Thus proceed to the next stage. This ends the construction.

For any $f_s$, if the requirement $R_{i_s}$ is satisfied during the construction then $f_s$ is not a correct reduction from $A$. So consider any $f_i$ for which $R_i$ is never satisfied. First, this means that there are infinitely many $n$ such that $\text{card}(f_i(z)) \geq h(n)$, so that (i) holds.

Second, there is a stage $s_0$ such that all requirements $R_{i_s}$, with $s' < s$ which are ever satisfied during the construction have been satisfied before stage $s_0$. Then there is an index $s$ such that at all stages $s \geq s_0$, $i = i_s$ is in the list of uncanceled requirements. By the choice of $\pi$, there are infinitely many $s \geq s_0$ such that $\pi(s) = \ell$, and at each such stage, pumping has priority. Hence the construction provides a set $D^*_\ell \subseteq A \cap \Sigma^*$ of cardinality at least $h(n)/2$ on which $f_s$ is injective.

Since $f_s$ is polynomial-time computable, there is a number $k$ such that $|f_s(z)| \leq |z|^k$ for all $z$, $|z| \geq 2$. Thus $f_s(A) \cap \Sigma^{2m^k}$ has cardinality at least $h(n)/2$. Hence for some $m \leq n^k$, $C_m := \text{card}(f_s(A) \cap \Sigma^{2m^k}) \geq h(n)/2m^k$. By our assumption on $h$, $C_m \geq 2^{n/2}/2m^k = 2^{n/2} \cdot 1 - \log_2 n$. Hence for all sufficiently large $n$, $C_m > 2^{|n|/3}$. This also implies that $m > n/3$, since $C_m \leq \text{card}(\Sigma^{2m})$. Since $m \leq n^k$, $m \geq m^{1/k}$, and so $C_m > 2(m^{1/k})^3$. For large enough $m$, in fact $m \geq 3(k+1)^4$, we have $m^{1/k}/3 > m^{1/k+1}$, and so $C_m > 3^{m^{1/k+1}}$.

Hence with $c := 1/k+1$ we can conclude that there are infinitely many $m$ such that $\text{card}(f_s(A) \cap \Sigma^{2m^k}) \geq 2^m n^c$. This shows (ii) and completes the proof.

**Corollary 6.2.** There is a positive $1$-SL $A \in \text{EXPSPACE}$ such that for all sets $B$, if $A \subseteq B$ then neither $B$ nor $\overline{B}$ has strongly subexponential density.

**Proof.** Apply the theorem with $h(n) := 2^n/2$, but change the assignment of $n_s$ in the “pumping” section of any stage $s$ to $n_s := n_r$ rather than $n_s := n_{s-1} + 1$. Thus we ensure that $D_{s-1} \subseteq A \cap \Sigma^*$. For any correct reduction $f_s$ and late-enough stage $s$ with $\pi(s) = \pi(f_s)$ is injective on $D_s \setminus D^*_s$. The same calculation shows that for any $\varepsilon > 0$ there are $\omega$-many $m$ such that $\text{card}(f_s(A) \cap \Sigma^{2m^k}) \geq m^{1/k} n^c$, so if $f_s(A) \subseteq B$, then $B$ cannot have any smaller density. The choice of $h(\cdot)$ makes it possible to compute $A$ in exponential time. (We suspect, in fact, that “elongating” the construction can make $A$ $n^{\log_2 m^k}$-time decidable, and so on.)

The strongly subexponential lower bound on the density of $f(A)$ is best possible, since the function $f : z \mapsto z \cdots z (n^{k-1} \text{ times})$ is $n^{k-1}$-time computable and $f(\Sigma^*)$ has density $2^{m^{1/k}}$. It is nice that the presence of $h(\cdot)$ does not interfere with this bound. The result would be much cleaner if this freedom also applied to the bound on the injectivity of those $f_i$ which are correct reductions. Leaving aside the element of “pumping” for clarity, we can show:

**Theorem 6.3.** There is a (possibly nonrecursive) positive $1$-SL $A \subseteq \Sigma^*$ such that the only correct $\Sigma^2$-reductions from $A$ are fully exponentially injective.
The reason our above "method of segments" construction doesn't suffice is that satisfying a requirement \( R_i \) lowers the constant of proportionality \( c_j = 2^{n_j} \), and this can go down often enough to prevent us from satisfying a requirement \( R_i < R_j \). To provide an eventually-constant \( c_i \) for each \( R_i \), we resort to a less straightforward construction.

**Proof.** Let \( [f_i]_{i=1}^n \) be as before. We dynamically associate to each requirement \( R_i \) a number \( b(i) \), which says we can only try to satisfy \( R_i \) using strings \( x,y \in \Sigma^*_n \). At the beginning, \( b(i) := 0 \) for all \( i \geq 1 \). (Our algorithm actually requires us just to assign \( b(0) := 0 \).) Unlike the previous proof, here requirements can become "uncanceled" (i.e., injured) at a later stage. Hence at each stage \( s \) we maintain \( b(i) \) for all \( i \leq s \), even if \( i \) has been canceled.

At each stage \( s \) at most one requirement \( R_i \) will be satisfied, and if so, all \( i' > i \) are then uncanceled, and \( b(i') \) is re-assigned to a value determined only by the action taken for \( R_i \). At this stage membership in \( A \) is defined for all strings of length \( \leq b(i') \), so \( b(i') \) provides a measure of how far the construction has progressed. An invariant of the program is that \( j > i \Rightarrow b(j) \geq b(i) \) at any stage. The aim is to make \( b(i) \) converge to a final value \( b(i) \) for all \( i \), with \( b(i) \rightarrow m \) as \( s \rightarrow m \), meanwhile satisfying \( R_i \) for all \( i \) such that \( f_i \) is not \( \Sigma^* \)-injective.

**Stage s:** Let \( i_1, \ldots, i_s = s \) be the currently-uncanceled requirements. Set \( b(s) := b(s-1) \). Say \( i_j \) is enactable at stage \( s \) if there exists \( n_j \leq s \) such that \( f_i \) is not injective on \( D_j := \Sigma^{n_j-1} \). If none is enactable then skip to the next stage. If so, choose the least such \( j \) and do:

1. Find \( x,y \in D_j \) and \( m \leq n_j \) such that \( f_{i_j}(x) = f_{i_j}(y) \) and \( x_m \neq y_m \).
2. Place \( \Sigma^{m-1} \) into \( A \), adding just those strings of length \( \leq n_j \) which are forced by this and previously-added strings of length \( \leq b(i_j) \). Free all (other) strings of length \( > n_j \).
3. For all \( i', i_j < i' \leq s \), uncancel \( i' \), and reset \( b(i') \) by defining \( b(i') := n_j \). Then proceed to stage \( s+1 \).

This ends the construction. We first claim that for every \( i \), the sequence \( b_k(i) \) giving \( b(i) \) at stage \( s \) is eventually constant (meaning also that \( b_k(i) \) is never even reset to the same value.) The only way \( b_k(i) \) can be reset from \( b_{k-1}(i) \) is if some uncanceled \( R_j \) with \( j < i \) is acted upon. By induction one can show that this can happen at most \( 2^{2-2} \) times, proving the claim.

Second, note that always \( n_j > b(i_j) \) because we must have \( |D_j| \geq 2 \). So \( b(i') > b(i_j) \) for all \( i' > i_j \). Since infinitely-many \( f_i \) are constant, the actions taken to satisfy the corresponding \( R_i \) ensure that \( b_i(i) \rightarrow m \) as \( i,s \rightarrow \infty \). Hence \( A \) is well-defined: given \( x \), there is a least \( i \) such that the final value of \( b(i) \) is \( \geq |x| \). Let \( s \) be the stage at which this final value is reached; then \( x \) belongs to \( A \) iff \( x \in A \) was forced at stage \( s \).

Third, if \( i \) is canceled after the stage \( s \) at which \( b_i(i) \) stabilizes, then \( f_i \) is not a correct reduction from \( A \), since all further action involves only strings in \( \Sigma^* \). If \( f_i \) is uncanceled at at every stage after \( s \), then \( f_i \) is injective on \( \Sigma^n \) for all \( n \geq 0 \), and so its injectivity is \( \Omega(2^n) \).

Note that whether \( i_j \) is enactable at stage \( s \) is decidable, and that the whole algorithm proceeds recursively and deterministically. The reason why \( x \in A \) may not be decidable is that we may have no way of knowing at any stage \( s \) whether \( b_i(i) \) has already stabilized. For some \( j < i \) with \( b_i(j) < |x| \) there may be an \( n > s \) such that \( f_j \) is injective on \( \Sigma^m \) for all \( m < n \) but not for \( m = n \). \( A \) is, however, recursive in \( 0' \) (i.e., in the Halting Problem). If there does exist a recursive bound on \( n \) which is appropriately uniform in \( i \) for all \( f_j \) which are not \( \Sigma^* \)-injective (we leave the exact sense of this unspecified), then the language \( A \) is recursive. We suspect, however, that no such bound exists, and that our particular language \( A \) is actually \( 0' \)-hard under recursion-theoretic many-one reducibility. This may yield (or derive from) a general theorem on the behavior of effective enumerations [\( [f_i]_{i=1}^n \)] of recursive functions.

**Open question:** Can Theorem 6.3 be made to hold with a recursive language \( A \)?

A negative answer would send a warning that diagonalization constructions which produce \( k \)-SLs, such as might be used to countermand our restricted \( p \)-isomorphism conjecture in a suitably "non-collapsing" \( p \)-degree, may not always work on the recursive sets. If the difficulty really owes in general to the closure of \( k \)-SLs under \( \Sigma \), then this may even apply to our quest in Section 4 for a \( k \)-SL \( A \) such that \( R_k \not\in A \), since any family of \( k \)-DNF formulas which one would conceivably try to learn should be recursive.
7. Conclusion And Prospects

We have shown that the finitary substructure languages form an interesting class, with connections to many other concepts studied in the literature of structural complexity. They represent the idea of problems whose difficulty depends on how well a certain finite amount of information is hidden, if the information is present at all. We regard them as somehow "perpendicular" to the notion of a sparse set: every nonempty $k$-SL has exponential density, but only a polynomially-graded information content. They may also be perpendicular to languages of strings having low resource-bounded Kolmogorov complexity, as every $k$-SL $A \neq \emptyset$ contains many random strings, but there may be parallels via the Kolmogorov complexity of $p_A$.

In the notation of [BoKo87], we have shown that for every $k \geq 1$ the class $k$-$SL$ is contained in $P_{atm}(TALLY)$ but not in $P_{atm}(SPARSE)$. Whether every $k$-$SL$ $A$ belongs to $E_2(TALLY)$ (i.e., whether $A \neq T$ for some tally set $T$) relates to the open problems in Section 4 about whether $T_\alpha \not\leq^p A$. These and the problems in Section 5 suggest the following goals, for both of which the indicated strategy seems to be diagonalization:

1. (1) Construct a $k$-$SL$ $A$ such that $p_A \not\leq^p A$.

2. (2) Construct a $T^*SL$ $A \in \Sigma^*$ such that $A$ is not $p$-isomorphic to $A \times \Sigma^*$, where $A$ as well as $\lambda$ has exponential density.

While we do not know of any implication that accomplishing either would solve an open problem known to be very hard, our work in Section 6 suggests that neither will be easy. The contrariwise goals of showing that $p_A$ does reduce to $A$, or that every $*$SL $L$ (subject to minimal conditions of good behavior) is a polynomial cylinder, are also amenable to attack, and offer combinatorics of a different stripe from the familiar methods of analyzing $NP$-complete problems. It may also be possible to moot one goal by constructing an oracle which satisfies its opposite number.

The object of further study which seems likely to hold the most lasting interest concerns how $k$-SLs relate to $NP$-complete languages. Which properties distinguish those known $NP$-complete languages which arise as natural marked unions of $k$-SLs from those which do not? Can one find algorithms for recognizing $k$-SLs in $T$, such as $k$-$ CLIQUE$ for fixed $k$, which use significantly less than $n^k$ time? These take us into questions about concrete decision problems which may lead to new approaches for explaining their apparent difficulty.

References


