$\mathcal{L} \neq \mathcal{NP}$.

J. Andres Montoya
Universidad Nacional de Colombia, Bogota
Centro de Pensamiento Antonio Sanchez de
Cozar, San Gil, Colombia

April 26, 2024

Abstract

We prove that the class LOGSPACE ($\mathcal{L}$, for short) is different from the class $\mathcal{NP}$.

We prove that $\mathcal{L}$ is different from $\mathcal{NP}$. Let us sketch the proof strategy. First, some definitions.

Definition 1 *We use the symbol $\mathcal{NRT}$ to denote the class of quasi real time languages. These are the languages in $\mathcal{NP}$ that are accepted by nondeterministic Turing machines that run in real time [2]. We use the symbol $\mathcal{RT}$ to denote the class of real time languages. This is the class of languages that are accepted by deterministic Turing machines that run in real time.*

The proof of the separation $\mathcal{L} \neq \mathcal{NP}$ is based on the following four facts:

1. The class $\mathcal{L}$ is equal to the union of a strict hierarchy: the *pebble hierarchy.* We discuss this fact in section two. We use the symbol $\mathcal{REG}_n$ to denote the n-th level of this hierarchy. Level $\mathcal{REG}_n$ equals the set of languages that are accepted by deterministic pebble (marker) automata provided with n pebbles, see [6]. The containment $\mathcal{REG}_n \subseteq \mathcal{REG}_{n+1}$ holds.

2. The levels of the pebble hierarchy are closed under *Mealy reductions* (inverse images of generalized syntactic morphisms, see [4]). Or, in order to be more exact:

   Suppose that L belongs to $\mathcal{REG}_n$ and suppose that T is Mealy reducible to L, we have that T belongs to $\mathcal{REG}_{n+1}$ (see Theorem 12).

3. There exists a quasi real-time language L_R that is hardest for $\mathcal{NRT}$ under Mealy reductions [4].

4. The class $\mathcal{NRT}$ *goes high* in the pebble hierarchy. This means that there does exist a sequence included in $\mathcal{NRT}$, say sequence $\{L_n\}_{n\geq 0}$, such that for all $m \geq 0$ there exists n for which the relation $L_n \notin \mathcal{REG}_m$ does hold, see Corollary **??**.

Remark 2 *We say that sequence* $\{L_n\}_{n\geq 0}$ *is high in the pebble hierarchy.*

Let us assume the above four facts. We can easily prove that $\mathcal{NRT}$ is not contained in $\mathcal{L}$. Suppose to the contrary that $\mathcal{NRT}$ is contained in $\mathcal{L}$. *Greibach's hardest quasi real-time language* L_R gets included in some level of the pebble hierarchy, say level k_R. This implies that $\mathcal{NRT}$ is included in level k_R+1. This contradicts the fact that for all $k \geq 1$ there exists a quasi real-time language that does not belong to $\mathcal{RECG}_k$.

It remains to ensure that the above four facts actually hold. Let us discuss those four duties:

1. The first fact is a well established result, see [13], [9] and [10]. We discuss this fact in section 2. We prove that $\mathcal{L}$ is equal to $\bigcup_{k\geq 1} \mathcal{RECG}_k$.

2. The third fact is a fundamental result of Sheila Greibach. She proved that the grammar class $\mathcal{CFL}$ and the class $\mathcal{NRT}$ each contain languages that are hardest under Mealy reductions see [4].

   Notation 3 *We use the symbol* $\mathcal{CFL}$ *to denote the set of context-free languages.*

3. We prove that the pebble hierarchy is invariant under Mealy reductions, see Theorem 12.

4. The highness of $\mathcal{NRT}$ could be obtained as an easy corollary of Theorem 1, page 76, in reference [6]. This theorem asserts that a certain sequence of context-free languages, say sequence $\{H_n\}_{n\geq 1}$, is high in the pebble hierarchy. This assertion implies that $\mathcal{CFL}$ (and hence $\mathcal{NRT}$) is high in the pebble hierarchy, and it also implies the strong separation $\mathcal{L} \neq \mathcal{P}$. However, this theorem is false and its proof is wrong, see [11]. We prove that $\mathcal{NRT}$ is high in the pebble hierarchy, see Theorem 19.

Organization of the work and contributions. This work is organized into three sections besides this introduction. In section one we introduce Mealy machines, Mealy reductions and we discuss Greibach's argument. In section two we study pebble automata and the related pebble hierarchy. In section three we outline the proof of the main result of this paper, namely that $\mathcal{L}$ is different from $\mathcal{NP}$.

1 Mealy reductions and Complete problems

We study separations between complexity classes. We use a very weak type of algorithmic reduction that comes from the theory of formal languages. We refer to *Mealy reductions*.

Definition 4 *A Mealy machine is a tuple*

$$\mathcal{M} = (Q, q_0, \Sigma, \Gamma, \delta, \lambda)$$

such that:

1. Q *is a finite set of states.*
2. $q_0 \in Q$ *is the initial state.*
3. Σ *is the input alphabet.*
4. Γ *is the output alphabet.*
5. $\delta : Q \times \Sigma \to Q$ *is the transition function, which maps a state and an input symbol to a state.*
6. $\lambda : Q \times \Sigma \to \Gamma^*$ *is the output function, which maps a state and an input symbol to an output string.*

The output of a Mealy machine $\mathcal{M}$, on input $w = w_1 \cdots w_n$, is denoted with the symbol $O_{\mathcal{M}}(w)$. This output is defined recursively as follows:

- For all $q \in Q$ and for all $a \in \Sigma$ the equality $\widehat{\lambda}(q, a) = \lambda(q, a)$ holds.
- For all $w \in \Sigma^*$ the equality

  $$\widehat{\lambda}(q, w) = \lambda(q, w[1])\, \widehat{\lambda}(\delta(q, w[1]), w[2, ..., n])$$

  holds.
- $O_{\mathcal{M}}(w) = \widehat{\lambda}(q_0, w)$.

Definition 5 *Let $L \subset \Sigma^*$ and $T \subset \Gamma^*$ be two languages. We say that L is Mealy-reducible to T if and only if there exists a Mealy machine $\mathcal{M}$ such that $L = O_{\mathcal{M}}^{-1}(T)$. We use the symbol $L \preceq_M T$ to indicate that L is Mealy reducible to T.*

Let us introduce the corresponding notion of complete (hardest) problem.

Definition 6 *Let $\mathcal{C}$ be a set of languages and let L be a language in $\mathcal{C}$. We say that L is $\mathcal{C}$-hardest (or $\mathcal{C}$-complete) if and only if any language in $\mathcal{C}$ is Mealy-reducible to L.*

Greibach proved two fundamental results [4], namely:

1. There exist context-free languages that are $\mathcal{CFL}$-complete.
2. There exist quasi-real time languages that are $\mathcal{NRT}$ -complete.

Let us introduce some related terminology.

Definition 7 *Let $\mathcal{C}$ be a set of languages.*

1. *We say that $\mathcal{C}$ is a complexity class if and only if $\mathcal{C}$ is closed under Mealy reductions.*
2. *Suppose that $\mathcal{C}$ is a complexity class. We say that $\mathcal{C}$ is principal if and only if there exists $L \in \mathcal{C}$ such that L is $\mathcal{C}$-complete.*
3. *Suppose that $\mathcal{C}$ is equal to $\bigcup_{i\geq 1} \mathcal{C}_i$. We say that $\bigcup_{i\geq 1} \mathcal{C}_i$ is an infinite hierarchy if and only if the following conditions hold:*
   - *For all i there exists $j > i$ such that $\mathcal{C}_i \subset \mathcal{C}_j$.*
   - *There exists a constant d such that given given $T \in \mathcal{C}_i$ and given $L \preceq_M T$ the language L belongs to $\mathcal{C}_{i+d}$.*
4. *We say that class $\mathcal{C}$ is stratified if and only if $\mathcal{C}$ is equal to the union of an infinite hierarchy.*
5. *Let $\mathcal{C} = \bigcup_{i\geq 1} \mathcal{C}_i$ be a stratified class and let $\mathcal{D}$ be a complexity class. We say that $\mathcal{D}$ is high in $\bigcup_{i\geq 1} \mathcal{C}_i$ if and only if for all $n \geq 1$ there exists $L \in \mathcal{D} - \mathcal{C}_n$.*

We have

Theorem 8 *Let $\mathcal{C}$,$\mathcal{D}$ be two complexity classes. Suppose that $\mathcal{C} = \bigcup_{i\geq 1} \mathcal{C}_i$ is stratified, suppose that $\mathcal{D}$ is principal and suppose that $\mathcal{D}$ is high in $\mathcal{C}$. We get that $\mathcal{D}$ is not contained in $\mathcal{C}$.*

Proof. Suppose that $\mathcal{D}$ is contained in $\mathcal{C}$. Let L_0 be $\mathcal{D}$-complete. Let k be a positive integer such that $L_0 \in \mathcal{C}_k$. We get that there exists d such that $\mathcal{D}$ is contained in $\mathcal{C}_{k+d}$. This contradicts the fact that $\mathcal{D}$ is high in $\mathcal{C}$. The theorem is proved. ■

We can use the previous theorem, that we call *Greibach's argument*, to prove separations between complexity classes. We use this argument to prove that $\mathcal{L}$ is different from $\mathcal{NP}$. To this end we prove that $\mathcal{L}$ is stratified and we prove that $\mathcal{NRT}$ is high in $\mathcal{L}$. We get that $\mathcal{NRT}$ is not contained in $\mathcal{L}$ since the former class is principal, see [4]. Note that $\mathcal{NRT}$ is contained in $\mathcal{NP}$. We obtain as a corollary the separation $\mathcal{L} \neq \mathcal{NP}$.

2 The Pebble Hierarchy

Pebble automata are two-way automata provided with a bounded amount of pebbles, each of which can be distinguished from one another. Let us introduce the formal definition that we use in this work.

Definition 9 *Let $k \geq 0$, a deterministic k-pebble automaton is a tuple*

$$\mathcal{M} = (Q, q_0, \Sigma, H, A, \delta)$$

such that:

1. *Q is a finite set of states.*
2. *$q_0 \in Q$ is the initial state.*
3. *$H \subset Q$ is the set of halting states.*
4. *$A \subset H$ is the set of accepting states.*
5. *Σ is the input alphabet.*
6. *The transition function*

$$\delta : Q \times \Sigma \times \left(\mathcal{P}\left(\{0, ..., k\}\right)\right)^2 \to Q \times \left(\mathcal{P}\left(\{0, ..., k\}\right)\right)^2 \times \{-1, 0, 1\}$$

is deterministic.

Let $\mathcal{M}$ be a k-pebble automaton. Automaton $\mathcal{M}$ is a two-way deterministic finite state automaton provided with k pebbles. This automaton has the following capabilities:

- *It can place any one of its pebbles on the tape.*
- *It can sense the pebbles that lie on the current cell.*
- *It can pick specific pebbles from this cell when required.*

Let (q, a, A, B) be a tuple such that A, B are disjoint subsets of $\{1, ..., k\}$. Suppose that $\mathcal{M}$ is processing the input w. Suppose that:

- *q is the the inner state reached by $\mathcal{M}$ at time t.*
- *a is the character being scanned.*
- *A is the set of pebbles that are placed on the current cell.*
- *B is the set of available pebbles.*

Suppose that $\delta(q, a, A, B) = (p, C, D, \varepsilon)$. We get that $A \cup B = C \cup D$, and we get that $\mathcal{M}$ has to change its inner state from q to p, it has to place the pebbles belonging to C (and only these pebbles) on the current cell, and it has to move its input head in the direction indicated by ε.

We use the symbol $\mathcal{RECG}_k$ to denote the set of languages that are accepted by deterministic automata provided with k pebbles. We have.

Theorem 10 *The equality $\mathcal{L} = \bigcup_{k \geq 1} \mathcal{RECG}_k$ holds.*

Proof. Notice that a single pebble can be easily tracked using a binary tape of logarithmic size. This means that given a k-pebble automaton $\mathcal{M}$, one can construct a Turing machine $\mathcal{N}$ provided with k work tapes of logarithmic size and which simulates $\mathcal{M}$. We get that $\bigcup_{k\geq 1} \mathcal{REG}_k$ is included in $\mathcal{L}$.

Let us prove that $\mathcal{L}$ is included in $\bigcup_{k\geq 1} \mathcal{REG}_k$. Let L be a language in $\mathcal{L}$. There exists $k \geq 0$ and there exists a Turing machine $\mathcal{M}$ such that:

- $\mathcal{M}$ accepts L.
- $\mathcal{M}$ is provided with a read-only input tape and k binary tapes of logarithmic size.

Let us prove that each one of those binary tapes can be simulated using three pebbles. Let $w \in \Sigma^n$ be the input of $\mathcal{M}$. Let us focus on the i-th tape and let us consider the configuration reached by this tape at instant t. We represent this configuration as a pair $(1n_L, n_R1) \in \left(\{0,1\}^*\right)^2$. This pair tells us that n_Ln_R is the work tape content, and it also tells us that the head assigned to this tape is located on cell $|n_L|$. Let $(n_R1)^R$ be the reverse of n_R1. Note that $1n_L$ and $(n_R1)^R$ are binary strings whose lengths are bounded above by $\log(n)$. These binary strings encode two positive integers m_L and m_R that belong to the interval $\{1, ..., n\}$. We use this to represent configuration $(1n_L, n_R1)$ using pebbles. To this end we place one pebble on the m_L-th cell of the input tape, and we place a second pebble on the m_R-th cell of this tape. We can use these pebbles to simulate the changes that occur on tape i. Let us suppose, for instance, that the transition function of $\mathcal{M}$ forces the head (of tape i) to move one cell to the left, after replacing with 0 the character 1 that was written on the current cell (which is cell $|n_L|$). Let $(1n_L^*, n_R^*1)$ be the configuration reached at time $t+1$. Let m_L^* and m_R^* be the corresponding integers. Notice that

$$m_L^* = \frac{m_L - 1}{2} \text{ and } m_R^* = 2m_R$$

This means that we have to move the aforementioned two pebbles to the cells $\frac{m_L-1}{2}$ and $2m_R$. This can be done with the help of a third pebble. This means that we can simulate this transition using three pebbles. Notice that we can simulate any other transition of the i-th tape using the same three pebbles. This means that we can replace tape i with three pebbles. This also means that we can replace all the k tapes of $\mathcal{M}$ with $3k+1$ pebbles (observe that $2k+2$ is enough). If we do the latter we obtain a $(3k+1)$-pebble automaton that accepts L. We get that $L \in \mathcal{REG}_{3k+1}$. We conclude that $\mathcal{L}$ is included in $\bigcup_{k\geq 1} \mathcal{REG}_k$. The theorem is proved. ■

We use the term pebble hierarchy to designate the hierarchy

$$\mathcal{REG}_1 \subseteq \mathcal{REG}_2 \subseteq \mathcal{REG}_3 \subseteq \cdots$$

It can be proved that this hierarchy is strict. We owe the first proof attempt to Hsia and Yeh, see [6]. Unfortunately their proof is incorrect. Hsia and Yeh construct a sequence of context-free languages, say $\{H_n\}_{n\geq 1}$, and they present an incorrect proof that this sequence is high in the pebble hierarchy [6]. If Hsia-Yeh's claim were correct we would obtain as a corollary that $\mathcal{L}$ is different from $\mathcal{P}$. This follows easily from Greibach's argument:

Suppose that $\{H_n\}_{n\geq 1}$ is high in the pebble hierarchy. The class $\mathcal{CFL}$ is principal. We get that $\mathcal{CFL}$ is not contained $\mathcal{L}$. We have, on the other hand, that $\mathcal{CFL}$ is contained in $\mathcal{P}$.

Let us show that the pebble hierarchy has infinite many levels. Monien proved that the *head hierarchy* is strict, see Theorem 3, page 81 in reference [9]. This means that there exists a sequence of languages, say $\{M_n\}_{n\geq 1}$, such that M_n is accepted by a multihead automaton provided with n heads but such that this same language cannot be accepted with $n-1$ heads. Let us observe that the *head hierarchy* and the pebble hierarchy are interleaved: k heads can be simulated with k pebbles, while k pebbles can be simulated with $k+1$ heads, see [10]. Let $k \geq 2$, we get that language M_k belongs to $\mathcal{RECG}_k - \mathcal{RECG}_{k-2}$.

Remark 11 *The sequence $\{M_k\}_{k\geq 2}$ is not contained in $\mathcal{CFL}$.*

It can be proved that the pebble hierarchy is invariant under Mealy reductions.

Theorem 12 *The pebble hierarchy is invariant under Mealy reductions.*

Proof. Let us prove that L can be accepted with $k+1$ pebbles.

Let $\mathcal{M}$ be a k-pebble automaton that accepts $T \subset \Gamma^*$, and let

$$\mathcal{N} = (Q, q_0, \Sigma, \Gamma, \delta, \lambda)$$

be a Mealy machine that reduces L in T. We construct a $(k+1)$-pebble automaton $\mathcal{S}$ that receives as input $w \in \Sigma^*$ and simulates the computation of $\mathcal{M}$ on input $O_{\mathcal{N}}(w)$. Notice that $O_{\mathcal{N}}(w)$ is equal to the concatenation of the strings

$$\lambda(q_0, w[1]), \lambda(\delta(q_0, w[1]), w[2]), ..., \lambda\left(\widehat{\delta}(q_0, w[1, ..., |w|-1]), w[|w|]\right)$$

Let

$$N = \max\{|\lambda(q,a)| : q \in Q \text{ and } a \in \Sigma\}$$

The simulation works as follows:

Suppose that $\mathcal{M}$ is in state q, the input head is located on the r-th character of the factor

$$\lambda\left(\widehat{\delta}(q_0, w[1, ..., i-1]), w[i]\right),$$

and $B \subset \{1, ..., k\}$ is the set of available pebbles. Suppose also that the simulation has worked well up this point. The latter means that:

1. $\mathcal{S}$ keeps in its inner memory the state $\widehat{\delta}\left(q_0, w\left[1, ..., i\right]\right)$ and the integer $r \leq N$. Remember that r denotes the position of the input head of $\mathcal{M}$ within the factor
$$\lambda\left(\widehat{\delta}\left(q_0, w\left[1, ..., i-1\right]\right), w\left[i\right]\right)$$

2. The head of $\mathcal{S}$ is located on cell i.

3. $B \cup \{k+1\}$ is the set of available pebbles for $\mathcal{S}$.

4. Let $p \in \{1, ..., k\}$, let $l \in \{1, ..., N\}$ and suppose that p-th pebble of $\mathcal{M}$ is located on the l-th character of the factor
$$\lambda\left(\widehat{\delta}\left(q_0, w\left[1, ..., j-1\right]\right), w\left[j\right]\right)$$
Then, the p-th pebble of $\mathcal{S}$ is located on cell j, and the pair (p, l) is kept safe in the finite state memory of $\mathcal{S}$. This pair indicates to $\mathcal{S}$ that the p-th pebble is on the l-th character of the factor where it is currently located.

Automaton $\mathcal{S}$ can use its finite state memory to compute:

- The string
$$\lambda\left(\widehat{\delta}\left(q_0, w\left[1, ..., i-1\right]\right), w\left[i\right]\right)$$
- The location of the input head of $\mathcal{M}$ within this short string. Notice that the length of this string is not greater than N.
- The configuration of pebbles that lie within this factor.

Automaton $\mathcal{S}$ can use its finite state memory to simulate the computation of $\mathcal{M}$ from this point until the following condition is met:

Either the input head of $\mathcal{M}$ leaves the factor going to the right, or it leaves the factor going to left or it halts within the factor.

Suppose that $\mathcal{M}$ leaves the factor going to the right. In this case automaton $\mathcal{S}$ moves its input head one position to the right and updates its inner memory. Now suppose that $\mathcal{M}$ has to leave the factor going to the left. Automaton $\mathcal{S}$ has to use, in this case, pebble $k+1$. This is so because $\mathcal{S}$ has to reconstruct the value of
$$\lambda\left(\widehat{\delta_{\mathcal{N}}}\left(q_0, w\left[1, ..., i-2\right]\right), w\left[i-1\right]\right)$$
Let us see how the simulation proceeds in this case:

- Automaton $\mathcal{S}$ places pebble $k+1$ on cell $i-1$.
- $\mathcal{S}$ moves its input head until the left end of the tape.
- $\mathcal{S}$ begins to move rightward searching for pebble $k+1$ while simultaneously computing the transduction λ.

- $\mathcal{S}$ computes
$$\lambda\left(\widehat{\delta_{\mathcal{N}}}\left(q_0, w\left[1, ..., i-2\right]\right), w\left[i-1\right]\right)$$
and updates its inner memory.

The theorem is proved. ■
We get the following corollary.

Corollary 13 *The class $\mathcal{L}$ is stratified.*

3 $\mathcal{L}$ is different from $\mathcal{NP}$

We want to use Greibach's argument to prove the separation $\mathcal{L} \neq \mathcal{NP}$. We know that $\mathcal{NRT}$ is principal and we know that $\mathcal{L}$ is stratified. It only remains to prove tat $\mathcal{NRT}$ is high in the pebble hierarchy. We prove this in this section.

Definition 14 *Let $w_1, ..., w_n$ be n strings in Σ^* and let $\#$ be a fresh character not in Σ. We use the symbol $s(w)$ to denote the string $w_1\# \cdots \# w_n$. We say that $w_1\# \cdots \# w_n$ is a sequence over the alphabet Σ and we say that the strings $w_1, ..., w_n$ are the factors of $s(w)$. We assign to sequence $s(w)$ the binary string*

$$\varepsilon^L_{s(w)} = \varepsilon^L(w_1) \cdots \varepsilon^L(w_n),$$

where given $u \in \Sigma^*$ the bit $\varepsilon^L(u)$ is equal to 1 if and only if $u \in L$.

Definition 15 *Let $\{0,1\} \subset \Sigma$ and let $u \in \Sigma^*$. We use the symbol $\|u\|$ to denote the Hamming weight of u, i.e.*

$$\|u\| = |\{i \leq n : u[i] = 1\}|$$

We define $Ham(L)$ as the language

$$Ham(L) = \left\{ s(w) \#\#^T 0^{|s(w)|-T} : \sum_{i \leq n} \varepsilon^L(w_i) \|w_i\| \neq T \right\}$$

Remark 16 *Let $s(w) \#\#^T 0^K$ be an instance of $Ham(L)$. We assume, without loss of generality, that the equality $K = |s(w)| - T$ holds. We emphasize this assumption using the notation $s(w) \#\#^T 0^{K_T}$*

We have.

Proposition 17 *Let $L \in \mathcal{RT}$, the language $Ham(L)$ also belongs to $\mathcal{RT}$.*

Proof. Let L be a language in $\mathcal{RT}$ and let $\mathcal{M}$ be a deterministic real time machine that accepts L. Let $\mathcal{N}$ be the deterministic real time machine that works, on input $s(w) \#\#^T 0^{K_T}$, as follows.

1. Let us suppose that the equality

$$s(w) = w_1\# \cdots \# w_n$$

holds. In the first phase of the computation the machine $\mathcal{N}$ scans the prefix $s(w)$ while simulating, in real time, the computations of $\mathcal{M}$, on each one of the strings $w_1, ..., w_n$. When the input head of $\mathcal{N}$ reaches the right end of $w_i\#$, machine $\mathcal{N}$ knows whether the string w_i belongs to L. Depending on this machine $\mathcal{N}$ marks the current cell, which is precisely the right end of $w_i\#$.

2. The first phase ends when $\mathcal{N}$ scans for the first time the pattern $\#\#$. When this occurs machine $\mathcal{N}$ has read $s(w)$ and it has marked the right end of the factors in $s(w)$ that belong to L. The second phase begins. $\mathcal{N}$ has to compare two quantities, namely: the total number of 1's that occur in the marked factors that lie on the left of the current cell, and the number of #'s that lie on the right of this cell. Notice that the current cell is the central location of the input string. This latter fact is ensured by the equality $|s(w)| = T + K_T$. This fact ensures that the remaining computation can be done in real time. To do this we use two additional heads. The first of those heads moves leftward looking for the marked 1's. The second head moves rightward, over the block $\#^T$, counting the marked 1's that are scanned by its companion head. The computation ends when the first of these heads reaches the left end of the input string and, simultaneously, the input head of $\mathcal{N}$ reaches the right end of the input string. This implies that $\mathcal{N}$ is a real time machine.

We can use the real time machine $\mathcal{N}$ to accept the language $Ham(L)$, the proposition is proved. ■

Let us introduce a sequence of quasi-real time languages that goes high in the pebble hierarchy.

Definition 18 *Let $\{L_k\}_{k\geq 0}$ be the sequence of quasi real-time languages that is defined as follows:*

- $L_0 = EQ$, *where*

$$EQ = \left\{1^i 0^j 1^i : i, j \geq 1\right\}$$

Note that EQ belongs to $\mathcal{RT}$.

- *Let $k \geq 0$. Set $\Sigma_k = \{0, 1, \#_1, ..., \#_k\}$. Suppose $L_k \subset \{0, 1, \#_1, ..., \#_k\}^*$. Let $\#_{k+1}$ be a fresh symbol not in Σ_k. Set $L_{k+1} = Ham(L_k)$.*

Note that $\{L_n\}_{n\geq 0}$ is included in $\mathcal{RT}$. This follows from the previous proposition. Moreover, we have.

Theorem 19 *Sequence $\{L_n\}_{n\geq 1}$ is high in the pebble hierarchy.*

Remark 20 *The above theorem implies that $\mathcal{RT}$ is high in $\mathcal{L}$. This does not imply that $\mathcal{RT}$ is not contained in $\mathcal{L}$. It happens since $\mathcal{RT}$ does not have complete problems. This is consequence of the class $\mathcal{RT}$ being stratified, see [1].*

3.1 Proving that Sequence $\{L_n\}_{n\geq 1}$ is High

We discuss, in the remainder of this extended abstract, the proof of Theorem 19.

Let $\mathcal{M}$ be a k-pebble automaton that accepts L. Let $w \in \Sigma^l$ be an input of $\mathcal{M}$. The *configurations* that are accessed by $\mathcal{M}$, on input w, are $(k+2)$-tuples in the set $Q \times \{0,1,...,l\}^{k+1}$. Let

$$\mathcal{D}_t = (q, m, p_1, ..., p_k)$$

be one of those tuples, say the configuration reached by $\mathcal{M}$ at instant t. We have:

- The symbol q denotes the inner state of the machine at instant t.
- The small positive integers $m, p_1, ..., p_k$ denote the locations, at time t, of the input head and the k pebbles of $\mathcal{M}$. Suppose that the i-th pebble is available. We assume that the equality $p_i = m$ holds, (we assume that the input head carries with it the available pebbles).

Definition 21 *Let $\mathcal{D}_t$ be as above. Let $\mathcal{C}_t = (p_1, ..., p_k)$. We say that $\mathcal{C}_t$ is the pebble configuration reached by $\mathcal{M}$ at instant t.*

Definition 22 *Let X be a random variable distributed over the set $\{1, ..., l\}$. The Shannon entropy of X, (or just the entropy of X), is defined as*

$$H(X) = -\sum_{i \leq l} \Pr[X = i] \log(\Pr[X = i])$$

The maximum entropy is achieved by the random variables that are uniformly distributed [12]. Thus, given $l \geq 0$ and given X_l, a random variable that is distributed over $\{1, ..., l\}$, the inequality $H(X_l) \leq \log(l)$ holds. Let us suppose that we are confronted with a random variable X that is distributed over an unknown set A. Suppose that there exists $\gamma < \frac{1}{2k}$ such that the inequality

$$H(X) > (1 - \gamma) k \log(l)$$

holds. We can conclude that the inequality $|A| \geq l^{k - \frac{1}{2}}$ holds. We can use this elementary fact as a method for proving lower bounds. We call this method *The Entropy Method* [8].

Example 23 *Let $\mathcal{M}$ be a pebble automaton and let $p(\mathcal{M})$ be the number of its pebbles. Let $l > 0$ and let $A_l \subset \{0, ..., l\}^{p(\mathcal{M})}$ be the set of pebble configurations that are visited by $\mathcal{M}$, at least once, when this automaton processes inputs of length l. Let X_l be a random variable that is distributed over the set A_l. Suppose that there exists $\gamma < \frac{1}{2k}$ such that the inequality*

$$H(X_l) > (1 - \gamma) k \log(l)$$

holds for all l large enough. We obtain that $p(\mathcal{M})$ is greater than $k - 1$.

Definition 24 *Let $\mathcal{G}$ be a pebble automaton that accepts the language $L \subset \Sigma^*$. Let $\mathcal{S} \subset \Sigma^*$ be an infinite set. Let $l > 0$, suppose $\mathcal{S} \cap \Sigma^l \neq \emptyset$ and let $X_{\mathcal{G}}(\mathcal{S}, l)$ be the random variable that is defined as follows:*

1. *Choose uniformly at random $w \in \mathcal{S} \cap \Sigma^l$.*
2. *Choose uniformly at random one of the pebble configurations visited by $\mathcal{G}$ during the processing of w.*
3. *Let $\mathcal{C}$ be the configuration chosen in the previous step. Set $X_{\mathcal{G}}(\mathcal{S}, l) = \mathcal{C}$.*

Definition 25 *Let $X_{\mathcal{G}}(\mathcal{S}, l)$ be as above. We say that $H(X_{\mathcal{G}}(\mathcal{S}, l))$ is the entropy of $\mathcal{G}$ over the set $\mathcal{S} \cap \Sigma^l$.*

Next proposition is an easy application of The Entropy Method.

Proposition 26 *Suppose that there exists an infinite set $\mathcal{S} \subset \Sigma^*$ such that for all $\mathcal{G}$ that accepts L and for all $\gamma > 0$ the inequality*

$$H(X_{\mathcal{G}}(\mathcal{S}, l)) > (1 - \gamma) k \log(l)$$

holds asymptotically. Language L cannot be accepted with $k - 1$ pebbles.

Proof. Suppose to the contrary that there exists a $(k-1)$-pebble automaton $\mathcal{H}$ that accepts L. We have that for all infinite set $\mathcal{S} \subseteq \Sigma^*$ and for all l the inequality

$$H(X_{\mathcal{H}}(\mathcal{S}, l)) \leq (k - 1) \log(l)$$

holds. We have, on the other hand, that there exists a positive integer $K_{\mathcal{H}}$ such that for all $l \geq K_{\mathcal{H}}$ the inequality

$$H(X_{\mathcal{H}}(\mathcal{S}, l)) > \left(1 - \frac{1}{2k+1}\right) k \log(l) > \left(k - \frac{1}{2}\right) \log(l)$$

holds. We get a contradiction and the proposition is proved. ■

We use this proposition in the proof of Theorem 19. It remains to prove that for all $k \geq 0$ there exists a set $\mathcal{S}_k \subset \Sigma_k^*$ that behaves, with respect to the language L_k, exactly as in the statement of the previous theorem.

Definition 27 *Let $k \geq 0$ and let $\mathcal{S}_k \subset \Sigma_k^*$ be an infinite set. Let $l \geq 1$ and let X_l be a random variable that is uniformly distributed over the set $\mathcal{S}_k \cap \Sigma_k^l \neq \emptyset$. We say that $\mathcal{S}_k$ is a H-set for L_k if and only if for all $\gamma > 0$ the inequality*

$$H(\|X_l\|) > (1 - \gamma) \log(l)$$

holds asymptotically.

Notation 28 *Let $\mathcal{S}_k$ be a H-set for L_k and let $l > 0$. We use the symbol $\mathcal{S}_k(l)$ to denote the set $\mathcal{S}_k \cap \Sigma_k^l$.*

Let L_k^* be the language

$$\left\{\varepsilon_{s(w)}^{L_k}\#_{k+1}w_1\#_{k+1}\cdots\#_{k+1}w_n\#_{k+1}u : u \notin L_k\right\}$$

Let us suppose that $\mathcal{S}_k$ is a H-set for L_k. Let $r, l \geq 1$ and let $\mathcal{S}_{k,r,l}^*$ be the set

$$\left\{ \begin{array}{c} \varepsilon_{s(w)}^{L_k}\#_{k+1}w_1\#_{k+1}\cdots\#_{k+1}w_n\#_{k+1}u : n = \log^r(l) - 1 \text{ and} \\ w_1, ..., w_n, u \in \mathcal{S}_k(l) \end{array} \right\}$$

Remark 29 *Recall that $\varepsilon_{s(w)}^{L_k}$ denotes the L_k-characteristic function of the sequence*

$$s(w) = w_1\#_{k+1}\cdots\#_{k+1}w_n$$

Definition 30 *Let $\mathcal{G}^*$ be a pebble automaton. We say that $\mathcal{G}^*$ is a promise automaton for the pair $\left(L_k^*, \bigcup_{l\geq 1}\mathcal{S}_{k,r,l}^*\right)$ if and only if the following two conditions hold:*

1. *$\mathcal{G}^*$ accepts the strings in $\left(\bigcup_{l\geq 1}\mathcal{S}_{k,r,l}^*\right) \cap L_k^*$.*
2. *$\mathcal{G}^*$ rejects the strings in $\left(\bigcup_{l\geq 1}\mathcal{S}_{k,r,l}^*\right) \cap$ co-L_k^*, where co-L_k^* denotes the complement of L_k^*.*

Notation 31 *We use the symbol $\mathcal{S}_{k,r}^*$ to denote the set $\bigcup_{l\geq 1}\mathcal{S}_{k,r,l}^*$. We use the symbol K_l to denote the length of the strings that belong to $\mathcal{S}_{k,r,l}^*$. Note that K_l equals $(l+1)\log^r(l) + \log^r(l) - 1$. We use the symbol $X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right)$ to denote the random variable $X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r}^*, K_l\right)$.*

Proposition 32 *Let $\mathcal{G}$ be a pebble automaton that accepts L_k. Let $r \geq 0$. There exists a promise automaton $\mathcal{G}^*$ for the pair $\left(L_k, \mathcal{S}_{k,r}^*\right)$ such that the inequality*

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right)\right) \leq H\left(X_{\mathcal{G}}\left(\mathcal{S}_k, l\right)\right)$$

holds.

Proof. Let $\mathcal{G}^*$ be the pebble automaton that simulates, on input

$$\varepsilon_1\cdots\varepsilon_n\#_{k+1}w_1\#_{k+1}\cdots\#_{k+1}w_n\#_{k+1}u,$$

the computation of $\mathcal{G}$, on input u. Automaton $\mathcal{G}^*$ is a promise automaton for the pair $\left(L_k^*, \mathcal{S}_{k,r}^*\right)$ and the inequality

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right)\right) \leq H\left(X_{\mathcal{G}}\left(\mathcal{S}_k, l\right)\right)$$

holds. ■

Remark 33 *We want to show that the automata that accept L_k have high entropy over the set $\mathcal{S}_k$. It suffices if we prove that there exists $r \geq 0$ such that the promise automata for the pair $\left(L_k^*, \mathcal{S}_{k,r}^*\right)$ all have high entropy over the set $\mathcal{S}_{k,r}^*$.*

Let $\mathcal{G}^*$ be a promise automaton for the pair $\left(L_k^*, \mathcal{S}_{k,r}^*\right)$. Let $l \geq 2$ and let

$$W_l = \varepsilon_1 \cdots \varepsilon_n \#_{k+1} w_1 \#_{k+1} \cdots \#_{k+1} w_n \#_{k+1} u$$

be an input of $\mathcal{G}^*$. Let us suppose $n = \log^r(l) - 1$ and let us suppose that W_l is uniformly distributed over the set $\mathcal{S}_{k,r,l}^*$. Let us assume, against our own interest, that $\mathcal{G}^*$ has preprocessing capabilities that help it to reduce the number of pebble configurations that are visited during the processing of u. Let us assume that $\mathcal{G}^*$ can process the honest prefix

$$\varepsilon_1 \cdots \varepsilon_n \#_{k+1} w_1 \#_{k+1} \cdots \#_{k+1} w_n,$$

and compute some advice using zero pebbles for this purpose. What is the advice (statistics) that $\mathcal{G}^*$ is allowed to compute in this preprocessing phase? We let $\mathcal{G}^*$ to run, with zero cost, a *mining algorithm $\mathcal{M}$ for the language L_k^*.*

Definition 34 *Let W_l be as above and let*

$$s(W_l) = w_1 \#_{k+1} \cdots \#_{k+1} w_n$$

A mining algorithm for L_k^* is an algorithm $\mathcal{M}$ that has restricted access to W_l. Algorithm $\mathcal{M}$ has access to the sequence $s(W_l)$ and computes, on this input string, sets

$$S_{\mathcal{M}_1}(W_l), ..., S_{\mathcal{M}_{t_\mathcal{M}}}(W_l) \subseteq \{1, ..., n\}$$

whose number (i.e. the quantity $t_\mathcal{M}$) does not depend on $s(W_l)$. The output of $\mathcal{M}$, on input $s(W_l)$, is the tuple

$$\left(S_{\mathcal{M}_1}(W_l), ..., S_{\mathcal{M}_{t_\mathcal{M}}}(W_l), \Phi_1^\mathcal{M}(W_l), ..., \Phi_{t_\mathcal{M}}^\mathcal{M}(W_l)\right)$$

where given $j \leq t_\mathcal{M}$ the symbol $\Phi_j^\mathcal{M}(W_l)$ denotes the sum

$$\sum_{j \in S_{\mathcal{M}_j}} \|w_j\|$$

We use the symbol $\mathcal{M}(W_l)$ to denote the output of $\mathcal{M}$.

Remark 35 *Note that a mining algorithm like $\mathcal{M}$ fulfills two constraints, namely: $\mathcal{M}$ does not have access to u, the restricted form of the output.*

We assume, against our own interest, that:

1. $\mathcal{G}^*$ can run a (optimal) mining algorithm for L_k^* using zero pebbles for this purpose. We use the symbol $\mathcal{M}$ to denote this algorithm. Algorithm $\mathcal{M}$ computes (the best possible) advice from $s(W_l)$.

2. $\mathcal{G}^*$ can save the tuple

$$\left(S_{\mathcal{M}_1}(W_l), ..., S_{\mathcal{M}_{t_{\mathcal{M}}}}(W_l), \Phi_1^{\mathcal{M}}(W_l), ..., \Phi_{t_{\mathcal{M}}}^{\mathcal{M}}(W_l)\right)$$

using zero pebbles for this purpose. We use the symbol $\mathcal{M}(W_l)$ to denote this tuple.

3. $\mathcal{G}^*$ can use $\mathcal{M}(W_l)$ as an oracle in order to reduce the entropy that occurs during the processing of u. Notice that $\mathcal{M}(W_l)$ is a random variable distributed over the set

$$\mathcal{P}(\{1, ..., n\})^{t_{\mathcal{M}}} \times \{0, ..., nl\}^{t_{\mathcal{M}}}$$

We encode the above three assumptions by focusing on the conditional entropy

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right) \mid \mathcal{M}(W_l)\right),$$

see reference [3].

Definition 36 *Let $\mathcal{S}_k$ be a H-set for L_k. We say that $\mathcal{S}_k$ is a high entropy set for L_k if and only if for all $r > 0$, for all promise automaton $\mathcal{G}^*$ for the pair $\left(L_k^*, \mathcal{S}_{k,r}^*\right)$, for all mining algorithm $\mathcal{M}$ for L_k^* and for all $\gamma > 0$ the inequality*

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right) \mid \mathcal{M}(W_l)\right) \geq (1-\gamma) k \log(l)$$

holds asymptotically.

We have.

Theorem 37 *Suppose that $\mathcal{S}_k$ is a high entropy set for language L_k. Then, for all pebble automaton $\mathcal{G}$ that accepts L_k and for all $\gamma > 0$ the inequality*

$$H(X_{\mathcal{G}}(\mathcal{S}_k, l)) \geq (1-\gamma) k \log(l)$$

holds asymptotically.

Proof. Let $\mathcal{G}$ be an automaton that accepts the language L_k and let $r \geq 1$. There exists a promise automaton $\mathcal{G}^*$ for the pair $\left(L_k^r, \mathcal{S}_{k,r}^*\right)$ and such that for all $l \geq 1$ the inequality

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right)\right) \leq H(X_{\mathcal{G}}(\mathcal{S}_k, l))$$

holds. Let $\mathcal{M}$ be a mining algorithm for L_k^*. Conditioning a random variable can only reduce its entropy. We get that for all $\gamma > 0$ the inequalities

$$\begin{aligned}(1-\gamma) k \log(l) &\leq H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right) \mid \mathcal{M}(W_l)\right) \\ &\leq H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k,r,l}^*\right)\right) \leq H(X_{\mathcal{G}}(\mathcal{S}_k, l))\end{aligned}$$

hold, the first of these inequalities holds asymptotically. The theorem is proved. ■

Let us construct a sequence $\{\mathcal{S}_k \subset \Sigma_k^* : k \geq 0\}$ such that $\mathcal{S}_k$ is a high entropy set for L_k. We proceed by induction. The construction goes as follows.

1. Let us first consider the case $k = 0$. Set $\mathcal{S}_0^+$ equal to EQ and set

$$\mathcal{S}_0^- = \left\{1^{i-1}0^{l-2i}1^{i+1} : l \geq 1 \text{ and } 2i < l\right\}$$

   Then, set $\mathcal{S}_0 = \mathcal{S}_0^+ \cup \mathcal{S}_0^-$. Observe that:

   - $\mathcal{S}_0$ is infinite.
   - For all $i < \left\lfloor \frac{l}{2} \right\rfloor$ there exists exactly one string in $\mathcal{S}_0^+ \cap \{0,1\}^l$ and exactly one string in $\mathcal{S}_0^- \cap \{0,1\}^l$ whose Hamming weights are equal to $2i$. This ensures that the condition imposed in Definition 27 holds for $\mathcal{S}_0$.
   - The condition imposed in Definition 36 trivially holds for $k = 0$.

2. Let us assume that there exists a high entropy set for L_k, say the set $\mathcal{S}_k \subset \Sigma_k^*$. We construct a high entropy set for the language $L_{k+1} \subset \Sigma_{k+1}^*$. We proceed as follows:

   Let $l \geq 1$. Recall that we use the symbol $\mathcal{S}_k(l)$ to denote the set $\mathcal{S}_k \cap \Sigma_k^l$. Suppose $\mathcal{S}_k(l) \neq \emptyset$. Let $\mathcal{S}_{k+1}\left(2(l+1)\log^2(l)\right)$ be equal to the set

$$\left\{ \begin{array}{c} s(w)\#_{k+1}\#_{k+1}^T 0^{K_T} : \\ s(w) = w_1\#_{k+1}\cdots\#_{k+1}w_{\log^2(l)} \text{ and} \\ w_1, ..., w_{\log^2(l)} \in \mathcal{S}_k(l) \text{ and} \\ T \in \left\{0, ..., l\log^2(l)\right\} \end{array} \right\}$$

   Set

$$\mathcal{S}_{k+1} = \bigcup_{l \geq 2} \mathcal{S}_{k+1}(l)$$

We prove that $\mathcal{S}_{k+1}$ is a high entropy set for L_{k+1}.With this we finish the proof of the separation $\mathcal{L} \neq \mathcal{NP}$. Let us first prove that $\mathcal{S}_{k+1}$ is a H-set.

Lemma 38 *Let $l > 0$ and let $X_1^l, ..., X_{\log^2(l)}^l$ be random variables i.i.d over the set $\mathcal{S}_k(l)$. Let X_Σ^l be equal to $\sum_{i \leq \log^2(l)} X_i^l$. Suppose that for all $\gamma > 0$ the inequality*

$$H\left(X_1^l\right) > (1-\gamma)\log(l)$$

holds asymptotically. Then, for all $\gamma > 0$ the inequality

$$H\left(X_\Sigma^l\right) > (1-\gamma)\log\left(2(l+1)\log^2(l)\right)$$

holds asymptotically.

Proof. Let $n \geq 1$, and let $X_1^l, ..., X_n^l$ be random variables *i.i.d* over the set $\mathcal{S}_k(l)$. Set $X_\Sigma^l(n) = \sum_{i \leq n} X_i^l$. Let us invoke *Shannon entropy power inequality* [12]. This inequality states that given m random variables $Y_1, ..., Y_m$ the relation

$$e^{2H\left(\sum_{i \leq m} Y_i\right)} \geq \sum_{i \leq m} e^{2H(Y_i)}$$

holds. Then, if the Y's are random variables *i.i.d* we get the inequality

$$e^{2H\left(\sum_{i \leq m} Y_i\right)} \geq m e^{2H(Y_1)},$$

and hence we get

$$H\left(\sum_{i \leq m} Y_i\right) \geq H(Y_1) + \frac{\ln(m)}{2}$$

Let $\gamma > 0$, we have

$$\begin{aligned} H\left(X_\Sigma^l(n)\right) &\geq H\left(X_1^l\right) + \frac{\log(n)}{2\ln(e)} \\ &> (1-\gamma)\log(l) + \frac{\log(n)}{2\log(e)}, \end{aligned}$$

where the strict inequality holds asymptotically. Notice that $X_\Sigma^l = X^l\left(\log^2(l)\right)$. We get that

$$H\left(X_\Sigma^l\right) > (1-\gamma)\log(l) + \frac{\log(\log(l))}{\log(e)}$$

And we also get that for all $\gamma > 0$ the inequality

$$H\left(X_\Sigma^l\right) > (1-\gamma)\left(\log\left(2(l+1)\log^2(l)\right)\right)$$

holds asymptotically. The lemma is proved. ■

We get the following corollary.

Corollary 39 *Let $l > 0$, let $X_1^l, ..., X_{\log^2(l)}^l$ be random variables i.i.d over the set $\mathcal{S}_k(l)$ and let $\varepsilon_1, ..., \varepsilon_{\log^2(l)}$ be random variables i.i.d. distributed over the set $\{0,1\}$ according to the distribution*

$$\Pr(\varepsilon) = \begin{cases} \frac{1}{l\log^2(l)}, & \text{if } \varepsilon = 0 \\ 1 - \frac{1}{l\log^2(l)}, & \text{if } \varepsilon = 1 \end{cases}$$

Let X_Σ^l be equal to $\sum_{i \leq \log^2(l)} \varepsilon_i X_i^l$. Suppose that for all $\gamma > 0$ the inequality

$$H\left(X_1^l\right) > (1-\gamma)\log(l)$$

holds asymptotically. Then, for all $\gamma > 0$ *the inequality*

$$H\left(X_{\Sigma}^{l}\right) > (1-\gamma)\log\left(2\left(l+1\right)\log^{2}\left(l\right)\right)$$

holds asymptotically.

We also get the following corollary

Corollary 40 $\mathcal{S}_{k+1}$ *is a H-set for* L_{k+1}.

Let us now prove that $\mathcal{S}_{k+1}$ is a high entropy set for L_{k+1}. This is the statement of next lemma. We assume that $\mathcal{S}_k$ is a high entropy set for L_k

Lemma 41 *Let* $k, r \geq 0$ *and let* $\mathcal{G}^*$ *be a promise automaton for the pair* $\left(L_{k+1}^{*}, \mathcal{S}_{k+1,r}^{*}\right)$. *Let* $\mathcal{M}$ *be a mining algorithm for* L_{k+1}^{*}. *Given* $l \geq 2$ *we use the symbol* W_l *to denote a random variable that is uniformly distributed over the set* $\mathcal{S}_{k+1,r,l}^{*}$. *For all* $\gamma > 0$ *the inequality*

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k+1,r,l}^{*}\right) \mid \mathcal{M}\left(W_l\right)\right) \geq (1-\gamma)\left(k+1\right)\log\left(l\right)$$

holds asymptotically with respect to l.

Proof. Let $r > 0$. Let $\mathcal{G}^*$ be a promise automaton for the pair $\left(L_{k+1}^{*}, \mathcal{S}_{k+1,r}^{*}\right)$. Let $\mathcal{M}$ be a mining algorithm for L_{k+1}^{*}. Let

$$W_l = \varepsilon_1 \cdots \varepsilon_n \#_{k+2} w_1 \#_{k+2} \cdots \#_{k+1} w_n \#_{k+2} u$$

be a random variable uniformly distributed over the set $\mathcal{S}_{k+1,r,l}^{*}$. Let $i \leq n = \log^{r}\left(l\right) - 1$. Let us suppose that w_i is equal to

$$v_1^i \#_{k+1} \cdots \#_{k+1} v_{\log^2(l)}^i \#_{k+1} \#_{k+1}^{T_i} 0^{K_{T_i}}$$

and let us suppose that u is equal to

$$v_1 \#_{k+1} \cdots \#_{k+1} v_{\log^2(l)} \#_{k+1} \#_{k+1}^{T} 0^{K_T},$$

We observe that the random variables

$$v_1^1, ..., v_{\log^2(l)}^1, ..., v_1^n, ..., v_{\log^2(l)}^n, v_1, ..., v_{\log^2(l)}$$

are independently and uniformly distributed over the set $\mathcal{S}_k\left(l\right)$. We also observe that the random variables $T_1, ..., T_n, T$ are independently and uniformly distributed over the set $\left\{1, ..., l\log^2\left(l\right)\right\}$. Finally, we observe that the random variables v_i^j, v_t, T_h and T are independent.

We assume that the computation of the promise automaton $\mathcal{G}^*$, on input W_l, reduces to check whether $u \notin Ham\left(L_k\right)$. This means that $\mathcal{G}^*$ has to check whether the equality

$$\sum_{j \leq \log^2(l)} \varepsilon^{L_k}\left(v_j\right) \|v_j\| = T$$

holds. This forces $\mathcal{G}^*$ to check all the factors $v_1, ..., v_{\log^2(l)}$, and sum up the Hamming weights of those belonging to L_k. Let us choose uniformly at random one of the configurations visited by $\mathcal{G}^*$ during the processing of u. We use the symbol $X_{\mathcal{G}^*}\left(\mathcal{S}^*_{k+1,r,l}\right)$ to denote this random variable. We can assume that $\mathcal{G}^*$ is checking whether some factor v_i belongs to L_k. Let us now choose, uniformly at random, one of the configurations visited by $\mathcal{G}^*$ during the processing of factor $v_{\log^2(l)}$, and let us assume that $v_{\log^2(l)}$ is the last factor checked by $\mathcal{G}^*$. We use the symbol $Z_{\mathcal{G}^*}\left(\mathcal{S}^*_{k+1,r,l}\right)$ to denote this random variable. The entropy of $Z_{\mathcal{G}^*}\left(\mathcal{S}^*_{k+1,r,l}\right)$ is bounded above by the entropy of $X_{\mathcal{G}^*}\left(\mathcal{S}^*_{k+1,r,l}\right)$. Then, we can assume, without loss of generality, that $\mathcal{G}^*$ is checking the factor $v_{\log^2(l)}$.

The processing of $s_{\log^2(l)}$ compels $\mathcal{G}^*$ to simulate, on this factor, a pebble automaton $\mathcal{G}$ that accepts L_k, possibly utilizing advice from the string

$$\varepsilon_1 \cdots \varepsilon_n \#_{k+2} w_1 \#_{k+2} \cdots \#_{k+2} w_n$$

Let us use the symbol $Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right)$ to denote the configuration of $\mathcal{G}$ that is being simulated by $\mathcal{G}^*$, and set

$$\Phi\left(W_l\right) = \sum_{t<\log^2(l)} \left\|v_t\right\| \varepsilon^{L_k}\left(v_t\right)$$

Consider the jointly distributed random variable $\left(Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right), \Phi\left(W_l\right)\right)$. The inequality

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}^*_{k+1,r,l}\right) \mid \mathcal{M}\left(W_l\right)\right) \geq H\left(Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right), \Phi\left(W_l\right) \mid \mathcal{M}\left(W_l\right)\right)$$

holds since $H\left(Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right), \Phi\left(W_l\right) \mid X_{\mathcal{G}^*}\left(\mathcal{S}^*_{k+1,r,l}\right)\right)$ equals zero. We prove that for all $\gamma > 0$ the inequality

$$H\left(Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right), \Phi\left(W_l\right) \mid \mathcal{M}\left(W_l\right)\right) \geq (1-\gamma)(k+1)\log(l)$$

holds asymptotically. The *chain rule for Shannon entropy* (see [3]) entails the equality

$$\begin{aligned} & H\left(Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right), \Phi\left(W_l\right) \mid \mathcal{M}\left(W_l\right)\right) \\ = \; & H\left(Y_{\mathcal{G}}\left(\mathcal{S}^*_{k+1,r,l}\right) \mid \Phi\left(W_l\right), \mathcal{M}\left(W_l\right)\right) + H\left(\Phi\left(W_l\right) \mid \mathcal{M}\left(W_l\right)\right) \end{aligned}$$

Let us analyze the two terms that occur at the right hand side of this equality.

For all $\gamma > 0$ the inequality

$$H\left(\Phi\left(W_l\right) \mid \mathcal{M}\left(W_l\right)\right) \geq (1-\gamma)\log(l)$$

holds asymptotically. This is a consequence of the following two facts:

1. For all $\gamma > 0$ the inequality $H\left(\Phi\left(W_l\right)\right) \geq (1-\gamma)\log(l)$ holds asymptotically since $\mathcal{S}_{k+1}$ is a H-set.

2. The random variables

$$v_1^1, ..., v_{\log^2(l)}^1, ..., v_1^n, ..., v_{\log^2(l)}^n, v_1, ..., v_{\log^2(l)}$$

are independently and uniformly distributed over the set $\mathcal{S}_k(l)$.

Let us finish with the proof. It remains to prove that for all $\gamma > 0$ the inequality

$$H\left(Y_{\mathcal{G}}\left(\mathcal{S}_{k+1,r,l}^*\right) \mid \Phi^{\mathcal{N}}(W_l), \mathcal{M}(W_l)\right) \geq (1-\gamma) k \log(l)$$

holds asymptotically.

Let U_{W_l} be equal to the string

$$\begin{array}{l}
\varepsilon^{L_k}\left(v_1^1\right) \cdots \varepsilon^{L_k}\left(v_{\log^2(l)}^1\right) \cdots \varepsilon^{L_k}\left(v_1^n\right) \cdots \varepsilon^{L_k}\left(v_{\log^2(l)}^n\right) \\
\varepsilon^{L_k}\left(v_1\right) \cdots \varepsilon^{L_k}\left(v_{\log^2(l)-1}\right) \#_{k+1} \\
v_1^1 \#_{k+1} \cdots \#_{k+1} v_{\log^2(l)}^1 \#_{k+1} \cdots \#_{k+1} v_1^n \#_{k+1} \cdots \#_{k+1} v_{\log^2(l)}^n \#_{k+1} \\
v_1 \#_{k+1} \cdots \#_{k+1} v_{\log^2(l)-1}
\end{array}$$

Let us observe that $U_{W_l} \#_{k+1} v_{\log^2(l)}$ is uniformly distributed over the set $\mathcal{S}_{k,r+2,l}^*$ (recall that the equality $n = \log^r(l) - 1$ holds). Thus, we have that $U_{W_l} \#_{k+1} v_{\log^2(l)}$ is a typical instance of any promise automaton for the pair $\left(L_k^*, \mathcal{S}_{k,r+2}^*\right)$. Let $i \leq n$, and set

$$w_i^* = v_1^i \#_{k+1} \cdots \#_{k+1} v_{\log^2(l)}^i \#_{k+1} \#_{k+1}^{T_i^*} 0^{K_{T_i^*}},$$

where $T_i^* = \sum_{j \leq \log^2(l)} \left\|v_j^i\right\| \varepsilon^{L_k}\left(v_j^i\right)$. Set

$$s\left(W_l^*\right) = w_1^* \#_{k+2} \cdots \#_{k+2} w_n^*$$

Notice that $s\left(W_l^*\right)$ can be easily computed from U_{W_l}. Let $\mathcal{K}$ be a mining algorithm for L_k^* that works, on input $s\left(U_{W_l}\right)$, as follows:

1. Computes the set

$$S_{W_l} = \left\{i < \log^2(n) : \varepsilon^{L_k}(v_i) = 1\right\}$$

2. Computes the sum

$$\Phi^{\mathcal{K}}\left(U_{W_l}\right) = \sum_{i \in S_{W_l}} \|v_i\|$$

3. Computes $s\left(W_l^*\right)$.

4. Simulates the computation of $\mathcal{M}$, on input $s\left(W_l^*\right)$, and outputs the tuple

$$\left(\begin{array}{c} S_{W_l}, S_{\mathcal{M}_1}\left(W_l^*\right), ..., S_{\mathcal{M}_{t_{\mathcal{M}}}}\left(W_l^*\right), \\ \Phi^{\mathcal{K}}\left(U_{W_l}\right), \Phi_1^{\mathcal{M}}\left(W_l^*\right), ..., \Phi_{t_{\mathcal{M}}}^{\mathcal{M}}\left(W_l^*\right) \end{array}\right)$$

We use the symbol $\left(\Phi^{\mathcal{K}}\left(U_{W_l}\right), \mathcal{M}\left(W_l^*\right)\right)$ to denote this tuple.

The inequality

$$\begin{array}{rl} & H\left(X_{\mathcal{G}}\left(\mathcal{S}_{k,r+2,l}^*\right) \mid \mathcal{K}\left(U_{W_l}\right)\right) \\ = & H\left(X_{\mathcal{G}}\left(\mathcal{S}_{k,r+2,l}^*\right) \mid \Phi^{\mathcal{K}}\left(U_{W_l}\right), \mathcal{M}\left(W_l^*\right)\right) \\ \leq & H\left(Y_{\mathcal{G}}\left(\mathcal{S}_{k+1,r,l}^*\right) \mid \Phi\left(W_l\right), \mathcal{M}\left(W_l\right)\right) \end{array}$$

holds asymptotically, (the equality holds from the definition of $\mathcal{K}\left(U_{W_l}\right)$). This follows from the following facts:

1. $\Phi^{\mathcal{K}}\left(U_{W_l}\right) = \Phi\left(W_l\right)$.
2. Suppose $\varepsilon_i = 0$. We get

$$T_i = T_i^* = \sum_{j \leq \log^2(l)} \left\| v_i^j \right\| \varepsilon^{L_k}\left(v_i^j\right)$$

Suppose $\varepsilon_i = 1$. Recall that T_i is a random integer uniformly distributed over the set $\left\{0, ..., l\log^2\left(l\right)\right\}$. Recall that the random variables T_i and $v_{\log^2(l)}$ are independent. This implies that the condition

$$T_i \neq \sum_{j \leq \log^2(l)} \left\| v_i^j \right\| \varepsilon^{L_k}\left(v_i^j\right)$$

conveys null information about the random variable $Y_{\mathcal{G}}\left(\mathcal{S}_{k,r+2,l}^*\right)$. Then, we can ignore T_i and replace it by T_i^*. Observe that this is the way we construct $s\left(W_l^*\right)$ from $s\left(U_{W_l}\right)$.

Let us invoke at this point the inductive hypothesis regarding the set $\mathcal{S}_k$ $(\mathcal{S}_{k,r+2,l}^*)$. We get that for all $\gamma > 0$ the inequality

$$H\left(X_{\mathcal{G}}\left(\mathcal{S}_{k,r+2,l}^*\right) \mid \mathcal{K}\left(U_{W_l}\right)\right) \geq \left(1-\gamma\right) k \log\left(l\right)$$

holds asymptotically. We conclude that for all $r \geq 1$, for all promise automaton $\mathcal{G}^*$ for the pair $\left(L_{k+1}^*, \mathcal{S}_{k+1,r}^*\right)$, for all mining algorithm $\mathcal{M}$ for L_{k+1}^* and for all $\gamma > 0$ the inequality

$$H\left(X_{\mathcal{G}^*}\left(\mathcal{S}_{k+1,r,l}^*\right) \mid \mathcal{M}\left(W_l\right)\right) \geq \left(1-\gamma\right)\left(k+1\right)\log\left(l\right)$$

holds asymptotically. The lemma is proved. ■

We get as corollaries of the above results the following facts:

1. For all $k \geq 1$ the set $\mathcal{S}_k$ is a high entropy set for L_k.
2. The language L_k cannot be accepted with $k-1$ pebbles.
3. The sequence $\{L_k\}_{k\geq 1}$ is high in the pebble hierarchy.
4. The separation $\mathcal{L} \neq \mathcal{NP}$ holds.

References

[1] Aanderaa, S.: On k-Tape Versus $(k-1)$-tape Real Time Computation. In proceedings of SIAM-AMS **7** (1974) 75-96

[2] Book, R. and Greibach, S.: Quasi-realtime languages, Math. Systems Theory **4** (1970) 97-111

[3] Cover, T.: *Elements of Information Theory*. Wiley, 1991.

[4] Greibach, S.: The Hardest Context-Free Language. SIAM Journal on Computing **2(4)** (1973) 304-310

[5] Greibach, S.: Jump PDA's and Hierarchies of Deterministic Context-Free Languages. SIAM Journal on Computing **3(2)** (1974) 111-127

[6] Hsia, P and Yeh, R.: Marker Automata. Information Sciences **8(1)** (1975) 71-88

[7] Mealy, G.: A Method for Synthesizing Sequential Circuits. Bell System Technical Journal **34(5)** (1955) 1045-1079

[8] Mitzenmacher, M. and Upfal, E.: *Probability and Computing*. Cambridge University Press

[9] Monien, B.: Two-way multihead automata over a one-letter alphabet. Informatique Théorique et Applications **14** (1980) 67-82

[10] Petersen, H.: The Equivalence of Pebbles and Sensing Heads for Finite Automata. In Proceedings of FCT (1997) 400-410

[11] Petersen, H.: A Census Technique for Simple Computing Devices. Unpublished Manuscript

[12] Shannon, C.: A Mathematical Theory of Communication". Bell Systems Technical Journal. **27(3)** (1948) 379-423

[13] Sudborough, I.: On Tape-Bounded Complexity Classes and Multihead Finite Automata. Journal of Comp and Syst Sci **10(1)** (1975) 62-76