The Structure of Logarithmic Advice Complexity Classes *

José L. Balcázar
Department of Software (LSI)
Universitat Politècnica de Catalunya
Pau Gargallo 5, E-08028 Barcelona, Spain
balqui@lsi.upc.es

Montserrat Hermo
Department of Software (LSI)
Universidad del País Vasco, P.O. Box 649
E-20080 San Sebastián, Spain
jiphehum@si.ehu.es

May, 1997

Abstract

A nonuniform class called here Full-P/log, due to Ko, is studied. It corresponds to polynomial time with logarithmically long advice. Its importance lies in the structural properties it enjoys, more interesting than those of the alternative class P/log; specifically, its introduction was motivated by the need of a logarithmic advice class closed under polynomial-time deterministic reductions. Several characterizations of Full-P/log are shown, formulated in terms of various sorts of tally sets with very small information content. A study of its inner structure is presented, by considering the most usual reducibilities and looking for the relationships among the corresponding reduction and equivalence classes defined from these special tally sets.

*Partially supported by the E.U. through the ESPRIT Long Term Research Project 20244 (ALCOM-IT) and through the HCM Network CHRX-CT93-0415 (COLORET); by the Spanish DGICYT through project PB95-0787 (KOALA), and by Acciones Integradas Hispano-Alemanas HA-119-B and AL-201-B.

1 Introduction

Nonuniform complexity classes were essentially introduced in [17], where the main relationships to uniform classes were already shown. In order to capture characteristics of nonuniform models of computation, in which fixed input lengths are compulsory, in the definition of nonuniform classes a bounded amount of extra information, the "advice", dependent of the length of the input, is provided. The two most natural families of bound functions for the advice information are polynomials (as in the class P/poly) and logarithms (as in the class P/log). In structural terms, the class P/poly has a high interest, due to its many characterizations, and has been studied in considerable depth.

However, P/log not being closed under the most usual reducibilities, its structural study has less interest. We consider in this paper a variant, proposed by Ker-I Ko in [19] (and also treated marginally in [11] and [8]) of the complexity class P/log, in which closure under polynomial time reductions is obtained via a more restrictive condition in the definition. Our aim is to argue here that this relative of P/log, that we call Full-P/log, not only is closed under reducibilities but also has an internal structure worth study, comparable to that of P/poly, with many similarities and a few interesting differences.

This internal structure has also consequences for the learnability of certain circuit expression classes of logarithmic Kolmogorov complexity. The reason is that Full-P/log characterizes the concepts that can be described by these representation classes, similarly to the characterization of P/poly by polynomial size circuits. This aspect of Full-P/log is treated in depth elsewhere [7].

We must mention in passing another motivation to study this class. In [24], a model of neural networks is described, and the language recognition power of these networks is characterized in terms of the types of numbers employed as weights. Also a precise correspondence is establish between the choice of integer, rational, or real weights and the respective classes of languages. When the computation time of the networks is constrained to be polynomial in the input size, the classes recognized by the respective nets are: regular, P, and P/poly [17]. It may be argued that any net with real weights that is computationally feasible to implement must admit a short description of its real-valued weights. Thus, setting a logarithmic bound on the resource-bounded Kolmogorov complexity of the real weights, it is proved in [9] that the languages recognized correspond to the class Full-P/log.

This paper is structured in two different parts. The first one shows several characterizations of Full-P/log as a reduction class. The proofs are quite interesting in that they require to define another technical variant of the class, based on prefix-closed advice words, and to introduce a new proof technique, by which information is selected at a doubly exponential rate, skipping all information corresponding to intermediate advice words. This kind of argument has been heavily used in [16]. As applications of this technique, we obtain several characterizations of Full-P/log as the reduction class of special sets: tally sets whose words follow a given regular pattern, and tally sets that are regular in a resource-bounded Kolmogorov complexity sense.

The second part focuses on the inner structure of Full-P/log, trying to understand the

power of this sort of special tally sets when they are used as oracles. To do that, we explain some of the relationships which exist among reductions classes defined from them. In addition, we describe how equivalence classes defined in terms of the same tally sets are related.

The results of sections 3 to 5 appeared in preliminary form in [10]. The results of section 6 appeared in preliminary form in [15]. Almost all the results described here are from [16].

2 Preliminaries

An alphabet Σ is any non-empty, finite set. We use here the alphabets $\{0,1\}$ and $\{0\}$. Given any alphabet Σ, a finite string (or word) over Σ is a finite sequence of symbols from Σ. We denote words by lower case Latin letters, such as $x, y, \ldots$. Any set (finite of infinite) of these strings is called a language. The language of all possible finite strings over Σ is denoted by Σ^*. There is a special string in Σ^* that is the unique word consisting of zero symbols. We call it λ.

Given a set A, we indicate the cardinality of A with the expression $|A|$. In the same way, when we refer to the number of symbols in some string x, we use $|x|$.

Regarding words, $x_{i:j}$ means the substring formed by x from the i-th symbol on up to and including the j-th symbol. As our strings are usually sequences over $\Sigma = \{0,1\}$, sometimes we speak about the j-th bit of x instead of the j-th symbol.

A natural operator, the concatenation, is defined among the strings of Σ^*. Given two words x, y, the concatenation of x with y produces the new word xy. A prefix of a word y is any word z such that for some word w, $y = zw$. The notation $z \sqsubseteq y$ denotes the fact that z is a prefix of y.

Let A be a language over some alphabet. The set of all strings x in A whose length is less than or equal to n ($|x| \leq n$) is denoted by $A^{\leq n}$. When we refer to the strings of A that have exactly length n, we use the notation $A^{=n}$. We express that the set A is included in the set B by writing $A \subseteq B$, but when the inclusion is strict we say that $A \subset B$. The complement of a language $A \subseteq \Sigma^*$ is denoted by $\overline{A}^{\Sigma}$; when Σ is known, we omit it, leaving just $\overline{A}$. Given any Σ with at least two symbols, say 0 and 1, the join or marked union of two languages A and B over Σ is:

$$A \oplus B = \{x0 \mid x \in A\} \cup \{x1 \mid x \in B\}$$

Definition 1 A language is *tally* if and only if it is included in $\{0\}^*$.

The class of all tally sets is denoted by Tally. The upper case initial is used to express the complete class while the lower case initial means "the property of belonging to Tally".

Next let us define the concept of *characteristic sequence* of a particular set:

Definition 2 Given a language $A \subseteq \Sigma^*$

1. Its characteristic sequence χ^A is an infinite string of $\{0,1\}^\infty$ such that, for all n, the n-th bit of χ^A is 1 if the n-th finite string of Σ^*, ordered by lengths and in lexicographical order within each length, belongs to A. Otherwise this bit is 0.

2. Its characteristic sequence up to a particular length n, $\chi^{A^{\leq n}}$ is a finite string fulfilling that $|\chi^{A^{\leq n}}| = 2^{n+1} - 1$ and $\chi^{A^{\leq n}} \sqsubseteq \chi^A$.

When the set A is a tally set, χ^A is the characteristic sequence relative to $\{0\}^*$, that means, we only take into account the words 0^n for any n. In the same way, slightly abusing the notation, when we use sets L such that $L \subseteq \{0^{2^n} \mid n \in \mathbb{N}\}$, the characteristic sequence χ^L will be relative to the language formed by a power of 2 many 0's. More generally, arbitrary infinite strings are denoted by Greek letters as $\alpha, \beta, \ldots$.

Throughout the paper, all logarithms are to base 2.

2.1 Different Types of Reductions

Our uniform computational model is the multi-tape Turing machine, with a read-only input tape. An oracle Turing machine has an additional write-only oracle tape. The machines can be deterministic (DTM) or nondeterministic (NDTM).

We work with classes of languages recognized by oracle Turing machines that work in polynomial time. Since the access to the oracle can be restricted in different ways the following reducibilities can be obtained:

Definition 3 A language A is r-reducible to B in polynomial time ($A \in \mathrm{P}_r(B)$) if and only if there exists an r-restricted deterministic polynomial-time oracle Turing machine M such that A is the language recognized by M querying oracle B.

Now we explain the meaning of r for a fixed reduction:

1. When r is the Turing reduction ($r = \mathrm{T}$), there are no restrictions on the oracle machine M.

2. When r is the truth-table reduction ($r = \mathrm{tt}$), the machine M is able to write down on a separate tape all the queries to be made during the computation before the first word is queried.

3. A particular case of the truth-table reduction is the k-truth-table reduction (written $r = k\text{-tt}$), when $k > 0$. Now there exist polynomial-time computable functions f and h such that for all x, $f(x)$ is a list of k strings, $f(x) = \langle x_1, x_2, \ldots, x_k \rangle$, $h(x)$ is a truth table with k variables, and $x \in A$ if and only if the truth-table $h(x)$ evaluates to *true* on the boolean k-tuple $\langle x_1 \in B, x_2 \in B, \ldots, x_k \in B \rangle$.

   Set A is bounded truth-table reducible (r= btt) to set B, if there is a positive k such that A is k-tt reducible to B.

4. When r is the many-one reduction ($r = \mathrm{m}$), it is only allowed to make a unique query to the oracle in the last step of the computation, in such a way that M accepts if and only if the answer is "YES". An equivalent definition of the many-one reduction is as follows: $A \leq_m B$ if and only if there exists a polynomial-time function f such that for all x
$$x \in A \Longleftrightarrow f(x) \in B$$

We say that the Turing reduction is an adaptive process because the queries depend on the answers given by the oracle, on the previous steps of the computation. Conversely, the truth-table approach is non-adaptive, since the queries are completely independent of the answers. Actually, the idea of the truth-table reduction is sometimes described in terms of making all the queries to the oracle at the same time, in parallel.

Given a class of languages $\mathcal{G}$, the class of sets that are r-reducible to some set in $\mathcal{G}$ is expressed by writing $\mathrm{P}_r(\mathcal{G})$. Since we only work with NDTM's that access to the oracle without any restriction, $\mathrm{NP}(A)$ identifies the class of languages that are Turing reducible to A via a NDTM.

We also define the classes of sets that are not only "reducible to" but also "inter-reducible with":

Definition 4 Given a family of languages $\mathcal{G}$, a set A is in the class $\mathrm{E}_r(\mathcal{G})$ when there exists $B \in \mathcal{G}$ such that $A \in \mathrm{P}_r(B)$ and $B \in \mathrm{P}_r(A)$.

2.2 About Functions

There exist some functions with a special property known as *honesty*.

Definition 5 A function f is honest if and only if for every value y in the range of f there is an x in the domain of f such that $f(x) = y$ and $|x| \leq |y|^k$ for some fixed constant k.

It is not known whether there are polynomial time computable honest functions whose inverses are not polynomial time computable. In fact, this open problem is equivalent to determining whether $\mathrm{P} \neq \mathrm{NP}$.

Theorem 6 (see [8]) $\mathrm{P} = \mathrm{NP}$ if and only if every honest partial function computable in polynomial time has an inverse computable in polynomial time.

We need to encode several words into one in such a way that both computing the encoding, and recovering the coded words can be easily done. We have chosen a pairing function $\langle\ \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$. Given x and y, the word $\langle x, y \rangle$ is obtained by duplicating each bit of x, appending to this the word y, inserting a 01 in between. Assuming that the lengths of x and y are n and m respectively, the length of the pairing function applied to (x, y) is $|\langle x, y \rangle| = 2n + 2 + m$. The computations of $\langle\ \rangle$ and its inverses needs very little resources.

The pairing function can be applied to tuples as follows: $\langle x, y, z\rangle = \langle\langle x, y\rangle, z\rangle$ and so on. However, sometimes, we use a different function in order to encode a finite or infinite sequence of strings. Namely, the notation

$$x_1 \# x_2 \# x_3 \# \ldots \# x_n$$

means that we append together all words in the sequence $\{x_i\}_{i\leq n}$, duplicating each bit of each x_i, separating each x_i from the next by the mark $\# = 01$.

2.3 Kolmogorov Complexity

Fix a Universal Turing machine U. The Kolmogorov complexity of a string w (resp. the Kolmogorov complexity relative to y) is the length of the shortest program (resp. pair $\langle \text{program}, y\rangle$) which, when given as input to U, will lead U to write down w as output.

Hartmanis [14] and Sipser [25] modified the original idea of Kolmogorov complexity to include the running time or space used by the Universal Turing machine, in order to produce an output. Ko, in [18][1], followed the same approach but applying it to the notion of infinite sequences with respect to polynomial-time and space complexity. The sets of bounded Kolmogorov complexity strings $\mathrm{K}[f(n), g(n)]$ is defined as follows:

Definition 7

$$\mathrm{K}[f(n), g(n)] = \{x \mid \exists y, |y| \leq f(|x|), U(y) = x \text{ in at most } g(|x|) \text{ steps}\}$$

2.4 Nonuniform classes

The basic nonuniform complexity classes are P/poly and P/log. The conditions for a language L to be in some of these classes are:

Definition 8 [17]

1. $L \in$ P/poly iff there exist $B \in$ P and a polynomial p, such that

$$\forall n\ \exists w_n\ (|w_n| \leq p(n)) \text{ such that } \forall x\ (|x| = n), x \in L \Longleftrightarrow \langle x, w_n\rangle \in B$$

2. $L \in$ P/log iff there exists $B \in P$ and a constant c, such that

$$\forall n\ \exists w_n\ (|w_n| \leq c \log n) \text{ such that } \forall x\ (|x| = n), x \in L \Longleftrightarrow \langle x, w_n\rangle \in B$$

It is easy to see that $\mathrm{P_T}(\mathrm{P/log}) = \mathrm{P/poly}$, since tally sets are in P/log (see [16] for more details). Since P/log $\neq$ P/poly (see for instance [11]), we have:

Theorem 9 $\mathrm{P_T}(\mathrm{P/log})$ is not included in P/log.

Moreover, one can see that even $\mathrm{P_m}(\mathrm{P/log})$ is not included in P/log. (see again [16]).

[1] Preliminary versions of [18] circulated simultaneously to [14] and [25].

3 The classes Full-P/log and Pref-P/log

Since the logarithmic analog to P/poly is not closed under most usual reducibilities, an alternative approach was introduced by Ko [19]. Ko's class, although with a different name, is introduced in the following definition.

Definition 10 A set A is in Full-P/log if

$$\forall n\ \exists w_n\ (|w_n| \leq c \log n)\ \forall x\ (|x| \leq n)\ x \in A \Longleftrightarrow \langle x, w_n \rangle \in B$$

where $B \in$ P and c is a constant.

Note that the difference respect to P/log lies in the range of the "$\forall x$" quantifier.

The idea is quite natural. For instance, if the definition of P/poly is changed according to this, we obtain the same class P/poly. That is to say:

Definition 11 A set A is in Full-P/poly if

$$\forall n\ \exists w_n\ (|w_n| \leq n^c)\ \forall x\ (|x| \leq n)\ x \in A \Longleftrightarrow \langle x, w_n \rangle \in B$$

where $B \in$ P and c is a constant.

Proposition 12 Full-P/poly = P/poly.

Proof: By definition, Full-P/poly $\subseteq$ P/poly. Now suppose that $A \in$ P/poly. That means:

$$\forall n\ \exists w_n\ (|w_n| \leq n^c)\ \forall x\ (|x| = n)\ x \in A \Longleftrightarrow \langle x, w_n \rangle \in B$$

where $B \in$ P and c is a constant. Denote by v_n the concatenation of all the advice words $w_1, w_2, \ldots w_n$:

$$v_n = w_1 \# w_2 \# \ldots w_n$$

For all n, the length of v_n is in $n^{O(1)}$, and v_n can be used as advice word by all the lengths up to n. Therefore $A \in$ Full-P/poly. □

It is easy to see that Full-P/log is closed under polynomial-time Turing reducibility:

Proposition 13 [19] P_T(Full-P/log) = Full-P/log.

Proof: The nontrivial inclusion is P_T(Full-P/log) $\subseteq$ Full-P/log. Suppose that $A \in P_T(C)$, with $C \in$ Full-P/log, and that the Turing reduction is done via a DTM M, which works in time n^q. In order to decide whether a fixed word x is in A, simulate the computation of the machine M on input x with the extra information of the advice word for the set C and the length $|x|^q$. Each time a query is made to C, the answer is given using the advice word. That means, for all input x, the advice word for C and the length $|x|^q$ can be used as advice word for A and the length $|x|$. The size of these advice words is a function in $O(\log n^q) = O(\log n)$, therefore $A \in$ Full-P/log. □

As a result of the closure of Full-P/log under Turing reducibility, the closure of the class under the other common polynomial-time reducibilities is also obtained, and since by Theorem 9, P/log is not closed under polynomial reducibilities, we get:

Corollary 14 P/log $\neq$ Full-P/log.

Actually, also their restrictions to tally sets are different: in Theorem 29 we precisely characterize the tally sets in Full-P/log as those of low Kolmogorov complexity.

In order to characterize the class Full-P/log we present a technical variant of full logarithmic advice, in which the advice words corresponding to various lengths are not independent but highly correlated.

Definition 15 A set A has prefix logarithmic full advice, briefly $A \in$ Pref-P/log, if A is in Full-P/log via an infinite sequence of advice words w_n having the additional property that for all $n \leq m$, w_n is a prefix of w_m.

Thus, each advice is simply an extension, with some extra bits, of the previous advice. In the limit, therefore, the sequence of advice words converges towards a unique infinite word α, such that, for all n, $w_n = \alpha_{1:c \log n}$, the first $c \log n$ bits of α. Observe also that here the advice length is so tightly bounded that, for most values of n, the corresponding advice w_n does not have room to include one more bit than its predecessor. Indeed, $c \log n$ only increases by one when n increases by a multiplicative factor of $2^{(c^{-1})}$. Thus, very frequently $w_n = w_{n+1}$, and only exponentially often can w_n be a proper prefix of w_{n+1}.

Of course, the definition can be straightforwardly rephrased to apply to other bounds on the advice length or to other uniform complexity classes. For instance, a similar definition for polynomial advice gives exactly P/poly:

Definition 16 A set A is in Pref-P/poly when A is in P/poly via an infinite sequence of polynomial advice words w_n such that for all $n \leq m$, w_n is a prefix of w_m.

Proposition 17 Pref-P/poly = P/poly.

Proof: The construction of Proposition 12 yields a sequence of advice words having the requested prefix property. □

4 Characterization of Pref-P/log

This section shows that Pref-P/log can be characterized by polynomial-time Turing reduction classes of regularly structured tally sets, as well as using bounded query machines.

Theorem 18 The following classes of languages are the same:

i/ $\bigcup_L \mathrm{P}_\mathrm{T}(L)$ where $L \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$.

ii/ $\bigcup_L \mathrm{P}_\mathrm{T}(L)$ via polynomial-time machines whose queries have lengths at most $\mathrm{O}(\log n)$, where $L \subseteq \{0\}^*$.

iii/ $\bigcup_B \mathrm{P_T}(B)$ via polynomial-time machines whose queries have lengths at most $\log(\log n) + \mathrm{O}(1)$.

iv/ Pref-P/log.

Observe that no constant factors are allowed on the term $\log(\log n)$ in part iii/: only additive constants can be accommodated in the bound. Part i/ is quite interesting, in that it shows that Pref-P/log is the reduction class of tally sets exhibiting a high degree of regularity. All query bounds mentioned assume, as usual, that n is the length of the input.

Proof: The proofs that i/ implies ii/ and that ii/ implies iii/ are simple and similar: both are tantamount to a change of scale in the oracle set. Let A be a set in $\mathrm{P_T}(L)$, with $L \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$. Define $L' = \{0^k \mid 0^{2^k} \in L\}$. It is easy to see that $A \in \mathrm{P_T}(L')$, querying 0^k instead of 0^{2^k} when required. Observe that the length of the queries is now logarithmic, since $k \in \mathrm{O}(\log n)$ whenever $2^k \in n^{\mathrm{O}(1)}$. Now we repeat the argument: if $A \in \mathrm{P_T}(L')$ with $\mathrm{O}(\log n)$ length queries, define $B = \{k \mid 0^k \in L'\}$. Again $A \in \mathrm{P_T}(B)$, and the maximum length of the oracle queries is $\log(c * \log n) = \mathrm{O}(1) + \log(\log n)$.

To prove that iii/ implies iv/, we will employ the characteristic function of B as an infinite word limiting the sequence of advice words. Let A be a set with $A \in \mathrm{P_T}(B)$ via a DTM M. By hypothesis, the lengths of the queries are smaller than or equal to $d + \log(\log n)$. Therefore, the number of different queries that M can make, is bounded by $2^{d+\log(\log n)} = c \log n$, for an appropriate constant c. Moreover, these are the first $c \log n$ words. So we define the advice w_n as the characteristic sequence of B up to the element in place $c \log n$. With this information, each query to oracle B can be answered. Thus, $A \in$ Pref-P/log.

Finally, the proof of iv/ implies i/ is essentially a converse of the composition of the three arguments. Suppose that $A \in$ Pref-P/log, where the infinite word α is the limit of the sequence of advice words. Let L be the tally set

$$L = \{0^{2^k} \mid \text{the } k\text{-th bit of } \alpha \text{ is } 1\ \}$$

Now $A \in \mathrm{P_T}(L)$ by simply querying the words 0^{2^i} for $i = 1$ to $i = c \log |x|$ to extract the necessary advice of length $c \log |x|$ from the tally oracle, and then using it. □

From now on, we denote the following classes of tally sets as Tally2:

$$\text{Tally2} = \{L \mid L \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}\}$$

Hence, parts i/ and iv/ in the theorem characterize Pref-P/log as $\mathrm{P_T}$(Tally2).

5 Characterization of Full-P/log

In this section, one of the main contributions of this paper is presented: both in results by relating Full-P/log to the classes already described, and in technical contents by explaining

a technique which consists of selecting information separated by a doubly exponential gap. Several examples of the application of this technique are presented in [16]. Now we will give two of them, both giving somewhat surprising characterizations of Full-P/log. The first one shows that Full-P/log equals the seemingly more restrictive class of sets with logarithmic prefix-closed advice, and the second one will show that Full-P/log equals a less restrictive class defined in terms of Kolmogorov-regular tally sets.

Theorem 19 Full-P/log = Pref-P/log = P_{T}(Tally2).

Equivalently, whenever a set is decidable in polynomial time with full logarithmic advice, then it is possible to construct equivalent advice words for the set, within of the same logarithmic length bounds, and obeying the restriction that each advice word is a prefix of all the following ones.

Proof: By definition, Pref-P/log is a subclass of Full-P/log. The relevant part of the theorem is of course the converse inclusion. Suppose that $A \in$ Full-P/log. This means that there is a set $B \in \mathrm{P}$ and a sequence of advice words $\{w_n \mid n \in \mathbb{N}\}$ with $|w_n| \in \mathrm{O}(\log n)$ so that $\forall x, \forall m \geq |x|, x \in A \iff \langle x, w_m \rangle \in B$. We will use the result in the previous section, characterizing Pref-P/log as P_{T}(Tally2). Thus we will define a tally set L containing only words of length a power of 2, and will prove that $A \in \mathrm{P}_{\mathrm{T}}(L)$.

Since Full-P/log is closed under polynomial-time Turing reducibility, it is strictly smaller than P/log, and therefore our proof now *must* unavoidably exploit the property that one advice can help all the smaller lengths. The main idea is to keep only the information corresponding to some selected advice strings, instead of storing all of them in the oracle set. Of course, we have to select for the oracle infinitely many advice words; but the fact that each of them is good for all the words smaller than the length it is designed for allows us to select them arbitrarily far apart.

We must find a balance between two contradictory restrictions. If we select advice strings for the oracle too frequently then they will need too many bits, and some of them will be encoded too far away in the oracle; but if we select them too separated, then for some words the nearest valid advice would be too long to be extracted from the oracle by a polynomial-time machine.

It turns out that there is a way of skipping advice words for which the balance is satisfactory. We will encode in the oracle all the advice words corresponding to lengths 2^{2^m} for all $m \in \mathbb{N}$ and skip all the intermediate ones. Each bit, of each of the selected advice strings, will be encoded by the presence or absence of a word of the form 0^{2^m} in the set L.

Let k be a constant, such that $|w_n| \leq k \log n$ for all n, and without loss of generality assume that $|w_n| = k \log n$ by padding out each $|w_n|$ with a suffix word from 10^* up to the desired length.

The advice words corresponding to the length 2^{2^0}, respectively $2^{2^1} \ldots 2^{2^m}$, have size k, respectively $2k \ldots 2^m k$. We use the first k powers of two, from 0^2 until 0^{2^k}, to encode the advice for length 2^{2^0}. (The empty string is not used here.) The second advice string to store has length $2k$, so this information needs $2k$ powers of two: use the next ones, from

$0^{2^{k+1}}$ until $0^{2^{k+2k}}$. In general, the information of the advice corresponding to the length 2^{2^m} is encoded in the tally set L from the element

$$0^{2^{k+2k+2^2k+\cdots+2^{m-1}k+1}}$$

up to

$$0^{2^{k+2k+2^2k+\cdots+2^mk}}$$

So let L be

$$L = \{0^{2^{\left(\sum_{i\leq m-1} 2^i k\right)+p}} \mid 1 \leq p \leq 2^m k \text{ such that the } p\text{-th bit of } w_{2^{2^m}} \text{ is } 1\}$$

We prove first that $A \in \mathrm{P_T}(L)$. On input x, find an integer m such that $2^{2^{m-1}} < |x| \leq 2^{2^m}$. This can be done in polynomial time. Since $2^{2^m} = (2^{2^{m-1}})^2 < |x|^2$, this selection ensures that $\log(\log|x|) \leq m < \log(\log|x|^2)$.

Now, for each value of p from 1 to $2^m k$, ask whether $0^{2^{\left(\sum_{i\leq m-1} 2^i k\right)+p}} \in L$ and, in this way, obtain all the bits of the advice $w_{2^{2^m}}$, which now can be used to decide whether $x \in A$ in polynomial time. It remains to be seen whether the queries can be asked in polynomial time; it suffices to prove that they are polynomially long.

The number of queries is bounded by $k \log|x|^2$. A bound on the length of the oracle queries is

$$2^{k+2k+2^2k+\cdots+2^mk} = 2^{(1+2+2^2+\cdots+2^m)k} < 2^{2^{m+1}k}$$

As $m < \log(\log|x|^2)$, the queries have length at most

$$2^{k2^{\log(\log|x|^d)}} = (2^{2^{\log(\log|x|^d)}})^k = |x|^{d'}$$

for appropriate constants d and d'. So $A \in \mathrm{P_T}(L)$. □

We have chosen to keep those advice strings corresponding to the length 2^{2^m}. Let us briefly describe how crucial the arithmetic properties of the double exponential are for this proof. Naively it may seem that a single exponential separation should suffice; but this fails because for each advice there are logarithmically many smaller advice words of logarithmic length to be encoded, i.e. a total of $(\log n)^2$ bits: when distributed over the tally set, they cover a broad region up to length $n^{\log n}$ which cannot be scanned in polynomial time. Surprisingly, as described above, a double exponential works. However, if we would try to select advice strings with triply exponential gaps, skipping all advice words except those corresponding to lengths $2^{2^{2^m}}$, then these advice strings are too large, although there are fewer of them: the first appropriate m would be such that $2^{2^{2^{m-1}}} < |x| \leq 2^{2^{2^m}}$, and straightforward computation shows that the corresponding advice might be $n^{\log n}$ long.

We give now a second application of the doubly exponential skip technique. Considering the previous results, it is clear in what sense the tally sets used exhibit a regularity: their words can appear at only selected, specific places such as powers of 2. Many other similar

notions of tally sets with regularities can be proposed, but among them there is one that is particularly natural: regularity could be defined in terms of resource-bounded Kolmogorov complexity. We could consider tally sets that are regular in the sense that there is a short, say logarithmic, way of describing their characteristic function and a resource-bounded, say polynomial time, algorithm to recover it. Observe that the tally sets used in the proof of Theorem 19 fulfill this regularity property.

In principle the class obtained would be larger, since it is conceivable that some tally sets are Kolmogorov regular but encode more information than a set having such an extreme regularity as implied by the superset $\{0^{2^k} \mid k \in \mathbb{N}\}$. We will show that, modulo polynomial-time Turing reducibility, this is not the case: the reduction class of Kolmogorov-regular sets is again Full-P/log. As before, L denotes a tally set.

Theorem 20 The following two classes coincide:

i/ Full-P/log.

ii/ $\bigcup_L \mathrm{P}_\mathrm{T}(L)$ where there exists a positive constant c (depending only on L) such that, for all n, $\chi^{L^{\leq n}} \in \mathrm{K}[c \log n, n^c]$.

Proof:

Again, we use the characterization of the class Full-P/log as P_T(Tally2) which follows from the previous result. Then it is easy to see that i/ implies ii/: to construct the characteristic sequence of a tally2 set L up to a fixed length we only need to know which ones among the logarithmically many words of the form 0^{2^k} are in L; these are the only potential non-zeros in χ^L. Thus given, as a logarithmically long seed, the characteristic function of L relative to $\{0^{2^k} \mid k \in \mathbb{N}\}$, we can easily print out an initial segment of χ^L in time polynomial in the length of the output.

To see that ii/ implies i/, again we apply the doubly exponential skip technique. Observe first that an easier proof seems possible. Consider $A \in \mathrm{P}_\mathrm{T}(L)$ where L is Kolmogorov-regular; we can show that A can be accepted with the help of a short advice. On input x, the maximum oracle query is $0^{|x|^q}$ for some q. To decide whether $x \in A$ it suffices to know an initial segment of the sequence χ^L up to $|x|^q$ bits (recall that the characteristic sequence of a tally set is taken with respect to $\{0\}^*$). We can obtain this sequence in polynomial time from a seed of size $\log |x|^q = \mathrm{O}(\log |x|)$, which we take as an advice word. It follows that $A \in$ P/log. However, this does not prove that $A \in$ Full-P/log. It may be the case that together with x we get a seed for an advice creating an exponential part of the characteristic function, which is much more than we need; but the relevant part of it may take too long to be constructed, and then there is no way to decide x in polynomial time.

We resort again to a doubly exponential skip: for a given length n, select as advice not a single seed but a sequence of them, corresponding to lengths of the form 2^{2^m}, up to the smallest one allowing us to construct n^q bits of χ^L. This one corresponds to

$$2^{2^{m-1}} < n^q \leq 2^{2^m}$$

so that $m \leq \log(\log n^q) = \log(\log n) + \mathrm{O}(1)$. For length 2^{2^i}, the length of the seed is $c \log 2^{2^i} = c2^i$, and thus as before the total length of the sequence of seeds selected for the advice is $\sum_{i \leq m} c2^i = c2^{m+1} \in \mathrm{O}(\log n)$. Now the difficulty explained above can be avoided. If, together with x, we get the advice for a much longer length, we can scan it and select a seed large enough to create χ^L up to $|x|^q$ but not much more: there is one there for 2^{2^m} with $2^{2^{m-1}} < |x|^q \leq 2^{2^m}$, which implies $2^{2^m} < |x|^{2q}$, only quadratically longer. Therefore $A \in$ Full-P/log. □

From now on, we denote the class of Kolmogorov-regular sets as Lowtally, on the basis of their low Kolmogorov complexity. That is to say: $L \in$ Lowtally if and only if $L \in$ Tally and there exist a positive constant c such that, for all n, $\chi^{L \leq n} \in \mathrm{K}[c \log n, n^c]$.

Thus, for polynomial-time machines, tally2 oracle sets have exactly the same power as lowtally oracle sets. Again we have a phenomenon like the one discussed previously: longer and longer prefixes of the characteristic function of the tally oracle, which require $\log n$ new bits linearly often to be described, can be replaced by a much simpler oracle which, exponentially often, adds a constant number of bits.

6 The inner structure of Full-P/log

Full-P/log has been characterized in previous sections in terms of the reduction class to tally sets with very small information content. We focus here on the most usual reducibilities and investigate the corresponding reduction classes to these special tally sets. In addition to other results, we show that using tally2 languages as oracles, there are more sets that can be recognized under Turing (or, equivalently, truth-table) reducibility than under bounded truth-table reductions. The analogous problem for lowtally sets remains open.

To provide some context, let us mention the paper by Book and Ko [12]. There, the classes of sets that can be reduced to sparse and tally sets under different notions of reducibilities are studied. On the other hand, Tang and Book [26] and Allender and Watanabe [3] studied sets that are not only reducible to arbitrary tally and sparse languages, but also inter-reducible with them. With the same approach, we consider here reduction and equivalence classes to tally2 and lowtally sets, and study the relationships between them: although the truth-table and the Turing equivalence classes (i.e. degrees) of arbitrary tally sets do not coincide, in the case of using our restricted tally sets the problem of deciding whether they are different is at least as difficult as solving P $\neq$ NP. In the same way, it is argued that separating the class of languages that are bounded truth-table equivalent to lowtally sets, from the class of languages that are m-equivalent to the same class of sets is also a difficult task. However the relationship between these two degrees, when tally2 sets are used as oracles, is clear: both degrees differ.

As a consequence of our results, we can present a rather complete (read "not too incomplete") map of the inner structure of Full-P/log.

6.1 Relationships among Reduction Classes

In this subsection we focus on the reduction classes defined from tally2 and lowtally sets, and explain some of the relationships that exist among them.

The Turing and truth-table reducibilities in polynomial time are equivalent when tally sets are used as oracle. Using the same arguments, it is easy to see that the following holds:

Proposition 21

$$\mathrm{P_T}(\text{Lowtally}) = \mathrm{P_T}(\text{Tally2}) = \mathrm{P_{tt}}(\text{Lowtally}) = \mathrm{P_{tt}}(\text{Tally2}).$$

Proof: Suppose that T is a tally2 set, and $A \in L(M,T)$ where M runs in time bounded by n^q. Let m be such that $2^{m-1} < n \leq 2^m$. The number of possible nontrivial (i.e. potentially answered "yes") queries to T, made by M on input of size n, is bounded by mq. Therefore, there is a logarithmic quantity of potential queries to the oracle which we know in advance. We can make all these queries at the beginning of the computation. This fact implies that tally2 sets produce the same information under truth-table than under Turing reductions.

By the characterization of Full-P/log the reduction classes $\mathrm{P_T}(\text{Lowtally})$ and $\mathrm{P_T}(\text{Tally2})$ coincide. From the above argument, $\mathrm{P_T}(\text{Tally2}) = \mathrm{P_{tt}}(\text{Tally2})$, therefore, $\mathrm{P_T}(\text{Lowtally}) = \mathrm{P_{tt}}(\text{Tally2})$. But since the class Tally2 is included in the class Lowtally, $\mathrm{P_{tt}}(\text{Tally2}) \subseteq \mathrm{P_{tt}}(\text{Lowtally}) \subseteq \mathrm{P_T}(\text{Tally2})$, and all four coincide. □

We now study the relationship between many-one and bounded truth-table reductions over tally2 and lowtally sets. As we shall see next, both reducibilities, applied to lowtally languages, have the same power; however the behaviour of them over tally2 sets is different.

Proposition 22 $\mathrm{P_{btt}}(\text{Lowtally}) = \mathrm{P_m}(\text{Lowtally})$.

Proof: The proof is based on the fact that the boolean closure of any class $\mathrm{P_m}(A)$ is precisely the class $\mathrm{P_{btt}}(A)$ [20], and follows Book and Ko's steps in [12]. Applying this fact, if we show that $\mathrm{P_m}(\text{Lowtally})$ is closed under boolean operations, then we obtain that $\mathrm{P_{btt}}(\text{Lowtally}) = \mathrm{P_m}(\text{Lowtally})$.

Let T be any lowtally set, therefore for all length n, $\chi^{T^{\leq n}} \in \mathrm{K}[d \log n, n^d]$ for some constant d. The same seed for $\chi^{T^{\leq n}}$ can be used to produce $\chi^{\overline{T}^{\leq n}}$, that is to say

$$\chi^{\overline{T}^{\leq n}} \in \mathrm{K}[d \log n, n^d].$$

Then $\mathrm{P_m}(\text{Lowtally})$ is closed under complementation.

Closure under intersection follows from the following fact: Let T_A and T_B be lowtally sets. Suppose that f reduces $A \leq_m T_A$ and g reduces $B \leq_m T_B$. Then we can define the set $T = \{0^{\langle n,m \rangle} \mid 0^n \in T_A \text{ and } 0^m \in T_B\}$ whose characteristic sequence can be also constructed in polynomial time from logarithmic seeds. The function h such that $h(x) = 0^{\langle n,m \rangle}$ when $f(x) = 0^n \in T_A$ and $g(x) = 0^m \in T_B$, or $h(x) = 1$ otherwise, testifies that $A \cap B \leq_m T$. □

The behaviour of tally2 sets is completely different as we point out in next results. Although $\leq_m$ is equivalent to $\leq_{1\text{-tt}}$, for all $k \geq 2$ holds that $\leq_{k\text{-tt}}$ and $\leq_{(k-1)\text{-tt}}$ do not coincide. Essentially, the analogous argument using $h(x) = 0^{2^{\langle n,m \rangle}}$ for $f(x) = 0^{2^n}$ and $g(x) = 0^{2^m}$ breaks down because $|0^{2^{\langle n,m \rangle}}| = 2^{\langle n,m \rangle}$ is about $2^{n \times m}$, which is not polynomial in $2^n + 2^m$.

Proposition 23 $\mathrm{P_m}(\mathrm{Tally2}) = \mathrm{P}_{1\text{-tt}}(\mathrm{Tally2})$.

Proof: Let T be a tally set in Tally2, and assume that $A \in \mathrm{P}_{1\text{-tt}}(T)$ via a Turing machine M. We define another tally2 set L containing information about T and its complement.

$$L = \{0^{2^{2^m}} \mid 0^{2^m} \in T\} \cup \{0^{2^{2^{m+1}}} \mid 0^{2^m} \notin T\}.$$

Using as oracle the set L and a new machine N we can see that $A \in \mathrm{P}_m(L)$. N works as M, but whenever M makes a query of the form 0^{2^m} (w.l.o.g. we assume that all queries made by M are of this form), it has to change as follows:

1. Suppose that the computation of M is independent from the answer given by T. In this case, N does not ask, and follows the same steps as M.

2. Suppose that M rejects when the answer is NO, and accepts when the answer is YES. Now N makes the query $0^{2^{2^m}}$ to L, and accepts if and only if the answer is YES.

3. If M rejects when the answer is YES, otherwise accepts, then N makes the query $0^{2^{2^{m+1}}}$ to L, and accepts if and only if the answer is YES.

N testifies that $A \in \mathrm{P}_m(L)$, and therefore $A \in \mathrm{P}_m(\mathrm{Tally2})$. □

The following theorem states that there exists an strict hierarchy among the reductions classes $\mathrm{P}_{k\text{-tt}}(\mathrm{Tally2})$ when $k \geq 1$.

Theorem 24 $\forall k > 1 \;\; \mathrm{P}_{(k-1)\text{-tt}}(\mathrm{Tally2}) \subset \mathrm{P}_{k\text{-tt}}(\mathrm{Tally2})$.

We construct a language in $\mathrm{P}_{k\text{-tt}}(\mathrm{Tally2}) - \mathrm{P}_{(k-1)\text{-tt}}(\mathrm{Tally2})$, by diagonalization. Let $\{f_i\}_{i \in \mathbb{N}}$ be an enumeration of the polynomial-time computable functions that for any string x yields a list of $k-1$ strings. Let $\{h_j\}_{j \in \mathbb{N}}$ be an enumeration of all polynomial-time computable functions that for any single string x yields one of the $2^{2^{k-1}}$ $(k-1)$-tt conditions. We can enumerate all of the $(k-1)$-tt reduction machines as $\{M_{i,j}\}_{i,j \in \mathbb{N}}$ where on input x, $M_{i,j}$ computes the list $f_i(x) = \langle x_1, x_2, \ldots, x_{k-1} \rangle$ and the $(k-1)$-tt condition $h_j(x)$.

We assume for now that k is an odd number. The even case needs just slight syntactic changes in the proof. The language we want to construct is defined in terms of a particular tally2 set T, and therefore it is denoted by $L(T)$:

$$L(T) = \{0^{2^{r_1}} 1^{2^{r_2}} 0^{2^{r_3}} \ldots 1^{2^{r_k - 1}} 0^{2^{r_k}} \mid \forall s \, (1 \leq s \leq k) \, 0^{2^{r_s}} \in T \text{ and } r_1 < r_2 < \ldots < r_k\}$$

At stage $m = \langle q, i, j \rangle$ we expand the set $L(T_{m-1})$ constructed so far, adding a word in such a way that $L(T_m)$ cannot be $k-1$-reducible, via f_i (whose running time is bounded

by the function n^q), and h_j, to any tally2 set. At the beginning T_0 is the empty set. In order to do this, we study the function f_i, because we use different strategies depending on whether it is injective.

Similarly to $f_i(x) = \langle x_1, x_2, \ldots, x_{k-1} \rangle$, denote $f_i(y) = \langle y_1, y_2, \ldots, y_{k-1} \rangle$ to define that x and y are alike for T under f_i if the two boolean vectors

$$\overline{f_i}^T(x) = \langle x_1 \in T, x_2 \in T, \ldots, x_{k-1} \in T \rangle \text{ and } \overline{f_i}^T(y) = \langle y_1 \in T, y_2 \in T, \ldots, y_{k-1} \in T \rangle$$

coincide.

Stage $m = \langle q, i, j \rangle$

Consider $M_{i,j}$ with running time bounded by the function n^q;
Let $words_m = \{w \in \{0,1\}^* \mid w = 0^{2^{r_1}} 1^{2^{r_2}} 0^{2^{r_3}} \ldots 1^{2^{r_k - 1}} 0^{2^{r_k}}$
and $\frac{2^m}{k} < r_1 < r_2 < \ldots < r_k \leq \frac{2^{m+1}}{k}\}$;
For each of the $2^{2^{k-1}}$ $(k-1)$-tt conditions t,
let $G_t = \{x \in words_m \mid h_j(x) = t\}$.
Choose any t such that $|G_t|$ is maximum.
if f_i is not injective on G_t
then look for two different words $0^{2^{r_1}} \ldots 0^{2^{r_k}}, 0^{2^{t_1}} \ldots 0^{2^{t_k}} \in G_t$
such that $f_i(0^{2^{r_1}} \ldots 0^{2^{r_k}}) = f_i(0^{2^{t_1}} \ldots 0^{2^{t_k}})$;
$T_m := T_{m-1} \cup \{0^{2^{r_1}}, \ldots, 0^{2^{r_k}}\}$;
else look for two different words $0^{2^{r_1}} \ldots 0^{2^{r_k}}, 0^{2^{t_1}} \ldots 0^{2^{t_k}} \in G_t$
that are alike under f_i for all $T \in$ Tally2
$T_m := T_{m-1} \cup \{0^{2^{r_1}}, \ldots, 0^{2^{r_k}}\}$;
end if;

Given the function f_i working in time n^q, at stage $m = \langle q, i, j \rangle$, either there exists two different words $w, v \in G_t$ such that, $f_i(w) = f_i(v)$. In this case the words added to T_{m-1} witnesses that the set $L(T_m)$ is not $(k-1)$-reducible to any set via f_i and h_j; or for all words $w, v \in G_t$, $f(w) \neq f(v)$. The key point now is the cardinality of G_t.

On the one hand, the number of different words in the set $words_m$ is exactly $\binom{\frac{2^m}{k}}{k}$, which is $\Omega((\frac{2^m}{k^2})^k) = \Omega(\frac{2^{km}}{k^{2k}})$.

On the other hand, there are $2^{2^{k-1}}$ $(k-1)$-tt conditions t: there must exist some t whose G_t has at least $\frac{2^{km}}{2^{2^{k-1}} k^{2k}} \in \Omega(2^{km})$ words. Therefore, the G_t chosen in the algorithm has at least $\Omega(2^{km})$ different words. From now on we work with this G_t.

In the case that the function is injective on G_t, we need the following lemma; recall from above that, for each tally2 set T, the function $\overline{f_i}^T$ is defined as

$$\overline{f_i}^T(x) = \langle x_1 \in T, x_2 \in T, \ldots, x_{k-1} \in T \rangle.$$

Lemma 25 If f_i is injective on G_t, then there exists at least two words $x, y \in G_t$, that are alike under f_i for all tally2 sets T: $\overline{f_i}^T(x) = \overline{f_i}^T(y)$.

Applying this lemma, as we add to T_{m-1} the corresponding strings in order that $x \in L(T_m)$, without adding the corresponding ones for y, we conclude that the set $L(T_m)$ can not be $(k-1)$-tt reducible to any tally2 set via $M_{i,j}$.

Now we prove the lemma.

Proof: The length of the largest string in the set $words_m$ is bounded by $2^{2^{m+1}}$. Since $M_{i,j}$ works in time bounded by the function n^q, the lengths of the queries made by $M_{i,j}$ are bounded by $(2^{2^{m+1}})^q = 2^{q2^{m+1}}$. The strings of the tally2 sets are of the form 0^{2^j} with $j \in \mathbb{N}$, but $M_{i,j}$ only can query words 0^{2^j} with $0 \leq j \leq q2^{m+1}$. Therefore there are $1 + q2^{m+1}$ many different words in the tally2 sets that could be queried by $M_{i,j}$.

Now we show that there must exist two different words x and y of G_t fulfilling that $f_i(x) \neq f_i(y)$. because the function f_i is injective on G_t, but for each l such that $1 \leq l \leq k-1$, if $x_l \neq y_l$, then x_l and y_l are not of the form 0^{2^j} with $0 \leq j \leq q2^{m+1}$. That is, either x_l and y_l are different, and then both strings are outside of any tally2 set, or $x_l = y_l$. This ensures that for any tally2 set T, $\overline{f_i}^T(x) = \overline{f_i}^T(y)$.

Fix a string outside of any tally2 set, for example the string: 11. Suppose that each time that $M_{i,j}$, on any input in G_t, queries a string that is not of the form 0^{2^j} with $0 \leq j \leq q2^{m+1}$, this string is always 11. Then we can show that the amount of different words x in G_t, with the property that $M_{i,j}(x)$ makes at least one query of the form 0^{2^j} with $0 \leq j \leq q2^{m+1}$, is fewer than the cardinality of G_t.

On the one hand, the quantity of strings x in G_t, such that $M_{i,j}(x)$ makes a unique query of the form 0^{2^j} with $0 \leq j \leq q2^{m+1}$ is at most $\binom{k-1}{1}(1 + q2^{m+1})$. In the same way, the number of strings x in G_t, such that $M_{i,j}(x)$ makes exactly two queries that could be in some tally2 set is $\binom{k-1}{2}(1 + q2^{m+1})^2$. In general, the number of strings x in G_t, such that $M_{i,j}(x)$ makes exactly i queries $(1 \leq i \leq k-1)$ that could be in some tally2 set is $\binom{k-1}{i}(1 + q2^{m+1})^i$. Using that $q \leq m$, so that $m^{k-1} < 2^m$ for large enough m, the total amount of strings with the above property is:

$$\Sigma_{i=1}^{k-1} \binom{k-1}{i} (1 + q2^{m+1})^i \leq (2 + q2^{m+1})^{k-1} \in o(2^{km}).$$

On the other hand $|G_t| \geq \Omega(2^{km})$. Therefore, there must exist at least two words x and y in G_t such that when $M_{i,j}(x)$ makes a query of the form 0^{2^j} with $0 \leq j \leq q2^{m+1}$, then $M_{i,j}(y)$ makes the same query (otherwise both query outside every tally2 set). This implies that $\overline{f_i}^T(x) = \overline{f_i}^T(y)$. □

We show now how the lowtally sets provide more information than the tally2 sets used as oracles under btt-reduction.

Theorem 26 $\mathrm{P}_{\mathrm{btt}}(\mathrm{Tally2}) \subset \mathrm{P}_{\mathrm{btt}}(\mathrm{Lowtally})$.

Proof: We prove the stronger fact that the class Lowtally is not included in $\mathrm{P}_{k\text{-tt}}(\mathrm{Tally2})$ for any constant $k \geq 1$; i.e., we find a set $L \in$ Lowtally such that, given a polynomial-time

computable function f, that for any word x yields a list of k strings, it is not the case that $L \in \mathrm{P}_{k\text{-tt}}(\text{Tally2})$.

In order to define such a lowtally set L, for each i we denote by I_i the following set

$$\mathrm{I}_i = \{0^n \mid n = j * 2^{2^i},\ \text{with } 2 \leq j \leq 2^{2^i}\}$$

and we define the set L in such a way that, for all $i \geq 0$, there is exactly one word in I_i that belongs to set L. Actually, this unique word is called x_i and fulfills that

$$x_i = 0^{j_i * 2^{2^i}} \text{ with } 2 \leq j_i \leq 2^{2^i}$$

Hence, in each I_i, there are $2^{2^i} - 1$ possibilities of choosing this string.

If we get L of this form, then we ensure that $L \in$ Lowtally. To know the characteristic sequence of L up to length n, we look for the integer m fulfilling $2^{2^{m-1}} < n \leq 2^{2^m}$. Let g be a function that on each i, $g(i)$ expresses what x_i is chosen given the information j_i.

$$g(i) = 0^{2^i - |j_i|} j_i$$

so that $|g(i)| = 2^i$. In order to obtain $\chi^{L^{\leq n}}$ we can use as seed the following word s

$$s = g(0)g(1)\ldots g(m-1)g(m)$$

that is formed by concatenating each $g(i)$ for $i \leq m$. The length of s is exactly $|s| = \Sigma_{i=0}^{m} 2^i = 2^{m+1} - 1$ that is bounded by $4 \log n$. Thus $\chi^{L^{\leq n}} \in \mathrm{K}[d \log n, n^d]$ (d is a constant) and $L \in$ Lowtally.

We construct the set L by diagonalization. Let $\{f_i\}_{i \in \mathbb{N}}$ be an enumeration of the polynomial-time computable functions that for any string x yields a list of k strings. Let $\{h_j\}_{j \in \mathbb{N}}$ be an enumeration of all polynomial-time computable functions that for any single string x yields one of the 2^{2^k} k-tt conditions. We can enumerate all of the k-tt reduction machines as $\{M_{i,j}\}_{i,j \in \mathbb{N}}$ where on input x, $M_{i,j}$ computes the list $f_i(x) = \langle x_1, x_2, \ldots, x_k \rangle$ and the k-tt condition $h_j(x)$.

In each stage $m = \langle q, i, j \rangle$ we are adding to the set L one word of the form $0^{j * 2^{2^m}}$, $(2 \leq j \leq 2^{2^m})$, in such a way that, L cannot be k-reducible, via f_i and h_j, to any tally2 set. As in the diagonalization of Theorem 24, in the case of having an injective function f_i, we can find two words alike under f_i for all tally2 sets; so, if only one of these words is included in the set L, then we diagonalize over Tally2.

Stage $m = \langle q, i, j \rangle$

Consider $M_{i,j}$ with running time bounded by the function n^q;
Let $\mathrm{I}_m = \{0^p \mid p = j * 2^{2^m},\ \text{with } 2 \leq j \leq 2^{2^m}\}$;
For each of the 2^{2^k} k-tt conditions t, let $G_t = \{x \in \mathrm{I}_m \mid h_j(x) = t\}$.
Choose any t such that $|G_t| \geq \frac{2^{2^m}-1}{2^{2^k}}$.
if f_i is not injective on G_t.
then look for $0^{p_1}, 0^{p_2} \in \mathrm{I}_m$, $p_1 \neq p_2$, such that $f_i(0^{p_1}) = f_i(0^{p_2})$;

$L_m := L_{m-1} \cup \{0^{p_1}\}$;
else look for $0^{p_1}, 0^{p_2} \in G_t$, $p_1 \neq p_2$, alike under f_i for all $T \in$ Tally2
$L_m := L_{m-1} \cup \{0^{p_1}\}$;
end if;

When the function is not injective on G_t, the selection of 0^{p_1} ensures that L is not k-tt reducible to any set via $M_{i,j}$. In the other case, for each tally2 set T, again let $\overline{f_i}^T$ be the function such that

$$\overline{f_i}^T(x) = \langle x_1 \in T, x_2 \in T, \ldots, x_k \in T \rangle.$$

A similar result to Lemma 25 holds:

Lemma 27 If f_i is injective on G_t, then there exist at least two words x, y in G_t, such that, for any tally2 set T, $\overline{f_i}^T(x) = \overline{f_i}^T(y)$.

Applying this lemma, if we add to L_{m-1} the string x, without adding y, then we conclude that the set L cannot be k-tt reducible to any tally2 set via $M_{i,j}$.

The proof of the lemma follows the same steps of Lemma 25. The largest string in the set I_m is $0^{2^{2^m} * 2^{2^m}}$ whose length is $2^{2^{m+1}}$. Therefore there are $1 + q2^{m+1}$ many different words in the tally2 sets that could be queried by $M_{i,j}$.

Fixing a word outside the tally2 for all queries made by $M_{i,j}$ that are not of the form 0^{2^j} with $0 \leq j \leq q2^{m+1}$, the amount of different words x in G_t, with the property that $M_{i,j}(x)$ makes at least one query that could be in some tally2 set, is fewer than the cardinality of G_t. Namely this amount is

$$\Sigma_{i=1}^{k} \binom{k}{i} (1 + q2^{m+1})^i \leq (2 + q2^{m+1})^k.$$

While the cardinality of G_t is greater than or equal to $\frac{2^{2^m}-1}{2^{2^k}}$. Therefore, since $\frac{2^{2^m}-1}{2^{2^k}} > (2 + q2^{m+1})^k$ (for large enough m), there must exist at least two words x and y in G_t such that $\overline{f_i}^T(x) = \overline{f_i}^T(y)$. □

This result, together with the fact that in $\mathrm{P}_{\mathrm{btt}}$(Tally2) there exist languages that are not lowtally sets, ensure that both classes are incomparable. This is not the case when Turing (or equivalently truth-table) reduction is used, because the class Lowtally is strictly included in P_{T}(Tally2). The relationship between Lowtally and P_{T}(Tally2) is explained in more detail in the following theorem, but before, let us introduce a lemma which will be helpful later on.

Lemma 28 For each lowtally set L, there is a tally2 set $T2$ such that $L \in P_{\mathrm{tt}}(T2)$ and $T2 \in P_{\mathrm{tt}}(L)$.

Proof: Let L be any lowtally set. By definition of the class Lowtally, there exists a constant c, such that for every length n, there exists a seed s_n, with $|s_n| \leq c \log n$, that produces $\chi^{L^{\leq n}}$ in polynomial time. Without loss of generality we consider that every seed s_n has exactly $c \log n$ number of bits. Moreover, if there would be many different seeds for the same prefix of the characteristic sequence, we choose only the first seed in lexicographical order. The idea is to encode the seeds s_n into a tally2 set $T2$, in order to produce $\chi^{L^{\leq n}}$.

The way of encoding these seeds is again based on the "doubly exponential skip" technique, and consists on keeping only the information s_n corresponding to $n = 2^{2^m}$ with $m \in \mathbb{N}$.

For all m, the seed corresponding to length 2^{2^m}, has size $2^m c$. So, as in Theorem 19, let $T2$ be

$$T2 = \{0^{2^{\left(\sum_{i \leq m-1} 2^i c\right)+p}} \mid 1 \leq p \leq 2^m c \text{ such that the } p\text{-th bit of } s_{2^{2^m}} \text{ is } 1\}$$

On the one hand, $L \in \mathrm{P}_{\mathrm{tt}}(T2)$: to decide whether a word of the form 0^n is in L, we can generate easily $\chi^{L^{\leq n}}$ querying $T2$. The steps to follow are exactly those given in the proof of Theorem 19: first find an integer m such that $2^{2^{m-1}} < n \leq 2^{2^m}$, and then obtain the seed $s_{2^{2^m}}$ querying the tally2 set $T2$. It is easy to see that the sizes of the queries are polynomially long, and there are polynomially many queries. Furthermore, they are nonadaptive, as required.

On the other hand, $T2 \in \mathrm{P}_{\mathrm{tt}}(L)$. On input 0^{2^n}, the following steps suffice to decide whether $0^{2^n} \in T2$:

1. Look for the number m fulfilling:

   $$c(2^m - 1) < n \leq c(2^{m+1} - 1)$$

   Thus, $2^n = 2^{c(2^m-1)+p}$ with $1 \leq p \leq c2^m$.

2. Find the characteristic sequence of L up to length 2^{2^m} querying, in a nonadaptive way, L. Note that 2^{2^m} is polynomial in 2^n because:

   $$c(2^m - 1) < n \Longrightarrow 2^m < \frac{n+c}{c} \Longrightarrow 2^{2^m} < 2^{\frac{n}{c}+1}$$

3. When $\chi^{L^{\leq 2^{2^m}}}$ is known, check which is the first seed (in lexicographical order) that produces (in time bounded by an appropriate polynomial) this characteristic sequence among all the possible seeds of length less than or equal to $c2^m$. As the number of seeds is bounded by 2^{c2^m}, this process can be done in polynomial time in 2^n.

4. In the above seed, if the p-th bit is 1, then the word 0^{2^n} is in the set $T2$, otherwise 0^{2^n} is not in $T2$.

This shows that $T2 \in \mathrm{P}_{\mathrm{tt}}(L)$. □

Now we present the relationship between lowtally sets and the class Full-P/log.

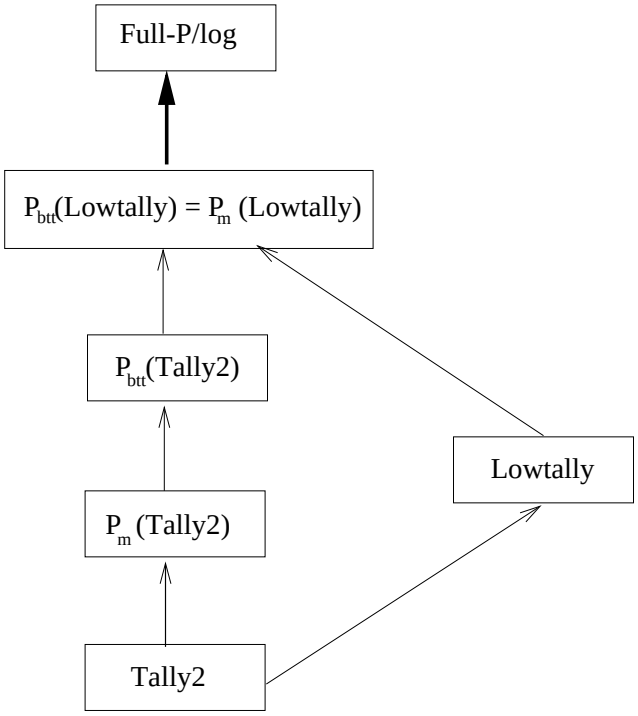

Figure 1: Reduction classes to special tally sets

Theorem 29 Lowtally = Tally $\cap$ Full-P/log.

Proof: The class Lowtally is included in Tally and, by the previous lemma, it is also included in $\mathrm{P_T}$(Tally2) = Full-P/log. Conversely, we see that any tally set in $\mathrm{P_T}$(Tally2) is in particular a lowtally language. Suppose that $L \in \mathrm{P_T}(T2)$ via a DTM M that works in time n^c. Here c is a constant and $T2$ is a tally2 set. The characteristic sequence of L up to n can be generated from M and the characteristic sequence of $T2$ up to length n^c. Since $\chi^{(T2)^{\leq n^c}}$ is relative to the set $\{0^{2^m} \mid m \in \mathbb{N}\}$ and M is a fixed DTM, the information needed is logarithmic in n. □

Up to now, we have not been able to show a precise relationship between the class Full-P/log and $\mathrm{P_{btt}}$(Lowtally) = $\mathrm{P_m}$(Lowtally).

Figure 1 describes the relationships among reduction classes. The arrows mean inclusions: the boldface arrow relates classes whose exact relationship remains open, while the others mean that the inclusions are strict. The proper infinite hierarchy of k-tt reduction classes betweem m-reduction and btt-reduction is not shown.

6.2 Relationships among Equivalence Classes

We move now to the study of the relationships between the classes of languages that are equivalent to tally2 sets and lowtally sets under various notions of reducibility.

The first problem is to determine whether the m-equivalence classes to tally2 and lowtally sets are different. The answer is provided by following lemma from Tang and Book [26] relating reducibility and inter-reducibility.

Lemma 30 [26] Let $\mathcal{C}_1$ and $\mathcal{C}_2$ two classes of sets, and let $\leq_r$ and $\leq_s$ two reducibilities with $r, s \in \{\text{m, btt, tt, T}\}$. If $\text{P}_r(\mathcal{C}_1) \neq \text{P}_s(\mathcal{C}_2)$, then $\text{E}_r(\mathcal{C}_1) \neq \text{E}_s(\mathcal{C}_2)$.

Proof: The proof is based on the operator $\oplus$. Suppose that there exists a set $A \in \text{P}_r(\mathcal{C}_1) - \text{P}_s(\mathcal{C}_2)$. Since $A \in \text{P}_r(\mathcal{C}_1)$, there exists a set $C \in \mathcal{C}_1$ such that $A \leq_r C$. Thus $A \oplus C \leq_r C$ and $C \leq_r A \oplus C$, so $A \oplus C \in \text{E}_r(\mathcal{C}_1)$. If $\text{E}_r(\mathcal{C}_1) = \text{E}_s(\mathcal{C}_2)$ then there exists a set $D \in \mathcal{C}_2$ such that $A \oplus C \leq_s D$, but this implies that $A \leq_s D$ and this is impossible. □

Combining the above lemma with the results obtained in the previous section we get the following consequences:

Corollary 31

- $\text{E}_{\text{m}}(\text{Tally2}) \subset \text{E}_{\text{m}}(\text{Lowtally})$.
- $\text{E}_{\text{btt}}(\text{Tally2}) \subset \text{E}_{\text{btt}}(\text{Lowtally})$.
- $\text{E}_{\text{m}}(\text{Tally2}) \subset \text{E}_{\text{btt}}(\text{Tally2})$.

Corollary 32 For all reducibilities $\leq_r$ with $r \in \{\text{m, btt}\}$ and $\leq_s$ with $s \in \{\text{T, tt}\}$, we have $\text{E}_r(\text{Tally2}) \subset \text{E}_s(\text{Tally2})$.

We have also the corresponding extension to all k-tt-reductions to Tally2.

As $\text{P}_{\text{T}}(\text{Tally2})$, $\text{P}_{\text{tt}}(\text{Tally2})$, $\text{P}_{\text{T}}(\text{Lowtally})$ and $\text{P}_{\text{tt}}(\text{Lowtally})$ coincide, the above argument does not work neither in the case of Turing, nor in the case of truth-table equivalence. Indeed, the equalities hold as well:

Theorem 33

$\text{E}_{\text{T}}(\text{Lowtally}) = \text{E}_{\text{T}}(\text{Tally2})$ and $\text{E}_{\text{tt}}(\text{Lowtally}) = \text{E}_{\text{tt}}(\text{Tally2})$

Proof: $\text{E}_{\text{T}}(\text{Tally2}) \subseteq \text{E}_{\text{T}}(\text{Lowtally})$ because Tally2 $\subseteq$ Lowtally. The other inclusion is not trivial. Suppose that A is Turing equivalent to a lowtally set L. That means that $A \in \text{P}_{\text{T}}(L)$ and $L \in \text{P}_{\text{T}}(A)$. By Lemma 28 there exists a tally set $T2$ such that $L \in \text{P}_{\text{tt}}(T2)$ and $T2 \in \text{P}_{\text{tt}}(L)$. Since $A \in \text{P}_{\text{T}}(L)$, $A \in \text{P}_{\text{T}}(\text{P}_{\text{tt}}(T2))$, thus $A \in \text{P}_{\text{T}}(T2)$. Conversely, since $T2 \in \text{P}_{\text{tt}}(L)$, and again by transitivity $T2 \in \text{P}_{\text{T}}(A)$.

As Lemma 28 is in terms of truth-table reduction, using the same argument as before we can prove that $\text{E}_{\text{tt}}(\text{Lowtally}) \subseteq \text{E}_{\text{tt}}(\text{Tally2})$. □

The problem of whether $\mathrm{E_m(Lowtally)} \cup \Sigma^*$ is equal to $\mathrm{E_{btt}(Lowtally)}$ is now studied. Actually, we show first that distinguishing equivalence and reducibility to lowtally sets, under the many-one reduction, would imply that P $\neq$ NP, in the same way that it is not possible to separate equivalence and reducibility to sparse sets, for many-one reductions, if P = NP (for more information see [2]). As a consequence of this result, we also obtain that separating $\mathrm{E_m(Lowtally)} \cup \Sigma^*$ and $\mathrm{E_{btt}(Lowtally)}$ becomes a difficult task too.

Theorem 34 $\mathrm{P} = \mathrm{NP} \Longrightarrow \mathrm{P_m(Lowtally)} = \mathrm{E_m(Lowtally)} \cup \Sigma^*$.

Proof: The inclusion $\mathrm{E_m(Lowtally)} \cup \Sigma^* \subseteq \mathrm{P_m(Lowtally)}$ is obvious. So, it is only necessary to see the converse. let L and LT be sets such that $L \leq_m LT$ via g, where LT is a lowtally set and $L \neq \Sigma^*$. We define the set LT', using the method from [22], and following the steps of [2], as follows:

$$LT' = \{0^{\langle l,m\rangle} \mid \exists y, |y| = l \wedge g(y) = 0^m \in LT\}$$

On the one hand, $L \leq_m LT'$ via a function h defined in this way:

$$h(y) = 0^{\langle |y|,|g(y)|\rangle}$$

Indeed, h can be calculated in polynomial time, and for all y

$$y \in L \Longleftrightarrow h(y) \in LT'$$

When $y \in L$, the word $0^{\langle |y|,|g(y)|\rangle}$ is in LT' by the definition of LT' itself. Moreover, if $h(y) = 0^{\langle |y|,|g(y)|\rangle} \in LT'$, then $g(y) \in LT$, so that $y \in L$.

On the other hand, since h is honest and using the hypothesis that P = NP, by Theorem 6, it is possible to compute an inverse function of h (denote it by f) in polynomial time. Thus f fulfills that

$$0^{\langle l,m\rangle} \in LT' \Longleftrightarrow f(0^{\langle l,m\rangle}) \in L$$

Therefore, $LT' \leq_m L$.

To finish this proof we show that, in our context, LT' is itself a lowtally set, proving that there exists a constant d, such that, for all n, $LT'^{\leq n} \in \mathrm{K}[d \log n, n^d]$. From the hypothesis that P = NP, we get the seeds for LT that are also seeds for LT', so that, when we take a seed for $LT^{\leq n}$, then we can produce $LT'^{\leq n}$. The algorithm is as follows:

input a seed s producing $\chi^{LT^{\leq n}}$;
produce from s, $\chi^{LT^{\leq n}}$;
for $i := 1$ to n **do**
    let $i = \langle l, m\rangle$;
    guess y such that $|y| = l \wedge g(y) = 0^m$;
    if $0^m \in LT$
      then $0^{\langle l,m\rangle} \in LT'$;
      else $0^{\langle l,m\rangle} \notin LT'$;
    end if;
end for;
output $LT'^{\leq n}$;

If P = NP, then this algorithm is in P. □

This result can be lifted to btt-reductions:

Theorem 35 $\mathrm{P} = \mathrm{NP} \Longrightarrow \mathrm{P}_{\mathrm{btt}}(\mathrm{Lowtally}) = \mathrm{E}_{\mathrm{btt}}(\mathrm{Lowtally})$.

Proof: By Proposition 22

$$\mathrm{P}_{\mathrm{m}}(\mathrm{Lowtally}) = \mathrm{P}_{\mathrm{btt}}(\mathrm{Lowtally})$$

By the hypothesis that P=NP and Theorem 34

$$\mathrm{P}_{\mathrm{btt}}(\mathrm{Lowtally}) = \mathrm{E}_{\mathrm{m}}(\mathrm{Lowtally}) \cup \Sigma^*$$

but it is clear that

$$\mathrm{E}_{\mathrm{m}}(\mathrm{Lowtally}) \cup \Sigma^* \subseteq \mathrm{E}_{\mathrm{btt}}(\mathrm{Lowtally})$$

Therefore $\mathrm{P}_{\mathrm{btt}}(\mathrm{Lowtally}) \subseteq \mathrm{E}_{\mathrm{btt}}(\mathrm{Lowtally})$, but the other inclusion also holds, thus both classes coincide. □

As a whole, under the assumption that P = NP, all the classes $\mathrm{E}_{\mathrm{m}}(\mathrm{Lowtally}) \cup \Sigma^*$, $\mathrm{E}_{\mathrm{btt}}(\mathrm{Lowtally})$, $\mathrm{P}_{\mathrm{m}}(\mathrm{Lowtally})$ and $\mathrm{P}_{\mathrm{btt}}(\mathrm{Lowtally})$ coincide.

In order to present the relationship between the truth-table and the Turing equivalence classes, we focus on a new approach that studies the complexity of producing advice words for sets A in Full-P/log relative to A itself. For instance, there exists a characterization of $\mathrm{E}_{\mathrm{T}}(\mathrm{Tally2})$ according to this.

Theorem 36 The following facts are equivalent:

i/ $A \in \mathrm{E}_{\mathrm{T}}(\mathrm{Tally2})$.

ii/ A has a family of logarithmic advice words that can be obtained in polynomial time making queries to A.

Proof: First we prove i/ $\Longrightarrow$ ii/.

Let A be a set in $\mathrm{E}_{\mathrm{T}}(\mathrm{Tally2})$. That is, there exists a tally2 set L fulfilling $A \in \mathrm{P}_{\mathrm{T}}(L)$ and $L \in \mathrm{P}_{\mathrm{T}}(A)$. Let n^j and n^k be respectively, the polynomials that bound the running time of the machines that query oracle L and oracle A respectively. $L^{\leq n^j}$ suffices to decide which words of length n are in A. Therefore $\chi^{L^{\leq n^j}}$ is a logarithmic advice word for length n, and because $L \in \mathrm{P}_{\mathrm{T}}(A)$, there exists a polynomial-time algorithm that with input n in unary, constructs the advice word $\chi^{L^{\leq n^j}}$ querying A.

Second, we show ii/ $\Longrightarrow$ i/. Let A be a set recognized by a family of logarithmic advice words $\{w_n \mid n \in \mathbb{N}\}$ which can be obtained in polynomial time querying A.

Using again the doubly exponential skip technique, we can keep only some selected w_n's in order to encode them into a tally2 set $T2$, in such a way that $A \in \mathrm{E}_{\mathrm{T}}(T2)$. The definition of $T2$ is as in Lemma 28 the following:

$$T2 = \{0^{2^{\left(\Sigma_{i \leq m-1}\, 2^i c\right)+p}} \mid 1 \leq p \leq 2^m c \text{ and } p\text{-th bit of } w_{2^{2^m}} \text{ is } 1\}$$

Note that in Lemma 28 we deal with logarithmic seeds instead of logarithmic advice words, but the definition of $T2$ is the same. Following similar steps as in that lemma, it is easy to see that

1. $A \in \mathrm{P_T}(T2)$, since given x as input, the advice word $w_{|x|}$, which can be produced querying $T2$, suffices to decide whether $x \in A$.
2. $T2 \in \mathrm{P_T}(A)$ using the hypothesis: to decide whether 0^{2^n} is in $T2$ suffices to look for the value m such that $2^n = 2^{c(2^m-1)+p}$ with $1 \leq p \leq c2^m$, and then to check the p-th bit of $w_{2^{2^m}}$, which is produced querying A.

Therefore $A \in \mathrm{E_T}(T2)$ and this implies that $A \in \mathrm{E_T}(\mathrm{Tally2})$. □

A similar statement can be obtained if instead of using adaptiveness and Turing reductions, nonadaptive queries and truth-table reductions are considered.

Theorem 37 The following facts are equivalent:
i/ $A \in \mathrm{E_{tt}}(\mathrm{Tally2})$.
ii/ A has a family of logarithmic advice words that can be obtained in polynomial time making queries to A in a nonadaptive way.

The next theorem provides an easy upper bound on the complexity of producing logarithmic advice words for sets in Full-P/log.

Theorem 38 For every set $A \in$ Full-P/log there exists a family of advice words for A that can be obtained in polynomial time, making logarithmically many queries to $\mathrm{NP}(A)$.

Proof: Suppose $A \in$ Full-P/log. Then

$$\forall n\ \exists w_n\ (|w_n| \leq c \log n)\ \forall x\ (|x| \leq n)\ (x \in A \Longleftrightarrow \langle x, w_n \rangle \in B)$$

where $B \in \mathrm{P}$ and c is a constant.

For each n, we can construct w_n by a prefix-search algorithm querying an oracle in $\mathrm{NP}(A)$, which is identified in more detail next:

Definition 39 Let y be a word such that $|y| \leq c \log n$. We say that y is "good for n" (in the sense of being a correct advice) if and only if

$$\forall u\ |u| \leq n\ (\langle u, y \rangle \in B \Longleftrightarrow u \in A).$$

Let GA be the following oracle set:

$$GA = \{\langle z, 0^n \rangle \mid |z| \leq c \log n \text{ and } \exists y\ z \sqsubseteq y,\ |y| \leq c \log n, \text{ and } y \text{ is "good for } n\text{"} \}$$

GA and $\overline{GA}$ are respectively in co-$\mathrm{NP}(A)$ and $\mathrm{NP}(A)$. Note that they depend both on A and B. Given the length n in unary, a good advice word corresponding to n can be produced in polynomial time querying $\overline{GA}$ logarithmically many times. □

Logarithmically many queries to $\mathrm{NP}(A)$ implies, at most, a polynomial number of different queries, which moreover can be computed in polynomial time. Therefore all of them can be asked at the beginning of the computation, and the following holds.

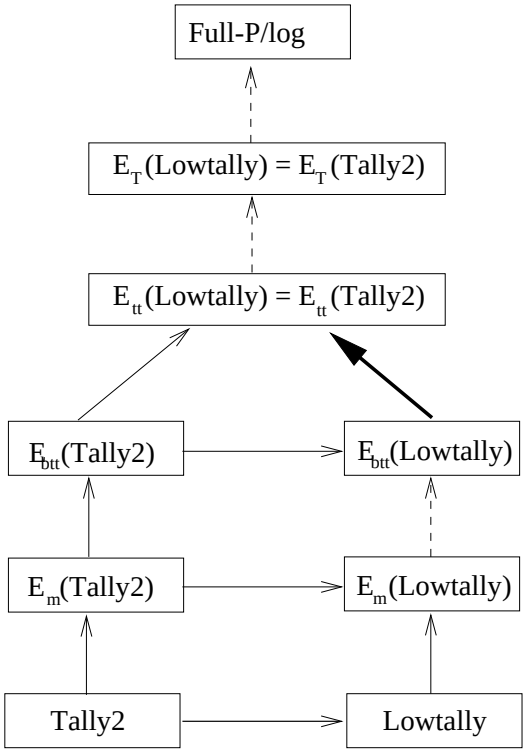

Figure 2: Equivalence classes to special tally sets

Corollary 40 For every set $A \in$ Full-P/log there exists a family of advice words for A that can be obtained in polynomial time, making queries to NP(A) in a nonadaptive way.

Wagner used a similar argument in [27], where the power of polynomial-time machines with restricted access to an NP oracle was studied.

The notions of instance complexity and the class IC[log, poly] of sets of strings with low instance complexity were introduced in [23]; we do not need the precise definition here, only two known properties. Specifically, the fact that Full-P/log is included in IC[log, poly] together with the fact that IC[log, poly] is in the first level of the extended low hierarchy (EL_1) [1, 6, 13, 21], allow us to show that:

Theorem 41 The following statements are equivalent:

i/ $A \in$ Full-P/log.

ii/ A has a family of logarithmic advice words that can be obtained in polynomial time making queries to $(A \oplus \mathrm{SAT})$ in a nonadaptive way to A.

Proof: The nontrivial direction is from i/ to ii/, and it is a consequence of previous results from [4, 5][2]. Actually, in [5] it was proven that IC[log, poly] is in the first level of the extended hierarchy. That is, $\mathrm{NP}(A) \subseteq \mathrm{P}(A \oplus \mathrm{SAT})$ for all sets A in IC[log, poly]. Moreover, the proof of this shows that for each language L in $\mathrm{NP}(A)$ there exists a deterministic algorithm that decides L in polynomial time querying $(A \oplus \mathrm{SAT})$, and although it has an adaptive access to SAT, the queries to A are made in a nonadaptive way.

Therefore ii/ holds, since Full-P/log is included in IC[log, poly]. □

Now we apply the hypothesis that P = NP: all the queries made by the algorithm to oracle SAT $\in$ NP can be replaced by a polynomial-time computation. This together with previous results suffice to see the following.

Theorem 42 Full-P/log = E_{T}(Tally2) = E_{tt}(Tally2), if P = NP.

Figure 2 presents the results regarding equivalence classes. Now the discontinuous arrows appear when, under assumption that P = NP, the inclusions turn out to be equalities. There is again an open question, indicated by the boldface arrow; note that it matches (and actually refines) its corresponding "open question" arrow in figure 1. We believe that a different toolkit is necessary to close our two remaining fully open questions. On the other hand, forthcoming work by the authors, jointly with H. Buhrman (and constituting the archival version of [7]) reduces optimally the strength of the necessary complexity-theoretic condition P = NP used in several of our theorems.

Acknowledgment

We would like to thank Ricard Gavaldà for helpful discussions, including alternative proofs of theorem 42, and for pointing out that a proof in [15] of the fact that P_{m}(Lowtally) is different from P_{T}(Lowtally) was erroneous.

References

1. E. Allender and L. Hemachandra. Lower Bounds for the Low Hierarchy. *Journal of the Association for Computing Machinery*, 39(1):234–250, 1992.

2. E. Allender, L. Hemachandra, M. Ogiwara, and O. Watanabe. Relating Equivalence and Reducibility to Sparse Sets. *SIAM Journal on Computing*, 21(3):521–539, 1992.

3. E. Allender and O. Watanabe. Kolmogorov Complexity and Degrees of Tally Sets. *Information and Computation*, 86(2):160–178, 1990.

[2] A direct proof exists although we prefer to mention previous results.

4. V. Arvind, Y. Han, L. Hemachandra, J. Koebler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, and T. Thierauf. Reductions to Sets of Low Information Content. In *Complexity Theory*, pages 1–46. Cambridge University Press, eds K. Ambos-Spies, S. Homer, U. Schöning, 1993.

5. V. Arvind, J. Köbler, and M. Mundhenk. Upper Bounds on the Complexity of Sparse and Tally Descriptions. *Mathematical Systems Theory*, 29(1):63–94, 1996.

6. J. Balcázar, R. Book, and U. Schöning. Sparse Sets, Lowness, and Highness. *SIAM Journal on Computing*, 15(3):739–747, 1986.

7. J. Balcázar, H. Buhrman, and M. Hermo. Learnability of Kolmogorov-Easy Circuit Expression Via Queries. In *Computational Learning Theory, Proc. Eurocolt'95*, volume 904, pages 112–124. Lecture Notes in Artificial Intelligence. Springer-Verlag, 1995.

8. J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Second Edition. Texts in Theoretical Computer Science. Springer-Verlag, 1995.

9. J. Balcázar, R. Gavaldà, H. Siegelmann, and E. Sontag. Some Structural Complexity Aspects of Neural Computation. In *Proc. Structure in Complexity Theory 8th annual conference*, pages 253–265. IEEE Computer Society Press, 1993.

10. J. Balcázar, M. Hermo, and E. Mayordomo. Characterizations of Logarithmic Advice Complexity Classes. In *Proc. of the IFIP 12th World Computer Congress*, volume 1, pages 315–321. North-Holland, 1992.

11. J. Balcázar and U. Schöning. Logarithmic Advice Classes. *Theoretical Computer Science*, 99:279–290, 1992.

12. R. Book and K. Ko. On Sets Truth-table Reducible to Sparse Sets. *SIAM Journal on Computing*, 17(5):903–919, 1988.

13. J. Castro and C. Seara. Complexity Classes Between Θ_k^P and Δ_k^P. *Informatique Théorique et Applications*, 30(2):101–121, 1996.

14. J. Hartmanis. Generalized Kolmogorov Complexity and the Structure of Feasible Computations. In *Proc. 24th IEEE Symposium on Foundations of Computer Science*, pages 439–445, 1983.

15. M. Hermo. Degrees and Reducibilities of Easy Tally Sets. In *Proc. 19th Symposium on Mathematical Foundations of Computer Science*, volume 841, pages 403–412. Lecture Notes in Computer Science. Springer-Verlag, 1994.

16. M. Hermo. *Nonuniform Complexity Classes with Sub-Linear Advice Functions.* Doctoral Thesis, 1996.

17. R. Karp and R. Lipton. Some Connections between Nonuniform and Uniform Complexity Classes. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 302–309, 1980.

18. K. Ko. On the Notion of Infinite Pseudorandom Sequences. *Theoretical Computer Science*, 48:9–13, 1986.

19. K. Ko. On Helping by Robust Oracle Machines. *Theoretical Computer Science*, 52:15–36, 1987.

20. J. Köbler, U. Schöning, and K. Wagner. The Difference and Truth-Table Hierarchy for NP. *Informatique Théorique et Applications*, 21(4):419–435, 1987.

21. T. Long and M. Sheu. A Refinement of the Low and High Hierarchies. *Mathematical Systems Theory*, 28(4):299–327, 1995.

22. S. Mahaney. Sparse Complete Sets for NP: Solution of a Conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25:130–143, 1982.

23. P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance Complexity. *Journal of the Association for Computing Machinery*, 41:96–122, 1994.

24. H. Siegelmann and E. Sontag. Analog Computation via Neural Networks. *Theoretical Computer Science*, 131:331–360, 1994.

25. M. Sipser. A Complexity-Theoretic Approach to Randomness. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

26. S. Tang and R. Book. Reducibilities on Tally and Sparse Sets. *Informatique Théorique et Applications*, 25(3):293–302, 1991.

27. K. Wagner. Bounded Query Classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.