Characterizations of logarithmic advice complexity classes

José L. Balcázar*, Montserrat Hermo†, and Elvira Mayordomo*

Conference submission: extended abstract plus appendix, 3–1–92

Abstract

The complexity classes P/log and Full-P/log, corresponding to the two standard forms of logarithmic advice for polynomial time, are studied. The novel proof technique of "doubly exponential skip" is introduced, and characterizations for these classes are found in terms of several other concepts, among them easy-to-describe boolean circuits and reduction classes of tally sets with high regularity. Similar results hold for many other complexity classes.

In this extended abstract most of the proofs are deferred to the appendix, where they are provided for the interested reader but are intended to be read discretionarily.

1. Introduction

The study of nonuniform complexity classes stems from the comparison between uniform models of computation, in which a program is valid for arbitrarily long inputs, and nonuniform models in which each program is valid only for inputs of a fixed length. There are many well-known models for both. Typical examples of uniform models are the Turing machine and other equivalent models such as the RAM; the most important representative of the nonuniform family is the boolean circuit model, although there are many others.

---

† Facultad de Informática, Universidad del País Vasco, Apdo. 649, 20080 San Sebastián, Spain.

* Departament de Llenguatges i Sistemes Informàtics, Univ. Politècnica de Catalunya (ed. FIB), 08028 Barcelona, Spain. E-mail contact: balqui@lsi.upc.es. This research was partially supported by the ESPRIT-II Basic Research Actions Program of the European Community under contract no. 3075 (project ALCOM) and by a Spanish Government grant FPI PN90.

There are several connections between them. Uniformity conditions can be set on the nonuniform model to obtain uniform complexity classes, or vice-versa advice functions can be used to extend uniform classes into nonuniform ones. Our framework is this second approach.

In this context, the most widely studied class is P/poly of the sets decidable in polynomial time with the help of polynomially long advice strings. This class can be characterized in various manners (see below), one of them as the class of languages decidable by boolean circuits whose size grows polynomially on the length of the input. This important class has been studied in depth (see for instance [1] and the references there).

Advice functions were introduced in [5], where two main classes were selected as bounds for the length of the advice words: polynomial bounds and logarithmic bounds. We consider here the class P/log corresponding to polynomial time with the help of logarithmically long advice strings. After the results of [5], additional results regarding this class have been much more infrequent. In [1] a characterization is given in terms of uniformly computable functions. Other results were presented in [2]. One of the points we address here is the question of finding a characterization of this class in terms of circuits.

For this purpose, the concept of languages having "small circuits with easy descriptions" was introduced in [5], intending to characterize P/log with them. Such easy descriptions are again circuits, of logarithmic size, describing the interconnection pattern of the circuit. However it was proved only that these sets lie between P/log and P/$O(\log n * \log\log n)$. Here we explain why this concept does not characterize precisely P/log: we show that it corresponds exactly to P/$O(\log n * \log\log n)$, which can be proved to be a different class.

Our proof shows that the reason for the circuits with easy descriptions not corresponding to P/log is essentially the Shannon-Lupanov effect, by which boolean circuits are able to encode a nontrivial amount of information into their interconnection pattern. Thus, this suggests that in order to characterize P/log, Turing programs must be used instead of logarithmic size circuits. This idea corresponds to the concept of resource-bounded Kolmogorov complexity; indeed, we present here two characterizations of P/log with circuits of low resource-bounded Kolmogorov complexity. (The results of this section were announced in the internal report [4]).

Then we essay to characterize P/log as the reduction class of some special family of sets, in the spirit of the existing characterizations of P/poly. We will easily see that such task is not possible since P/log is not closed under polynomial time Turing reducibility.

The natural answer (take the closure of the class) is not really an answer: it brings us back to P/poly. Thus we consider a modification of P/log proposed in [7] and also studied in [2], in which the closure under polynomial time reductions is obtained via a more restrictive condition in the definition. Departing from the original name (which may mislead the reader due to its similarity to other different concepts of structural complexity theory), we denote it Full-P/log. To our knowledge no characterization of this class as a reduction class has been published before, and indeed the task is nontrivial.

We present here several such characterizations. 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. 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, tally sets that are regular in a resource-bounded Kolmogorov complexity sense, and logarithmically sparse-capturable sets. We obtain also characterizations in terms of oracle Turing machines with bounded query tape.

2. Preliminaries

Complexity classes are sets of formal languages. A formal language is a set of words over the alphabet $\{0,1\}$. By standard encoding methods, any other finite, fixed alphabet could be assumed if necessary provided that it has at least two different symbols. We denote by $w_{1:k}$ the word consisting of the first k symbols of w; this is valid too when w is an infinite sequence. The length of a word w is denoted $|w|$, and overloading the notation we denote by $|A|$ the cardinality of the finite set A.

For any alphabet Σ, Σ^* is the set of all words over Σ; $\Sigma^{\leq n}$ is the set of all words of length at most n, and $A^{\leq n} = A \cap \Sigma^{\leq n}$; similarly we have $\Sigma^{=n}$ and $A^{=n}$. Here we will use in particular the alphabet $\Sigma = \{0,1\}$ and $\Sigma = \{0\}$. A tally set is a set of words over this single letter alphabet $\{0\}$. A sparse set is a set having at most polynomially many strings of each length: $|A^{\leq n}| \leq p(n)$ for some polynomial p. The strings of Σ^* are ordered by lengths and lexicographically within each length.

If A is a set of words, $\chi_A \in \{0,1\}^\infty$ is the characteristic sequence of A, defined in the standard way: the i^{th} bit of the sequence is 1 if and only if the i^{th} word of Σ^* is in A. Similarly, $\chi_{A^{\leq n}}$ is the characteristic sequence of $A^{\leq n}$ relative to $\Sigma^{\leq n}$. In both cases Σ is taken as the smallest alphabet containing all the symbols occurring in words of A, so that for a tally set T, χ_T denotes the characteristic sequence of T relative to $\{0\}^*$.

Throughout this paper, $\log n$ means the function $\max(1, \lceil \log_2 n \rceil)$.

Many of our complexity classes can be defined in a completely standard way by time-bounded or space-bounded multitape Turing machines. We will focus on in the class P solvable by deterministic polynomial time algorithms. Occasionally we mention other classes such as PSPACE, which corresponds to deterministic polynomial space algorithms, and NP, which corresponds to nondeterministic polynomial time algorithms.

Oracle Turing machines are used to define reducibilities between sets. These machines are furnished with an unbounded oracle tape, on which the machine writes down a query word. In a single query step the machine is informed of whether the query word is in the oracle and the query tape is erased. We denote $\mathrm{P}(A)$ the class of all sets that can be decided in polynomial time with oracle A. Further details can be found e.g. in [1].

Boolean circuits are finite directed acyclic graphs with nodes ("gates") of indegree up to 2. Nodes of indegree zero are the input nodes $x_1, x_2, \ldots, x_n$ or constant gates labelled 0 or 1; nodes with indegree one are labelled $\neg$, and nodes with indegree two are labelled with a two-input boolean function. It is well-known that we can restrict these functions to $\wedge$ and $\vee$ if necessary. One of the nodes is specified as output node. A circuit with $|x|$ inputs accepts a word x if, after giving the values of the individual bits of x to the inputs and evaluating the circuit, the output evaluates to 1. The size of a circuit is the number of gates.

Each gate i of a circuit is described by a tuple $\langle \hat{\imath}, \hat{\jmath}, \hat{k}, B \rangle$ specifying the name or index $\hat{\imath}$ of the gate, the names of its input gates $\hat{\jmath}, \hat{k}$, and the boolean function B computed at gate i. In this way we can associate to a circuit C its connection language B_C defined as the finite set of all the tuples correctly describing gates of C. We identify the circuit with the sequence of all tuples describing its gates.

The circuit value problem is the set CVP of pairs $\langle x, c \rangle$ where c is a circuit with $|x|$ inputs that accepts x. It is not difficult to design a polynomial time algorithm for evaluating a circuit, so that $CVP \in P$.

There are various forms of simulating the computation of a Turing machine by means of a boolean circuit. In particular, we will use a simulation due to Savage:

Theorem 1. Given a Turing machine M bounded in time by the function $T(n)$ and an input x of length n, there is a circuit of size $T(n)^2$ which accepts x if and only if M does.

More effective simulations exist, reducing the size of the circuit by a nontrivial factor. The interest of this simulation is that the circuit is exceedingly regular, consisting essentially of a two-dimensional array of replications of a fixed finite subcircuit obtained from the transition function of the Turing machine. See [1] and the references there for more information on this construction and on other connections between circuits and Turing machines.

The notion of advice function was introduced in [5] to provide connections between uniform computation models such as resource-bounded Turing machines and nonuniform computation models such as bounded-size boolean circuits.

Definition 2. Given a class of sets C and a class of bounding functions F, the class C/F is formed by the sets A such that

$$\forall n \; \exists w \; (|w| \leq h(n)) \; \forall x \; (|x| = n) \; x \in A \iff \langle x, w \rangle \in B$$

where $B \in C$ and $h \in F$.

The words w mentioned in the definition are frequently called “advice words”. The corresponding Skolem function mapping each n into an appropriate advice w_n for length n is called “advice function”. C is usually a uniform complexity class, most frequently P, whereas the class poly $= \{n^{O(1)}\}$ of polynomials and the class log $= O(\log n)$ of logarithms are the most frequent bounding functions. In particular, [5] focused on the study of the classes P/poly and P/log, and proved that for certain problems, the hypothesis of being in nonuniform classes has implications on the structure of uniform classes.

Some interesting known facts about P/poly are the following:

Theorem 3. The following classes coincide:

- i/ P/poly.
- ii/ $\bigcup_S P(S)$ where S is sparse.
- iii/ $\bigcup_T P(T)$ where T is a tally set.
- iv/ $\bigcup_B P(B)$ where B is arbitrary and the length of the queries of the polynomial time machines is bounded by $O(\log n)$.
- v/ The class of sets A such that for all n the set $A^{=n}$ can be decided by a circuit of size polynomial in n.

Other characterizations exist. Observe that as a consequence it immediately follows that P/poly is closed under polynomial time Turing reducibility.

The goal of this paper is to try to obtain similar characterizations of P/log, for which much less is known. A characterization in terms of uniformly generated circuits is given in [1], but no results similar to theorem 3. If we try to obtain them, we find an immediate difficulty in the fact that P/log is not closed under polynomial time Turing reducibility. Indeed, it is immediate to see that every tally set is in P/log, and therefore the closure of P/log under this reducibility is already P/poly. It is easy to prove (see below) that P/log and P/poly differ.

The need for a variant of P/log closed under this reducibility prompted Ko to introduce the following class [7]:

Definition 4. A set A is in Full-P/log if

$$\forall n \; \exists w \; (|w| \leq c \log n) \; \forall x \; (|x| \leq n) \; x \in A \iff \langle x, w \rangle \in B$$

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

The name given to this class in [7] was STRONG-P/log; we prefer to avoid this name since there are in complexity theory "strong nondeterministic machines" and their definition has no relation whatsoever with the concept we study here.

The definition is quite natural. For instance, if the definition of P/poly is changed according to this one, using polynomially long advices, one gets the same class P/poly again. Note that now the advice corresponding to a length is valid also as an advice for all the smaller lengths. It is easy to see that Full-P/log is closed under polynomial time Turing-reducibility, so that it is different from P/log.

We have to introduce finally a tool we will use in this paper: resource-bounded Kolmogorov complexity. Fix any universal Turing machine U. Denote by $U(y) = x$ the fact that on input y the machine U outputs x and halts. Define the sets of bounded Kolmogorov complexity strings $K[f(n), g(n)]$ and $KS[f(n), g(n)]$ as follows:

Definition 5.

i/ $x \in K[f(n), g(n)]$ if there exists y, $|y| \leq f(|x|)$, such that $U(\langle |x|, y \rangle) = x$ in at most $g(|x|)$ steps. If no condition is imposed on the running time we simply say $x \in K[f(n)]$.

ii/ $x \in KS[f(n), g(n)]$ if there exists y, $|y| \leq f(|x|)$, such that $U(\langle |x|, y \rangle) = x$ using at most $g(|x|)$ space.

Observe that we explicitly provide the universal machine with the length of the word to be constructed. This is required to work under certain bounds, although many of our results in this paper would hold even if the machine does not know the length beforehand.

A simple counting argument shows that if f is sublinear then not all words are in $K[f(n), g(n)]$. We can also talk of low Kolmogorov complexity of sets. We focus on the two classes we will use in this paper.

Definition 6.

i/ $A \in K[\log, \text{poly}]$ if there exist constants c and k such that $A \subseteq K[c \log n, n^k]$.

ii/ $A \in KS[\log, \log]$ if there exist constants c and d such that $A \subseteq KS[c \log n, d \log n]$.

Observe that here we are defining classes of sets, as opposed to the sets of strings presented in definition 5. Since logarithmic space implies polynomial time but not vice-versa, $KS[\log, \log]$ is more restrictive than $K[\log, \text{poly}]$.

As an easy example of application of this concept, let us state a hierarchy theorem for nonuniform classes depending on the amount of advice available. This result will be useful later on. The proof is sketched in the appendix, and is based on an argument from [2].

Proposition 7. Let C and D be recursive classes, with $\text{DTIME}(O(n))$ included in D. Let F and G be function classes such that there exists $s \in G$, with $s \in o(2^n)$, such that for all $r \in F$ it holds $r \in o(s)$. Then D/G and C/F are different.

3. Kolmogorov-simple circuits and easy descriptions

The purpose of this section is to discuss characterizations of the logarithmic advice class P/log. Characterizations of Full-P/log will be presented in the next sections. Here we show that a characterization of P/log exists using Turing descriptions, and prove that a previous suggestion in terms of circuit descriptions cannot achieve the desired characterization.

The idea from which we start is quite natural and was already put forward in [5]: if polynomial advice corresponds to polynomial size circuits because the advice describes the circuit, logarithmic advice requires such a circuit to be easy to describe. Thus there the following definition is proposed:

Definition 8. [5] A language S has small circuits with easy descriptions if and only if there are sequences of circuits $\{C_n\}$ and $\{D_n\}$, a polynomial p and a constant c such that:

i/ for all n, C_n has size at most $p(n)$ and accepts $S^{=n}$;

ii/ for all n, D_n has size at most $c \log n$ and accepts B_{C_n}.

In this way it is required that the circuits for S can be described shortly by descriptions that are again circuits. Notice that it makes no sense to require that the circuits D_n have less than logarithmic number of gates since an input to D_n is an instance of B_{C_n}, and thus simply the input gates are already logarithmically many.

We address below the reasons why this proposal is unsatisfactory for the purposes of characterizing P/log. However, a natural alternative is to consider Turing programs to describe the circuits, instead of using circuits again. Our characterization of P/log follows this idea, by proving that resource-bounded Kolmogorov complexity provides appropriate bounds on the descriptions of circuits. We obtain the following characterization of P/log in terms of resource-bounded Kolmogorov complexity:

Theorem 9. The following are equivalent:

i/ $A \in \mathrm{P/log}$.

ii/ A is accepted by a family $\{C_n\}$ of circuits of polynomial size such that $\{C_n\} \in KS[\log, \log]$.

iii/ A is accepted by a family $\{C_n\}$ of circuits of polynomial size such that $\{C_n\} \in K[\log, \mathrm{poly}]$.

The proof uses standard arguments and is given in the appendix.

Thus polynomial size circuits with Kolmogorov simple descriptions indeed characterize the class P/log. Now, what holds for the "easy descriptions" of [5]? As stated there, it can be seen easily that,

Theorem 10. [5] If $S \in \mathrm{P/log}$ then S has small circuits with easy descriptions.

However, this is not a full characterization. When trying to prove the converse, it seems that one has to be satisfied with the following partial achievement:

Theorem 11. [5] Sets having small circuits with easy descriptions are in $\mathrm{P}/O(\log n * \log\log n)$.

The problem is that the encoding of D_n requires $O(\log n * \log\log n)$ space, since the number of gates is logarithmic but each gate needs $O(\log\log n)$ bits to be described. As mentioned above, reducing still further the number of gates to accomodate the circuit in a logarithmic description is infeasible.*

We will explain next why the converse of theorem 10 seemed difficult to achieve, by actually proving that it does not hold. We do it by exactly characterizing the class of sets having small circuits with easy descriptions as $\mathrm{P}/O(\log n * \log\log n)$, since we show that the converse of theorem 11 holds. By proposition 7, we know that P/log and $\mathrm{P}/O(\log n * \log\log n)$ differ.

Theorem 12. Let $A \in \mathrm{P}/O(\log n * \log\log n)$. Then A has small circuits with easy descriptions.

The proof is given in the appendix.

---

* The journal version [6] of the conference paper [5] does not treat the class P/log.

Thus, essentially easy descriptions by means of circuits allow us to save in the encoding exactly as much information as they additionally require to be written down. From theorems 11 and 12 we obtain:

Corollary 13. The following are equivalent:
i/ $A \in P/O(\log n * \log(\log n))$.
ii/ A has small circuits with easy descriptions.

Since by proposition 7 in the preliminaries we know that P/log and P/$O(\log n * \log\log n)$ differ, it follows that small circuits with easy descriptions do not characterize P/log, as they were intended for when proposed.

We can generalize these results using the same arguments, with small adjustments. The appendix lists some characterizations which can be obtained in this way. Also, results similar to theorem 9 can be obtained for many other nonuniform complexity classes, such as PSPACE/ log, which corresponds to sets that can be described by quantified boolean formulas of low Kolmogorov complexity, NP/ log, which corresponds to small generators with low Kolmogorov complexity, and many others.

4. Prefix-closed advice

This section discusses a technical variant of full logarithmic advice, in which the advice words corresponding to the various lengths are not independent but highly correlated. We prove here a simple but interesting characterization. This weaker notion will simplify the discussion of the standard concept of full logarithmic advice in the next section.

Definition 14. A set A has prefix-closed logarithmic full advice, briefly $A \in$ Pref-P/log, if A is in Full-P/log via an infinite sequence of advices 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 w, so that, for all n, $w_n = w_{1:c\log n}$, the first $c \log n$ bits of w. 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.

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 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}.

Let us see how this somewhat artificially defined class can be characterized by polynomial time Turing reduction classes of regular tally sets, as well as using bounded query machines. Here (and later on as well) the symbol T will always denote a tally set.

Theorem 15. The following classes of languages are the same:
i/ $\bigcup_T P(T)$ where $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$.
ii/ $\bigcup_T P(T)$ via a polynomial time machine whose queries have length at most $O(\log n)$.
iii/ $\bigcup_B P(B)$ via a polynomial time machine whose queries have length at most $\log\log n + O(1)$.
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 accomodated 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. The proof is given in the appendix.

5. The "doubly exponential skip" technique

This section presents one of the main contributions of the paper, 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. Two examples of application of this technique will be presented, both giving 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 class seemingly more powerful defined in terms of Kolmogorov-regular tally sets.

Theorem 16. Full-P/log = Pref-P/log.

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 the same logarithmic length bounds, and obeying the restriction that each advice word is a prefix of all the forthcoming ones. By the discussion in the previous section, it hence turns out that in the context of advice information, constantly many bits added exponentially often give as much computational power as logarithmically many bits changed linearly often.

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 $B \in$ P and a sequence of advice words $\{w_n \mid n \in \mathbb{N}\}$ with $w_n \leq k \log n$ 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 $\bigcup_T P(T)$ where $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$. Thus we will define a tally set T containing only words of length a power of 2, and will prove that $A \in P(T)$.

It follows from results in [2] that the class Full-P/log 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 with 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 as the presence or absence of a word of the form 0^{2^m} in the set T.

The advice corresponding to length 2^{2^0}, respectively $2^{2^1} \ldots 2^{2^m}$, has 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 $2^{2^0} = 2$. The second 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 T by the elements $0^{2^{k+2k+2^2k+\cdots+2^{m-1}k+1}}$ until $0^{2^{k+2k+2^2k+\cdots+2^m k}}$.

So let T be

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

We prove first that $A \in P(T)$. On input x, find an integer m such that $2^{2^{m-1}} < |x| \leq 2^{2^m}$. This can be done in linear time. Observe that 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

$0^{2^{\left(\Sigma_{i\leq m-1}2^i k\right)+p}} \in T$ 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 see that the queries can be asked in polynomial time; it suffices to see 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 $2^{k+2k+2^2k+\cdots+2^m k} = 2^{(1+2+2^2+\cdots+2^m)k} < 2^{2^{m+1}k}$. 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, small constants d and d'. So $A \in P(T)$. ∎

We have chosen to keep exactly the 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 advices of logarithmic length to be encoded, i.e. a total of $\log^2 n$ 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 advices with triply exponential gaps, skipping all advices but those corresponding to lengths $2^{2^{2^m}}$, then these advices are too large although there are less 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 it is $n^{\log n}$ 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 two. Many other similar notions of tally sets with regularities can be proposed, but among them there is one particularly natural after the results we have presented in previous sections: 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 16 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, T denotes always a tally set.

Theorem 17. The following two classes coincide:

i/ Full-P/log.

ii/ $\bigcup_T P(T)$ where there exist positive constants c and d such that, for all n, $\chi_{T^{\leq n}} \in K[c \log n, n^d]$.

The proof is given in the appendix.

Thus, for polynomial time machines, regular tally oracle sets in the sense that their words are all of length a power of 2 have exactly the same power as Kolmogorov-regular tally 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, can be replaced by a much simpler oracle which adds constantly many bits exponentially often.

Now we have several ways of characterizing Full-P/log in terms of tally oracles. Comparing with theorem 3, it is natural to ask whether there is a restriction on sparse oracles achieving the analogous results. The answer is affirmative, although the notion we have to use to restrict sparse sets is somewhat artificial. Let us define some terms.

Definition 18. A set A is p-printable if and only if there is a polynomially computable function f from $\{0\}^*$ to Σ^* such that $f(0^n)$ is a word coding the subset $A^{\leq n}$.

Polynomial printability was introduced in [3]. Printability obviously implies sparseness. It is is not the appropriate notion to characterize P/log since every printable set is in P. However, it allows us to define a special kind of sparse sets which will give us the characterization:

Definition 19. A set S is logarithmically sparse-capturable (briefly, log-capturable) if and only if S is a subset of a p-printable set R such that $|R^{\leq n}| \leq c \log n$ for a constant c.

Immediate examples of log-capturable sets are the tally sets T used frequently up to now, for which $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$. Here $\{0^{2^k} \mid k \in \mathbb{N}\}$ plays the role of the p-printable set R. We prove next that, up to polynomial time Turing equivalence, these tally sets are representatives of all log-capturable sets. From this fact we will immediately obtain a characterization of Full-P/log in terms of log-capturability.

Proposition 20. For every log-capturable set S, there is a tally set $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$ such that $S \in \mathrm{P}(T)$.

Proof . Let S be log-capturable, being R a p-printable superset of S with logarithmic density. It suffices to define the following tally set:

$$T = \{0^{2^k} \mid \text{the } k^{th} \text{ element of } R \text{ is in } S\}$$

Now it is easy to check that instead of the sparse set S we can use T as oracle for any arbitrary polynomial time Turing machine. ∎

Corollary 21. The following two classes coincide:

i/ Full-P/log.

ii/ $\bigcup_S P(S)$ where S is log-capturable.

Proof . Use again the characterization of Full-P/log as $\bigcup_T P(T)$ where $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$, and apply the fact that every such tally set is log-capturable and the previous proposition. ∎

6. Acknowledgements

Thanks are due to a number of people; particularly we want to mention Juris Hartmanis, Marius Zimand, and Harry Buhrman for suggesting some of the problems addressed here and providing helpful hints, Kim Gabarró and Ron Book for interesting discussions, Ricard Gavaldà and Jacobo Torán for proofreading, and Juanjo Nieto for his patience at lunch time.

7. References

[1] J. L. Balcázar, J. Díaz, J. Gabarró: *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, Vol. 11, Springer-Verlag 1988.

[2] J. L. Balcázar, U. Schöning: Logarithmic advice classes. *Theoretical Computer Science*, to appear.

[3] J. Hartmanis, Y. Yesha: Computation times of NP sets of different densities. *Theoretical Computer Science* 34 (1984), 17–32.

[4] M. Hermo, E. Mayordomo: Polynomial size circuits with low resource-bounded Kolmogorov complexity. Report LSI–91–20 (1991).

[5] R. M. Karp, R. J. Lipton: Some connections between nonuniform and uniform complexity classes. In: *Proc. 12th Annual Symposium on Theory of Computing* (1980), 302–309.

[6] R. M. Karp, R. J. Lipton: Turing machines that take advice. *L'Enseignement Mathématique* (series 2), 28 (1982), 191–209.

[7] K.-I Ko: On helping by robust oracle machines. *Theoretical Computer Science* 52 (1987), 15–36.

[8] I. Wegener: *The Complexity of Boolean Functions.* Wiley-Teubner Series in Computer Science, 1987.

8. Appendix

Proof of proposition 7

We define a set A in D/G but not in C/F. For each m let γ_m be a string such that $|\gamma_m| = s(m)$ but $\gamma_m \notin K[n]$, i.e. the universal machine cannot generate it from data shorter than γ_m itself. Let the characteristic string of $A^{=m}$ be $\gamma_m 10^{2^m - |\gamma_m| - 1}$.
It is easy to prove that A is in D/G, by taking γ_m itself as advice; however, A is not in C/F, since otherwise the universal machine could construct γ_m from an advice bounded by some $r(m)$ plus a program of constant length, contradicting the incompressibility of γ_m. ■

As an immediate corollary we obtain that the classes $\mathrm{P}/O(1)$, $\mathrm{P}/O(\log^{(m)} n)$, $\mathrm{P}/\log$, $\mathrm{P}/O(\log n * \log^{(m)} n)$, $\mathrm{P}/O(\log n * \log\log n)$, $\mathrm{P}/O((\log n)^k)$, and P/poly are all different. Here $\log^{(m)} n$ denotes the m-fold iteration of the function $\log n$.

Proof of theorem 9

To prove that i/ implies ii/, we start from a set $A \in \mathrm{P}/\log$. Consider any length n, with its associated advice w_n with $|w_n| = c \log n$, so that for all x of length n, $x \in A \iff \langle x, w_n \rangle \in B$ where $B \in \mathrm{P}$. Consider the polynomial time machine M which on inputs x and $w_{|x|}$ computes the pairing $\langle x, w_{|x|} \rangle$ and then checks whether it is in B. Observe that $|\langle x, w_{|x|} \rangle|$ only depends on $|x|$. Applying to M the construction described in theorem 1 in the preliminaries, and fixing the inputs corresponding to w_n, we obtain a circuit C_n that accepts exactly the set $A^{=n}$.
Let us discuss the resources needed to construct this circuit. As indicated, given the (fixed) polynomial time machine M and given n, it is possible to construct the circuit within $\log n$ space. The construction needs to know also the actual values of the bits of w_n to substitute for the corresponding inputs. The total amount of information required as seed to produce C_n is given by n and w_n, both of length $O(\log n)$, and since $O(\log n)$ space suffices to perform the construction, $\{C_n\} \in KS[\log, \log]$.
It is immediate to see that ii/ implies iii/. To prove that iii/ implies i/, let $\{C_n\}$ be the polynomial size family that accepts A. Since $\{C_n\} \in K[\log, \mathrm{poly}]$, there exists for each n a logarithmically long seed w_n, say $|w_n| \leq c \log n$, such that $U(w_n) = C_n$ in time bounded by n^k; both c and k are constants independent of n. Change U into U' by adding a clock that stops it after exhausting n^k steps, so that still $U'(w_n) = C_n$ but now U' is guaranteed to halt in polynomial time.

Define the following set:

$$B = \{\langle x, y\rangle \mid \langle x, z\rangle \in CVP \text{ where } z = U'(y)\}$$

Since CVP is in P and U' halts in polynomial time, it is clear that B is in P. Now $x \in A \iff \langle x, w_{|x|}\rangle \in B$ and therefore we have $A \in \mathrm{P}/\log$. ■

Proof of theorem 12

The description of the polynomial time computation will be encoded into a circuit appealing once more to theorem 1. Again, we simply hardwire the advice w_n into the circuit. In this way we obtain a circuit C_n accepting exactly $A^{=n}$. Assume that $|w_n| \leq c \log n * \log\log n$. Assume also that the constant gates of C_n corresponding to the advice are tagged with the lowest numbers in the enumeration of the gates. Now we have to exhibit a new circuit D_n describing the circuit C_n using only $O(\log n)$ gates. The parts of C_n corresponding to the simulation of the computation pose no problem: the circuit has an extremely regular structure, as mentioned before, and given the name of a gate it is an easy task to find out its type and connections. Thus given $\langle \hat{\imath}, \hat{\jmath}, \hat{k}, B\rangle$, if we identify that the ith gate is not a constant from the advice, then we can easily decide whether it is in B_{C_n}.

Let us see how to handle the advice, which might seem too long. We have to make the encoding of the circuit "swallow" somehow the additional $\log\log n$ factor. Fortunately there is an important result in boolean complexity which comes to help us: the Shannon-Lupanov effect, that states that boolean functions on m inputs, or equivalently, truth-tables of size 2^m, can be computed by circuits of size smaller than 2^m. Let us state it as found in chapter 4 of [8] (theorem 2.2): Given a boolean function f from $\{0,1\}^m$ to $\{0,1\}$, the size of the smallest circuit $C(f)$ computing it is bounded by $2^m m^{-1} + o(2^m m^{-1})$. Actually a bound of $O(2^m m^{-1})$ already suffices for our purposes.

Thus given $\langle \hat{\imath}, \hat{\jmath}, \hat{k}, B\rangle$ where $\hat{\imath}$ is a constant gate from the advice, notice that $\hat{\imath}$ has only $\log\log n + \log\log\log n + O(1)$ significant bits, since advice gates had the lowest numbers. The bits regarding gates j and k are irrelevant, but B indicates a value of zero or one for the gate i and we have to check whether this is correct.

Thus, for some constant c, D_n has to compute a function

$$f : \{0,1\}^{\log\log n + \log\log\log n + c} \to \{0,1\}$$

Appealing to the Shannon-Lupanov effect, it is possible to compute it by means of a circuit of size bounded by

$$O\left(\frac{\log n * \log\log n}{\log\log n + \log\log\log n}\right) \leq O(\log n)$$

Hence, the total number of gates of the corresponding circuit D_n is logarithmic, so that C_n has easy descriptions. ∎

Using the same technique, we find the following coincidences between classes. For sublogarithmic bounds the Kolmogorov complexity has to be taken relative to the output length, as we have defined it, whereas for larger bounds a more standard notion will do.

- P$/O(1)$ is the class of sets having polynomial size circuits in $K[O(1), \text{poly}]$, resp. $KS[O(1), \log]$.
- P$/O(\log^{(m)} n)$ is the class of sets having polynomial size circuits in $K[O(\log^{(m)} n), \text{poly}]$, resp. $KS[O(\log^{(m)} n), \log]$.
- P$/\log$ is the class of sets having polynomial size circuits in $K[\log, \text{poly}]$, resp. $KS[\log, \log]$.
- P$/O(\log n * \log^{(m)} n)$ is the class of sets having polynomial size circuits in $K[O(\log n * \log^{(m)} n), \text{poly}]$, resp. $KS[O(\log n * \log^{(m)} n), \log]$.
- P$/O(\log n * \log\log n)$ is the class of sets having polynomial size circuits in $K[O(\log n * \log\log n), \text{poly}]$, resp. $KS[O(\log n * \log\log n), \log]$, and is also the class of sets having polynomial size circuits with circuit descriptions of size $O(\log n)$.
- P$/O((\log n)^k)$ is the class of sets having polynomial size circuits in $K[O((\log n)^k), \text{poly}]$, resp. $KS[O((\log n)^k), \log]$, and is also the class of sets having polynomial size circuits with circuit descriptions of size $O((\log n)^k / \log\log n)$.
- P/poly is exactly the class of sets having polynomial size circuits.

Proof of theorem 15

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 $P(T)$, with $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$. Define $T' = \{0^k \mid 0^{2^k} \in T\}$. It is easy to see that $A \in P(T')$, querying 0^k instead of 0^{2^k} when required. Observe that the length of the queries is now logarithmic, since $k \in O(\log n)$ whenever $2^k \in n^{O(1)}$. Now essentially repeat the argument: if $A \in P(T')$ with $O(\log n)$ length queries, define $B = \{k \mid 0^k \in T'\}$. Again $A \in P(B)$, and the maximum length of the oracle queries is $\log(c \cdot \log n) = \log\log n + O(1)$.

To prove that iii/ implies iv/, we will resort to the characteristic function of B as an infinite word limiting the sequence of advice words. Let A be a set with $A \in P(B)$ where the size of the queries is bounded as indicated. The number of different queries that can be made is bounded by $2^{\log\log n + O(1)} = O(\log n)$, and moreover these are the first $O(\log n)$ words. So we define the advice w_n as the characteristic sequence of B up to the element in place $c \log n$ for an appropriately selected constant c, and this proves that $A \in$ 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 w is the limit of the sequence of advice words. Let T be the tally set

$$T = \{0^{2^k} \mid \text{the } k^{th} \text{ bit of } w \text{ is } 1\ \}$$

Now $A \in P(T)$ 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. ∎

Proof of theorem 17

We use again the characterization of the class Full-P/log as $\bigcup_T P(T)$ where $T \subseteq \{0^{2^k} \mid k \in \mathbb{N}\}$, which follows from the previous result. Then it is easy to see that i/ implies ii/: to construct the characteristic sequence of T 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 T; these are the only potential nonzeros in χ_T. Thus given as a logarithmically long seed the characteristic function of T relative to $\{0^{2^k} \mid k \in \mathbb{N}\}$, we can easily print out an initial segment of χ_T in time polynomial on the output.

To see that ii/ implies i/, we apply again the doubly exponential skip technique. Observe first that an easier proof seems possible. Consider $A \in P(T)$ where T 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_T 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 $\log(|x|^q) = O(\log |x|)$, which we take as 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 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_T. This one corresponds to

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

so that $m \leq \log\log(n^q) = \log\log n + O(1)$. For length 2^{2^i}, the length of the seed is $c \log 2^{2^i} = c2^i$ for some constant c, and thus as before the total length of the sequence of seeds selected for the advice is $\sum_{i \leq m} c2^i = c2^{m+1} \in O(\log n)$. 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_T up to $|x|^q$ but not much more: there is one 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. ■