

JOSÉ L. BALCÁZAR

JOAQUIM GABARRÓ

**Nonuniform complexity classes specified by
lower and upper bounds**

RAIRO. Informatique théorique et applications, tome 23, n° 2 (1989),
p. 177-194

http://www.numdam.org/item?id=ITA_1989__23_2_177_0

© AFCET, 1989, tous droits réservés.

L'accès aux archives de la revue « RAIRO. Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

NONUNIFORM COMPLEXITY CLASSES SPECIFIED BY LOWER AND UPPER BOUNDS (*)

by José L. BALCÁZAR ⁽¹⁾ and Joaquim GABARRÓ ⁽¹⁾

Communicated by J. DIAZ

Abstract. – We characterize in terms of oracle Turing machines the classes defined by exponential lower bounds on some nonuniform complexity measures. After, we use the same methods to give a new characterization of classes defined by polynomial and polylog upper bounds, obtaining an unified approach to deal with upper and lower bounds. The main measures are the initial index, the context-free cost, and the boolean circuits size. We interpret our results by discussing a trade-off between oracle information and computed information for oracle Turing machines.

Résumé. – Nous caractérisons en termes de machines de Turing avec oracles les classes définies par des bornes inférieures exponentielles pour des mesures de complexité non uniformes. Nous utilisons ensuite les mêmes méthodes pour donner une nouvelle caractérisation des classes définies par des bornes supérieures polynomiales et polylogarithmiques, obtenant ainsi une approche unifiée pour les bornes inférieures et supérieures. Les mesures principales sont l'index initial, le coût grammatical et la taille des circuits booléens. Nous interprétons nos résultats en étudiant, pour les machines de Turing avec oracle, la relation entre l'information due à l'oracle et l'information calculée par la machine.

I. INTRODUCTION

There are two fundamental approaches to complexity theory, known as “uniform” and “nonuniform”. The uniform approach considers each problem as a global object and measures the amount of resources needed to solve it using models of computation usually equivalent to Turing machines. The nonuniform approach considers finite approximations to the problem and measures the complexity of these approximations. This nonuniform approach uses computational models adequate to the finiteness of the approximations. The most widely used model is the boolean circuit model [20]. Many other

(*) Received February 1986, revised August 1987.

⁽¹⁾ Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain.

models of nonuniform complexity measures exist, as the initial index [11] or the context-free cost [6].

Many connections exist between uniform and nonuniform measures [1, 4, 16, 20]. These papers are mainly devoted to the study of classes defined by small (for example polynomial) upper bounds on the uniform resources and/or the nonuniform complexity.

We investigate here the connection between uniform and nonuniform models for problems requiring a high amount of resources. Several interesting results can be shown about high lower bounds on different nonuniform models:

– Consider the initial index [11] (*see* definitions below). To prove exponential lower bounds for problems is not difficult: the two type-parenthesis language has an exponentially growing initial index. The same happens with the set of palindromes over an alphabet of at least two letters. It is also known [22] that almost all languages in Σ^n have an initial index of

$$(1 - \varepsilon) \cdot \frac{\|\Sigma\|^n}{n \cdot \log_2 \|\Sigma\|}.$$

– Consider the context-free cost. For almost all the languages in Σ^n a similar bound as the one above has been established in [12]. Another interesting property regarding context-free cost appears in [10]. It is shown there that the set of words containing an square is not context-free; actually, the proof shows that this set has exponential context-free complexity.

– Consider the boolean circuit complexity [20]. It is well known (*see* [13]) that almost all boolean functions of n variables need

$$(1 - \varepsilon) \cdot \frac{2^n}{n}$$

gates to be synthesized.

This set of facts induces us to consider the classes defined by exponential lower bounds on these nonuniform measures. We give a characterization of these families in terms of oracle Turing machines. As usual (*see* [1]) the oracles are used to “break up” the uniformity. These results are presented in section III; section II is devoted to basic definitions and facts.

In section IV we give new characterizations of the classes defined by upper bounds. These characterizations use methods similar to the methods employed in the previous section. Here we approach the problem from the view point of the “quantity” of oracle as opposed to the approach of the “quality” of

oracle taken in [1]. With these new characterizations, lower and upper bounds adopt a very similar treatment and appear as dual forms of the same phenomenon. We close this paper with a short section of conclusions.

II. PRELIMINARIES

We assume the reader familiar with the basic concepts of Formal Language Theory. Problems are encoded as subsets of words over a fixed finite alphabet Σ consisting of at least two symbols. Hence a *language* is a subset of Σ^* . Computational models are finite automata, pushdown automata, and time or space bounded Turing machines, in their respective deterministic or non-deterministic versions. Relativized computations are performed by oracle Turing machines which query the oracle in one step about words written in a special oracle tape.

On-line machines are also used. In these machines the input tape head is only one-way, and the machine is not allowed to read back its input. For this model we assume that the length of the input is given to the machine in binary on a separate read-only tape. For undefined notions *see* [15].

We study the following nonuniform complexity measures:

– Given a language L , its *initial index* [11] is given by the function $a_L(n)$ whose value is the size (*i. e.*, number of states) of the smallest nondeterministic finite automaton accepting exactly $L \cap \Sigma^n$.

– Given a language L , its *context-free cost* [6] is given by the function $cf_L(n)$ whose value is the size (*i. e.*, number of rules) of the smallest context-free grammar accepting exactly $L \cap \Sigma^n$.

– Given a language L , its *boolean size complexity* or *combinational complexity* [20] is given by the function $c_L(n)$ whose value is the size (*i. e.*, number of gates) of the smallest boolean circuit with fan-in 2 synthesizing the characteristic function of $L \cap \Sigma^n$.

Other similar functions, like branching program size [24], boolean formula size [20], and circuit depth [20], will be introduced in section IV.

III. EXPONENTIAL LOWER BOUNDS

In this section we characterize the classes defined by exponential lower bounds on the nonuniform complexity of the languages. Our notation for

orders of magnitude follows [23]; see also [2] and [17]. It is as follows:

NOTATION: Given a function f , we denote:

(1) $O(f)$ is the set of functions g such that for some $c > 0$ and for all but finitely many n , $g(n) < c \cdot f(n)$.

(2) $o(f)$ is the set of functions such that

$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0.$$

(3) $\Omega_\infty(f)$ is the set of functions g such that for some $c > 0$ and for infinitely many n , $g(n) > c \cdot f(n)$.

(4) $\omega(f)$ is the set of functions g such that

$$\limsup_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \infty.$$

(5) The family of functions with an exponential lower bound (“exponentials” for short) is

$$\Omega_\infty(\text{exp}) = \bigcup_{c > 0} \Omega_\infty(2^c \cdot n).$$

(6) The family of subexponentials is

$$o(\text{exp}) = \bigcap_{c > 0} o(2^c \cdot n).$$

It is not difficult to prove that the following fact holds. Related facts and discussions can be found in [2].

Fact. $o(\text{exp}) = O(2^{o(n)})$

Let m be a nonuniform complexity measure associating to every language L a function m_L , such as the ones defined in section II. The family of sets having *exponential lower bound on m* is:

$$m(\Omega_\infty(\text{exp})) = \{ L \mid m_L(n) \in \Omega_\infty(\text{exp}) \}.$$

We can also define classes by considering subexponential upper bounds.

The family of sets having *subexponential upper bound on m* is:

$$m(o(\text{exp})) = \{ L \mid m_L(n) \in o(\text{exp}) \}.$$

Denoting by $P(\Sigma^*)$ the powerset of Σ^* , the following holds:

Fact 1

- (1) $m(\Omega_\infty(\text{exp})) \cup m(o(\text{exp})) = P(\Sigma^*)$;
- (2) $m(\Omega_\infty(\text{exp})) \cap m(o(\text{exp})) = \emptyset$.

There is no full agreement in the literature about the “correct” definition of space bounded oracle machine; several definitions have been proposed, bounding in different ways the oracle tape. See [19] for a discussion and a smart proposal. We take here the position of expliciting a bound on the oracle tape, which may or may not coincide with the worktape space bound.

Thus, in order to give the characterization in terms of Turing machines, we introduce some simple notation. Consider the oracle Turing machine model; it presents two very different memory devices: work tapes and oracle tape. A bound on the work space available imposes a limitation on the calculation capabilities of the machine, whereas a bound on the oracle tape imposes a bound on the amount of external help obtained by the machine. We denote:

$$\forall(\text{Space}(\Omega_\infty(n)) \vee \text{Oracle}(\Omega_\infty(n)))$$

the class of all sets L such that *for every* machine M and oracle A with $L = L(M, A)$, either the work space or the oracle tape used by M are in $\Omega_\infty(n)$. Dually, we denote

$$\exists(\text{Space}(o(n)) \wedge \text{Oracle}(o(n)))$$

the class of all sets L such that *there is* a machine M and an oracle A with $L = L(M, A)$, such that both the work space and the oracle tape of M are bounded by functions in $o(n)$.

This notation is not standard. A notation like $\text{Space}, \text{Oracle}(f, g)$ for the class that we call $\exists[\text{Space}(f) \wedge \text{Oracle}(g)]$ would match better standard expressions like $\text{Time-Space}(t, s)$. However, it is not easy to design a clear, coherent notation for classes specified by lower bounds, like $\forall[\text{Space}(\Omega_\infty(f)) \vee \text{Oracle}(\Omega_\infty(g))]$. We choose to use dual notations for upper and lower bounds, instead of sticking to a more classical notation for upper bounds and designing a different one for lower bounds.

The following very easy fact will be called “duality principle”. Similar duality principles for other measures will be introduced later on.

Fact 2

Given any set L ,

$L \in \forall (\text{Space}(\Omega_\infty(n)) \vee \text{Oracle}(\Omega_\infty(n)))$ if and only if

$L \notin \exists (\text{Space}(o(n)) \wedge \text{Oracle}(o(n)))$.

Now we prove the relationship between exponential initial index and the oracle Turing machine classes as given in the following theorem.

THEOREM 3: *For initial index and nondeterministic on-line oracle Turing machines:*

(1) $a(\Omega_\infty(\text{exp})) = \forall (\text{Space}(\Omega_\infty(n)) \vee \text{Oracle}(\Omega_\infty(n)))$;

(2) $a(o(\text{exp})) = \exists (\text{Space}(o(n)) \wedge \text{Oracle}(o(n)))$.

Proof: We start by proving (2). Then (1) follows from the duality principle and lemma 1. Let L be a language with subexponential initial index. We construct an oracle D and a machine M with sublinear space bounds, such that $L = L(M, D)$.

Construction of D

For each n , let A_n be a finite automaton defined by

$$A_n = \langle \Sigma, Q_n, \delta_n, 0, F_n \rangle$$

where Q_n can be assumed to be $\{0, 1, 2, \dots\}$, such that $L(A_n) = L \cap \Sigma^n$ and $\|Q_n\| \in O(2^{o(n)})$. As a consequence a sublinear number of bits suffices for encoding each state of Q_n . Define the oracle D by:

$$D = \{n \# x \# i \# j \# u \mid j \in \delta_n(i, x) \text{ and } u = \text{if } j \in F_n \text{ then } 1 \text{ else } 0\}$$

where n , i and j are written in binary. Observe that the words in D have sublinear length, because the code of n is logarithmic and the codes of i and j are sublinear.

Construction of M

Machine M will simulate A_n on inputs of length n . This is done by storing in the work tapes the current state i and the next state j . M obtains j from the oracle by guessing it and querying the oracle about words of the form $n \# x \# i \# j \# u$. Both the work tapes and the oracle tape are bounded by $\max\{\log_2 n, \log_2 \|Q_n\|\} \in o(n)$.

Conversely, let $L = L(M, D)$ where M has k work tapes with a sublinear space bound. Consider the automata A_n having states of the form $\langle q, c_o, c_w \rangle$, where q is a state of M , c_o is a configuration of M 's oracle tape, and

$c_w = \langle c_1 \dots c_k \rangle$ is a configuration of the k work tapes. As M has a sublinear space bound, the number of configurations in each work or oracle tape is in $O(2^{o(n)})$. Hence the number of states of A_n is in $O(2^{o(n)})$. Transitions among the states are defined by mimicking the transitions of M . Observe that the automata read the input correctly since M is on-line. It is immediate that $L(A_n)$ is then $L \cap \Sigma^n$; hence $a_L(n) \in O(2^{o(n)})$. This completes the proof. ■

We consider now the on-line versions of the auxiliary pushdown automata defined in [9]. See also [5] and [8]. The conventions regarding space bounds on the work tapes and/or oracle tape are as before. Note that there is no bound on the size of the pushdown store. Intuitively speaking, the theorem we prove amounts to adding a pushdown to “both sides” of the previous theorem.

The corresponding notation for denoting linear lower bounds for this computation model is as follows:

$$\forall (\text{Space}(\Omega_\infty(n)) \vee \text{Oracle}(\Omega_\infty(n)), \text{Pushdown})$$

denotes the class of all sets L such that for every auxiliary pushdown machine M and oracle A with $L = L(M, A)$, either the work space or the oracle tape used by M are in $\Omega_\infty(n)$. Dually, we denote

$$\exists (\text{Space}(o(n)) \wedge \text{Oracle}(o(n)), \text{Pushdown})$$

the class of all sets L such that there are an auxiliary pushdown machine M and an oracle A with $L = L(M, A)$, such that both the work space and the oracle tape of M are bounded by functions in $o(n)$.

An obvious duality principle, similar to the stated above, holds:

Fact 4

For any set L ,

$L \in \forall (\text{Space}(\Omega_\infty(n)) \vee \text{Oracle}(\Omega_\infty(n)), \text{Pushdown})$ if and only if

$L \notin \exists (\text{Space}(o(n)) \wedge \text{Oracle}(o(n)), \text{Pushdown})$.

Using these classes we obtain a characterization of the languages having exponential context-free cost. We state first some auxiliary definitions and lemmas. From now on, “grammar” is to be understood always as *context-free grammar*.

Given a grammar G , we define its *size* as the number of rules it contains, and denote it by $\|G\|$. For a rule $p = v \rightarrow w$ of G , its *length*, denoted by $|p|$, is defined as $|w|$. The *length of a grammar*, denoted by $|G|$, is the sum of the lengths of all its rules.

Given a *pda* P , we define its *size* $\|P\|$ as the number of states, and its *length* $|P|$ as the sum of the lengths of its transitions, where the length of a transition $(q, u, Z) \vdash (q', Z_1 \dots Z_r)$ is just $|quZq'Z_1 \dots Z_r|$.

The next lemma proves that we can define equivalently context-free cost in terms of lengths of grammars or lengths of *pda*'s.

LEMMA 5: For any L , the following are equivalent:

(1) There exists a succession of grammars G_n with $L(G_n) = L \cap \Sigma^n$ such that $\|G_n\| \in O(2^{o(n)})$.

(2) There exists a succession of grammars G_n with $L(G_n) = L \cap \Sigma^n$ such that $|G_n| \in O(2^{o(n)})$.

(3) There exists a succession of *pda*'s P_n with $L(P_n) = L \cap \Sigma^n$ such that $|P_n| \in O(2^{o(n)})$.

Proof: (1) \Rightarrow (2). Transform each G_n into an equivalent G'_n without rules $v \rightarrow \lambda$. This can be done with small overhead. In the grammar G'_n every rule $v \rightarrow w$ has length $|v| + |w| \leq n + 1$. Hence the length of G'_n is in $n \cdot 2^{o(n)} = 2^{o(n)}$.

(2) \Rightarrow (3). Consider a *pda* having transitions of the form

$$(q, \lambda, v) \vdash (q, w^R)$$

$$(q, x, x) \vdash (q, \lambda)$$

for each rule $v \rightarrow w$ of the grammar and each terminal symbol x . It has the same length as the grammar.

(3) \Rightarrow (1). As every transition in a *pda* allows to reach at most one new state, the number of states is also bounded by $2^{o(n)}$. The classical construction of Ginsburg [15] gives a grammar G_n of size $\|G_n\| \in O(2^{o(n)})$. ■

Observe that the size of the *pda* constructed above from a given grammar is 1. In order to relate the size of the *pda* to the context-free cost of L , we introduce the following construction: given a pushdown automaton for $L \cap \Sigma^n$ it is easy to obtain another pushdown automaton, with stack alphabet $Z_n = \{0, 1\}$, where the pushing rules are of the types

$$(i, u, 0) \rightarrow (j, 00)$$

$$(i, u, 0) \rightarrow (j, 01)$$

$$(i, u, 0) \rightarrow (j, \lambda)$$

$$(i, u, 1) \rightarrow (j, 10)$$

$$(i, u, 1) \rightarrow (j, 11)$$

$$(i, u, 1) \rightarrow (j, \lambda)$$

We call such automata *normalized*. The relationship between context-free cost and the number of states of a normalized automaton is given by:

LEMMA 6: *The following conditions are equivalent:*

- (1) $cf_L(n) \in o(\exp)$.
- (2) *There exists a succession of normalized pda P_n with sets of states Q_n such that P_n accepts $L \cap \Sigma^n$, and $\|P_n\| \in o(\exp)$.*

Note that, by duality, the sets L with $cf_L(n) \in \Omega_\infty(\exp)$ can be characterized analogously in terms of the number of states of every succession of normalized pda recognizing $L \cap \Sigma^n$.

Now we can state our announced characterization:

THEOREM 7: *For context-free cost and nondeterministic on-line auxiliary pushdown machines:*

- (1) $cf(\Omega_\infty(\exp)) = \forall(\text{Space}(\Omega_\infty(n)) \vee \text{Oracle}(\Omega_\infty(n)), \text{Pushdown})$.
- (2) $cf(o(\exp)) = \exists(\text{Space}(o(n)) \wedge \text{Oracle}(o(n)), \text{Pushdown})$.

Proof: (1) will follow from (2) by duality. We must prove (2). Let L be in $cf(o(\exp))$. There exists a succession of normalized automata P_n with sets of states of cardinality $\|Q_n\| \in O(2^{o(n)})$. Define the oracle D by:

$$D = \{n \# i \# x \# Z_1 \# j \# Z_2 \# Z_3 \mid (i, x, Z_1) \vdash (j, Z_2, Z_3) \text{ is a rule of } P_n\}.$$

Then i and j are always of sublinear length. The simulation by an auxiliary pushdown machine follows the same ideas of theorem 3; the pushdown of P_n is simulated by the auxiliary pushdown.

Reciprocally, assume that $L = L(M, D)$ where M is a sublinear space bounded auxiliary pushdown machine. Without loss of generality, we can assume that M uses its pushdown in the same way as a normalized pda. Consider as states of the pda tuples describing configurations of the work and oracle tapes of M . As the length of each tape is $o(n)$, the number of states is $O(2^{o(n)})$. The transitions of the pda simulate the computation of M over inputs of length n . ■

Our last result of this section relates exponential boolean circuit complexity with a class defined in terms of time and oracle space bounds of Turing machines. Define the following notation.

$$\forall(\text{Time}(\Omega_\infty(\exp)) \vee \text{Oracle}(\Omega_\infty(n)))$$

denoting the class of all sets L such that for every oracle Turing machine M and oracle A with $L = L(M, A)$, either the running time is in $\Omega_\infty(\exp)$ or the

amount of oracle tape used by M is in $\Omega_\infty(n)$. Dually, we denote by

$$\exists(\text{Time}(o(\text{exp})) \wedge \text{Oracle}(o(n)))$$

the class of all sets L such that there are an oracle Turing machine M and an oracle A with $L=L(M, A)$, such that the running time of M is bounded by a function in $o(\text{exp})$ and the oracle tape of M is bounded by a function in $o(n)$. Again we can state a “duality principle”:

Fact 8

For any set L ,

$L \in \forall(\text{Time}(\Omega_\infty(\text{exp})) \vee \text{Oracle}(\Omega_\infty(n)))$ if and only if

$L \notin \exists(\text{Time}(o(\text{exp})) \wedge \text{Oracle}(o(n)))$.

Using these classes we prove the following:

THEOREM 9: *For circuit complexity and deterministic Turing machines:*

(1) $c(\Omega_\infty(\text{exp})) = \forall(\text{Time}(\Omega_\infty(\text{exp})) \vee \text{Oracle}(\Omega_\infty(n)))$.

(2) $c(o(\text{exp})) = \exists(\text{Time}(o(\text{exp})) \wedge \text{Oracle}(o(n)))$.

Proof: Again we only prove part (2); (1) will follow by duality. Let L be a language with subexponential cost. We construct a machine M and an oracle D such that M accepts L in subexponential time using sublinear oracle tape.

Construction of D

Consider the circuit corresponding to inputs of size n . The gates can be numbered so that $f(n)$ bits suffice to write down each number, with $f(n) \in o(n)$. The oracle contains information about the value $f(n)$, and also information about the connections of the circuit. The value of $f(n)$ is needed by the deterministic machine in order to know when to stop querying the oracle for new gates.

Define:

$$D = \{ n \# 1^{f(n)} \} \cup \{ n \# \# i \# j \# k \# \theta \mid \text{gate } i \text{ of the circuit}$$

for length n is the operation θ over gates $j, k \}$.

Observe that the words in D needed to find the circuit for inputs of length n are of length $o(n)$, because i, j and k can be written down with a sublinear quantity of hits.

Construction of M

The machine M works in two different phases. First, it constructs on a worktape (which we call "circuit tape") the circuit corresponding to the length n of the input, with the help of the oracle D , after it evaluates the circuit over the input. As the number of gates is subexponential, the whole circuit can be constructed and evaluated by levels within subexponential time. We omit the evaluation procedure, which is well-known.

Let us present the construction of the circuit. First, the machine queries the oracle successively about the words of the form $n\#1^t$ for $t=1, 2, \dots$, storing the current value of t in a separate tape. A positive answer means that the tape contains the length of the maximum number of a gate of the circuit $f(n)$. Then the machine queries D about all the words of the form $n\#i\#j\#k\#\theta$ for i, j and k having length less than or equal to $f(n)$, and each allowed binary boolean operation θ . Each time a positive answer is received, it means that one more gate is known and stored in the circuit tape. As $f(n)$ is in $o(n)$, the time employed in the construction is $O(2^{o(n)})$. This machine accepts L in subexponential time with oracle D .

Conversely, let $L=L(M, D)$ where M works in subexponential time and uses oracle space $f(n)$, where f is a sublinear function. Let c_0 be the characteristic function of D up to size $f(n)$. It is possible to construct a circuit of subexponential size that synthesizes c_0 , by comparing the input with each of the $O(2^{o(n)})$ words in D with length up to $f(n)$.

Now a circuit can be built for the machine M assuming no oracle queries are made, as in [18]. The size of the circuit is the square of the running time of M : this is

$$(2^{o(n)})^2 = 2^{o(n)+o(n)} = 2^{o(n)}$$

and hence is also subexponential.

Finally, combine both circuits by plugging a copy of the circuit for the oracle into each step of the machine's computation, so that if on some input a query is made at this step of the computation, it can be correctly answered. The full circuit is again subexponential, and it is able to decide L for inputs of length n . Therefore the boolean cost of L is subexponential, as was to be shown. ■

IV. UPPER BOUNDS

Given a nonuniform measure we can define families of languages by imposing upper bounds on the measure. In this section three classes of functions are considered as upper bounds: logarithmic functions, polylog [i. e. $(\log_2 n)^k$] functions, and polynomials. For any nonuniform measure m , the family of languages defined by polynomial upper bounds is:

$$m(O(\text{poly})) = \{ L \mid \exists k \in \mathbb{N} \text{ with } m_L(n) = O(n^k) \}.$$

When m is the initial index, the context-free cost, or the boolean size complexity, the families defined in this way were characterized in [1] (the last case is due to A. Meyer). These characterizations used the notion of “sparse oracles”, with *no bound set* over the length of the oracle tape. Define the class SP of all the so-called *sparse* oracles, as the class formed by all the oracle sets consisting of a polynomially growing number of words. The characterizations presented there were the following:

THEOREM 10[1]: (1) *Consider nondeterministic log-space on-line oracle Turing machines; denote $\text{NLOG}_{\text{on}}(S)$ the corresponding uniform class for oracle S . Then*

$$a(O(\text{poly})) = \bigcup_{S \in \text{SP}} \text{NLOG}_{\text{on}}(S).$$

(2) *Denote $\text{ANLOG}_{\text{on}}(S)$ the analogous class defined by on-line log-space auxiliary pushdown machines. Then*

$$cf(O(\text{poly})) = \bigcup_{S \in \text{SP}} \text{ANLOG}_{\text{on}}(S).$$

(3) *Let $P(S)$ denote the standard polynomial time class relativized to S . Then*

$$c(O(\text{poly})) = \bigcup_{S \in \text{SP}} P(S).$$

Several corollaries were drawn regarding both uniform and nonuniform complexity classes, and similar characterizations were shown for other nonuniform measures.

We present here another set of characterizations of classes defined by polynomial upper bounds, based on the space available for oracle queries, rather than on the structure of the oracle set. We characterize also the dual family

$$m(\omega(\text{poly})) = \{ L \mid m_L(n) \in \omega(\text{poly}) \}$$

for which a duality principle with $m(O(\text{poly}))$ can be stated.

THEOREM 11: *For on-line nondeterministic oracle Turing machines.*

$$(1) a(O(\text{poly})) = \exists (\text{Space } (O(\log n)) \wedge \text{Oracle } (O(\log n))).$$

$$(2) a(\omega(\text{poly})) = \forall (\text{Space } (\omega(\log n)) \vee \text{Oracle } (\omega(\log n))).$$

Moreover, for on-line nondeterministic auxiliary pushdown machines.

$$(3) cf(O(\text{poly})) = \exists (\text{Space } (O(\log n)) \wedge \text{Oracle } (O(\log n)), \text{Pushdown}).$$

$$(4) cf(\omega(\text{poly})) = \forall (\text{Space } (\omega(\log n)) \vee \text{Oracle } (\omega(\log n)), \text{Pushdown}).$$

Finally, for deterministic oracle Turing machines.

$$(5) c(O(\text{poly})) = \exists (\text{Time } (O(\text{poly})) \wedge \text{Oracle } (O(\log n))).$$

$$(6) c(\omega(\text{poly})) = \forall (\text{Time } (\omega(\text{poly})) \vee \text{Oracle } (\omega(\log n))).$$

where the classes that appear in right hand sides are defined in the same manner as before.

The even-numbered statements follow from the odd-numbered statements by duality principles. Proofs of the odd-numbered statements can be constructed by rephrasing the proofs of theorems 3, 7 and 9, adjusting the bounds to polynomials and checking that the necessary conditions of closure of the class of bounding functions hold for polynomials.

Observe that (5) [and similarly (1) and (3)] can be interpreted in the following manner: a set L has polynomial size boolean circuits if and only if there is an (*arbitrary*) oracle set A such that L is in $P(A)$, and the polynomial time machine witnessing this fact uses only a part of the oracle tape which is bounded by a logarithm. About the family $a(\omega(\text{poly}))$, it is possible to define quite natural sets L (see [7]) such that $a_L(n) \in \theta(n^{c \cdot \log n})$, and hence $L \in a(\omega(\text{poly}))$.

In [1] only polynomials were used as upper bounds for defining nonuniform classes. For this bound, the exact relationship between boolean formulae (or, equivalently, circuits of fan-out 1) and branching programs is open, although it is generally conjectured that they are not equivalent. A branching program is a directed, acyclic graph with an initial and some final nodes. Nodes are labeled by integers. Each internal node has exactly two outgoing edges labeled 0 and 1. Its computation starts at the initial node, and at node with label n it follows the path labeled with the value of the n -th bit of the input. See [3] and [24], for example, for material related to branching programs. However, the following can be proven:

THEOREM 12: *A set L has polynomial size branching programs iff there is an (*arbitrary*) oracle set A such that L is accepted by a deterministic off-line*

oracle Turing machine, using oracle A , with logarithmic bounds both on the worktape space and on the oracle tape.

Proof: Let L have polynomial size branching programs. Construct an oracle having words of the form $n\#i\#j\#u\#k$, meaning that node with code i in branching program accepting $L \cap \Sigma^n$ is labeled by j , and the outcoming edge from i labeled by u (which is 0 or 1) goes to node k . Then a logspace machine can simulate it by keeping the current node i and the position of the input head in the work tapes cycling over all the words of the form $n\#i\#j\#0\#k$ and $n\#i\#j\#1\#k$ until the oracle indicates the correct one, bringing the input head to position j on the input tape, and deciding the correct path. If no such word is in the oracle then a final node has been reached. It is easy to see that the required space is logarithmic.

Conversely, let M be a machine as specified, and assume without loss of generality that M always halts. Each configuration of the machine, including the oracle tape, is considered to be a node of the branching program. Its label is the position of the input head in the configuration, and the edges labeled 0 and 1 are defined by the transitions of the machine and the answers of the oracle. The size of such branching program is polynomially bounded. ■

Let us turn for a moment to the depth of the circuits for a set. Polynomial bounds no longer have a meaning, because every boolean function can be synthesized with circuits of linear depth [20]. In the following we consider polylog functions and we use them to bound the depth of the circuits. If only depth is bounded and no bound is set over the size, then there is no difference between unbounded fan-out and fan-out 1, because fan-out may be trivially reduced to L “unfolding” the circuit, with no increment of the depth. Denote as d_L the nonuniform measure which gives the minimum depth needed to synthesize the characteristic function of $L \cap [0, 1]^n$.

We shall use the following well-known results about depth and size of formulae: first, $d_L(n) \in \theta(\log c_L^1(n))$ (Spira’s theorem; see [20]), where c_L^1 denotes the boolean complexity with gates of fan-out 1, or equivalently the formula size complexity. Second, if L is in $DSPACE(S)$ then $d_L(n) \in O(S^2(n))$ (see [4]).

Recall the standard notation of [16] for nonuniform complexity classes: a set A is in the class C/F if and only if there is a set B in C and a function h from $\{0\}^* \rightarrow \Sigma^*$ whose length is bounded by a function of F , such that

$$x \in A \text{ if and only if } \langle x, h(0^{|\cdot|}) \rangle \in B.$$

A very interesting situation arises when considering polylog functions and n^{polylog} functions as upper bounds. We summarize our results in the following theorem.

THEOREM 13: *The following seven classes are the same:*

- (1) $d(O(\text{polylog}))$.
- (2) *The class of sets having boolean formulae of n^{polylog} size.*
- (3) $\exists (\text{Space}(O(\text{polylog})) \wedge (\text{Oracle}(O(\text{polylog}))))$, *measured by off-line deterministic Turing machines.*
- (4) *The class*

$$\bigcup_{S \in \text{SP}} \text{SPACE}(\text{polylog}, S)$$

measured by off-line deterministic Turing machines that use polylog work space. No upper bound is set on the length of the oracle tape.

- (5) $\text{SPACE}(\text{polylog})/n^{\text{polylog}}$, *measured by off-line deterministic Turing machines.*
- (6) *The class of sets having initial index, measured by two-way finite automata, bounded above by a n^{polylog} function.*
- (7) *The class of sets having branching programs of n^{polylog} size.*

Proof: (1) \Leftrightarrow (2). Immediate from Spira's theorem.

(1) \Leftrightarrow (3). Spira's theorem is not required here. Let L have formulae of polylog depth. The size of these formulae cannot be more than $2^{\text{polylog}} = n^{\text{polylog}}$. Assigning a number to each node in the formula for length n , we can encode the formulae into an oracle D by words of the form $n \# i \# j \# k \# \theta$ exactly like in theorem 9. We evaluate the formula using top-down computation. A stack of polylog size suffices, because both the depth of the formula and the size of the index of each node are bounded by a polylog function.

Conversely, let $L = L(M, D)$ with polylog space. The oracle can be encoded into a polylog depth formula as in theorem 9. The simulation of the polylog-space machine M , without queries, can be done in polylog depth by the result of Borodin [4] cited above.

(1) \Leftrightarrow (4). The proof is very similar to (1) \Leftrightarrow (3). The only change is that the oracle is formed by words of the form $0^n \# i \# j \# k \# \theta$. It is easy to show that such oracle must be sparse. Of course the bound on the length of the oracle tape no longer holds; this is the only change on the accepting machine. For the converse, it is very easy to see that sparse oracles can be encoded in polynomial depth formulae. The results in [1] follow the same technique.

(2) \Rightarrow (5). Consider an advice function such that for each n it gives the formula corresponding to length n . It is bounded in length by a 2^{polylog} function. It can be evaluated within polylog space as in the case (1) \Leftrightarrow (3).

(5) \Rightarrow (3). Given the advice function h of 2^{polylog} length for L , we construct an oracle D with words of the form $n\#i\#u$ where u is the i -th bit of $h(n)$. A polylog space machine M can use it to simulate the advice by keeping in binary (in polylog space) the position i of the head on the advice, and querying D about $n\#i\#0$ and $n\#i\#1$ in order to decide each move. Both the work space and the oracle tape are polylog.

(3) \Leftrightarrow (6). It is exactly like (1) in theorem 11, with the only change that both the automata and the polylog space machines have two-way input heads.

(3) \Leftrightarrow (7). Simulations between polylog space, polylog oracle machines and n^{polylog} size branching programs can be performed exactly as in theorem 12. ■

V. CONCLUSIONS

Our main results are characterizations of classes defined by upper and lower bounds on some nonuniform complexity measure. We characterize these classes by considering standard computational devices (oracle Turing machines) and bounding simultaneously some resources like the work space or the running time, and the length of the queries. The first bound is a bound on the computational capabilities of the machine, whereas the second bound gives an estimate of the amount of external information given to the machine. The techniques used for the lower bound case and for the upper bound case are very similar, and in some sense they could be unified under appropriate hypothesis of the families of functions used as bounds. The uniform model, the Turing machine, is made nonuniform by using the information from the oracle. Similar results might be possible by forcing instead the nonuniform measures to be uniform by requiring that the automata (or whatever) be easily computable.

Previous results [1] gave other characterizations depending on the structure of the oracle set. These results could suggest a trade-off between the amount of work space or time and the amount of information coded by the oracle, similar to the trade-offs known for other computational resources (like time for space in [14]). However, our characterizations (for arbitrary oracles) in terms of the way the machines query the oracle show that this trade-off only holds in a very trivial way, where the two extreme situations are “almost

everything is given by the oracle” and “almost everything is computed”, and NO intermediate nontrivial situation holds. Let us present a simple example of this fact. Consider the language $C = \{ w \# w \mid w \in \{0,1\}^* \}$ of the squares with a central separator. It is not difficult to show [11] that its initial index is exponential. Our first theorem shows that in order to recognize this set we need a linear amount of space. On the one side, a machine may use no oracle space, copy w in a separate tape until the $\#$ is reached and then check that the second half matches w ; we need linear work space. On the other side, a machine can use no work space, just copy the input in the oracle tape, and then query an oracle for C ; we need linear oracle space. By our first theorem, we know that we can do no better: any machine for C must use either linear work space or linear oracle space, and hence the two trivial machines just presented are in some sense optimal. A similar argument for auxiliary push-down machines follows from our theorems and the fact that C has exponential context-free cost [12], and indicates that the trivial machines outlined above are optimal even if they are furnished with an unbounded pushdown store.

Furthermore, we have shown that this technique works for already well-studied classes as the ones defined by polynomial upper bounds. A duality principle can be always stated relating an upper bound with its corresponding lower bound, and this duality carries over to the uniform characterizations. This phenomenon provides new characterizations of classes defined by polynomial and n^{polylog} upper bounds. Finally, we have shown that measures that we expected to differ for polynomial upper bounds, like formula size and branching program size, do not differ for these n^{polylog} upper bounds. For this family we have obtained several different characterizations. We consider that this collection of results presents an interesting new view of the landscape of nonuniform complexity classes.

ACKNOWLEDGEMENT

The authors wish to thank an anonymous referee for pointing out some mistakes and suggesting several improvements of the presentation.

REFERENCES

1. J. L. BALCÁZAR, J. DÍAZ and J. GABARRÓ, *Uniform Characterizations of Nonuniform Complexity Measures*, Information and Control, Vol. 67, Nos. 1-3, 1985, pp. 53-69.
2. J. L. BALCÁZAR and J. GABARRÓ, *Some Comments About Notations of Orders of Magnitude*, Bull. EATCS, Vol. 30, 1986, pp. 34-42.

3. D. BARRINGTON, *Bounded-width Polynomial-size Branching Programs Recognize Exactly Those Languages in NC^1* , In: *18th ACM Symp. Th. of Comp.*, 1986, pp. 1-5.
4. A. BORODIN, *On Relating Time and Space to Size and Depth*, *SIAM J. Comp.*, Vol. 6. No. 4, 1977, pp. 733-744.
5. F. BRANDENBURG, *On One-way Auxiliary Pushdown Automata*, In: *3rd GI Conf. on Theor. Comp. Sci.*, 1977, Springer Verlag, Lect. Notes in Comp. Sci., Vol. 48, pp. 132-144.
6. W. BUCHER, K. CULIK, H. MAURER and D. WOTSCHKE, *Concise Description of Finite Languages*, *Theor. Comp. Sci.*, Vol. 14, No. 3, 1981, pp. 227-246.
7. R. CASAS and J. GABARRÓ, *About LOG-ON languages*, Internal report RR 85/02, Facultat d'Informàtica de Barcelona.
8. M. CHYTIL, *Almost context-free languages*, Manuscript, 1984.
9. S. COOK, *Characterizations of Pushdown Machines in Terms of Time-Bounded Computers*, *Journal ACM*, Vol. 18, No. 1, 1971, pp. 4-18.
10. A. EHRENFUECH and G. ROZENBERG, *On the Separating Power of EOL Systems*, *RAIRO Inf. Theor.*, Vol. 17, No. 1, 1983, pp. 13-22.
11. J. GABARRÓ, *Funciones de complejidad y su relación con las familias abstractas de lenguajes*. Ph. D. dissertation, 1983. See also: *Initial Index: a new Complexity Function for Languages*, In: *10th Int. Coll. on Aut. Lang. and Prog.*, 1983, Springer Verlag, Lect. Notes in Comp. Sci., Vol. 154, pp. 226-236.
12. G. GOODRICH, R. LADNER and M. FISCHER, *Straight-line Programs to Compute Finite Languages*, *Conf. Theor. Comp. Sci.*, Waterloo, 1977.
13. M. HARRISON, *Introduction to Switching and Automata Theory*, McGraw Hill, New York, 1965.
14. J. HOPCROFT, W. PAUL and L. VALIANT, *On Time Versus Space and Related Problems*, *Journal ACM*, Vol. 2, 1977, pp. 332-337.
15. J. HOPCROFT and J. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading (Mass.), 1979.
16. R. KARP and R. LIPTON, *Some Connections between Nonuniform and Uniform Complexity Classes*. In: *12th ACM Symp. Th. of Comp.*, 1980, pp. 302-309.
17. D. KNUTH, *Big Omicron and Big Omega and Big Theta*, *SIGACT News*, Apr.-June 1976, pp. 18-24.
18. R. LADNER, *The Circuit Value Problem is Log Space Complete for P*, *SIGACT News*, January 1975, pp. 18-20.
19. W. RUZZO, J. SIMON and M. TOMPA, *Space-bounded Hierarchies and Probabilistic Computations*, *J. Comp. Syst. Sci.*, Vol. 28, 1984, pp. 216-230.
20. J. SAVAGE, *The Complexity of Computing*, Wiley Interscience, 1976.
21. C. SCHNORR, *The Network Complexity and the Turing Machine Complexity of Finite Functions*, *Acta Informàtica*, Vol. 7, 1976, pp. 95-107.
22. M. J. SERNA, *Asymptotical Behaviour of Some Non-Uniform Measures*, *Inf. Théor. et Appl.*, (to appear).
23. P. VITANYI and L. MEERTENS, *Big Omega Versus the Wild Functions*, *Bull. EATCS*, 22 Feb. 1984, pp. 14-19.
24. I. WEGENER, *On the Complexity of Branching Programs and Decision Trees for Clique Functions*, *Journal ACM*, Vol. 35, 1988, pp. 461-471.