

# Complexity Classes between $\Theta_k^P$ and $\Delta_k^P$

Jorge Castro

Dept. Llenguatges i Sistemes Informàtics

Universitat Politècnica de Catalunya

Pau Gargallo 5

08028 Barcelona, Spain

E-mail: castro@lsi.upc.es

Carlos Seara

Dept. Matemàtica Aplicada II

Universitat Politècnica de Catalunya

Pau Gargallo 5

08028 Barcelona, Spain

E-mail: seara@ma2.upc.es

## Abstract

We give different characterizations of the classes  $P_{\log^i}(\text{NP})$ , which are located between  $\Theta_2^P$  and  $\Delta_2^P$ . First we show that these classes are equal to the classes  $\text{AC}^{i-1}(\text{NP})$ . Second we prove that they are also equivalent to certain binary search classes over NP. Third we show that they can be characterized as the classes defined by circuits of size  $O(\log^i n)$  with NP oracle gates. All the proofs given for these relationships remain valid under relativizations, so taking  $\Sigma_k^P$  instead of NP we can obtain similar characterizations for the classes  $P_{\log^i}(\Sigma_k^P)$ , which are located between  $\Theta_{k+1}^P$  and  $\Delta_{k+1}^P$ . These relationships can be used to prove  $\Theta$ -lowness properties for some classes. In particular, we clarify the situation of the classes in  $L_2^{P,\Delta}$  whose membership to  $L_2^{P,\Theta}$  was not clear. With these results we solve open questions that arose in [Wa-90], [AW-90] and [LS-91]. Finally, we give an oracle relative to which classes  $P_{\log^i}(\text{NP})$  and  $P_{\log^{i+1}}(\text{NP})$  are different.

## 1. Introduction

The works of Buss and Hay [BH-91] and Wagner [Wa-90] showed several characterizations for the class of languages logspace Turing reducible to an NP set. Among other characterizations, they proved independently that this class coincides with the following ones:

- $P_{\log}(\text{NP})$ , the class of languages recognized by deterministic polynomial time oracle machines making only  $O(\log n)$  queries to an NP set.
- $P_{\parallel}(\text{NP})$ , the class of languages accepted by deterministic polynomial time oracle machines doing one round of parallel queries to an NP set.
- $\text{NP}(n^{O(1)})$ , the class of problems that can be solved doing a binary search over witnesses for strings in sets in NP. The bound  $n^{O(1)}$  is the range over which the search is allowed to take place.

All these equivalences remain valid replacing NP with any level  $\Sigma_k^P$  of the polynomial time hierarchy ( $k \geq 2$ ). This multiplicity of equivalent definitions indicates that classes  $P_{\log}(\Sigma_k^P)$  are natural ones, and it suggests to consider the classes  $\Theta_k^P$ , defined by  $\Theta_0^P = P$  and  $\Theta_{k+1}^P = P_{\log}(\Sigma_k^P)$ , as constitutional parts of the polynomial time hierarchy [Wa-90].

Wilson [Wi-87][Wi-90] introduced the notion of relativized circuits by allowing oracle gates in the circuits, and he studied properties of relativized versions of AC and NC. He defined an oracle gate as a  $k$ -input, one-output gate that, on input  $x$  of length  $k$ , will produce the value 1 on its output edge if and only if  $x$  is in the specified oracle set. A straightforward simulation shows that the classes of languages accepted by  $\Sigma_k^P$ -relativized AC and NC circuits are included in  $\Delta_{k+1}^P$ . The first motivation of our work was to find relationships between these classes and those mentioned in the first paragraph.

Below we show connections between classes of languages recognized by polynomial time oracle machines which make polylog queries to an NP set and other classes defined on different ways. More exactly, the class  $P_{\log^i}(\text{NP})$  of problems which can be solved by deterministic polynomial time oracle machines making  $O(\log^i n)$  queries to NP is shown to be equivalent to the following ones:

- $\text{AC}^{i-1}(\text{NP})$ , the class of sets accepted by NP-relativized AC circuits of depth  $O(\log^{i-1} n)$ .
- $\text{NP}(2^{O(\log^i n)})$ , the class of problems that can be solved doing a binary search over witnesses for strings in sets in NP. The bound  $2^{O(\log^i n)}$  is the range over which the search is allowed to take place.
- The class of languages accepted by NP-relativized circuits of size  $O(\log^i n)$ .

These three equivalences are proved, respectively, in sections 3, 4 and 5 below, and they solve questions asked by Wagner in [Wa-90] and by Allender and Wilson in [AW-90]. All the proofs given for these results are valid under relativizations. In particular, they remain valid replacing NP with any level  $\Sigma_k^P$  of the polynomial time hierarchy ( $k \geq 2$ ); so we can obtain similar equivalences for the classes  $P_{\log^i}(\Sigma_k^P)$ , which are located between  $\Theta_{k+1}^P$  and  $\Delta_{k+1}^P$ . Recently Ogiwara in [Og-94] completed these results proving that the class  $\text{AC}^{i-1}(\text{NP})$  coincides with  $\text{NC}^i(\text{NP})$ . Therefore, he showed a new characterization of  $P_{\log^i}(\text{NP})$  in terms of NP-relativized NC circuits.

In section 6 we apply to the low hierarchy some results from section 3. The low and high hierarchies in NP were defined by Schöning [Sc-83], to provide a formal framework for analyzing the internal structure of NP. Later he also introduced a refinement of the low and high hierarchies based on the  $\Delta$ -levels of the polynomial time hierarchy [Sc-86b]. In [Sc-86b], [KS-85], [BB-86], [BS-92] and [AH-92] several classes of sets were located in the low hierarchy. Long and Sheu [LS-91] introduced a new refinement of the low and high hierarchies based on the  $\Theta$ -levels of the polynomial time hierarchy, and they sharpened most of the known lowness results. However they could not locate in the new refinement the NP sets that are Turing equivalent to some tally set or NP sets that are either standard or general left cuts for some real number  $x$ . We show that results of section 3 can be used to prove  $\Theta$ -lowness properties for some complexity classes. In fact, applying those results, we clarify the situation of the aforementioned classes.

In the last section we prove that there is an oracle relative to which classes  $P_{\log^i}(\text{NP})$  and  $P_{\log^{i+1}}(\text{NP})$  are different for any integer  $i \geq 1$ . With this result we improve the relativized separations for  $\Theta_2^P$  and  $\Delta_2^P$  given by Buss and Hay in [BH-91] and by Lozano and Torán in [LT-91].

## 2. Definitions and notation

$\Sigma$  denotes an arbitrary alphabet of size at least two. For  $x \in \Sigma^*$ ,  $|x|$  denotes the length of  $x$ . A tally set is any set over  $\{1\}^*$ .  $L(M)$  denotes the set accepted by Turing machine  $M$ , and  $L(M, A)$  (or alternatively,  $L(M^A)$ ) denotes the set accepted by an oracle machine  $M$  using the oracle set  $A$ . The classes P and NP have their standard definitions, and  $P(A)$  and  $\text{NP}(A)$  are, respectively, their  $A$ -relativized counterparts.

Polynomial time many-one reducibility (denoted by  $\leq_m^P$ ) and polynomial time Turing reducibility (denoted by  $\leq_T^P$ ) have their usual definitions. We say that a class C is closed under  $\leq_m^P$ -reducibility (respectively  $\leq_T^P$ -reducibility) if it verifies:

$$B \in C \text{ and } A \leq_m^P B \text{ (resp. } A \leq_T^P B) \implies A \in C.$$

**Definition 2.1** For any class  $C$  and for any bounding function  $f$ , we denote by  $P_f(C)$  the class of languages accepted by deterministic polynomial time oracle machines that on inputs of length  $n$  make at most  $O(f(n))$  queries to an oracle set in  $C$ .

**Definition 2.2** [BH-91]  $P_{\parallel}(C)$  is the class of languages accepted by deterministic polynomial time oracle machines which make one round of parallel queries to a set  $B \in C$ . By a round of parallel queries to  $B$  we mean that the oracle machine writes a set of strings separated by delimiters on a query tape and then invokes an oracle for  $B$ ; the oracle returns a string of YES/NO answers on an answer tape that specify membership of each query string to  $B$ . Note that within one round of parallel queries, all of them must be formulated before any answer is known. In general, we denote by  $P_{\parallel f}(C)$  the class of sets recognized by deterministic polynomial time oracle machines that on inputs of length  $n$  make  $O(f(n))$  adaptive rounds of parallel queries to a set  $B \in C$ .

We say that the sets  $A$  and  $B$  are Turing equivalent if  $A \leq_T^P B$  and  $B \leq_T^P A$ . We use the symbol  $\equiv_T^P$  to denote Turing equivalent sets.

**Definition 2.3** [BT-89]  $E_T(\text{TALLY})$  is the class of sets Turing equivalent to a tally set,

$$E_T(\text{TALLY}) = \{A \mid \text{there exists a tally set } T \text{ such that } A \equiv_T^P T\}.$$

The classes of the polynomial time hierarchy, including  $\Theta$ -classes, are  $\{\Sigma_k^P, \Pi_k^P, \Delta_k^P, \Theta_k^P \mid k \geq 0\}$ , where:

$$\begin{aligned} \Sigma_0^P &= \Pi_0^P = \Delta_0^P = \Theta_0^P = P, \text{ and for } k \geq 0, \\ \Sigma_{k+1}^P &= \text{NP}(\Sigma_k^P), \\ \Pi_{k+1}^P &= \text{co-}\Sigma_{k+1}^P, \\ \Delta_{k+1}^P &= P(\Sigma_k^P), \\ \Theta_{k+1}^P &= P_{\log}(\Sigma_k^P). \end{aligned}$$

For any set  $A$ ,  $\{\Sigma_k^P(A), \Pi_k^P(A), \Delta_k^P(A), \Theta_k^P(A) \mid k \geq 0\}$  are the classes of the polynomial time hierarchy relative to  $A$ .

In section 6 of [Wa-90] the Extended Boolean hierarchy is introduced as the hierarchy of classes  $\text{NP}(r(n))$  defined as follows:

$$A \in \text{NP}(r(n)) \iff \exists B \in \text{NP} \text{ such that } c_B(x, l+1) \leq c_B(x, l) \text{ for all } l, 1 \leq l < r(|x|) \text{ and } c_A(x) \equiv \sum_{l=1}^{r(|x|)} c_B(x, l) \pmod{2};$$

where  $c_A$  denotes the characteristic function of the set  $A$ . It was known that  $\text{NP}(n^{O(1)}) = \Theta_2^P$  (see [BH-88] and [Wa-90]), and  $\text{NP}(2^{n^{O(1)}}) = \Delta_2^P$  (see [Wa-90]), but similar results for other superpolynomial bounding functions were not known. We shall study this problem in section 4. In a similar way the hierarchy of classes  $C(r(n))$  can be defined for any class  $C$ : just taking  $C$  instead of  $\text{NP}$  in the definition of  $\text{NP}(r(n))$  (see [AW-90]).

For definitions and notation about circuits we follow Wilson [Wi-87] [Wi-90]. A Boolean circuit with bounded fan-in is an acyclic directed graph whose nodes are labelled with an operator. Nodes of indegree zero are the input and constant ones, nodes of indegree one are labelled by negations and nodes of indegree two are labelled by AND and OR operators. Circuits with unbounded fan-in have no restriction on the indegree of nodes labelled AND or OR. Since we are interested in deciding set membership, the circuits have only a single output gate; the circuit

accepts an input string if the length of the string in binary is the same as the number of input gates of the circuit and it outputs 1 when the string is given on its input gates.

The *size* of the circuit is the number of nodes of the circuit and its *depth* is the length of the longest directed path from the input to the output in the graph. The size represents a measure of the hardware resources and the depth is a measure of the parallel time. A circuit family  $\{C_n\}_{n \geq 1}$  accepts a set  $L$  if, for all  $n$ ,  $C_n$  has  $n$  input nodes and accepts only those strings in  $L$  of length  $n$ . Note that, if the  $n^{\text{th}}$  circuit could be independent of the  $(n-1)^{\text{th}}$  circuit then there would exist a family of circuits which would accept a nonrecursive set. Therefore, it is necessary to require some kind of uniformity in the description of the circuits  $\{C_n\}_{n \geq 1}$ . We shall use in this paper one of the most frequently used concepts of uniformity:

**Definition 2.4** A circuit family  $\{C_n\}_{n \geq 1}$  is  $O(\log n)$ -uniform if there is a deterministic logspace Turing machine that on any input of length  $n$  outputs an encoding of  $C_n$ .

Classes of languages  $\text{NC} = \bigcup_{i \geq 0} \text{NC}^i$  and  $\text{AC} = \bigcup_{i \geq 0} \text{AC}^i$  are defined as follows:

$$L \in \text{NC}^i (\text{AC}^i) \iff \exists \{C_n\}_{n \geq 1}, \text{ bounded fan-in (unbounded fan-in) circuits } O(\log n)\text{-uniform with size } O(n^{O(1)}) \text{ and depth } O(\log^i n) \text{ which recognizes } L.$$

In this paper we work with relativized circuits, that is, oracle gates are allowed. These gates can determine the membership of a string to an oracle set. In an NC circuit an oracle gate that has  $m$  input bits is defined to have size 1 and depth  $\lceil \log m \rceil$ . In an AC circuit (unbounded fan-in) the size and depth of these gates is 1.

We shall use the notation  $\text{AC}^i(C)$  ( $\text{NC}^i(C)$ ) for the class of languages recognized by a family of unbounded fan-in (bounded fan-in) circuits with polynomial size and  $O(\log^i n)$  depth using as oracle a set in  $C$ .

Low and high hierarchies within NP relative to the  $\Sigma_k^P$  and  $\Delta_k^P$ -classes of the polynomial time hierarchy were introduced by Schöning in [Sc-83] and [Sc-86b]. Afterwards, a refinement of these hierarchies based on the  $\Theta_k^P$ -classes was introduced by Long and Sheu in [LS-91]. Taking into account that for any set  $A \in \text{NP}$  it holds that

$$\Sigma_k^P \subseteq \Sigma_k^P(A) \subseteq \Sigma_{k+1}^P \text{ for any } k \geq 1,$$

and as a consequence

$$\begin{aligned} \Delta_k^P &\subseteq \Delta_k^P(A) \subseteq \Delta_{k+1}^P \text{ for any } k \geq 1 \\ \Theta_k^P &\subseteq \Theta_k^P(A) \subseteq \Theta_{k+1}^P \text{ for any } k \geq 2, \end{aligned}$$

they defined the low and the high hierarchies within NP, relative to the  $\Sigma_k^P$ ,  $\Delta_k^P$  and  $\Theta_k^P$ -classes, in a natural way. Concretely, they defined the low hierarchy as follows:

**Definition 2.5**

1. [Sc-83] Relative to the  $\Sigma_k^P$ -classes of the polynomial time hierarchy, the  $\Sigma$ -classes of the low hierarchy are

$$\{\mathbb{L}_k^{\text{P}, \Sigma} \mid k \geq 0\}, \text{ where } \mathbb{L}_k^{\text{P}, \Sigma} = \{A \in \text{NP} \mid \Sigma_k^P(A) \subseteq \Sigma_k^P\}.$$

2. [Sc-86b] Relative to the  $\Delta_k^P$ -classes of the polynomial time hierarchy, the  $\Delta$ -classes of the low hierarchy are

$$\{\mathbb{L}_k^{\text{P}, \Delta} \mid k \geq 1\}, \text{ where } \mathbb{L}_k^{\text{P}, \Delta} = \{A \in \text{NP} \mid \Delta_k^P(A) \subseteq \Delta_k^P\}.$$

3. [LS-91] Relative to the  $\Theta_k^P$ -classes of the polynomial time hierarchy, the  $\Theta$ -classes of the low hierarchy are

$$\{L_k^{P,\Theta} \mid k \geq 2\}, \text{ where } L_k^{P,\Theta} = \{A \in \text{NP} \mid \Theta_k^P(A) \subseteq \Theta_k^P\}.$$

Starting from previous results showed in [Sc-83], Long and Sheu proved some basic relationships among the  $\Sigma$ ,  $\Delta$  and  $\Theta$ -classes of the low hierarchy. One of them shows that

$$L_{k-1}^{P,\Sigma} \subseteq L_k^{P,\Theta} \subseteq L_k^{P,\Delta} \subseteq L_k^{P,\Sigma}.$$

Besides these basic relationships, they showed  $\Theta$ -lowness results for several classes located in the  $\Delta$ -levels of the low hierarchy. In section 6 we will use a property of  $\Theta_k^P$  classes shown in section 3 that will enable us to prove  $\Theta$ -lowness properties for some classes.

### 3. Relating $P_{\log^i}(\text{NP})$ with $\text{AC}^{i-1}(\text{NP})$

We start this section proving a relation between  $P_{\log^i}(\text{C})$  and  $\text{AC}^{i-1}(\text{C})$  for a wide range of classes  $\text{C}$ .

**Proposition 3.1** Let  $\text{C} \neq \emptyset$  be a class of languages closed under  $\leq_m^P$ -reducibility; then for all  $i \geq 1$ , it holds  $P_{\log^i}(\text{C}) \subseteq \text{AC}^{i-1}(\text{C})$ .

**Proof** We proceed by induction on  $i$ . First we consider the case  $i = 1$ . Let  $M^A$  be a fixed polynomial time oracle machine that on inputs of length  $n$  makes  $d(n) \in O(\log n)$  queries to a set  $A$  in  $\text{C}$ . We define  $L$  as the language of pairs  $\langle x, y \rangle$ , such that, interpreting  $y$  as giving the first  $|y|$  query answers of  $M^A$  on input  $x$ , it holds  $|y| < d(|x|)$  and the answer to the query number  $|y| + 1$  is “YES”, or  $|y| = d(|x|)$  and  $M$  accepts  $x$ .

It is easy to see that  $L \leq_m^P A$ ; so language  $L$  belongs to  $\text{C}$ . We shall use this language as oracle to simulate the machine  $M^A$  by a family of circuits. Concretely, the member of this family which operates on inputs of length  $n$  works as follows.

On input  $x$ , the circuit has a first level of queries  $\langle x, y \rangle \in L?$  for all  $y$ ,  $|y| \leq d(n)$ . After that, it determines the right answers (and therefore the right computation) to the queries of  $M^A$  on  $x$  by computing in parallel the unbounded AND's of all the query answers  $\langle x, y \rangle$ ,  $\langle x, y' \rangle$  that satisfy  $y' = y1$  and  $\langle x, y \rangle \in L$  or  $y' = y0$  and  $\langle x, y \rangle \notin L$ . Finally, the circuit computes the OR of all the AND gates and gives the result of  $M^A$  on  $x$ . As  $d(n) \in O(\log n)$  this circuit has polynomial size and constant depth, so the case  $i = 1$  is shown.

We suppose now that the result is true up to  $i - 1$  by induction hypothesis. Let  $M^A$  be a fixed polynomial time oracle machine that on inputs of length  $n$  makes  $d(n) \in O(\log^i n)$  queries to  $A$  in  $\text{C}$ , and let  $p(n)$  be a polynomial bounding the length of configurations of  $M^A$ .

Let us consider the following language:

$$L = \left\{ \langle y, x, j \rangle \mid M^A \text{ starting at the configuration } y \text{ reaches after } \lceil d(|x|) \log^{-1} |x| \rceil \text{ queries a configuration which has a 1 as the } j^{\text{th}} \text{ bit} \right\},$$

It is clear that  $L$  is in  $P_{\log^{i-1}}(A)$  and then, by induction hypothesis, it is recognized by a circuit family  $\{C_{L,n}\}_{n \geq 1}$  in  $\text{AC}^{i-2}(\text{C})$ .

Now, we can simulate  $M^A$  with the circuit family  $\{C_n\}_{n \geq 1}$  defined as follows. The circuit  $C_n$ , which works on inputs of length  $n$ , has  $\log n$  levels each one composed by  $p(n)$  copies of

$C_{L,l(n)}$ , with  $l(n) = p(n) + n + \lceil \log p(n) \rceil$ , placed in parallel. The input of the  $j^{\text{th}}$  circuit  $C_{L,l(n)}$  in level  $k + 1$  is formed by the output of level  $k$  ( $p(n)$  bits), the input of the circuit ( $n$  bits) and  $j$  ( $\lceil \log p(n) \rceil$  bits). On input  $x$ ,  $|x| = n$ , level  $k$  of circuit  $C_n$  gives the configuration of  $M^A$  on  $x$  after  $k \lceil d(n) \log^{-1} n \rceil$  queries have been issued. So, last level gives the configuration  $f$  of  $M^A$  on  $x$  after all the queries have been made.  $C_n$  ends with a small circuit  $D \in \text{AC}^0(\text{P})$  that computes whether  $x \in L(M^A)$  knowing the configuration  $f$ .

Clearly, the language accepted by the family of circuits  $\{C_n\}_{n \geq 1}$  is  $L(M^A)$ , this family is logspace uniform, has polynomial size and  $O(\log^{i-1} n)$  depth.  $\square$

In the next theorem we will see that, when we consider NP, the inclusions are actually equations. This result can be shown using a generalization of the census technique, but we give a detailed proof in order to later also have a proof for other relationships shown in the next sections (see theorem 4.1 and proposition 5.1).

**Theorem 3.2** For all  $i \geq 1$ ,  $\text{P}_{\log^i}(\text{NP}) = \text{AC}^{i-1}(\text{NP})$ .

**Proof** Note that, by proposition 3.1, we only have to prove that  $\text{AC}^{i-1}(\text{NP}) \subseteq \text{P}_{\log^i}(\text{NP})$ . Given a circuit with oracle gates we define the level of a query as follows: queries at the first level are the queries that depend on no other queries, queries at the second level all depend on some query at the first level, and so on.

Let  $\{C_n\}_{n \geq 1}$  be a family of circuits in  $\text{AC}^{i-1}(\text{NP})$  with depth  $d(n) \in O(\log^{i-1} n)$  and  $p(n)$  a polynomial bound on the number of queries in each level. We define  $X$  as the set of sequences  $\langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle$  belonging to  $\{0, 1\}^* \times \{0, 1, \dots, p(n)\}^{d(n)} \times \{0, 1\}$  (where  $n = |x|$ ) such that there exist strings  $s_1, s_2, \dots, s_{d(n)}$  in  $\{0, 1\}^{\leq p(n)}$ , each of them representing possible answers to the queries at levels  $1, 2, \dots, d(n)$  respectively and verifying the following properties:

$s_{d(n)}$  has  $j_{d(n)}$  “YES” answers and these answers are correct if the inputs to queries of level  $d(n)$  are computed from  $x, s_1, s_2, \dots, s_{d(n)-1}$ .

$s_{d(n)-1}$  has  $j_{d(n)-1}$  “YES” answers and these answers are correct if the inputs to queries of level  $d(n) - 1$  are computed from  $x, s_1, s_2, \dots, s_{d(n)-2}$ .

...

$s_1$  has  $j_1$  “YES” answers and these answers are correct if the input to the circuit is  $x$ .

Finally,  $\langle j_1, j_2, \dots, j_{d(n)}, o_j \rangle \geq \langle i_1, i_2, \dots, i_{d(n)}, o_i \rangle$ , where  $\geq$  denotes the lexicographical order and  $o_j \in \{0, 1\}$  is the output of the circuit  $C_n$  on input  $x$  taking  $s_1, s_2, \dots, s_{d(n)}$  as the answers to the queries.

*Facts:*

1.  $X \in \text{NP}$ .
2.  $X$  is closed by lexicographical  $\leq$  order: if  $\langle a_1, a_2, \dots, a_{d(n)}, o_a \rangle \leq \langle b_1, b_2, \dots, b_{d(n)}, o_b \rangle$  and  $\langle x, b_1, b_2, \dots, b_{d(n)}, o_b \rangle \in X$ , then  $\langle x, a_1, a_2, \dots, a_{d(n)}, o_a \rangle$  is also in  $X$ .
3. Fixed  $x$  of length  $n$ , we define
 
$$\langle m_1, m_2, \dots, m_{d(n)}, o_m \rangle = \max \{ \langle i_1, i_2, \dots, i_{d(n)}, o_i \rangle \mid \langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle \in X \}.$$

This maximum verifies:

For all  $l$ ,  $1 \leq l \leq d(n)$ ,  $m_l$  is the number of “YES” answers of  $C_n$  on input  $x$  at the  $l^{\text{th}}$  level of queries.

$o_m$  is the value computed by the circuit.

Fact 1 is easy to prove if we note that the circuits have polynomial size and, in each level of queries, we only have to check the “YES” answers (of queries to a set in NP) supposing that the answers of queries in previous levels are known.

Fact 2 is an easy consequence of the definition of  $X$ .

To show fact 3 just observe that the tuple  $\langle k_1, k_2, \dots, k_{d(n)}, o_k \rangle$  where for all  $l$ ,  $1 \leq l \leq d(n)$ ,  $k_l$  is the number of “YES” answers of  $C_n$  on input  $x$  at the  $l^{\text{th}}$  level of queries and  $o_k$  is the value computed by  $C_n$  on input  $x$ , satisfies that  $\langle x, k_1, k_2, \dots, k_{d(n)}, o_k \rangle \in X$ . Moreover, if there was  $\langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle \in X$  such that  $\langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle > \langle x, k_1, k_2, \dots, k_{d(n)}, o_k \rangle$  it would imply that either

$$\exists j, 1 \leq j \leq d(n) : i_1 = k_1, \dots, i_{j-1} = k_{j-1}, i_j > k_j$$

or otherwise  $o_i > o_k$ . In the first case,  $i_1 = k_1, \dots, i_{j-1} = k_{j-1}$  implies that the queries answered “YES” in the first  $j - 1$  levels of queries must be exactly the same (remember the meaning of  $k_l$ 's). Therefore, queries to oracle gates of level  $j$  are the same in both cases and  $i_j > k_j$  is an obvious contradiction. On the other hand, in the second case, if  $o_i > o_k$  and for all  $j$ ,  $1 \leq j \leq d(n)$  is  $i_j = k_j$ , using a similar argument we also get a contradiction.

Now, using facts 1, 2 and 3, it is easy to prove that  $\text{AC}^{i-1}(\text{NP}) \subseteq \text{P}_{\log^i}(\text{NP})$ . Given an input  $x$  of length  $n$  to the circuit  $C_n$  we can determine if  $C_n$  accepts  $x$  with an oracle machine which proceeds as follows:

Using binary search, it determines

$$\langle m_1, m_2, \dots, m_{d(n)}, o_m \rangle = \max \{ \langle i_1, i_2, \dots, i_{d(n)}, o_i \rangle \mid \langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle \in X \}.$$

As for all  $l$ ,  $1 \leq l \leq d(n)$ , it holds  $0 \leq i_l \leq p(n)$  and  $0 \leq o_i \leq 1$ , the search can be done with at most  $d(n) \log(p(n) + 1) + 1 \in O(\log^i n)$  queries to  $X$ . Knowing the maximum, the machine accepts  $x$  iff  $o_m = 1$ . This oracle machine recognizes the language accepted by the circuits.

Finally, knowing that  $X \in \text{NP}$  by fact 1, we conclude that

$$L(\{C_n\}_{n \geq 1}) \in \text{P}_{\log^i}(\text{NP}),$$

where  $L(\{C_n\}_{n \geq 1})$  denotes the language accepted by the family  $\{C_n\}_{n \geq 1}$ . □

**Remark 3.3** Proof of theorem 3.2 remain valid under relativizations. Therefore, we can generalize the statement of this theorem in the following way: “For any class  $C$  and for all  $i \geq 1$ ,  $\text{P}_{\log^i}(\text{NP}(C)) = \text{AC}^{i-1}(\text{NP}(C))$ ”. In particular, for all  $i \geq 1$  and  $k \geq 1$ , it holds  $\text{P}_{\log^i}(\Sigma_k^P) = \text{AC}^{i-1}(\Sigma_k^P)$ . Moreover, as this proof will be the kernel of those of corollary 3.4 and the results of sections 3 and 4 below, this remark about relativizations will remain valid in those points.

With a similar reasoning and considering  $\text{P}_{\parallel \log^{i-1}}(\text{NP})$  instead of  $\text{AC}^{i-1}(\text{NP})$ , we can rewrite theorem 3.2 in terms of  $\text{P}_{\parallel \log^{i-1}}(\text{NP})$ .

**Corollary 3.4** For any integer  $i \geq 1$ ,  $\text{P}_{\log^i}(\text{NP}) = \text{P}_{\parallel \log^{i-1}}(\text{NP})$ .

**Proof** ( $\subseteq$ ) Observe that  $\text{AC}^{i-1}(\text{NP}) \subseteq \text{P}_{\parallel \log^{i-1}}(\text{NP})$ , therefore this direction follows directly from theorem 3.2.

( $\supseteq$ ) (sketch) Given a polynomial time oracle machine  $M$  that on inputs of length  $n$  makes at most  $O(\log^{i-1} n)$  rounds of parallel queries, we can define (doing a few evident changes) the set  $X$  as in the proof of theorem 3.2 and verifying the same properties. Now, this direction is also clear. □

Using last theorem, we finish this section showing that polynomial time oracle machines that make  $O(\log n)$  queries to a set in  $\Theta_{k+1}^P$  can not recognize sets outside  $\Theta_{k+1}^P$ . This is a surprising property which helps to show  $\Theta$ -lowness results, as we will see in section 6.

**Theorem 3.5** For  $k \geq 1$ , it holds  $P_{\log}(\Theta_{k+1}^P) = \Theta_{k+1}^P$ .

**Proof** First we note that  $\Theta_{k+1}^P$  is closed under  $\leq_m^P$ -reducibility. Then, applying proposition 3.1, we have

$$P_{\log}(\Theta_{k+1}^P) \subseteq AC^0(\Theta_{k+1}^P).$$

Rewriting  $\Theta_{k+1}^P$  as  $AC^0(\Sigma_k^P)$  (see remark 3.3), we get

$$P_{\log}(\Theta_{k+1}^P) \subseteq AC^0(AC^0(\Sigma_k^P)).$$

Note that for any class  $C$ ,  $AC^0(AC^0(C)) = AC^0(C)$ : replacing oracle gates by the corresponding  $AC^0(C)$  circuits we will get an  $AC^0(C)$  circuit (see [Wi-90] for formal proofs of this kind of relationships). Therefore, we can write

$$P_{\log}(\Theta_{k+1}^P) \subseteq AC^0(\Sigma_k^P) = \Theta_{k+1}^P.$$

□

## 4. Binary search over NP with superpolynomial bounding functions

The Boolean hierarchy BH was defined independently by Köbler [Kö-85], Wechsung and Wagner [WW-85], and Cai and Hemachandra [CH-86]. From the various equivalent formulations of the classes  $NP(k)$  of the Boolean hierarchy, we consider the following. For  $k \geq 1$ ,

$$A \in NP(k) \iff \exists B \in NP \text{ such that } c_B(x, l+1) \leq c_B(x, l) \text{ for all } l, 1 \leq l < k \text{ and} \\ c_A(x) \equiv \sum_{l=1}^k c_B(x, l) \pmod{2},$$

and  $BH = \bigcup_{k \geq 1} NP(k)$ . It was shown that the Boolean hierarchy coincides with the constant query classes (for this result and other interesting properties of BH see [KSW-87], [Be-91] and [CGHHSWW-88]).

The Extended Boolean hierarchy, introduced in [Wa-90], considers classes  $NP(r(n))$  for bounding functions  $r(n)$ . Naturally, classes  $NP(r(n))$  are defined as

$$A \in NP(r(n)) \iff \exists B \in NP \text{ such that } c_B(x, l+1) \leq c_B(x, l) \text{ for all } l, 1 \leq l < r(|x|) \text{ and} \\ c_A(x) \equiv \sum_{l=1}^{r(|x|)} c_B(x, l) \pmod{2}.$$

It is shown in [BH-91] and [Wa-90] that  $NP(n^{O(1)}) = \Theta_2^P$ . Also, in the second paper it is proved that

$$NP\left(2^{n^{O(1)}}\right) = \Delta_2^P.$$

Wagner asked in [Wa-90] what could be said about other superpolynomial bounding functions. We answer this question below, using ideas from the proof of theorem 3.2.

**Theorem 4.1** For all  $i \geq 1$ ,

$$P_{\log^i}(\text{NP}) = \text{NP} \left( 2^{O(\log^i n)} \right).$$

**Proof** ( $\supseteq$ ) Given a set  $A \in \text{NP}(r(n))$ , where  $r(n) \in 2^{O(\log^i n)}$ , we know by definition that there exists a set  $B \in \text{NP}$  such that

$$c_B(x, l+1) \leq c_B(x, l) \quad \text{for all } 1 \leq l < r(|x|) \quad \text{and}$$

$$c_A(x) \equiv \sum_{l=1}^{r(|x|)} c_B(x, l) \pmod{2}.$$

We consider a Turing machine  $M$  with oracle  $B$  which on input  $x$  works as follows:

First, it determines the maximum  $l$  such that  $\langle x, l \rangle \in B$ . By the first property of  $B$  this can be done with a binary search with only  $\log r(|x|)$  queries to  $B$ .

Second, using the second property of  $B$ ,  $M$  accepts  $x$  iff the maximum  $l$  is odd.

Clearly,  $L(M, B) = A$ ,  $M$  works in polynomial time and on inputs of length  $n$  makes at most  $O(\log^i n)$  queries.

( $\subseteq$ ) By the theorem 3.2, it is sufficient to prove that  $\text{AC}^{i-1}(\text{NP}) \subseteq \text{NP} \left( 2^{O(\log^i n)} \right)$ .

Given  $\{C_n\}_{n \geq 1}$  a family of circuits in  $\text{AC}^{i-1}(\text{NP})$  with depth  $d(n) \in O(\log^{i-1} n)$  and  $p(n)$  a polynomial bound on the number of queries in each level, we define the set  $X$  as in the proof of theorem 3.2. Remember facts 1, 2 and 3 that this set verifies. From fact 3, it is easy to see that

$$x \in L(C_n) \iff (m = \max \{ \langle i_1, i_2, \dots, i_{d(n)}, o_i \rangle \mid \langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle \in X \}) \text{ is odd.}$$

Taking into account that for all  $l$ ,  $1 \leq l \leq d(n)$ , it holds  $0 \leq i_l \leq p(n)$  and  $0 \leq o_i \leq 1$ , we have,

$$m \leq 2(p(n) + 1)^{d(n)} \in 2^{O(\log^i n)}.$$

Now, considering  $B = \{ \langle x, l \rangle \mid \langle x, l \rangle = \langle x, \langle i_1, i_2, \dots, i_{d(n)}, o_i \rangle \rangle \in X \}$  and using facts 1 and 2, we conclude that

$$L(\{C_n\}_{n \geq 1}) \in \text{NP} \left( 2^{O(\log^i n)} \right).$$

□

## 5. $P_{\log^i}(\text{NP})$ and circuits of small size

Allender and Wilson showed in [AW-90] that for functions  $r(n)$  bounded by a polynomial in  $n$ ,  $\text{NP}(O(r(n)))$  is equal to the class of languages which are reducible to languages in  $\text{NP}$  via reductions of size  $\log r(n) + O(1)$ . Thus, as a consequence, they obtained circuit characterizations of  $P_{\log}(\text{NP})$ . However, it was not known whether similar results hold for other kinds of bounding functions. Now, we shall prove that the same results remain true for functions  $r(n)$  bounded by  $2^{n^{O(1)}}$ .

**Proposition 5.1** For any function  $r(n)$  bounded by  $2^{n^{O(1)}}$ ,  $\text{NP}(O(r(n)))$  is equal to the class of languages which are reducible to languages in  $\text{NP}$  via reductions of size bounded by  $\log r(n) + O(1)$ .

**Proof** ( $\subseteq$ ) This direction can be done following exactly the (left to right) proof of theorem 4 of [AW-90]. Given  $A \in \text{NP}(r(n))$  let  $B$  be the set in NP such that

1.  $x \in A \iff \max \{l \leq r(n) \mid \langle x, l \rangle \in B\}$  is odd,
2. If  $\langle x, l \rangle \in B$ , then  $\langle x, l-1 \rangle \in B$ .

First, query if  $\langle x, 10 \dots 0 \rangle$  is in  $B$ ; call the answer to this query  $b_1$ . Next query if  $\langle x, b_1 1 \dots 0 \rangle$  is in  $B$ . It is clear how to proceed. Note that exactly  $\lceil \log r(n) \rceil$  gates are necessary.

( $\supseteq$ ) Let  $A$  be reducible to a complete set  $B$  in NP via circuits  $\{C_n\}_{n \geq 1}$  of size  $\log r(n) + O(1)$ . Let  $d(n)$  be the depth of circuit  $C_n$  and let  $p_1(n), p_2(n), \dots, p_{d(n)}(n)$  be the number of queries of that circuit on levels  $1, 2, \dots, d(n)$  respectively. Note that these functions can not be superpolynomial and:

$$p_1(n) + p_2(n) + \dots + p_{d(n)}(n) \leq \log r(n) + a,$$

where  $a$  is a constant. We define the set  $X$  as in the proof of theorem 3.2 (but with the new bounds  $p_1(n), \dots, p_{d(n)}(n)$  on the levels of queries). Clearly, facts 1, 2 and 3 remain true. Now, as in the proof of theorem 4.1, we have

$$x \in L(C_n) \iff (m = \max \{ \langle i_1, i_2, \dots, i_{d(n)}, o_i \rangle \mid \langle x, i_1, i_2, \dots, i_{d(n)}, o_i \rangle \in X \}) \text{ is odd};$$

where for all  $l$ ,  $1 \leq l \leq d(n)$ , it holds  $0 \leq i_l \leq p_l(n)$  and  $0 \leq o_i \leq 1$ . Thus

$$m \leq 2(p_1(n) + 1)(p_2(n) + 1) \dots (p_{d(n)}(n) + 1) \in O(r(n)).$$

□

Now, from theorem 4.1 and proposition 5.1 we have a new characterization for  $P_{\log^i}(\text{NP})$ .

**Corollary 5.2** For all  $i \geq 1$ ,  $P_{\log^i}(\text{NP})$  is equal to the class of languages which are reducible to languages in NP via circuits of size  $O(\log^i n)$ .

This corollary gives a strong connection between classes defined by polynomial oracle machines with few queries to an NP set and classes defined by small size circuits with NP oracle gates.

## 6. Applications to the low hierarchy

With the contribution of several authors, the following classes have been located in the  $\Delta$ -levels of the low hierarchy

$L_2^{\text{P}, \Delta}$  : sparse NP sets [KS-85],  $\text{NP} \cap (\text{co-NP}/\log)$  [BS-92],  $\text{NP} \cap \text{APT}$  [KS-85], NP sets that are standard or general left cuts for some real number  $x$  [AH-92], NP sets that are  $\leq_m^{\text{P}}$ -reducible to some sparse set [AH-92], and NP sets that are  $\leq_T^{\text{P}}$ -equivalent to some tally set [BB-86].

$L_3^{\text{P}, \Delta}$  : co-sparse NP sets [KS-85],  $\text{NP} \cap \text{P-close}$  [Sc-86b], NP sets that are  $\leq_{1-tt}^{\text{P}}$ -reducible to some sparse set [AH-92], and NP sets that are  $\leq_T^{\text{P}}$ -equivalent to some sparse set [Sc-86a].

For each of these classes, sparse sets play a key role in their being low classes. The  $\Delta$ -lowness of sparse NP sets can be shown starting from the following result of Mahaney [Ma-82].

(1) if  $S$  is a sparse set in NP, then  $\text{NP}(S) \subseteq \Delta_2^P$ .

Now, applying the P-operator to both sides of the inclusion  $\Sigma_1^P(S) \subseteq \Delta_2^P$  (1), we get

$$\Delta_2^P(S) = P(\Sigma_1^P(S)) \subseteq P(\Delta_2^P) = \Delta_2^P,$$

and immediately,  $S \in L_2^{P,\Delta}$ .

Kadin [Ka-89] later improved Mahaney's result (1) in the following way,

(2) if  $S$  is a sparse set in NP, then  $\text{NP}(S) \subseteq \Theta_2^P$ .

As it is noted in [LS-91], this result suggests that sparse NP sets may actually belong to  $L_2^{P,\Theta}$ . In fact, Long and Sheu proved that all the classes in  $L_3^{P,\Delta}$  given before are actually in  $L_3^{P,\Theta}$ , and that almost all ones in  $L_2^{P,\Delta}$  actually belong to  $L_2^{P,\Theta}$  (see [LS-91]). The only classes from  $L_2^{P,\Delta}$  that they could not locate in  $L_2^{P,\Theta}$  were the classes  $\text{NP} \cap E_T(\text{TALLY})$  and NP sets that are standard or general left cuts for some real number  $x$ . Concretely, for  $\text{NP} \cap E_T(\text{TALLY})$ , they established the following: if  $A \in \text{NP} \cap E_T(\text{TALLY})$  and there is a tally set  $T \in \text{NP}$  such that  $A \equiv_T^P T$ , then  $A \in L_2^{P,\Theta}$ . On the other hand, if every tally set  $T$  such that  $A \equiv_T^P T$  is in  $\Delta_2^P - \Theta_2^P$ , then  $A \notin L_2^{P,\Theta}$ . Finally, if neither of these two cases occur, they did not know whether  $A$  belongs to  $L_2^{P,\Theta}$ .

Using theorem 3.5 of section 3 we can solve the problem from above. First at all, we point out how theorem 3.5 can be applied to show  $\Theta$ -lowness properties. For example, to see that sparse NP sets are in  $L_2^{P,\Theta}$ , we apply the  $P_{\log}$ -operator to both sides of the inclusion  $\Sigma_1^P(S) \subseteq \Theta_2^P$  (2) to obtain

$$\Theta_2^P(S) = P_{\log}(\Sigma_1^P(S)) \subseteq P_{\log}(\Theta_2^P) = \Theta_2^P,$$

(where the last equivalence comes from theorem 3.5) and therefore  $S \in L_2^{P,\Theta}$ . Concerning the  $\Theta$ -lowness of  $\text{NP} \cap E_T(\text{TALLY})$ , first we show

**Proposition 6.1** Let  $A$  be a set such that  $A \leq_T^P T$  where  $T$  is a tally set in the class  $\Theta_2^P$ , then  $\text{NP}(A) \subseteq \Theta_2^P$ .

**Proof** Let  $N$  be a nondeterministic polynomial time oracle machine with time bounded by  $q(n)$  on inputs of length  $n$ . We will show that  $L(N^A) \in \text{AC}^0(\text{NP})$ , and then, by theorem 3.2, we will get the result.

On inputs of length  $n$  the machine  $N$  only asks queries of length at most  $q(n)$  to  $A$ . By hypothesis, there exists a deterministic polynomial time oracle machine  $M$  with time bounded by  $p(n)$  such that  $A$  is the language recognized by  $M^T$ . On inputs of length  $m$ , the machine  $M$  can ask queries of length at most  $p(m)$  to  $T$  and the relevant queries are a subset of the set  $\{1, 1^2, \dots, 1^{p(m)}\}$ .

Let  $t(n)$  be the string of length  $p(q(n))$  that represents the answers to the queries  $1 \in T?, 1^2 \in T?, \dots, 1^{p(q(n))} \in T?$  If string  $t(n)$  is known, it is easy to simulate machine  $N^A$  on inputs of length  $n$  without using any oracle. In other words, there exists a nondeterministic polynomial time Turing machine  $N'$  such that  $N'(x, t(|x|)) = N^A(x)$ .

Now, we can construct the  $\text{AC}^0(\text{NP})$  circuit that recognizes  $L(N^A)$ . On inputs of length  $n$  the circuit has a first level of query gates  $1 \in T?, 1^2 \in T?, \dots, 1^{p(q(n))} \in T?$ ; note that each one of them can be replaced by an  $\text{AC}^0(\text{NP})$  circuit because  $T$  is in  $\Theta_2^P$ . Let  $t(|x|)$  be the string of outputs of these gates; following the first level of queries the circuit has a gate with input  $(x, t(|x|))$  querying to the language recognized by  $N'$ . The result is an  $\text{AC}^0(\text{NP})$  circuit that accepts  $L(N^A)$ .  $\square$

Now, applying the  $P_{\log}$ -operator to the containment  $\text{NP}(A) \subseteq \Theta_2^P$ , it follows

**Corollary 6.2** Let  $A$  be a set such that  $A \leq_T^P T$  where  $T$  is a tally set in the class  $\Theta_2^P$ , then  $\Theta_2^P(A) = \Theta_2^P$ .

As a consequence of corollary 6.2 we can answer affirmatively the question before. Therefore, the situation for the class  $\text{NP} \cap E_T(\text{TALLY})$  is as follows: if  $A$  is in  $\text{NP}$  and  $A \equiv_T^P T$ , where  $T \in \Delta_2^P - \Theta_2^P$ , then  $A \notin L_2^{P,\Theta}$ . On the other hand, if  $A \equiv_T^P T$ , where  $T \in \Theta_2^P$ , then  $A \in L_2^{P,\Theta}$ . These two cases cover all the possibilities for  $T$  because if  $A$  is in  $\text{NP}$  and  $A \equiv_T^P T$ ,  $T$  has to belong to  $\Delta_2^P$ . For the class of  $\text{NP}$  sets that are general or standard left cuts of some real number  $x$  the situation is analogous because these sets are in  $E_T(\text{TALLY})$  (see [AH-92]).

## 7. Separation with oracles

In [LT-91] a set  $A$  is given such that the  $A$ -relativized counterparts of  $\Theta_2^P$  and  $\Delta_2^P$  are different (this is also shown in [BH-91]). Here, applying similar methods, we extend that result by showing that there exists a relativized world where  $P_{\log^i}(\text{NP})$  is different from  $P_{\log^{i+1}}(\text{NP})$  for any integer  $i \geq 1$ .

**Theorem 7.1** There exists a recursive set  $A$  such that for any integer  $i \geq 1$ :

$$P_{\log^{i+1}}(\text{NP}(A)) - P_{\log^i}(\text{NP}(A)) \neq \emptyset.$$

**Proof** We will introduce a language  $L_{i+1}(A)$  with the property that for any  $A$ ,  $L_{i+1}(A) \in P_{\log^{i+1}}(\text{NP}(A))$ . Afterwards, a specific oracle  $A$  will be constructed so that for all  $i \geq 1$ ,  $L_{i+1}(A) \notin P_{\log^i}(\text{NP}(A))$ . Let  $L_{i+1}(A)$  be the set defined as follows:

$$L_{i+1}(A) = \left\{ 0^n \mid A^{\lceil \log^{i+1} n \rceil} = \emptyset \text{ or the minimum word of length } \lceil \log^{i+1} n \rceil \text{ in } A \text{ is even} \right\}.$$

First, we can see that for any  $A$ ,  $L_{i+1}(A) \in P_{\log^{i+1}}(\text{NP}(A))$ : Note that the language  $S = \{y \mid \exists x \in A, |x| = |y|, x \leq y\}$  belongs to  $\text{NP}(A)$ , and for any integer  $m \geq 0$ , it verifies:

i)  $\min S^m = \min A^m$

ii) if  $z$  and  $y$  are words of length  $m$  such that  $z \geq y$  and  $y \in S$ , then  $z$  belongs to  $S$ .

Now it is clear how to proceed, doing a binary search over the words of length  $\lceil \log^{i+1} n \rceil$  we only have to do  $\lceil \log^{i+1} n \rceil$  queries to  $S$  to find the minimum.

Second, we construct a specific oracle  $A$  such that  $L_{i+1}(A) \notin P_{\log^i}(\text{NP}(A))$ . Note that this class is the same as  $P_{\log^i}(\text{NP}(A))$ , as it is shown in corollary 3.4. Let us consider  $M_1, M_2, \dots$  and  $N_1, N_2, \dots$  enumerations of the deterministic and nondeterministic Turing machines respectively, where a machine  $a$  has running time bounded by a polynomial  $p_a$ . It is well known that, for every set  $B$ , the set:

$$K(B) = \{ \langle j, y, 0^t \rangle \mid N_j \text{ with oracle } B \text{ accepts } y \text{ in at most } t \text{ steps} \}$$

is complete for the class  $\text{NP}$  relative to  $B$ . So, we can suppose that deterministic machines always make queries of type  $\langle j, y, 0^t \rangle$  to the oracle  $K(A)$ .

$A$  is constructed in stages. At stage  $s = \langle a, i, c \rangle$ , if the machine  $M_a$  on input  $0^{n_s}$  makes at most  $c \log^{i-1} n_s$  rounds of parallel queries, we will add words of length  $\lceil \log^{i+1} n_s \rceil$  to  $A$ , diagonalizing away from the language recognized by  $M_a$ .

Let us suppose that  $M_a$  makes at most  $c \log^{i-1} n_s$  rounds of parallel queries (otherwise, we finish stage  $s$ ). We say that machine  $M_a$  behaves correctly with input  $0^{n_s}$  and oracle  $K(A)$  if it accepts (rejects), and the minimum word of length  $\lceil \log^{i+1} n_s \rceil$  in  $A$  is even (odd). If so, we will add a new odd (even) word  $w$  to  $A$  of length  $\lceil \log^{i+1} n_s \rceil$  and that is smaller than the existing ones, in such a way that it does not affect the positive answers of the queries to  $K(A)$ . This property can be achieved by keeping a list  $A'$  of words that are not valid choices for  $w$ : for every query  $q = \langle j, y, 0^t \rangle$  of  $M_a$  to  $K(A)$  answered positively, we will include in  $A'$  the list of words that are not in  $A$  and are queried in a certain accepting path of  $N_j$  on input  $y$ .

Machine  $M_a$ , with the new oracle, either behaves incorrectly or has a round of queries  $r$ , ( $1 \leq r \leq c \log^{i-1} n_s$ ) such that, at any previous round  $r'$ ,  $1 \leq r' < r$ , the answers to the queries remains the same, but at round  $r$  there is, at least, one more query answered positively. In other words, if we denote by  $\langle u_1, u_2, \dots, u_{d(n_s)} \rangle$ , where  $d(n_s) = c \log^{i-1} n_s$ , the sequence of numbers of queries answered positively in each round by the machine  $M_a$  with oracle  $K(A)$ , the new sequence  $\langle v_1, v_2, \dots, v_{d(n_s)} \rangle$ , corresponding to  $M_a$  with the new oracle  $K(A \cup \{w\})$ , will be greater (in lexicographical order) than  $\langle u_1, u_2, \dots, u_{d(n_s)} \rangle$ . Since the number of different sequences is bounded by  $b(n_s) = (p_a(n_s) + 1)^{d(n_s)}$  that is a function in  $2^{O(\log^i n_s)}$ , repeating the procedure above  $b(n_s)$  times we can diagonalize away from  $M_a$ .

**stage 0**

$$A(0) := \emptyset$$

$$n_0 := 0$$

**endstage**

**stage  $s = \langle a, i, c \rangle$**

Let  $n_s$  be the smallest integer such that

(1)  $\log^{i+1} n_s$  is larger than the length of any string queried or added to  $A$  at any previous stage

(2)  $2(b(n_s) + p_a^2(n_s)) < 2^{\log^{i+1} n_s}$ , where  $b(n_s) = (p_a(n_s) + 1)^{c \log^{i-1} n_s}$

Check that  $M_a(0^{n_s})$  has at most  $c \log^{i-1} n_s$  rounds. If not, skip the stage

$$A(s) := A(s-1)$$

$$d := 1; u_d := 1^{\lceil \log^{i+1} n_s \rceil}$$

**repeat**

$$A(s) := A(s) \cup \{u_d\}$$

$$A' := \emptyset$$

compute all the queries made by  $M_a$  on input  $0^{n_s}$  with oracle  $A(s)$

**for** all the queries  $q_l = \langle j_l, y_{j_l}, 0^{t_{j_l}} \rangle$  **do**

**if**  $N_{j_l}$  with oracle  $A(s)$  accepts  $y_{j_l}$  in less than  $t_{j_l}$  steps **then**

$$A' := A' \cup \{\text{words queried in the minimum accepting path for } N_{j_l}^{A(s)} \text{ that are not in } A(s)\}$$

**endif**

**endfor**

$$d := d + 1$$

$$u_d := \max\{x : |x| = \lceil \log^{i+1} n_s \rceil \text{ and } (x \text{ is even} \iff u_{d-1} \text{ is odd}) \\ \text{and } x \notin A' \text{ and } x < u_{d-1}\}$$

**until**  $0^{n_s} \in L(M_a, K(A(s))) \iff 0^{n_s} \in L(M_a, K(A(s) \cup \{u_d\}))$

**if**  $M_a(K(A(s)))$  behaves correctly **then**

$A(s) := A(s) \cup \{u_d\}$

**endif**

**endstage.**

Note that the selected word  $u_d$  included in  $A(s)$  in each iteration always exists since in  $A'$  there are at most  $p_a^2(n_s)$  reserved words and before including  $u_d$  we could have included at most  $b(n_s) \in 2^{O(\log^i n_s)}$  words of length  $\lceil \log^{i+1} n_s \rceil$  in  $A$ .  $\square$

**Remark 7.2** The set  $A$  constructed above gives a relativized separation between  $P_{\log^i}(\text{NP})$  and  $P_g(\text{NP})$  for any function  $g(n) \in \omega(\log^i n)$ : Essentially, the only necessary change in the proof is to replace all references to words of length  $\lceil \log^{i+1} n \rceil$  by references to words of length  $\lceil g(n) \rceil$ . With this change, the condition (2) of the algorithm becomes  $2(b(n_s) + p_a^2(n_s)) < 2^{\lceil g(n_s) \rceil}$  that is also satisfiable.

## Conclusions

We have shown different circuit characterizations of the class  $P_{\log^i}(\text{NP})$ : this class coincides with  $\text{AC}^{i-1}(\text{NP})$  and also with the class of sets recognized by NP-relativized circuits of size  $O(\log^i n)$ . All the proofs given are valid under relativizations, so these characterizations can be generalized to arbitrary levels of the polynomial time hierarchy. Ogiwara has completed this kind of equivalences obtaining an NC circuit characterization of these classes: he has shown that  $\text{AC}^{i-1}(\text{NP}) = \text{NC}^i(\text{NP})$  (see [Og-94]). Note that these results show that the well known inclusions  $\text{AC}^0 \subset \text{NC}^1 \subseteq L$  collapse when the relativized versions to NP are considered, in contrast with the general relativized case (see [Wi-87]).

## Acknowledgements

We are very grateful to José L. Balcázar and Borja Valles for reading earlier versions of this paper. We would like to thank Eric Allender, Fred Green, Birgit Jenner, Ming-Jye Sheu and Chris Wilson for helpful discussions. Two anonymous referees deserve our thanks for many valuable suggestions and comments.

## References

- [AH-92] E. Allender, L. Hemachandra, Lower bounds for the low hierarchy, *Journal of the ACM*, **39** (1), (1992), pp. 234-250.
- [AW-90] E. Allender, C. B. Wilson, Width-bounded reducibility and binary search over complexity classes, *Proc. 5th IEEE Conf. on Structure in Complexity Theory*, (1990), pp. 122-129.
- [BB-86] J. L. Balcázar, R. Book, Sets with small generalized Kolmogorov complexity, *Acta Informatica*, **23**, (1986), pp. 679-688.
- [BH-91] S. Buss, L. Hay, On truth-table reducibility to SAT, *Information and Computation*, **91**, (1991), pp. 86-102.
- [BS-92] J. L. Balcázar, U. Schöning, Logarithmic advice classes, *Theoretical Computer Science*, **99**, (1992), pp. 279-290.
- [BT-89] R. Book, S. Tang, A note on sparse sets and the polynomial-time hierarchy, *Information Processing Letters*, **33** (3), (1989), pp. 141-143.

- [Be-91] R. J. Beigel, Bounded queries to *SAT* and the Boolean hierarchy, *Theoretical Computer Science*, **84**, (1991), pp. 199-223.
- [CGHHSWW-88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, G. Wechsung, The Boolean hierarchy I: Structural Properties, *SIAM J. Comp.*, **17** (6), (1988), pp. 1232-1252.
- [CH-86] J. Cai, L. Hemachandra, The Boolean hierarchy: hardware over NP, *Proc. 1st Conference on Structure in Complexity Theory, LNCS*, **223**, Springer-Verlag, (1986), pp. 105-124.
- [KS-85] K. Ko, U. Schöning, On circuit-size and the low hierarchy in NP, *SIAM J. Comput.*, **14** (1), (1985), pp. 41-51.
- [KSW-87] J. Köbler, U. Schöning, K. Wagner, The difference and the truth-table hierarchies for NP, *R.A.I.R.O.*, **21**, (1987), pp. 419-435.
- [Ka-89] J. Kadin,  $P^{NP[\log n]}$  and sparse Turing-complete sets for NP, *J. Comput. System Sci.*, **39** (3), (1989), pp. 282-298.
- [Kö-85] J. Köbler Untersuchung verschiedener polynomieller Reduktionsklassen von NP, *thesis*, University of Stuttgart, (1985).
- [LS-91] T. Long, M-J. Sheu, A refinement of the low and high hierarchies, Technical report OSU-CISRC-2/91-TR6, The Ohio State University, (1991).
- [LT-91] A. Lozano, J. Torán, Self-reducible sets of small density, *Math. Systems Theory*, **24**, (1991), pp. 83-100.
- [Ma-82] S. Mahaney, Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis, *J. Comput. System Sci.*, **25**, (1982), pp. 130-143.
- [Og-94] M. Ogiwara,  $NC^k(NP) = AC^{k-1}(NP)$ , *STACS 94, LNCS*, **775**, Springer-Verlag, (1994), pp. 313-324.
- [Sc-83] U. Schöning, A low and a high hierarchy within NP, *J. Comput. System Sci.*, **27**, (1983), pp. 14-28.
- [Sc-86a] U. Schöning, Complete sets and closeness to complexity classes. *Mathematical Systems Theory* **19**, (1986), pp. 29-41.
- [Sc-86b] U. Schöning, Complexity and Structure, *LNCS*, **211**, Springer-Verlag, (1986).
- [WW-85] G. Wechsung, K. Wagner, On the Boolean closure of NP, manuscript 1985 (extended abstract as: G. Wechsung, On the Boolean closure of NP, *Proc. Conf. Fundam. Comp. Theory, Cottbus 1985, LNCS*, **199**, (1985), pp. 485-493).
- [Wa-90] K. Wagner, Bounded query classes, *SIAM J. Comput.*, **19** (5), (1990), pp. 833-846.
- [Wi-87] C. B. Wilson, Relativized NC, *Math. Systems Theory*, **20**, (1987), pp. 13-29.
- [Wi-90] C. B. Wilson, Decomposing NC and AC, *SIAM J. Comput.*, **19** (2), (1990), pp. 384-396.