Characterizations of some complexity classes between Θ_2^p and Δ_2^p

Jorge Castro
Carlos Seara

Report LSI–90–27

Abstract

We give some characterizations of the classes $P^{NP}[O(\log^k n)]$. First, we show that these classes are equal to classes $AC^{k-1}(NP)$. Second, we prove that they are also equivalent to some classes defined in the Extended Boolean hierarchy. Finally, we show that there exists a strong connection between classes defined by polynomial time Turing machines with few queries to an NP oracle and classes defined by small size circuits with NP oracle gates. With these results we solve open questions arosed by K. W. Wagner and by E. Allender and C. B. Wilson.

Resum

En aquest report donem algunes caracteritzacions de les classes $P^{NP}[O(\log^k n)]$. En primer lloc demostrem que aquestes classes són iguals a les classes $AC^{k-1}(NP)$. A continuació provem que les classes anteriors són també equivalents a algunes classes definides en la jerarquia Booleana Extesa. Finalment demostrem que existeix una forta connexió entre classes definides per màquines de Turing de temps polinòmic amb poques preguntes a un oracle NP i classes definides per circuits de tamany petit amb portes oracle que consulten conjunts a NP. Amb aquests resultats resolem problemes oberts plantejats per K. W. Wagner i per E. Allender i C. B. Wilson.

Characterizations of some complexity classes between Θ_2^p and Δ_2^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. Matemática Aplicada II
Universitat Politècnica de Catalunya
Pau Gargallo 5
08028 Barcelona, Spain

Abstract

We give some characterizations of the classes $P^{NP}[O(\log^k n)]$. First, we show that these classes are equal to classes $AC^{k-1}(NP)$. Second, we prove that they are also equivalent to some classes defined in the Extended Boolean hierarchy. Finally, we show that there exists a strong connection between classes defined by polynomial time Turing machines with few queries to an NP oracle and classes defined by small size circuits with NP oracle gates. With these results we solve open questions that arose in [Wa-88] and [AW-90].

1. Introduction

Wagner gave in his paper [Wa-88] some characterizations of the class Θ_2^p. One of them characterizes Θ_2^p as the class of languages recognized by polynomial time Turing machines which make at most $O(\log n)$ queries to an NP oracle on inputs of length n. Another one characterizes Θ_2^p as the class of languages obtained by placing polynomial bounds on the levels of the Extended Boolean hierarchy.

Wilson introduced the notion of relativized circuits by allowing oracle gates in the circuits, and he has studied properties of relativized versions of NC and AC. He defined an oracle gate as a k-input, one-output gate which, on an 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. For NC classes each oracle gate counts as a depth logarithmic in the number of its input wires, while for AC classes these gates count 1. The motivations of these definitions are argued in [Wi-87] and [Wi-90] (compare also with the notion of NC^1-reducibility introduced in [Co-85]). In this paper we only consider relativizations to sets in NP; note that with this restriction we can only obtain classes of languages which are included in Δ_2^p. The first motivation of our work was to find relationships between these classes and those mentioned in [Wa-88].

Below we show connections between classes of languages recognized by polynomial time Turing machines which make polylog queries to an NP oracle and other classes defined on different ways. More exactly, in section 3 we show that the class of languages defined by P^{NP} Turing machines with at most $O(\log^k n)$ queries is the same as the class defined by AC^{k-1} circuits with oracle gates in NP. In section 4 we characterize the Extended Boolean hierarchy for some superpolynomial bounding functions such as the classes just mentioned. Finally, in section 5 we show a last characterization as the class of languages which are reducible to languages in NP via circuits of polylog size. With these results we solve questions asked by Wagner and by Allender and Wilson in their papers [Wa-88] and [AW-90].

* Work partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075, project ALCOM.

2. Definitions and notation

We fix the notation that we shall use in the following sections.

Definition For any class C, P^C denotes the class of languages accepted by deterministic polynomial time machines using a set in C as oracle.

$$L \in P^C \iff \exists B \in C : L \leq_T^P B.$$

Definition For all $k \geq 1$, $P^C[O(\log^k n)]$ is the class of languages in P^C that are accepted by machines that make at most $O(\log^k n)$ queries on inputs of length n.

$$L \in P^C[O(\log^k n)] \iff \exists B \in C : L \leq^P_{T[O(\log^k n)]} B.$$

Since queries to any oracle of NP can be translated into queries for SAT, $P^{NP} = P^{SAT}$. We use these terms as synonyms and we also use the terms $P^{NP}[O(\log^k n)]$ and $P^{SAT}[O(\log^k n)]$ as equivalent. For any string x, we write $|x|$ for the length of x.

Definition $P_{\|}^{NP}$ is the class of languages accepted by deterministic polynomial time Turing machines using an oracle $B \in NP$ such that, on any input a list of all the queries is formed before any of them is made.

The classes $\Theta_2^p = P^{NP}[O(\log n)]$ and $\Delta_2^p = P^{NP}$ are well known, and the first one has a lot of different characterizations as is shown in [Wa-88]. The Boolean hierarchy has been studied independently by different authors who gave equivalent definitions for this hierarchy. In the Section 8 of [Wa-88] the reader can find the definitions and main results about the Boolean hierarchy. In the Section 9 of that paper, it is introduced the Extended Boolean hierarchy as the hierarchy of classes $NP(r)$ (r is a function) defined as follows:

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

where c_A denotes the characteristic function of the set A. It was known that $NP(n^{O(1)}) = \Theta_2^P$ and $NP\left(2^{n^{O(1)}}\right) = \Delta_2^P$, but similar results for other superpolynomial bounding functions were not known.

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 the circuit 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 resource 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^{th} circuit could be independent of the $(n-1)^{th}$ 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 A circuit family $\{C_n\}$ is $O(\log n)$-uniform if there is a logspace deterministic Turing machine which on any input of length n outputs an encoding of C_n.

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

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

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

We shall use the notation $AC^k(A)$ for the class of languages recognized by a family of unbounded fan-in circuits with polynomial size and $O(\log^k n)$ depth using A as oracle; and we define $AC^k_{[i]}(A)$ as the class obtained if we restrict the circuits to at most i oracle gates on each path from an input node to the output node.

3. Relating $P^{NP}[O(\log^k n)]$ with $AC^{k-1}(NP)$

Let us begin characterizing Θ_2^p as the class of languages recognized by families of circuits of polynomial size, constant depth and with at most one oracle gate on each path from an input node to the output node.

Proposition 1 $P^{NP}[O(\log n)] = AC^0_{[1]}(NP)$

Proof Let M^{SAT} be a fixed polynomial time Turing machine that makes at most $c \log n$ queries to SAT on inputs of length n. We define the language:

$$L = \{\langle x, y\rangle \mid \text{the answer to the query number } |y|+1 \text{ of } M^{SAT} \text{ running on } x \text{ is "YES" supposing that the string } y \text{ records the answers to the first } |y| \text{ queries}\}$$

We consider also the language:

$$L' = \{\langle x, a\rangle \mid M \text{ accepts } x \text{ using } a \text{ as oracle string (the answer to the } k^{th} \text{ query is interpreted as the } k^{th} \text{ bit of } a)\}$$

It is easy to see that the language L is in NP and the language L' is in P. We shall use these languages as oracles to simulate the machine M^{SAT} by a family of circuits. The Figure 1 shows the member of this family which operates on inputs of length n.
On input x, this circuit has a first level of queries $\langle x, y\rangle \in L$ for all y, $|y| \leq c \log n$; with the answers it can determine the right answers to the queries of M^{SAT} on x. Moreover, in the same level, the circuit asks if $\langle x, a\rangle$ is in L' for all a, $|a| \leq c \log n$. Note that in this level there are only a polynomial number of queries because the length of y and a are logarithmic. When the circuit knows what is the right string y of answers of M^{SAT} on x, it chooses the suitable a with the help of a selector circuit.

The other inclusion is trivial considering that $AC^0_{[1]}(NP)$ is included in $P^{NP}_{\|}$ and this one is equal to $P^{NP}[O(\log n)]$ (Th. 5.4 [Wa-88]). □

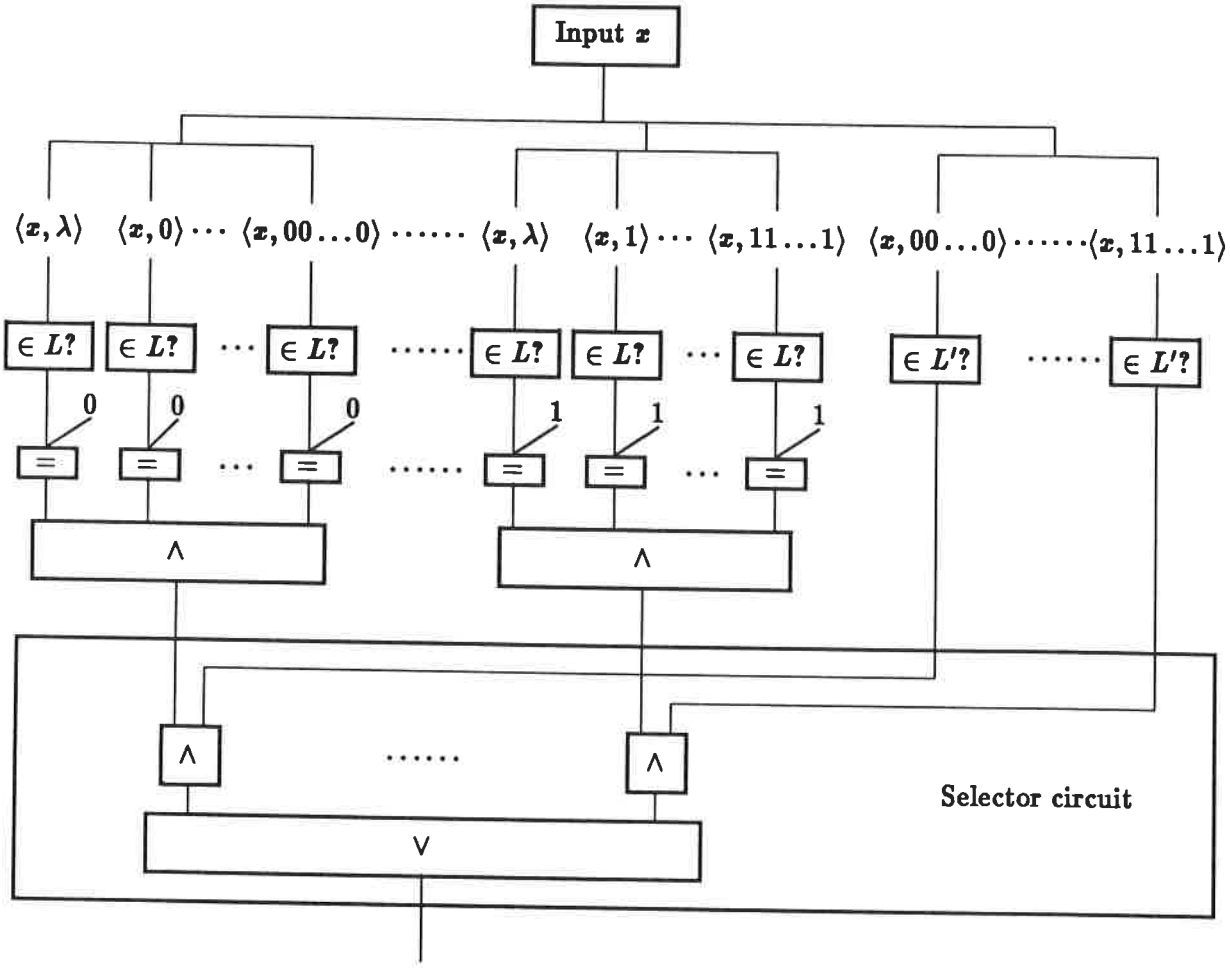

Figure 1.

Buss and Hay [BH-88] showed that, for polynomial time Turing machines, a constant number of rounds of parallel queries to SAT can be reduced to one round of parallel queries. By a round of parallel queries to SAT we mean that the Turing machine writes a set of strings separated by delimiters on a query tape and then invokes an oracle for SAT; the oracle returns a string of YES/NO answers on an answer tape which specify membership of each query string to SAT. Note that within one round of parallel queries, all of them must be formulated before any answer are known. Using this fact it is not difficult to prove that $AC^0(NP)$ is equal to $P^{NP}[O(\log n)]$: observe that every language belonging to the first class is recognized by a polynomial time Turing machine which makes a constant number of rounds of parallel queries to SAT and then, by [BH-88] it can be recognized by another one which makes only one round of parallel queries. Finally, considering the equality $P_{\|}^{NP} = P^{NP}[O(\log n)]$ mentioned before, we have:

Proposition 2 $P^{NP}[O(\log n)] = AC^0(NP)$ □

Now, we shall generalize this last result by allowing more queries to SAT, more exactly, we ask about $O(\log^k n)$ queries to SAT. First we prove the following proposition.

Proposition 3 For all $k \geq 1$, $P^{NP}[O(\log^k n)] \subseteq AC^{k-1}(NP)$

Proof We proceed by induction on k:

$k = 1$, it is part of Proposition 2.

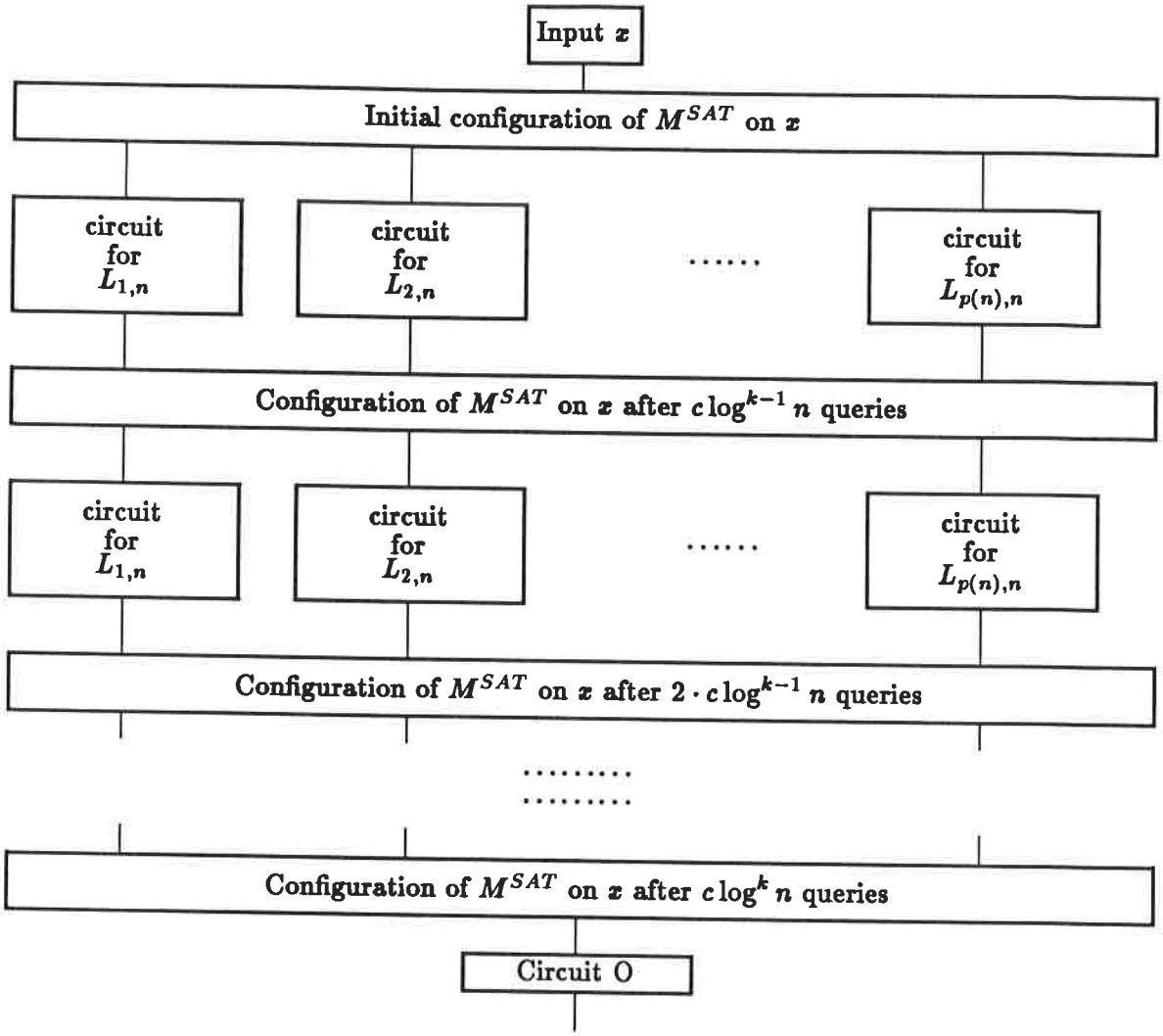

Figure 2.

We suppose that the result is true up to $k-1$ by induction hypothesis. Fix a polynomial time Turing machine M with at most $c\log^k n$ queries to SAT on inputs of length n, and let us consider the following languages:

$$L_{1,n} = \{y \mid M^{SAT} \text{ starting at the configuration } y \text{ reaches after } c\log^{k-1} n \text{ queries a configuration which has a 1 as the first bit }\}$$
$$L_{2,n} = \{y \mid M^{SAT} \text{ starting at the configuration } y \text{ reaches after } c\log^{k-1} n \text{ queries a configuration which has a 1 as the second bit}\}$$
$$\ldots$$
$$L_{p(n),n} = \{y \mid M^{SAT} \text{ starting at the configuration } y \text{ reaches after } c\log^{k-1} n \text{ queries a configuration which has a 1 in the } p(n)^{th} \text{ bit}\}$$

where $p(n)$ is a polynomial bounding the length of configurations corresponding to inputs of length n.

Note that in these languages there may exist configurations y such that M^{SAT} does not reach them on any input. It is clear that all these languages are in $P^{NP}[O(\log^{k-1} n)]$ and then, by induction hypothesis, all of them have $AC^{k-2}(NP)$ circuits. Now, let C_n be the circuit on inputs of length n described by Figure 2.

This circuit on input x computes in $\log n$ levels the configuration f of M^{SAT} on x after all the queries have been done. All the levels are equal and they are composed by $AC^{k-2}(NP)$ circuits for $L_{1,n}, L_{2,n}, \ldots L_{p(n),n}$ placed in parallel. The circuit C_n ends with a small circuit $O \in AC^0(P)$ which computes whether $x \in L\left(M^{SAT}\right)$ knowing what is the configuration f.

Clearly, the language accepted by the family of circuits $\{C_n\}$, $n \geq 1$ is $L\left(M^{SAT}\right)$, this family is logspace uniform, have polynomial size and $O(\log^{k-1} n)$ depth. □

Now, we shall see that the inclusions in proposition 3 are actually equations.

Theorem 4 For all $k \geq 1$, $P^{NP}[O(\log^k n)] = AC^{k-1}(NP)$

Proof We only have to prove that $AC^{k-1}(NP) \subseteq P^{NP}[O(\log^k n)]$. 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 $\{C_n\}$ be a family of circuits in $AC^{k-1}(NP)$ with depth $d(n) \in O(\log^{k-1} n)$ and $p(n)$ a polynomial bound on the number of queries in each level. We define X as the set formed by sequences $\langle x, i_1, i_2, \ldots, i_{d(n)}, o\rangle$ belonging to $\{0,1\}^* \times \{0,1,\ldots,p(n)\}^{d(n)} \times \{0,1\}$ (where $n = |x|$) such that there exist strings $A_1, A_2, \ldots, A_{d(n)}$ in $\{0,1\}^{\leq p(n)}$, each of them representing possible answers to the queries at levels $1, 2, \ldots, d(n)$ respectively and verifying the following:

$A_{d(n)}$ has $j_{d(n)}$ "YES" answers and these answers are correct if the inputs to queries of level $d(n)$ are computed from $x, A_1, A_2, \ldots, A_{d(n)-1}$.
$A_{d(n)-1}$ 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 $x, A_1, A_2, \ldots, A_{d(n)-2}$.
...
...
A_1 has j_1 "YES" answers and these answers are correct if the input to the circuit is x.
Finally, $\langle j_1, j_2, \ldots, j_{d(n)}, o'\rangle \geq \langle i_1, i_2, \ldots, i_{d(n)}, o\rangle$, where $\geq$ denotes the lexicographical order and $o' \in \{0,1\}$ is the output of the circuit C_n on input x taking $A_1, A_2, \ldots, A_{d(n)}$ as the answers to the queries.

Facts:

1. $X \in NP$.
2. X is closed by lexicographical $\leq$ order: if $\langle a_1, a_2, \ldots, a_{d(n)}, o_a\rangle \leq \langle b_1, b_2, \ldots, b_{d(n)}, o_b\rangle$ and $\langle x, b_1, b_2, \ldots, b_{d(n)}, o_b\rangle \in X$, then $\langle x, a_1, a_2, \ldots, a_{d(n)}, o_a\rangle$ is also in X.
3. Fixed x of length n, we define

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

This maximum verifies:

m_1 is the number of "YES" answers of C_n on input x at the first level of queries.
m_2 is the number of "YES" answers of C_n on input x at the second level of queries.
.....
$m_{d(n)}$ is the number of "YES" answers of C_n on input x at the $d(n)^{th}$ 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 SAT) supposing that it is known what were the answers of queries in previous levels.

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

To show fact 3 just observe that the tuple $\langle k_1, k_2, \ldots, k_{d(n)}, o_k \rangle$ where k_1 is the number of "YES" answers of C_n on input x at the first level of queries, k_2 is the number of "YES" answers of C_n on input x at the second level of queries... and o_k is the value computed by C_n on input x, satisfies that $\langle x, k_1, k_2, \ldots, k_{d(n)}, o_k \rangle \in X$. Moreover, if there was $\langle x, i_1, i_2, \ldots, i_{d(n)}, o_i \rangle \in X$ such that $\langle x, i_1, i_2, \ldots, i_{d(n)}, o_i \rangle > \langle x, k_1, k_2, \ldots, k_{d(n)}, o_k \rangle$ it would imply that either

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

or otherwise $o_i > o_k$. In the first case, $i_1 = k_1, \ldots, i_{j-1} = k_{j-1}$ 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 $AC^{k-1}(NP) \subseteq P^{NP}[O(\log^k n)]$. Given an input x of length n to the circuit C_n we can determine if C_n accepts x with a Turing machine which proceeds as follows:

Using binary search, it determines

$$\langle m_1, m_2, \ldots, m_{d(n)}, o_m \rangle = \max \left\{ \langle i_1, i_2, \ldots, i_{d(n)}, o_i \rangle \;\middle|\; \langle x, i_1, i_2, \ldots, i_{d(n)}, o_i \rangle \in X \right\};$$

this can be done with at most $d(n)\log p(n)$ queries to X (observe that $d(n)\log p(n) \in O(\log^k n)$). Then it accepts x iff $o_m = 1$. This Turing machine recognizes the language accepted by the circuits.

Finally, knowing that $X \in NP$ by fact 1, we may conclude that

$$L(\{C_n\}, n \geq 1) \in P^{NP}[O(\log^k n)]$$

□

Similar sets, but simpler, were defined by Buss and Hay to show that, for reducibility to the NP-complete problem SAT, polynomial time truth-table reducibility via Boolean circuits is equivalent to logspace truth-table reducibility via Boolean formulas (Th. 2 [BH-88]).

4. The Extended Boolean hierarchy with superpolynomial bounding functions

Cai and Hemachandra introduced the Boolean hierarchy in [CH-86]. They considered the boolean functions:

$$\begin{aligned}
h_1(x_1) &= x_1 \\
h_{2k}(x_1, x_2, \ldots, x_{2k}) &= h_{2k-1}(x_1, x_2, \ldots, x_{2k-1}) \wedge \neg x_{2k} \\
h_{2k+1}(x_1, x_2, \ldots, x_{2k+1}) &= h_{2k}(x_1, x_2, \ldots, x_{2k}) \vee x_{2k+1}
\end{aligned}$$

for all k, $k \geq 1$. With these functions they define the classes of languages $NP(k)$:

$$\begin{aligned}
A \in NP(k) \iff & \exists\, B_1, \ldots, B_k \in NP \text{ such that} \\
& c_A(x) = h_k(c_{B_1}(x), \ldots, c_{B_k}(x)) \text{ for all } x.
\end{aligned}$$

Finally, they defined the Boolean hierarchy

$$BH = \bigcup_{k \geq 1} NP(k).$$

Köbler, Schöning, Wagner and Wechsung gave in their papers [KSW-87] and [WW-85] different definitions of the Boolean hierarchy. With the results presented in [WW-85] it could be proved that all three definitions are equivalent; they gave the following characterization for $NP(k)$:

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

From these results and those presented in [Be-87], it could be proved that the Boolean hierarchy coincides with the constant query classes:

$$BH = P_{\|}^{NP}[O(1)] = P^{NP}[O(1)].$$

The Extended Boolean hierarchy considers classes $NP(r)$ for non-constant bounding functions r. Naturally, classes $NP(r)$ are defined as

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

It is shown in [BH-88] and [Wa-88] that $NP\left(n^{O(1)}\right) = \Theta_2^p$. Also, in the second paper it is proved that

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

Wagner asked in his paper [Wa-88] what could be said about other superpolynomial bounding functions. We answer this question below, using ideas from the proof of Theorem 4.

Theorem 5 For all $k \geq 1$,

$$P^{NP}[O(\log^k n)] = NP\left(n^{O(\log^{k-1} n)}\right)$$

Proof ($\supseteq$) Given a set $A \in NP(b(n))$, where $b(n) \in n^{O(\log^{k-1} n)}$, we know by definition that there exists a set $B \in NP$ such that

$$c_B(x, i+1) \leq c_B(x, i) \quad \text{for all } i \leq b(|x|) \quad \text{and}$$
$$c_A(x) = \sum_{i=1}^{b(|x|)} c_B(x, i) \bmod 2.$$

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

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

Second, using the second property of B, M accepts x iff i is odd.

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

($\subseteq$) By the theorem 4, it is sufficient to prove that $AC^{k-1}(NP) \subseteq NP\left(n^{O(\log^{k-1} n)}\right)$.

Given $\{C_n\}$, $n \geq 1$ a family of circuits in $AC^{k-1}(NP)$ with depth $d(n) \in O(\log^{k-1} n)$ 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 4. Remember facts 1, 2 and 3 that this set verifies. From fact 3, it is easy to see that

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

Obviously,

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

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

$$L(\{C_n\}, n \geq 1) \in NP\left(n^{O(\log^{k-1} n)}\right).$$

□

5. $P^{NP}[O(\log^k n)]$ and circuits of small size

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

Proposition 6 For any bounding function $b(n) \leq 2^{n^{O(1)}}$, $NP(O(b(n)))$ is equal to the class of languages which are reducible to languages in NP via reductions of size $\log b(n) + O(1)$.

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

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

First, query if $\langle x, 10\ldots0\rangle$ is in B; call the answer to this query b_1. Next query if $\langle x, b_1 1\ldots0\rangle$ is in B. It is clear how to proceed. Note that exactly $\lceil \log b(n)\rceil$ gates are necessary.

($\supseteq$) Let A be reducible to SAT via circuits $\{C_n\}$, $n \geq 1$ of size $\log b(n) + O(1)$. Let us suppose that these circuits have depth $d(n)$ and at most $p(n)$ queries per level. Note that $d(n)$ and $p(n)$ can not be superpolynomial functions.

We define the set X as in the proof of Theorem 4 (but with the new bounds $d(n)$ and $p(n)$). Clearly, facts 1, 2 and 3 remain true. Now, as in the proof of Theorem 5, we have

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

where $i_j \leq p(n)$. Thus

$$m \leq 2p(n)^{d(n)} \leq b(n)$$

□

Now, from Theorem 5 and Proposition 6 we have a new characterization for $P^{NP}[O(\log^k n)]$.

Corollary 7 For all $k \geq 1$, $P^{NP}[O(\log^k n)]$ is equal to the class of languages which are reducible to languages in NP via circuits of size $O(\log^k n)$.

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

Conclusions

We have shown different characterizations for some complexity classes between Θ_2^p and Δ_2^p. In particular, we have determined what the classes of languages AC^k reducible to NP are and we have obtained circuit characterizations for the Extended Boolean hierarchy with superpolynomial bounding functions. However, all the circuit characterizations are based on AC^k circuits or on circuits of small size; we have not obtained characterizations based on NC^k circuits, but will continue our work in this direction.

It seems an interesting problem to study the NC^k complexity classes with oracle gates in NP and to show relationships between these classes and those present in this paper. Concretely, it would be nice to show that $NC^1(NP) = \Theta_2^p$: in this case the well known inclusions $AC^0 \subseteq NC^1 \subseteq L$ would collapse when the relativized versions to NP were considered, in contrast with the general relativized case (see [Wi-87]). On the other hand, if the equality $NC^1(NP) = \Theta_2^p$ fails we would find an irregular situation: while it holds that $AC^0 \subseteq NC^1 \subseteq L$ unrelativized, the first and the last relativized to NP counterparts would coincide, but the middle one would not.

Acknowledgements

We are very grateful to José L. Balcázar for reading earlier versions of this paper and for many interesting suggestions. We would like to thank Eric Allender, Birgit Jenner and Chris Wilson for helpful discussions which took place in Barcelona, during the Structure Conference 1990.

References

[AW-90] E. Allender, C. B. Wilson, *Width-Bounded Reducibility and Binary Search over Complexity Classes,* Proc. 5th IEEE Conference on Structure in Complexity Theory, (1990), pp. 122-129.

[Be-87] R. J. Beigel, *Bounded queries to SAT and the Boolean hierarchy,* To appear in TCS.

[BH-88] S. Buss, L. Hay, *On truth-table reducibility to SAT and the difference hierarchy over NP,* Proc. 3th IEEE Conference on Structure in Complexity Theory, (1988), pp. 224-233.

[CH-86] J. Cai, L. A. Hemachandra, *The Boolean hierarchy: hardware over NP,* Proc. 1st IEEE Conference on Structure in Complexity Theory, (1986), pp. 105-124.

[Co-85] S. Cook, *A taxonomy of problems with fast parallel algorithms,* Information and Control, (1985), vol. 64, pp. 2-22.

[KSW-87] J. Köbler, U. Schöning, K. W. Wagner, *The difference and the truth-table hierarchies for NP,* R.A.I.R.O. 21(1987), pp. 419-435

[Wa-88] K. W. Wagner, *Bounded query computations,* Proc. 3th IEEE Conference on Structure in Complexity Theory, (1988), pp. 260-277.

[WW-85] G. Wechsung, K. W. Wagner, *On the Boolean closure of NP*, manuscript 1985 (extended abstract as: Wechsung G., On the Boolean closure of *NP*, Proc. Conf. Fundam. Comp. Theory, Cottbus 1985, LNCS 199(1985), 485-493.

[Wi-87] C. B. Wilson, *Relativized NC*, Math. Systems Theory, (1987) pp. 13-29, vol. 20.

[Wi-90] C. B. Wilson *Decomposing NC and AC*, SIAM J. Comput. (1990) pp. 384-396 vol. 19, 2.