The $P_{\log^i}$ and AC^{i-1} operators on the polynomial time hierarchy

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

In [CS-92] we studied the agreement of operators $P_{\log^i}$ and AC^{i-1} acting on NP. In this article we extend this work to other classes of the polynomial time hierarchy. We show that on Σ_k^P, Π_k^P, Δ_k^P and Θ_k^P-classes both operators have the same behaviour, but this coincidence does not seem to be true on other classes included in the PH hierarchy: we give a set A such that, relativized to A, $P_{\log^i}(P_{\log^j}(NP))$ is different from $AC^{i-1}(P_{\log^j}(NP))$. As a result of these characterizations we show $P_{\log}(\Theta_k^P) = \Theta_k^P$, an equality that is useful to show lowness properties. In fact, we get easily the Θ-lowness results given by Long and Sheu in [LS-91]. Besides, we clarify the situation of the classes in $L_2^{P,\Delta}$ for which their membership to $L_2^{P,\Theta}$ was not clear.

1. Introduction

In this article we study the relationships of two basic mechanisms acting on different classes included in the polynomial time hierarchy. Concretely, we consider polynomial time Turing machines that make a polylogarithmic number of oracle queries (the $P_{\log^i}$ operator), in contrast with circuits, with oracle gates, of polynomial size and polylogarithmic depth (the AC^{i-1} operator). The motivation comes from a first study of this situation [CS-92], where it is shown the agreement of the operators $P_{\log^i}$ and AC^{i-1} on NP.

In section 3 we show that the two operators have the same effect when the oracles considered are the Σ_k^P, Π_k^P, Δ_k^P and Θ_k^P-classes. However, we find differences, in a relativized world, when we apply the operators on classes that are between Θ_2^P and Δ_2^P, concretely on the $P_{\log^j}(NP)$-classes.

In section 4 we develop some applications of results obtained in section 3 to the low hierarchy. The low and high hierarchies in NP were defined by Schöning [Sc-83], which provided a formal framework for analyzing the internal structure of NP. Later, in [Sc-86], he also introduced a refinement of the low and high hierarchies based on the Δ-levels of the polynomial time hierarchy. Schöning [Sc-86], Ko and Schöning [KS-85], Balcázar and Book [BB-86], Balcázar and Schöning [BS-92] and Allender and Hemachandra [AH-92] located several classes of sets in the low hierarchy. Long and Sheu [LS-91] introduced a new refinement of the low and high hierarchies based on the Θ-levels of the polynomial time hierarchy, and they sharpened most of the known lowness results. However they did not know how to locate 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 in the new refinement. Applying results of section 3 we get easily the results about Θ-lowness given in [LS-91] and we also clarify the situation of the above classes.

2. Definitions and notation

Σ 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 Turing machine M using oracle set A. The classes P and NP have their standard definitions. $P(A)$ and $NP(A)$ are, respectively, their A-relativized counterparts.

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

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

Definition 2.1. For any class C and for all $i \geq 1$, $P_{\log^i}(\mathrm{C})$ is the class of languages accepted by deterministic polynomial time Turing machines that make at most $O(\log^i n)$ queries to an oracle set in C on inputs of length n,

$$L \in P_{\log^i}(\mathrm{C}) \Longleftrightarrow \exists B \in \mathrm{C} : L \leq_{T[O(\log^i n)]}^P B.$$

$\mathrm{E}_T(TALLY)$ ($\mathrm{E}_{tt}(TALLY)$) denotes the class of sets Turing equivalent (truth-table equivalent) to a tally set.

We assume that the reader is familiar with the habitual concepts in relation to classes of languages recognized by Boolean circuits [Wi-87] [Wi-90]. Concretely, in section 3 we work with the classes AC^i and NC^i, which are defined as follows:

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

where here "uniform" means logspace uniform. Besides, we work with relativized circuits. In these circuits oracle gates are allowed which can determine the membership of a string to an oracle set. In an AC circuit the size and depth of these gates are 1; in a NC circuit an oracle gate which has k input bits is defined to have size 1 and depth $\lceil \log k \rceil$ (both are standard conventions, see [Wi-87]). For any class C, $AC^i(\mathrm{C})$ ($NC^i(\mathrm{C})$) denotes 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 an oracle set in C.

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

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

The Θ-levels of the polynomial time hierarchy have been studied by Wagner [Wa-90] and he obtained several characterizations of that levels.

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.

The low hierarchy within NP relative to the Σ_k^P and Δ_k^P-classes of the polynomial time hierarchy was introduced by Schöning in [Sc-83] and [Sc-86]. Taking into account that for any set $A \in NP$ and $k \geq 1$ it holds that

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

and as a consequence

$$\Delta_k^P \subseteq \Delta_k^P(A) \subseteq \Delta_{k+1}^P,$$

he defined the low and the high hierarchies within NP, relative to the Σ_k^P and Δ_k^P-classes, in a natural way. In particular, he defined the low hierarchy as follows:

Definition 2.2. Relative to the Σ_k^P-classes of the polynomial time hierarchy, the Σ-classes of the low hierarchy are

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

Definition 2.3. Relative to the Δ_k^P-classes of the polynomial time hierarchy, the Δ-classes of the low hierarchy are

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

In [LS-91] it is introduced a refinement of these hierarchies based on the Θ_k^P-classes of the polynomial time hierarchy. The authors noted that for every set $A \in NP$ and $k \geq 2$,

$$\Theta_k^P \subseteq \Theta_k^P(A) \subseteq \Theta_{k+1}^P,$$

and therefore it makes sense to define low and high hierarchies based on Θ-levels. In particular, they defined

Definition 2.4. Relative to the Θ_k^P-classes of the polynomial time hierarchy, the Θ-classes of the low hierarchy are

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

Starting from previous results showed in [Sc-83] they proved some basic relationships among the Σ, Δ and Θ-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 Θ-lowness results for several classes located in the Δ-levels of the low hierarchy. In section 4 we will use a property of the $P_{\log}$ operator showed in section 3 that will enable us to prove easily their Θ-lowness results.

3. The $P_{\log^i}$ and AC^{i-1} operators on the the polynomial time hierarchy

In this section we study the effect of applying operators $P_{\log^i}$ and AC^{i-1} on the polynomial time hierarchy. We show that they define exactly the same classes when they operate on Σ_k^P, Π_k^P, Δ_k^P and Θ_k^P-classes. However, it seems that this agreement will not be true on other classes included in the previous ones: we give an oracle A such that the action of operators $P_{\log^i}$ and AC^{i-1} on $P_{\log^j}(NP(A))$ is different.

Extending previous results given in [CS-92], we get two useful theorems. The first one relates operators $P_{\log^i}$ and AC^{i-1} when acting on a general class. The second one states the equivalence of both operators when acting on Σ_k^P-classes. For the sake of completeness, the proofs of these theorems are given in the appendix.

Theorem 3.1. Let $\mathrm{C} \neq \emptyset$ be a class of languages closed by $\leq_m^P$-reducibility; then for all $i \geq 1$, $P_{\log^i}(\mathrm{C}) \subseteq AC^{i-1}(\mathrm{C})$.

Theorem 3.2. For all $i \geq 1$ and $k \geq 0$, $P_{\log^i}(\Sigma_k^P) = AC^{i-1}(\Sigma_k^P)$.

Note that in fact we can proof that for all $i \geq 1$ and for any class C, $P_{\log^i}(NP(C)) = AC^{i-1}(NP(C))$ then as a consequence get the theorem 3.2.

When we consider the Π_k^P, Δ_k^P and Θ_{k+1}^P-classes the agreement of the operators $P_{\log^i}$ and AC^{i-1} still remains true. While this was to be expected for Π_k^P and Δ_k^P, surprisingly, the classes obtained on the Θ_{k+1}^P-classes are just the same classes we get when the operators act on Σ_k^P or Π_k^P-classes.

Theorem 3.3. For all $i \geq 1$ and $k \geq 0$,

a) $P_{\log^i}(\Pi_k^P) = AC^{i-1}(\Pi_k^P) = P_{\log^i}(\Sigma_k^P)$.

b) $P_{\log^i}(\Delta_k^P) = AC^{i-1}(\Delta_k^P) = \Delta_k^P$.

c) $P_{\log^i}(\Theta_{k+1}^P) = AC^{i-1}(\Theta_{k+1}^P) = P_{\log^i}(\Sigma_k^P)$.

Proof. a) From theorem 3.2 follows the equivalence on Π_k^P-classes. Note that for any set A, it is $P_{\log^i}(A) = P_{\log^i}(A^c)$ and $AC^{i-1}(A) = AC^{i-1}(A^c)$.

b) Follows directly from the definition of Δ_k^P and the observation that operator P absorbs operators $P_{\log^i}$ and AC^{i-1}: both operators are weaker than operator P, and for any class C, $P(P(\mathrm{C})) = P(\mathrm{C})$.

c) First of all, we show $AC^{i-1}(\Theta_{k+1}^P) = P_{\log^i}(\Sigma_k^P)$. Rewriting Θ_{k+1}^P as $AC^0(\Sigma_k^P)$ (theorem 3.2), we have $AC^{i-1}(\Theta_{k+1}^P) = AC^{i-1}(AC^0(\Sigma_k^P))$. Note that for any class C, $AC^j(AC^0(\mathrm{C})) = AC^j(\mathrm{C})$: replacing oracle gates by the corresponding $AC^0(\mathrm{C})$ circuits we get an $AC^j(\mathrm{C})$ circuit (see [Wi-90] for formal proofs of this kind of relationships). Then,

$$AC^{i-1}(\Theta_{k+1}^P) = AC^{i-1}(\Sigma_k^P)$$

and the characterization follows from theorem 3.2.

Now, the equality

$$P_{\log^i}(\Theta_{k+1}^P) = P_{\log^i}(\Sigma_k^P)$$

comes from the inclusion $P_{\log^i}(\Theta_{k+1}^P) \subseteq AC^{i-1}(\Theta_{k+1}^P)$ (theorem 3.1), the characterization we have just seen, and the obvious relation $P_{\log^i}(\Sigma_k^P) \subseteq P_{\log^i}(\Theta_{k+1}^P)$. □

In particular, for $i = 1$ we have the following equality that we will use in the next section.

Corollary 3.4. For all $k \geq 1$, $P_{\log}(\Theta_k^P) = \Theta_k^P$.

Now we study whether this agreement is true for other classes which are included in the above ones. In particular we study the behavior of the operators when they act on classes that are between the Θ_{k+1}^P and Δ_{k+1}^P-classes, concretely we consider the $P_{\log^j}(\Sigma_k^P)$-classes for any $j \geq 1$.

First of all, we have the following relation

Proposition 3.5. For all $i \geq 1$ and $j \geq 1$, $P_{\log^i}(P_{\log^j}(\Sigma_k^P)) \subseteq P_{\log^{i+j-1}}(\Sigma_k^P)$.

Proof. Classes $P_{\log^j}(\Sigma_k^P)$ are closed by $\leq_m^P$-reducibility, so that, applying the $P_{\log^i}$ operator, we get $P_{\log^i}(P_{\log^j}(\Sigma_k^P)) \subseteq AC^{i-1}(P_{\log^j}(\Sigma_k^P)) = AC^{i-1}(AC^{j-1}(\Sigma_k^P))$. Considering a result of Wilson [Wi-90] that proves $AC^a(AC^b) = AC^{a+b}$ it can be shown that for any class C, it is $AC^a(AC^b(\mathrm{C})) = AC^{a+b}(\mathrm{C})$. Now, the result follows from $AC^{i-1}(AC^{j-1}(\Sigma_k^P)) = AC^{i+j-2}(\Sigma_k^P) = P_{\log^{i+j-1}}(\Sigma_k^P)$. □

As a consequence of the proof, if operators $P_{\log^i}$ and AC^{i-1} had the same behavior on $P_{\log^j}(\Sigma_k^P)$-classes the inclusion would be actually an equality. However, the equality does not seem to be true. Below, we show that there is a set A such that, relative to A, classes $P_{\log^i}(P_{\log^j}(NP))$ and $P_{\log^{i+j-1}}(NP)$ are different when i and j are greater than 1. In fact we construct an oracle set such that, if $\max(i,j) \neq \max(i',j')$ then, relative to this oracle the classes $P_{\log^i}(P_{\log^j}(NP))$ and $P_{\log^{i'}}(P_{\log^{j'}}(NP))$ are different.

Given a set A and for all $k \geq 1$ we define the language

$$L_k(A) = \left\{ 0^n \;\middle|\; A^{=\log^k n} = \emptyset \text{ or the minimum word of length } \log^k n \text{ in } A \text{ is even } \right\}.$$

It is not difficult to see that for any A, $L_k(A) \in P_{\log^k}(NP(A))$. Moreover, we can give an specific A such that when $\max(i,j) < k$ then $L_k(A)$ does not belong to $P_{\log^i}(P_{\log^j}(NP(A)))$. Concretely, we can write the following:

Theorem 3.6. There exists a recursive set A such that for all k, i, j with $\max(i,j) < k$, the language $L_k(A)$ does not belong to $P_{\log^i}(P_{\log^j}(NP(A)))$.

Proof. Let us consider $M_1, M_2, \ldots$ and $N_1, N_2, \ldots$ enumerations of the deterministic and non-deterministic 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:

$$K(B) = \left\{ \; \langle u, y, 0^t \rangle \;\middle|\; N_u \text{ with oracle } B \text{ accepts } y \text{ in at most } t \text{ steps } \right\}$$

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

A is constructed in stages. At stage $s = \langle a, b, k, i, j, c, d \rangle$, if $\max(i,j) < k$ and the machine M_a (resp. M_b) on inputs of length n_s (resp. of length $\leq p_a(n_s)$) makes at most $c \log^i n_s$ ($d \log^j p_a(n_s)$) queries, we will add words of length $\log^k n_s$ to A diagonalizing away from the language recognized by machine M_a with oracle $L(M_b, K(A))$.

Let us suppose that k, i, j and the machines M_a and M_b satisfy the preceding restriction (otherwise, we finish stage s). Let Q be the chain of queries $q_1, q_2, \ldots, q_{f(n_s)}$ (where $f(n_s) \leq c \log^i n_s$) that machine M_a makes on input 0^{n_s} and oracle $L(M_b, K(A))$. Without loss of generality we can suppose each query q_l depends on all the answers of the previous queries $q_1, \ldots, q_{l-1}$. We denote by $q_{l,1}, q_{l,2}, \ldots q_{l,g(n_s)}$ (where $g(n_s) \leq d \log^j p_a(n_s)$) the list of queries to $K(A)$ that machine M_b makes on input q_l, $1 \leq l \leq f(n_s)$.

We say that machine M_a behaves correctly with input 0^{n_s} and oracle $L(M_b, K(A))$ if it accepts (rejects) and the minimum word of length $\log^k n_s$ in A is even (odd). If so, we add a new odd (even) word w to A of length $\log^k n_s$ and smaller than the existing ones in such a way it does not affect the positive answers of the queries $q_{l,m}$ to $K(A)$. This property can be reached keeping a list A' of words that are not valid choices for w: for every query $q_{l,m} = \langle u_{l,m}, y_{l,m}, 0^{t_{l,m}} \rangle$ of M_b to $K(A)$ answered positively, we include in A' the list of words that are not in A and are queried in a certain accepting path of $N_{u_{l,m}}$ on input $y_{l,m}$.

Machine M_a, with the new oracle $L(M_b, K(A \cup \{w\}))$, either behaves incorrectly, or a query $q_{l,m}$ of M_b to $K(A)$ that was answered negatively now is answered affirmatively. If the second possibility happens, either M_a makes a new chain of queries $Q' = q'_1, q'_2, \ldots, q'_{f(n_s)}$ to $L(M_b, K(A \cup \{w\}))$ with $Q' \neq Q$ or there is a change in the list of queries generated by the last query $q_{f(n_s)}$ of M_a. Now, it can be established the following facts:

Fact 1. After $2^{g(n_s)}$ steps (adding words to A) we can affirm that we have got a chain Q' different from the initial one (or M_a works incorrectly): (proof sketch) Note that, after $2^{g(n_s)}$ consecutive changes in the list of queries generated by $q_{f(n_s)}$, all the queries of the list must have answer yes. At that moment the list generated by $q_{f(n_s)}$ does not admit changes anymore.

Fact 2. If a chain Q is reached $f(n_s)2^{g(n_s)}$ times then it can not be reached anymore (or M_a works incorrectly): (proof sketch) Note that, if a chain Q is reached $f(n_s)2^{g(n_s)}$ times then, all its secondary queries $q_{l,m}$ must have yes answers.

Fact 3. There exist at most $2^{f(n_s)}$ chains of queries Q.

Taking into account these three facts we can force machine M_a to work incorrectly before $2^{g(n_s)} \times f(n_s)2^{g(n_s)} \times 2^{f(n_s)}$ word additions to A. Let us call this number $h(n_s)$; note that $h(n_s)$ is $2^{O(\log^{\max(i,j)} n_s)}$. Before doing a word addition we have at most $g(n_s) \times f(n_s) \times p_b(p_a(n_s))$ reserved words in A': each query to $K(A)$ answered positively needs at most $p_b(p_a(n_s))$ word reserves. Then, choosing n_s the smallest integer such that:

1. $\log^k n_s$ is larger than anything queried or added to A at any previous stage,
2. $2(h(n_s) + g(n_s) \times f(n_s) \times p_b(p_a(n_s))) < 2^{\log^k n_s}$;

we have enough room to make all the process of word additions. □

We wonder what kind of classes $P_{\log^i}(P_{\log^j}(NP))$ are, for i and j greater than 1. Concretely, we want to know whether $P_{\log^i}(P_{\log^j}(NP))$ is equal to $P_{\log^k}(NP)$ for some k. Last theorem suggests that if for some k the equality is true, then k has to be $\max(i,j)$ unless there exists a proof for the equality which does not relativize.

We have not got any proof for

$$P_{\log^i}(P_{\log^j}(NP)) = P_{\log^{\max(i,j)}}(NP).$$

Moreover, we suspect that this equality can be false. In outline, this feeling is based on the characterization of these classes as binary search classes: In [CS-92] we showed that a set A is in $P_{\log^k}(NP)$ if there exists a set $B \in NP$ with the following properties

1. $x \in A \iff Q(x,k,B)$,
2. $\langle x,n\rangle \in B \Longrightarrow \langle x,n-1\rangle \in B$,

where

$$Q(x,k,B) \equiv \max\left\{n \leq 2^{O(\log^k |x|)} \mid \langle x,n\rangle \in B\right\} \text{ is odd.}$$

It is not hard to give an analogous characterization for $P_{\log^i}(P_{\log^j}(NP))$. The set A belongs to this class if there exists a set $B \in NP$ with the following properties

1. $x \in A \iff \max\left\{m \leq 2^{O(\log^i |x|)} \mid Q\left(\langle x,m\rangle, j, B\right)\right\}$ is odd,
2. $Q\left(\langle x,m\rangle, j, B\right) \Longrightarrow Q\left(\langle x,m-1\rangle, j, B\right)$,
3. $\langle\langle x,m\rangle, n\rangle \in B \Longrightarrow \langle\langle x,m\rangle, n-1\rangle \in B$.

Roughly speaking, the first characterization describes $P_{\log^k}(NP)$ as a single binary search class, and the second one characterizes $P_{\log^i}(P_{\log^j}(NP))$ as a double binary search class. From these descriptions seems that $P_{\log^i}(P_{\log^j}(NP))$ is a more powerful class than $P_{\log^{\max(i,j)}}(NP)$. This discussion suggests that $P_{\log^i}(P_{\log^j}(NP))$ can be different from all $P_{\log^k}(NP)$-classes.

We finish this section with a few words about the action of operator NC^i on the polynomial time hierarchy. In a recent paper [Og-93] Ogiwara showed $AC^{i-1}(NP) = NC^i(NP)$ for all $i \geq 1$. Using Ogiwara's arguments that result can be extended to Σ_k^P-classes and as a consequence, to

Π_k^P and Δ_k^P-classes. On Θ_{k+1}^P-classes the agreement of both operators still remains true: note that $\Theta_{k+1}^P = AC^0(\Sigma_k^P) = NC^1(\Sigma_k^P)$, now $AC^{i-1}(AC^0(\Sigma_k^P)) = AC^{i-1}(\Sigma_k^P) = NC^i(\Sigma_k^P) = NC^i(NC^1(\Sigma_k^P))$. The last equality can be obtained starting from a result of Wilson [Wi-90] that shows $NC^a(NC^b) = NC^{a+b-1}$, for all $a \geq 1$ and $b \geq 1$.

In contrast with the situation between AC^{i-1} and $P_{\log^i}$, operators AC^{i-1} and NC^i on $P_{\log^j}(\Sigma_k^P)$ define exactly the same classes for all $j \geq 1$: Using Wilson's results and rewriting $P_{\log^j}(\Sigma_k^P)$ as $AC^{j-1}(\Sigma_k^P)$, we have $AC^{i-1}(AC^{j-1}(\Sigma_k^P)) = AC^{i+j-2}(\Sigma_k^P) = NC^{i+j-1}(\Sigma_k^P) = NC^i(NC^j(\Sigma_k^P)) = NC^i(AC^{j-1}(\Sigma_k^P))$.

4. Applications

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

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

$L_3^{P,\Delta}$: co-sparse NP sets, $NP \cap P$-close, NP sets that are $\leq_{1-tt}^P$-reducible to some sparse set, and NP sets that are $\leq_T^P$-equivalent or $\leq_{1-tt}^P$-equivalent to some sparse set.

Note that for these classes sparse sets play an important role. The basic proof technique for establishing the Δ-lowness of these sets comes from a result of Mahaney [Ma-82] generalized by Book and Tang.

Theorem 4.1. [BT-89] For $k \geq 1$, if S is a sparse set in Σ_k^P, then $\Sigma_k^P(S) \subseteq \Delta_{k+1}^P$.

Now, for example, the Δ-lowness of sparse NP sets can be shown as follows. Starting from $\Sigma_1^P(S) \subseteq \Delta_2^P$ apply the P-operator to both sides of the containment to get

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

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

Later, Kadin improved Mahaney's result in [Ka-87]. He showed that if S is a NP sparse set, then $NP(S) \subseteq \Theta_2^P$. As it is noted in [LS-91], this result suggests that sparse NP sets may actually belong to $L_2^{P,\Theta}$.

It is not difficult to extend Kadin's result showing that for $k \geq 1$

$$\text{if } S \text{ is a sparse set in } \Sigma_k^P, \text{ then } \Sigma_k^P(S) \subseteq \Theta_{k+1}^P.$$

In fact, Long and Sheu showed in their paper [LS-91] that $\Theta_{k+1}^P(S) \subseteq \Theta_{k+1}^P$ and, as a consequence, they obtained Θ-lowness results for several classes. However, their way to show Θ-lowness was different from the way to prove Δ-lowness. Concretely, starting from Kadin's theorem $\Sigma_1^P(S) \subseteq \Theta_2^P$ and applying the operator $P_{\log}$ to both sides of the containment they obtained

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

but from this expression they could not deduce Θ-lowness for S because it was not known whether $P_{\log}(\Theta_2^P) = \Theta_2^P$. But now this equality is just the corollary 3.4 showed in section 3 and as a consequence we can get easily Long and Sheu's results of Θ-lowness.

Using the corollary 3.4 it can be proved that all the classes given before in $L_3^{P,\Delta}$ are actually in $L_3^{P,\Theta}$, as well that for almost all ones given in $L_2^{P,\Delta}$ actually belong to $L_2^{P,\Theta}$. The reader can

see in [LS-91] formal demonstrations for these statements. However, Long and Sheu left open what happens, in relation to Θ-lowness properties, with the classes $NP \cap \mathrm{E}_T(TALLY)$ and NP sets that are standard or general left cuts for some real number x, both in $L_2^{P,\Delta}$. Concretely, for $NP \cap \mathrm{E}_T(TALLY)$, they established the following: if $A \in NP \cap \mathrm{E}_T(TALLY)$ and there is a tally set $T \in NP$ 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 corollary 3.4 we can solve this problem. First we show

Proposition 4.2. Let A be a set such that $A \leq_T^P T$ where T is a tally set in the class Θ_2^P, then $NP(A) \subseteq \Theta_2^P$.

Proof. Let N be a nondeterministic polynomial time Turing machine with time bounded by $q(n)$ on inputs of length n. We will show that $L(N^A) \in AC^0(NP)$, 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 Turing 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 $\{1, 1^2, ..., 1^{p(m)}\}$.

Let $t(n)$ be the string of length $p(q(n))$ that represents the answers to the queries $1 \in T?, 1^2 \in T?, \ldots, 1^{p(q(n))} \in T?$ 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 $AC^0(NP)$ circuit that recognizes $L(N^A)$. On inputs of length n the circuit has a first level of query gates $1 \in T?, 1^2 \in T?, \ldots, 1^{p(q(n))} \in T?$; note that each one of them can be replaced by an $AC^0(NP)$ circuit because T is in Θ_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 a $AC^0(NP)$ circuit that accepts $L(N^A)$. □

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

Corollary 4.3. Let A be a set such that $A \leq_T^P T$ where T is a tally set in the class Θ_2^P, then $\Theta_2^P(A) = \Theta_2^P$.

As a consequence of corollary 4.3 we can answer affirmatively the question before. Therefore, the situation for the class $NP \cap \mathrm{E}_T(TALLY)$ is as follows: if A is in 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 NP and $A \equiv_T^P T$, T has to belong to Δ_2^P. For the class of NP sets that are general or standard left cuts of some real number x the situation is analogous because these sets are in $\mathrm{E}_T(TALLY)$. Note that, using the same techniques it can be shown that $NP \cap \mathrm{E}_{tt}(TALLY)$ is included in $L_2^{P,\Theta}$.

Now, we can wonder whether for every set A in NP such that A is Turing equivalent to a tally set then, necessarily A is equivalent to some tally in Θ_2^P. If the answer is affirmative then $NP \cap \mathrm{E}_T(TALLY) \subseteq L_2^{P,\Theta}$. Unfortunately, we do not know the answer.

Acknowledgements

We are grateful to José L. Balcázar for his helpful and encouragement. Ming-Jye Sheu deserve our thanks for interesting suggestions and comments.

References

[AH-92] E. Allender and L. Hemachandra, *Lower bounds for the low hierarchy,* J. ACM, 39(1), (1991), pp. 234-251.

[BB-86] J. L. Balcázar and R. Book, *Sets with small generalized Kolmogorov complexity,* Acta Informatica, 23, (1986), pp. 679-688.

[BS-92] J. L. Balcázar and U. Schöning, *Logarithmic advice classes,* Theoretical Computer Science, 99, (1992), pp. 279-290.

[BT-89] R. Book and S. Tang, *A note on sparse sets and the polynomial-time hierarchy,* Information Processing Letters, 33(3), (1989), pp. 141-143.

[CS-92] J. Castro and C. Seara, *Characterizations of some complexity classes between Θ_2^P and Δ_2^P,* STACS 92, LNCS 577, Springer-Verlag, (1992), pp. 305-317.

[Ka-87] J. Kadin, *$P^{NP[\log n]}$ and sparse Turing-complete sets for NP,* Proc. Struc. in Complexity Theory, (1987), pp. 33-40.

[KS-85] K. Ko and U. Schöning, *On circuit-size and the low hierarchy in NP,* SIAM J. Comput. 14(1), (1985), pp. 41-51.

[LS-91] T. Long and M-J. Sheu, *A refinement of the low and high hierarchies,* Technical report OSU-CISRC-2/91-TR6, The Ohio State University, (1991).

[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-93] M. Ogiwara, *$NC^k(NP) = AC^{k-1}(NP)$,* STACS 94, LNCS 775, Springer-Verlag, (1994), pp. 313-323.

[Sc-83] U. Schöning, *A low and a high hierarchy within NP,* J. Comput. System Sci., 27, (1983), pp. 14-28.

[Sc-86] U. Schöning, *Complexity and Structure,* LNCS, vol. 211, Springer-Verlag, (1986).

[Wa-90] K. W. Wagner, *Bounded query classes,* SIAM J. Comp., vol. 19, 5 (1990), pp. 833-846.

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

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

Appendix

Theorem 3.1. Let C $\neq \emptyset$ be a class of languages closed by $\leq_m^P$-reducibility; then for all $i \geq 1$, $P_{\log^i}(\mathrm{C}) \subseteq AC^{i-1}(\mathrm{C})$.

Proof. We proceed by induction on i:

The case $i = 1$ follows the proof in [CS-92] that shows $P_{\log}(NP) \subseteq AC^0(NP)$. Let M^A be a fixed polynomial time Turing machine that makes at most $c \log n$ queries to an oracle A in C on inputs of length n. We define the languages:

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

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

It is easy to see that $L \leq_m^P A$ and $L' \in P$; therefore languages L and L' belong to C. We shall use these languages as oracles to simulate the machine M^A by a family of circuits. The Figure 1 shows the member of this family which operates on inputs of length n.

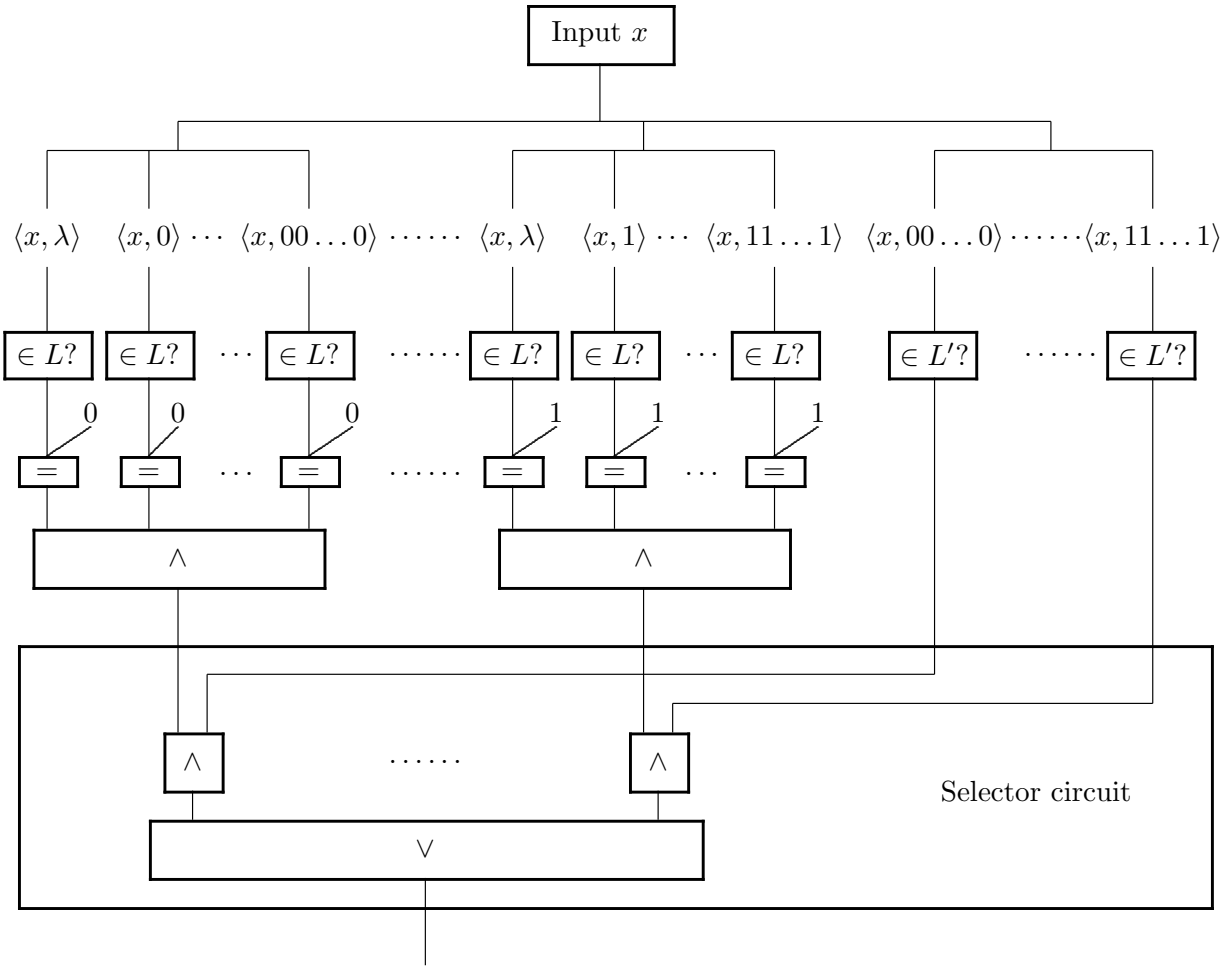

Figure 1.

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^A 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^A on x, it chooses the suitable a with the help of a selector circuit.

Now, we suppose that the result is true up to $i-1$ by induction hypothesis.

Let M^A be a fixed polynomial time Turing machine that makes at most $c \log^i n$ queries to A in C on inputs of length n, and let us consider the following languages:

$$
\begin{aligned}
L_{1,n} &= \{y \mid M^A \text{ starting at the configuration } y \text{ reaches after } c\log^{i-1} n \\
&\qquad \text{queries a configuration which has a 1 as the first bit } \} \\
L_{2,n} &= \{y \mid M^A \text{ starting at the configuration } y \text{ reaches after } c\log^{i-1} n \\
&\qquad \text{queries a configuration which has a 1 as the second bit}\} \\
&\ldots \\
L_{p(n),n} &= \{y \mid M^A \text{ starting at the configuration } y \text{ reaches after } c\log^{i-1} n \\
&\qquad \text{queries a configuration which has a 1 as the } p(n)^{th} \text{ bit}\}
\end{aligned}
$$

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^A does not reach them on any input. It is clear that all these languages are in $P_{\log^{i-1}}(A)$ and then, by induction hypothesis, all of them have $AC^{i-2}(\mathrm{C})$ circuits. Now, let C_n be the circuit that on input x of length n computes in $\log n$ levels the configuration f of M^A on x after all the queries have been done. All the levels are equal and they are composed by $AC^{i-2}(\mathrm{C})$ circuits for $L_{1,n}, L_{2,n}, \ldots, L_{p(n),n}$ placed in parallel given, at level j, the configuration of M^A on x after $jc\log^{i-1} n$ queries. The circuit C_n ends with a small circuit $O \in AC^0(P)$ which computes whether $x \in L\left(M^A\right)$ knowing what is the configuration f.

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

Theorem 3.2. For all $i \geq 1$ and $k \geq 0$, $P_{\log^i}(\Sigma_k^P) = AC^{i-1}(\Sigma_k^P)$.

Proof. We follow ideas shown in [CS-92]. Note that, by theorem 3.1, we only have to prove that $AC^{i-1}(\Sigma_k^P) \subseteq P_{\log^i}(\Sigma_k^P)$. 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 $AC^{i-1}(\Sigma_k^P)$ 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 $\langle x, i_1, i_2, \ldots, i_{d(n)}, o\rangle$ 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 satisfying 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. $X \in \Sigma_k^P$.
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
   $\langle m_1, m_2, \ldots, m_{d(n)}, o_m \rangle = \max \left\{ \langle i_1, i_2, \ldots, i_{d(n)}, o_i \rangle \;\middle|\; \langle x, i_1, i_2, \ldots, i_{d(n)}, o_i \rangle \in X \right\}$.
   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 consider the characterization of Σ_k^P in terms of alternating bounded quantifiers and 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 Σ_k^P) 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 $AC^{i-1}(\Sigma_k^P) \subseteq P_{\log^i}(\Sigma_k^P)$. 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 $d(n) \log p(n) + 1$ queries to X (observe that $d(n) \log p(n) \in O(\log^i n)$). Then it accepts x iff $o_m = 1$. This Turing machine recognizes the language accepted by the circuits.

Finally, knowing that $X \in \Sigma_k^P$ by fact 1, we may conclude that $L(\{C_n\}, n \geq 1) \in P_{\log^i}(\Sigma_k^P)$.

□