

**The Θ -operator
and the low hierarchy**

**Jorge Castro
Carlos Seara**

Report LSI-92-16-R

The Θ -operator and the low hierarchy.

(July, 1992)

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

Long and Sheu in their paper [LS-91] introduced a refinement of the low hierarchy based on the Θ -levels of the polynomial time hierarchy which gives a deeper sight of the internal structure of NP . In this paper we show a surprising property of the Θ -operator and as a consequence, we get easily the Θ -lowness results given 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

Schöning [Sc-83] defined the low and high hierarchies in NP , 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.

In this paper we show a surprising property of the Θ -operator and we study the consequences this property has. In particular, we get easily the results about Θ -lowness 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 $L(M^A)$ 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:

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

We denote by $E_T(TALLY)$ the class of sets Turing 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. Concretely, in section 3 we work with the class AC^0 , which is defined as follows:

$$L \in AC^0 \iff \exists \{C_n\}, n \geq 1, \text{ uniform unbounded fan-in circuits} \\ \text{with size } O(n^{O(1)}) \text{ and depth } O(1) \text{ which recognizes } L,$$

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. The size and depth of these gates are 1. We use the notation $AC^0(A)$ for the class of languages recognized by a family of unbounded fan-in circuits with polynomial size and $O(1)$ depth using A as oracle.

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:

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

where for any class C , $\Theta(C)$ denotes the class of languages recognized by polynomial time Turing machines that make at most $O(\log n)$ queries on inputs of length n to an oracle set in C . 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 .

Low hierarchy within NP relative to the Σ and Δ -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 and Δ -classes, in a natural way. In particular, he defined the low hierarchy as follows:

Definition 2.1 Relative to the Σ -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 2.2 Relative to the Δ -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 -levels 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 2.3 Relative to the Θ -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 the following section we shall show a property of the Θ -operator that will enable us to prove their Θ -lowness results easily.

3. A property of the Θ -operator and some consequences

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, $NP \cap (\text{co-}NP/\log)$, $NP \cap APT$, 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_{1-tt}^P -equivalent or that are \leq_T^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 3.1 [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, $S \in L_2^{P,\Delta}$.

Kadin improved Mahaney's result later in his paper [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 generalize 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 Θ to both sides of the containment they obtained

$$\Theta_2^P(S) = \Theta(\Sigma_1^P(S)) \subseteq \Theta(\Theta_2^P),$$

but from this expression they could not deduce Θ -lowness for S because it was not known whether $\Theta(\Theta_2^P) = \Theta_2^P$.

In this section we show the surprising equality $\Theta(\Theta_k^P) = \Theta_k^P$. As a consequence we can get easily Long and Sheu's results of Θ -lowness and we can clarify the situation for some classes in $L_2^{P,\Delta}$ where their membership to $L_2^{P,\Theta}$ was not clear.

First, taking advantage of previous results showed in [CS-92], we get the following useful propositions that are proved in the appendix.

Proposition 3.2 Let $C \neq \emptyset$ be a class of languages closed by \leq_m^P -reducibility; then $\Theta(C) \subseteq AC^0(C)$.

Proposition 3.3 For all $k \geq 1$, $\Theta_{k+1}^P = AC^0(\Sigma_k)$.

The last proposition states that, on Σ_k^P -classes, Θ and AC^0 -operators define exactly the same classes. Now, we can establish the main result.

Theorem 3.4 For $k \geq 2$, it holds $\Theta(\Theta_k^P) = \Theta_k^P$.

Proof Note that Θ_k^P is closed by \leq_m^P -reducibility. Then, applying proposition 3.2, we have

$$\Theta(\Theta_k^P) \subseteq AC^0(\Theta_k^P).$$

Rewriting Θ_k^P as $AC^0(\Sigma_{k-1}^P)$ (proposition 3.3), we get

$$\Theta(\Theta_k^P) \subseteq AC^0(AC^0(\Sigma_{k-1}^P)).$$

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. Therefore, we can write

$$\Theta(\Theta_k^P) \subseteq AC^0(\Sigma_{k-1}^P) = \Theta_k^P.$$

□

As a consequence of this theorem 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 $NP \cap E_T(TALLY)$ and NP sets that are standard or general left cuts for some real number x , both in $L_2^{P,\Delta}$. Concretely, for $NP \cap E_T(TALLY)$, they established the following: if $A \in NP \cap E_T(TALLY)$ 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 theorem 3.3 we can solve this problem. First we show

Proposition 3.5 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 proposition 3.3, 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 $\{1, 1^2, \dots, 1^{p(m)}\}$.

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 $AC^0(NP)$ circuit that recognizes $L(N^A)$. On inputs of length n the circuit has a first level of query gates $1 \in T?, 1^2 \in T?, \dots, 1^{p(q(n))} \in T?$; 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 $L(N^A)$. \square

Now, applying the Θ -operator to the containment $NP(A) \subseteq \Theta_2^P$, it follows

Corollary 3.6 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 3.6 we can answer affirmatively the question before. Therefore, the situation for the class $NP \cap E_T(TALLY)$ 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 $A \in L_2^{P,\Theta}$. For the class of NP sets that are general or standard left cuts of some real number α the situation is analogous because these sets are in $E_T(TALLY)$. Note that, using the same techniques can be shown that $NP \cap E_{tt}(TALLY)$ 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 $NP \cap E_T(TALLY) \subseteq L_2^{P,\Theta}$. 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.
- [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).

Appendix

Proposition 3.2 Let $C \neq \emptyset$ be a class of languages closed by \leq_m^P -reducibility, then $\Theta(C) \subseteq AC^0(C)$.

Proof We follow the proof in [CS-92] that shows $P^{NP}[\log n] \subseteq AC^0(NP)$. 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:

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

and

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

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 .

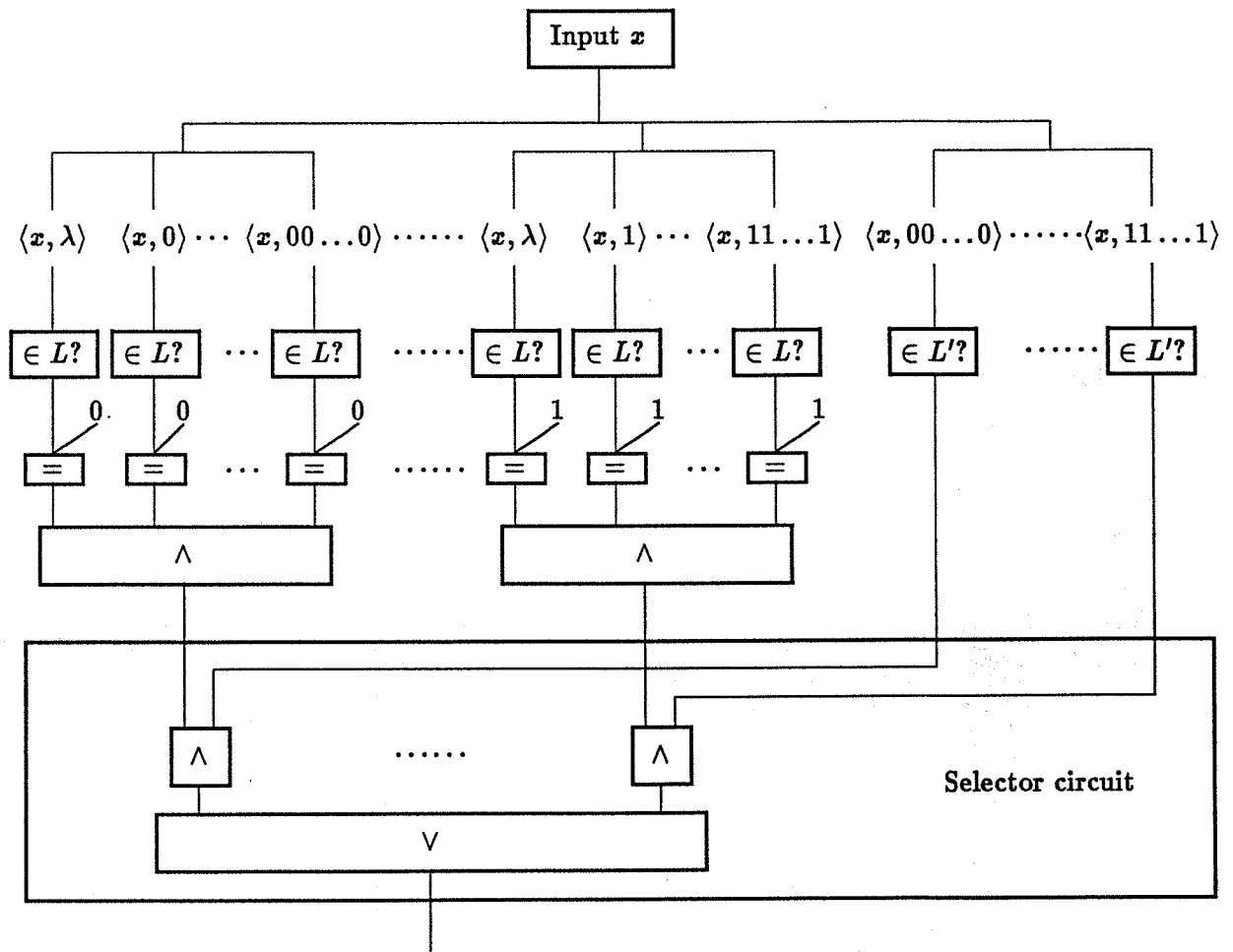


Figure 1.

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. \square

Proposition 3.3 For all $k \geq 1$, $\Theta_{k+1}^P = AC^0(\Sigma_k)$.

Proof We follow ideas showed in [CS-92]. Note that, by proposition 3.2, we only have to prove that $AC^0(\Sigma_k^P) \subseteq \Theta_{k+1}^P$. 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 $AC^0(\Sigma_k^P)$ with depth d and $p(n)$ a polynomial bound on the number of queries in each level. We define X as the set of sequences $\langle x, i_1, i_2, \dots, i_d, o \rangle$ belonging to $\{0, 1\}^* \times \{0, 1, \dots, p(n)\}^d \times \{0, 1\}$ (where $n = |x|$) such that there exist strings A_1, A_2, \dots, A_d in $\{0, 1\}^{\leq p(n)}$, each of them representing possible answers to the queries at levels $1, 2, \dots, d$ respectively and verifying the following:

A_d has j_d "YES" answers and these answers are correct if the inputs to queries of level d are computed from $x, A_1, A_2, \dots, A_{d-1}$.

A_{d-1} has j_{d-1} "YES" answers and these answers are correct if the inputs to queries of level $d-1$ are computed from $x, A_1, A_2, \dots, A_{d-2}$.

...

...

A_1 has j_1 "YES" answers and these answers are correct if the input to the circuit is x .

Finally, $\langle j_1, j_2, \dots, j_d, o' \rangle \geq \langle i_1, i_2, \dots, i_d, o \rangle$, 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, \dots, A_d as the answers to the queries.

Facts:

1. $X \in \Sigma_k^P$.
2. X is closed by lexicographical \leq order: if $\langle a_1, a_2, \dots, a_d, o_a \rangle \leq \langle b_1, b_2, \dots, b_d, o_b \rangle$ and $\langle x, b_1, b_2, \dots, b_d, o_b \rangle \in X$, then $\langle x, a_1, a_2, \dots, a_d, o_a \rangle$ is also in X .
3. Fixed x of length n , we define

$$\langle m_1, m_2, \dots, m_d, o_m \rangle = \max \{ \langle i_1, i_2, \dots, i_d, o_i \rangle \mid \langle x, i_1, i_2, \dots, i_d, o_i \rangle \in X \}.$$

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 is the number of "YES" answers of C_n on input x at the d^{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 $\langle k_1, k_2, \dots, k_d, o_k \rangle$ 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 $\langle x, k_1, k_2, \dots, k_d, o_k \rangle \in X$. Moreover, if there was $\langle x, i_1, i_2, \dots, i_d, o_i \rangle \in X$ such that $\langle x, i_1, i_2, \dots, i_d, o_i \rangle > \langle x, k_1, k_2, \dots, k_d, o_k \rangle$ it would imply that either

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

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_i '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 $j, 1 \leq j \leq d$ 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 $AC^0(\Sigma_k^P) \subseteq \Theta_{k+1}^P$. 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

$$\langle m_1, m_2, \dots, m_d, o_m \rangle = \max \{ \langle i_1, i_2, \dots, i_d, o_i \rangle \mid \langle x, i_1, i_2, \dots, i_d, o_i \rangle \in X \};$$

this can be done with at most $d \log p(n) + 1$ queries to X . 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

$$L(\{C_n\}, n \geq 1) \in \Theta_{k+1}^P.$$

□