On Dlogtime and Polylogtime Reductions*
(Extended Abstract)

Carme Àlvarez †
Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica Catalunya
Pau Gargallo 5, 08028 Barcelona, Spain

Birgit Jenner ‡
Fakultät für Informatik
Technische Universität München
80290 München. Germany

July 29. 1993

Abstract

We investigate properties of the relativized NC and AC hierarchies in their DLOGTIME-, respectively, ALOGTIME-uniform setting and show that these hierarchies can be characterized in terms of adaptive reducibility in logarithmic or polylogarithmic time, i.e. $O(\log n)^i$ for $i \geq 0$. As a corollary, the relationship between AC^i and NC^{i+1} reducibility is clarified by the result stating that if DLOGTIME-uniform AC^i and ALOGTIME-uniform NC^{i+1} reducibility coincide for $i = 0$ when applied to an arbitrary function class $\mathcal{F}$, then they coincide on $\mathcal{F}$ for all $i \geq 0$. Our results substantially generalize various previous results [Wi 90], [ABJ 91], [Ba 91].

Topics: *computational complexity, parallel algorithms*

1 Introduction

The concept of reducibility is a central one in complexity theory. Intuitively, a problem A is reducible to a problem B when an algorithm for B can be efficiently transformed into an algorithm for A. Loosely speaking, the complexity of A is bounded by the complexity of B "plus" the complexity of the reduction. Here, a natural assumption is that the complexity of the reduction should not be greater than the complexity of the problem we are reducing to. With the study of smaller and smaller complexity classes finer and finer reductions using lesser and lesser resources have become neccessary. At the extreme end, we have reducibilities via constant depth circuits [CSV 84] or DLOGTIME reducibility [Bu 87], suitable for studying completeness for classes such as AC^0 and NC^1.

Reducibilities can also be used to characterize complexity classes. For example, the classes Δ_2^P [St 77] and Θ_2^P [Wa 90] are, respectively, the closure of NP under polynomial-time Turing and truth-table reducibility. Θ_2^P has a variety of further characterizations by other types of reducibilities, e.g., logspace Turing reducibility or reducibility via AC^0 circuits (for an overview see [JT 93]).

A further recent example are the NC and AC hierarchies. Wilson shows that any class NC^{i+j} or, respectively, AC^{i+j} of these hierarchies can be characterized by applying AC^i or alternatively NC^{i+1} reducibility to NC^j, respectively, AC^j for $i \geq 0$, $j \geq 1$. Hence, when applied to classes in the NC and AC hierarchies, AC^i and NC^{i+1} reducibility coincide.

*Partially supported by DAAD and Spanish Government (Acción Integrada 1992 131-B, 313-AI-e-es/zk).

†Research supported by the ESPRIT Basic Research Actions Program of the EC under contract No. 7141 (project ALCOM II). E-mail address: alvarez@lsi.upc.es

‡On leave until March 1995, visiting Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica Catalunya supported by a Habilitationsstipendium of Deutsche Forschungsgemeinschaft (DFG-Je 154 2/1). E-mail address: jenner@lsi.upc.es

Wilson argued in [Wi 90] that a further investigation of the circumstances under which AC^i and NC^{i+1} reducibility coincide may be helpful in understanding the relationship between the classes AC^i and NC^{i+1}. His results are not completely satisfying for the following two principal reasons:

- The proofs do not relativize, and hence it was left open whether AC^i and NC^{i+1} reducibility also coincide when applied to classes not in the NC or AC hierarchy.

  For the classes L and NL (deterministic and nondeterministic logspace) this question was solved positively in [ABJ 91] with the help of adaptive logspace Turing reductions. But unfortunately, the proof techniques apply only to these two particular cases.

- The techniques used in [Wi 90] and [ABJ 91] essentially presuppose logspace uniformity of the circuits, which is unsatisfactory for the low level classes AC^0 and NC^1, since then the complexity of the circuit constructor exceeds the complexity of the circuit. Here DLOGTIME- and ALOGTIME-uniformity is preferable (see [Ru 81] [BIS 90] [BCGR 89]).

In this article, we show how the above mentioned two problems can be overcome by introducing the notion of adaptive reducibility in deterministic logarithmic and polylogarithmic time, i.e. in time $O((\log n)^i)$ for $i \geq 0$. We call this reducibility adaptive DLOG^{i}TIME reducibility (DLOGTIME reducibility for $i = 1$). We show that adaptive DLOG^{i}TIME-reductions are both fine and strong enough to decompose the relativized AC- and NC-hierarchies such that for $i \geq 0$, $j \geq 1$ any class NC^{i+j} can be characterized by applying adaptive DLOG^{i}TIME reducibility to the function classes FNC^j, and, analogously, for $i, j \geq 0$, any class AC^{i+j} by applying DLOG^{i}TIME reducibility to the class FAC^j. (Since DLOG^{i}TIME reducibilities for technical reason reduces to functions, here FAC^j and FNC^j denote the classes of functions (rather than the class of languages) corresponding to AC^j and, respectively, NC^j circuits.) For these results, it suffices to assume DLOGTIME-uniformity and, respectively, ALOGTIME-uniformity for the circuits. As a corollary, the relationship between DLOGTIME-uniform AC^i and ALOGTIME-uniform NC^{i+1} reducibility is clarified by the following result that strengthens a similiar result for logspace-uniform circuits announced in [Ba 91]: AC^i and NC^{i+1} reducibility coincide for all $i \geq 0$ when applied to a class $\mathcal{F}$ if they coincide on $\mathcal{F}$ for $i = 0$, i.e., if $AC^0(\mathcal{F}) = NC^1(\mathcal{F})$. In particular, this is the case if $\mathcal{F}$ is closed under NC^1 reducibility.

Hence, the results in [ABJ 91] mentioned above are generalized in two respects. First, they do not only apply to L and NL, but also to other classes known to be closed under NC^1 reducibility, like LOGCFL [Su 78] [BCDRT 89] or optL [AJ 92].

Our paper is organized as follows. In Section 2 we define the notions of DLOG^{i}TIME reducibility and uniform circuit classes with functional oracle gates. Previous uniformities below logspace do not apply to circuits with arbitrary fan-in and fan-out. Hence, we generalize the notion of direct (*DCL*) and extended connection language (*ECL*) (see [Ru 81]) to this case and obtain a DLOGTIME-*DCL*-uniformity for unbounded-fan-in circuits and an ALOGTIME-*ECL*-uniformity (DLOGTIME-*ECL*-uniformity) for bounded-fan-in circuits. These uniformities are extensions of corresponding notions in [BIS 90] and [Ru 81]. Besides decomposing parallel classes as mentioned above, we show in Section 3, that DLOG^{i}TIME-reducibility may also be used to decompose classes that may rather be called "sequential": firstly, "small" simultaneously space and time bounded classes, and, secondly, classes defined via polynomial-time bounded oracle Turing machines that may query $O(\log^i n)$ times an oracle in a function class $\mathcal{F}$. In Section 4, we state some normal forms for AC^i and NC^{i+1} circuits, and finally, in Section 5, we present the main result concerning the decomposition of the NC and AC hierarchies by DLOG^{i}TIME reductions. Due to space restrictions, we only give brief sketches of some of the proofs. (For details we refer to the full version of the paper [AJ 93].)

2 DLOG^{i}TIME-Reducibility and Circuit Classes

In this section, we define the notion of adaptive DLOG^{i}TIME reducibility and the uniform circuit classes that we will consider. Throughout this paper $\log^i n$ means the function $max(1, \lceil \log_2 n \rceil^i)$. All logarithms

are to the base two. We treat sets of words over a finite, fixed alphabet which when required we will identify with the set $\{0,1\}$. The empty word is denoted by λ. Functions map $\{0,1\}^*$ into $\{0,1\}^*$, and they satisfy that (i) all the values $f(x)$ have the same length for all x of the same length (a condition implicitly given if the function belongs to a circuit class), and (ii) they are polynomially-length bounded, i.e., that for all x, $|f(x)| \leq |x|^k$ for some constant k.

2.1 *Adaptive* DLOG^{i}TIME *Reducibility*

For polylogarithmic-time bounded classes, the standard Turing machine model with its sequential input access does not make sense, since only a prefix of the input could be read. To enable access to any bit of the input, we will use a known variant of the standard model (see e.g. [CKS 81]).

An *indirect-access machine* is an usual Turing machine that additionally has a special tape to access the input (input index tape) and four distinguished states ("read input", "input 1", "input 0", "input undefined"). When a string i is written in the input index tape and the machine is in the special state "read input" then it enters either the "input 1" or "input 0" state, depending on the ith bit of the input. We assume that the input is written in the leftmost cells of the input tape. If an input position greater than the input length is queried, then the machine enters an special state "input undefined". We will further assume that the content of the input index tape is not erased after querying the input. (The latter two assumptions allow to compute the input length in logarithmic time [BLT 92].)

Definition 2.1 *Using the machine model described above we define the following complexity classes:*

$$\begin{aligned} \textit{For } i \geq 0 \quad \text{DLOG}^i\text{TIME} &:= \textit{DTIME}(O(\log^i n)) \\ \textit{In particular,} \quad \text{DLOGTIME} &:= \textit{DTIME}(O(\log n)) \end{aligned}$$

These complexity classes are language classes (classes of 0–1 functions). The corresponding function classes are defined in the obvious way by supplying the indirect-access machine with an additional output tape. We will denote the function classes with the prefix "F" (e.g. FDLOGTIME).

By introducing nondeterminism or alternation the machine model described above gives rise to the classes NLOGTIME or, respectively, ALOGTIME (see [CKS 81] for such machines).

The concept of adaptive polylogtime reducibility is defined by attaching a functional oracle to the indirect-access machine in the following way.

An *indirect-access machine with adaptive oracle queries to an oracle function* is an indirect-access machine with three additional tapes, (oracle) query tape, (oracle) answer tape, answer index tape, and five distinguished states ("query oracle", "read answer", "answer 1", "answer 0", "answer undefined"). As for the input index tape, we assume that querying any bit position greater than the length of the oracle answer is notified by transition to a special state "answer bit undefined". Similarly, we assume that the answer index tape is not erased after querying.

The machine may construct queries on the (one-way write-only) oracle query tape and will receive the answer (the value of the function) on the oracle answer tape. To access the kth bit of the answer, the answer index tape is used in the same fashion as the input index tape: When a string k is written in the answer index tape and the machine is in the special state "read answer", it enters either the "answer 0", "answer 1", or the "answer undefined" state depending on the kth bit of the answer.

We assume that the input is always a prefix of the query, and that it is written initially in the oracle query tape. Furthermore, to obtain adaptivity, the last oracle answer will always be part of the query (following the input, but separated by a \$-symbol). Both input and last oracle answer are prevented to be overwritten by the machine, by letting the write-head for the new part of the query be positioned immediately after them. Whenever the machine is in the special state "query oracle", we assume the following to happen in one step: The contents of both query and answer tape are erased and the oracle answer appears on the answer tape as well as on the query tape. In the query tape, it is preceded by the input (separated by a \$-symbol and followed by a \$-symbol in order to separate it from the new part of the following query). The write head of the oracle query tape is positioned immediately after the second \$-symbol.

Initially:	$x\$_\wedge$	x input, "$_\wedge$" position of query tape head
Before first query:	$x\$q_{1\wedge}$	q_1 (new part of) first query
Just after first query:	$x\$a_1\$_\wedge$	a_1 answer of first query ($a_1 = f(x\$q_1)$ for oracle function f
	$\vdots$	
Just after ith query:	$x\$a_i\$_\wedge$	a_i answer of ith query
Before $i+1$st query:	$x\$a_i\$q_{i+1\wedge}$	q_{i+1} (new part of) $i+1$st query

Table 1: Content of the oracle query tape (adaptive queries)

Observe that the content of the oracle query tape at any time of the computation may be described according to Table 1 above.

Definition 2.2 *For $i \geq 1$,* DLOG^{i}TIME($\mathcal{F}$) *is the class of languages that can be computed by $O(\log^i n)$-time bounded indirect-access machines with adaptive oracle queries to a function in $\mathcal{F}$. If $\mathcal{F}$ is a language class (i.e. a class of 0–1 functions), then we obtain with the above model a* DLOG^{i}TIME *Turing reducibility. We denote* DLOG^{1}TIME *by* DLOGTIME.

2.2 *Boolean Circuits*

Boolean circuits with *bounded fan-in* are finite directed acyclic graphs with nodes or gates up to indegree 2 with a certain label or type. Nodes of indegree zero are the input nodes $x_0, x_1, \ldots, x_{n-1}$ or nodes labelled 0 or 1; nodes with indegree one are labelled $\neg$ or *id* (the identity function), and nodes with indegree two are labelled $\wedge$ or $\vee$. Some of the nodes are specified as output nodes $y_0, y_1, \ldots, y_{m-1}$. In circuits with *unbounded fan-in* there is no restriction on the indegree of the $\vee$ and $\wedge$ nodes. In this case these nodes are also called existential and universal, respectively.

A *circuit family* $\{C_n\} := (C_1, C_2, \ldots)$ computes a function f, if the output of C_n on input x of length n is the same as $f(x)$ for all x. If $f(x) \in \{0,1\}$ for all x, i.e., there is only one output node in each circuit of the family, then we can also say that $\{C_n\}$ accepts the set of all x for which $f(x) = 1$. For a circuit C, the size of C *size*(C), is the number of nodes C contains. The depth of C, *depth*(C), is the length of the longest path from some input node to some output node.

We will allow our circuits to have access to *oracle gates*, which compute the value $f(x)$ of x for a functional oracle $f : \{0,1\}^l \to \{0,1\}^m$ (see [Co 85]). Note that usually oracle nodes determine the membership of a string x in an oracle set; this corresponds in our approach to taking $m = 1$, i.e. using a 0–1 valued function f instead of the set $L_f := \{x \mid f(x) = 1\}$ (see e.g. [Wi 87], [Wi 90]). For unbounded fan-in circuits, oracle nodes have depth 1. In the case of bounded fan-in circuits an oracle gate with k inputs and k' outputs contributes $\log k + \log k'$ to the depth of the circuit. This is the standard way of counting the depth of oracle nodes (see e.g. [Co 85]). The reason for this choice is that the use of an oracle gate for, say an unbounded fan-in existential gate g, should be charged alike as would be the bounded fan-in subcircuit computing g that has depth logarithmic in the number of its inputs.

2.3 *Uniformity*

We restrict ourselves to *uniform circuit families* for which the codification of the nth circuit may be "easily" determined from n. There exists a variety of uniformity conditions for circuits without oracle nodes [Ru 81] [BIS 90] [Bu 87] [BCGR 89]. For AC^i or NC^{i+1} with $i \geq 1$, most of these uniformities result in the same class, namely the commonly considered logspace uniform AC^i or NC^{i+1}. For $i = 0$ there are differences and much effort has been devoted in particular to develop uniformity conditions that

		Logspace-uniform NC^1 ($=U_{BC}$-uniform NC^1 [Ru 81])
		$\mid$
		$\mid$
ALOGTIME	=	ALOGTIME-ECL-uniform NC^1 ($=U_{E^*}$-uniform NC^1 [Ru 81])
	=	ALOGTIME-Formula-language-uniform NC^1 [BIS 90]
	=	DLOGTIME-ECL-uniform NC^1 ($=U_E$-uniform NC^1 [Ru 81])
	=	DLOGTIME-Formula-language-uniform NC^1 [BIS 90]
		$\mid$
		$\mid$
		DLOGTIME-DCL-uniform NC^1 ($=U_D$-uniform NC^1 [Ru 81])
		$\mid$
		$\mid$
LH	=	DLOGTIME-DCL-uniform AC^0 [BCGR 89] [BIS 90]
	=	DLOGTIME-Expression-language-uniform AC^0 [BIS 90]
	=	FO-DCL-uniform AC^0 [BIS 90]
	=	FO-Expression-language-uniform AC^0 [BIS 90]
	=	FO-definable languages [BIS 90]

Table 2: Uniformities for AC^0 and NC^1

make AC^0 equal to the logtime hierarchy (LH) and NC^1 equal to ALOGTIME (for an overview see Table 2.3).

In the following we generalize the notions of extended [Ru 81] and direct connection language [BIS 90] [BCGR 89] for bounded fan-in, and, respectively, unbounded fan-in circuits with oracle gates of arbitrary fan-in and fan-out. For both kinds of circuits we assume that each oracle gate number is a 3-tuple (o, k, k'), that contains the fan-in k and fan-out k' of the gate. (Here suffices any codification for which the fan-in and fan-out of any oracle gate can be determined in DLOGTIME.) To be able to refer to the output gates in DLOGTIME, we furthermore assume that the output gates of a circuit C_n are numbered $2^{|size(C_n)|}, \ldots, 2^{|size(C_n)|} + m - 1$.

For a bounded fan-in circuit C with or without oracle gates for a function f, let y be a gate and let $p \in ([0], [1], [(k, k', i, j)])^*$ such that $k, k', i, j \in \{0,1\}^*$. Then $g(p)$ (for $p \neq \lambda$) is the gate reached when p is followed as a path towards the inputs of C by starting at g, going left (or right) when [0] (or [1]) appears and following input $i + 1$ when $[(k, k', i, j)]$ appears. $[(k, k', i, j)]$ stands for an oracle gate with fan-in k, fan-out k' that is reached via its jth output and left via its ith input. For example, $g([1])$ is the right input gate of g, and $g([0][(11, 1, 10, 1])$ is the 2nd input (of the oracle gate with fan-in 3 and fan-out 1 that has been reached via its first output)) of the left input to g.

For a bounded fan-in circuit family $\{C_n\}$ with or without oracle gates, the *extended connection language* for C_n, ECL_C, consists of 4-tuples (x, g, p, y), where $|x| = n$, $g \in \{0,1\}^*$ (g is the gate number), $y \in \{x_1, x_2, \ldots x_n, \neg, id, \wedge, \vee\} \cup \{0,1\}^*$ (y is either gate type or gate number), and $p \in ([0], [1], [(k, k', i, j)])^*$ with $|p| \in O(\log(size(C_n)))$ such that (i) if $p = \lambda$ then y is the type of gate g, and (ii) if $p \neq \lambda$ then y is the gate number of $g(p)$, including, if $g(p)$ is oracle gate, the number of $g(p)$'s output.

As the extended connection language of Ruzzo [Ru 81], ECL_C encodes local information about the wiring of C_n within distance $\log(size(C_n))$. But it additionally takes into account that a gate with fan-in k and fan-out k' contributes $\log k + \log k'$ to the depth of the circuit.

Such aspects do not play any role for unbounded fan-in circuits. Hence, we are here only interested in the information about direct wiring between two gates. Generalizing the notion of [BIS 90], we define for an unbounded fan-in circuit family $\{C_n\}$ with or without oracle gates the *direct connection language* for C_n, DCL_C, is as above, except $p \in \{\lambda, [0], [(k, k', i, j)]\}$ such that (i) if $p = \lambda$ then y is the type of gate g, (ii) if $p = [0]$ then y is the gate number of an input to g, (iii) if $p = [(k, k', i, j)]$ then y is an oracle

gate (with fan-in k and fan-out k') and y is the ith input to g.

Definition 2.3 *An unbounded fan-in circuit family with or without oracle gates $\{C_n\}$ is DLOGTIME-DCL-uniform if there is a DLOGTIME machine that recognizes DCL_C. A bounded fan-in circuit family with or without oracle gates $\{C_n\}$ is DLOGTIME-ECL-uniform (*ALOGTIME-*ECL-uniform) if there is a DLOGTIME(*ALOGTIME*) machine that recognizes ECL_C.*

Note that for circuits without oracle gates DLOGTIME-DCL-uniformity coincides with DLOGTIME-*DCL*-uniformity of [BIS 90] (see also [BCGR 89] for a linear time version), and ALOGTIME-ECL-uniform NC^1 (DLOGTIME-*ECL*-uniformity) coincides with the U_E*-uniform NC^1 (U_E-uniformity) of [Ru 81].

Definition 2.4 *We denote by* FAC^i *for* $i \geq 0$ *and by* FNC^i *for* $i \geq 1$ *the class of functions computable by DLOGTIME-DCL-uniform unbounded fan-in and, respectively,* ALOGTIME-*ECL-uniform bounded fan-in circuit families $\{C_n\}$ of polynomial size, $size(C_n) = n^{O(1)}$, and polylogarithmic depth, $depth(C_n) = O(\log^i n)$. The corresponding classes of languages (0–1 functions) are denoted by* AC^i *and* NC^i, *respectively. If the circuits contain functional oracle nodes for a function f in a function class $\mathcal{F}$, we will refer to* $FAC^i(\mathcal{F})$ *and* $FNC^i(\mathcal{F})$, *or in the case of languages, to* $AC^i(\mathcal{F})$ *and* $NC^i(\mathcal{F})$.

Note that we have defined the classes $NC^i(\mathcal{F})$ in terms of ALOGTIME-*ECL*-uniformity rather than DLOGTIME-*ECL*-uniformity, although it was shown by Ruzzo [Ru 81] that there is no difference for the unrelativized classes. Ruzzo's proof for ALOGTIME-*ECL*-uniform NC^1= DLOGTIME-*ECL*-uniform NC^1 (see Theorem 3 plus 4 in [Ru 81]) is not applicable for the general relativized case, i.e., when oracle gates of arbitrary fan-in and fan-out are present. It seems that for any determination of the fan-out or the relevant output of an oracle gate logarithmic time is needed, independently of the size of the fan-in or fan-out of the gate. Thus, in general applying Ruzzo's technique results in an circuit of depth $O(\log^2 n)$. A similiar phenomenon occurs in theorem 5.3 of Section 5 where we have to presuppose the weaker ALOGTIME uniformity condition to achieve decomposability. Ruzzo's proof *is* applicable for 0–1 valued functions with a fixed input size $p(n)$ for each circuit C_n. Hence, the difficulty lies in padding up the input size of the oracle gates and splitting the oracle into a 0–1 function that specifies any output bit. Precisely when this is possible for a class $\mathcal{F}$ (like e.g. NC^k), we may assume DLOGTIME-*ECL*-uniformity rather than ALOGTIME-*ECL*-uniformity.

3 Some Properties of DLOG^{i}TIME Reducibility

If in our definition of adaptive DLOG^{i}TIME reducibility instead of arbitrary functional oracles 0–1 functions, i.e. set oracles, are used, we obtain a DLOGTIME or DLOG^{i}TIME Turing reducibility. Many complexity classes are closed under DLOGTIME Turing reducibility, e.g. AC^0, NC^1, L, NL, AC^i,SC^i, and NC^j for $i \geq 1, j \geq 2$. Closure under DLOG^{i}TIME Turing reducibility for $i \geq 2$ seems to hold only for SC^i. Although we have DLOG^{i}TIME $\subseteq NC^i \cap AC^i$ it is doubtful, whether NC^i or AC^i are closed under DLOG^{i}TIME reducibility. This emphasizes the (possible) difference between NC^i and SC^i.

We will be interested in the more general case of DLOG^{i}TIME reductions to functional oracles that are not restricted to one output symbol. In Section 5 we will see that in this case there is a precise characterization of the classes DLOG^{i}TIME($\mathcal{F}$) for any (function) class $\mathcal{F}$ closed under FAC^0 reducibility, (like the functional versions of the classes AC^0, NC^1, L, NL, AC^i,SC^i, and NC^j for $i \geq 1, j \geq 2$, mentioned above) in terms of unbounded or bounded fan-in circuits of depth $O(\log^i n)$ with oracle gates for $\mathcal{F}$. In this section we show characterizations of the class DLOG^{i}TIME($\mathcal{F}$) for function classes $\mathcal{F}$ not known to be closed under FAC^0 reducibility. The two results that we present below may be interpreted as decomposition results of "sequential" classes by DLOG^{i}TIME-reducibility.

Denote by DSPACE,TIME($O(\log^i n), O(\log^j n)$) the class of languages that are accepted by simultaneously $O(\log^i n)$-space bounded and $O(\log^j n)$-time bounded Turing machines.

Proposition 3.1 *For all $i \geq 0, j \geq 1$ such that $i \leq j$ it holds*
DSPACE,TIME($O(\log^j n), O(\log^{i+j} n)$) = DLOG^{i}TIME(FDLOG^{j}TIME).

Proof. For the inclusion from left to right, let M be a machine that is simultaneously $c \times \log^j n$-space and $c \cdot \log^{i+j} n$-time bounded. We divide the computation of M on input x in $\log^i n$ steps of size $c \cdot \log^j n$. After step k, M will be in configuration c_k. Note that any configuration of M is bounded in size by $O(\log^j n)$. To simulate the computation of M on a given input x, define an oracle function f_M, $f_M(x\$c_k) = c_{k+1}$, where c_{k+1} is the successor configuration of c_k in a computation of M on input x, if it exists, and c_k otherwise. Let the last bit of c_k indicate whether c_k is a final accepting configuration or not. It is not hard to see that $f \in \mathrm{FDLOG}^j\mathrm{TIME}$ is satisfied. To simulate M with f as oracle, a machine M' writes the query $x\$c_0$, where c_0 is the initial configuration of M, in the oracle query tape and then just queries $\log^i n$ times. After each query, a unary counter initially set to $\log^i n$ is decremented, and M stops querying when the counter reaches 0. The adaptive query mechanism ensures that successively $x\$c_0, x\$c_1, x\$c_2, \ldots, x\$c_{\log^i n}$ are queried, since the input x and the last oracle answer c_k are always part of the following query. With the query $x\$c_{\log^i n}$, f eventually will have a final configuration of M on its oracle answer tape, and M' can check acception or rejection by looking at the last bit. For this, M' computes the length of the oracle answer by binary search in time $O(\log(\log^i n))$.

For the inclusion from right to left, we completely simulate the $O(\log^i n)$-time bounded base machine inclusive the oracle computation. For this, we need time bounded by the product of the time of the base machine and the time of a machine for the oracle function, i.e. $O(\log^{i+j})$. Since $i \leq j$, the space used by the base machine is bounded by the length of the oracle answers, and the simulation thus consumes space $O(\log^j n)$.

□

In particular, we have for $i = 1$, $\mathrm{DSPACE,TIME}(O(\log n), O(\log^{j+1} n)) = \mathrm{DLOGTIME}(\mathrm{FDLOG}^j\mathrm{TIME})$. This may be contrasted with the result of Ruzzo $\mathrm{ASPACE,TIME}(O(\log n), O(\log^{j+1} n)) = \mathrm{NC}^{j+1}$ for $j \geq 1$ [Ru 81] that shows that alternating simultaneously logspace and $O(\log^{j+1})$-time bounded Turing machines are equivalent to logspace-uniform bounded fan-in circuits of depth $O(\log^{j+1})$. Since, as shown in Section 5, $\mathrm{NC}^{j+1} = \mathrm{DLOGTIME}(\mathrm{FNC}^j)$, DLOGTIME reducibility has the peculiar property of behaving sequentially, when applied to sequential classes, and parallel, when applied to parallel classes.

Our second example of a decomposition result for sequential classes concerns the class $\mathrm{P}_{\log^k}(\mathcal{F})$ of languages that are accepted by polynomial-time bounded oracle Turing machines that may query $O(\log^k n)$ times an oracle function in the function class $\mathcal{F}$.

Proposition 3.2 *Let $\mathcal{F}$ be an arbitrary class of functions. Then,*
$\mathrm{P}_{\log^{i+j}}(\mathcal{F}) = \mathrm{DLOG}^i\mathrm{TIME}(FP_{\log^j}(\mathcal{F}))$ *for $i \geq 1, j \geq 0$.*

Proof. For the inclusion from left to right, let x be an input of an $\mathrm{P}_{\log^{i+j}}(f)$ machine M with oracle function $f \in \mathcal{F}$. Let the bound on the number of queries of M on inputs of length n be $c \cdot \log^{i+j}$. Define a function f, $f(x\$v) = vv'$ that computes for input x and a sequence of oracle answers v, $v \in \{0,1\}^{\leq c \cdot \log^{i+j} n}$, the next $c \cdot \log^j$ oracle answers v', $v' \in \{0,1\}^{c \cdot \log^j n} \cup \{\top, \bot\}$, that M receives when the sequence v of oracle answers is followed on a computation on input x. If no further query is posed, then $f(x\$v) = \top$ or $f(x\$v) = \bot$, depending on whether M accepts or rejects. Clearly, $f \in \mathrm{FP}_{\log^j}(f)$. As in the proof above, now it suffices for a $\mathrm{DLOG}^i\mathrm{TIME}$ machine with adaptive oracle f to query just $\log^i n$ times and then check the last bit of the final answer for acception ($\top$) or rejection ($\bot$).

The other inclusion is obtained via straightforward simulation. Observe that a polynomial-time bounded machine can store any oracle answer and hence can produce a complete new query that consists of input, last oracle answer and the new constructed part. The time bound of the base machine and the query bound on the transducer for the oracle yield a bound of $O(\log^{i+j})$ on the number of queries.

□

In particular, we obtain $\mathrm{P}_{\log^i}(\mathrm{NP}) = \mathrm{DLOG}^i\mathrm{TIME}(\mathrm{FP}_1(\mathrm{NP}))$ by setting $\mathcal{F} = \mathrm{NP}$, and with $i = 1$ a new characterization of $\Theta_2^{\mathrm{P}} = \mathrm{P}_{\log}(\mathrm{NP}) = \mathrm{DLOGTIME}(\mathrm{FP}_1(\mathrm{NP}))$. $\mathrm{DLOG}^i\mathrm{TIME}$-reducibility here nicely separates adaptiveness and "sequential" computation power of $\mathrm{P}_{\log^i}(\mathcal{F})$ computations into the complexity of the base machine on the one hand, and, the computational power of its oracle, on the other.

Results similiar to Proposition 3.2 hold for adaptive logspace reductions (see [Al 93], where the relationship between adaptive DLOG^{i}TIME reducibility and logspace adaptive reductions has been investigated).

4 Normal form circuits for $\mathrm{NC}^i(\mathcal{F})$ and $\mathrm{AC}^i(\mathcal{F})$

To obtain our main theorem (see section 5), the decomposition of $\mathrm{NC}^{i+1}(\mathcal{F})$ or $\mathrm{AC}^i(\mathcal{F})$ for $i \geq 0$, we will use some normal form lemmas for circuits with oracle gates. Similar lemmas without oracle gates and for LOGSPACE uniformity were introduced by Wilson in [Wi 90].

In the following lemma we construct from a circuit C_n with oracle gates an equivalent layout circuit C'_n that can be easily subdivided. For any gate g of a circuit C, let $path(g)$ be the length of the largest path from some input node to g. The input gates have a path equal to 0. For a circuit C, level p of C consists of all gates g with $path(g) = p$ for $0 \leq p \leq depth(C)$. A uniform family of circuits is *layout* uniform if (1) for any gate g path(g) is part of the gate name, and (2) no gate at any level p is connected with some gate at level less than $p - 1$.

Lemma 4.1 *For any DLOGTIME-ECL-uniform (DLOGTIME-DCL-uniform) bounded (unbounded) fan-in circuit family $\{C_n\}$ of depth $d(n)$ and with oracle gates for a function f there is an equivalent layout uniform circuit family $\{C'_n\}$ with the same oracle and uniformity, fan-in and depth restrictions.*

Proof. *(sketch)* We can extend the leveling technique of Wilson (Theorem 3.2 of [Wi 90]). C'_n has $d(n)$ levels as the circuit depth, and in each level r there is a "copy" of each original gate s. The name of this copy is (r, s). Gates (r, s) and (r', s') are connected by a path if gates s and s' in C_n are connected by the same path of length $r - r'$. For the input gates g_s the corresponding gate in level $r \geq 1$, $g_{r,s}$ is an identity gate that has as input $g_{r-1,s}$. These identity gates pass the input values from level to level.

With respect to uniformity, we get that C'_n preserves paths between gates of different levels. Hence by simulation of the DLOGTIME-machine that accepts ECL_C on the appropiate inputs, $ECL_{C'}$ can be accepted by a DLOGTIME-machine. □

Note that only in the case of unbounded fan-in circuits the depth of a gate determines its level and may be read off its name, for layout uniform circuits. For bounded fan-in circuits, since the depth of a gate g essentially depends on the fan-in of the oracle gates that occur on paths between the inputs and g, a gate on level p may have depth greater than p. Because of this fact it is not possible to subdivide NC^i circuits into subcircuits of equal depth by looking at the gate number of each gate only. Rosmanith (see [Ba 91]) suggested a trick, introduce identity gates, to avoid this problem. In the following lemma we formalize this trick for DLOGTIME-*ECL*-uniform circuits with oracle gates.

Let C_n be a circuit and let g_i be a gate of C_n. We denote by C_n^i the circuit that is identical to C_n but has only one output gate, the gate g_i. Let C_n and D_n be two circuits and g_i and g'_i be gates of C_n and D_n, respectively. We say that g'_i is a replica of g_i iff C_n^i and D_n^i compute the same function.

Lemma 4.2 *For any DLOGTIME-ECL-uniform bounded fan-in circuit family $\{C_n\}$ with oracle gates for a function g, there is an equivalent DLOGTIME-ECL-uniform bounded fan-in circuit family $\{C'_n\}$ with oracle gates for g such that for each gate g_i in C_n there is a replica g'_i of g_i in C'_n that satisfies $depth(g'_i) \leq 2depth(g_i)$.*

Proof. *(sketch)* Let f be the function computed by $\{C_n\}$. We can construct a DLOGTIME-*DCL*-uniform circuit family $\{C'_n\}$ such that for each gate g_i in C_n there is a replica g'_i of it in C'_n. and the longest path from an input to gate g'_i coincides with the depth of g_i in C_n. To ensure $depth(g'_i) \leq 2depth(g_i)$. we add for each oracle gate g_i a chain of identity gates of length $\lceil \log(\text{fan-in}(g_i)) \rceil$ between g_i and each of its input gates. Each gate name will contain information not only about its fan-in and fan out, but it will be composed of its old name i and whether it is the lth element in the added chain between the jth input gate to the oracle gate g_i, or whether it is a replica of a gate in C_n.

Since $\{C_n\}$ is DLOGTIME-*ECL*-uniform, there exists a DTM M that accepts ECL_C corresponding to $\{C_n\}$ in logarithmic time. $ECL_{C'}$ will differ from ECL_C not only with the gate names, but also in paths p between gates. First let us compare a path description p' in $ECL_{C'}$ and the corresponding path p in ECL_C:
(i) There is no oracle gate in the path. Hence the path description in ECL_C is not different from the path description in $ECL_{C'}$, $p = p'$.
(ii) There is at least one oracle gate in the path. Then $p' = p_1 \dots p_{i_1} \dots p_{i_r} \dots p_s$ where $p_{i_l} \in [k, k', in, out]$ for $k, k', in, out \in \{0,1\}^*$ and the rest of the components belong to $\{[0],[1]\}$. (We mark the components of p' corresponding to oracle gates with two subindexes). Differences with respect to the original path p in ECL_C are the components of p' that follow an oracle gate. In C'_n we have added one path between each input gate to an oracle gate and this oracle gate. Hence, by the construction of C'_n from C_n, we have that $p = p_1 \dots p_{i_1} p_{i_1+\alpha_1+1} \dots p_{i_r} p_{i_r+\alpha_r+1} \dots p_s$ where $\alpha_l = \lceil \log(in_{i_l}) \rceil$ and in_{i_l} is the fan-in codified in p_{i_l}.
Since p' contains the fan-in of each oracle gate which is a node in the path, we can construct a DLOGTIME-machine M' that deletes parts of p' corresponding to chains added to oracle gates. By simulation of M, M' can verify in DLOGTIME that the new path is a correct path in C_n. □

Once we have obtained the layout uniform circuit family where the name of each gate contains information about its depth, we can divide each circuit in subcircuits of equal depth obtaining an equivalent circuit which consists of a chain of oracle gates.

Lemma 4.3 *Let f be a function. Then*
(i) For any DLOGTIME-ECL-uniform $\mathrm{FNC}^{i+j}(f)$ *circuit family $\{C_n\}$, for $i \geq 0$ and $j \geq 1$, there exists an equivalent DLOGTIME-ECL-uniform circuit family $\{C'_n\}$, where each circuit is a chain of length $O(\log^i n)$ of DLOGTIME-ECL-uniform* $\mathrm{FNC}^j(f)$ *oracle gates.*
(ii) For any DLOGTIME-DCL-uniform $\mathrm{FAC}^{i+j}(f)$ *circuit family , for $i, j \geq 0$, there exists an equivalent DLOGTIME-DCL-uniform circuit family where each circuit is a chain of length $O(\log^i n)$ of DLOGTIME-DCL-uniform* $\mathrm{FAC}^j(f)$ *oracle gates.*

Proof. *(sketch)* The proof of (i) and (ii) follow the same lines, we only sketch the proof of (i). (For (ii) we do not need to apply Lemma 4.2 because depth and path coincide for unbounded fan-in circuits.) Let f be a function computed by an NC^{i+j} circuit family $\{C_n\}$ with oracle gates for a function g. By Lemmas 4.1 and 4.2 there is a layout uniform $\mathrm{NC}^{i+j}(\mathcal{F})$ circuit family $\{C'_n\}$ equivalent to $\{C_n\}$ such that the depth of a gate $depth(g_{r,s}) \leq 2r$. We use this layout form of the circuit in order to divide the computation into identical parts of the circuit. Each layout circuit C'_n can be seen as a sequence of subcircuits where each subcircuit consists of 2^t consecutive levels of C'_n where $t = j|n'|$ and $n' = |n|$, and each L^k_n receives as input the output of L^{k-1}_n, for $k > 0$. Hence each subcircuit has a power of two depth bounded by a function in $O(\log^j n)$. All subcircuits but L^0_n are identical. L^0_n has as input the same input than C'_n while in the rest of subcircuits the input is the value computed by the subcircuit immediately before. Hence we only distinguish two subcircuits, L^0_n and L^1_n. where L^0_n computes the first 2^t levels of C'_n and L^1_n computes 2^t consecutive levels from the level given as input. Now it only rests to define an oracle function h that computes each subcircuit which results fromthe splitted C'_n. We define h in such a way that it computes L^0_n or L^1_n depending the input:

$$h(x\$z) = \begin{cases} L^0_{|x|}(x)\$z + 1, & \text{if } z = 0: \\ L^1_{|x|}(x)\$z + 1, & \text{otherwise.} \end{cases}$$

By the definition of L^0_n and L^1_n, it is not difficult to see that h can be computed by a circuit family $\{H_n\}$ in $\mathrm{FNC}^j(\mathcal{F})$.

In order to prove the DLOGTIME-*ECL*-uniformity of $\{H_n\}$ we can prove the uniformity of each of the subcircuits that define H_n. It is easy to see that the sequential composition of these DLOGTIME-*ECL*-uniform circuits is DLOGTIME-*ECL*-uniform. And in order to prove the uniformity of L^0_n or L^1_n

we can use the DLOGTIME-machine for $ECL_{C'}$. We only want to remark two tricks used in order to keep the DLOGTIME complexity:
(i) In the construction of the circuit, considering $2^t + 1$ the depth of each subcircuit, allows to compute the depth $O(\log^j n)$ in time $O(\log n)$.
(ii) We need to know the fan-out of each oracle gate in order to define L_n^0 and L_n^1. Since it can be the case that an oracle gate is part of the output level of L_n^0 or L_n^1. Hence we define a new output gate for each oracle gate output bit instead of a unique output gate.
□

5 Decomposing NC($\mathcal{F}$) and AC($\mathcal{F}$) using DLOG^{i}TIME reductions

The main result of this section is the decomposition of the relativized NC - and AC - hierarchies in terms of adaptive DLOG^{i}TIME reductions. More specifically we show the following:

Main theorem: Let $\mathcal{F}$ be an arbitrary function class. Then,
(i) $\mathrm{NC}^{i+j}(\mathcal{F})$=DLOG^{i}TIME(FNC$^j(\mathcal{F})$) for $i \geq 0$ and $j \geq 1$.
(ii) $\mathrm{AC}^{i+j}(\mathcal{F})$=DLOG^{i}TIME(FAC$^j(\mathcal{F})$) for $i, j \geq 0$.

Hence the introduction of DLOG^{i}TIME time reducibility allows us to study the relativized NC and AC hierarchies considering ALOGTIME and DLOGTIME uniformities respectively, and allows us to answer the question raised by Wilson in [Wi 90]: under which circumstances do relativized NC^{i+1} and AC^i coincide. The answer follows directly from the **Main theorem**.

Corollary: If a function class $\mathcal{F}$ satisfies $\mathrm{AC}^0(\mathcal{F}) = \mathrm{NC}^1(\mathcal{F})$ then $\mathrm{AC}^i(\mathcal{F}) = \mathrm{NC}^{i+1}(\mathcal{F})$ for all $i \geq 0$.

This result generalizes the results in [ABJ 91], it does not only apply to L and NL, but also to other classes known to be closed under NC^1 reducibility.

The proof of the main theorem follows the same lines for (i) and (ii). It consists of three lemmas. We present the lemmas for the NC case which is a little more involved than (ii). For showing the inclusion from left to right we use the "normal form" lemmas for circuits with oracle gates introduced in Section 4. In this way we get a circuit which consists of a chain of oracle nodes, and then it is easy to define a sequential DLOG^{i}TIME machine with adaptive acces to the functional oracle corresponding to the oracle nodes.

Lemma 5.1 $\mathrm{NC}^{i+j}(\mathcal{F}) \subseteq$ DLOG^{i}TIME(FNC$^j(\mathcal{F})$), *for* $i \geq 0$ *and* $j \geq 1$ *considering DLOGTIME-ECL-uniformity.*

Proof. *(sketch)* By Lemma 4.3 we obtain from a $\mathrm{NC}^{i+j}(\mathcal{F})$ circuit family, a circuit family in normal form where each circuit consists of a chain of $O(\log^i n)$ oracle nodes in FNC$^j(\mathcal{F})$. It remains to modify the oracle function in such a way that it can be used as functional oracle in a sublinear time machine with adaptive access to the oracle. The oracle function not only evaluates a sequence of levels of the original circuit, but it also increments a counter of the number of queries. The base machine first initializes the queries counter and then makes $O(\log^i n)$ queries to functional oracle. □

Now it only remains to prove the right to left inclusion of the main theorem. We divide the proof in two parts. The first part consists in the simulation of the DLOG^{i}TIME sequential machine with adaptive access to a functional oracle by a circuit with oracle gates. The second part consists in the substitution of the oracle nodes by the corresponding subcircuits, for a circuit family in $\mathrm{NC}^i(\mathrm{NC}^j(f))$. Since we use ALOGTIME-*ECL*-uniformity the proof is considerably more involved than the Wilson proof of Theorem 3.1 in [Wi 90]. The following two lemmas formalize the two parts of the proof.

Lemma 5.2 $\mathrm{DLOG^iTIME}(\mathcal{F}) \subseteq \mathrm{NC}^{i+1}(\mathcal{F})$ *for* $i \geq 0$ *considering DLOGTIME-ECL-uniformity.*

Proof. *(sketch)* Use Ladner's simulation of a sequential machine by a boolean circuit [La 75]and integrate the indirect access to the input tape and to the answer oracle tape. □

Now it only rests to substitute the oracle gates by circuits. In this case we are not able to prove the result considering DLOGTIME-*ECL*-uniformity instead of ALOGTIME-*ECL*-uniformity. Given a path in the circuit resulting from substituting the oracle gates by the corresponding subcircuits, we do not know whether the path goes through some of these subcircuits. In order to decompose the path correctly we use the nondeterminism for guessing the different parts of the path, and, afterwards, in order to verify that these parts are correct, we use the ALOGTIME algorithms for the *ECL* of the original circuit family and for the circuit family which computes the oracle function.

Lemma 5.3 *Let* $\mathcal{F}$ *be an arbitrary function class. Then,*
$\mathrm{NC}^i(\mathrm{FNC}^j(\mathcal{F})) \subseteq \mathrm{NC}^{i+j-1}(\mathcal{F})$, *for* $i, j \geq 1$ *considering* ALOGTIME-*ECL-uniformity.*

Proof. *(sketch)* Let A be a language accepted by an ALOGTIME-*ECL*-uniform circuit family $\{C_n\}$ with oracle gates for a function $g \in \mathrm{FNC}^j(\mathcal{F})$. Each circuit C_n has depth $c_1 \log^i n$ for some constant c_1, and size $p(n)$ for some polynomial p. Since $g \in \mathrm{FNC}^j(\mathcal{F})$ there exists an ALOGTIME-*ECL*-uniform family of circuits $\{O_m\}$ with oracle gates for a function $f \in \mathcal{F}$ that computes g. Each circuit O_m has depth $c_2 \log^j m$ for some constant c_2, and size $q(m)$, for some polynomial q. We will construct an ALOGTIME-*ECL*-uniform family $\{C'_n\}$ that accepts A by substituting each oracle gate by the circuit which computes g on inputs of length equal to oracle gates fan-in. We define C'_n as a sequence of subcircuits S_r, one for each each gate g_r of the original circuit C_n, where the definition of S_r depends of whether g_r is an oracle gate, and the connections between subcircuits S_r and S_t depend of the connections between gates g_r and g_t in C_n. For each r corresponding to a gate number of C_n, and $r > 0$ we define S_r as follows :
-case 1: If g_r is an oracle gate then S_r is defined by gates $g_{r,t}$ one for each gate g_t of the circuit O_{fin_r} where fin_r =fan-in(g_r). The type and the inputs of $g_{r,t}$ depend of whether g_t is an input, an output or an internal gate in O_{fin_r}:
-case 1.1: g_t is an input gate. We define the type of $g_{r,t}$ as the *identity*. Since g_t is the $t+1$th input of O_{fin_r}, the input of $g_{r,t}$ depends of the $t+1$th input of oracle gate g_r. Let g_s be the $t+1$th input of g_r in C_n. Then it could be the case that
-case 1.1.1: g_s is an oracle gate and its lth output is the tth input of g_r. Hence the input of $g_{r,t}$ is $g_{l,2\lceil\log(fin_s)\rceil+l-1}$ (It is because we substitute g_s by the corresponding circuit O_{fin_s} where fin_s = fan-in(g_s)).
-case 1.1.2: g_s is not an oracle gate. In this case the input gate of $g_{r,t}$ is $g_{0,s}$, that we will define as the replica gate of g_s.
-case 1.2: g_t is an output gate. As in the input gate case, *identity* is the type of $g_{r,t}$. If g_s is the input gate of g_t in O_{fin_r} then we define that $g_{r,s}$ is the input gate of $g_{r,t}$.
-case 1.3: g_t is an internal gate. We define $g_{r,t}$ as g_t, it has the same type, but we have to rename its inputs. For each input g_s of g_t in O_{fin_r}, we define $g_{r,s}$ the corresponding input of $g_{r,t}$. If g_s is the left (right) input of g_t then $g_{r,s}$ is the left (right) input of $g_{r,t}$; if g_s is an oracle gate and its lth output is the kth input of g_t then the lth output of $g_{r,s}$ is the kth input of $g_{r,t}$.
-case 2: g_r is not an oracle gate. Then S_r is undefined and the replica gate of g_r in the new circuit C'_n is $g_{0,r}$. We define $g_{0,r}$ as g_r, it has the same type. but its inputs have to be renamed. (See *cases* 1.1 and 1.3).

By the construction of C'_n is easy to see that $\{C'_n\}$ accepts the same language A since we only have substituted the oracle gates by the subcircuit which computes the oracle function with the appropiate fan-in. We also obtain that C'_n has a size bounded by $p(n)q(p(n))$, since each oracle gate has at most fan-in $p(n)$ and it is substituted by the circuit O_m where $0 \leq m \leq p(n)$. The depth of C'_n can be computed following Theorem 3.1 of [Wi 90] and it remains $O(\log^{i+j-1} n)$.

Now it only rests to prove that $\{C'_n\}$ is an ALOGTIME-*ECL*-uniform family of circuits. Since $\{C_n\}$ and $\{O_m\}$ are ALOGTIME-*ECL*-uniform, there exists two ALOGTIME-*ECL*-machines M_C and M_O that

accept ECL_C and ECL_O respectively. We define a machine $M_{C'}$ that guesses a decomposition of the input path p into subpaths and it simulates M_C and M_O in order to verify that these paths are correct. Important facts to be considered in the construction are the following:
(i) $M_{C'}$ uses nondeterminism to choose a decomposition of the original path.
(ii) $M_{C'}$ never guesses the oracle gate names. It is enough to guess a tuple $[fin, fout, in, out]$ for each subpath in a subcircuit which substitutes an oracle gate g, and where fin and $fout$ correspond to the fan-in and fan-out of g and in and out correspond to the number of input and output gates that are connected by the subpath. M' substitutes the subpaths corresponding to subcircuits that compute the oracle function g by these tuples and it verifies that the new path is a correct path in C_n. And $M_{C'}$ verifies that subpaths are correct in O_{fin} since given the tuple $[fin, fout, in, out]$ it can compute the input gate, g_{in}, the ouput gate $g_{2|q(fin)|+out-1}$, and by simulating M_O, it can verify that there is a path connecting these output and input gates. Otherwise, if M guesses the oracle gate names then it can be the case that it expends $O(\log^2 n)$ time. □

As mentioned before, the results concerning unbounded fan-in circuits can be obtained analogously. Hence the **Main theorem** follows from Lemma 5.1 for the left to right inclusion, and Lemmas 5.2 and 5.3 for the right to left inclusion. As a second corollary of Lemmas 5.1, 5.2, and 5.3 we obtain the following:

Theorem 5.4 *Let $\mathcal{F}$ be an arbitrary function class. Then.*
(i) $\mathrm{NC}^{i+j}(\mathcal{F}) = \mathrm{NC}^{i+1}(\mathrm{NC}^j(\mathcal{F}))$, *for* $i \geq 0$ *and* $j \geq 1$.
(ii) $\mathrm{AC}^{i+j}(\mathcal{F}) = \mathrm{AC}^i(\mathrm{AC}^j(\mathcal{F}))$, *for* $i, j \geq 0$.

Acknowledgement

We thank Ricard Gavalda and Jacobo Torán for many helpful comments.

References

[ABJ 91] C. Àlvarez, J.L. Balcázar and B. Jenner, Adaptive logspace reducibility and parallel time, to appear in *Mathematical Systems Theory*. (preliminary version in Proc. 8th STACS, Lecture Notes in Computer Science 480 (Springer, Berlin, 1991). pp. 422–433.)

[AJ 92] C. Àlvarez and B. Jenner, A Note on Log Space Optimization. Tech. Report LSI–92–30–R, Dept. LSI.

[AJ 93] C. Àlvarez and B. Jenner, On Dlogtime and Polylogtime Reductions, Tech. Report LSI–93, Dept. LSI, (to appear).

[Al 93] C. Àlvarez, Polylogtime and Logspace Adaptive Reductions, Tech. Report LSI–93, Dept. LSI, (to appear).

[Ba 91] J.L. Balcázar, Adaptive logspace and depth-bounded reducibilities, Proc. of the 6th Structure in Complexity Theory Conference (1991), pp. 240–254.

[BCDRT 89] A. Borodin, S.A. Cook, P.W. Dymond, W.L. Ruzzo, M. Tompa, Two applications of inductive counting for complementation problems, *SIAM Journal of Computing* **18**,3 (1989), pp. 559–578. And Erratum, *SIAM Journal of Computing* **18**,6 (1989).

[BCGR 89] S. Buss, S. Cook, A. Gupta, V Ramachandran, An optimal parallel algorithm for formula evaluation, typescript, Univ. of Toronto, 1989.

[BIS 90] D.A. Mix Barrington, N. Immerman, H. Straubing, On uniformity within NC^1, *J. of Computer and System Sciences* **41**,3 (1990), pp. 274–306.

[BLT 92] J.L. Balcázar, A. Lozano, J. Torán, The Complexity of Algorithmic Problems on Succinct Instances, R. Baeza-Yates, U. Manber (eds.), *Computer Science*, Plenum Press, New York, (1992), pp. 351–377.

[Bu 87] S.R. Buss, The formula value problem is in ALOGTIME, Proc. 19th ACM STOC Symp., 1987, pp. 123–131.

[CKS 81] A.K. Chandra, D. Kozen, L.J. Stockmeyer, Alternation, *Journal of the ACM* **28** (1981), pp. 114–133.

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

[CS 92] J. Castro, C. Seara, Characterizations of some complexity classes between Θ_2^p and Δ_2^p, Report LSI–90–27, Univ. Politècnica de Catalunya. Proc. 9th STACS, *Lecture Notes in Computer Science* **577** (Springer, Berlin, 1992), pp. 305–319.

[CSV 84] A.K. Chandra, L.J. Stockmeyer, U. Vishkin, Constant depth reducibility, SIAM Journal on Computing **13** (1984), pp. 423–439.

[JT 93] B. Jenner, J. Torán, Parallel queries to NP, Proc. of the 8th Structure in Complexity Theory Conference (1993), pp. 280–291.

[La 75] R. Ladner, The circuit value problem is log space complete for P, *SIGACT News* **7** (1975), pp. 18–20.

[Ru 81] W.L. Ruzzo, On uniform circuit complexity, *J. of Comput. and System Sci.* **22** (1981), pp. 365–383.

[Si 83] M. Sipser, Borel sets and circuit complexity, Proc. 15th ACM STOC Symp. (1983), pp. 61–69.

[Su 78] I.H. Sudborough, On the tape complexity of deterministic context-free languages, *Journal of the ACM* **25** (1978), pp. 405–414.

[St 77] L. Stockmeyer, The polynomial-time hierarchy, *Theoretical Computer Science* **3** (1977), pp 1–22.

[Wa 90] K.W. Wagner, Bounded Query Classes, SIAM Journal of Computing 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 Journal of Computing* 19,2 (1990), pp. 384–396. (preliminary version in 4th Structure in Complexity Theory Conference, 1989)