INFORMATION AND COMPUTATION 80, 269–288 (1989)

The Logarithmic Alternation Hierarchy Collapses: $\mathbf{A}\Sigma_2^{\mathcal{L}} = \mathbf{A}\Pi_2^{\mathcal{L}}$

BIRGIT JENNER AND BERND KIRSIG

Fachbereich Informatik, University of Hamburg, Rothenbaumch. 67/9, 2000 Hamburg 13, West Germany

AND

KLAUS-JÖRN LANGE

Institut für Informatik, TU München, Arcisstr. 21, 8000 Munich 2, West Germany

We show that $\mathbf{A}\Sigma_2^{\mathcal{L}} = \mathbf{A}\Sigma_k^{\mathcal{L}}$, $k \geqslant 2$, by proving that $\mathbf{A}\Sigma_2^{\mathcal{L}}$ coincides with $\mathbf{A}\Pi_2^{\mathcal{L}}$. Essentially this is done by reducing the $\mathbf{A}\Sigma_2^{\mathcal{L}}$-complete set $(\mathrm{GAP} \not\in \mathrm{CoGap})^{(3)}$ to the question whether of two vectors A and B of n components, A contains more "solvable" components, i.e., components which are contained in GAP, than B. Moreover, using a similar technique we show $\mathbf{A}\Sigma_2^{\mathcal{L}} = \mathbf{L}_{\mathbf{hd}}(\mathbf{NL})$. Finally, we consider the relevance of our proof technique for polynomial time classes, e.g., the Boolean **NP**-Hierarchy.

1. INTRODUCTION

As is well known the investigation of the "$\mathbf{P} = \mathbf{NP}$?-question" soon led to the construction of the polynomial hierarchy (Meyer and Stockmeyer, 1972; Stockmeyer, 1976). This hierarchy can be characterized by oracle Turing machines as well as by alternating Turing machines with a polynomial time bound. With the attempt to carry over these methods to the "$\mathbf{DSPACE(log\ n)} = \mathbf{NSPACE(log\ n)}$?-question" no problems turned up in defining the logarithmic alternation hierarchy by restricting the alternating Turing machines to a logarithmic space bound (Chandra, Kozen, and Stockmeyer, 1981). But a logarithmic oracle hierarchy was not so easy to define, since using the machine model of Ladner and Lynch (1976) one gets a hierarchy which essentially duplicates the polynomial hierarchy. Ruzzo, Simon, and Tompa (1984) restricted the nondeterministic oracle machines to make deterministic use of the oracle tape only, thus defining a hierarchy that is located between $\mathbf{DSPACE(log\ n)}$ and $\mathbf{P}$ as desired. But surprisingly this logarithmic oracle hierarchy—at least on first sight—did not seem to coincide with the logarithmic alternation hierarchy. At that

0890-5401/89 $3.00

stage it could only be concluded that the entire logarithmic alternation hierarchy is contained in the second deterministic level of the logarithmic oracle hierarchy (Ruzzo, Simon, and Tompa, 1984). Lange (1986) showed that with restricting nondeterministic machines which underlie the RuSiTo-condition to one oracle call only, one can exactly characterize the logarithmic alternation hierarchy (see also Rosier and Yen, 1986). Another characterization was found by bounded quantification of **DSPACE(log n)**-predicates, where the quantified words are given as one-way input (Lange, 1986).

The main result of this paper (which is an extended version of Lange, Jenner, and Kirsig, 1987) is the collapse of the logarithmic alternation hierarchy on its second level. Thus for the first time one of the hierarchies in complexity theory conjectured to be infinite could be proven to be finite. Our goal here is not only to state the proof but to make the paper self-contained by briefly sketching its development which will inevitably involve considering polynomial time classes, too.

After stating the preliminaries in Section 2 which consist mainly in an overview of our notation of previously known concepts, we introduce in Section 3 the notion of logarithmic space bounded disjunctive (resp. conjunctive) deterministic Turing reduction and show that this notion provides yet another characterization of the logarithmic alternation hierarchy. Furthermore, it also turns out that it builds up a hierarchy on the polynomial side, a "deterministic polynomial hierarchy" (see also Jenner, 1987), which includes the entire Boolean **NP**-Hierarchy in its second level but nevertheless does not seem to duplicate the polynomial hierarchy since it is contained in $\Delta_2^{\mathcal{P}}$, the second deterministic level of the polynomial hierarchy. Third, we show that the notion of disjunctive deterministic Turing reduction—appropriately restricted—even turns out to characterize the Boolean **NP**-hierarchy.

These characterizations enable us to show that the entire deterministic polynomial hierarchy is contained in $\mathbf{L}_{\mathbf{bf}}(\mathbf{NP})$ and that the entire logarithmic alternation hierarchy is contained in $\mathbf{L}_{\mathbf{bf}}(\mathbf{NL})$, the closure of **NP**, resp. **NL**, under logarithmic space bounded Boolean formula reductions, and that their second levels contain $\mathbf{L}_{\mathbf{bd}}(\mathbf{NP})$ and, resp., $\mathbf{L}_{\mathbf{bd}}(\mathbf{NL})$, i.e., the closure of **NP**, resp. **NL** under logarithmic space Hausdorff reductions, a rather restricted form of Boolean formula reduction. (Boolean formula and Hausdorff reductions are reductions in power "between" Turing and many–one which were studied in Wagner (1986a) with a polynomial time bound.) Wagner (1986a) was able to show that $\mathbf{P}_{\mathbf{bd}}(\mathbf{NP}) = \mathbf{P}_{\mathbf{bf}}(\mathbf{NP})$. We show that this implies the collapse of the deterministic polynomial hierarchy on its second level: $\mathbf{D}\Sigma_2^{\mathcal{P}} = \mathbf{D}\Pi_2^{\mathcal{P}} = \mathbf{D}\Sigma_k^{\mathcal{P}}$ for $k \geqslant 2$. But since the proof of Wagner's result makes essential use of the monotone circuit value problem, which is **P**-complete (Goldschlager, 1977) and still open whether

solvable in **NL**, the methods used in Wagner (1986a) do not yield $\mathbf{L}_{\mathbf{bf}}(\mathbf{NL}) \subseteq \mathbf{L}_{\mathbf{hd}}(\mathbf{NL})$. Thus for the logarithmic alternation hierarchy a collapse cannot be achieved analogously.

Now, one wonders if there is a simpler proof for the collapse of the deterministic polynomial hierarchy. Our goal will be to show in Section 4 that there is. This proof consists in a *direct* reduction of a $\mathbf{D}\Sigma_2^{\mathscr{P}}$-complete language to a language in $\mathbf{D}\Pi_2^{\mathscr{P}}$, which due to its simplicity and the structural similarity of the deterministic polynomial hierarchy and the logarithmic alternation hierarchy turns out to yield $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{A}\Pi_2^{\mathscr{L}}$, too.

In particular, we are able to show that the $\mathbf{A}\Sigma_2^{\mathscr{L}}$-complete set

$$(\mathrm{GAP} \not\in \mathrm{CoGAP})^{(\exists)}$$
$$:= \{G_1 \not\in G_1' \$ G_2 \not\in G_2' \$ \cdots \$ G_n \not\in G_n' \$ \mid n > 0,\ G_j, G_j' \in X^*,\ 1 \leqslant j \leqslant n,\ \not\in, \$ \notin X,$$
$$\text{and } \exists 1 \leqslant i \leqslant n\colon G_i \in \mathrm{GAP} \wedge G_i' \in \mathrm{CoGAP}\}$$

can be reduced in log space to a set that is contained in $\mathbf{A}\Pi_2^{\mathscr{L}}$. Furthermore, using the same technique we show $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{L}_{\mathbf{hd}}(\mathbf{NL})$. The gist of our proofs will be to consider counting sets of the type

$$\mathrm{COUNTA}$$
$$:= \{\langle F_1, F_2, \ldots, F_n; i\rangle \mid n \geqslant 0,\ i \geqslant 0,\ F_j \in X^*,\ 1 \leqslant j \leqslant n,\ |\{j \mid F_j \in A\}| \geqslant i\}.$$

By applying padding techniques we prove that $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{A}\Pi_2^{\mathscr{L}}$ can be generalized to alternating space hierarchies with arbitrary space constructible bounds $S(n) \geqslant \log(n)$. This includes the collapse of the alternating linear space hierarchy at its second level, shown independently by Toda (1987).

Finally, in Section 5 we investigate our proof technique when applied to the Boolean **NP**-hierarchy, which shows that the proof of $\mathbf{D}\Sigma_2^{\mathscr{P}} = \mathbf{D}\Pi_2^{\mathscr{P}}$ is in a certain sense optimal.

2. Preliminaries

The reader is assumed to be familiar with the basic notions and facts of complexity theory as they are contained in Hopcroft and Ullman (1979).

The cardinality of a set A is denoted by $|A|$. For a language A let CoA be its complement. (We do not fix the underlying alphabet, since its exchange needs intersections and unions with regular sets only, and these operations have no influence on the complexity of a problem.) For classes **A** and **B** of languages we define $\mathbf{CoA} := \{CoL \mid L \in \mathbf{A}\}$, $\mathbf{A} \wedge \mathbf{B} := \{L_1 \cap L_2 \mid L_1 \in \mathbf{A},\ L_2 \in \mathbf{B}\}$, and $\mathbf{A} \vee \mathbf{B} := \{L_1 \cup L_2 \mid L_1 \in \mathbf{A},$

$L_2 \in \mathbf{B}\}$. W.l.o.g. let $\not\in, \$, \#$ not be contained in any of the alphabets X, Y, Z. These symbols will be used as special marker symbols in order to separate words over X^*, Y^*, resp. Z^*.

In the following we set $\mathbf{DL} := \mathbf{DSPACE(log\ n)}$ and $\mathbf{NL} := \mathbf{NSPACE(log\ n)}$. Furthermore, let $\mathbf{LOG}(A)$ ($\mathbf{L}(A)$) denote the class of all sets log space many–one (log space Turing) reducible to an (oracle) set A. For a class $\mathbf{A}$ of languages we define: $\mathbf{LOG(A)} := \bigcup_{L \in \mathbf{A}} \mathbf{LOG}(L)$ and $\mathbf{L(A)} := \bigcup_{L \in \mathbf{A}} \mathbf{L}(L)$.

The *logarithmic alternation hierarchy* (according to Chandra, Kozen, and Stockmeyer, 1981) is defined as follows: $\forall k > 0$ let $\mathbf{A\Sigma_k^{\mathscr{L}}}$ denote the class of all languages that can be accepted by a log space bounded alternating Turing machine which starts in an existential state and which alternates in each computation at most $k-1$ times between existential and universal states. Furthermore let $\mathbf{A\Pi_k^{\mathscr{L}}} := \mathbf{CoA\Sigma_k^{\mathscr{L}}}$, and for the sake of completeness let $\mathbf{A\Sigma_0^{\mathscr{L}}} := \mathbf{DL}$.

The *Boolean* **NP**-*hierarchy* (see Wechsung, 1985; Cai and Hemachandra, 1986) is defined as the set $\{\mathbf{NP(k)}, \mathbf{CoNP(k)} \mid k \geqslant 0\}$, where $\mathbf{NP(0)} := \mathbf{CoNP(0)} := \mathbf{P}$ and $\mathbf{NP(k+1)} := \{L_1 \setminus L_2 \mid L_1 \in \mathbf{NP},\ L_2 \in \mathbf{NP(k)}\}$.

The *satisfiability problem* (after Cook, 1971) will be denoted by SAT. As is well known SAT is **NP**-complete with respect to log reductions.

The *graph accessibility problem* (after Jones, 1973; sometimes referred to as "graph reachability problem" (see Hopcroft and Ullman, 1979)) will be denoted by GAP. Although any other representation would work as well, let us assume that each directed graph G discussed has vertex set $V_n = \{1, 2, ..., n\}$ for some n. Its structure will be presented by its adjacency matrix M, which has "1" in entry (i, j) in case (i, j) is a directed edge of G and "0" otherwise. Let $\bar{G}$ denote the encoding of M. We define:

$$\mathrm{GAP} := \{\bar{G} \mid G \text{ is a directed graph on } V_n \text{ which has a path from vertex 1 to vertex } n\}.$$

As is well known GAP is **NL**-complete with respect to log reductions (Savitch, 1970; Jones, 1973). In the following we will, for simplicity, not distinguish between G and $\bar{G}$.

Let c_A be the characteristic function of A. A set $A \subseteq X^*$ is *Boolean formula reducible* to a set $B \subseteq Y^*$ in logarithmic space (resp. polynomial time) ($A \leqslant_{bf}^{\mathscr{L}} B$ (resp. $A \leqslant_{bf}^{\mathscr{P}} B$)) iff there exists a log space (resp. pol time) computable function f, $f\colon X^* \to Z^* \#(Y^*\#)^*$ with $\forall x \in X^*\colon f(x) = z \# y_1 \# y_2 \# \cdots \# y_n \#$ (note that n and z depend on x) such that $x \in A \Leftrightarrow h_z(c_B(y_1), c_B(y_2), ..., c_B(y_n)) = 1$, with z an encoding of a Boolean formula in $\wedge$, $\vee$, $\neg$, h_z the Boolean function represented by z, and $y_i \in Y^*$, $1 \leqslant i \leqslant n$. Let $\mathbf{L_{bf}(A)}$ (resp. $\mathbf{P_{bf}(A)}$) denote the closure of a class of languages $\mathbf{A}$ under $\leqslant_{bf}^{\mathscr{L}}$ (resp. $\leqslant_{bf}^{\mathscr{P}}$).

The notion of Hausdorff reducibility ($\leqslant_{hd}$) was originally introduced in Wagner (1986a) as a restriction of polynomial time Boolean formula reducibility ($\leqslant_{bf}$) and then examined for logarithmic space in Kirsig (1986).

A set $A \subseteq X^*$ is *Hausdorff reducible* to a set $B \subseteq Y^*$ in logarithmic space (resp. polynomial time) ($A \leqslant_{hd}^{\mathscr{L}} B$ (resp. $A \leqslant_{hd}^{\mathscr{P}} B$)) iff there exists a log space (resp. pol time) computable function f, $f: X^* \to (Y^* \#)^*$ with $\forall x \in X^*: f(x) = y_1 \# y_2 \# \cdots \# y_{2n-1} \# y_{2n} \#$ (note that n depends on x) such that $y_i \in Y^*$, $1 \leqslant i \leqslant 2n$, and

(1) $x \in A \Leftrightarrow \exists 1 \leqslant i \leqslant n: y_{2i-1} \in B \wedge \neg (y_{2i} \in B)$, and

(2) $\forall 1 \leqslant i < 2n: y_{i+1} \in B \Rightarrow y_i \in B$.

Let $\mathbf{L_{hd}(A)}$ (resp. $\mathbf{P_{hd}(A)}$) denote the closure of a class of languages $\mathbf{A}$ under $\leqslant_{hd}^{\mathscr{L}}$ (resp. $\leqslant_{hd}^{\mathscr{P}}$). Note that $\mathbf{L_{hd}(A)}$ is closed under log reductions.

3. Disjunctive (and Conjunctive) Deterministic Turing Reductions

In Ladner, Lynch, and Selman (1975) and Ladner and Lynch (1976) the concept of various reductions in power between many-one and Turing, i.e., truth-table reductions, Boolean formula reductions, disjunctive and conjunctive reductions were introduced and investigated. The latter reductions can be understood as realized by certain restricted deterministic oracle Turing machines, namely disjunctive and conjunctive oracle Turing machines. These concepts now turn out to yield

(1) a new characterization of the logarithmic alternation hierarchy

(2) a new "hierarchy" on the polynomial side, which contains the Boolean **NP**-hierarchy in its second level and is itself contained in the second level of the polynomial hierarchy

and (appropriately restricted)

(3) a new characterization of the Boolean **NP**-hierarchy.

We say that an oracle machine works *disjunctively* if it accepts an input immediately after the first positive (i.e., "YES") oracle answer (or *conjunctively* if it rejects an input immediately after the first negative (i.e., "NO") oracle answer). Furthermore, we say that an oracle Turing machine is *n-query bounded* if it accepts an input with at most n queries to its oracle. Combining these two restrictions of oracle machines we get n-query bounded Turing machines which work disjunctively (resp. conjunctively). With this we define:

Definition 3.1. *For languages A, B let $A \leqslant_d^{\mathscr{L}} B$ (resp. $A \leqslant_c^{\mathscr{L}} B$) iff there is a log space bounded disjunctive (resp. conjunctive) oracle Turing*

machine M such that M accepts A with oracle set B. For $k \geqslant 0$ let $A \leqslant_{d_k}^{\mathscr{L}} B$ (resp. $A \leqslant_{c_k}^{\mathscr{L}} B$) iff there is a log space bounded k-query bounded oracle machine M which works disjunctively (resp. conjunctively) and accepts A with oracle set B. Let $\mathbf{L}_{\mathbf{d}}(A) := \{L \mid L \leqslant_d^{\mathscr{L}} A\}$, $\mathbf{L}_{\mathbf{d}_k}(A) := \{L \mid L \leqslant_{d_k}^{\mathscr{L}} A\}$ and $\mathbf{L}_{\mathbf{d}}(\mathbf{A}) := \{L \mid \exists A \in \mathbf{A}: L \leqslant_d^{\mathscr{L}} A\}$, $\mathbf{L}_{\mathbf{d}_k}(\mathbf{A}) := \{L \mid \exists A \in \mathbf{A}: L \leqslant_{d_k}^{\mathscr{L}} A\}$ for a language A and a class of languages $\mathbf{A}$, and let $\mathbf{L}_{\mathbf{c}}(A)$, $\mathbf{L}_{\mathbf{c}_k}(A)$, $\mathbf{L}_{\mathbf{c}}(\mathbf{A})$, and $\mathbf{L}_{\mathbf{c}_k}(\mathbf{A})$ be similarly defined.

We first state some basic properties of $\leqslant_d^{\mathscr{L}}$ and $\leqslant_{d_k}^{\mathscr{L}}$.

LEMMA 3.2. *Let $A \subseteq X^*$, and $k \geqslant 1$. Then it holds:*

(i) $\mathbf{L}_{\mathbf{d}}(\mathbf{LOG}(A)) = \mathbf{LOG}(\mathbf{L}_{\mathbf{d}}(A)) = \mathbf{L}_{\mathbf{d}}(A)$,

(i′) $\mathbf{L}_{\mathbf{d}_k}(\mathbf{LOG}(A)) = \mathbf{LOG}(\mathbf{L}_{\mathbf{d}_k}(A)) = \mathbf{L}_{\mathbf{d}_k}(A)$,

(ii) $\mathbf{L}_{\mathbf{d}}(A) = \mathbf{CoL}_{\mathbf{c}}(CoA)$,

(ii′) $\mathbf{L}_{\mathbf{d}_k}(A) = \mathbf{CoL}_{\mathbf{c}_k}(CoA)$,

and for $A \neq \varnothing$:

(iii) $\mathbf{L}_{\mathbf{d}}(A) = \mathbf{LOG}((X^*\$)^* A\$(X^*\$)^*)$,

(iii′) $\mathbf{L}_{\mathbf{d}_k}(A) = \mathbf{LOG}(\bigcup_{i=0}^{k-1} (X^*\$)^i A\$(X^*\$)^{k-i-1})$.

Proof. (i) and (i′) are obvious. We first state the proof of (ii). The proof of (ii′) follows analogously.

(ii) If $A = \varnothing$ then $\mathbf{L}_{\mathbf{d}}(A) = \mathbf{DL} = \mathbf{CoDL} = \mathbf{CoL}_{\mathbf{c}}(X^*) = \mathbf{CoL}_{\mathbf{c}}(CoA)$ for some alphabet X. Now assume $A \neq \varnothing$. Let $L \in \mathbf{L}_{\mathbf{d}}(A)$ and M a disjunctive oracle Turing machine accepting L. W.l.o.g. all $w \in L$ are accepted by a positive oracle answer. Now, obviously $w \notin L$ if and only if all of M's oracle queries are answered with "NO". Take a new machine N that behaves as M but whose accepting states are the rejecting states of M. Give N CoA as oracle set. Then we have: M rejects w if and only if all queries of M are answered with "NO" if and only if all queries of N are answered with "YES" if and only if N accepts w. Obviously, N works conjunctively and accepts CoL. Hence, $CoL \in \mathbf{L}_{\mathbf{c}}(CoA)$ and $L \in \mathbf{CoL}_{\mathbf{c}}(CoA)$. Thus $\mathbf{L}_{\mathbf{d}}(A) \subseteq \mathbf{CoL}_{\mathbf{c}}(CoA)$. $\mathbf{CoL}_{\mathbf{c}}(CoA) \subseteq \mathbf{L}_{\mathbf{d}}(A)$ can be proven analogously.

(iii) "$\subseteq$": Let $L \in \mathbf{L}_{\mathbf{d}}(A)$ be accepted by the log space bounded disjunctive OTM M with oracle set A. As M is log space bounded it can be simulated by a log space transducer, a machine M' with the same space bound and an output tape instead of the oracle tape. Let M' behave as if all oracle queries which are posed by M are answered negatively and instead of sending a query to the oracle M' prints the query word followed by a \$ onto its output tape in order to separate the query words. If M has

queried the oracle with the words $x_1, x_2, ..., x_k$ M''s output will be $x_1\$x_2\$ \cdots \$x_k\$$ and M is correctly simulated until this point (since as M works disjunctively none of the queries x_1 to x_{k-1} were answered positively). Now assume that $x_k \in A$ causes M to halt and accept, but M' behaves as if the oracle answere is "NO" and continues its computation. Nevertheless the total output of M' will be $x_1\$x_2 \cdots \$x_k\$ \cdots \$x_r\$ \in (X^*\$)^* A\$(X^*\$)$. As M' performs a log reduction, we get $L \in \mathbf{LOG}((X^*\$)^* A\$(X^*\$)^*)$.

"$\supseteq$": As is easily seen a log space bounded disjunctive oracle machine M with oracle set A accepts $(X^*\$)^* A\$(X^*\$)^*$ as follows: On input $w_1\$w_2\$ \cdots \$w_n\$$, $w_i \in X^*$, $1 \leqslant i \leqslant n$, M can copy each w_i onto its oracle tape and decide whether $w_i \in A$ or not using its oracle mechanism and accept iff one of the queries is answered positively. Thus it holds: $(X^*\$)^* A\$(X^*\$)^* \in \mathbf{L}_{\mathbf{d}}(A)$. The claim (iii) follows with (i). (iii') follows by similar arguments as (iii). ▮

Consequently, we see that for any language $A \subseteq X^*$:

$$A^{(\exists)} := (X^*\$)^* A\$(X^*\$)^* \text{ is a } \mathbf{L}_{\mathbf{d}}(A)\text{-complete language}$$

and

$$A^{(\exists_k)} := \bigcup_{i=0}^{k-1} (X^*\$)^i A\$(X^*\$)^{k-i-1} \text{ is a } \mathbf{L}_{\mathbf{d_k}}(A)\text{-complete language, for } k \geqslant 1.$$

And because of (ii) and (ii') we obviously have for any non-empty language $A \subseteq X^*$:

$$A^{(\forall)} := (A\$)^* \text{ is complete for } \mathbf{L}_{\mathbf{c}}(A)$$

and

$$A^{(\forall_k)} := (A\$)^k \text{ is complete for } \mathbf{L}_{\mathbf{c_k}}(A),\ k \geqslant 1.$$

Our first goal will now be to show that with the notion of disjunctive (and conjunctive) deterministic Turing reduction a new characterization of the logarithmic alternation hierarchy can be obtained (see also Kirsig, 1986). This is the content of the following theorem.

THEOREM 3.3.

(i) $\mathbf{A}\boldsymbol{\Sigma}_{\mathbf{2}}^{\mathscr{L}} = \mathbf{L}_{\mathbf{d}}(\mathbf{NL} \wedge \mathbf{CoNL})$

(ii) $\mathbf{A}\boldsymbol{\Sigma}_{\mathbf{k+1}}^{\mathscr{L}} = \mathbf{L}_{\mathbf{d}}(\mathbf{A}\boldsymbol{\Pi}_{\mathbf{k}}^{\mathscr{L}})$ *for all* $k \geqslant 2$.

Proof. (i) "$\subseteq$": Let $L \in \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{2}}^{\mathscr{L}}$ and M be a log space bounded alter-

nating Turing machine which accepts L alternating exactly once. Now, define:

$A := \{w\not\in K_i \mid M$ can reach configuration K_i in a computation on w without alternation$\}$,

$B := \{w\not\in K_i \mid K_i$ is a universal configuration and all computations of M on w starting in K_i halt accepting$\}$.

Obviously, $A \in \mathbf{NL}$ and $B \in \mathbf{CoNL}$; and, of course, we now have:

$$w \in L \Leftrightarrow \exists K_i, K_j\colon w\not\in K_i \in A \wedge w\not\in K_j \in B \wedge M \text{ can reach } K_j \text{ in one step from } K_i \text{ on input } w. \qquad (*)$$

Now the deterministic log space bounded OTM N with oracle set $A\$B$ accepts L as follows: N computes on input w all existential configurations of M (one by one, say, in lexicographical order) and checks for each of them if M is able to reach an universal configuration in one step. Let $K_j, \ldots, K_{j+t_i}$ be the one-step-successor (universal) configurations of K_i (existential). N queries the oracle with $w\not\in K_i\$w\not\in K_j, \ldots, w\not\in K_i\$w\not\in K_{j+t_i}$ (one after the other) and accepts if one of the queries is answered positively. If all answers are negative or K_i has no universal successors N computes the (lexicographically) next existential configuration K_{i+1} and repeats the process. If all existential configurations are checked without a positive oracle answer N rejects. Thus because of $(*)$ N accepts if and only if $w \in L$. Hence, $L \in \mathbf{L_d}(A\$B)$. But since $A\$B = A\$X^* \cap X^*\$B$, we have: $A\$B \in \mathbf{NL} \wedge \mathbf{CoNL}$. Thus, $L \in \mathbf{L_d}(\mathbf{NL} \wedge \mathbf{CoNL})$ and as L was arbitrarily chosen we get $\mathbf{A\Sigma_2^{\mathscr{L}}} \subseteq \mathbf{L_d}(\mathbf{NL} \wedge \mathbf{CoNL})$.

"$\supseteq$": Let $L \in \mathbf{L_d}(\mathbf{NL} \wedge \mathbf{CoNL})$. Thus L is accepted by a disjunctive log space bounded OTM M with oracle set $A \cap B \subseteq X^*$, $A \in \mathbf{NL}$, $B \in \mathbf{CoNL}$. By Lemma 3.2(iii) we have: $L \leqslant_{\log} (X^*\$)^* (A \cap B)\$(X^*\$)^*$. But obviously $(X^*\$)^* (A \cap B)\$(X^*\$)^*$ can be recognized by an $A\Sigma_2^{\mathscr{L}}$-machine N as follows. On input w N checks $w \in (X^*\$)^+$ and rejects if the input is not of this form. If $w = w_1\$w_2\$\cdots\$w_n\$$, $w_i \in X^*$, $1 \leqslant i \leqslant n$, N guesses nondeterministically $i \leqslant n$ and checks $w_i \in A$. If N does not succeed in verifying $w_i \in A$ the input is rejected. Otherwise N alternates and checks in universal states $w_i \in B$, accepting if $w_i \in B$ and rejecting if not. Thus N accepts the input w if and only if $w = w_1\$\cdots\$w_n\$$, $w_i \in X^*$ and there is an $i \leqslant n$ such that $w_i \in A \cap B$, which is: $w \in (X^*\$)^* (A \cap B)\$(X^*\$)^*$. As $\mathbf{A\Sigma_2^{\mathscr{L}}}$ is closed under logarithmic space reductions $L \in \mathbf{A\Sigma_2^{\mathscr{L}}}$ follows.

(ii) "$\subseteq$": Let $L \in \mathbf{A\Sigma_{k+1}^{\mathscr{L}}}$, $k \geqslant 2$, and M be the $A\Sigma_{k+1}^{\mathscr{L}}$-machine which accepts L by using exactly k alternations. Following the argumentation in the proof of (i) we obtain $L \in \mathbf{L_d}(A\$B')$, where A is as above (in the proof of (i)) and B' is defined as

$B' := \{w\not\in K_i \mid K_i$ is a universal configuration and M started in K_i accepts $w\}$.

We have $A \in \mathbf{NL}$ and obviously $B' \in \mathbf{A}\Pi_k^{\mathscr{L}}$. Since for all $k \geqslant 0$ $\mathbf{A}\Pi_k^{\mathscr{L}}$ is obviously closed under marked concatenation (i.e., for $L_1, L_2 \in \mathbf{A}\Pi_k^{\mathscr{L}}$ it holds $L_1 \$ L_2 \in \mathbf{A}\Pi_k^{\mathscr{L}}$), we have $A\$B' \in \mathbf{A}\Pi_k^{\mathscr{L}}$, which implies $L \in \mathbf{L}_d(\mathbf{A}\Pi_k^{\mathscr{L}})$ and we get $\mathbf{A}\Sigma_{k+1}^{\mathscr{L}} \subseteq \mathbf{L}_d(\mathbf{A}\Pi_k^{\mathscr{L}})$.

"$\supseteq$": If $L \in \mathbf{L}_d(\mathbf{A}\Pi_k^{\mathscr{L}})$ then there is a set $A \in \mathbf{A}\Pi_k^{\mathscr{L}}$, $A \subseteq X^*$, such that $L \in \mathbf{L}_d(A)$. But $L \in \mathbf{L}_d(A)$ implies $L \leqslant_{\log} (X^*\$)^* A\$(X^*\$)^*$ and $(X^*\$)^* A\$(X^*\$)^* \in \mathbf{A}\Sigma_{k+1}^{\mathscr{L}}$ is obvious. Hence $\mathbf{L}_d(\mathbf{A}\Pi_k^{\mathscr{L}}) \subseteq \mathbf{A}\Sigma_{k+1}^{\mathscr{L}}$. ∎

Remark. Note that there is a close connection between $\mathbf{L}_d(\cdot)$ and nondeterministic logarithmic space reductions $\mathbf{NLOG}\langle \cdot \rangle$ which were studied in Lange (1986). There it was shown that $\mathbf{NLOG}\langle A \rangle = \mathbf{LOG}((\mathrm{GAP}_{\notin A})^{(\exists)})$. Hence we have the following alternative characterization of the logarithmic alternation hierarchy: $\mathbf{A}\Sigma_{k+1}^{\mathscr{L}} = \mathbf{NLOG}\langle \mathbf{A}\Pi_k^{\mathscr{L}} \rangle$ for all $k \geqslant 1$.

Since $\mathbf{NL} \wedge \mathbf{CoNL} = \mathbf{LOG}((\mathrm{GAP}_{\notin \mathrm{CoGAP}}))$ we now immediately get with Lemma 3.2(iii):

COROLLARY 3.4. $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{LOG}((\mathrm{GAP}_{\notin \mathrm{CoGAP}})^{(\exists)})$.

Having obtained a new characterization of the logarithmic alternation hierarchy by deterministic Turing reductions we now show that with this notion a hierarchy on the polynomial side can be defined which does not seem to duplicate the polynomial hierarchy.

DEFINITION 3.5. *The deterministic polynomial hierarchy is the set* $\{\mathbf{D}\Sigma_k^{\mathscr{P}}, \mathbf{D}\Pi_k^{\mathscr{P}} \mid k \geqslant 0\}$, *where*

$$\mathbf{D}\Sigma_0^{\mathscr{P}} := \mathbf{D}\Pi_0^{\mathscr{P}} := \mathbf{LOG}(0 \cdot \mathrm{SAT} \cup 1 \cdot \mathrm{CoSAT}), \text{ and for all } k \geqslant 0\text{:}$$

$$\mathbf{D}\Sigma_{k+1}^{\mathscr{P}} := \mathbf{L}_d(\mathbf{D}\Pi_k^{\mathscr{P}}),$$

$$\mathbf{D}\Pi_{k+1}^{\mathscr{P}} := \mathbf{CoD}\Sigma_{k+1}^{\mathscr{P}}.$$

Remark. Note that the same hierarchy is built up if disjunctive *polynomial time* bounded reductions are used instead of disjunctive log space bounded reductions since it is easy to show that $\mathbf{L}_d(\mathbf{POL}(A))$ is equal to $\mathbf{P}_d(A)$ for any non-empty language A.

Obviously, it holds $\bigcup_{k=0}^{\infty} \mathbf{D}\Sigma_k^{\mathscr{P}} \subseteq \Delta_2^{\mathscr{P}}$, i.e., the whole deterministic polynomial hierarchy is contained in $\Delta_2^{\mathscr{P}}$, the second level of the polynomial hierarchy. Furthermore, with the following theorem it will be immediate that the second level of the deterministic polynomial hierarchy $\mathbf{D}\Sigma_2^{\mathscr{P}}$ contains the whole Boolean **NP**-hierarchy since it is just its ω-jump.

For this (and the following section) notice that $\mathbf{D}\Sigma_1^{\mathscr{P}} = \mathbf{NP} \vee \mathbf{CoNP} = \mathbf{CoD}^{\mathbf{P}}$, and thus $\mathbf{D}\Pi_1^{\mathscr{P}} = \mathbf{NP} \wedge \mathbf{CoNP} = \mathbf{D}^{\mathbf{P}}$ (see Papadimitriou and

Yannakakis, 1984) and $\mathbf{D}\Sigma_2^{\mathscr{P}} = \mathbf{L_d}(\mathbf{NP} \wedge \mathbf{CoNP})$. Therefore Lemma 3.2(iii) yields:

LEMMA 3.6. $\mathbf{D}\Sigma_2^{\mathscr{P}} = \mathbf{L_d}(\mathbf{NP} \wedge \mathbf{CoNP}) = \mathbf{LOG}((\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists)})$. ∎

THEOREM 3.7. $\mathbf{NP(2k)} = \mathbf{L_{d_k}}(\mathbf{NP} \wedge \mathbf{CoNP})$ *for all* $k \geqslant 1$.

Proof. "$\subseteq$": Let $L \in \mathbf{NP(2k)}$, $k \geqslant 1$. Obviously, $L \in \mathbf{NP(2k)}$ iff there are $A_1, ..., A_k, A_i \in \mathbf{NP}$ and $B_1, ..., B_k, B_i \in \mathbf{CoNP}$, $1 \leqslant i \leqslant k$, such that $L = \bigcup_{i=1}^{k} (A_i \cap B_i)$. Let w.l.o.g. $A_i, B_i \subseteq X^*$. Now there are log reductions $f_1, ..., f_k$ and $g_1, ..., g_k$ with $f_i, g_i: X^* \to Y^*$, $1 \leqslant i \leqslant k$, such that for all $w \in X^*$: $w \in A_i \Leftrightarrow f_i(w) \in \mathrm{SAT}$ and $w \in B_i \Leftrightarrow g_i(w) \in \mathrm{CoSAT}$, which gives us

$$\begin{aligned} w \in L &\Leftrightarrow w \in \bigcup_{i=1}^{k} (A_i \cap B_i) \\ &\Leftrightarrow \exists 1 \leqslant i \leqslant k: w \in A_i \wedge w \in B_i \\ &\Leftrightarrow \exists 1 \leqslant i \leqslant k: f_i(w) \in \mathrm{SAT} \wedge g_i(w) \in \mathrm{CoSAT} \\ &\Leftrightarrow h(w) := f_1(w) \not\in g_1(w)\$ \cdots \$ f_k(w) \not\in g_k(w)\$ \in (\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists_k)}, \end{aligned}$$

where $h: X^* \to (Y^* \not\in Y^*\$)^*$ is obviously a log reduction. Thus $L \in \mathbf{LOG}((\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists_k)})$ and with Lemma 3.2(iii′) we have $\mathbf{NP(2k)} \subseteq \mathbf{L_{d_k}}(\mathbf{NP} \wedge \mathbf{CoNP})$, since $\mathrm{SAT}\not\in\mathrm{CoSAT}$ is $\mathbf{D^P} = (\mathbf{NP} \wedge \mathbf{CoNP})$-complete w.r.t. log reductions (Papadimitriou and Yannakakis, 1984).

"$\supseteq$": Because of Lemma 3.2(iii′) and since all the levels of the Boolean **NP**-hierarchy are closed under log reductions it suffices to show that $(\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists_k)} \in \mathbf{NP(2k)}$ for $k \geqslant 1$. But as

$$\begin{aligned} &(\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists_k)} \\ &\quad := \{F_1 \not\in F_1'\$ \cdots \$ F_k \not\in F_k'\$ \mid F_j, F_j' \in X^*,\ 1 \leqslant j \leqslant k, \\ &\qquad \exists 1 \leqslant i \leqslant k: F_i \in \mathrm{SAT} \wedge F_i' \in \mathrm{CoSAT}\}, \end{aligned}$$

we obviously have

$$\begin{aligned} &(\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists_k)} \\ &\quad = \bigcup_{i=0}^{k-1} (X^* \not\in X^*\$)^i\, \mathrm{SAT}\not\in\mathrm{CoSAT}\$(X^* \not\in X^*\$)^{k-i-1} \\ &\quad = \bigcup_{i=0}^{k-1} ((X^* \not\in X^*\$)^i\, \mathrm{SAT}\not\in X^*\$(X^* \not\in X^*\$)^{k-i-1} \\ &\qquad \cap (X^* \not\in X^*\$)^i\, X^* \not\in \mathrm{CoSAT}\$(X^* \not\in X^*\$)^{k-i-1}) \\ &\quad = \bigcup_{i=1}^{k} (A_i \cap B_i), \end{aligned}$$

where $A_{i+1} := (X^*\not\in X^*\$)^i\,\mathrm{SAT}\not\in X^*\$(X^*\not\in X^*\$)^{k-i-1}$ and $B_{i+1} := (X^*\not\in X^*\$)^i\,X^*\not\in\mathrm{CoSAT}\$(X^*\not\in X^*\$)^{k-i-1}$ for $0 \leqslant i \leqslant k-1$. Obviously, $A_i \in \mathbf{NP}$ and $B_i \in \mathbf{CoNP}$ for all $1 \leqslant i \leqslant k$, and hence $(\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists k)} \in \mathbf{NP(k)}$. ▮

COROLLARY 3.8. $\mathbf{NP(k)} = \mathbf{LOG}((\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists k)})$ *for all* $k \geqslant 1$.

Since $(\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists)} = \bigcup_{k=1}^{\infty} (\mathrm{SAT}\not\in\mathrm{CoSAT})^{(\exists k)}$ it is now immediate that the whole Boolean **NP**-hierarchy is contained in $\mathbf{D}\Sigma_2^{\mathscr{P}}$.

With the following theorem we show that on the polynomial side the deterministic polynomial hierarchy fulfills similar inclusion relationships as the logarithmic alternation hierarchy on the logarithmic side.

THEOREM 3.9. *For all* $k \geqslant 2$ *it holds*:

(i) $\mathbf{L_{hd}(NP)} \subseteq \mathbf{D}\Sigma_2^{\mathscr{P}} \cap \mathbf{D}\Pi_2^{\mathscr{P}} \subseteq \mathbf{D}\Sigma_k^{\mathscr{P}} \subseteq \mathbf{L_{bf}(NP)}$

(ii) $\mathbf{L_{hd}(NL)} \subseteq \mathbf{A}\Sigma_2^{\mathscr{L}} \cap \mathbf{A}\Pi_2^{\mathscr{L}} \subseteq \mathbf{A}\Sigma_k^{\mathscr{L}} \subseteq \mathbf{L_{bf}(NL)}$.

Proof. We prove (ii). The proof of (i) follows by similar arguments with Lemma 3.6.

(ii) First we show $\mathbf{L_{hd}(NL)} \subseteq \mathbf{LOG}((\mathrm{GAP}\not\in\mathrm{CoGAP})^{(\exists)})$. Let $L \subseteq X^*$, $L \in \mathbf{L_{hd}(NL)}$. Then there exists a function f, $f: X^* \to (Y^*\#)^*$, and a set $B \subseteq Y^*$, $B \in \mathbf{NL}$ such that $f(x) = y_1 \# y_2 \# \cdots \# y_{2n-1} \# y_{2n} \#$ and $x \in L \Leftrightarrow \exists 1 \leqslant i \leqslant n\colon y_{2i-1} \in B \wedge \neg(y_{2i} \in B)$. Since $B \in \mathbf{NL}$ and GAP is **NL**-complete w.r.t. log reductions, there exists a (many–one) log reduction $h\colon Y^* \to Z^*$ such that $\forall y \in Y^*\colon y \in B \Leftrightarrow h(y) \in \mathrm{GAP}$. Now define a function g, $g\colon X^* \to (Z \cup \{\not\in, \$\})^*$ with

$$g(x) := h(y_1)\not\in h(y_2)\$h(y_3)\not\in h(y_4)\$ \cdots \$h(y_{2n-1})\not\in h(y_{2n})\$ \qquad \text{for all} \quad x \in X^*.$$

g is log space computable since f and h are, and we obtain

$$\begin{aligned} x \in L &\Leftrightarrow \exists 1 \leqslant i \leqslant n\colon h(y_{2i-1}) \in \mathrm{GAP} \wedge \neg(h(y_{2i}) \in \mathrm{GAP}) \\ &\Leftrightarrow g(x) \in (\mathrm{GAP}\not\in\mathrm{CoGAP})^{(\exists)}. \end{aligned}$$

Since L was arbitrarily chosen $\mathbf{L_{hd}(NL)} \subseteq \mathbf{LOG}((\mathrm{GAP}\not\in\mathrm{CoGAP})^{(\exists)})$ follows. By Corollary 3.4 and since $\mathbf{L_{hd}(NL)}$ is closed under complement (see Kirsig, 1986; Wagner, 1986a) we get $\mathbf{L_{hd}(NL)} \subseteq \mathbf{A}\Sigma_2^{\mathscr{L}} \cap \mathbf{A}\Pi_2^{\mathscr{L}}$, and since obviously $\mathbf{L_{bf}(NL)}$ is closed under $\leqslant_d^{\mathscr{L}}$ and complementation we also have with Theorem 3.3: $\mathbf{A}\Sigma_k^{\mathscr{L}} \subseteq \mathbf{L_{bf}(NL)}$, for all $k \geqslant 0$, i.e., the entire logarithmic alternation hierarchy is contained in $\mathbf{L_{bf}(NL)}$. ▮

Having shown that the logarithmic alternation hierarchy and the deterministic polynomial hierarchy not only are similarly structured but also fulfill similar inclusion relationships, we know by the following result of

Wagner (1986a) that the deterministic polynomial hierarchy collapses at its second level.

THEOREM 3.10 (Wagner, 1986a). $\mathbf{P}_{hd}(\mathbf{NP}) = \mathbf{P}_{bf}(\mathbf{NP})$.

Since $\mathbf{P}_{hd}(A) = \mathbf{L}_{hd}(\mathbf{POL}(A))$ for any language A, and hence $\mathbf{P}_{bf}(\mathbf{NP}) = \mathbf{L}_{bf}(\mathbf{NP})$ the collapse of the deterministic polynomial hierarchy follows:

$$\mathbf{D}\Sigma_2^{\mathscr{P}} = \mathbf{D}\Pi_2^{\mathscr{P}} = \mathbf{D}\Sigma_k^{\mathscr{P}} \quad \text{for all} \quad k \geqslant 2. \tag{**}$$

But for the logarithmic alternation hierarchy a collapse cannot be achieved analogously, since the methods used in Wagner (1986a) essentially involve the evaluation of monotone circuits, which is a **P**-complete problem (Goldschlager, 1977) and therefore do not yield $\mathbf{L}_{bf}(\mathbf{NL}) \subseteq \mathbf{L}_{hd}(\mathbf{NL})$. Thus one wonders if there is a simpler proof of (**). We show in the following section that there is.

4. THE COLLAPSE OF THE LOGARITHMIC ALTERNATION HIERARCHY

A simpler proof for $\mathbf{D}\Sigma_2^{\mathscr{P}} = \mathbf{D}\Pi_2^{\mathscr{P}}$ (and thus (**)) can be obtained by constructing a reduction from $(\mathrm{SAT} \not\in \mathrm{CoSAT})^{(\exists)}$ (which is $\mathbf{D}\Sigma_2^{\mathscr{L}}$-complete w.r.t. log reductions (Lemma 3.6)) to a language in $\mathbf{D}\Pi_2^{\mathscr{P}}$. Due to the structural similarity of the deterministic polynomial hierarchy and the logarithmic alternation hierarchy a similar reduction can be constructed from $(\mathrm{GAP} \not\in \mathrm{CoGAP})^{(\exists)}$ (which is $\mathbf{A}\Sigma_2^{\mathscr{L}}$-complete (Corollary 3.4)) to $\mathbf{A}\Pi_2^{\mathscr{L}}$.

For this we shall construct sets of the type COUNTA. A word $\langle w_1, w_2, \ldots, w_n; j \rangle$ is an element of COUNTA iff at least j of the w_i's are elements of A. Similar constructions occur in Köbler, Schöning, and Wagner (1985) and Wagner (1986a) in connection with the Boolean **NP**-hierarchy and in Wagner (1986b) in connection with the counting polynomial hierarchy. We define:

$$\begin{aligned} \mathrm{COUNTSAT} := \{ \langle F_1, F_2, \ldots, F_n; i \rangle \mid n > 0,\ i \geqslant 0,\ F_j \in X^*, \\ 1 \leqslant j \leqslant n,\ |\{j \mid F_j \in \mathrm{SAT}\}| \geqslant i \}, \\ \mathrm{COUNTGAP} := \{ \langle G_1, G_2, \ldots, G_n; i \rangle \mid n > 0,\ i \geqslant 0,\ G_j \in X^*, \\ 1 \leqslant j \leqslant n,\ |\{j \mid G_j \in \mathrm{GAP}\}| \geqslant i \}. \end{aligned}$$

Obviously, $\mathrm{COUNTSAT} \in \mathbf{NP}$ and $\mathrm{COUNTGAP} \in \mathbf{NL}$. The construction of COUNTSAT (resp. COUNTGAP), enables us to count how many components of a vector are (at least) contained in SAT (resp. GAP), and thus to compare two vectors.

Using the notion of the conjunction $F \wedge F'$ of two formulae F, F', the basic idea of the new simple proof for $\mathbf{D}\Sigma_2^{\mathscr{P}} = \mathbf{D}\Pi_2^{\mathscr{P}}$ can be stated as:

$F_1 \not\in F_1' \$ F_2 \not\in F_2' \$ \cdots \$ F_n \not\in F_n' \$ \in (\mathrm{SAT} \not\in \mathrm{CoSAT})^{(\exists)}$ if and only if the vector $\langle F_1, F_2, ..., F_n \rangle$ contains more components which are contained in SAT than the vector $\langle F_1 \wedge F_1', F_2 \wedge F_2', ..., F_n \wedge F_n' \rangle$ if and only if $\forall 0 \leqslant i \leqslant n: \langle F_1, F_2, ..., F_n; i+1 \rangle \in \mathrm{COUNTSAT} \vee \langle F_1 \wedge F_1', F_2 \wedge F_2', ..., F_n \wedge F_n'; i \rangle \notin \mathrm{COUNTSAT}$.

Since obviously the last statement is a $\mathbf{D}\Pi_2^{\mathscr{P}}$-question, the function which maps $w \in (X^* \not\in X^* \$)^*$, $w = F_1 \not\in F_1' \$ F_2 \not\in F_2' \$ \cdots \$ F_n \not\in F_n' \$$ onto

$$\begin{aligned} &\langle F_1, ..., F_n; 1 \rangle \not\in \langle F_1 \wedge F_1', ..., F_n \wedge F_n'; 0 \rangle \$ \langle F_1, ..., F_n; 2 \rangle \\ &\quad \not\in \langle F_1 \wedge F_1', ..., F_n \wedge F_n'; 1 \rangle \$ \\ &\quad \cdots \$ \langle F_1, ..., F_n; n+1 \rangle \not\in \langle F_1 \wedge F_1', ..., F_n \wedge F_n'; n \rangle \$ \end{aligned}$$

is just a log space reduction from $(\mathrm{SAT} \not\in \mathrm{CoSAT})^{(\exists)}$ to $(\mathrm{COUNTSAT} \not\in X^* \cup X^* \not\in \mathrm{CoCOUNTSAT})^{(\forall)} \in \mathbf{D}\Pi_2^{\mathscr{P}}$ (for details cf. Jenner, 1987).

Now, using just this idea we want to show in more detail how $(\mathrm{GAP} \not\in \mathrm{CoGAP})^{(\exists)}$ can be reduced to $(\mathrm{COUNTGAP} \not\in X^* \cup X^* \not\in \mathrm{CoCOUNTGAP})^{(\forall)} \in \mathbf{A}\Pi_2^{\mathscr{L}}$, thus obtaining $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{A}\Pi_2^{\mathscr{L}}$. To construct our vectors as above we now need the notion of joining graphs, an operation behaving on graphs as the operation conjunction on Boolean formulae, where $F \wedge F'$ is satisfiable iff F is satisfiable and F' is satisfiable. We define:

Let G and G' be graphs, given by their adjacency matrices $M = (a_{ij})_{1 \leqslant i,j \leqslant n}$ and $M' = (b_{ij})_{1 \leqslant i,j \leqslant m}$ with $a_{ij}, b_{ij} \in \{0, 1\}$. Then the *joined graph* $G \subsetneq G'$ is represented by

$$M'' = (c_{i,j})_{1 \leqslant i,j \leqslant n+m} \quad \text{with} \quad c_{i,j} := \begin{cases} a_{i,j} & \text{if } 1 \leqslant i, j \leqslant n; \\ b_{i-n,j-n} & \text{if } n+1 \leqslant i, j \leqslant n+m; \\ 1 & \text{if } i = n \text{ and } j = n+1; \\ 0 & \text{otherwise.} \end{cases}$$

If G or G' are not of appropriate form, i.e., not matrices with entries from $\{0, 1\}$, let

$$G \subsetneq G' := \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

so that the join operator is defined for arbitrary words.

Obviously, $G \subsetneq G'$ can be constructed in log space, and the construction ensures that there is a path from vertex 1 to vertex $n+m$ in $G \subsetneq G'$ iff there is a path from vertex 1 to vertex n in G and a path from vertex 1 to vertex m in G', since $c_{n,n+1} = 1$ joins the goal vertex of G to the start vertex of G' and nothing else has been changed.

Using the notion of the join operator, the basic idea of our main result can be stated as: $G_1 \not\in G_1' \$ G_2 \not\in G_2' \$ \cdots \$ G_n \not\in G_n' \$ \in (\text{GAP} \not\in \text{CoGAP})^{(\exists)}$ iff the vector $\langle G_1, G_2, ..., G_n \rangle$ contains more components which are contained in GAP than the vector $\langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', ..., G_n \subsetneq G_n' \rangle$.

LEMMA 4.1. $G_1 \not\in G_1' \$ G_2 \not\in G_2' \$ \cdots \$ G_n \not\in G_n' \$ \in (\text{GAP} \not\in \text{CoGAP})^{(\exists)} \Leftrightarrow \exists 1 \leqslant i \leqslant n$: $\langle G_1, G_2, ..., G_n; i \rangle \in \text{COUNTGAP} \wedge \langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', ..., G_n \subsetneq G_n'; i \rangle \notin \text{COUNTGAP}$.

Proof. It is easy to see that $G_1 \not\in G_1' \$ G_2 \not\in G_2' \$ \cdots \$ G_n \not\in G_n' \$ \in (\text{GAP} \not\in \text{CoGAP})^{(\exists)}$ if and only if $G_1 \not\in G_1 \subsetneq G_1' \$ G_2 \not\in G_2 \subsetneq G_2' \$ \cdots \$ G_n \not\in G_n \subsetneq G_n' \$ \in (\text{GAP} \not\in \text{CoGAP})^{(\exists)}$. But $G_i \subsetneq G_i' \in \text{GAP}$ implies $G_i \in \text{GAP}$, hence the number of solvable instances in $\langle G_1, G_2, ..., G_n \rangle$ is not smaller than the number of solvable instances in $\langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', ..., G_n \subsetneq G_n' \rangle$, which is: $\langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', ..., G_n \subsetneq G_n'; i \rangle \in \text{COUNTGAP}$ implies $\langle G_1, G_2, ..., G_n; i \rangle \in \text{COUNTGAP}$. But obviously the existence of a pair $G_i \not\in G_i \subsetneq G_i' \in \text{GAP} \not\in \text{CoGAP}$ means that $\langle G_1, G_2, ..., G_n \rangle$ contains *properly* more solvable instances than $\langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', ..., G_n \subsetneq G_n' \rangle$. And it can be easily verified that if $0 \leqslant a, b \leqslant n$, then $a > b \Leftrightarrow \exists 1 \leqslant i \leqslant n$: $a = i \wedge b < i \Leftrightarrow \exists 1 \leqslant i \leqslant n$: $a \geqslant i \wedge b < i$. ∎

Now we are able to prove our main result:

THEOREM 4.2. $\mathbf{A}\boldsymbol{\Sigma}_2^{\mathscr{L}} = \mathbf{A}\boldsymbol{\Pi}_2^{\mathscr{L}}$.

Proof. For $0 \leqslant a, b \leqslant n$ the equivalence

$$\exists 1 \leqslant i \leqslant n\text{:}\quad a \geqslant 1 \wedge b < i \;\Leftrightarrow\; \forall 0 \leqslant j \leqslant n\text{:}\quad a \geqslant j+1 \vee b < j$$

is easy to verify. Hence

$$\exists 1 \leqslant i \leqslant n\text{: } \langle G_1, G_2, ..., G_n; i \rangle \in \text{COUNTGAP} \wedge \langle G_1 \subsetneq G_1', ..., G_n \subsetneq G_n'; i \rangle \notin \text{COUNTGAP}$$

is equivalent to

$$\forall 0 \leqslant j \leqslant n\text{: } \langle G_1, G_2, ..., G_n; j+1 \rangle \in \text{COUNTGAP} \vee \langle G_1 \subsetneq G_1', ..., G_n \subsetneq G_n'; j \rangle \notin \text{COUNTGAP}.$$

Let $A_i := \langle G_1, G_2, ..., G_n; i\rangle$ and $B_i := \langle G_1 \subsetneq G_1', ..., G_n \subsetneq G_n'; i\rangle$ for $0 \leqslant i \leqslant n+1$. Then we obtain

$$\begin{aligned} G_1 \notin G_1' \$ G_2 \notin G_2' \$ \cdots \$ G_n \notin G_n' \$ &\in (\mathrm{GAP} \notin \mathrm{CoGAP})^{(\exists)} \\ &\Leftrightarrow \exists 1 \leqslant i \leqslant n\colon A_i \in \mathrm{COUNTGAP} \wedge B_i \notin \mathrm{COUNTGAP} \\ &\Leftrightarrow \forall 0 \leqslant j \leqslant n\colon A_{j+1} \in \mathrm{COUNTGAP} \vee B_j \notin \mathrm{COUNTGAP} \\ &\Leftrightarrow A_1 \notin B_0 \$ A_2 \notin B_1 \$ \cdots \$ A_{n+1} \notin B_n \$ \\ &\quad \in (\mathrm{COUNTGAP} \notin X^* \cup X^* \notin \mathrm{CoCOUNTGAP})^{(\forall)}. \end{aligned}$$

Obviously, A_i and B_i can be computed from $G_1 \notin G_1' \$ \cdots \$ G_n \notin G_n' \$$ in log space for any i, $0 \leqslant i \leqslant n+1$. Thus $A_1 \notin B_0 \$ \cdots \$ A_{n+1} \notin B_n \$$ can be computed in log space. As COUNTGAP is an element of **NL** obviously (COUNTGAP$\notin X^* \cup X^* \notin$CoCOUNTGAP) is an element of **NL** $\vee$ **CoNL**. Because of Lemma 3.2 (ii) and Theorem 3.3, we conclude (COUNTGAP$\notin X^* \cup X^* \notin$CoCOUNTGAP)$^{(\forall)} \in \mathbf{A\Pi}_2^{\mathscr{L}}$, which implies $\mathbf{A\Sigma}_2^{\mathscr{L}} \subseteq \mathbf{A\Pi}_2^{\mathscr{L}}$ and with $\mathbf{A\Sigma}_2^{\mathscr{L}} = \mathbf{CoA\Pi}_2^{\mathscr{L}}$ we obtain $\mathbf{A\Sigma}_2^{\mathscr{L}} = \mathbf{A\Pi}_2^{\mathscr{L}}$. ▮

Because of Theorem 3.3(ii), we get the collapse of the logarithmic alternation hierarchy with $\mathbf{A\Sigma}_{k+1}^{\mathscr{L}} = \mathbf{L}_{\mathbf{d}}(\mathbf{A\Pi}_k^{\mathscr{L}}) = \mathbf{L}_{\mathbf{d}}(\mathbf{A\Sigma}_k^{\mathscr{L}}) = \mathbf{A\Sigma}_k^{\mathscr{L}}$, since $\mathbf{A\Sigma}_k^{\mathscr{L}}$ is already closed under log space disjunctive Turing reductions.

COROLLARY 4.3. $\mathbf{A\Sigma}_2^{\mathscr{L}} = \mathbf{A\Pi}_2^{\mathscr{L}} = \mathbf{A\Sigma}_k^{\mathscr{L}}$ *for all* $k \geqslant 2$. ▮

As shown with the following lemma by applying padding techniques this result generalizes to all alternating space hierarchies with space constructible bounds $S(n) \geqslant \log(n)$. This includes the collapse of the alternating linear space hierarchy at its second level $\mathbf{A\Sigma_2SPACE}(n) = \mathbf{A\Pi_2SPACE}(n) = \mathbf{A\Sigma_k SPACE}(n)$ for all $k \geqslant 2$, which was independently shown by Toda (1987).

LEMMA 4.4. *Let* $L \subseteq X^*$, $\# \notin X$, *and define* $L_f := \{w \#^m \mid w \in L, m = 2^{f(|w|)} - |w|\}$. *Then it holds for all fully space constructible* $f\colon \mathbb{N} \to \mathbb{N}$ *with* $f(n) \geqslant \log n$: $L \in \mathbf{A\Sigma_k SPACE}(f(n)) \Leftrightarrow L_f \in \mathbf{A\Sigma}_k^{\mathscr{L}}$.

Proof. "$\Rightarrow$": Given $w \#^m$, check first if $m = 2^{f(|w|)} - |w|$. For this obviously $f(|w|)$ space suffices (deterministically). Now, check $w \in L$ using $f(|w|)$ space with $k-1$ alternations, i.e., with $\mathbf{A\Sigma_k SPACE}(f(|w|))$. Thus the total amount of space used is $O(\log(|w \#^m|))$. Hence $L_f \in \mathbf{A\Sigma}_k^{\mathscr{L}}$.

"$\Leftarrow$": Given w, to represent $m = 2^{f(|w|)} - |w|$ in binary consumes $f(|w|)$ space. Now, first construct bin(m), the binary representation of m, on the working tape. As f is space constructible $f(|w|)$ space suffices. Then simulate the $A\Sigma_k^{\mathscr{L}}$-machine that recognizes L_f on $w \#^m$ where instead of

$\#^m$ bin(m) is given. The space necessary is $\log(|w\,\#^m|) = f(|w|)$. Thus the total amount of space is $O(f(|w|))$. Hence $L \in \mathbf{A\Sigma_k SPACE}(f(n))$. ∎

Corollary 4.3 and Lemma 4.4 yield

COROLLARY 4.5. *For all fully space constructible* $f: \mathbb{N} \to \mathbb{N}$ *with* $f(n) \geqslant \log n$ *it holds*:

$$\mathbf{A\Sigma_2 SPACE}(f(n)) = \mathbf{A\Pi_2 SPACE}(f(n)) = \mathbf{A\Sigma_k SPACE}(f(n)) \quad \textit{for all } k \geqslant 2.$$

Having shown that the logarithmic alternation hierarchy collapses at its second level a careful analysis shows that $\mathbf{A\Sigma}_2^{\mathscr{L}}$ is even equal to $\mathbf{L_{hd}(NL)}$.

Set $B := \{H\$H' \mid H \in \text{COUNTGAP} \wedge H' \in \text{COUNTGAP}\}$, and for $G_1 \not\subseteq G_1' \$ G_2 \not\subseteq G_2' \$ \cdots \$ G_n \not\subseteq G_n' \$$ define for all $1 \leqslant i \leqslant n$,

$$F_{2i-1} := \langle G_1, G_2, \ldots, G_n; i\rangle \,\$\langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', \ldots, G_n \subsetneq G_n'; i-1\rangle$$

$$F_{2i} := H_0 \$ \langle G_1 \subsetneq G_1', G_2 \subsetneq G_2', \ldots, G_n \subsetneq G_n'; i\rangle,$$

where H_0 is some fixed element of COUNTGAP.

The following properties of our construction are obvious.

LEMMA 4.6. (i) $\exists 1 \leqslant i \leqslant n$: $\langle G_1, \ldots, G_n; i\rangle \in \text{COUNTGAP} \wedge \langle G_1 \subsetneq G_1', \ldots, G_n \subsetneq G_n'; i\rangle \notin \text{COUNTGAP} \Leftrightarrow \exists 1 \leqslant i \leqslant n$: $F_{2i-1} \in B \wedge \neg(F_{2i} \in B)$,

(ii) $\forall 1 \leqslant j < 2n$: $F_{j+1} \in B \Rightarrow F_j \in B$, *and*

(iii) $B \in \mathbf{NL}$.

THEOREM 4.7. $(\text{GAP}\not\in\text{CoGAP})^{(\exists)} \in \mathbf{L_{hd}(NL)}$.

Proof. Define $f: (X \cup \{\not\in, \$\})^* \to (Y^* \#)^*$ with

$$f(w) := \begin{cases} F_1 \# \cdots \# F_{2n} \#, & \text{if } w \in (X^* \not\in X^* \$)^+, \\ & \quad w = G_1 \not\in G_1' \$ \cdots \$ G_n \not\in G_n' \$ \ (F_i \text{ as above}), \\ \# \ \#, & \text{else.} \end{cases}$$

Notice that $\# \ \# = \lambda \# \lambda \#$, where λ denotes the empty word. As $\lambda \notin B$, obviously $\# \ \#$ is an element of Co$B \ \#$ Co$B \ \#$ (B as above). Thus $f(w) = \# \ \#$ implies $w \notin (\text{GAP}\not\in\text{CoGAP})^{(\exists)}$. For all $w \in (X^* \not\in X^* \$)^+$, $w = G_1 \not\in G_1' \$ \cdots \$ G_n \not\in G_n' \$$ we get by Lemma 4.1 and Lemma 4.6(i),

$$G_1 \not\in G_1' \$ \cdots \$ G_n \not\in G_n' \$ \in (\text{GAP}\not\in\text{CoGAP})^{(\exists)}$$
$$\Leftrightarrow \exists 1 \leqslant i \leqslant n: F_{2i-1} \in B \wedge \neg(F_{2i} \in B).$$

And by Lemma 4.6(ii),

$$\forall 1 \leqslant i < 2n: F_{i+1} \in B \Rightarrow F_i \in B.$$

Since the F_i can be constructed in log space—as can be easily verified—f is log space computable. Hence $(\mathrm{GAP}\notin\mathrm{CoGAP})^{(\exists)} \leqslant_{hd}^{\mathscr{L}} B$. It follows with Lemma 4.6(iii): $(\mathrm{GAP}\notin\mathrm{CoGAP})^{(\exists)} \in \mathbf{L}_{\mathbf{hd}}(\mathbf{NL})$. ∎

COROLLARY 4.8. $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{L}_{\mathbf{hd}}(\mathbf{NL})$.

Proof. Theorem 3.9, Corollary 3.4, and Theorem 4.7 yield the result. ∎

5. DISCUSSION

Having shown that $(\mathrm{SAT}\notin\mathrm{CoSAT})^{(\exists)}$ can be reduced to $((\mathrm{COUNTSAT}\notin X^*) \cup (X^*\notin\mathrm{CoCOUNTSAT}))^{(\forall)}$ and that $(\mathrm{GAP}\notin\mathrm{CoGAP})^{(\exists)}$ can be reduced to $((\mathrm{COUNTGAP}\notin X^*) \cup (X^*\notin\mathrm{CoCOUNTGAP}))^{(\forall)}$, now, of course, the question arises whether for any $k \geqslant 1$ $(\mathrm{SAT}\notin\mathrm{CoSAT})^{(\exists k)}$ can be reduced to $((\mathrm{COUNTSAT}\notin X^*) \cup (X^*\notin\mathrm{CoCOUNTSAT}))^{(\forall k)}$ and/or $((\mathrm{GAP}\notin\mathrm{CoGAP}))^{(\exists k)}$ can be reduced to $((\mathrm{COUNTGAP}\notin X^*) \cup (X^*\notin\mathrm{CoCOUNTGAP}))^{(\forall k)}$. With Theorem 3.7 we know that this is just the question whether the Boolean **NP**-hierarchy collapses at any even level. Very recently it was shown by Immerman (1987) and (independently) by Szelepcsényi (1987) that, surprisingly, $\mathbf{NL} = \mathbf{CoNL}$ and thus—improving our result—not only the logarithmic alternation hierarchy but the Boolean **NL**-hierarchy (the analog of the Boolean **NP**-hierarchy), too, collapses even at its first level. Note that this result, too, improves the also recently obtained collapse of the logarithmic oracle hierarchy to $\mathbf{L}_{\mathbf{hd}}(\mathbf{NL})$ by Schöning and Wagner (1987). All these proofs, including the one we stated, necessarily make use of the polynomial bounded number of configurations reachable from the initial configuration for logarithmic space bounded classes.

The situation on the polynomial side seems to be quite different. Here we have exponentially many possible configurations reachable from the initial configuration, and it is widely conjectured that $\mathbf{NP} \neq \mathbf{CoNP}$, i.e., a collapse of the Boolean **NP**-hierarchy on its first level is unlikely. That a collapse on any higher level seems to be equally hard has recently been shown by Kadin (1987), since it implies the collapse of the polynomial hierarchy for which now quite a few very different characterizations have been obtained (Stockmeyer, 1976; Chandra, Kozen, and Stockmeyer, 1981; Jenner and Kirsig, 1988) and which is widely conjectured to be infinite.

Coming to the end of this paper we want to show how the proof technique enabled showing $\mathbf{D}\Sigma_2^{\mathscr{L}} = \mathbf{D}\Pi_2^{\mathscr{L}}$ and $\mathbf{A}\Sigma_2^{\mathscr{L}} = \mathbf{A}\Pi_2^{\mathscr{L}}$ must be improved to yield $\mathbf{NP(2k)} = \mathbf{CoNP(2k)}$ if applied to any class $\mathbf{NP(2k)}$, $k \geqslant 1$, of the Boolean **NP**-hierarchy (see also Jenner, 1987). This consideration will emphasize that—in the light of Kadin (1987)—our proof technique seems to be optimal for the polynomial time case.

Consider the language $(\mathrm{SAT} \notin \mathrm{CoSAT})^{(\exists_k)}$ for any $k \geqslant 1$. With an easily obtained analog of Lemma 4.1 and Theorem 4.2 we get

$$
\begin{aligned}
& F_1 \notin F_1' \$ \cdots \$ F_k \notin F_k' \$ \in (\mathrm{SAT} \notin \mathrm{CoSAT})^{(\exists_k)} \\
& \quad \Leftrightarrow \exists 1 \leqslant i \leqslant k\colon \langle F_1, \ldots, F_k; i \rangle \in \mathrm{COUNTSAT} \\
& \qquad \wedge \langle F_1 \wedge F_1', \ldots, F_k \wedge F_k'; i \rangle \notin \mathrm{COUNTSAT} \\
& \quad \Leftrightarrow \forall 0 \leqslant i \leqslant k\colon \langle F_1, \ldots, F_k; i+1 \rangle \in \mathrm{COUNTSAT} \\
& \qquad \vee \langle F_1 \wedge F_1', \ldots, F_k \wedge F_k'; i \rangle \notin \mathrm{COUNTSAT} \\
& \quad \Leftrightarrow \langle F_1, \ldots, F_k; 1 \rangle \in \mathrm{COUNTSAT} \\
& \qquad \wedge \langle F_1 \wedge F_1', \ldots, F_k \wedge F_k'; k \rangle \notin \mathrm{COUNTSAT} \\
& \qquad \wedge \forall 1 \leqslant i \leqslant k-1\colon \langle F_1, \ldots, F_k; i+1 \rangle \in \mathrm{COUNTSAT} \\
& \qquad \vee \langle F_1 \wedge F_1', \ldots, F_k \wedge F_k'; i \rangle \notin \mathrm{COUNTSAT}. \qquad (***)
\end{aligned}
$$

As can be seen by looking at the normal forms of $\mathbf{NP(2k)}$ for any $k \geqslant 1$ (compare Wechsung, 1985),

$$\mathbf{NP(2k)} = \bigvee_{i=1}^{k} (\mathbf{NP} \wedge \mathbf{CoNP}) = \mathbf{NP} \wedge \mathbf{CoNP} \wedge \bigwedge_{i=1}^{k-1} (\mathbf{NP} \vee \mathbf{CoNP}),$$

$(***)$ is obviously a $\mathbf{NP(2k)}$-question, and furthermore, the reduction obviously leads from the first normal form of $\mathbf{NP(2k)}$ to the second. Since one of the normal forms of $\mathbf{CoNP(2k)}$ is just

$$\mathbf{CoNP(2k)} = \bigwedge_{i=1}^{k} (\mathbf{NP} \vee \mathbf{CoNP}),$$

the reduction we constructed from $(\mathrm{SAT} \notin \mathrm{CoSAT})^{(\exists)}$ to $((\mathrm{COUNTSAT} \notin X^*) \cup (X^* \notin \mathrm{CoCOUNTSAT}))^{(\forall)}$ was thus obviously optimal in the sense that if it had led to any equally structured conjunction as $(***)$ with less than $k+1$ conjunction terms it would have been a reduction from $(\mathrm{SAT} \notin \mathrm{CoSAT})^{(\exists_k)}$ to a $\mathbf{CoNP(2k)}$-language, which would give a proof of the collapse of the Boolean **NP**-hierarchy.

RECEIVED November 10, 1987; ACCEPTED March 11, 1988

REFERENCES

CAI, J., AND HEMACHANDRA, L. (1986), The Boolean hierarchy: Hardware over NP, *in* "Proceedings, Conf. Structure in Complexity Theory, Univ. of California, Berkeley," Lect. Notes in Comput. Sci. Vol. 223, pp. 105–124, Springer-Verlag, New York/Berlin.

CHANDRA, A. K., KOZEN, D. C., AND STOCKMEYER, L. J. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**, No. 1, 114–133.

COOK, S. A. (1971), The complexity of theorem proving procedures, *in* "Proceedings, Third Annual ACM Symposium on the Theory of Computing," pp. 151–158.

GOLDSCHLAGER, L. (1977), The monotone and planar circuit value problems are LOG space complete for P, *SIGACT News* **9**, No. 35, 25–29.

HOPCROFT, J., AND ULLMAN, J. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.

IMMERMAN, N. (1987), "Nondeterministic Space Is Closed under Complement," Technical Report YALEU/DCS/TR 552, Yale University, July 1987.

JENNER, B. (1987), "Die deterministische polynomielle Hierarchie—Ein polynomielles Analogon der Logarithmischen Alternierungshierarchie—," Diploma thesis, University of Hamburg.

JENNER, B., AND KIRSIG, B. (1988), Characterizing the polynomial hierarchy by alternating auxiliary pushdown automata, *in* "Proceedings, 5th STACS," Lect. Notes in Comput. Sci. Vol. 294, pp. 118–125, Springer-Verlag, New York/Berlin.

JONES, N. D. (1973), Reducibility among combinatorial problems in $\log n$ space, *in* "Proceedings, 7th Annual Princeton Conf. on Information Sciences and Systems," pp. 547–551.

KADIN, J. (1987), The polynomial hierarchy collapses if the Boolean hierarchy collapses, manuscript.

KIRSIG, B. (1986), Logarithmic Hausdorff reductions, submitted for publication.

KÖBLER, J., SCHÖNING, U., AND WAGNER, K. W. (1985), The difference and truth-table hierarchies for NP, manuscript.

LADNER, R. E., AND LYNCH, N. (1976), Relativization of questions about log space computability, *Math. Systems Theory* **10**, 19–32.

LADNER, R. E., LYNCH, N., AND SELMAN, A. (1975), A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1**, 103–123.

LANGE, K.-J. (1986), Two characterizations of the logarithmic alternation hierarchy, *in* "Proceedings, 12th Symp. of Math. Foundations of Comput. Science," Lect. Notes in Comput. Sci. Vol. 233, pp. 518–526, Springer-Verlag, New York/Berlin.

LANGE, K.-J., JENNER, B., AND KIRSIG, B. (1987), The logarithmic alternation hierarchy collapses: $A\Sigma_2^{\mathscr{L}} = A\Pi_2^{\mathscr{L}}$, *in* "Proceedings, 14th ICALP, Karlsruhe," Lect. Notes in Comput. Sci., Vol. 267, pp. 531–541, Springer-Verlag, New York/Berlin.

MEYER, A., AND STOCKMEYER, L. (1972), The equivalence problem for regular expressions with squaring requires exponential space, *in* "Proceedings, 13th Annual IEEE Symp. on Switching and Automata Theory 1972," pp. 125–129.

PAPADIMITRIOU, C. H., AND YANNAKAKIS, M. (1984), The complexity of facets (and some facets of complexity), *J. Comput. System Sci.* **28**, 244–259.

ROSIER, L. E., AND YEN, H. (1986), Logspace hierarchies, polynomial time and the complexity of fairness problems concerning ω-machines, *in* "Proceedings, 3rd STACS," Lect. Notes in Comput. Sci. Vol. 210, pp. 306–320, Springer-Verlag, New York/Berlin.

RUZZO, W., SIMON, J., AND TOMPA, M. (1984), Space-bounded hierarchies and probabilistic computations, *J. Comput. System Sci.* **28**, 216–230.

SAVITCH, W. (1970), Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **4**, 177–192.

SCHÖNING, U., AND WAGNER, K. W. (1987), Collapsing oracle hierarchies, census functions and logarithmically many queries, Report 140, Universität Augsburg.

STOCKMEYER, L. J. (1976), The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3**, 1–22.

SZELEPSCÉNYI, R. (1987), The method of forcing for nondeterministic automata, *Bull. European Associ. Theoret. Comput. Sci.* **33**, 96–100.

TODA, S. (1987), Σ_2SPACE(n) is closed under complement, *J. Comput. System Sci.* **35**, No. 2 145–152.

WAGNER, K. W. (1986a), More complicated questions about maxima and minima, and some closures of NP, report, University of Passau.

WAGNER, K. W. (1986b), The complexity of combinatorial problems with succint input representation, *Acta Inform.* **23**, No. 3, 325–356.

WECHSUNG, G. (1985), On the Boolean closure of NP, *in* "Proceedings, FCT Conf.," Lect. Notes in Comput. Sci. Vol. 199, pp. 485–493, Springer-Verlag, New York/Berlin.