SIAM J. COMPUT.
Vol. 18, No. 3, p. 559-578, June 1989
© 1989 Society for Industrial and Applied Mathematics
009

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TWO APPLICATIONS OF INDUCTIVE COUNTING FOR COMPLEMENTATION PROBLEMS*

ALLAN BORODIN†, STEPHEN A. COOK†, PATRICK W. DYMOND‡, WALTER L. RUZZO§, AND MARTIN TOMPA¶

Abstract. Following the recent independent proofs of Immerman [*SIAM J. Comput.*, 17 (1988), pp. 935-938] and Szelepcsényi [*Bull. European Assoc. Theoret. Comput. Sci.*, 33 (1987), pp. 96-100] that nondeterministic space-bounded complexity classes are closed under complementation, two further applications of the inductive counting technique are developed. First, an errorless probabilistic algorithm for the undirected graph s-t connectivity problem that runs in $O(\log n)$ space and polynomial expected time is given. Then it is shown that the class *LOGCFL* is closed under complementation. The latter is a special case of a general result that shows closure under complementation of classes defined by semi-unbounded fan-in circuits (or, equivalently, nondeterministic auxiliary pushdown automata or tree-size bounded alternating Turing machines). As one consequence, it is shown that small numbers of "role switches" in two-person pebbling can be eliminated.

Key words. complementation, inductive counting, connectivity, symmetric computation, probabilistic algorithm, random walk, *LOGCFL*, *NC*, semi-unboundedness, pebbling, hierarchy

AMS(MOS) subject classifications. 68Q15, 68Q25, 68Q75, 68R10, 60J15, 94C99, 03D55, 90D05

1. Introduction.

Let *NL* denote the class of languages accepted by nondeterministic Turing machines running in $O(\log n)$ space. Let

$$STCON = \{(G, s, t) \mid G \text{ is a directed graph containing a directed path from vertex } s \text{ to vertex } t\}.$$

Using any reasonable encoding of graphs, it is well known that *STCON* is in *NL* and, moreover, is complete for *NL* with respect to deterministic log space reductions (Savitch [29]). In a surprising development, Immerman [17] and Szelepcsényi [33] have shown independently that $\overline{STCON}$, the complement of *STCON*, is also in *NL*; that is, *NL* is closed under complementation.

Their proofs rely on an inductive counting technique similar to counting techniques used in related results, for instance, Mahaney [24], Lange, Jenner, and Kirsig [22], Hemachandra [15], Toda [34], Buss, Cook, Dymond, and Hay [4], and Schöning and Wagner [30]. (For additional background and references see Hartmanis [14].) It seems inevitable that this technique should have further application and in this paper we develop two such applications.

For our first application, we consider reachability in undirected graphs, a problem

* Received by the editors October 28, 1987; accepted for publication (in revised form) May 26, 1988. This material is based on work supported in part by the Natural Sciences and Engineering Research Council of Canada, and by the National Science Foundation under grants DCR-8604031 and CCR-8703196. Much of the work was performed while Allan Borodin and Larry Ruzzo were visitors at the IBM Thomas J. Watson Research Center, and Patrick Dymond was a visitor at the University of Toronto.

† Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4.

‡ Department of Computer Science and Engineering, C-014, University of California at San Diego, La Jolla, California 92093.

§ Department of Computer Science, FR-35, University of Washington, Seattle, Washington 98195.

¶ IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

not known to be complete for *NL*. Indeed, Aleliunas et al. [2] have shown that

$$USTCON = \{(G, s, t) \mid G \text{ is an undirected graph containing a path from vertex } s \text{ to vertex } t\}$$

is in *RLP*, the class of languages accepted by probabilistic Turing machines running in $O(\log n)$ space and polynomial time. In this model, the machine never reaches an accepting state on any input not in *USTCON*, and has probability at least $\frac{1}{2}$ of reaching an accepting state on any input in *USTCON*. *USTCON* is a complete problem for *SL*, the class of languages accepted by symmetric Turing machines (Lewis and Papadimitriou [23]) running in $O(\log n)$ space. It follows that $SL \subseteq RLP \subseteq NL$.

It is tempting to believe that Immerman's and Szelepcsényi's proofs extend to show that *SL* is closed under complementation. However, a direct translation of their technique fails to establish this result, as explained in § 2.2. In that section we prove the somewhat weaker result that $\overline{USTCON}$ is in *RLP* or, equivalently, that *USTCON* is in *ZPLP*, the class of languages accepted by errorless probabilistic Turing machines running in $O(\log n)$ space and polynomial expected time. (The equivalence is due to the fact that $RLP \cap \text{co}RLP = ZPLP$.) This answers a problem raised by Aleliunas et al. [2]. In § 2.3 we extend this result to show that Reif's symmetric log space complementation hierarchy [26] is also contained in *ZPLP*.

In our second application we show the closure under complementation of a number of complexity classes that are (seemingly) more powerful than *NL*. The classes we consider may be characterized in terms of several different models. The most intuitively appealing model is perhaps the semi-unbounded fan-in circuit model (see Venkateswaran [37]). In this model, we allow OR gates with arbitrary fan-in, whereas all AND gates have bounded fan-in. Input variables and their negations are supplied, but negations are prohibited elsewhere.

For simplicity we will restrict the discussion to polynomial-size circuits, although the results can be generalized. Of particular interest is the class SAC^k of languages accepted by polynomial-size, $O(\log^k n)$ depth, uniform semi-unbounded fan-in circuits. (See Cook [7] for an appropriate definition of uniformity.) SAC^1 is the most often studied of these classes. SAC^1 is equal to the class *LOGCFL* of languages log space reducible to context-free languages [32], [37]. It is known that $NC^1 \subseteq NL \subseteq SAC^1 \subseteq AC^1$ where, as is customary, we let NC^k and AC^k denote the classes analogous to SAC^k for bounded fan-in and (respectively) unbounded fan-in uniform circuits. More generally, $NC^k \subseteq SAC^k \subseteq AC^k \subseteq NC^{k+1}$.

Both NC^k and AC^k are easily seen to be closed under complementation by application of De Morgan's laws. However, SAC^k does not yield to the same technique, since it would produce circuits with unbounded fan-in AND gates. In fact, it is known that there is a language accepted by polynomial size, constant depth, uniform semi-unbounded fan-in circuits, but whose complement is not accepted by semi-unbounded fan-in circuits of depth $o(\log n)$ and arbitrary size, even nonuniformly (Venkateswaran [36], [37]). The main result of § 3.1 is to show that polynomial-size semi-unbounded circuit classes *are* closed under complementation for all depths that are $\Omega(\log n)$. Closure under complementation of classes defined by auxiliary pushdown automata, tree-size bounded alternating Turing machines, and simple first-order formulae then follows by known equivalences (see § 3.1).

In § 3.2, we examine some consequences of the closure of semi-unbounded circuit classes under complementation. In the same way that the alternating and oracle hierarchies based on *NL* [5], [28] collapse because of Immerman's and Szelepcsényi's result, hierarchies based on semi-unbounded circuit classes also collapse. As a con-

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

sequence, the "role switch" resource in the pebble game introduced by Venkateswaran and Tompa [38] is shown to be much weaker than previously seemed plausible. For instance, we demonstrate that any fixed number of role switches can be eliminated.

2. Errorless algorithms related to undirected reachability.

2.1. Probabilistic complexity classes. An explanation of our taxonomy for probabilistic complexity classes is in order. Table 1 illustrates the sense in which the names *RLP* and *ZPLP* encountered in § 1 are consistent with the notation *ZPP*, *BPP*, and *PP* of Gill [12], *RP* of Welsh [40], and *BPL* and *PL* of Ruzzo, Simon, and Tompa [28].

TABLE 1
Taxonomy for probabilistic complexity classes.

Type of error	Polynomial expected time	$O(\log n)$ space	$O(\log n)$ space and polynomial expected time
Zero-sided	*ZPP*	*ZPL*	*ZPLP*
One-sided	*RP*	*RL*	*RLP*
Bounded two-sided	*BPP*	*BPL*	*BPLP*
Unbounded two-sided	*PP*	*PL*	*PLP*

Since this section concentrates on the classes *RLP* and *ZPLP*, we give them careful definitions here. We say that a language A is in *RLP* if and only if there is a probabilistic Turing machine M such that

(1) For all inputs, M runs in space $O(\log n)$ and expected time $n^{O(1)}$,

(2) If $w \in A$, $\Pr[M \text{ reaches an accepting state on input } w] \geqq \frac{1}{2}$, and

(3) If $w \notin A$, $\Pr[M \text{ reaches an accepting state on input } w] = 0$.

ZPLP is the class obtained by replacing condition 2 by 2′:

(2′) If $w \in A$, $\Pr[M \text{ reaches an accepting state on input } w] = 1$.

The class *RLP* remains unchanged if we require polynomial time rather than just polynomial expected time. Results of Gill [12], Immerman [17], and Szelepcsényi [33] show that, when the polynomial time bound is removed, the corresponding one-sided (*RL*) and zero-sided (*ZPL*) classes are equal to *NL*. Ruzzo, Simon, and Tompa [28] and Simon [31] have shown that *PL* and *BPL* are closed under complementation. Jung [20] has shown that $PL = PLP$. These relations and others (including those proved in this paper) are summarized in Fig. 1. (Those complexity classes whose complements are not explicitly shown in Fig. 1 are closed under complementation. *DL* denotes $DSPACE(\log n)$. *DET* is the set of languages reducible to computing integer matrix determinants [7]. $\bigcup_k C\sum_k^{SL}$ is the symmetric log space hierarchy, discussed in § 2.3.)

2.2. An errorless algorithm for undirected reachability. Immerman's and Szelepcsényi's proofs that $\overline{STCON} \in NL$ rely on computing

$$N_k = \#\{v \mid v \text{ is within distance } k \text{ of } s\}$$

by induction on k. As mentioned in § 1, it is tempting to believe that the same method can be used to show that *SL* is closed under complementation. Perhaps the easiest way to see the difficulty (and importance) of such a result is to observe that Immerman's and Szelepcsényi's algorithm also computes the length of a shortest path from s to t. By a known reduction (Ladner (personal communication)), *STCON* is log space reducible to the problem of determining if the length of a shortest path from s to t in an undirected graph is $n-1$. Hence a direct translation of Immerman's and Szelepcsényi's proof to *SL* would also solve this problem, thus implying $SL = RLP = NL$.

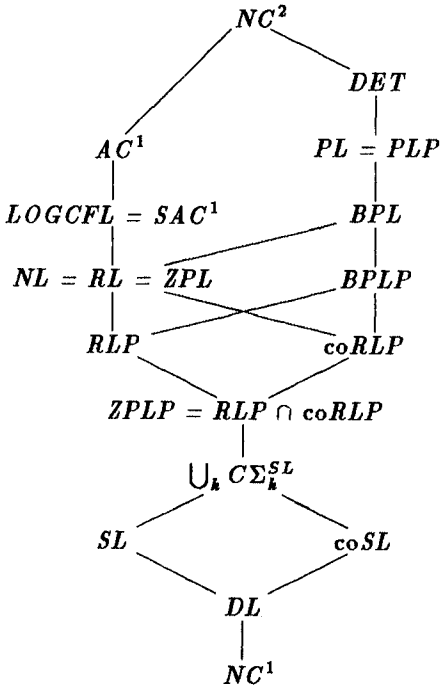

FIG. 1. *Inclusion relations among* $O(\log n)$ *space bounded complexity classes.*

The specific obstacle in applying Immerman's and Szelepcsényi's proof technique to *SL* is that a symmetric computation cannot "nondeterministically count," which seems to be the key feature in their method. (This was also noted in [16].) In particular, suppose that, for any value of *count*, there is a configuration (*R*, *count*) from which the computation can proceed nondeterministically on either of the following paths:

(1) $(R, count) \vdash^* (R', count)$.

(2) $(R, count) \vdash^* (R', count+1)$.

Since all moves in a symmetric machine are reversible, the machine can realize the computation sequence $(R, count) \vdash^* (R', count+1)\ {}^*\!\dashv (R, count+1)$. Hence the *count* can be increased (or decreased) arbitrarily.

The main result of this section is Theorem 1.

THEOREM 1. *USTCON* $\in$ *ZPLP* (*or, equivalently,* $SL \subseteq ZPLP$).

This theorem will follow immediately from Lemma 6.

In proving Theorem 1 we no longer face the handicap of symmetric computations discussed above, but we do face another difficulty: the random walk approach of Aleliunas et al. [2] does not seem to provide any useful information on the distance between vertices. To circumvent our inability to compute undirected distances with an *RLP* computation, we use the following idea, which is basic to the algorithms of Kleene [21], Floyd [9], and Warshall [39]. Let $u \to_k^* v$ denote the existence of a path between u and v that does not pass through any intermediate vertices l with $l > k$. Let

$$P_k = \{(u, v) \mid u \to_k^* v\}.$$

The *ZPLP* algorithm for *USTCON* is outlined in the remainder of this paragraph, and described fully thereafter. It proceeds similarly to Immerman's and Szelepcsényi's algorithm, by iteratively computing $\# P_k$, the cardinality of P_k. Having computed $\# P_n$,

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

we alternately attempt to verify that $(G, s, t) \in USTCON$ via a random walk, or that $(G, s, t) \in \overline{USTCON}$ by verifying that there are $\#P_n$ pairs $(u, v) \neq (s, t)$ that are in P_n. We compute $\#P_k$ from $\#P_{k-1}$ by determining, for each pair (u, v), whether or not it is in P_k. We note that $(u, v) \in P_k$ if and only if either $(u, v) \in P_{k-1}$, or both $(u, k) \in P_{k-1}$ and $(k, v) \in P_{k-1}$. Again, $\#P_{k-1}$ is used to insure that the algorithm never makes a mistake when claiming some (g, h) is or is not in P_{k-1}.

The following lemma is obvious.

LEMMA 2. *$(u, v) \in P_k$ if and only if u and v are in the same connected component of the subgraph $G(k, u, v)$ of G induced by vertices $\{1, 2, 3, \cdots, k, u, v\}$.*

The basis for our algorithm is the random walk technique as used in Aleliunas et al. [2]. It is modified slightly for our purposes in the following algorithm.

```
ALGORITHM WALK(k, u, v).
comment: WALK(k, u, v) simulates a random walk on the subgraph
  G(k, u, v) starting at u, returns FOUND if v is encountered, and
  NOT FOUND otherwise;
begin
    if u = v then return FOUND;
    n ← #V;
    x ← u;
    repeat 2n^3 log_2 (2n^2) times
        begin
            choose y randomly and uniformly from among the neighbors
              of x in G(k, u, v);
            if y = v then return FOUND else x ← y
        end;
    return NOT FOUND
end.
```

LEMMA 3. (1) *If $(u, v) \in P_k$, $\Pr[WALK(k, u, v) = FOUND] \geqq 1 - 1/2n^2$.*

(2) *If $(u, v) \notin P_k$, $\Pr[WALK(k, u, v) = FOUND] = 0$.*

(3) *$WALK(k, u, v)$ uses space $O(\log n)$.*

(4) *$WALK(k, u, v)$ uses expected time $O(n^5 \log n)$ (on a probabilistic Turing machine).*

Proof. The main result of Aleliunas et al. [2] is that the expected number of edge traversals a random walk requires to visit all vertices of a connected undirected graph, beginning at any vertex, is at most n^3. By Markov's inequality [3] and Lemma 2, the probability is at most $\frac{1}{2}$ that $WALK(k, u, v)$ does not encounter v within any specified $2n^3$ iterations of the **repeat** loop, given that $(u, v) \in P_k$. The correctness assertions follow from this.

For the time complexity, we assume that G is presented as an $n \times n$ adjacency matrix. Locating the row of this matrix corresponding to x, computing the degree in $G(k, u, v)$ of x, and selecting a neighbor y can be done in $O(n^2)$ time. There is a technical detail if we assume $\{0, 1\}$ valued probabilistic choices when the degree need not be a power of 2. Suppose $2^r \leqq \text{degree}(x) < 2^{r+1}$. We then choose $r+1$ random bits to compute a random integer $i \in [0, 2^{r+1} - 1]$. If $i > \text{degree}(x)$ we discard i and try again. The expected number of random integers i that need be generated (to obtain $i \leqq \text{degree}(x)$) is at most 2. Thus $WALK(k, u, v)$ uses expected time $O(n^2 n^3 \log n) = O(n^5 \log n)$. □

If $\#P_k$ is known exactly, we need an errorless probabilistic algorithm that determines whether or not (u, v) is in P_k.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

ALGORITHM *PATH*(k, cpk, u, v).
comment: *PATH*$(k, \#P_k, u, v)$ returns *TRUE* iff $(u, v) \in P_k$;
repeat forever
begin
if *WALK*$(k, u, v) =$ *FOUND* **then return** *TRUE*;
$c \leftarrow 0$;
for all $(g, h) \neq (u, v)$ **do**
if *WALK*$(k, g, h) =$ *FOUND* **then** $c \leftarrow c+1$;
if $c = cpk$ **then return** *FALSE*
end.

LEMMA 4. (1) *If PATH*$(k, \#P_k, u, v)$ *returns TRUE, then* $(u, v) \in P_k$.
(2) *If PATH*$(k, \#P_k, u, v)$ *returns FALSE, then* $(u, v) \notin P_k$.
(3) *PATH*$(k, \#P_k, u, v)$ *uses space* $O(\log n)$.
(4) *PATH*$(k, \#P_k, u, v)$ *uses expected time* $O(n^7 \log n)$.

Proof. The correctness and space complexity of *PATH* are immediate from Lemma 3. For the expected time, note that each iteration of *PATH's* **repeat** loop uses at most n^2 calls to *WALK*, that is, expected time $O(n^7 \log n)$. The probability that a given iteration of *PATH* fails to return a value is equal to the probability that an incorrect answer is given by one or more of its invocations to *WALK*, which is at most $n^2/2n^2 = \frac{1}{2}$, by Lemma 3. Hence, the expected number of iterations is at most 2. □

We now show how to extend $\#P_{k-1}$ to $\#P_k$.

ALGORITHM *COUNT*$(k, cpkm1)$.
comment: *COUNT*$(k, \#P_{k-1})$ returns the correct value for $\#P_k$;
begin
$cpk \leftarrow 0$;
for all (u, v) **do**
if *PATH*$(k-1, cpkm1, u, v)$
or (*PATH*$(k-1, cpkm1, u, k)$ **and** *PATH*$(k-1, cpkm1, k, v)$)
then $cpk \leftarrow cpk+1$;
return *cpk*
end.

LEMMA 5. (1) *COUNT*$(k, \#P_{k-1})$ *returns* $\#P_k$.
(2) *COUNT*$(k, \#P_{k-1})$ *uses space* $O(\log n)$.
(3) *COUNT*$(k, \#P_{k-1})$ *uses expected time* $O(n^9 \log n)$.

Proof. By Lemma 4, the calls to *PATH* correctly determine whether or not $(u, v), (u, k), (k, v)$ are in P_{k-1}. Correctness follows since $(u, v) \in P_k$ if and only if $(u, v) \in P_{k-1}$ or $((u, k) \in P_{k-1}$ and $(k, v) \in P_{k-1})$. For the time complexity, there are $O(n^2)$ invocations of *PATH*, each of which runs in expected time $O(n^7 \log n)$, by Lemma 4. □

It only remains to state the main routine.

ALGORITHM *USTCON*(G, s, t).
begin
comment: Initialize $\#P_0$: $P_0 = \{(u, u)\} \cup \{(u, v) | \{u, v\} \in E\}$;
$cpk \leftarrow \#V + 2\#E$;
for k **from** 1 **to** $\#V$ **do** $cpk \leftarrow$ *COUNT*(k, cpk);
comment: *cpk* is now set to $\#P_n$;
return *PATH*$(\#V, cpk, s, t)$
end.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

LEMMA 6. (1) *USTCON*(G, s, t) returns *TRUE* if and only if $(G, s, t) \in$ *USTCON*.

(2) *USTCON* uses space $O(\log n)$.

(3) *USTCON* uses expected time $O(n^{10} \log n)$.

Proof. This follows immediately from Lemmas 4 and 5. □

It is interesting to compare the running time of the errorless algorithm (Lemma 6) to that of the version with one-sided error (Lemma 3).

2.3. An errorless algorithm for symmetric space hierarchies. As previously mentioned, *USTCON* is a complete problem for *SL*, the class of languages accepted by nondeterministic $O(\log n)$ space machines whose next move relation is symmetric. Reif [26] defined an "alternating" hierarchy based on *SL* in a manner analogous to the alternating hierarchy based on *NL* defined by Chandra, Kozen, and Stockmeyer [5]. While Immerman and Szelepcsényi's result shows that the *NL*-based hierarchy collapses to *NL*, the *SL* hierarchy may be infinite. For example, "bounded degree planarity" is in the hierarchy but is not known to be in *SL* [26]. The main result of this section is Theorem 9, which extends Theorem 1 by showing that the entire *SL* hierarchy is in *ZPLP*.

For technical reasons related to the problem of nondeterministic counting discussed in § 2.2, Reif's hierarchy is defined in terms of Turing machines with complementing moves, rather than existential and universal states as is standard for alternating machines. In Reif's complementing machines, a configuration p_0 is "accepting" if and only if there is a finite computation sequence $p_0 \vdash p_1 \vdash p_2 \cdots \vdash p_j, j \geqq 0$, with no complementing moves such that either

(1) p_j is in an accepting state, or

(2) there is at least one complementing move from p_j and for all complementing moves (p_j, p'), p' is not "accepting."

In a symmetric complementing machine, all noncomplementing moves must be symmetric. The kth level of the symmetric complementation hierarchy is

$$C\Sigma_k^{SL} = \{B \mid B \text{ is accepted by an } O(\log n) \text{ space-bounded symmetric complementing machine making at most } k-1 \text{ complement moves on any computation sequence}\}.$$

The following result is an easy modification of Theorem 5 in Ruzzo, Simon, and Tompa [28]. (See that reference for a discussion of space-bounded oracle machines.)

LEMMA 7. $\bigcup_k C\Sigma_k^{SL} \subseteq DL^{SL}$.

Proof. Consider a $C\Sigma_k^{SL}$ computation. Let $E(p)$ be the set of configurations q reachable from p using only noncomplementing moves. Note that an *SL* oracle can decide if $q \in E(p)$, since all noncomplementing moves are symmetric. Let $ACC(p, k) =$ *TRUE* if and only if there is a $p_j \in E(p)$ such that either

(1) p_j is an accepting state, or

(2) $k \geqq 1$, and there is a complementing move from p_j, and $\neg ACC(p', k-1)$ for all complementing moves (p_j, p').

If p_0 is the initial configuration, then the $C\Sigma_k^{SL}$ computation accepts if and only if $ACC(p_0, k-1) =$ *TRUE*. We determine the existence of p_j by deterministically enumerating all configurations and calling the appropriate *SL* oracle. The recursive calls on *ACC* are tested by maintaining a stack of k configurations. □

Ruzzo, Simon, and Tompa [28] use the notation $A^{\langle B \rangle}$ to denote a restricted form of relativization in which the query tape is subject to the relativized machine's space

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

bound, but the oracle receives that machine's input together with each query. This restriction is required to ensure the space bound in the following simulation.

LEMMA 8. $ZPLP^{\langle ZPLP\rangle} = ZPLP$.

Proof. Consider a relativized *ZPLP* machine M_1 that uses an oracle A accepted by a *ZPLP* machine M_2. Let w be the input to M_1. $M_1^{\langle A\rangle}$ is simulated by a *ZPLP* machine M_3. M_3 operates as if it were M_1 until M_1 invokes the oracle with some query x. M_3 then simulates M_2 on input $w\$x$. Whenever M_2 decides its output, M_3 is able to continue its simulation of M_1. If $M_1(M_2)$ runs in expected time p_1 (respectively, p_2) then the expected time of M_3 is $O(p_1(n)p_2(n+O(\log n))) = n^{O(1)}$. □

THEOREM 9. $\bigcup_k C\Sigma_k^{SL} \subseteq ZPLP$.

Proof. By Theorem 1 and Lemmas 7 and 8,

$$\bigcup_k C\Sigma_k^{SL} \subseteq DL^{SL} \subseteq DL^{ZPLP} \subseteq ZPLP^{\langle ZPLP\rangle} = ZPLP.$$

The third inclusion follows from the fact that the deterministic oracle machine can be assumed to write short queries, namely its configuration as it is about to write a long query. (See [28, Lemma 7] for more details.) □

This improves Reif's result that $\bigcup_k C\Sigma_k^{SL} \subseteq BPLP$ (defined in § 2.1).

Reif also considered the implications of his result for probabilistic parallel models. Simulation of *DL* by $O(\log n)$ time parallel models such as concurrent-read, exclusive-write, parallel random access machines (CREW-PRAMs) [10] and hardware modification machines [8] was well known. It has been observed (see, for example, Reif [25], [26]) that these simulations extend to the simulation of *RLP* and *BPLP* by one-sided and (respectively) bounded two-sided error probabilistic parallel machines. Reif noted that, as a corollary, any language in $\bigcup_k C\Sigma_k^{SL}$ can be recognized by such a probabilistic parallel machine with bounded two-sided error in $O(\log n)$ time.

Theorem 9 improves this result also, since any language in *ZPLP* can be accepted by an errorless probabilistic hardware modification machine (and thus by an errorless probabilistic CREW-PRAM) in expected time $O(\log n)$ using polynomially many processors. The simulation of an expected time $p(n)$ *ZPLP* algorithm proceeds as follows. Simulate $2p(n)$ steps of the *ZPLP* algorithm in time at most $c \log n$ using at most $(p(n))^c$ processors (for some constant c) as in Reif [25], [26]. If the simulated algorithm has not halted within that time, restart the simulation, using independently chosen random moves. The expected number of repetitions of this procedure is at most two.

Finally, an oracle hierarchy based on *SL* could be defined in the same manner as the $O\Sigma_k^L$ hierarchy of Ruzzo, Simon, and Tompa [28]. If we are careful about the definition of a symmetric oracle machine (see [16] for one possible definition), we would expect to find that, for all oracles A, $SL^{\langle A\rangle} \subseteq ZPLP^{\langle A\rangle}$ and thus by induction that $\bigcup_k O\Sigma_k^{SL} \subseteq ZPLP$. However, we have not pursued this question.

3. Semi-unbounded circuits, LOGCFL, and pebbling.

3.1. Complementation of semi-unbounded fan-in circuits. In this section we show closure under complementation of the class of languages accepted by semi-unbounded fan-in circuits.

The class of languages recognizable by size- and depth-bounded semi-unbounded fan-in circuits has been characterized in terms of several other models. The oldest is the nondeterministic auxiliary pushdown automaton of Cook [6]. Ruzzo has related space and time on such machines to space and *tree-size* of alternating Turing machines, where tree-size is the number of nodes in the smallest accepting subtree of the computation tree [27]. Venkateswaran has related them to space and alternations on

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

semi-unbounded alternating Turing machines—ones where there are no two consecutive universal configurations on any path in the computation [37]. The relation to semi-unbounded fan-in circuits follows from this. Immerman has shown relations to uniform families of short first-order formulae with a fixed number of variables and Boolean universal quantifiers—a property analogus to the semi-unbounded fan-in restriction [18]. These equivalences are summarized in the propositions below. They also generalize to larger space bounds.

PROPOSITION 10 [18], [27], [37]. *For* $T(n) = \Omega(\log n)$, *the following (uniform) complexity classes are identical.*

(1) NauxPDA Space, Time $(O(\log n), 2^{O(T(n))})$.

(2) ATM Space, Tree-size $(O(\log n), 2^{O(T(n))})$.

(3) Semi-Unbounded ATM Space, Alternations $(O(\log n), O(T(n)))$.

(4) Uniform Semi-Unbounded Fan-in Circuit Size, Depth $(2^{O(\log n)}, O(T(n)))$.

(5) Uniform Var&Sz $(B\forall)[O(1), O(T(n))]$.

PROPOSITION 11. *The equivalences in Proposition* 10 *also hold among the nonuniform versions of the models.*

In all these models, closure under complementation seems surprising. In nondeterministic auxiliary pushdown automata we face the usual problems of nondeterminism, in addition to the difficulties introduced by large stacks, and perhaps by super-polynomial running times. Although alternating Turing machine space and/or time classes are easily seen to be closed under complementation, the same proof (basically De Morgan's laws) converts a computation of small tree-size into one of large tree-size. Similar issues thwart the De Morgan approach to complementing circuits with bounded fan-in AND gates and formulate with Boolean universal variables—they become bounded fan-in OR gates and Boolean existentials instead. Nevertheless, closure under complementation follows for all these models from the theorem below.

THEOREM 12. *There is a constant* $\bar{c}$ *such that, for any n-input semi-unbounded fan-in circuit* α_n *of size (number of gates plus inputs)* Z *and depth* D, *there is a semi-unbounded fan-in circuit* $\overline{\alpha_n}$ *of size at most* $\bar{c}DZ^2 \log Z$ *and depth at most* $\bar{c}(D + \log Z)$ *that computes the complement of the function computed by* α_n. *The same result holds uniformly for uniform families of circuits* $\{\alpha_n\}$.

Proof. The key idea in the proof is again the use of inductive counting to verify "negative" information. In this case we are interested in counting the number of gates at a given depth that evaluate to 1.

Suppose we are given a circuit α_n of size Z and depth D. It is convenient to first convert it to a circuit β_n of size $2DZ + 2n$ and depth $2D$ that is:

(1) Synchronous—i.e., vertices can be assigned to levels so that input variables and their negations are on level 0, any gate on level i receives all its inputs from vertices on level $i-1$, and all output gates are on level $2D$;

(2) Fixed width—i.e., each level $i \geqq 1$ contains exactly Z gates;

(3) Strictly alternating—i.e., for all $i \geqq 0$, all gates on level $2i+1$ are OR gates and all gates on level $2i+2$ are AND gates; and

(4) Equivalent—i.e., it computes the same function as α_n.

This normal form is easily achieved. For example, for each level of β_n except the 0th, make a replica of each vertex of α_n. The replica on level i of an input vertex g is a trivial gate (of the appropriate type) whose only input is the replica of g on level $i-1$. Similarly, the replica of an AND gate g on an OR level i is a trivial OR gate whose only input is the replica of g on level $i-1$. A replica of an AND gate g on an AND level i has as inputs the replicas on level $i-1$ of g's inputs. OR gates are handled similarly, but with nontrivial replicas only on OR levels. Level 0 contains only the $2n$

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

vertices representing the input literals. Level 1 replicas of gates whose inputs are not all input literals are arbitrarily connected to literals. We can show by induction that all vertices of depth at most i in α_n will be correctly computed by their replicas on levels greater than or equal to $2i$ in β_n.

The "counting" we need in the construction is easily accomplished by threshold functions. Thus, rather than producing the circuit advertised by the theorem directly, it is easier to first produce a circuit γ_n containing THRESHOLD gates. (A c-THRESHOLD gates has unbounded fan-in, and outputs 1 if and only if at least c of its inputs have value 1.) This circuit will have the following properties:

(1) It has bounded fan-in AND gates and unbounded fan-in OR gates, i.e., it has semi-unbounded fan-in.

(2) It contains arbitrary THRESHOLD gates.

(3) No path from output to input contains more than two THRESHOLD gates.

(4) It has size $4D(Z+1)^2+2DZ+2n-1=O(DZ^2)$.

(5) It has depth at most $2D+3$.

(6) It computes the complement of the function computed by β_n.

The theorem will then follow by replacing the THRESHOLD gates by monotone SAC^1 threshold circuits. (Monotonicity is needed to preserve the semi-unbounded fan-in property. By property (3) above, the depth of the threshold subcircuits will increase the overall depth only additively.)

For $0 \leqq k \leqq D$, let

$$P_k = \{g \mid g \text{ is a vertex of } \beta_n \text{ on level } 2k \text{ having value } 1\}.$$

The main quantities we will be interested in counting are $\# P_k$, the cardinalities of the sets P_k.

Our main construction, of γ_n from β_n, follows. It is sketched in Fig. 2.

Construction Step 1. The entire circuit β_n is a subcircuit of γ_n. The gates in this subcircuit will be referred to as "original" gates. (Note that this means original to β_n, not to α_n.)

Construction Step 2. For $1 \leqq k \leqq D$, each original gate g on level $2k-1$ or $2k$, and each $0 \leqq c \leqq Z$, there is a "contingent complement" gate $\text{CC}(g \mid c)$ whose value will be the complement of the value of g *if* $c = \# P_{k-1}$; if $c \neq \# P_{k-1}$, then the value of $\text{CC}(g \mid c)$ is irrelevant. We compute $\text{CC}(g \mid c)$ as follows:

(1) If g is an AND gate, say $\text{AND}(a, b)$, then $\text{CC}(g \mid c)$ is the OR of $\text{CC}(a \mid c)$ and $\text{C}(b \mid c)$. (Fan-in greater than two is handled similarly.)

(2) If g is an OR gate (on level $2k-1$), then $\text{CC}(g \mid c)$ is a c-THRESHOLD gate whose inputs are all original gates on level $2k-2$ that are *not* inputs to g.

We now argue the correctness of this part of the construction.

LEMMA 13. *For* $k \geqq 1$, *and all original gates* g *on level* $2k-1$ *or* $2k$,

$$\text{CC}(g \mid \# P_{k-1}) \equiv \neg g.$$

Proof.

Case 1. g is an OR gate on level $2k-1$, $k \geqq 1$. Then g evaluates to 0 if and only if all g's inputs evaluate to 0, if and only if all $\# P_{k-1}$ original gates on level $2(k-1)$ that evaluate to 1 are *not* inputs to g, if and only if the THRESHOLD gate $\text{CC}(g \mid \# P_{k-1})$ evaluates to 1.

Case 2. g is an AND gate on level $2k$, $k \geqq 1$. Its inputs a and b are OR gates on level $2k-1$. From Case 1 we know that $\text{CC}(a \mid \# P_{k-1}) \equiv \neg a$ and $\text{C}(b \mid \# P_{k-1}) \equiv \neg b$, so

$$\neg g \equiv \neg(a \wedge b) \equiv \neg a \vee \neg b \equiv \text{CC}(a \mid \# P_{k-1}) \vee \text{CC}(b \mid \# P_{k-1}) \equiv \text{CC}(g \mid \# P_{k-1}).$$

The proof for fan-in greater than two is analogous. □

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

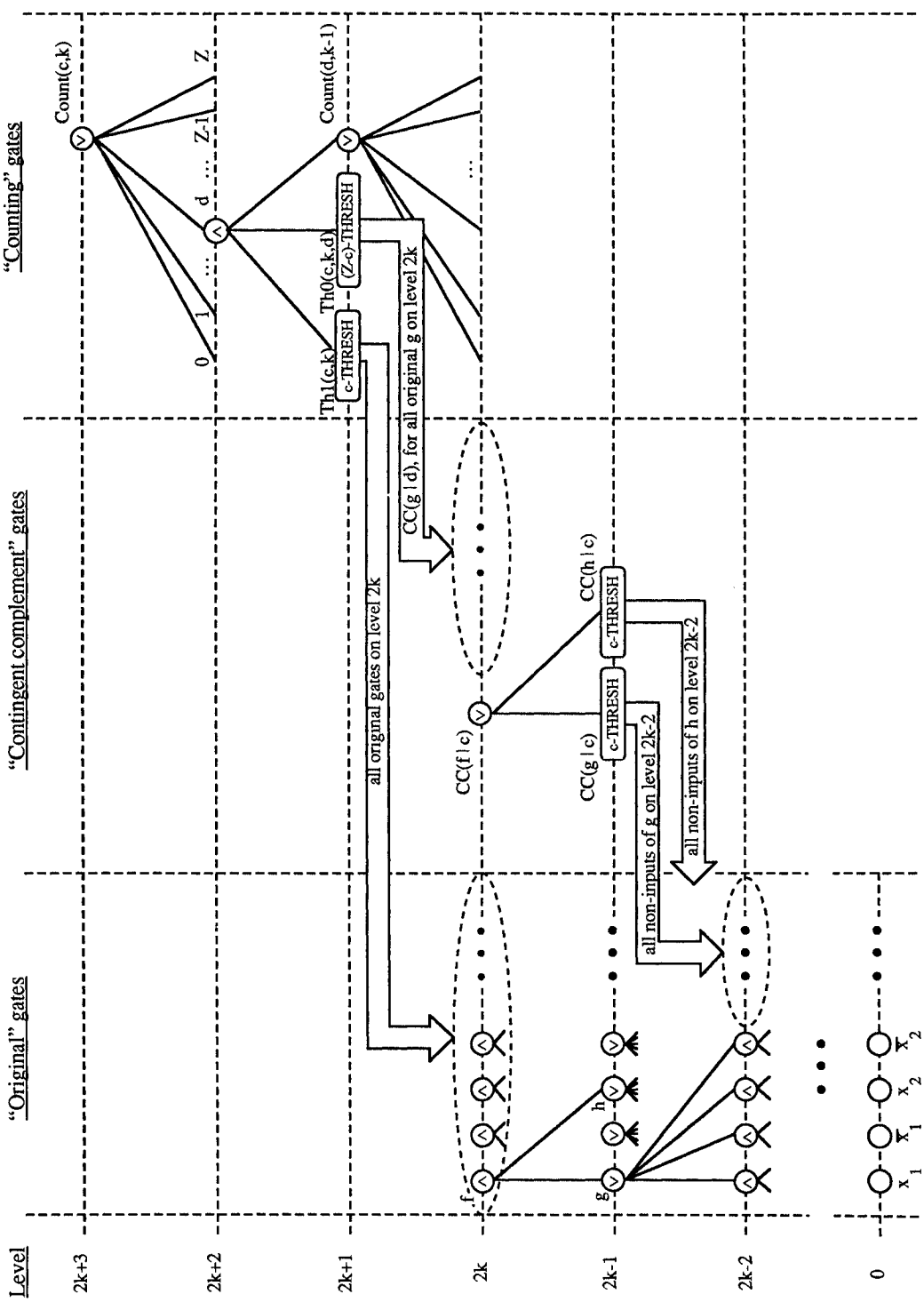

FIG. 2. *Construction of* γ_n.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Construction Step 3. Continuing the construction, there is a "counting" gate COUNT(c, k), for all $0 \leqq c \leqq Z$ and $0 \leqq k \leqq D-1$, whose value will be 1 if and only if $\#P_k = c$. This is computed as follows:

$$\text{COUNT}(c,k) = \begin{cases} 1 \Leftrightarrow (c = n) & \text{if } k = 0, \\ \bigvee_{d=0}^{Z} \text{AND}(\text{COUNT}(d, k-1), \text{TH1}(c,k), \text{TH0}(c,k,d)) & \text{if } k \geqq 1, \end{cases}$$

where

TH1(c, k) is the c-THRESHOLD of all original gates on level $2k$, and
TH0(c, k, d) is the $(Z-c)$-THRESHOLD of the contingent complement gates CC$(g \mid d)$, where g ranges over all original gates on level $2k$.

LEMMA 14. COUNT$(c, k) = 1$ *if and only if* $\#P_k = c$.

Proof. The proof proceeds by induction on k.

Basis $(k = 0)$. There are exactly $2n$ vertices on level 0, namely the n inputs and their complements. Exactly n of them evaluate to 1.

Induction $(k > 0)$. By induction, COUNT$(d, k-1)$ evaluates to 1 for exactly one value of d, namely $d = \#P_{k-1}$. Thus the only term in the disjunction $\bigvee_{d=0}^{Z} \cdots$ that could possibly evaluate to 1 is $d = \#P_{k-1}$. For this term, each contingent complement gate CC$(g \mid d)$ counted by TH0(c, k, d) computes $\neg g$ by Lemma 13. Thus, TH1$(c, k) \wedge$ TH0(c, k, d) evaluates to 1 if and only if, on level $2k$, there are at least c original gates with value 1 and at least $(Z-c)$ original gates with value 0, and hence exactly c with value 1. Thus, COUNT(c, k) evaluates to 1 if and only if $c = \#P_k$. □

Construction Step 4. We complete the construction by defining the outputs of γ_n. For all original gates g that are outputs of β_n (hence on level $2D$), γ_n contains a gate COMP(g) that evaluates to 1 if and only if g evaluates to 0. COMP(g) is computed as follows:

$$\text{COMP}(g) = \bigvee_{c=0}^{Z} \text{AND}(\text{COUNT}(c, D-1), \text{CC}(g \mid c)).$$

Correctness is easily shown. By Lemma 14, COUNT$(c, D-1)$ evaluates to 1 if and only if $\#P_{D-1} = c$, and by Lemma 13, CC$(g \mid \#P_{D-1}) \equiv (\neg g)$. Hence COMP$(g) \equiv (\neg g)$.

Thus γ_n correctly computes the complement of the function computed by α_n.

Analysis. Next we will analyze the size and depth of γ_n.

Define the THRESHOLD-*depth* of a gate g of γ_n to be the maximum number of THRESHOLD gates, including g, along any path from g to an input vertex.

For each original gate g on level k, g is also at depth k in γ_n, and has THRESHOLD-depth 0. For all $0 \leqq c \leqq Z$, gate CC$(g \mid c)$ also has depth k, and THRESHOLD-depth 1, since the THRESHOLD gates among CC$(g \mid c)$ depend *only* on original gates, not on other contingent complement gates.

The gates TH1(c, k) have depth $2k+1$ and THRESHOLD-depth 1. The gates TH0(c, k, d) have depth $2k+1$ and THRESHOLD-depth 2.

The gates COUNT(c, k) have THRESHOLD-depth 2 and depth d_k, where

$$d_k = \begin{cases} 0 & \text{if } k = 0, \\ 2 + \max(d_{k-1}, 2k+1) & \text{if } k \geqq 1. \end{cases}$$

Thus $d_k \leqq 2k+3$.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Finally, the output gates COMP(g) have THRESHOLD-depth 2 and depth $2+\max(d_{D-1}, 2D) \leqq 2D+3$.

The size of the circuit is easily seen to be polynomial in Z and D. The dominant contributions are from the contingent complement subcircuits, which contain $O(DZ^2)$ OR and THRESHOLD gates, and from the COUNT subcircuits, which contain $O(DZ^2)$ AND and THRESHOLD gates. A careful analysis gives the exact bound of $4D(Z+1)^2+2DZ+2n-1$ claimed above.

To complete the proof of Theorem 12, we need to replace the THRESHOLD gates in γ_n by monotone SAC^1 threshold circuits, and analyze the size and depth of the resulting Boolean circuit $\overline{\alpha_n}$.

The existence of monotone SAC^1 threshold circuits is easily established. A simple divide-and-conquer technique gives n input monotone SAC^1 circuits of size $O(n^2)$ and depth at most $2\lceil \log_2 n \rceil$: the k-THRESHOLD of n bits can be computed as the OR over $0 \leqq j \leqq k$ of the AND of the j-THRESHOLD of the first half of the bits and the $(k-j)$-THRESHOLD of the last half of the bits:

$$k\text{-THRESHOLD}(x_1, \cdots, x_n)$$

$$= \bigvee_{j=0}^{k} \text{AND}(j\text{-THRESHOLD}(x_1, \cdots, x_{\lfloor n/2 \rfloor}),$$

$$(k-j)\text{-THRESHOLD}(x_{\lfloor n/2 \rfloor+1}, \cdots, x_{2n})).$$

Asymptotically, the smallest known monotone SAC^1 threshold circuits are actually NC^1 circuits: the $O(n \log n)$ size "AKS" sorting networks of Ajtai, Komlós, and Szemerédi [1]. Replacing each THRESHOLD gate in γ_n by one of these subcircuits, and noting that each THRESHOLD gate has at most Z inputs, would give an overall size for the Boolean circuit $\overline{\alpha_n}$ of $O(DZ^3 \log Z)$.

One observation reduces this substantially. Namely, a *single* n-input AKS network computes the c-threshold of its inputs for *all* $1 \leqq c \leqq n$. Thus, although an OR gate g gives rise to Z THRESHOLD gates CC($g \mid c$), $1 \leqq c \leqq Z$, all of these values can be computed by one AKS network. Similar combination is possible among the TH0($c, -, -$) and TH1($c, -$) gates, $1 \leqq c \leqq Z$. This reduces the size of $\overline{\alpha_n}$ to $O(DZ^2 \log Z)$, as claimed.

The depth of $\overline{\alpha_n}$ is the depth of γ_n plus twice the depth of the AKS network (since γ_n has THRESHOLD-depth 2), which is $O(D+\log Z)$, as claimed.

For the uniform case of the theorem, we observe that the transformation from α_n to $\overline{\alpha_n}$ is quite simple and regular. We leave it to the reader to verify that this transformation preserves uniformity. (The AKS networks are known to be uniform.) □

COROLLARY 15. *For all $k \geqq 1$, SAC^k and nonuniform SAC^k are closed under complementation.*

Cook [7] defined *CFL** as the set of functions each computable by a uniform family $\{\alpha_n\}$ of circuits, where α_n has n inputs, bounded fan-in AND, OR, and NOT gates, unbounded fan-in oracle gates for some context-free language, and $O(\log n)$ depth. An oracle gate with fan-in f is defined to contribute $\lceil \log_2 f \rceil$ to the depth of any path containing it.

COROLLARY 16. *Any function in CFL* can be computed by a uniform family of polynomial size, $O(\log n)$ depth, semi-unbounded fan-in circuits.*

Proof. Let α_n be a *CFL** circuit with oracle gates for some context-free language L. With a doubling of size and no increase in depth, α_n can be simulated by a *CFL**

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

circuit β_n whose NOT gates appear only at the inputs, provided β_n is allowed oracle gates for both L and $\bar{L}$. (See, for example, [13].) The result now follows from the fact that LOGCFL $= SAC^1$ [37], together with Corollary 15. □

Similarly, it is true that any 0-1 valued function in NL^* (see Cook [7]) is also in NL. This follows from Immerman's [17] and Szelepcsényi's theorems [33], and was noted independently by S. Buss (personal communication).

3.2. The weakness of role switches in pebbling. There are a number of ways in which we might define a "*LOGCFL* hierarchy." One consequence of Corollary 15 is that, for many reasonable ways of doing so, the resulting hierarchy collapses to *LOGCFL*. In this section we consider one such hierarchy that collapses. As a consequence, the "role switch" resource [38] in pebbling is shown to be much weaker than previously seemed plausible.

(In contrast, following a preliminary version of the present paper, Jenner and Kirsig [19] considered an alternative formulation of a hierarchy based on *LOGCFL*, showing that it coincides with the polynomial hierarchy and hence presumably does *not* collapse.)

We begin by presenting the hierarchy. Define a (z, d, k, f)-*circuit* as an unbounded fan-in circuit of size z and depth d, where negations appear only at the inputs, and the vertices can be partitioned into k "layers" that alternately have AND fan-in at most f and OR fan-in at most f. More precisely, the vertices can be partitioned into blocks $B_k, B_{k-1}, \cdots, B_1$ (B_1 containing the outputs) such that:

(1) Any wire (u, v) with $u \in B_i$ and $v \in B_j$ satisfies $i \geqq j$;

(2) All AND gates of odd-numbered blocks have fan-in at most f; and

(3) All OR gates of even-numbered blocks have fan-in at most f.

For any fixed k, let

$$\Sigma_k^{CFL} = \{L \mid L \text{ can be recognized by a (nonuniform) family of } (n^{O(1)}, O(\log n), k, O(1))\text{-circuits}\}.$$

For instance, Σ_1^{CFL} is nonuniform SAC^1. Corollary 18 demonstrates that this hierarchy collapses.

THEOREM 17. *Let $\{\alpha_n\}$ be a family of $(z, d, k, O(1))$-circuits, where z, d, and k may be functions of n. Then there is a family $\{\beta_n\}$ of $(O(dz^2 \log z), O(d + k \log z), 1, O(1))$-circuits such that β_n computes both the outputs of α_n and their negations.*

Proof. This is proved by induction on k, using Theorem 12 and De Morgan's laws in a straightforward way. □

COROLLARY 18. *For any fixed $k \geqq 1$, $\Sigma_k^{CFL} = \Sigma_1^{CFL}$.*

Proof. This follows from Theorem 17 by substituting $z = n^{O(1)}$ and $d = O(\log n)$. □

To see how much more general Theorem 17 is than Corollary 18, consider the case $z = n^{O(1)}$ and $d = O(\log^i n)$, for any $i \geqq 1$, which might be called the "SAC^i hierarchy." Not only does $k = \Theta(1)$ fail to add power to nonuniform SAC^i, but $k = \Theta(\log^{i-1} n)$ does as well.

The purpose of this section is to apply this result to a pebble game introduced by Venkateswaran and Tompa [38]. They defined a new resource called "role switches." It will follow from Theorems 19 and 20 that Σ_k^{CFL} is the set of languages recognized by polynomial-size circuits pebbleable with $O(1)$ pebbles, $O(\log n)$ rounds, and $k-1$ role switches. Hence, by Corollary 18, any such circuit can be simulated by one pebbleable with $O(1)$ pebbles, $O(\log n)$ rounds, and zero role switches. Similarly, any

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

polynomial-size circuit pebbleable with $O(1)$ pebbles, $O(\log^i n)$ rounds, and $O(\log^{i-1} n)$ role switches can be simulated by one pebbleable with $O(1)$ pebbles, $O(\log^i n)$ rounds, and zero role switches.

The pebble game introduced by Venkateswaran and Tompa is referred to as the *dual interpreted two-person pebble game.* This game is played by two players, called Player 0 and Player 1, on a circuit α_n together with values for its n inputs and their negations (referred to collectively as *literals*). There are two minor differences between Venkateswaran and Tompa's game and the one used in this section: for convenience, we assume that α_n is *nonuniform* and has *unbounded* fan-in. The latter condition does not affect the resources considered, provided the depth of α_n is $\Omega(\log n)$.

At any point, one of the players takes on the role of the Challenger and the other that of the Pebbler. The Challenger is responsible for selecting the "currently challenged vertex"; the Pebbler has a collection of pebbles that it can place on or remove from the vertices of α_n. The role of a player is automatically determined as part of the circuit information as follows. The vertices in α_n are partitioned into two sets, those of "challenge type" 0 and those of "challenge type" 1. If the currently challenged vertex is of challenge type 0 (challenge type 1), then Player 0 (Player 1) is the Challenger in the next round. A Boolean circuit augmented with this role information for each of its vertices will be referred to as an *augmented* circuit. For convenience, it is assumed that the output vertex has challenge type 0.

The objective of Player 0 (Player 1) is to establish that the output of the circuit evaluates to 0 (1). A pebble placement or challenge on a vertex v by Player 0 (Player 1) corresponds to asserting that v evaluates to 0 (1). A pebble placed by Player 0 (Player 1) will be referred to as a 0-pebble (1-pebble).

The initial challenge is on the output vertex. The game proceeds in rounds, with a round consisting of the following three parts. (a) If the game is not over at the currently challenged vertex u according to the conditions below, then Player 0 is the Challenger for this round if u is of challenge type 0 and the Pebbler otherwise. (b) In the *pebbling move*, the Pebbler picks up zero or more of its own pebbles from vertices already pebbled and places pebbles on any nonempty set of vertices. (c) In the *challenging move*, the Challenger either rechallenges the currently challenged vertex or challenges one of the vertices that acquired a pebble in the current round.

Player 1 wins the game if, immediately following the Challenger's move, the currently challenged vertex is a literal with value 1, or an OR gate at least one of whose immediate predecessors is 1-pebbled, or an AND gate all of whose immediate predecessors are 1-pebbled. Player 0 wins if, immediately following the Challenger's move, the currently challenged vertex is a literal with value 0, or an OR gate all of whose immediate predecessors are 0-pebbled, or an AND gate at least one of whose immediate predecessors is 0-pebbled. It is also possible to have a winner in an infinite play of the game, namely that player (if either) who is the Pebbler in only finitely many rounds. (The purpose of this last rule is to force each player to make progress as the Pebbler.)

These notions are defined more precisely below. Fix an augmented circuit α_n and its input x.

A *configuration* of the game is a tuple $(t, P_1, P_0, R_1, R_0, v)$, where
$t \in \{P, C\}$ indicates whether it is the Pebbler's or the Challenger's turn to move,
P_1 is the set of vertices with 1-pebbles on them from previous rounds,
P_0 is the set of vertices with 0-pebbles on them from previous rounds,
R_1 is the set of vertices 1-pebbled in the current round,
R_0 is the set of vertices 0-pebbled in the current round, and
v is the currently challenged vertex.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

The initial configuration of the game is $(P, \varnothing, \varnothing, \varnothing, \varnothing, s)$, where s is the output of the circuit.

A configuration $(P, P_1, P_0, \varnothing, \varnothing, v)$ is *terminal* if v is a literal, or an OR (AND) gate with some (all) of its immediate predecessors in P_1 or all (some) of its immediate predecessors in P_0.

A *move* in the game is made in accordance with a binary relation $\vdash$ on configurations defined as follows (where P_0, P_1, S_0, and S_1 are arbitrary sets of vertices, and R_0 and R_1 are arbitrary nonempty sets of vertices):

$(P, P_1, P_0, \varnothing, \varnothing, v) \vdash (C, P_1 - S_1, P_0, R_1, \varnothing, v)$, for all configurations $(P, P_1, P_0, \varnothing, \varnothing, v)$ that are not terminal and where v is of challenge type 0,

$(P, P_1, P_0, \varnothing, \varnothing, v) \vdash (C, P_1, P_0 - S_0, \varnothing, R_0, v)$, for all configurations $(P, P_1, P_0, \varnothing, \varnothing, v)$ that are not terminal and where v is of challenge type 1,

$(C, P_1, P_0, R_1, \varnothing, v) \vdash (P, P_1 \cup R_1, P_0, \varnothing, \varnothing, v')$, for all $v' \in R_1 \cup \{v\}$,

$(C, P_1, P_0, \varnothing, R_0, v) \vdash (P, P_1, P_0 \cup R_0, \varnothing, \varnothing, v')$, for all $v' \in R_0 \cup \{v\}$.

The *game tree* T is a maximal rooted tree whose nodes are labeled by configurations of the game, whose root is labeled by the initial configuration, and whose edge relation is given by $\vdash$. Note that the leaves of the tree are labeled by terminal configurations.

A *finite play* of the game is a finite path in the game tree from the root to some leaf labeled by the terminal configuration $(P, P_1, P_0, \varnothing, \varnothing, v)$. It is a *winning finite play for Player* 1 if v is a literal with value 1, or if v is an OR gate at least one of whose immediate predecessors is in P_1, or if v is an AND gate all of whose immediate predecessors are in P_1; otherwise it is a *winning finite play for Player* 0. An infinite path Π in the game tree is a *winning infinite play* for the player (if either) that is the Pebbler in only finitely many configurations on Π.

A *winning strategy for Player* 1 (if it exists) is a subtree W of T such that:

(1) W contains the root of T;

(2) W contains exactly one child of every nonterminal node in W that is labeled by a configuration in which it is Player 1's turn to move;

(3) W contains all children of every nonterminal node in W that is labeled by a configuration in which it is Player 0's turn to move; and

(4) All paths in W are winning (finite or infinite) plays for Player 1.

A *winning strategy for Player* 0 is defined dually.

Let $\{\alpha_n\}$ accept the language L, where each member α_n of the family is an augmented circuit with n inputs. The game on α_n with input $x \in L \cap \{0,1\}^n$ can be *played simultaneously in* $p(n)$ *space,* $r(n)$ *rounds, and* $s(n)$ *role switches* if and only if there is a winning strategy for Player 1 in which:

(1) Every pebbling configuration $(P, P_1, P_0, \varnothing, \varnothing, v)$ along every path satisfies $|P_1| \leqq p(n)$,

(2) On any path, there are at most $r(n)$ edges $(P, P_1, P_0, \varnothing, \varnothing, v) \vdash (C, P_1 - S_1, P_0, R_1, \varnothing, v)$ with v or challenge type 0, and

(3) On any path, there are at most $s(n)$ edges $(C, P_1, P_0, R_1, R_0, v) \vdash (P, P_1', P_0', \varnothing, \varnothing, v')$ having the challenge types of v and v' unequal.

Resources on inputs $x \notin L$ are defined dually, considering winning strategies for Player 0 in place of those for Player 1.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Finally, α_n is *pebbleable* simultaneously in $p(n)$ space, $r(n)$ rounds, and $s(n)$ role switches if and only if, for all x of length n, the game on α_n can be played simultaneously in $p(n)$ space, $r(n)$ rounds, and $s(n)$ role switches. Note that only the resources used by the winning player are counted.

Theorems 19 and 20 demonstrate the intimate relationship between layered circuits and pebbling.

THEOREM 19. *Suppose* $\{\alpha_n\}$ *is a family of* $(z, r, s+1, p)$*-circuits, where* z, r, s, *and* p *may be functions of* n. *Then there is an assignment of challenge types to the vertices of* α_n *such that* α_n *is pebbleable simultaneously in* p *space,* r *rounds, and* s *role switches.*

Proof. Any vertex in a layer with AND fan-in (OR fan-in) bounded by p is assigned challenge type 0 (respectively, 1). Suppose α_n outputs 1 on input x. A winning strategy for Player 1 follows from the claim below. The winning strategy for Player 0 when α_n outputs 0 is dual.

CLAIM. Let v be the currently challenged vertex of α_n. Suppose v evaluates to 1 on input x, no predecessor of v is 0-pebbled, and the subcircuit induced by v and its predecessors is a $(z', r', s'+1, p')$-circuit. Then Player 1 can win the game with p' pebbles, r' rounds, and s' role switches.

The claim is proved by induction on r'. The basis $r' = 0$ is immediate. The induction depends on the role of Player 1, as follows.

Case 1. The challenge type of v is 0; i.e., v is in a layer with bounded AND fan-in. Then Player 1 is the Pebbler, and begins by removing all 1-pebbles from the circuit.

Case 1.1. v is an OR gate. Then Player 1 pebbles any one immediate predecessor u of v that evaluates to 1. (Note that Player 1 does not lose immediately, as no predecessors of u are 0-pebbled.) Player 0 must move the challenge to u in order not to lose immediately. If the challenge type of u is 1, then a role switch occurs and s' decreases by at least 1. In any case, the claim now follows from the induction hypothesis.

Case 1.2. v is an AND gate with bounded fan-in. Then Player 1 pebbles every immediate predecessor of v. The claim follows as in Case 1.1.

Case 2. The challenge type of v is 1. Then Player 1 is the Challenger. If Player 0 never pebbles a predecessor of v that evaluates to 1, Player 1 retains its challenge on v and wins using no resources. Suppose Player 0 pebbles a predecessor of v that evaluates to 1. Consider the first such move. Let u be such a predecessor of minimum depth among the vertices that are pebbled. Player 1 moves its challenge to u. Notice that no predecessor of u is 0-pebbled. The depth of the challenged vertex is decreased without Player 1 using any resources (with the possible exception of a role switch), so the result again follows from the induction hypothesis. □

THEOREM 20. *Suppose a family* $\{\alpha_n\}$ *of augmented circuits of size* z *is pebbleable simultaneously in* p *space,* r *rounds, and* s *role switches, where* z, p, r, *and* s *may be functions of* n. *Then there is a family* $\{\beta_n\}$ *of* $((r+1)^2(s+1)z^{O(p)}, 4r+5, s+1, p+1)$*-circuits that recognizes the same language as* $\{\alpha_n\}$.

Proof. Construction. β_n has one vertex $g(A, r_1, r_0, \hat{s})$ for every $0 \leqq r_1, r_0 \leqq r$, every $0 \leqq \hat{s} \leqq s$, and every configuration $A = (t, P_1, P_0, R_1, R_0, v)$ with $\#(P_1 \cup R_1) \leqq p$ and $\#(P_0 \cup R_0) \leqq p$. $r_1(r_0)$ will be used to count the number of rounds in which Player 1 (respectively, 0) has been Pebbler, and $\hat{s}$ will be used to count the number of role switches that have occurred. If $A = (t, P_1, P_0, R_1, R_0, v)$ is terminal with v a literal x_j, then $g(A, r_1, r_0, \hat{s})$ is the literal x_j. If $A = (t, P_1, P_0, R_1, R_0, v)$ is terminal with appropriate immediate predecessors of the challenged gate v in P_i, then $g(A, r_1, r_0, \hat{s})$ is the constant i. Otherwise $g(A, r_1, r_0, \hat{s})$ is a gate of type OR (AND) if and only if in A it is the turn of Player 1 (respectively, 0). The output gate is $g(I, 0, 0, 0)$, where I is the initial configuration on α_n. Let $g(A, r_1, r_0, \hat{s})$ be a gate of β_n, where $A =$

$(t, P_1, P_0, R_1, R_0, v)$ and let $A' = (t', P_1', P_0', R_1', R_0', v')$ satisfy $A \vdash A'$. For $i \in \{0, 1\}$, let

$$r_i' = \begin{cases} r_i + 1, & \text{if } t = P \text{ and } v \text{ is of challenge type } 1 - i, \\ r_i, & \text{otherwise,} \end{cases}$$

$$\hat{s}' = \begin{cases} \hat{s} + 1, & \text{if the challenge types of } v \text{ and } v' \text{ are unequal,} \\ \hat{s}, & \text{otherwise.} \end{cases}$$

If $r_i' > r$ or $\#(P_i' \cup R_i') > p$, then there is a wire from the constant $1 - i$ to $g(A, r_1, r_0, \hat{s})$ (indicating that Player $1 - i$ must win from A', since the winner never exceeds its resources). If $\hat{s}' > s$, there is a wire from an arbitrary constant to $g(A, r_1, r_0, \hat{s})$. Otherwise, there is a wire from $g(A', r_1', r_0', \hat{s}')$ to $g(A, r_1, r_0, \hat{s})$.

Correctness. Suppose α_n evaluates to 1 on x. The fact that β_n evaluates to 1 on x is shown by arguing that there is an "accepting subcircuit" S of β_n, that is, a set of vertices including the output all of which evaluate to 1. (The proof when α_n evaluates to 0 is dual.)

Since α_n evaluates to 1 on x, there is a winning strategy W for Player 1 in which Player 1 uses at most p pebbles, r rounds, and s role switches. By construction, there is a corresponding subcircuit S of β_n that contains the output, and in which one immediate predecessor of each OR gate in S and all immediate predecessors of each AND gate in S are also in S. The major difference between W and S is that some plays (in particular, all infinite plays) in W are truncated in S due to the conditions $r_0 > r$ or $\#(P_0 \cup R_0) > p$. (Neither $r_1 > r$ nor $\#(P_1 \cup R_1) > p$ nor $\hat{s} > s$ ever occurs in S, since Player 1's strategy uses at most r rounds, at most p pebbles, and at most s role switches.) All literals or constants of S that correspond to leaves of W evaluate to 1, since Player 1 wins at all leaves of W. Furthermore, any literal or constant of S that does not correspond to a leaf of W is the constant 1 by construction, since it arises from a truncation due to $r_0 > r$ or $\#(P_0 \cup R_0) > p$. Thus, β_n outputs 1.

Analysis. The size bound of β_n follows from the fact that the number of configurations of the pebble game is $z^{O(p)}$. The depth bound of $4r + 5$ follows from the fact that either r_0 or r_1 increases each round, and there are two moves (one for each player) per round. The fan-in bound follows from the fact that, whenever it is the turn of Player 0 (1) who is the Challenger in A, the AND fan-in (respectively, OR fan-in) of $g(A, r_1, r_0, \hat{s})$ is at most $p + 1$, since only $p + 1$ configurations (corresponding to the possibilities for the next challenged vertex) follow from A by a move of the Challenger. Finally, there is one layer for each of the $s + 1$ values of $\hat{s}$, since the same player remains Challenger as long as $\hat{s}$ remains unchanged. □

Let *ROUNDS, SWITCHES*$(r(n), s(n))$ denote the set of languages each of which is accepted by a nonuniform family of polynomial-size circuits pebbleable simultaneously with $O(1)$ pebbles, $r(n)$ rounds, and $s(n)$ role switches.

COROLLARY 21.

$$\textit{ROUNDS, SWITCHES}(r(n), s(n)) \subseteq \textit{ROUNDS, SWITCHES}(O(r(n) + s(n) \log n), 0).$$

Proof. This follows from Theorems 17, 19, and 20, noting for the size bound that $r(n)$ and $s(n)$ are $n^{O(1)}$. □

The final corollary states an explicit threshold beyond which role switches appear to add power. Equations (1) and (3) are the nonuniform analogues of Theorems 4 and 5 in [38].

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

COROLLARY 22. *For any* $i \geq 1$,

(1) $\quad$ *nonuniform* $SAC^i = ROUNDS, SWITCHES(O(\log^i n), 0)$

(2) $\quad = ROUNDS, SWITCHES(O(\log^i n), O(\log^{i-1} n))$;

(3) $\quad$ *nonuniform* $AC^i = ROUNDS, SWITCHES(O(\log^i n), O(\log^i n))$.

Proof. Equation (1) follows from Theorems 19 and 20, since nonuniform SAC^i is, by definition, the set of languages each of which is accepted by a family of $(n^{O(1)}, O(\log^i n), 1, O(1))$-circuits. Equation (3) follows by the same argument, since nonuniform AC^i is characterized similarly by $(n^{O(1)}, O(\log^i n), O(\log^i n), O(1))$-circuits. Equation (2) follows from Corollary 21. □

4. Open problems. Figure 1 indicates the relationships among space-bounded complexity classes between NC^1 and NC^2. Since it is still possible that $NC^1 = NC^2$ (or indeed $NC^1 = NP$), there are no proven proper inclusions or incomparability results among these classes.

In addition to the problems identified by Cook [7], we call attention to certain questions suggested by this paper:

(1) Is *SL* closed under complementation? If so, then the $\bigcup_k C\Sigma_k^{SL}$ hierarchy collapses to *SL*.

(2) Is *RLP* closed under complementation? If so, then $RLP = ZPLP$. If not, what is a new candidate for a language in *RLP* (or *BPLP*) that is not in *ZPLP*?

(3) Assuming that $RLP \neq NL = RL = ZPL$, we see that the expected polynomial-time bound is important in the case of errorless and one-sided error probabilistic $O(\log n)$ space computations, whereas $PL = PLP$ in the case of two-sided unbounded error. The case of two-sided bounded error remains open; that is, is $BPL = BPLP$? Is there a candidate for a language in *BPLP* (or *BPL*) that is not in *NL*?

It seems surprising that the NC^1 AKS networks [1] provide the best known SAC^1 networks for Boolean sorting. An interesting question is whether we could exploit the availability of unbounded fan-in OR gates to get a simpler $O(n \log n)$ size monotone Boolean sorter, and/or one with a more favorable constant hidden in the big-*O*. Indeed, size $o(n \log n)$ is not out of the question for this model. See Friedman [11] and Valiant [35] for other recent approaches to threshold computation.

REFERENCES

[1] M. AJTAI, J. KOMLÓS, AND E. SZEMERÉDI, *Sorting in c* log *n parallel steps*, Combinatorica, 3 (1983), pp. 1-19.

[2] R. ALELIUNAS, R. M. KARP, R. J. LIPTON, L. LOVÁSZ, AND C. RACKOFF, *Random walks, universal traversal sequences, and the complexity of maze problems*, in 20th Annual IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, October 1979, pp. 218-223.

[3] P. BILLINGSLEY, *Probability and Measure*, John Wiley, New York, 1979.

[4] S. R. BUSS, S. A. COOK, P. W. DYMOND, AND L. HAY, *The log space oracle hierarchy collapses*, Tech. Report CS103, Department of Computer Science and Engineering, University of California, San Diego, CA, 1987.

[5] A. K. CHANDRA, D. C. KOZEN, AND L. J. STOCKMEYER, *Alternation*, J. Assoc. Comput. Mach., 28 (1981), pp. 114-133.

[6] S. A. COOK, *Characterizations of pushdown machines in terms of time-bounded computers*, J. Assoc. Comput. Mach., 18 (1971), pp. 4-18.

[7] ———, *A taxonomy of problems with fast parallel algorithms*, Inform. and Control, 64 (1985), pp. 2-22.

[8] P. W. DYMOND AND S. A. COOK, *Hardware complexity and parallel computation*, in 21st Annual IEEE Symposium on Foundations of Computer Science, Syracuse, NY, October 1980, pp. 360-372.

[9] R. W. FLOYD, *Algorithm 97: shortest path*, Comm. ACM, 5 (1962), p. 345.

[10] S. FORTUNE AND J. WYLLIE, *Parallelism in random access machines*, in Proc. 10th Annual ACM Symposium on Theory of Computing, San Diego, CA, May 1978, pp. 114-118.

Downloaded 07/08/13 to 128.100.3.42. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

[11] J. FRIEDMAN, *Constructing $O(n \log n)$ size monotone formulae for the k-th elementary symmetric polynomial of n Boolean variables*, in 25th Annual IEEE Symposium on Foundations of Computer Science, Singer Island, FL, October 1984, pp. 506–515.

[12] J. GILL, *Computational complexity of probabilistic Turing machines*, SIAM J. Comput., 6 (1977), pp. 675–695.

[13] L. M. GOLDSCHLAGER, *The monotone and planar circuit value problems are log space complete for P*, SIGACT News, 9 (1977), pp. 25–29.

[14] J. HARTMANIS, *The structural complexity column*, Bull. European Assoc. Theoret. Comput. Sci., 33 (1987), pp. 26–39.

[15] L. A. HEMACHANDRA, *The strong exponential hierarchy collapses*, in Proc. 19th Annual ACM Symposium on Theory of Computing, New York, NY, May 1987, pp. 110–122.

[16] R. R. HOWELL, L. E. ROSIER, AND H.-C. YEN, *Unary minimum cost path problems, alternating logspace, and Ruzzo, Simon and Tompa's DL^{NL}*, Department of Computer Sciences TR-87-13, University of Texas, Austin, TX, May 1987.

[17] N. IMMERMAN, *Nondeterministic space is closed under complementation*, SIAM J. Comput., 17 (1988), pp. 935–938.

[18] ———, *Upper and lower bounds on first order expressibility*, J. Comput. System Sci., 25 (1982), pp. 76–98.

[19] B. JENNER AND B. KIRSIG, *Characterizing the polynomial hierarchy by alternating auxiliary pushdown automata*, University of Hamburg, Federal Republic of Germany, 1987.

[20] H. JUNG, *On probabilistic time and space*, in Automata, Languages, and Programming, Springer-Verlag, Berlin, 1985, pp. 310–317.

[21] S. C. KLEENE, *Representation of events in nerve nets and finite automata*, in Automata Studies, C. E. Shannon and M. McCarthy, eds., Princeton University Press, Princeton, NJ, 1956, pp. 3–40.

[22] K.-J. LANGE, B. JENNER, AND B. KIRSIG, *The logarithmic alternation hierarchy collapses: $A\Sigma_2^{\mathcal{L}} = A\Pi_2^{\mathcal{L}}$*, in Automata, Languages, and Programming, Springer-Verlag, Berlin, 1987, pp. 531–541.

[23] H. R. LEWIS AND C. H. PAPADIMITRIOU, *Symmetric space-bounded computation*, Theoret. Comput. Sci., 19 (1982), pp. 161–187.

[24] S. R. MAHANEY, *Sparse complete sets for* NP: *Solution of a conjecture of Berman and Hartmanis*, J. Comput. System Sci., 25 (1982), pp. 130–143.

[25] J. H. REIF, *On synchronous parallel computations with independent probabilistic choice*, SIAM J. Comput., 13 (1984), pp. 46–56.

[26] ———, *Symmetric complementation*, in Proc. 14th Annual ACM Symposium on Theory of Computing, San Francisco, CA, May 1982, pp. 201–214.

[27] W. L. RUZZO, *Tree-size bounded alternation*, J. Comput. System Sci., 21 (1980), pp. 218–235.

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

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

[30] U. SCHÖNING AND K. WAGNER, *Collapsing oracle hierarchies, census functions, and logarithmically many queries*, in 5th Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag, Berlin, 1988, pp. 91–97.

[31] J. SIMON, *Space-bounded probabilistic Turing machine complexity classes are closed under complement*, in Proc. 13th Annual ACM Symposium on Theory of Computing, Milwaukee, WI, May 1981, pp. 158–167.

[32] I. H. SUDBOROUGH, *On the tape complexity of deterministic context-free languages*, J. Assoc. Comput. Mach., 25 (1978), pp. 405–414.

[33] R. SZELEPCSÉNYI, *The method of forcing for nondeterministic automata*, Bull. European Assoc. Theoret. Comput. Sci., 33 (1987), pp. 96–100.

[34] S. TODA, *$\Sigma_2 SPACE(n)$ is closed under complement*, J. Comput. System Sci., 35 (1987), pp. 145–152.

[35] L. G. VALIANT, *Short monotone formulae for the majority function*, J. Algorithms, 5 (1984), pp. 363–366.

[36] H. VENKATESWARAN, *Characterizations of parallel complexity classes*, Ph.D. thesis, University of Washington, Seattle, WA, August 1986; available as Department of Computer Science Tech. Report No. 86-08-06.

[37] ———, *Properties that characterize* LOGCFL, in Proc. 19th Annual ACM Symposium on Theory of Computing, New York, NY, May 1987, pp. 141–150.

[38] H. VENKATESWARAN AND M. TOMPA, *A new pebble game that characterizes parallel complexity classes*, SIAM J. Comput., this issue, pp. 533–549.

[39] S. WARSHALL, *A theorem on Boolean matrices*, J. Assoc. Comput. Mach., 9 (1962), pp. 11–12.

[40] D. J. A. WELSH, *Randomised algorithms*, Discrete Appl. Math., 5 (1983), pp. 133–145.