| － |  | Active |
| :---: | :---: | :---: |
|  | Cost share 排：G－36－368 | Rev 非： 0 |
| Center \＃：R6̈396－0A0 | Center shr \＃⿰：F6396－0A0 | OCA file 非： |
|  |  | Work type ：RES |
| Contractstarab711749 | Mod 非： | Document ：GRANT |
| Prime 非： |  | Contract entity：GTRC |
| Subprojects ？：N |  |  |
| Main project 非： |  |  |

Project unit：ICS Unit code：02．010．142

Project director（s）： HOLTY HRAN H $\operatorname{cs}^{\sec +}$

Sponsor／division codes 207
／ 000

Award period：$\quad$－ 7915 ．

| Sponsor amount | New this change | Total to date |
| :---: | :---: | :---: |
| Contract value | $55,000.00$ | $55,000.00$ |
| Funded | $55,000.00$ | $55,000.00$ |
| Cost sharing amount |  | $3,022.00$ |

Does subcontracting plan apply ？：$N$


PROJECT ADMINISTRATION DATA

| OCA contact mathemetionk | 894－4820 |
| :---: | :---: |
| Sponsor technical comtact | Sponsor issuing office |
| THOMAS J．HEAD | SHERRYE L．MCGREGOR |
| （202）357－7375 | （202）357－9621 |
| NATIONAL SCIENCE FOUNDATION | NATIONAL SCIENCE FOUNDATION |
| CISE／CCR | DGC／CISE |
| VASHINGTON，DC 20550 | WASHINGTON，DC 20550 |
|  | ONR resident rep．is ACO（Y／N） |
| Defense priority rating ：NONE | NSF supplemental sheet |
| Equipment title vests with：Sponsor | GIT X |

Administrative comments－
PROJECT INITIATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 05/23/91
Project No. 6-36-676
Center No. R6396-0AO $\qquad$
Project Director VENKATESWARAN H $\qquad$ School/Lab COMPUTING $\qquad$
Sponsor NATL SCIENCE FOUNDATION/GENERAL
Contract/Grant No. CCR-8711749. $\qquad$ Contract Entity GTRC
Prime Contract No. $\qquad$
Title STRUCTURE OF COMPUTATIONS IN PARALLEL COMPLEXITY CLASSES... $\qquad$
Effective Completion Date 910228 (Performance) 910531 (Reports)

| Closeout Actions Required: | $\mathrm{Y} / \mathrm{N}$ | Date <br> Submitted |
| :--- | :---: | :---: |
| Final Invoice or Copy of Final Invoice | N |  |
| Final Report of Inventions and/or Subcontracts | Y | $\mathbf{9 1 0 5 2 3}$ |
| Government Property Inventory \& Related Certificate | Y | - |
| Classified Material Certificate | N | - |
| Release and Assignment | N | - |
| Other | N | - |

CommentsBILLING VIA LINE-OF-CREDIT. 98A SUBMITTED WITH FINAL REPORT $\qquad$ SATISFIES REQUIREMENT FOR PATENT REPDRT.

Subproject Under Main Project No.
Continues Project No. $\qquad$

Distribution Required:
Project Director ..... Y
Administrative Network Representative ..... $\mathbf{Y}$
GTRI Accounting/Grants and Contracts ..... Y
Procurement/Supply Services ..... $\mathbf{Y}$
Research Property Managment ..... $\mathbf{Y}$
Research Security Services ..... N
Reports Coordinator (OCA) ..... Y
GTRC ..... $Y$
Project File ..... $\mathbf{Y}$
Other ..... NN

## Annual Report for the Period

15th September 1987 to 31st August 1988

Project Title : Structure of Computations in Parallel Complexity Classes.
NSF grant number : CCR-8711749
Principal Investigator : H: VENKATESWARAN

1. During this period, I undertook a study of semi-unbounded fan-in circuits as a model of computation. (A family of Boolean circuits is said to have semi-unbounded fan-in if there exists a constant $\mathrm{C}>\mathrm{O}$ such that in any circuit in the family all the AND gates have fan-in at most $C$ and any $O R$ gate can have unbounded fan-in.) New Characterizations of nondeterministic space and time classes were obtained on this model. Of particular interest is the definition of the important class NP as the class of languages accepted by uniform families of semi-unbounded fan-in circuits of exponential size and logarithmic depth. This is the first uniform circuit characterization of the class NP. It provides a framework to study some interesting questions about the class NP. The enclosed report "Circuit Definitions of Nondeterministic Complexity Classes" contains more details about the consequences of these characterizations.

I also studied other Boolean circuit characterizations of nondeterministic complexity classes. This was found useful in obtaining monotone arithmetic circuit characterizations of counting classes based on nondeterministic
time bounded computations. (Monotone arithmetic circuits are arithmetic circuits over the domain of non-negative integers and which use only the addition and multiplication operations.) An interesting consequence of this characterization is the definition of the well known counting class \#P as the set of functions computed by uniform families of monotone arithmetic circuits that have polynomial depth and polymonial degree. The degree measure here refers to the algebraic degree of the polynomial associated with the circuit.

A paper entitled, "Circuit Definitions of Nondeterministic Complexity Classes" containing these results has been accepted for presentation at the eight annual conference on 'Foundations of Software Technology and Theoretical Computer Science' to be held in Pune, India during 21-23 December 1988. A Georgia Institute of Technology technical report (GIT-ICS-88-09) with the same title as above and containing these results is enclosed herewith.
2. During this period, I also obtained some resulf about the complexity of some problems related to the computation of matchings in graphs. Given an undirected graph $G$, $a$ matching $M$ in $G$ is a collection of edges of $G$ such that no two edges in $M$ share a vertex. One of the important open questions in parallel computation is to find an NC algorithm to compute a maximum cardinalty matching in a graph. A random NC algorithm for this problem is known [Karp, upfal and Wigderson, "Constructing a Maximum Matching is in Random NC",

Combinatorica, 1986, pp.35-48]. I have shown that deciding whether a a given matching is a maximum cardinality matching is complete for the class NLOG (the class of problems solvable by nondeterministic Turning machines in logarithmic space). It follows that computing a maximum matching is unlikely to be in $\mathrm{NC}^{1}$.

A Georiga Institute of Technology technical report (GIT-ICS-88-10) entitled, "The Complexity of Some Problems Related to Matching", containing these results in enclosed herewith.
3. The following travels were undertaken during this period.
(a) I attended the 20th Annual ACM Symposium on Theory of Computing (STOC) Conference held in Chicago in May 1988.
(b) I visited Dr. Martin Tompa at the IBM Thomas J. Watson Research Center at Yorktown Heights in February 1988. While there I presented a seminar entitled, "Circuit Definitions of Nondeterministic Complexity Classes".

(H. VENKATESWARAN)

# Circuit Definitions of Nondeterministic Complexity Classes ${ }^{1}$ 

Technical Report GIT-ICS-88-09

H. Venkateswaran

March 1988

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332-0280


#### Abstract

We consider restrictions on Boolean circuits and use them to obtain new uniform circuit characterizations of nondeterministic space and time classes. We also obtain characterizations of counting classes based on nondeterministic time bounded computations on the arithmetic circuit model. It is shown how the notion of semi-unboundedness unifies the definitions of many natural complexity classes.


[^0]
## 1 Introduction

The Boolean circuit model has provided a very useful framework to study some of the important issues that arise in Turing machine based complexity theory. A major difficulty in translating questions from the Turing machine model to the circuit model is the non-uniform nature of circuits. An approach to handle this difficulty is to introduce a notion of uniformity for circuits. Uniformity for Boolean circuits was first suggested by Borodin[2], and later studied in-depth by Ruzzo[14] (see also the papers by Cook [4] and Pippenger [11]). Close connections have been established between complexity classes based on uniform circuits with those based on the machine model $[2,6,11,12,14]$. In one direction, complexity classes defined using the circuit model have been characterized using the machine model. NC is a well known example of such a complexity class defined using the uniform Boolean circuit model [11] that has been characterized using the alternating Turing machine model by Ruzzo [14]. In the other direction, traditional complexity classes based on the machine model have been characterized in the circuit model. The definition of the class $P$ using Boolean circuits $[8,12]$ is probably the first such result. Other results of this nature are the characterizations in the circuit model of the classes $A C^{1}$ [16] and LOGCFL [18]. The results by Ruzzo [14] also make it possible to obtain circuit characterizations of complexity classes defined using alternating Turing machines.

In the first part of this paper, we extend these results to characterize classes defined using nondeterministic Turing machines. We consider restrictions of Boolean circuits and use them to characterize nondeterministic space and time classes. We define skew circuits as ones in which all but one input of every AND gate are circuit inputs and use it to characterize nondeterministic space and time classes. Nondeterministic space is defined in terms of the size of such circuits and nondeterministic time is shown to correspond to the depth of these circuits. This should be contrasted with the well known correspondences between deterministic time and Boolean circuit size [12] and between nondeterministic space and Boolean circuit depth [2]. We had earlier considered a model of computation, called semi-unbounded fan-in circuits, in which the AND gates had bounded fan-in [18] and obtained a characterization of LOGCFL on this model. We show here how nondeterministic time can be defined in terms of semi-unbounded fan-in circuits. This correspondence is not surprising given the characterization of nondeterministic time by Ruzzo [13] using the tree-size resource on alternating Turing machines, and the close connections between tree-size and semi-unboundedness [18]. The semi-unbounded circuit model seerns useful to capture the definitions of many nondeterminisitic complexity classes.

In the second part of the paper, we use the monotone arithmetic circuit model to characterize
counting classes based on nondeterministic time bounded computations. Monotone arithmetic circuits are arithmetic circuits over the domain of non-negative integers and which use only the addition and multiplication operations. An interesting consequence of this characterization is the definition of the well known counting class $\sharp P$ as the set of functions computed by uniform families of monotone arithmetic circuits that have polynomial depth and polynomial degree. The degree measure here refers to the algebraic degree of the polynomial associated with the circuit.

It would be appropriate to mention some interesting consequences of the characterization results presented here.

- The circuit characterizations of $N P$ presented here are, to our knowledge, the first uniform circuit characterizations of this important complexity class. Of particular interest is the definition of $N P$ as the class of languages accepted by uniform families of semi-unbounded circuits of exponential size and log depth. This provides a framework to study some interesting questions about the class NP. Recently, Borodin et al. [3] proved that the language classes accepted by size and depth-bounded semi-unbounded fan-in circuits are closed under complement. The semi-unbounded fan-in circuit in their construction for complement recognition has depth $O(D+\log Z)$, where $Z$ is the size and $D$ is the depth of the given semi-unbounded fan-in circuit. Hence their result does not apply directly to $N P$. The relevant question here is whether the classes accepted by size $Z$ and depth $o(\log Z)$ semi-unbounded fan-in circuits are closed under complement. It is known that the classes accepted by size $Z$ and depth $o(\log n)$ semi-unbounded fan-in circuits are not closed under complement [18]. Another complexity question pertaining to $N P$ that can be phrased in this model is its relationship with the other classes definable using semi-unbounded fan-in circuits. A candidate class for comparison would be the class LOGCFL. It is known that LOGCFL can be characterized as the class of languages accepted by uniform families of polynomial size and log depth semi-unbounded fan-in circuits [18]. Therefore, the separation between NP and LOGCFL now becomes a question of the relative power of exponential size and polynomial size semi-unbounded fan-in circuits.
- The skew Boolean circuits provide a model to rephrase many of the famous separation questions among complexity classes. Thus the relationship between $P$ and NLOG translates into the question of the relative power of polynomial size Boolean circuits and polynomial size skew Boolean circuits. The $P$ versus PSPACE question becomes one of the relative power of polynomial size Boolean circuits and exponential size skew Boolean circuits. As another interesting example, the NP versus PSPACE question can be phrased as the question about polynomial depth for skew Boolean circuits versus polynomial depth for general Boolean circuits.
- The arithmetic characterization of $\sharp P$ presented here is the first alternative characterization of this class. It enables us to rephrase the famous open question about the relationship between $\sharp P$ and NP in terms of the relative power of arithmetic and Boolean circuits. It also touches on the power of arithmetic circuits over monotone arithmetic circuits.
- These characterizations also make it possible to identify appropriate circuit value problems that are complete for each of these complexity classes.

This paper is organized as follows. Section 1.1 contains some preliminary definitions. Boolean circuit characterizations of nondeterministic space and time classes are in section 2. Some characterizations of nondeterministic time that follow as simple consequences of known results are presented in section 3. Monotone arithmetic circuit characterization of counting classes based on nondeterministic time is presented in section 4.

### 1.1 Preliminaries

Boolean Circuits: A Boolean circuit $G_{n}$ with $n$ inputs is a finite acyclic directed graph with vertices having indegree zero or two and labelled as follows. Vertices of indegree zero are labelled from the set $\left\{0,1, x_{1}, x_{2}, \ldots, x_{n}, \bar{x}_{1}, \bar{x}_{2}, \ldots, \bar{x}_{n}\right\}$. All other vertices (also called gates) are labelled either AND or OR. It should be noted that not including negation gates in the definition of a Boolean circuit is done with no loss of generality. Vertices with outdegree zero are called outputs. The evaluation of $G_{n}$ on inputs of length $n$ is defined in the standard way. Typically, only circuits with one output vertex will be considered. This makes it convenient to consider circuits as language acceptors.

The size $C\left(G_{n}\right)$ of a circuit $G_{n}$ is the number of edges in $G_{n}$. The depth of a vertex $v$ in a circuit is the length of a longest path from any input to $v$. The depth of a circuit is the depth of its output vertex.

The language $L_{n}$ accepted by a Boolean circuit $G_{n}$ is the set of all length $n$ strings on which $G_{n}$ evaluates to one.

A family of circuits is a sequence $\left\{G_{\boldsymbol{n}} \mid n=0,1,2, \ldots\right\}$, where the $n$-th circuit $G_{\boldsymbol{n}}$ has $n$ inputs. The language $L$ accepted by a family $\left\{G_{n}\right\}$ of circuits is defined as follows: $L=\cup_{n \geq 0} L_{n}$, where $L_{n}$ is the language accepted by the $n$-th member $G_{n}$ of the family.

Skew Boolean Circuits: Let $G$ be a Boolean circuit. An AND gate $v$ in $G$ is said to be a skew gate if it has at most one input that is not an input of $G$. Without loss of generality, we will
assume that all but one of its inputs are inputs to the circuit $G$. We will refer to the input of $v$ that is not an input to $G$ as a non-skew input of $v$. The circuit $G$ is said to be a skew circuit if all AND gates in it are skew gates. A family $\left\{G_{n}\right\}$ of Boolean circuits is said to be a skew circuit family if all its members are skew circuits.

Note: One can define skewness with respect to OR gates also, but we will not pursue that in this paper.

Semi-Unbounded Fan-in Boolean Circuits: A family of Boolean circuits is said to have semi-unbounded fan-in if there exists a constant $c>0$ such that for any circuit in the family, the OR gates in the circuit can have unbounded fan-in and all the AND gates have fan-in at most $c$.

Semi-Unbounded Alternating Turing Machines: An alternating Turing machine is semiunbounded if there are no two consecutive universal configurations along any path in the computation tree of the machine. Without loss of generality, we will assume that every universal configuration of a semi-unbounded alternating Turing machine has exactly two existential configurations as immediate successors.

Uniformity: We will use the following notion of uniformity, called $U_{D}$-uniformity, defined by Ruzzo [14]. Define the direct connection language $L_{D C}$ of a family of Boolean circuits to be the set of strings of the form $\langle n, g, y\rangle$ such that either (i) $g$ and $y$ are gate names and $y$ is an input of the gate $g$, or (ii) $g$ is a gate name and $y$ is the type of the gate $g$, that is, $y$ is one of AND or OR or an input to $G_{n}$ or its negation. A family $\left\{G_{n}\right\}$ of Boolean circuits of size $C(n)$ is said to be uniform if the corresponding direct connection language can be recognized by a deterministic Turing machine in time $O(\log C(n))$.

For the space characterization results in section 2, it would have been sufficient to consider logspace uniformity defined by Borodin and Cook [4]. But a stronger uniformity condition is needed for the time characterization results to avoid the possibility of having a uniformity machine that is more powerful than the class being characterized. Such will be the case, for instance, in theorem 7 if we had used log-space uniformity since $\operatorname{NTME}(T(n)) \subseteq \operatorname{DSPACE}\left(T^{O(1)}(n)\right)$.

Accepting Subtrees [19]: The notion of an accepting subtree of a Boolean circuit given an input on which it evaluates to one is analogous to the notion of accepting subtrees of machines.

Let $B$ be a Boolean circuit, and let $T(B)$ be its tree equivalent. (The tree-equivalent of a graph is obtained by replicating vertices whose outdegree is greater than one until the resulting graph is a tree). Let $x$ be an input on which $B$ evaluates to one. An accepting subtree $H$ of the circuit $B$ on input $x$ is a subtree of $T(B)$ defined as follows:

- $H$ includes the output gate,
- for any AND gate $v$ included in $H$, all the immediate predecessors of $v$ in $T(B)$ are included as its immediate predecessors in $H$,
- for any OR gate $v$ included in $H$, exactly one immediate predecessor of $v$ in $T(B)$ is included as its only immediate predecessor in $H$, and
- any input vertex of $T(B)$ included in $H$ has value one as determined by the input $x$.

It is easy to verify the fact that the circuit $B$ evaluates to one given the input $x$ if and only if there is an accepting subtree of $T(B)$ on input $x$.

Tree-Size [19]: The tree-size measure for Boolean circuits can now be defined analogous to the tree-size measure for alternating Turing machines [14].

The circuit $B_{n}$ is said to have tree-size $Z(n)$ if, for every input $x$ accepted by $B_{n}$, there exists an accepting subtree with at most $Z(n)$ vertices.

Degree: We define the degree of a circuit to be the algebraic degree of the polynomial computed by the circuit. Thus, the constants have degree zero, the circuit inputs have degree one, the degree of an OR vertex is the maximum of the degrees of its inputs, and the degree of an AND vertex is the sum of the degrees of its inputs.

The following lemma [18] establishes a relationship between the measures degree and tree-size for Boolean circuits.

Lemma 1 Let $D(n), Z(n)$, and $d(n)$ be the degree, tree-size, and depth respectively of a Boolean circuit $B_{n}$. Then,

$$
Z(n) \leq D(n) d(n)+1 .
$$

Proof: The result to be proved also holds when the Boolean circuits considered have unbounded fan-in. Let the OR gates have fan-in at least $k$ and the AND gates have fan-in at least $l$.

Let $x$ be an input accepted by the circuit $B_{n}$. By hypothesis, there is an accepting subtree $H$ of $B_{n}$ of size at most $Z(n)$. Let $v$ be any vertex in $H$. Then the lemma follows from the claim below.

Claim: Let $Z(v)$ be the number of vertices in the subtree of $H$ rooted at $v, D(v)$ be the degree of $v$, and $d(v)$ be the depth of $v$. Then,

$$
Z(v) \leq D(v) d(v)+1 .
$$

Proof of the Claim: The proof of this claim is by induction on the depth of $v$.

The claim is clearly true when the depth of a vertex is zero. Assume that the claim holds for all vertices with depth less than $d(v)$. For the induction step, there are two cases:

Case 1: Let $v$ be an OR gate with inputs $v_{1}, \ldots, v_{k}$ with at least one non-constant input. Then,

$$
\begin{aligned}
Z(v) & \leq \max \left\{Z\left(v_{1}\right), Z\left(v_{2}\right), \ldots, Z\left(v_{k}\right)\right\}+1 \\
& \leq \max \left\{D\left(v_{1}\right) d\left(v_{1}\right), D\left(v_{2}\right) d\left(v_{2}\right), \ldots, D\left(v_{k}\right) d\left(v_{k}\right)\right\}+2 \\
& \leq \max \left\{\left(D\left(v_{1}\right), D\left(v_{2}\right), \ldots, D\left(v_{k}\right)\right\}(d(v)-1)+2\right. \\
& \leq D(v) d(v)-D(v)+2
\end{aligned}
$$

The claim follows from this since $D(v) \geq 1$.
Case 2: Let $v$ be an AND gate with inputs $v_{1}, \ldots, v_{l}$ with at least one non-constant input. Then,

$$
\begin{aligned}
Z(v) & =Z\left(v_{1}\right)+Z\left(v_{2}\right)+\ldots+Z\left(v_{l}\right)+1 \\
& \leq D\left(v_{1}\right) d\left(v_{1}\right)+D\left(v_{2}\right) d\left(v_{2}\right)+\ldots+D\left(v_{l}\right) d\left(v_{l}\right)+l+1 \\
& \leq\left(D\left(v_{1}\right)+D\left(v_{2}\right)+\ldots+D\left(v_{l}\right)\right)(d(v)-1)+l+1 \\
& \leq D(v) d(v)-D(v)+l+1
\end{aligned}
$$

The claim follows from this since $D(v) \geq 1$.

## 2 Characterizations of Space and Time Classes

This section contains the characterizations of nondeterministic space and time classes in terms of skew circuits and semi-unbounded fan-in circuits. Theorem 6 relates simultaneous space and time bounded nondeterministic classes to simultaneous size and depth bounded skew circuits. In this respect, it is similar to the result of Ruzzo [14] relating simultaneous space and time bounded alternating classes to simultaneous size and depth bounded circuits. However, the correspondence between the time and depth bounds in theorem 6 is only within a polynomial as opposed to the correspondence within a constant factor between circuit depth and alternating time shown by Ruzzo [14].

In the proof of lemma 3 below, we choose to use the alternating Turing machine model instead of directly constructing a semi-unbounded circuit corresponding to a skew circuit. This is done to
simplify the proof since we can use known simulation techniques. It also provides a new characterization of nondeterministic time on the alternating Turing machine model (see theorem 9). The correspondence between the machine and circuit models will be established through a sequence of lemmas.

Lemma 2 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$, $\operatorname{NSPACE}, \operatorname{TIME}(S(n), T(n)) \subseteq$ Uniform Skew Circuit SIZE, $\operatorname{DEPTH}\left(2^{\circ(S(n))}, T(n)\right)$.

Proof: Let $L$ be accepted by a nondeterministic Turing machine $M$ in $S(n)$ space and $T(n)$ time. The construction of a circuit family $\left\{G_{n}\right\}$ that accepts the same language as $M$ can be done using standard techniques $[14,18]$. For the sake of completeness, we will outline below the construction of $G_{n}$, the $n$-th member of this family.

The configurations of $M$ can be classified into two types: existential and read. We will assume that $M$ is deterministic while reading inputs.

For $0 \leq t \leq T(n)$, and a configuration $c$ of $M$ using space $S(n)$, there is a gate in the circuit in one of the following forms: $[t, c]$, or $[t, c, i]$, or $[t, c, i, b]$, where $0 \leq i \leq n$ is an integer and $b$ is either zero or one. The first component $t$ in a gate name is used to avoid cycles in the circuit. The type of a gate of the form $[t, c]([t, c, i],[t, c, i, b])$ is OR (OR, AND respectively).

Let $c_{I}$ be the initial configuration of $M$. The output gate is $\left[0, c_{I}\right]$.
The inputs of a gate are constructed as follows. Consider a gate $[t, c]$ corresponding to a nonread configuration $c$ of the machine. If $t+1>T(n)$, it has only one input, namely the constant zero. Otherwise, its inputs are constructed from the set $D$ of all configurations reachable by $M$ in one move from $c$. There will be one input corresponding to each $d \in D$. For any $d \in D$, if $d$ uses space $>S(n)$, then the corresponding input is the constant zero. For all other $d \in D$, there are two cases. If $d$ is an existential configuration the corresponding input is the gate $[t+1, d]$ and its inputs are constructed recursively. If $d$ is a read configuration in which $M$ reads the $i$ th symbol, the corresponding input is an OR gate $[t+1, d, i]$ with two inputs: $[t+1, d, i, 0]$ and $[t+1, d, i, 1]$. The gate $[t+1, d, i, 0]$ is an AND gate with two inputs: a constant one (zero) if the ith input has value zero (respectively, one), and the gate [ $t+2, e]$, where $e$ is the configuration to which $M$ moves from the read configuration $d$, if the $i$ th input read has value zero. The inputs of the gate $[t+2, e]$ are constructed recursively. The gate $[t+1, d, i, 1]$ is constructed in an analogous fashion.

It is clear from the construction of $G_{n}$ above that it is a skew circuit. The only AND gates constructed correspond to the read configurations of $M$. It is easy to show that $\left\{G_{n}\right\}$ accepts the same language as $M$. The size of the resulting circuit is $2^{O(S(n))}$. Its depth is $T(n)$.

It can be verified that the direct connection language of $\left\{G_{n}\right\}$ can be recognized by a deterministic Turing machine using $O(S(n))$ time, thus showing that the circuit family $\left\{G_{n}\right\}$ is uniform.

Lemma 3 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$,
Uniform Skew Circuit SIZE,DEPTH $\left(2^{\circ(S(n))}, T(n)\right) \subseteq$ Uniform Semi-Unbounded Circuit SIZE, $\operatorname{DEPTH}\left(2^{O(S(n))}, \log T(n)\right)$.

Proof: Let $\left\{G_{n}\right\}$ be a uniform family of skew circuits with the given size and depth bounds. Then $\left\{G_{n}\right\}$ has tree-size that is polynomial in $T(n)$. An alternating Turing machine $M$ that simulates $G_{n}$ on an input $x$ of length $n$ can be constructed as in the simulation by Ruzzo [14] of a space and tree-size bounded alternating Turing machine by a space and time bounded alternating Turing machine. The machine $M$ is semi-unbounded and uses space $O(S(n))$, alternations $O(\log T(n))$, and time $T^{O(1)}(n)$. Let the time used by $M$ be $T^{\prime}(n)=T^{a}(n)$ for some constant $a \geq 1$. Furthermore, $M$ is in a normal form such that only one input symbol is read along any path of the machine's computation tree. A uniform family $\left\{H_{n}\right\}$ of semi-unbounded fan-in circuits, with size $2^{O(S(n))}$ and depth $O(\log T(n))$, that accepts the same language as $M$ can be constructed by adapting known techniques [18]. The basic idea of the construction is to make as inputs to an OR (AND) gate all non-existential (non-universal) configurations of $M$ reachable through only existential (universal) configurations.

We will outline the construction of the $n$-th member $H_{n}$ of this family. The configurations of $M$ are assumed to be one of the following three types: existential, universal, and read.

Let $D(n)=\left\lceil\log _{2} T^{\prime}(n)\right\rceil$.
Gates in the circuit $H_{n}$ are all of the form [c], or [ $d^{\prime}$ ], or $[c, d]$, or $[s, c, d]$, or $[s, c, d, e]$, where $0 \leq s \leq D(n)$, and $c, d$ and $e$ are all configurations of $M$. The output gate of $H_{n}$ is [ $r_{0}$ ], where $r_{0}$ is the initial configuration of $M$. In general, the type of a gate of the form [ $c]$ is OR (AND) if the type of the configuration $c$ is existential (respectively, universal). Given a gate [c], its inputs are defined as follows.

Case 1: $[c]$ is an $O$ gate. Its inputs are gates $[c, d]$ for all configurations $d$ that are not existential. Each of the gates [ $c, d]$ is an AND gate and it has two inputs $[0, c, d]$ and [ $\left.d^{\prime}\right]$ defined as follows.

- The gate $[0, c, d]$ is the output of an $D(n)$ depth semi-unbounded fan-in circuit that checks that in $M$ the configuration $d$ is reachable from the configuration $c$ using only existential configurations of $M$. The following is a description of such a reachability circuit [18].

Given a gate $[s, c, d]$ with $0 \leq s \leq D(n)$, the goal is to describe a subcircuit of which this gate is the output, such that the subcircuit checks that $c$ is reachable from $d$ in $G_{n}$ using by a path of at most $2^{D(n)-s}$ OR gates (see also the construction by Borodin [2]).

If $d$ is an immediate predecessor of $c$ in $G_{n}$, then $[s, c, d]$ is the constant one. Otherwise, if $s+1>D(n)$, then $[s, c, d]$ is the constant zero. Otherwise, the gate $[s, c, d]$ is an OR gate. Its inputs are gates $[s+1, c, d, e]$ for all OR gates $e$ in $G_{n}$. Each of the gates $[s+1, c, d, e]$ is an AND gate, and it has the two inputs $[s+1, c, e]$ and $[s+1, e, d]$. These two subcircuits are constructed recursively.

- The gate $\left[d^{\prime}\right]$ is an OR gate with a single input [d] defined as follows. If $d$ is a read configuration with $a, i$ on its index tape, then [ $d$ ] is the $i$-th input to $H_{n}$. Otherwise, $d$ is a universal configuration. Then [ $d$ ] is an AND gate. Its inputs are constructed recursively.

Case 2: [c] is an AND gate. Let $d_{1}, d_{2}$ be the existential configurations of $M$ that immediately succeeds the configuration $c$. The inputs to $[c]$ are the OR gates $\left[d_{1}\right]$ and $\left[d_{2}\right]$. The inputs to these two OR gates are constructed recursively.

The circuit $H_{n}$ has size $2^{O(S(n))}$ and depth $O(\log T(n))$. Note that the OR gates in $H_{n}$ may have exponential fan-in whereas the fan-in of the AND gates is bounded by a constant. It is easy to show that $G_{n}$ and $H_{n}$ accept the same language. It is also straightforward to check that the direct connection language for the circuit family $\left\{H_{n}\right\}$ can be recognized by a deterministic Turing machine in time $O(S(n))$.

Lemma 4 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$,

$$
\begin{aligned}
& \text { Uniform Semi-Unbounded Circuit SIZE,DEPTH }\left(2^{O(S(n))}, \log T(n)\right) \subseteq \\
& \quad \text { NSPACE,TIME }\left(S(n) \log T(n), T^{O(1)}(n)\right) .
\end{aligned}
$$

Proof: This follows from the simulation of semi-unbounded fan-in circuits by nondeterministic auxiliary pushdown automata by Venkateswaran [18]. In this case, we are interested in the space and time used in the simulation.

Let $L$ be accepted by $\left\{G_{n}\right\}$, a uniform family of semi-unbounded fan-in circuits with size $2^{\circ}(\mathcal{S}(n))$ and depth $O(\log T(n))$. Given $x$ of length $n$, a nondeterministic machine $M$ checks whether the circuit evaluates to one on $x$ by doing a depth-first evaluation. The machine $M$ maintains a stack to do the circuit evaluation.
$M$ begins the simulation with the output gate $r_{0}$. Given a gate $v$ and its type, $M$ checks that $v$ evaluates to one on $x$ as follows. Let $C(v)$ denote the configuration of $M$ as it begins checking the gate $v$.

Case 1: $v$ is an OR gate. $M$ existentially guesses one of its true inputs $u$ and its type and verifies with the uniformity machine that the guesses are correct. It then recursively checks that the gate $u$ evaluates to one.

Case 2: $v$ is an AND gate. Then it has a constant number, say $k$, inputs. $M$ existentially guesses these inputs, say, $v_{1}, \cdots, v_{k}$, and their types and verifies with the uniformity machine that the guesses are correct. $M$ then pushes the gates $v_{2}, \cdots, v_{k}$ onto the stack. Along with a gate its type is also pushed onto the stack. $M$ then recursively checks that $v_{1}$ evaluates to one.

Case 3: $v$ is an input to the circuit. If its value is zero, $M$ rejects. Suppose $v$ has value one. $M$ makes its final pop move and accepts if the stack is empty. Otherwise, $M$ pops a gate $u$ and its type from the stack and recursively checks that $u$ evaluates to one.

For correctness, it can be shown, by induction, that the output $r_{0}$ of the circuit $G_{n}$ evaluates to one on input $x$ if and only if $M$ accepts starting from $C\left(r_{0}\right)$ and an empty stack [18].

Consider the space used by $M$ on input $x \in L$ of length $n$. In checking a gate $v, M$ must remember the gate $v$ and its type. If $v$ is an $O R$ gate, $M$ needs space to record information pertaining to a true input of $v$. This uses space $O(S(n))$. The space used for the gate $v$ can be reused at the next level of recursion. If $v$ is an AND gate, the information pertaining to all but one of its inputs is stored in the stack. This uses space $O(S(n))$. But, since the depth of the circuit is bounded by $O(\log T(n))$, the stack may have $O(\log T(n))$ such pieces of information using altogether $O(S(n) \log T(n))$ space. The uniformity machine uses $O(S(n))$ space. Therefore, the total space used in the simulation by $M$ is $O(S(n) \log T(n))$.

For the time bound of $M$, we first note that any accepting subtree of the circuit will have size $T^{O(1)}(n)$. The machine $M$, in verifying whether $G_{n}$ accepts its input, traverses such an accepting tree in a depth-first fashion visiting every vertex at most twice. For each node visited, $M$ uses time $O(S(n))$ to guess the information pertaining to the node and time $O(S(n))$ to invoke the uniformity machine to verify its guesses. Recall that the uniformity machine is a determinsitic machine using time $O(S(n))$. Since $S(n) \leq T(n)$, the total time used by $M$ is $T^{O(1)}(n)$.

In the proof of lemma 4 above, the space used for the stack can be completely avoided if the circuits being simulated are skew circuits. This observation leads immediately to the following lemma:

Lemma 5 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$, Uniform Skew Circuit SIZE,DEPTH $\left(2^{O(S(n))}, T^{O(1)}(n)\right) \subseteq \operatorname{NSPACE}, T I M E\left(S(n), T^{O(1)}(n)\right)$.

Lemmas 2 and 5 yield the following theorem:

Theorem 6 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$,
$\operatorname{NSPACE}, \operatorname{TIME}\left(S(n), T^{O(1)}(n)\right)=$ Uniform Skew Circuit SIZE, $\operatorname{DEPTH}\left(2^{O(S(n))}, T^{O(1)}(n)\right)$.

The following characterizations of nondeterministic time using skew circuits and semi-unbounded fan-in circuits are now immediate from lemmas 2,3 , and 4.

Theorem 7 For $T(n)=\Omega(n)$, the following complexity classes are equal:

1. $\operatorname{NTIME}\left(T^{O(1)}(n)\right)$
2. Uniform Skew Circuit $\operatorname{DEPTH}\left(T^{O(1)}(n)\right)$
3. Uniform Semi-Unbounded Circuit SIZE, $\operatorname{DEPTH}\left(2^{O(T(n))}, \log T(n)\right)$

As interesting consequences of theorems 6 and 7, we obtain the following Boolean circuit characterizations of the classes NLOG, PSPACE, and NP.

Corollary 8 . 1. NLOG $=$ Uniform Skew Circuit $\operatorname{SIZE}\left(n^{O(1)}\right)$.
2. $P S P A C E=$ Uniform Skew Circuit $\operatorname{SIZE}\left(2^{n^{O(1)}}\right)$.
3. $N P=$ Uniform Skew Circuit DEPTH $\left(n^{O(1)}\right)$.
4. $N P=$ Uniform Semi-Unbounded Circuit SIZE,DEPTH $\left(2^{n^{\circ(1)}}, \log n\right)$.

## 3 Other Characterizations of Nondeterministic Time

This section contains some characterizations of nondeterministic time that follow as simple consequences of known results. We will only consider bounded fan-in Boolean circuits in this section. Perhaps the most interesting of the characterizations here is the one using the depth and degree
measures for Boolean circuits. This suggests the characterization results in section 4 of counting classes based on nondeterministic time bounded computations.

Ruzzo [13] showed that nondeterministic time $T(n)$ is the class of languages accepted by alternating Turing machines simultaneously using space $O(T(n))$ and tree-size $O(T(n))$. This combined with the simulation by Ruzzo [13] of space and tree-size bounded alternating Turing machines by space and time-bounded alternating Turing machines (used in the proof of lemma 3) provides a new characterization of nondeterministic time bounded classes on the alternating Turing machine model. The close relationship between Boolean circuits and alternating Turing machines [14] also leads to another Boolean circuit characterization of nondeterminstic time in terms of size and treesize. Finally, the correspondence between degree and tree-size for Boolean circuits (see lemma 1) yields yet another Boolean circuit characterization of these classes in terms of degree and depth resources.

We will summarize these three characterizations in theorem 9 below. The proof of this theorem can be reconstructed from the results mentioned.

Theorem 9 For $T(n)=\Omega(n)$, the following complexity classes are equal:

1. $\operatorname{NTIME}\left(T^{O(1)}(n)\right)$
2. Semi-Unbounded ATIME,ALTERNATIONS $\left(T^{O(1)}(n), \log T(n)\right)$
3. Uniform Circuit SIZE,TREESIZE $\left(2^{T^{O(1)}}(n), T^{O(1)}(n)\right)$
4. Uniform Circuit DEPTH, DEGREE $\left(T^{O(1)}(n), T^{O(1)}(n)\right)$.

Thus, for instance, $N P$ has the following characterization in terms of degree and depth of Boolean circuits:

Corollary $10 N P=$ Uniform Circuit DEPTH,DEGREE $\left(n^{\circ}(1), n^{\circ}(1)\right.$.

The Boolean circuit characterization of NP in Corollary 10 should be contrasted with the following bounded fan-in Boolean circuit characterization of PSPACE [2,14]:
$\operatorname{PSPACE}=$ Uniform Circuit $\operatorname{DEPTH}\left(n^{O(1)}\right)=$ Uniform Circuit DEPTH,DEGREE $\left(n^{\circ(1)}, 2^{n^{\circ(1)}}\right)$.
Constant Depth Circuits: Before concluding this section, we mention another definition of $N P$ using constant depth unbounded fan-in circuits. We will show this by exhibiting a uniform family of constant depth Boolean circuits for the conjunctive normal form satisfiability problem.

Let SAT denote the language consisting of all strings that are (reasonable) encodings of satisfiable conjunctive normal form formulas. Let all length $r$ strings in SAT encode satisfiable formulas that have $n$ variables and $m$ clauses. The $r$-th member $G_{r}$ of a uniform circuit family $\left\{G_{r}\right\}$ that accepts SAT is described below. See figure 1.

- The output of $G_{r}$ is an OR gate labelled $[0, n, m]$. This gate evaluates to one on input $x$ if and only if the formula encoded by $x$ is satisfiable.
- The OR gate $[0, n, m]$ has as inputs AND gates labelled $[1, n, m, j]$ for $0 \leq j \leq 2^{n}-1$. An AND gate $[1, n, m, j]$ evaluates to one if and only if the input formula evaluates to one when the variables in the formula are assigned bit values from the integer $j$.
- Each AND gate labelled $[1, n, m, j$ ] has as inputs OR gates labelled $[2, n, m, j, k]$ for $1 \leq k \leq m$. An OR gate $[2, n, m, j, k]$ evaluates to one if and only if the $k$-th clause in the input formula evaluates to one when the variables in the formula are assigned bit values from the integer $j$.
- The inputs of an OR gate labelled [ $2, n, m, j, k$ ] are OR gates labelled $[3, n, m, j, k, l]$ for $1 \leq$ $l \leq n$. An OR gate $[3, n, m, j, k, l]$ is the output of a subcircuit that evaluates to one if and only if the $l$-th variable occurs in the $k$-th clause as a positive (negative) literal and the $l$-th bit of $j$ is one (respectively, zero). If the $l$-th variable does not occur in clause $k$ then a gate of the form $[3, n, m, j, k, l]$ evaluates to zero.

The family of Boolean circuits have size $O\left(m 2^{n}\right)$ and constant depth. The OR gates have fan-in at most $2^{n}$ and the AND gates have fan-in at most $m$. It can be verified that the direct connection language for $\left\{G_{r}\right\}$ can be recognized by a deterministic Turing machine in polynomial time, thus showing that this is a uniform family of circuits.

## 4 Monotone Arithmeic Circuits and Counting Classes

This section contains the characterizations of counting classes based on nondeterministic time bounded computations on the monotone arithmetic circuit model. A monotone arithmetic circuit is an arithmetic circuit using only the addition and multiplication operators and whose inputs are nonnegative integers. We will also characterize these classes in terms of the number of accepting subgraphs in the Boolean circuit model. As corollaries, we obtain characterizations of the class $\sharp P$ on these models.


Figure 1: Constant Depth Unbounded fan-in Circuits for CNF Satisfiability

### 4.1 Definitions

It will be convenient to consider Boolean circuits in which every AND gate has exactly two inputs.
Monotone Arithmetic Circuits: These are defined just as Boolean circuits, (thus, for instance, every product gate has exactly two inputs), except that the gates compute the sum and product of their inputs instead of computing the OR and AND functions. Although the results in this section, especially lemma 13, can be strengthened to handle $n$ bit integers as inputs to the circuit, it suffices to consider only single bit inputs.

We will denote a gate computing the sum (product) of its inputs as a PLUS (respectively, MULT) gate.

Uniformity: We will slightly modify the definition of uniformity in section 1.1 to do a parsimonious simulation in lemma 15.

Define the direct connection language of a family $\left\{G_{n}\right\}$ of Boolean circuits to be the set of strings of the form $<n, g, y, p>$ such that either (i) $g$ is a OR gate and $y$ is an input of $g$, or (ii) $g$ is a AND gate and $y$ is a left (right) input of $g$ if $p$ is $L$ (respectively, $R$ ), or (iii) $g$ is a gate name and $y$ is the type of the gate $g$. A family $\left\{G_{n}\right\}$ of Boolean circuits of size $C(n)$ is said to be uniform if the corresponding direct connection language can be recognized by a deterministic Turing machine in time $O(\log C(n))$.

The uniformity condition for monotone arithmetic circuits is defined exactly as for Boolean circuits with PLUS (MULT) gates replaced for OR (respectively, AND) gates.

Degree: The degree measure for monotone arithmetic circuits is defined anlaogous to Boolean circuits (see section 1.1). Thus, the constants have degree zero, the circuit inputs have degree one, the degree of a PLUS vertex is the maximum of the degrees of its inputs, and the degree of a MULT vertex is the sum of the degrees of its inputs.

Notations: Let $\mathcal{N}$ denote the set of natural numbers.
A function $f:\{0,1\}^{*} \rightarrow \mathcal{N}$ is in $\neq U n i f o r m$ Circuit SIZE,DEPTH,DEGREE $(Z(n), d(n), D(n))$ if and only if there exists a uniform family $\left\{G_{n}\right\}$ of Boolean circuits of size $O(Z(n))$, depth $O(d(n))$, and degree $O(D(n))$ such that for all strings $x$ of length $n, f(x)$ is the number of accepting subtrees of $G_{n}$ on input $x$.

The other counting classes are defined in a similar fashion.

### 4.2 The Characterization Results

The following fact can be used to set up a correspondence between Boolean and monotone arithmetic circuits. The proof of this fact is a direct consequence of the definition of an accepting subtree of a Boolean circuit (see section 1.1).

Fact 11 Let $B$ be a Boolean circuit that evaluates to one an input $x$. Given $x$ as an input, the number of accepting subtrees of $B$ rooted at an OR (AND) gate $v$ is the sum (respectively, product) of the number of accepting subgraphs of $B$ rooted at the inputs of $v$.

It may be noted that lemmas 12, 13, and 14 below are stronger statements than needed to prove the main results of this section, namely lemma 15 and theorem 17.

Lemma 12 Let $B$ be a Boolean circuit of size $Z$, depth $d$, and degree $D$. Then there exists an arithmetic circuit $A$ of size $Z$, depth $d$, and degree $D$ such that $B$ has $p$ accepting subtrees on an input $x$ on which it evaluates to one if and only if $A$ has value $p$ on input $x$.

Proof Sketch: Given a Boolean circuit $B$, let the arithmetic circuit $A$ be obtained by replacing all the OR (AND) gates of $B$ by PLUS (respectively, MULT) gates. Then the conclusion follows by using fact 11 .

Lemma 13 Let $A$ be a monotone arithmetic circuit of size $Z$, depth $d$, and degree $D$ with $n$ inputs from $\{0,1\}$. Then there exists a Boolean circuit $B$ of size $Z$, depth $d$, and degree $D$ such that $A$ has value $p$ if and only if $B$ has $p$ accepting subtrees given this input.

Proof Sketch: Given a monotone arithmetic circuit $A$, consider the Boolean circuit $B$ constructed from $A$ replacing all PLUS (MULT) gates by OR (respectively, AND) gates. The proof then follows by a simple inductive argument.

The circuits involved in the lemmas 12 and 13 above can be made uniform thereby showing the following correspondence between monotone arithmetic circuits and Boolean circuits.

Lemma 14 For $Z(n), D(n)=\Omega(n)$,
\#Uniform Circuit SIZE,DEPTH,DEGREE $\left(Z^{O(1)}(n), d(n), D(n)\right)=$ Uniform Monotone Arithmetic Circuit SIZE,DEPTH,DEGREE $\left(Z^{\circ(1)}(n), d(n), D(n)\right)$.

Lemma 15 below establishes the correspondence between the number of accepting paths in nondeterministic Turing machines and the number of accepting subtrees of Boolean circuits.

Lemma 15 For $T(n)=\Omega(n)$,

$$
\sharp \operatorname{NTIME}\left(T^{O(1)}(n)\right)=\sharp \text { Uniform Circuit DEPTH,DEGREE }\left(T^{O(1)}(n), T^{O(1)}(n)\right)
$$

Proof: Let $M$ be a nondeterministic Turing machine that runs in time $T(n)$. By theorem 7 , there exists a uniform family $\left\{B_{n}\right\}$ of $O(T(n))$ depth bounded skew circuits that accepts the same language as $M$. The degree of $B_{n}$ is $O(T(n))$. This is due to the fact that the degree of a depth $d$ skew circuit cannot exceed $d$. Any accepting subtree of $B_{n}$, given an input on which it evaluates to one, is a completely skewed binary tree. We claim that $M$ has $p$ accepting paths on an input $x$ of length $n$ if and only if $B_{n}$ has $p$ accepting subtrees.

To simplify the proof, we will assume that $M$ is deterministic while reading its inputs, and that the immediate successor of a read configuration is an existential configuration.

Let $x$ be an input of length $n$ accepted by $M$. Then $B_{n}$ evaluates to one on $x$. We will show that there is a bijective function that maps the accepting paths in the computation tree of $M$ on input $x$ with the accepting subtrees of $B_{n}$ on input $x$.

Let $p$ be an accepting path of $M$ on input $x$. The starting vertex of $p$ is labelled by the initial configuration $c_{I}$ of $M$. Consider the following subtree $A(p)$ of $B_{n}$ on input $x$. The root of $A(p)$ is the output gate $\left[0, c_{I}\right]$ of $B_{n}$. In general, the construction proceeds as follows. For the $t$-th vertex of $p$ labelled with an existential configuration $c$, pick the corresponding gate $[t, c]$ of $B_{n}$. The configuration $d$ that immediately succeeds $c$ along $p$ is either an existential configuration or a read configuration. If $d$ is an existential configuration pick as the input of the gate $[t, c]$ its input labelled $[t+1, d]$. Suppose $d$ is a read configuration in which $M$ reads the $i$-th input symbol and moves to an existential configuration $e(f)$ if the $i$-th input is zero (respectively, one). Consider the case when the $i$-th input symbol is zero. (The construction in the case when the $i$-th input symbol is one is analogous.) Then $d$ has the configuration $e$ as its immediate successor along $p$. Pick the gate $[t+1, d, i]$ as the input of the gate $[t, c]$, the AND gate $[t+1, d, i, 0]$ as the input of $[t+1, d, i]$, and the gate $[t+2, e]$ as the input of the gate $[t+1, d, i, 0]$. It is easy to see that $A(p)$ is an accepting subtree of $B_{n}$ on input $x$.

The mapping described above from accepting paths of $M$ on input $x$ to accepting subtrees of $B_{n}$ on input $x$ is well-defined. We will now argue that it is also a bijective function.

Suppose $p$ and $q$ are two distinct accepting paths of $M$ on input $x$. Let $A(p)$ and $A(q)$ be the corresponding subtrees defined by the above mapping. Now, $p$ and $q$ both have the same start
vertex namely, the one labelled with the initial configuration $c_{I}$. Let the initial common segment of $p$ and $q$ have $t$ vertices. Let the $t$-th vertex be labelled by the configuration $c$. Then $c$ must be an existential configuration. The corresponding gates in $A(p)$ and $A(q)$ are labelled by $[t, c]$. Since the immediate successor of $c$ in $p$ is different from that of $c$ in $q$, the input of the gate $[t, c]$ in $A(p)$ is different from that of $[t, c]$ in $A(q)$.

Suppose $A$ is an accepting subtree of $B_{n}$ on input $x$ of length $n$. We claim that there is an accepting path $p$ of $M$ on input $x$ such that $A$ is the image of $p$ as defined by the mapping above. The path $p$ is constructed as follows. The starting vertex of $p$ is labelled with the initial configuration $c_{I}$. Let $[t, c]$ be a vertex in $A$ where $c$ corresponds to an existential configuration of $M$ on input $x$. There are two cases.

Case 1: Suppose the gate $[t+1, d]$ is included in $A$ as the input of the gate $[t, c]$. Then $d$ is an existential configuration and it is an immediate successor of the configuration $c$ of $M$. Since $A$ is an accepting subtree on input $x$, the gate $[t+1, d]$ evaluates to one on input $x$. It follows that $d$ is an accepting configuration of $M$ on input $x$. Include a vertex labelled $d$ as the immediate successor of the vertex labelled $c$ along $p$.

Case 2: Suppose the gate $[t+1, d, i]$ is included in $A$ as the input of the gate $[t, c]$. Then $d$ is a read configuration that is an immediate successor of $c$. If $[t+1, d, i, 0]([t+1, d, i, 1])$ is the input of $[t+1, d, i]$ that is included in $A$, the $i$-th input symbol must be zero (respectively, one). Consider the case when the $i$-th input symbol is zero. (The case when the $i$-th input symbol is one is analogous.) Let the input of $[t+1, d, i, 0]$ included in $A$ be the gate $[t+2, e]$. Then include the vertex labelled $d$ as the immediate successor of $c$ and the vertex labelled $e$ as the immediate successor of $d$ along $p$. It is easy to verify that $p$ is an accepting path of $M$ on input $x$ and $A$ is an image of $p$ defined by the above mapping.

Conversely, let $\left\{B_{n}\right\}$ be a uniform family of Boolean circuits of depth $T^{O(1)}(n)$ and degree $T^{O(1)}(n)$. Let $M$ be a nondeterministic Turing machine that simulates $B_{n}$ on an input $x$ of length $n$ in a depth-first fashion as in the proof of lemma 4. The one difference here is the need to ensure that the simulation of an AND gate maintains the correspondence between the number of accepting paths of the machine and the number of accepting subtrees of the circuit. Let $C(v)$ denote the configuration of $M$ as it begins checking the gate $v$.

In simulating an AND gate $v, M$ does the following. It guesses the right input, say $v_{2}$, of $v$, verifies with the uniformity machine that the guess is correct, and pushes $v_{2}$ onto the stack. It then guesses the left input, say $v_{1}$, of $v$, verifies with the uniformity machine that the guess is correct, and verifies that $v_{1}$ evaluates to one. This will guarantee that there is a single accepting path segment from the configuration $C(v)$ to the configuration $C\left(v_{1}\right)$.

Then it follows, from the claim below, that $M$ has $p$ accepting paths on $x$ if and only if $B_{n}$ has $p$ accepting subtrees on input $x$.

Claim: Let $v$ be a vertex in $B_{n}$ that evaluates to one on input $x$. If $M$ begins its simulation at $v$, it has $p$ accepting paths rooted at $C(v)$ if and only if there are $p$ accepting subtrees of $B_{n}$ rooted at $v$.

Proof of the claim: This is by induction on the depth $d(v)$ of the vertex $v$.
The claim is clearly true for an input vertex $v$ with value one.

Suppose $v$ is an OR gate that evaluates to one on $x$. Let $v_{1}, \ldots, v_{m}$ be its inputs. Let $1 \leq q \leq m$ of these inputs, say $v_{i 1}, v_{i 2}, \ldots, v_{i q}$ evaluate to one on input $x$. The machine $M$, in checking whether $v$ evaluates to one, existentially chooses one of these $q$ inputs. Thus, the number of accepting paths rooted at $C(v)$ is given by the sum of the number of accepting paths rooted at $C\left(v_{i 1}\right), C\left(v_{i 2}\right), \ldots, C\left(v_{i q}\right)$. By induction hypothesis, this sum is equal to the sum of the accepting subtrees rooted at $v_{i 1}, v_{i 2}, \ldots, v_{i q}$. Since this is equal to the number of accepting subtrees of $B_{n}$ rooted at $v$, the claim follows.

Suppose $v$ is an AND gate that evaluates to one on $x$. Let $v_{1}$ and $v_{2}$ be its inputs. By construction, the number of accepting paths rooted at $C(v)$ is equal to the number of accepting paths rooted at $C\left(v_{1}\right)$. That is, if $M$ begins its simulation with the gate $v$, there is a single accepting path segment from $C(v)$ to $C\left(v_{1}\right)$. Thus, the number of accepting paths rooted at $C(v)$ is the same as the number of accepting paths rooted at $C\left(v_{1}\right)$. The machine $M$, in verifying $v_{1}$, traverses an accepting subtree of $B_{n}$ rooted at $v_{1}$. It then pops the vertex $v_{2}$. Hence there is a vertex labelled $C\left(v_{2}\right)$ along every accepting path of $M$ rooted at $C\left(v_{1}\right)$. Therefore, the number of accepting paths rooted at $C\left(v_{1}\right)$ is the product of the number of accepting path segments from $C\left(v_{1}\right)$ to $C\left(v_{2}\right)$ with the number of accepting paths rooted at $C\left(v_{2}\right)$. By induction hypothesis, the number of accepting path segments from $C\left(v_{1}\right)$ to $C\left(v_{2}\right)$ is the number of accepting subtrees rooted at $v_{1}$ of $B_{n}$, and the jumber of accepting paths rooted at $C\left(v_{2}\right)$ is the number of accepting subtrees of $B_{n}$ rooted at $v_{2}$. it follows that the number of accepting paths rooted at $C(v)$ is the number of accepting subtrees ooted at $v$ of $B_{n}$.

By lemma 1, the tree-size of $B_{n}$ is $T^{O(1)}(n)$. Since $B_{n}$ has size at most exponential in $T^{O(1)}(n)$, t follows, as in the simulation of lemma 4, that $M$ uses time $T^{O(1)}(n)$.

In lemma 15 above, we could have used semi-unbounded fan-in circuits instead of bounded an-in circuits to obtain the following result:

Theorem 16 For $T(n)=\Omega(n)$,
$\sharp \operatorname{NTIME}\left(T^{\circ(1)}(n)\right)=$
\#Uniform Semi-unbounded Fan-in Circuit SIZE,DEPTH,DEGREE $\left(2^{T^{O(1)}(n)}, T^{O(1)}(n), T^{O(1)}(n)\right)$.

Lemmas 14 and 15 together imply the following theorem:

Theorem 17 For $T(n)=\Omega(n)$,
$\sharp \operatorname{NTIME}\left(T^{O(1)}(n)\right)=$
Uniform Monotone Arithmetic Circuit DEPTH,DEGREE $\left(T^{O(1)}(n), T^{O(1)}(n)\right)$.

As a special case of the above theorem, we obtain the following new characterization of the important counting class $\sharp P$ :

Corollary 18
$\sharp P=$ Uniform Monotone Arithmetic Circuit DEPTH,DEGREE $\left(n^{O(1)}, n^{O(1)}\right)$.

### 4.3 Some Consequences

In this section, we will examine some consequences of the results in section 4.2.
Unique SAT: The Unique SAT problem is defined as follows [10]: Given an instance of SAT, does it have a unique solution? As another interesting corollary of theorem 17, we can identify an arithmetic circuit value problem that is equivalent to the Unique SAT problem.

Let $M$ be a fixed uniformity machine for a family $\left\{G_{n}\right\}$ of monotone arithmetic circuits of polynomial depth and polynomial degree. Given as input $n$ and an $n$ bit vector $x$, the MCVP1 problem is to determine whether the circuit $G_{n}$ evaluates to one on input $x$.

Corollary 19 There is a log space transformation from Unique SAT to MCVP1 and vice versa.
The Class UP and Arithmetic Circuits: The class UP was defined by Valiant [17] as :he class of languages accepted by polynomial time unambiguous Turing machines. These are zondeterministic Turing machines that are guaranteed to have at most one accepting computation or each input.

Define a monotone arithmetic circuit to be unambiguous if it evaluates to one or zero for all its aputs. Then he corollary below can be shown using the results in section 4.2

Corollary 20 The MCVP1 problem for unambiguous monotone arithmetic circuits is complete for the class UP.

New NP-Complete problems: Theorem 17 suggests a new arithmetic circuit value problem that is complete for $N P$. Let $M$ be a fixed uniformity machine for a family $\left\{G_{n}\right\}$ of monotone arithmetic circuits of polynomial depth and polynomial degree. Given as input $n$ and an $n$ bit vector $x$, the MCVP problem is to determine whether the circuit $G_{n}$ evaluates to a non-zero value on input $x$.

Proposition 21 The MCVP problem is NP-complete.
Theorem 17 also suggests versions of complete problems for $\# P$ that could be complete for $N P$. Define the NONZERO PERMANENT problem as follows:

Input: An $n$ by $n$ matrix $A$ with entries from $\{0,1\}$.
Property: The permanent of $A>0$.
The reduction used by Valiant [17], in showing that computing the value of the permanent of a matrix is complete for $\sharp P$, can also be used to show that this problem is NP-complete.

Proposition 22 The NONZERO PERMANENT problem is $N P$-complete.
Characterizing $\# P S P A C E$ Using Monotone Arithmetic Circuits: Using the known characterization of Boolean circuit depth by alternating time [14], the following analogue of lemma 15 can be proven using the techniques in the proof of that lemma:

Lemma 23 For $T(n)=\Omega(\log n)$,

$$
\sharp \operatorname{ATIME}\left(T^{O(1)}(n)\right)=\sharp \text { Uniform Circuit } \operatorname{DEPTH}\left(T^{O(1)}(n)\right) .
$$

This lemma combined with lemma 14 and the result by Ladner [9] that $\sharp P S P A C E=\sharp \operatorname{ATIME}\left(n^{O}{ }^{\circ}(1)\right)$ implies the following theorem:
Theorem 24

$$
\sharp P S P A C E=\text { Uniform Monotone Arithmetic Circuit DEPTH }\left(n^{\circ}(1)\right) .
$$

It should be noted here that Bertoni et al. [1] also characterized $\sharp P S P A C E$ as the class of functions computed by polynomial time Random Access Machines with the operations of addition, integer subtraction, multiplication, and integer division.

| OR fan-in | AND fan-in | SIZE | DEPTH | DEGREE | CLASS |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $n^{O(1)} /$ bounded | $n^{O(1)} /$ bounded | $n^{O(1)}$ |  | $n^{O(1)}$ | LOGCFL |
| $n^{O(1)}$ | bounded | $n^{O(1)}$ | $\log n$ |  | LOGCFL |
| $n^{O(1)}$ | $n^{O(1)}$ | $n^{O(1)}$ | $\log n$ |  | $A C^{1}$ |
| $n^{O(1)} /$ bounded | $n^{O(1)} /$ bounded | $n^{O(1)}$ |  |  | $P$ |
| $2^{n^{O(1)}}$ | bounded | $2^{n^{O(1)}}$ | $\log n$ |  | $N P$ |
| $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}}$ | $n^{O(1)}$ | $n^{O(1)}$ | $N P$ |
| $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}}$ |  | $2^{n^{O(1)}}$ | PSPACE |
| $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}}$ | $n^{O(1)}$ |  | PSPACE |

## Table 1: Circuit Definitions of Complexity Classes

## 5 Conclusion

This work provides a circuit framework in which some well-known open problems of complexity theory can be studied. We considered two constraints on the Boolean circuit model, namely skewness and semi-unboundedness, and used it to define nondeterministic space and time complexity classes. We also considered monotone arithmetic circuits to define counting classes based on nondeterministic time.

The known uniform Boolean circuit characterizations of classes between LOGCFL and PSPACE are summarized in table 1 (the definitions of the classes LOGCFL and $P$ in this table use log-space uniformity). It should not be too difficult to construct entries for classes above PSPACE.

As a consequence of these characterizations, we can define for each of these complexity classes a Boolean ircuit value problem that is a natural complete problem for the class. For example, the following circuit value problem is NP-complete. Let $M$ be a fixed uniformity machine for a family $\left\{G_{n}\right\}$ of Boolean circuits of polynomial depth and polynomial degree. Given as input $n$ and an $n$ bit vector $x$, the problem is to determine whether the circuit $G_{n}$ evaluates to one on input $x$.

We will conclude with a few remarks about the relevance of the semi-unboundedness notion for questions in complexity theory. From table 1, it can be seen that many of the well-known space and time complexity classes have definitions in terms of semi-unbounded fan-in circuits. Thus, for
instance, the following are definitions of some well known classes using the semi-unbounded fan-in circuit model:

$$
\begin{aligned}
\text { LOGCFL } & =\text { Uniform Semi-Unbounded Circuit SIZE,DEPTH }\left(n^{O(1)}, \log n\right) \\
P & =\text { Uniform Semi-Unbounded Circuit SIZE, } \operatorname{DEPTH}\left(n^{O(1)}, n^{O(1)}\right) \\
N P & =\text { Uniform Semi-Unbounded Circuit SIZE,DEPTH }\left(2^{n^{O(1)}}, \log n\right) \\
P S P A C E & =\text { Uniform Semi-Unbounded Circuit SIZE, } \operatorname{DEPTH}\left(2^{n^{O(1)}}, n^{O(1)}\right)
\end{aligned}
$$

One can define an analogue of the polynomial time hierarchy using semi-unbounded alternating Turing machines. Then, by theorem $9, N P$ is the class languages accepted by polynomial time semiunbounded alternating Turing machines using $O(\log n)$ alternations. This is interesting because it shows that with the constraint of semi-unboundedness $O(\log n)$ alternations is in $N P$, whereas without this constraint, even constant alternations is not known to be in NP.

## Acknowledgements

I am grateful to Martin Tompa for useful discussions. My thanks are due to Larry Ruzzo whose work on alternating Turing machines and Boolean circuits was a source of inspiration for the results reported here. I am also thankful to Gary Peterson for his comments.

## References

[1] Bertoni, A., G. Mauri, and N. Sabadini, Simulations Among Classes of Random Access Machines and Equivalence Among Numbers Succintly Represented, Annals of Discrete Mathematics 25, (1985), 65-90.
[2] Borodin, A., On Relating Time and Space to Size and Depth, SIAM Journal of Computing 6, (1977), 733-743.
[3] Borodin, A., S.A. Cook, P.W. Dymond, W.L. Ruzzo, and M. Tompa, Two Applications of Complementation via Inductive Counting, University of Washington Technical Report 87-1001, October 1987.
[4] Cook, S.A., Deterministic CFL's are accepted simultaneously in polynomial time and log squared space, Proc. 11th Annual ACM Symposium on Theory of Computing, (1979), 338-345.
[5] Cook, S.A., A Taxonomy of Problems with Fast Parallel Algorithms, Information and Control 64, 1-3 (Jan/Feb/Mar 1985), 2-22.
[6] Dymond, P.W. and S.A. Cook, Hardware Complexity and Parallel Computation, Proc. 21st Annual Symposium on Foundations of Computer Science, Toronto, 1980.
[7] Goldschlager, L.M., The Monotone and Planar Circuit Value Problems are log space Complete for $P$, SIGACT News 9, 2, 1977, 25-29.
[8] Ladner, R. E., The Circuit Value Problem is log space Complete for P, SIGACT News 7, 1, 1975, 18-20.
[9] Ladner, R. E., Polynomial Space Counting Problems, manuscript, May 1986.
[10] Papadimitriou, C. H., M. Yannakakis, The Complexity of Facets (and some Facets of Complexity), Journal of Computer and System Sciences 28, (1984), 244-259.
[11] Pippenger, N., On Simultaneous Resource Bounds, Proc. 20th Annual Symposium on Foundations of Computer Science, Puerto Rico, 1979.
[12] Pippenger, N., M.J. Fischer, Relations among Complexity Measures, Journal of the Association for Computing Machinery 26, (1979), 361-381.
[13] Ruzz , W.L., Tree-Size Bounded Alternation, Journal of Computer and System Sciences 20, (1980), 218-235.
[14] Ruzzo, W.L., On Uniform Circuit Complexity, Journal of Computer and System Sciences 22, (1981), 365-383.
[15] Skyum, S., L.G. Valiant, A Complexity Theory Based on Boolean Algebra, Journal of the Association for Computing Machinery 32, (1985), 484-502.
[16] Stockmeyer, L. and U. Vishkin, Simulation of Parallel Random Access Machines by Circuits, SIAM Journal of Computing 13, (1984), 409-422.
[17] Valiant, L.G., The Complexity of Computing the Permanent, Theoretical Computer Science 8, (1979), 189-201.
[18] Venkateswaran, H., Properties that Characterize LOGCFL, Proc. 19th Annual ACM Symposium on Theory of Computing, (1987), 141-150.
§] Venkateswaran, H. and M. Tompa, A New Pebble Game that Characterizes Parallel Complexity Classes, Proc. 27th Annual Symposium on Foundations of Computer Science, Toronto, 1986.

# The Complexity of Some Problems Related to Matching ${ }^{1}$ 

Technical Report GIT-ICS-88-10<br>H. Venkateswaran

March 1988

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332-0280


#### Abstract

Some problems related to matching are shown to be complete for the class nondeterministic logarithmic space. These include finding an augmenting path between two specified vertices with respect to a matching and deciding whether a given matching is a maximum matching. These results also help identify some problems that are complete for the class deterministic logarithmic space.


Keywords: augmenting path, complete problems, computational complexity, matching, nondeterministic log space

[^1]
## 1 Introduction

Given an undirected graph $G$, a matching $M$ in $G$ is a collection of edges of $G$ such that no two edges in $M$ share a vertex. The problem of computing a maximum cardinality matching in a graph has been well studied (see, for example, the survey by Galil [6]). In this paper, we examine the complexity of deciding whether a given matching is a maximum cardinality matching. Let NLOG denote the class of decision problems accepted by a nondeterministic Turing machine in space $O(\log n)$. We show that deciding the existence of an augmenting path with respect to a given matching in a graph is complete for $N L O G$ under $N C^{1}$ reduciblity (see section 2 below for a definition of $N C^{1}$ reducibility). This is then used to show that deciding whether a given matching is a maximum cardinality matching is complete for NLOG. These completeness results also hold when the graphs considered are bipartite. Let DLOG denote the class of decision problems accepted by a deterministic Turing machine in space $O(\log n)$. The results about complete problems for NLOG help to identify problems that are complete for $D L O G$.

Natural complete problems for the class NLOG are interesting because it is unlikely that they are solvable in logarithmic depth by a uniform family of Boolean circuits or in deterministic logarithmic space. It is also not known whether they can be implemented in simultaneous polynomial time and polylog space[16]. The first complete problem for NLOG was discovered by Savitch [15]. This problem is to decide whether there exists a directed path between two specified vertices in a directed graph. Since then many other natural problems have been shown to be complete for this class (see, for instance, Jones et al. [9] and Cook [3]). The purpose of this paper is to show that certain problems related to matching in graphs are also complete for NLOG. One of the important open questions in parallel computation has been to find a deterministic NC algorithm to compute a maximum cardinality matching in a graph. It has been shown by Karp et al. [11] that this problem is in Random NC. It follows from the results here that this problem is at least hard for NLOG, and hence unlikely to be in $N C^{1}$. The question of finding a deterministic $N C$ algorithm for computing a maximum cardinality matching in a graph remains open.

Section 2 contains some preliminary definitions and observations. The completeness results are presented in section 3.

## 2 Preliminaries

Augmenting Path: Let $G=(V, E)$ be an undirected graph. Given a matching $M$ in $G$, a path from a vertex $s$ to a vertex $t$ in $G$ is said to be an augmenting path [13] if

1. $s$ and $t$ have no edges from $M$ incident on them, and
2. every alternate edge along the path belong to $M$.

The following lemma, due to Berge [1] and Norman and Rabin [12], provides a characterization of a maximum cardinality matching in terms of augmenting paths. See the text by Papadimitriou and Steiglitz [13] for a proof.

Lemma 1 A matching $M$ in a graph $G$ is a maximum matching if and only if there is no augmenting path with respect to $M$ in $G$.
$N C^{1}$ Reduciblity: We will use alternating Turing machines [2] to define the notion of $N C^{1}$ reducibility. This is adapted from a definition by Cook [3].

Consider a function $f:\{0,1\}^{*} \rightarrow\{0,1\}^{*}$. Let $L_{f}$, the set associated with the function $f$ be defined as: $\{\langle x, i\rangle \mid$ the $i$ th bit of $f(x)$ is 1$\}$. The function $f$ is said to be $N C^{1}$ computable if for all $x \in\{0,1\}^{*}$ of length $n,|f(x)|=n^{O(1)}$, and the set $L_{f}$ associated with $f$ can be recognized by an alternating Turing machine using time $O(\log n)$. A language $L_{1}$ is said to be $N C^{1}$ reducible to a language $L_{2}$ if there exists an $N C^{1}$ computable function $f$ such that $x \in L_{1}$ if and only if $f(x) \in L_{2}$.

Completeness: A language $L$ is said to be complete for $N L O G$ ( $D L O G$ ) if (a) it is in $N L O G$ (respectively, $D L O G$ ), and (b) for all languages $L^{\prime}$ in $N L O G$ (respectively, $D L O G$ ), $L^{\prime}$ is $N C^{1}$ reducible to $L$.

## 3 The Completeness Results

In this section, we show that deciding whether a given matching in a graph is a maximum one is complete for $N L O G$ with respect to $N C^{1}$ reducibility. We begin by showing that a certain two color path problem for undirected graphs is complete for NLOG. Then we use this to show that the problem of finding an augmented path between two vertices in a graph given a matching in the graph is complete for NLOG. By reducing this problem to that of deciding whether a given matching in a graph is not a maximum matching, we show that the latter problem is complete for NLOG. Since NLOG is closed under complement[7], it follows that the problem of deciding whether a given matching in a graph is a maximum matching is also complete for NLOG. We also show some related problems complete for the class $D L O G$ under $N C^{1}$ reducibility.

In section 3.1, we will consider several path problems that will be useful in establishing the results about matching. The matching results themselves are presented in section 3.2.

### 3.1 Path Problems

## TWO COLORED PATH:

Input: An undirected graph $G=(V, E)$ whose edges are colored black and red, and two vertices $s, t \in V$.

Property: There is a path in $G$ between $s$ and $t$ that begins and ends with a black edge, and the edges along the path strictly alternate in color.

We will refer to a path with the above property as a two colored path.

Lemma 2 The TWO COLORED PATH problem is complete for $N L O G$ under $N C^{1}$ reducibility.

Proof Sketch: A nondeterministic Turing machine can decide this problem using $O(\log n)$ space by guessing a path between $s$ and $t$ one vertex at a time and verifying that it is indeed a two color path.

To show that this problem is hard for NLOG, we reduce the directed graph reachability problem to this problem. The directed graph reachability problem is: given a directed graph and two vertices $s$ and $t$ in it, is there a directed path in the graph from $s$ to $t$ ? It is well known that this problem is complete for $N L O G[15]$. The reduction from the directed graph reachability problem to the TWO COLORED PATH problem is an adaptation of the reduction of the directed Hamilonian circuit to the undirected Hamiltonian circuit problem (see, for instance, Karp[10]).

Given a directed graph $G=(V, E)$ and two vertices $s, t \in V$, construct an edge colored undirected graph $G^{\prime}=\left(V^{\prime}, E^{\prime}\right)$ as follows:

For every vertex $v \in V$, add to $V^{\prime}$ two vertices $(v, 0)$ and $(v, 1)$, and add to $E^{\prime}$ a black edge between $(v, 0)$ and $(v, 1)$. For every directed edge $u \rightarrow v$ in $E$, add to $E^{\prime}$ a red edge between ( $u, 0$ ) and ( $\ddot{v}, 1$ ).

It is easy to show that there exists a directed path in $G$ from $s$ to $t$ if and only if there exists a two colored path in $G^{\prime}$ from $(s, 0)$ to $(t, 1)$.

Let the vertices of $G$ be numbered from 0 to $n-1$. Let $x$ be the bit string consisting of $A$, the adjacency matrix of $G$, and the two vertices $s$ and $t$ of $G$. Let $f(x)$ be the bit string consisting of $B$, the adjacency matrix of $G^{\prime}$ and the two vertices $(s, 0)$ and $(t, 1)$ of $G^{\prime}$. Note that $B$ will be a matrix with $2 n$ rows and columns. It can be verified that an alternating Turing machine, on input $x$, can decide in time $O(\log n)$ whether the $i$ th bit of $f(x)$ is a one.

We now make several interesting observations about this reduction. First, note that the graph constructed in this reduction is a bipartite graph. This leads to the following corollary to lemma 2:

Corollary 3 The TWO COLORED PATH problem for bipartite graphs is complete for NLOG under $N C^{1}$ reducibility.

More interestingly, the graphs constructed have the property that every vertex has exactly one incident black edge. This makes it possible to identify a complete problem for NLOG that provides a link to problems concerning maximum matchings in graphs.

Corollary 4 The TWO COLORED PATH problem for graphs (both general and bipartite) with at most one incident black edge per vertex is complete for $N L O G$ under $N C^{1}$ reduciblity.

The reduction in lemma 2 above also suggests a problem that is complete for $D L O G$ under $N C^{1}$ reducibility. Let a bipartite graph $G=(V 1, V 2, E)$, in which every vertex in $V 2$ has degree at most two, be referred to as a restricted bipartite graph.

Let $G=\left(V_{1}, V_{2}, E\right)$ be a restricted bipartite graph whose edges are colored black and red such that every vertex in $G$ has at most one black edge incident on it. Consider the TWO COLORED PATH problem when the input consists of the graph $G$ and the vertices $s \in V_{1}$ and $t \in V_{2}$.

Theorem 5 The TWO COLORED PATH problem for restricted bipartite graphs is complete for DLOG under $N C^{1}$ reduciblity.

Proof Sketch: A deterministic machine $D$ initializes a counter to the number of vertices in $G$. It then starts from the vertex $s \in V_{1}$ and traverses the unique black edge incident on it. This brings it to a vertex in $V_{2}$ that has at most two incident edges. The machine $D$ accepts if this vertex is $t$. If the vertex is not $t$ and it has only one incident edge, then $D$ rejects. If the vertex is not $t$ and it has two incident edges, $D$ decrements the counter and traverses the edge that is not used. This can go on until either $D$ halts accepting its input, or it halts rejecting its input, or the count has become zero. In the last case, $D$ rejects its input. It is clear that the machine uses logarithmic space.

For showing that this problem is hard for $D L O G$, we use the graph reachability problem for directed graphs of outdegree one, a problem known to be complete for DLOG (see Jones [8] or Cook and McKenzie [4]). It is straightforward to check that the reduction in the proof of lemma 2 produces a restricted bipartite graph if the input graph is a directed graph with outdegree one.

Before we conclude this section, it is interesting to note that the undirected graph reachability problem is $N C^{1}$ equivalent to the following version of the TWO COLOR PATH problem.

SAME COLORED PATH:

Input: An undirected graph $G=(V, E)$ whose edges are colored black and red, a color $c \in$ \{black, red\}, and two vertices $s, t \in V$.

Property: There is a path in $G$ between $s$ and $t$ in which all edges have color $c$.

Theorem 6 The SAME COLORED PATH problem is $N C^{1}$ equivalent to the undirected graph reachability problem.

Proof Sketch: In one direction, given an undirected graph $G$ and two vertices $s$ and $t$, one can color all edges of $G$ black so that there is a path in this graph from $s$ to $t$ consisting of all black edges if and only if there is a path from $s$ to $t$ in the input graph $G$.

In the other direction, given a graph $G$ whose edges are colored red and black, two vertices $s$ and $t$, and a color $c \in\{$ black, red\}, construct the following graph $H$. For each vertex $v$ in $G$ have two copies $v_{b}$ and $v_{r}$. For each edge ( $u, v$ ) of $G$, if the edge is colored red (black) add an edge between $u_{r}$ and $v_{r}$ (respectively, $u_{b}$ and $v_{b}$ ). It is easy to show that there is a path from $s_{c}$ to $t_{c}$ in the graph $H$ if and only if there is a path from $s$ to $t$ with all edges colored $c$.

### 3.2 Matching Problems

The TWO COLORED PATH problem for graphs with no more than one incident black edge per vertex suggests a close relationship with the problem of finding an augmenting path given a matching in a graph. We show below that this problem is also complete for NLOG.

ST-AUGMENTING PATH:

Input: An undirected graph $G=(V, E)$, a matching $M \subseteq E$ of $G$, and two vertices $s, t \in V$.

Property: There is an augmenting path with respect to $M$ in $G$ between $s$ to $t$.

Theorem 7 The ST-AUGMENTING PATH problem (for both general and bipartite graphs) is complete for $N L O G$ under $N C^{1}$ reducibility.

Proof: A nondeterministic machine first checks that $s$ and $t$ have no incident matching edges, and then it guesses a path between $s$ to $t$ that strictly alternates between an unmatched edge and a matched edge. All these can be done in logarithmic space.

To show NLOG-hardness, we reduce the TWO COLORED PATH problem for graphs with at most one incident black edge to this problem. The theorem will then follow by corollary 4.

Given such a graph $G=(V, E)$ and two vertices $s$ and $t$, construct a graph $G^{\prime}=\left(V^{\prime}, E^{\prime}\right)$, a matching $M$, and two vertices $s^{\prime}$ and $t^{\prime}$ as follows. The vertex set $V^{\prime}$ consists of all vertices in $V$ and two additional vertices $s^{\prime}$ and $t^{\prime}$. The edge set $E^{\prime}$ consists of all edges in $E$ and two additional edges $\left(s, s^{\prime}\right)$ and $\left(t, t^{\prime}\right)$. The matching $M$ consists of all edges of $E$ that are colored black.

The reduction above can be slightly modified if the input graph is bipartite. Given the bipartite graph $G=\left(V_{1}, V_{2}, E\right)$, construct a bipartite graph $G^{\prime}=\left(V_{1}^{\prime}, V_{2}^{\prime}, E^{\prime}\right)$, a matching $M$, and two vertices $s^{\prime}$ and $t^{\prime}$ as follows. Without loss of generality, we will assume that $s \in V_{1}$ and $t \in V_{2}$. The vertex set $V_{1}^{\prime}\left(V_{2}^{\prime}\right)$ consists of all vertices in $V_{1}$ (respectively, $V_{2}$ ), and an additional vertex $t^{\prime}$ (respectively, $s^{\prime}$ ). The edge set $E^{\prime}$ consists of all edges in $E$ and two additional edges $\left(s, s^{\prime}\right)$ and ( $t, t^{\prime}$ ). The matching $M$ consists of all edges of $E$ that are colored black.

We will outline an encoding of the input and the output of these reductions that shows that they are $N C^{1}$ reductions. Let the vertex set of $G$ be $\{0,1, \ldots, n-1\}$. Define the matrix $A$ as follows: for $0 \leq i, j \leq n-1$, if there is no edge between vertex $i$ and vertex $j, A(i, j)=00$, if there is a black edge between vertex $i$ and vertex $j, A(i, j)=01$, and finally, if there is a red edge between vertex $i$ and vertex $j, A(i, j)=10$. Let the input $x$ be the bit string consisting of the matrix $A$ and the two vertices $s$ and $t$. For representing the output, define the matrix $B$ corresponding to the graph $G^{\prime}$ in a similar fashion. The matrix $B$ will have $n+2$ rows and columns. Let $s^{\prime}$ be the vertex $n$ and $t^{\prime}$ be the vertex $n+1$. Then, $B(i, j)=A(i, j)$ for $0 \leq i, j \leq n-1$, and $B(s, n)=B(n, s)=B(t, n+1)=B(n+1, t)=10$. Let $f(x)$ be the bit string consisting of the matrix $B$ and the two vertices $s^{\prime}$ and $t^{\prime}$. It can be seen from this description that the function $f$ is $N C^{1}$ computable.

The correctness of the reductions follow from the claim below which is easily proved.
Claim: There exists an augmenting path with respect to $M$ in $G^{\prime}$ between $s^{\prime}$ to $t^{\prime}$ if and only if there is a two color path in $G$ between $s$ and $t$.

The next step in our reductions, namely the problem of deciding whether a given matching in a graph is not a maximum matching, is facilitated by lemma 1 . We show that this problem is also complete for NLOG.

NOT MAXIMUM MATCHING:

Input: An undirected graph $G=(V, E)$, and a matching $M \subseteq E$ of $G$.
Property: $M$ is not a maximum cardinality matching in $G$.

Theorem 8 The NOT MAXIMUM MATCHING problem (for both general and bipartite graphs) is complete for $N L O G$ under $N C^{1}$ reduciblity.

Proof: By lemma 1, it suffices to decide whether there is an augmenting path with respect to the matching $M$ in the graph. A nondeterministic machine guesses two vertices $s$ and $t$ with no incident matching edges, and guesses an augmenting path between $s$ and $t$.

The theorem now follows from theorem 7 and the following reduction from the ST-AUGMENTING PATH problem to the NOT MAXIMUM MATCHING problem.

Given a graph $G=(V, E), s, t \in V$, and a matching $M$, construct a graph $G^{\prime}=\left(V^{\prime}, E^{\prime}\right)$ and a matching $M^{\prime}$ as follows. The vertex set $V^{\prime}$ consists of all vertices $V$, the edge set $E^{\prime}$ consists of all edges in $E$, and the matching $M^{\prime}$ consists of all edges in $M$. In addition, these three sets also contain the following new elements: for each vertex $w \in V$ such that $w \neq s$, and $w \neq t$, if $w$ has no edge from $M$ incident on it, add a new vertex $s_{w}$ to $V^{\prime \prime}$ and a new edge ( $s_{w}, w$ ) to $E^{\prime}$ and $M^{\prime}$.

If the input graph is a bipartite graph, then the following modification to the reduction above works. Given a bipartite graph $G=\left(V_{1}, V_{2}, E\right), s \in V_{1}, t \in V_{2}$, and a matching $M$, construct a bipartite graph $G^{\prime}=\left(V_{1}{ }^{\prime}, V_{2}{ }^{\prime}, E^{\prime}\right)$ and a matching $M^{\prime}$ as follows. The vertex set $V_{1}{ }^{\prime}\left(V_{2}{ }^{\prime}\right)$ consists of all vertices in $V_{1}$ (respectively, $V_{2}$ ), the edge set $E^{\prime \prime}$ consists of all edges in $E$, and the matching $M^{\prime}$ consists of all edges in $M$. In addition, these three sets also contain the following new elements: For each vertex $w \in V_{1}$ such that $w \neq s$, if $w$ has no edge from $M$ incident on it, add a new vertex $s_{w}$ to $V_{2}^{\prime}$ and a new edge $\left(s_{w}, w\right)$ to $E^{\prime}$ and $M^{\prime}$. Similarly, for each vertex $w \in V_{2}$ such that $w \neq t$, if $w$ has no edge from $M$ incident on it, add a new vertex $s_{w}$ to $V_{1}^{\prime}$ and a new edge $\left(s_{w}, w\right)$ to $E^{\prime}$ and $M^{\prime}$.

It can be verified that these reductions are $N C^{1}$ reductions. The correctness of these reductions follow from the two claims below.

Claim 1: $M^{\prime}$ is a matching in $G^{\prime}$.
Claim 2: $M^{\prime}$ is not a maximum cardinality matching in $G^{\prime}$ if and only if there is an augmenting path with respect to $M$ in $G$ from $s$ to $t$.

Proof of Claim 1: Any edge in $M^{\prime}$ is either an edge in $M$ or a new edge added in the construction of $G^{\prime}$. Consider an edge $e=(w 1, w 2)$ in $M^{\prime}$.

Case 1: $e \in M$. Then no new edge added in the construction is incident on $w 1$ or $w 2$. Since no other edge from $M$ shares a vertex with $e$, it follows that no other edge from $M^{\prime}$ shares a vertex with the edge $e$.

Case 2: $e \in E^{\prime}-E$. Let $w_{1}$ be the new vertex added in the construction. The other case is analogous. Then, the vertex $w_{2}$ has no edges from $M$ incident on it, and furthermore, from $M^{\prime}$ only the edge $e$ is incident on $w_{2}$. Since $w 1$ has only one incident edge, namely $e \in M^{\prime}$, it follows that no other edge from $M^{\prime}$ shares a vertex with the edge $e$.

Proof of Claim 2: By construction, every vertex in $G^{\prime}$, except possibly $s$ and $t$, have an edge from $M^{\prime}$ incident on it. The two vertices $s$ and $t$ in $V^{\prime}$ will have no edge from $M^{\prime}$ incident on them if and only if hey have no edges from $M$ incident on them in $G$. Therefore, any augmenting path with respect to $M^{\prime}$ in $G^{\prime}$, if it exists, must be between $s$ and $t$. Claim 2 now follows from claim 3 below as follows. Suppose there is an augmenting path with respect to $M$ in $G$ between $s$ and $t$ then, by claim 3 , there will be an augmenting path with respect to $M^{\prime}$ in $G^{\prime}$ which, by lemma 1 , implies that $M^{\prime}$ is not a maximum cardinality matching. In the other direction, if $M^{\prime}$ is not a maximum cardinality matching in $G^{\prime}$ then, by lemma 1 , there is an augmenting path with respect to $M^{\prime}$ in $G^{\prime}$. Since any augmenting path with respect to $M^{\prime}$ in $G^{\prime}$ is between $s$ and $t$, it follows, from claim 3 below, that there is an augmenting path with respect to $M$ in $G$ between $s$ and $t$.

Claim 3: There exists an augmenting path with respect to $M^{\prime}$ in $G^{\prime}$ between $s$ and $t$ if and only if there is an augmenting path with respect to $M$ in $G$ between $s$ and $t$.

Proof of Claim 3: Suppose there is an augmenting path $p$ with respect to $M$ in $G$ between $s$ and $t$. Then $s$ and $t$ have no edges from $M$ incident on them in $G$. Therefore, $s$ and $t$ will have no edges from $M^{\prime}$ incident on them in $G^{\prime}$. Since all edges in $M$ are also present in $M^{\prime}$, it follows that any matched edge with respect to $M$ in $G$ along the path $p$ is also a matched edge with respect to $M^{\prime}$ in $G^{\prime}$. For any edge $e \in E$ along $p$ such that $e$ is not a matched edge with respect to $M$ in $G$, $e$ is also not a matched edge with respect to $M^{\prime}$ in $G^{\prime}$ since $M^{\prime}$ consists of only edges from $M$ and the new edges in $E^{\prime}$ added by the construction of $G^{\prime}$. Finally, since every edge along $p$ is also an edge in $E^{\prime}, p$ is an augmenting path with respect to $M^{\prime}$ in $G^{\prime}$ between $s$ and $t$.

Conversely, let $p$ be an augmenting path with respect to $M^{\prime}$ in $G^{\prime}$ between $s$ and $t$. Clearly, $s, t \in V$. Since $s$ and $t$ have no edges from $M^{\prime}$ incident on them in $G^{\prime}$, it follows that $s$ and $t$ have no edges from $M$ incident on them in $G$. Now, for any edge $e \in E^{\prime}$ such that $e \notin M^{\prime}$, it is true that $e \notin M$ and $e \in E$. Therefore, if an edge $e$ along the path $p$ is an unmatched edge with respect to $M^{\prime}$ in $G^{\prime}$, it is also an unmatched edge with respect to $M$ in $G$. Consider a matched edge $e=(w 1, w 2) \in E^{\prime} \cap M^{\prime}$ along $p$. Each of $w 1$ and $w 2$, since they occur along the path $p$, have
at least two incident edges. Therefore, the edge $e$ cannot be a new edge, because at least one vertex of a new edge has only one incident edge. That is, the edge $e$ is also present in $E$, Since $e \in M^{\prime}$, we conclude that it is in $M$. It follows that $p$ is also an augmenting path with respect to $M$ in $G$.

We now consider the complement of this problem, namely that of deciding whether a given matching in a graph is a maximum cardinality matching.

## MAXIMUM MATCHING:

Input: An undirected graph $G=(V, E)$, and a matching $M \subseteq E$ of $G$.
Property: $M$ is a maximum cardinality matching in $G$.

The completeness of this result is a consequence of the following result of Immerman[7] that shows that the class NLOG is closed under complernent.

Lemma 9 ([7]) For any space constructible $s(n) \geq \log n$, the class $\operatorname{NSPACE}(s(n))$ is closed under complement.

Therefore, from theorem 8 and lemma 9, we obtain the following corollary:

Corollary 10 The MAXIMUM MATCHING problem (for general and bipartite graphs) is complete for $N L O G$ under $N C^{1}$ reducibility.

Finally, we want to note that for restricted bipartite graphs, the three problems considered in this section, namely the ST-AUGMENTING PATH problem, the NOT MAXIMUM MATCHING problem and the MAXIMUM MATCHING problem, are complete for DLOG. The reductions are the same as those in the corresponding problems for bipartite graphs.

Let $G=\left(V_{1}, V_{2}, E\right)$ be a restricted bipartite graph, and $M$ be a matching in $G$. Consider the ST-AUGMENTING PATH problem when the input consists of the graph $G$ and the vertices $s \in V_{2}$ and $t \in V_{1}$.

Theorem 11 The ST-AUGMENTING PATH problem, the NOT MAXIMUM MATCHING problem and the MAXIMUM MATCHING problem for restricted bipartite graphs are complete for $D L O G$ under $N C^{1}$ reducibility.

Proof Sketch: The proofs are analogous to those for bipartite graphs. The main difference is in showing that these problems are in DLOG. We will only show this for the ST-AUGMENTING PATH problem. The proof that the other two problems are in DLOG follow similar arguments.

That the ST-AUGMENTING PATH problem is in DLOG can be shown using an argument very similar to the upper bound argument in theorem 5. A deterministic machine $D$ begins by checking that the vertices $s$ and $t$ do not have any incident matched edges. It then initializes a counter to the number of vertices in $G$. Let $s$ have two neighbors, say, $u_{1}$ and $v_{1}$ in $V_{1}$. The machine $D$ starts from $s$ and traverses the edge that takes it to the vertex $u_{1}$. It accepts if $u_{1}$ is $t$. Otherwise, the machine decrements its counter, and if $u_{1}$ has a matching edge incident on it, $D$ traverses it and gets to a vertex in $V_{2}$. If $u_{1}$ has no matching edge incident on it, $D$ rejects its input. If this vertex has only one incident edge, $D$ rejects. If this vertex has two incident edges, $D$ decrements the counter and traverses the edge that is not used to get to a vertex that is in $V_{1}$. The machine $D$ continues this unitl it either accepts, or rejects, or the counter has value zero In the last case, it rejects its input. This traversal takes only logarithmic space.

## 4 Concluding Remarks

We will conclude by identifying some related problems all of which can be shown to be hard for the class NLOG by the results in section 3. Consider the problem of computing the size of a maximum matching in a graph. Since testing whether a given matching $M$ in a graph $G$ is a maximum matching can be done by testing whether the cardinality of $M$ is equal to the size of a maximum matching in $G$, it follows by corollary 10 , that computing the size of a maximum matching is at least hard for NLOG. This is true even in the case of bipartite graphs. Any problem to which the problem of computing the size of a maximum matching can be reduced is also NLOG-hard. Such problems include testing whether a graph has a perfect matching, finding a maximum matching in a graph, and computing the rank of a certain matrix of indeterminates [6].

## References

[1] Berge, C., Two Theorems in Graph Theory, Proc. National Academy of Sciences 43, (1957), 842-844.
[2] Chandra, A.K., D.C. Kozen, and L.J. Stockmeyer, Alternation, JACM 28, (1981), 114-133.
[3] Cook, S.A., A Taxonomy of Problems with Fast Parallel Algorithms, Information and Control 64, 1-3 (Jan/Feb/Mar 1985), 2-22.
[4] Cook, S.A., and P. McKenzie, Problems Complete for Deterministic Logarithmic Space, Journal of Algorithms 8, (1987), 385-394.
[5] Edmonds, J., Paths, Trees and Flowers, Canadian Journal of Mathematics 17, (1965), 449-467.
[6] Galil, Z., Sequential and Parallel Algorithms for Finding Maximum Matchings in Graphs, in Annual Review of Computer Science 1, (1986), 197-224.
[7] Immerman, N., Nondeterministic Space is Closed under Complement, YALEU/DCS/TR 552, Deaprtment of Computer Science, Yale University, (1987).
[8] Jones, N.D., Space Bounded Reducibility among Combinatorial Problems, Journal of Computer and System Sciences 11, (1975), 68-85.
[9] Jones, N.D., Y.E. Lien, and W.T. Laaser, New Problems Complete for Nondeterministic Log Space, Mathematical Systems Thepry 10, (1976), 1-17.
[10] Karp, R.M., Reducibility among Combinatorial Problems, in Complexity of Computer Computations, Plenum press, New York (1972), 85-103.
[11] Karp, R.M., E. Upfal, and A. Wigderson, Constructing a Maximum Matching is in Random NC, Combinatorica 6, (1986), 35-48.
[12] Norman, R.Z., and M.O. Rabin, An Algorithm for a Minimum Cover of a Graph, Proc. American Mathemtical Society 10, (1959), 315-319.
[13] Papadimitriou, C.H., and K. Steiglitz (1982), "Combinatorial Optimization: Algorithms and Complexity", Prentice-Hall, New Jersey.
[14] Ruzzo, W.L., On Uniform Circuit Complexity, JCSS 22, (1981), 365-383.
[15] Savitch, W.J., Relationships Between Nondeterministic and Deterministic tape Complexities, Journal of Computer and System Sciences 4, (1970), 177-192.
[16] Tompa, M., Two Familiar Transitive Closure Algorithms Which Admit No Polynomial Time, Sublinear Space Implementations, SIAM J. of Comput. 11, (1982), 130-137.

NATIONAL SCIENCE FOUNDATION
Washington, D.C. 20550

## FINAL PROJECT REPORT

 NSF FORM 98APLEASE READ INSTRUCTIONS ON REVERSE BEFORE COMPLETING PART I-PROJECT IDENTIFICATION INFORMATION

- Institution and Address

Georgia Institute of Technology Office of Contract Administration Atlanta, GA 30332
2. NSF Frogram Computer \&

Computation Theory

From 9/87 To 2/91
3. NSF Award Number

CCR-8711749
5. Cumulative Award Amount \$55,000
. Project Title
Structure of Computations in Parallel Complexity Classes

## PART II-SUMMARY OF COMPLETED PROJECT (FOR PUBLIC USE)

The project funded by this grant consisted of investigations of some computational complexity issues using the Boolean circuit framework. A study of semi-unbounded fan-in circuits as a model of computation was undertaken. (A family of Boolean circuits is said to have semi-unbounded fan-in if all the AND gates have constant fan-in and any OR gate can have unbounded fan-in.) It was shown that many standard complexity classes can be defined in terms of uniform semi-unbounded. fan-in Boolean circuits, which included a new characterization of the important class NP. Starting with these circuit definitions, a simple circuit based proof has been obtained for a very important complexity theory result by Toda. Studying the applications of combinatorial models such as the dual two-person pebble game model of synchronous parallel computation, which was one of the objectives of this project, a generalization of the game was obtained to model nondeterministic complexity classes. A closely related one-person pebble game which extended a standard one-person pebble game on Boolean circuits by taking the gate types also into account was also considered. It was shown that the extended games were more powerful than the original games for two natural circuits, corresponding to the Cocke-Kasami-Younger algorithm and the Warshall's algorithm. Among other things, this showed that lower bounds based on the one-person game are for an oblivious evaluation of circuits, but there may exist other small space evaluations that are not oblivious. Finally, to get a better persepective on theoretical models of parallel computation, an experimentation study of parallel algorithms designed for these models was undertaken using available parallel architectures such as Sequent and the Butterfly.

| PART III-TECHNICAL INFORMATION (FOR PROGRAM MANAGEMENT USES) |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ITEM (Check appropriate blocks) | NONE | ATTACHED | PREVIOUSLY FURNISHED | TO BE FURNISHED <br> SEPARATELY TO PROGRAM |  |
|  |  |  |  | Check ( $\sim$ ) | Approx. Date |
| a. Abstracts of Theses |  |  |  |  |  |
| b. Publication Citations |  | $\overline{\mathrm{X}}$ |  |  |  |
| c. Data on Scientific Collaborators |  |  |  |  |  |
| d. Information on Inventions |  |  |  |  |  |
| e. Technical Description of Project and Results |  | X |  |  |  |
| f. Other (specify) |  |  |  |  |  |
| . Principal Investigator/Project Director Name (Typed) H. Venkateswaran | 3. Prinicidal Inv | stigator/Proj enk | Director Sign |  | 4. Date $22 \text { May '91 }$ |

## PART IV - SUMMARY DATA ON PROJECT PERSONNEL

NSF Division _Computer_and_Computation Theory
The data requested below will be used to develop a statistical profile on the personnel supported through NSF grants. The information on this part is soliclted under the authorlty of the National Science Foundation Act of 1950, as amended. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. NSF requires that a single copy of this part be submitted with each Final Project Report (NSF Form 98A); however, submisslon of the requested Information is not mandatory and is not a precondition of future awards. If you do not wish to submit this information, please check this box

Please enter the numbers of individuals supported under this NSF grant.
Do not enter information for individuals working less than $\mathbf{4 0}$ hours in any calendar year.

|  | Pl's/PD's |  | Postdoctorals |  | Graduate Students |  | Undergraduates |  | Precoliege Teachers |  | Others |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Permanent Visa | Male | Fem. | Male | Fem. | Male | Fem. | Male | Fem. | Male | Fem. | Male | Fern. |
| American Indian or Aiaskan Native |  |  |  |  |  |  |  |  |  |  |  |  |
| Asian or Pacitic Islander | X |  |  |  |  |  |  |  |  |  |  |  |
| Black, Not of Hispanic Origin $\qquad$ |  |  |  |  |  |  |  |  |  |  |  |  |
| Hispanic ........... |  |  |  |  |  |  |  |  |  |  |  |  |
| White, Not of Hispanic Origin |  |  |  |  |  |  |  |  |  |  |  |  |
| otal U.S. Citizens ..... |  |  |  |  |  |  |  |  |  |  |  |  |
| on U.S. Citizens . . . . | 1 |  |  |  |  |  |  |  |  |  |  |  |
| otal U.S. \& Non- U.S. | 1 |  |  |  |  |  |  |  |  |  |  |  |
| umber of individuals who have a handicap that limits a major life activity. |  |  |  |  |  |  |  |  |  |  |  |  |

*Use the category that best describes person's ethnic/racial status. (If more than one category applies, use the one category that most closeiy reflects the person's recognition in the community.)

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America, and who maintains cultural identification through tribal affiliation or community recognition.

ASIAN OR PACIFIC ISLANDER: A person having origins in any of the original peoples of the Far East, Southeast Asia, the Indian subcontinent, or the Pacific islands. This area includes, for exemple, Ctine, India, Japan, Korea, the Phllippine Islands and Samoa.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in amy of the black raciel groupa of Africa.
HISPANIC: A person of Mexican, Puerto Rican. Cuban, Central or Sorth American or other Spanish culture or origln, regardless of race.
WHITE, NOT OF HISPANIC ORIGIN: A person heving origins in any of the original peoples of Europe, North Africa or the Middle East.
THIS PART WILL BE PHYSICALLY SEPARATED FROM THE FINAL PROJECT REPORT AND USED AS A COMPUTER SOURCE DOCUMENT. DO NOT DUPLICATE IT ON THE REVERSE OF ANY OTHER PART OF THE FINAL REPORT.

## Final Project Report

I was funded by a two year NSF grant CCR-8711749 beginning in September 1987. A one year no-cost extension of this grant was given when I was on a leave of absence for a year during 1988-89 from the Georgia Institute of Technology. The research activities supported by this grant and the results obtained are listed below. The papers and technical reports that have resulted thus far from the research conducted under this award are listed separately, and are referred to in this page.

1. A simple circuit based proof has been obtained for Toda's theorem which states that $\oplus P$ is hard for the Polynomial Hierarchy under randomized reductions (paper 1). This approach starts with uniform circuit definitions of the Polynomial Hierarchy and $\oplus P$, and applies the Valiant-Vazirani lemma on these circuits.
2. New circuit characterizations were obtained for well known complexity classes, including a characterization of NP in terms of uniform semi-unbounded fan-in Boolean circuits. This work also showed that many standard complexity classes can be defined in terms of uniform semi-unbounded fan-in Boolean circuits. A new uniform monotone arithmetic circuit characterization of the counting class $\sharp P$ was also obtained. The definitions of $N P$ and $\sharp P$ were the first uniform circuit definitions of these classes. A paper containing these results was presented at the eighth annual conference on Foundations of Software Technology and Theoretical Computer Science held at Pune, India in December 1988 (paper 2). This has also been submitted to the SIAM Journal of Computing.
3. Some preliminary results were obtained for the computation versus verification problem for the maximum matching problem. It was shown that the problem of verifying whether a given matching in a graph is a maximum one is hard for the class NLOG, the class of problems solvable by nondeterministic Turing machines using logarithmic work space (paper 3 ).
4. An extension of the one-person pebble game on Boolean circuits, in which the gate types are taken into account, was considered. The extended games were shown to
be more powerful than the original games for two natural circuits, corresponding to the Cocke-Kasami-Younger algorithm and the Warshall's algorithm (paper 4). Among other things, this showed that lower bounds based on the one-person game are for an oblivious evaluation of circuits, but there may exist other small space evaluations that are not oblivious.
5. The dual two-person pebble game defined by Venkateswaran and Tompa (SIAM J. of Computing, Vol. 18, June 1989, pp.533-549) was extended to model, among other things, the polynomial-time hierarchy (paper 5).
6. In addition to the theoretical aspects of parallel computation, I have also been pursuing some practical aspects such as the issues in implementing parallel algorithms designed for theoretical models on real parallel architectures. In this latter area, work is in progress in the following two directions:

- Experimentation with parallel algorithms designed for models such as PRAMs on available parallel architectures such as Sequent and the Butterfly (paper 6).
- Exploring new models and paradigms for message passing parallel architectures (paper 7).


## Publications resulting from work supported by the grant

1. Ravi Kannan, H. Venkateswaran, V. Vinay and A. Yao, A circuit-based proof of Toda's theorem, Georgia Institute of Technology Technical Report GIT-CC-90-33 (To appear in Information and Computation).
2. H. Venkateswaran, Circuit definitions of nondeterministic complexity classes, Proc. 8th Annual conference on Foundations of Software Technology and Theoretical Computer Science, Pune, India, December 1988. (Submitted to SIAM Journal of Computing.)
3. H. Venkateswaran, The complexity of some problems related to matching, Georgia Institute of Technology Technical Report GIT-ICS-88-10.
4. H. Venkateswaran, Two dynamic programming algorithms for which interpreted pebbling helps, Georgia Institute of Technology Technical Report GIT-ICS-89-23 (To appear in Information and Computation).
5. V. Vinay, H. Venkateswaran and C.E. Veni Madhavan, Circuits, Pebbling and Expressibility, Georgia Institute of Technology Technical Report GIT-CC-90-32, Proc. 5th annual conference on Structure in Complexity Theory Barcelona, Spain, June 1990.
6. Anand Sivasubramaniam, Gautam Shah, Joonwon Lee, Umakishore Ramachandran and H. Venkateswaran, Experimental evaluation of algorithmic performance on two shared memory multiprocessors, Proc. International Symposium on Shared Memory Multiprocessing, Tokyo, Japan, April 2-4, 1991.
7. Anand Sivasubramaniam, Umakishore Ramachandran and H. Venkateswaran, Exploring programming paradigms for message-passing architectures, Georgia Institute of Technology Technical Report GIT-CC-91-11.

## A Circuit-Based Proof of Coda's Theorem

Ravi Kennan *<br>School of Computer Science<br>Carnegie-Mellon University<br>Pittsburgh, Pennsylvania 15213<br>V.Vinay *<br>Department of Computer Science and Automation<br>Indian Institute of Science<br>Bangalore 560012, India<br>H. Venkateswaran ${ }^{\dagger}$<br>College of Computing<br>Georgia Institute of Technology<br>Atlanta, Georgia 30332-0280<br>Andrew C. Mao ${ }^{\ddagger}$<br>Department of Computer Science<br>Princeton University<br>Princeton, New Jersey 08544


#### Abstract

We present a simple proof of Coda's result (Coda, 1989) which states that $\oplus P$ is hard for the Polynomial Hierarchy under randomized reductions. Our approach is circuit-based in the sense that we start with uniform circuit definitions of the Polynomial Hierarchy and apply the Valiant-Vazirani (1986) lemma on these circuits.


[^2]
## 1 Introduction

In this paper we give a simple proof of Toda's result (Toda, 1989) which states that $\oplus P$ is hard for the Polynomial Hierarchy $(P H)$ under randomized reductions. The class $\in P$ is the class of all languages $L$ such that $L$ is accepted by a polynomial time nondeterministic Turing machine that accepts an input iff it has an odd number of accepting computations (Papadimitriou and Zachos, 1983). The original proof of this theorem is by Toda (1989) who uses it to prove the result that $P H \subseteq P^{P P}$. Our approach is circuit-based in the sense that we start with uniform circuit definitions of PH (Vinay, Venkateswaran and Veni Madhavan, 1990) and apply the Valiant-Vazirani (1986) lemma on these circuits.

Our results demonstrate the usefulness of uniform circuit characterizations of standard complexity classes such as NP. These circuit definitions served as a motivation to investigate the applicability of recent circuit complexity results to exponential size circuits. We claim that such characterizations have made it possible to isolate the combinatorial essence of Toda's proof more clearly. We also give new uniform circuit characterizations of the class $\oplus P$ and BP. $\oplus P$. Here BP. $\oplus P$ is the class of all languages $L$ for which there exists a language $A \in \oplus P$ and a constant $\delta>0$ such that, for any randomly chosen polynomial length string $y$, the probability that $[(x, y) \in A$ iff $x \in L]$ is at least $1 / 2+\delta$. These circuit characterizations provide a combinatorial and algebraic framework to study the properties of these classes.

Allender (1989) motivated by Toda's proof technique, studied the simulation of constant depth polynomial size circuits by threshold circuits. Later, this result was extended in (Allender and Hertrampf, 1990) to uniform circuits. Our results, which were discovered independently, are proved for exponential size circuits. Considering exponential size circuits makes it possible to present Toda's proof in a circuit framework. The uniformity condition used here is also different and is based on recognition of the connection language of the circuits rather than generating circuit descriptions. The language recognition version of the uniformity condition is used since the circuits considered have exponential size. The focus of the results in (Allender and Hertrampf, 1990) is on depth reduction for unbounded fan-in circuits whereas our focus was on providing a simple proof of Toda's theorm using well known circuit techniques. Although it is not too difficult to derive our results and the simulation in (Allender and Hertrampf, 1990) of $A C^{0}$ by a uniform family of
constant depth threshold circuits as corollaries of a general theorem, we believe that an explicit proof of Toda's theorem will be quite useful. Both our results use a lemma from Valiant and Vazirani (1986) to achieve uniformity.

## 2 Preliminaries

The Boolean circuits that we consider are, in general, uniform unbounded or semi-unbounded fan-in circuits. It is assumed that the negations appear only at the inputs. By semi-unbounded fan-in circuits we mean circuits where there is an asymmetry in the fan-in of the gates depending on the gate type. The uniformity condition for the circuit families considered is based on the recognition of the direct connection language defined as follows (Ruzzo, 1981): The direct connection language $L_{D C}$ of a family $\left\{G_{n}\right\}$ of Boolean circuits is the set of strings of the form $\langle n, g, y\rangle$ such that either (i) $g$ and $y$ are gate names and $y$ is an input of the gate $g$, or (ii) $g$ is a gate name and $y$ is the type of the gate $g$. A family $\left\{G_{n}\right\}$ of Boolean circuits of size $C(n)$ is said to be uniform if its direct connection language can be recognized by a deterministic Turing machine in time $O(\log C(n))$.

We will say that a language is recognized by a uniform family of randomized circuits $\left\{G_{n}\right\}$ if,

- the circuit $G_{n}$ has exactly $q(n)$ random (supplementary) inputs ( $q$ is a polynomial), in addition to the original $n$ inputs.
- for all inputs of length $n$ in the language, $G_{n}$ evaluates to 1 for more than two-thirds of the random inputs, and
- for all inputs of length $n$ not in the language, $G_{n}$ evaluates to 1 for less than one-third of the random inputs.

Our results use uniform circuit characterizations of $N P, P H$ and $\oplus P$. It is known (Venkateswaran, 1988) that $N P$ is the class of languages recognized by a uniform family of semi-unbounded fan-in circuits of exponential size and constant depth with the restriction that the OR gates can have exponential ( $2^{n^{O(1)}}$ ) fan-in but AND gates can have only polynomial fan-in. The characterizations of $P H$ and $\oplus P$ given below can all be obtained in a straightforward manner from the results in (Venkateswaran, 1988; Vinay, Venkateswaran and Veni Madhavan, 1990). We sketch their proofs
for completeness. It should be noted here that this characterization of $P H$ also follows from a parallel random access machine characterization of PH in (Immerman, 1989) and the well known correspondence between such machines and unbounded fan-in Boolean circuits (Stockmeyer and Vishkin, 1984).

Lemma 1 PH is exactly the class of languages recognized by a uniform family of constant depth, exponential size unbounded fan-in circuits over \{AND , OR \}. (We call such circuits PH -circuits.)

Proof. From the above definition of $N P$, it follows that Co-NP is the class of languages accepted by a uniform family of exponential size and constant depth semi-unbounded fan-in circuits with AND -gates with exponential fan-in and OR-gates with polynomial fan-in. A simple induction on $k$ can then be used to prove that $\Sigma_{k}^{p}$ is characterized by a uniform family of exponential size and constant depth circuits with alternating layers of $N P$ and Co-NP circuits, beginning with an $N P$ -layer.

Lemma $2 \oplus P$ is exactly the class of languages recognized by a uniform family of constant depth, exponential size unbounded fan-in circuits over \{ XOR,AND,OR\} with the restriction that the XOR gates can have exponential fan-in but the other gates have polynomial fan-in. (We call such circuits $\oplus P$-circuits.)

Proof. One direction follows from the fact that PARITY-SAT(the set of all conjunctive normal form formulas with an odd number of satisfying assignments) is complete for $\oplus P$. For the other direction, note that it is sufficient to consider a circuit with XOR and AND gates only. A simulation of such a circuit by a polynomial time nondeterministic Turing machine can be done as described in lemma 15 in (Venkateswaran, 1988). The machine maintains a stack to do the circuit evaluation. At an XOR gate, the machine existentially guesses and recursively simulates an immediate predecessor. At an AND gate, the machine guesses the input gate names of the gate in order and pushes them on the stack. (The uniformity machine can be easily modified to verify the order of the inputs of an AND gate.) The machine then recursively simulates the first input of the AND gate. At a circuit input, the machine rejects if the input has value zero. If the circuit input has value one the machine either accepts or pops the gate on the top of the stack and recursively simulates it depending on
whether the stack is empty or not. It is not difficult to see that the time taken by this machine is at.most the maximum AND fan-in raised to the depth and that the machine accepts if and only if the $\oplus P$ circuit evaluates to a one.

Corollary 3 BP. $\in P$ is exactly the class of languages recognized by a uniform family of randomized $\oplus P$-circuits.

## 3 The Result

The lemmas in the previous section together with the following theorem prove Toda's theorem.

Theorem 4 Let $\left\{G_{+}\right\}$be a family of uniform $P H$ - circuits. Then, there is a uniform family of randomized $\oplus P$-circuits that recognize the same language as $\left\{G_{r}\right\}$.

Proof. We assume that the fan-in of all the gates is exactly $2^{p}$. We will replace every unbounded fan-in gate, $v$, by a randomized $\oplus P$-circuit $F_{v}$. Consider an OR gate $v$ in $G_{r}$. We use the following lemma by Valiant and Vazirani (1986). Note that the inner product used is over (XOR , AND ).

Lemma 5 Let $S$ be a nonempty subset of $\{0,1\}^{p}$. Let $w_{1}, \cdots, w_{p}$ be random $p$ bit vectors, and let $S_{i}=\left\{j \mid j \in S, j \cdot w_{1}=\cdots=j \cdot w_{i}=0\right\}$ for each $1 \leq i \leq p$. Then, the probability that $\left|S_{i}\right|=1$ for some $1 \leq i \leq p$ is at least $1 / 4$.

Our randomized circuit has a random $d(r) \times p$ matrix $W$ as supplementary input ( $d(r)$ to be determined later). For $1 \leq i \leq d(r)$, row $i$ of this matrix consists of $p$ entries $w_{i, 1}, \cdots, w_{i, p}$, each of which is a $p$-bit vector. All $F_{v}$ receive the same set of random bits as supplementary input.

Replace $v$ by an OR gate $v^{\prime}$ with $d(r)$ input gates. Let these input gates be denoted as $u_{1}, \cdots, u_{d(r)}$. Each of the gates $u_{i}$ of $v^{\prime}$ is an OR gate. For $1 \leq i \leq d(r)$, the gate $u_{i}$ has as inputs $p$ XOR gates $u_{i k}, 1 \leq k \leq p$. Each of these XOR gates $u_{i k}$ have as inputs $2^{p}$ AND gates $u_{i k j}$ for $0 \leq j \leq 2^{p}-1$. An AND gate $u_{i k j}$ has two inputs: (a) the gate that is the $j$-th input of $v$, and
(b) a subcircuit that verifies that, for all $1 \leq h \leq k$, the inner product of the $p$-bit binary vector representing $j$ with the $p$-bit vector $w_{i k}$ is zero.

If the chosen gate $v$ were an AND gate in $G_{r}$, the above construction is used after transforming $v$ into an OR gate using DeMorgan's law and XOR gates. It is easy to check that the final circuit so obtained is a $\oplus P$-circuit if $d(\tau)$ is chosen to be a polynomial.

Let $\mathcal{H}_{r}$ denote the constructed circuit. We first show that the circuit family $\left\{\mathcal{H}_{r} \mid \tau \geq 1\right\}$ is uniform by giving a scheme to label the gates of $\mathcal{H}_{r}$. The labeling is illustrated for the circuit replacing the gate $v$.

Let $v$ be labeled as $L(v)$ in $G_{r}$. The output gate $v^{\prime}$ is labeled $L(v)$ also. For $1 \leq i \leq d(r)$, let $u_{i}$ be labeled as $[L(v), i]$. For $1 \leq i \leq d(r), 1 \leq k \leq p$, let $u_{i k}$ be labeled as [ $\left.L(v), i, k\right]$. For $1 \leq i \leq d(r), 1 \leq k \leq p, 0 \leq j \leq 2^{p}-1$, let $u_{i k j}$ be labeled as $[L(v), i, k, j]$.

To complete the proof, we have to show that the error committed by the resulting circuit is quite small. Let $C$ be the size of the circuit $\mathcal{H}_{r}$ and let depth $(u)$ denote the depth of the gate $u$ in this circuit. By choosing the set $S$ in lemma 5 to be $\left\{j \mid j\right.$ th input of $v$ in $G_{r}$ evaluates to 1$\}$, it can be verified that the error committed at any gate $u$ in $\mathcal{H}_{T}$ is at most $C^{\operatorname{depth}(u)} *\left(\frac{3}{4}\right)^{d(r)}$. So choosing $d(r)$ to be a suitably large polynomial we can make this error as small as possible. It should be noted that the number of random bits used is $p^{2} \times d(r)$ which is a polynomial since $d(r)$ is.

The corollary below shows that unboundedness does not help over semi-unboundedness for certain uniform families of circuits. The proof follows from the characterization of BP. $\oplus P$ in corollary 3 and the proof of theorem 4, which shows how to replace unbounded fan-in OR and AND gates by randomized $\oplus P$.circuits.

Corollary $6 \mathrm{BP} . \oplus P$ is exactly the class of languages recognized by a uniform family of constant depth, exponential size unbounded fan-in randomized circuits over the basis \{XOR,OR, AND \}.

Acknowledgements: The first and the third author thank the Forschungsinstitut für Discrete Mathematik and Institut für Operations Research of the University of Bonn. The second author is thankful to Gil Neiger, Gary Peterson, Larry Ruzzo and Martin Tompa for some very useful comments.

## References

E. Allender (1989), A note on the power of threshold circuits, in "Proceedings, 30th Annual IEEE Symposium on Foundations of Computer Science", pp. 580-584.
E. Allender and U. Hertrampf (1990), On the power of uniform families of constant depth threshold circuits, in "Proceedings, 15th International Symposium on Mathematical Foundations of Computer Science", pp. 158-164, Lecture Notes in Computer Science, Vol. 452, SpringerVerlag, New York/Berlin.
N. Immerman (1989), Expressibility and parallel complexity, SIAM J. Comput. 18, 625-638.
C.H. Papadimitriou and S. Zachos (1983), Two remarks on the power of counting, in "Proceedings, 6th GI Conference on Theoretical Computer Science", pp.269-276, Lecture Notes in Computer Science, Vol. 145, Springer-Verlag, New York/Berlin.
W.L. Ruzzo (1981), On uniform circuit complexity, J. Comput. System Sci. 22,365-383.
L. Stockmeyer and U. Vishkin (1984), Simulation of parallel random access machines by circuits, SIAM J. Comput. 13, 409-422.
S. Toda (1989), On the computational power of $P P$ and $\oplus P$, in "Proceedings, 30 th Annual IEEE Symposium on Foundations of Computer Science", pp. 514-519.
L.G. Valiant and V.V. Vazirani (1986), NP is as easy as detecting unique solutions, Theoretical Comput. Sci. 47, 85-93.
H. Venkateswaran (1988), Circuit definitions of nondeterministic complexity classes, in "Proceedings, 8th Annual Foundations of Software Technology aand Theoretical Computer Science Conference", pp. 175-192, Lecture Notes in Computer Science, Vol.338, Springer-Verlag, New York/Berlin.
V.Vinay, H.Venkateswaran, C.E.Veni Madhavan (1990), Circuits, Pebbling and Expressibility, in "Proceedings, 5th Annual Conference on Structure in Complexity Theory", Barcelona, Spain, pp. 223-230.

# Circuit Definitions of Nondeterministic Complexity Classes ${ }^{1}$ 

H. Venkateswaran<br>School of Information and Computer Science<br>Georgia Institute of Technology<br>Atlanta, Georgia 30332-0280


#### Abstract

We consider restrictions on Boolean circuits and use them to obtain new uniform circuit characterizations of nondeterministic space and time classes. We also obtain characterizations of counting classes based on nondeterministic time bounded computations on the arithmetic circuit model. It is shown how the notion of semi-unboundedness unifies the definitions of many natural complexity classes.


[^3]
## 1 Introduction

Uniform Boolean circuits have provided a very useful framework to study some of the important issues that arise in Turing machine based complexity theory. Close connections have been established between complexity classes based on uniform circuits with those based on the machine model $[2,6,11,12,14]$. In one direction, complexity classes defined using the circuit model have been characterized using the machine model. NC is a well known example of such a complexity class defined using the uniform Boolean circuit model [11] that has been characterized using the alternating Turing machine model by Ruzzo [14]. In the other direction, traditional complexity classes based on the machine model have been characterized in the circuit model. The definition of the class $P$ using Boolean circuits $[8,12]$ is probably the first such result. Other results of this nature are the characterizations in the circuit model of the classes $A C^{1}$ [16] and LOGCFL [18]. The results by Ruzzo [14] also make it possible to obtain circuit characterizations of complexity classes defined using alternating Turing machines. The work reported here extends these results to characterize classes defined using nondeterministic Turing machines.

In the first part of this paper, we consider restrictions of Boolean circuits and use them to characterize nondeterministic space and time classes. This includes a characterization of nondeterministic time classes on the semi-unbounded fan-in circuit model. Semi-unbounded fan-in circuits, which are Boolean circuits in which the OR gates are allowed arbitrary fan-in and the AND gates have bounded fan-in, have been previously used to define the class LOGCFL [18]. We define skew circuits as Boolean circuits in which all but one input of every AND gate are circuit inputs and use them to characterize nondeterministic space and time classes. Nondeterministic space is defined in terms of the size of such circuits and nondeterministic time is shown to correspond to the depth of these circuits. This should be contrasted with the well known correspondences between deterministic time and Boolcan circuit si\%e [12] and between nondetcrministic space and Boolean circuit depth [2].

In the second part of the paper. we use the monotone arithmetic circuit model to characterize comnting classes hased on mondeterministic time bounded computations. Monotone arithmetic rircuits are arithmetic circuits ofer the domain of non-negative integers and which use only the addition and multiplication operations. An interesting consequence of this characterization is the definition of the well kinw on comting class *P as the set of functions computed by unform families



presented here.

- The circuit characterizations of $N P$ presented here are, to our knowledge, the first uniform circuit characterizations of this important complexity class. Of particular interest is the definition of NP as the class of languages accepted by uniform families of semi-unbounded fan-in circuits of exponential size and log depth. This provides a framework to study some interesting questions about the class NP. Recently, Borodin et al. [3] proved that if a language is accepted by a family of semi-unbounded fan-in circuits of size $Z(n)$ and depth $D(n)$, then its complement is accepted by a family of semi-unbounded fan-in circuits of size polynomial in $Z(n)$ and depth $O(D(n)+\log Z(n))$. Their result does not apply directly to $N P$, since it only shows that $C O-N P$ is accepted by semi-unbounded fan-in circuits of exponential size and polynomial depth. The relevant question here is whether the classes accepted by size $Z(n)$ and depth $o(\log Z(n))$ semi-unbounded fan-in circuits are closed under complement. It is known that the classes accepted by polynomial size and $o(\log n)$ depth semi-unbounded fan-in circuits are not closed under complement [18]. Another complexity question pertaining to $N P$ that can be phrased in this model is its relationship with the other classes definable using semi-unbounded fan-in circuits. A candidate class for comparison would be the class LOGCFL. It is known that LOGCFL can be characterized as the class of languages accepted by uniform families of polynomial size and log depth semi-unbounded fan-in circuits [18]. Therefore, the separation between NP and LOGCFL now becomes a question of the relative power of exponential size and polynomial size semi-unbounded fan-in circuits of logarithmic depth.
- The skew Boolean circuits provide a model to rephrase many of the famous separation questions among complexity classes. Thus the relationship between $P$ and NLOG translates into the question of the relative power of polynomial size Boolean circuits and polynomial size skew Bonlean circuits. The $P$ wersus PSPACE question becomes one of comparing the relative power of polynomial size Boolean circuits and exponential size shew Boolean circuits. As another interesting example. the $N P$ versus PSPACE question can be phrased as the question about polynomial depth for skew Boolean circuits versus polynomial depth for general Boolean circuits.
- The arimmetie characterization of : P presented here is the first alternative characterization of



- These characterizations also make it possible to identify appropriate circuit value problems that are complete for each of these complexity classes.
- The semi-unbounded fan-in circuit model seems useful to capture the definitions of many nondeterministic complexity classes (see table 1).

This paper is organized as follows. Section 1.1 contains some preliminary definitions. Boolean circuit characterizations of nondeterministic space and time classes are in section 2. Some characterizations of nondeterministic time that follow as simple consequences of known results are presented in section 3. A monotone arithmetic circuit characterization of counting classes based on nondeterministic time is presented in section 4.

### 1.1 Preliminaries

Boolean Circuits: A Boolean circuit $G_{n}$ with $n$ inputs is a finite acyclic directed graph with vertices having indegree zero or at least two and labelled as follows. Vertices of indegree zero are labelled from the set $\left\{0,1, x_{1}, x_{2}, \ldots, x_{n}, \bar{x}_{1}, \bar{x}_{2}, \ldots, \bar{x}_{n}\right\}$. All other vertices (also called gates) are labelled either AND or OR. It should be noted that not including negation gates in the definition of a Boolean circuit is done with no loss of generality. Vertices with outdegree zero are called outputs. The evaluation of $G_{n}$ on inputs of length $n$ is defined in the standard way. Typically, only circuits with one output vertex will be considered. This makes it convenient to consider circuits as language acceptors.

The size $C\left(G_{n}\right)$ of a circuit $G_{n}$ is the number of edges in $G_{n}$. The depth of a vertex $v$ in a circuit is the length of a longest path from any input to $v$. The depth of a circuit is the depth of its output vertex.

The language $L_{n}$ accepted by a Boolean circuit $G_{n}$ is the set of all length $n$ strings on which Gin evaluates to one. A framily of circuits is a sequence $\left\{G_{n}^{\prime} \mid n=0,1,2 \ldots\right\}$, where the $n$-th circuit $G_{n}$ has $n$ inputs. The langage $L$ accepted by a family $\left\{G_{n}\right\}$ of circuits is defined as follows: $L=U_{n \geq 0} L_{n}$. where $L_{n}$ is the language accepted by the $n$-th member $G_{n}$ of the family.

Skew Boolean Circuits: La ( $B$ b a Boolean circuit. An AND gater in $G$ is said to be a skere getc if it has at most one input that is not an input of $G$. Without loss of generality, we will assume that all but one of its inputs are inputs to the circuit $G$. We will refer to the input of $v$ that is mot an input to (; as a non-stiew input of $r$. The rircuit (; is sain to be a skere circuit if all ANO gates in it are skew gates. A lamily $\left\{C_{n}\right\}$ of Boolean circuits is said to be a skew circuil frmily if all its members ans skew circuits.

Note: One can define skewness with respect to OR gates also, but we will not pursue that in this paper.

Semi-Unbounded Fan-in Boolean Circuits: A family of Boolean circuits is said to have semi-unbounded fan-in if there exists a constant $c>0$ such that for any circuit in the family, the OR gates in the circuit can have unbounded fan-in and all the AND gates have fan-in at most $c$.

Semi-Unbounded Alternating Turing Machines: An alternating Turing machine is semiunbounded if there are no two consecutive universal configurations along any path in the computation tree of the machine. Without loss of generality, we will assume that every universal configuration of a semi-unbounded alternating Turing machine has exactly two existential configurations as immediate successors.

Uniformity: We will use the following notion of uniformity, called $U_{D}$-uniformity, defined by Ruzzo [14]. Define the direct connection language $L_{D C}$ of a family of Boolean circuits to be the set of strings of the form $\langle n, g, y>$ such that either (i) $g$ and $y$ are gate names and $y$ is an input of the gate $g$, or (ii) $g$ is a gate name and $y$ is the type of the gate $g$, that is, $y$ is one of AND or OR or an input to $G_{n}$ or its negation. A family $\left\{G_{n}\right\}$ of Boolean circuits of size $C(n)$ is said to be uniform if the corresponding direct connection language can be recognized by a deterministic Turing machine in time $O(\log C(n))$.

For the space characterization results in section 2, it would have been sufficient to consider logspace uniformity defined by Borodin and Cook [4]. But a stronger uniformity condition is needed for the time characterization results to avoid the possibility of having a uniformity machine that is more powerful than the class being characterized. Such will be the case, for instance, in theorem 7 if we had used $\log$-space uniformity since $\operatorname{ATIME}(T(n)) \subseteq \operatorname{DSPACE}\left(T^{O(1)}(n)\right)$.

Accepting Subtrees [19]: The notion of an accepting subtrce of a Boolean circuit given an input on which it evaluates to one is analogous to the notion of accepting subtrees of machines.
I.et $B$ be a Boolean circuit, and let $T(B)$ be its tree equivalent. (The tree-equivalent of a graph is obtained by replicating vertices whose outdegree is greater than one until the resulting graph is a tree). Let $x$ be an input on which $B$ craluates to one. An accepting subtree $I$ of the circuit $B$ on input $x$ is a subtree of $T(B)$ defined as follows:

- II includes the output gate.
 as itw immediate prodecessors in $H$.
- for any OR gate $v$ included in $H$, exactly one immediate predecessor of $v$ in $T(B)$ is included as its only immediate predecessor in $H$, and
- any input vertex of $T(B)$ included in $H$ has value one as determined by the input $x$.

It is easy to verify the fact that the circuit $B$ evaluates to one given the input $x$ if and only if there is an accepting subtree of $T(B)$ on input $x$.

Tree-Size [19]: The tree-size measure for Boolean circuits can now be defined analogous to the tree-size measure for alternating Turing machines [14].

The circuit $B_{n}$ is said to have tree-size $Z(n)$ if, for every input $x$ accepted by $B_{n}$, there exists an accepting subtree with at most $Z(n)$ vertices.

Degree: We define the degree of a circuit to be the algebraic degree of the polynomial computed by the circuit. Thus, the constants have degree zero, the circuit inputs have degree one, the degree of an OR vertex is the maximum of the degrees of its inputs, and the degree of an AND vertex is the sum of the degrees of its inputs.

The following lemma [18] establishes a relationship between the degree and tree-size measures for Boolean circuits.

Lemma 1 Let $D(n), Z(n)$, and $d(n)$ be the degree, tree-size, and depth respectively of a Boolean circuit $B_{n}$. Then,

$$
Z(n) \leq D(n) d(n)+1
$$

Proof: The result to be proved also holds when the Boolean circuits considered have unbounded fan-in. Let the OR gates have fan-in at least $k$ and the AND gates have fan-in at least $l$.

Let $x$ be an input accepted by the circuit $B_{n}$. By hypothesis, there is an accepting subtree $H$ of $B_{n}$ of size at most $Z(n)$. Let $v$ be any vertex in $I I$. Then the lemma follows from the clain below.

Claim: Let $Z(v)$ be the number of vertices in the subtree of $I$ rooted at $v, D(v)$ be the degree of $n$, and $d(v)$ be the depth of $r$. Then.

$$
Z(v) \leq D(v) d(u)+1 .
$$

Proof of the Claim: The proof of this clam is be indmetion on the depth of e.
The clam is cleaty tre when the deph of a vertex is zere. Aswme that the claim holds for all werties with depth less than $d(r)$. For the induction step. there are two cases:

Case 1: Let $v$ be an or gate with inputs $v_{1}, \ldots, v_{k}$ with at least one non-constant input. Then,

$$
\begin{aligned}
Z(v) & \leq \max \left\{Z\left(v_{1}\right), Z\left(v_{2}\right), \ldots, Z\left(v_{k}\right)\right\}+1 \\
& \leq \max \left\{D\left(v_{1}\right) d\left(v_{1}\right), D\left(v_{2}\right) d\left(v_{2}\right), \ldots, D\left(v_{k}\right) d\left(v_{k}\right)\right\}+2 \\
& \leq \max \left\{\left(D\left(v_{1}\right), D\left(v_{2}\right), \ldots, D\left(v_{k}\right)\right\}(d(v)-1)+2\right. \\
& \leq D(v) d(v)-D(v)+2
\end{aligned}
$$

The claim follows from this since $D(v) \geq 1$.
Case 2: Let $v$ be an AND gate with inputs $v_{1}, \ldots, v_{l}$ with at least one non-constant input. Then,

$$
\begin{aligned}
Z(v) & =Z\left(v_{1}\right)+Z\left(v_{2}\right)+\ldots+Z\left(v_{l}\right)+1 \\
& \leq D\left(v_{1}\right) d\left(v_{1}\right)+D\left(v_{2}\right) d\left(v_{2}\right)+\ldots+D\left(v_{l}\right) d\left(v_{l}\right)+l+1 \\
& \leq\left(D\left(v_{1}\right)+D\left(v_{2}\right)+\ldots+D\left(v_{l}\right)\right)(d(v)-1)+l+1 \\
& \leq D(v) d(v)-D(v)+l+1
\end{aligned}
$$

The claim follows from this since $D(v) \geq 1 . \square$

## 2 Characterizations of Space and Time Classes

This section contains the characterizations of nondeterministic space and time classes in terms of skew circuits and semi-unbounded fan-in circuits. Theoren 6 relates simultancous space and time bounded nondeterministic classes to simultaneous size and depth bounded skew circuits. In this respect, it is similar to the result of Ruzzo [1.t] relating simultaneous space and time bounded altermating classes to simultaneons size and dept h bounded circuits. However, the correspondence between the tine and depth bonnds in theorem 6 is only within a polynomial as opposed to the correspondence within a constant factor between circuit depth and alternating time shown by R1izzo [L-4].

In the proof of lemma 3 below, we choose to use the alternating Turing machine model instead of directly constrmang a smitmboumded fan-in circuit cormponding to a skew circuit. This is done to simplify the proof sime we can use known simulation technigues. It also provides a new

 of lemmas.

Lemma 2 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$, $\operatorname{NSPACE}, \operatorname{TIME}(S(n), T(n)) \subseteq$ Uniform Skew Circuit SIZE,DEPTH $\left(2^{O(S(n))}, T(n)\right)$.

Proof: Let $L$ be accepted by a nondeterministic Turing machine $M$ in $S(n)$ space and $T(n)$ time. The construction of a circuit family $\left\{G_{n}\right\}$ that accepts the same language as $M$ can be done using standard techniques $[14,18]$. For the sake of completeness, we will outline below the construction of $G_{n}$, the $n$-th member of this family.

The configurations of $M$ can be classified into two types: existential and read. We will assume that $M$ is deterministic while reading inputs.

For $0 \leq t \leq T(n)$, and a configuration $c$ of $M$ using space $S(n)$, there is a gate in the circuit in one of the following forms: $[t, c]$, or $[t, c, i]$, or $[t, c, i, b]$, where $0 \leq i \leq n$ is an integer and $b$ is either zero or one. The first component $t$ in a gate name is used to avoid cycles in the circuit. The type of a gate of the form $[t, c]([t, c, i],[t, c, i, b])$ is OR (OR, AND respectively).

Let $c_{I}$ be the initial configuration of $M$. The output gate is $\left[0, c_{I}\right]$.
The inputs of a gate are constructed as follows. Consider a gate $[t, c]$ corresponding to a nonread configuration $c$ of the machine. If $t+1>T(n)$, it has only one input, namely the constant zero. Otherwise, its inputs are constructed from the set $D$ of all configurations reachable by $M$ in one move from $c$. There will be one input corresponding to each $d \in D$. For any $d \in D$, if $d$ uses space $>S(n)$, then the corresponding input is the constant zero. For all other $d \in D$, there are $t$ wo cases. If $d$ is an existential configuration the corresponding input is the gate $[t+1, d]$ and its inputs are constructed recursively. If $d$ is a read configuration in which $M$ zeads the $i$-th symbol, the corresponding input is an OR gate $[t+1, d, i]$ with two inputs: $[t+1, d, i, 0]$ and $[t+1, d, i, 1]$. The gate $[t+1, d, i .0]$ is an AND gate with two inputs: NOT $x_{i}$, where $x_{i}$ is the $i$-til input, and the gate $[t+2, c]$. where $c$ is the configuration to which $M$ moves from the read configuration $d$, if the $i$-th input read has value zero. The inputs of the gate $[t+2, \epsilon]$ are constructed recursively. The gate $[t+1, d, i, 1]$ is constructed in an analogous fashion.

It is clear from the construction of $G_{n}$ above that it is a skew circuit. The only And gates constructed comerspond to the read configurations of $M$. It is easy to show that $\left\{C_{n}\right\}$ arcepts the same language as.$U$. The size of the resulting circuit is $2^{\circ(S(n))}$. Its depth is $T(n)$.

It can he werified that the direct comedion languge of $\left\{C_{i n}\right\}$ an be rexguized ly a determin-


Lemma 3 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$,

$$
\begin{aligned}
& \text { Uniform Skew Circuit SIZE,DEPTH }\left(2^{O(S(n))}, T(n)\right) \subseteq \\
& \quad \text { Uniform Semi-Unbounded Fan-in Circuit SIZE,DEPTH }\left(2^{O(S(n))}, \log T(n)\right) .
\end{aligned}
$$

Proof: Let $\left\{G_{n}\right\}$ be a uniform family of skew circuits with the given size and depth bounds. Then $\left\{G_{n}\right\}$ has tree-size that is polynomial in $T(n)$. An alternating Turing machine $M$ that simulates $G_{n}$ on an input $x$ of length $n$ can be constructed as in the simulation by Ruzzo [13] of a space and tree-size bounded alternating Turing machine by a space and time bounded alternating Turing machine. The machine $M$ is semi-unbounded and uses space $O(S(n))$, alternations $O(\log T(n))$, and time $T^{O(1)}(n)$. Let the time used by $M$ be $T^{\prime}(n)=T^{2}(n)$ for some constant $a \geq 1$. Furthermore, $M$ is in a normal form such that only one input symbol is read along any path of the machine's computation tree. A uniform family $\left\{H_{n}\right\}$ of semi-unbounded fan-in circuits, with size $2^{O(S(n))}$ and depth $O(\log T(n))$, that accepts the same language as $M$ can be constructed by adapting known techniques [18]. The basic idea of the construction is to make as inputs to an OR (AND) gate all non-existential (non-universal) configurations of $M$ reachable through only existential (universal) configurations.

We will outline the construction of the $n$-th member $H_{n}$ of this family. The configurations of $M$ are assumed to be one of the following three types: existential, universal, and read.

Let $D(n)=\left\lceil\log _{2} T^{\prime}(n)\right\rceil$.
Gates in the circuit $H_{n}$ are all of the form [c]. or [ $\left.d{ }^{\prime}\right]$. or $[c, d]$. or $[s, c, d]$, or $[s, c, d . e]$. where $0 \leq s \leq D(n)$, and $c$. $d$ and $\varepsilon$ are all configurations of $M$. The output gate of $H_{n}$ is $\left[r_{0}\right]$, where $r_{0}$ is the initial configuration of $M$. In general, the type of a gate of the form [c] is OR (AND) if the type of the configuration $c$ is existential (respectively, universal). Given a gate [c]. its inputs are defined as follows.

Case 1: $[c]$ is an OR gate. Its inputs are gates $[c, d]$ for all configurations $d$ that are not existential. Each of the gates $[r, d]$ is an A:D gate and it has two inputs $[0, c, d]$ and [ $\left.d^{\prime}\right]$ defined as follows.

- The gate $[0, c, d]$ is the output of an $D(n)$ depth semi-unbounded fan-in circuit that checks that in . 3 the configuration $d$ is reachable from the configuration $c$ using onty existential configurations of . $1 /$. The following is a deseription of such a reachability circuit [18].
 is the oupht, such that he suberemit chechs that $r$ is rachable from $d$ in (in using a path


If $d$ is an immediate predecessor of $c$ in $G_{n}$, then $[s, c, d]$ is the constant one. Otherwise, if $s+1>D(n)$, then $[s, c, d]$ is the constant zero. Otherwise, the gate $[s, c, d]$ is an or gate. Its inputs are gates $[s+1, c, d, e]$ for all OR gates $e$ in $G_{n}$. Each of the gates $[s+1, c, d, e]$ is an AND gate, and it has the two inputs $[s+1, c, \epsilon]$ and $[s+1, e, d]$. These two subcircuits are constructed recursively.

- The gate [ $d^{\prime}$ ] is an OR gate with a single input [ $d$ ] defined as follows. Suppose $d$ is a read configuration with $a, i$ on its index tape. Then [d] is the $i$-th input to $H_{n}$ if $a=1$, and [ $d$ ] is the complement of the $i$-th input to $H_{n}$ if $a=0$. If $d$ is not a read configuration, then $[d]$ is an AND gate. Its inputs are constructed recursively.

Case 2: [ $c$ ] is an AND gate. Let $d_{1}, d_{2}$ be the existential configurations of $M$ that immediately succeeds the configuration $c$. The inputs to $[c]$ are the OR gates $\left[d_{1}\right]$ and $\left[d_{2}\right]$. The inputs to these two OR gates are constructed recursively.

The circuit $H_{n}$ has size $2^{O(S(n))}$ and depth $O(\log T(n))$. Note that the or gates in $H_{n}$ may have exponential fan-in whereas the fan-in of the AVD gates is bounded by a constant. It is easy to show that $G_{n}$ and $H_{n}$ accept the same language. It is also straightforward to check that the direct connection language for the circuit family $\left\{H_{n}\right\}$ can be recognized by a deterministic Turing machine in time $O(S(n))$.

Lemma 4 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$,

$$
\begin{aligned}
& \text { Uniform Semi-Unbounded Fan-in Circuit SIZE, } \operatorname{DEPTH}\left(2^{O(S(n))}, \log T(n)\right) \subseteq \\
& \quad \operatorname{NSPACE}, \operatorname{TME}\left(S(n) \log T(n), T^{O(1)}(n)\right) .
\end{aligned}
$$

Proof: This follows from the simulation of semi-unbounded fan-in circuits by nondeterministic ansiliary pushown automata by Venkateswaran [18]. In this case, we are interested in the space and time used in the simulation.

Let $l$ be accepted by $\left\{G_{n}\right\}$, a uniform family of semi-unbounded fan-in circuits with size $2^{\circ(S(n))}$ and depth $O(\log T(n))$. Given $x$ of length $n$ a nondeterninistic machine $M$ checks whether the circuit evaluates to one on $x$ by doing a depth-first evaluation. The machine $M$ maintains a stack w do 1 lar cirenit avalualion.

Whogins lhe simulation with the output gate $r_{y}$. Given a gate $r$ and its type. W checks that
 llengater.

Case 1: $v$ is an OR gate. $M$ existentially guesses one of its true inputs $u$ and its type and verifies with the uniformity machine that the guesses are correct. It then recursively checks that the gate $u$ evaluates to one.

Case 2: $r$ is an AND gate. Then it has a constant number, say $k$, inputs. $M$ existentially guesses these inputs, say, $v_{1}, \cdots, v_{k}$, and their types and verifies with the uniformity machine that the guesses are correct. $M$ then pushes the gates $v_{2}, \cdots, v_{k}$ onto the stack. Along with a gate its type is also pushed onto the stack. $M$ then recursively checks that $v_{1}$ evaluates to one.

Case 3: $v$ is an input to the circuit. If its value is zero, $M$ rejects. Suppose $v$ has value one. $M$ makes its final pop move and accepts if the stack is empty. Otherwise, $M$ pops a gate $u$ and its type from the stack and recursively checks that $u$ evaluates to one.

For correctness, it can be shown, by induction, that the output $r_{0}$ of the circuit $G_{n}$ evaluates to one on input $x$ if and only if $M$ accepts starting from $C\left(r_{0}\right)$ and an empty stack [18].

Consider the space used $\mathrm{b} y, M$ on input $x \in L$ of length $n$. In cheching a gate $v, M$ must remember the gate $v$ and its type. If $v$ is an or gate, $M$ needs space to record information pertaining to a true input of $v$. This uses space $O(S(n))$. The space used for the gate $v$ can be reused at the next level of recursion. If $v$ is an AND gate, the information pertaining to all but one of its inputs is stored in the stack. This uses space $O(S(n))$. But, since the depth of the circuit is bounded by $O(\log T(n))$, the stack may have $O(\log T(n))$ such pieces of information using altogether $O(S(n) \log T(n))$ space. The uniformity machine uses $O(S(n))$ space. Therefore, the total space used in the simulation by $M$ is $O(S(n) \log T(n))$.

For the time bound of $M$, we first note that an $y$ accepting subtree of the circuit will have size $T^{O(1)}(n)$. The machine $M$, in verifying whether $G_{n}$ accepts its input. traverses such an accepting tren in a depth-first fashion visiting every vertex at most twice. For each node visited. $M$ uses time $O(S(1))$ to guess the information pertaining to the node and time $O(S(n))$ to invoke the uniformity. machine to verily its guesses. Recall that the uniformity machine is a determinsitic machine using fime $O(S(n))$. Since $S(n) \leq T(n)$, the total time used by $M$ is $T^{O(1)}(n)$.

In the proof of lemma 4 above, the space used for the stack can be completely avoided if the circuit. being simulated are skew circuits. This observation leads immediately to the following lemma:

Lemma 5 for $S(n)=\Omega(\log n) . T(n)=\Omega(n)$, and $S(n) \leq T(n)$.

Lemmas 2 and 5 yield the following theorem:

Theorem 6 For $S(n)=\Omega(\log n), T(n)=\Omega(n)$, and $S(n) \leq T(n)$,

$$
\operatorname{NSPACE}, \operatorname{TIME}\left(S(n), T^{O(1)}(n)\right)=\text { Uniform Skew Circuit SIZE,DEPTH }\left(2^{O(S(n))}, T^{O(1)}(n)\right) . \sqsubset
$$

The following characterizations of nondeterministic time using skew circuits and semi-unbounded fan-in circuits are now immediate from lemmas 2,3 , and 4.

Theorem 7 For $T(n)=\Omega(n)$, the following complexity classes are equal:

1. $\operatorname{NTIME}\left(T^{O(1)}(n)\right)$
2. Uniform Skew Circuit $\operatorname{DEPTH}\left(T^{O(1)}(n)\right)$
3. Uniform Semi-Unbounded Fan-in Circuit SIZE, DEPTH $\left(2^{O(T(n))}, \log T(n)\right)$

As interesting consequences of theorems 6 and 7 , we obtain the following Boolean circuit characterizations of the classes $N L O G, P S P A C E$, and NP.

Corollary 8 1. NLOG $=$ Uniform Skew Circuit $\operatorname{SIZE}\left(n^{O(1)}\right)$.
2. PSPACE $=$ Uniform Skew Circuit $\operatorname{SIZE}\left(2^{n^{0(1)}}\right)$.
3. $N P=$ Uniform Skew Circuit $\operatorname{DEPTH}\left(n^{O(1)}\right)$.
4. $N P=$ Uniform Smi-Unbounded Fan-in Circnit SIZE.DEPTH $\left(2^{n^{O(1)}}, \log n\right)$.

## 3 Other Characterizations of Nondeterministic Time

This section contains some characterizations of nondeterministic time that follow as simple conseguenres of known results. We will only consider bounded fan-in Boolean circuits in this section. lerhaps the most interesting of ihe characterizations lere is the one using the depthand degree moasures for Boolean circuits. This suggests the characterization results in section 4 of connting dasses based on mondeterministic time bounded computations.
 nating Turing machines simutaneonsly using space $O(T(n))$ and tre-size $O(T(m)$ ). This combined
with the simulation by Ruzzo [13] of space and tree-size bounded alternating Turing machines by space and time-bounded alternating Turing machines (used in the proof of lemma 3) provides a new characterization of nondeterministic time bounded classes on the alternating Turing machine model. The close relationship between Boolean circuits and alternating Turing machines [14] also leads to another Boolean circuit characterization of nondeterminstic time in terms of size and treesize. Finally, the correspondence between degree and tree-size for Boolean circuits (see lemma 1) yields yet another Boolean circuit characterization of these classes in terms of degree and depth resources.

We will summarize these three characterizations in theorem 9 below. The proof of this theorem can be reconstructed from the results mentioned.

Theorem 9 For $T(n)=\Omega(n)$, the following complexity classes are equal:

1. $\operatorname{Ntime}\left(T^{O(1)}(n)\right)$
2. Semi-Unbounded ATIME.ALTERNATIONS $\left(T^{O(1)}(n), \log T .(n)\right)$
3. Uniform Circuit SIZE,TREESIZE $\left(2^{T^{O(1)}}(n), T^{O(1)}(n)\right)$
4. Uniform Circuit DEPTH, $\operatorname{DEGREE}\left(T^{O(1)}(n), T^{O(1)}(n)\right)$. $\square$

Thus, for instance, $N P$ has the following characterization in terms of degree and depth of Boolean circuits:

Corollary $10 \times P=$ Uniform Circuit DEPTH.DEGREE $\left(n^{O(1)} \cdot n^{O(1)}\right) \cdot \square$
The Boolean circuit characterization of $\lambda P$ in Corollary 10 should be contrasted with the following bounded fan-in Boolean circuit characterization of PSPACE [2, 14]:

PSPACE $=$ Vniform Circuit DEPTH $\left(n^{O(1)}\right)=$ Uniform Circuit DEPTH.DEGREE( $\left.n^{O(1)} \cdot 2^{n^{O(1)}}\right)$.
Constant Depth Circuits: Before concluding this section, we mention another definition of $\therefore P$ using constant depth ubbouded fan-in circuits. We will show this by exhibiting a unform fanily of constant deptli loolean circuits for the conjunctive normal form satisfiability problem.
I.et SAT denote the language consisting of all strings that are (reasonable) encodings of satisfiable comjunctive nomal form formulas. Let all lengtlor strings in SAT encode satisfiable formulas
 accopts SAT is described below. See figure 1.

- The output of $G_{r}$ is an or gate labelled $[0, n, m]$. This gate evaluates to one on input $x$ if and only if the formula encoded by $x$ is satisfiable.
- The OR gate $[0, n, m]$ has as inputs AND gates labelled $[1, n, m, j]$ for $0 \leq j \leq 2^{n}-1$. An AND gate $[1, n, m, j]$ evaluates to one if and only if the input formula evaluates to one when the variables in the formula are assigned bit values from the integer $j$.
- Each AND gate labelled $[1, n, m, j]$ has as inputs OR gates labelled $[2, n, m, j, k]$ for $1 \leq k \leq m$. An OR gate $[2, n, m, j, k]$ evaluates to one if and only if the $k$-th clause in the input formula evaluates to one when the variables in the formula are assigned bit values from the integer $j$.
- The inputs of an OR gate labelled $[2, n, m, j, k]$ are OR gates labelled $[3, n, m, j, k, p]$ for $1 \leq$ $p \leq n$. An OR gate $[3, n, m, j, k, p]$ is the output of a subcircuit that evaluates to one if and only if the $p$-th variable occurs in the $k$-th clause as a positive (negative) literal and the $p$-th bit of $j$ is one (respectively, zero). If the $p$-th variable does not occur in clause $k$ then a gate of the form $[3, n, m, j, k, p]$ evaluates to zero.

The family of Boolean circuits have size $O\left(m 2^{n}\right)$ and constant depth. The or gates have fan-in at most $2^{n}$ and the AND gates have fan-in at most $m$. It can be verified that the direct connection language for $\left\{G_{r}\right\}$ can be recognized by a deterministic Turing machine in polynomial time, thus showing that this is a uniform family of circuits.

## 4 Monotone Arithmeic Circuits and Counting Classes

This section contains the characterizations of counting classes based on nondeterministic time bounded computations on the monotone arithmetic circuit model. A monotone arithmetic circuit is an arithmetic circuit using only the addition and multiplication operators and whose inputs are nonnegative integers. We will also characterize these classes in terms of the number of accepting subtrens in the Boolean circuit model. As corollaries, we obtain characterizations of the class : $P$ on these models.

### 4.1 Definitions

If will be conmenient to consider Boolean circuits in which every ANI) gate has exactly two inputs.
Monotone Arithmetic Circuits: These ate defined just as Boolran circuits. (thens. for insablen. wery product gate has exactly two inputs). except that the gates compme the sum and


Figure 1: Constant Depth Cubounded fan-in Circuits for CNF Satisfiability
product of their inputs instead of computing the OR and AND functions. Although the results in this section, especially lemma 13 , can be strengthened to handle $n$ bit nonnegative integers as inputs to the circuit, it suffices to consider only single bit inputs.

We will denote a gate computing the sum (product) of its inputs as a PLUS (respectively, MULT) gate.

Uniformity: We will slightly modify the definition of uniformity in section 1.1 to do a parsimonious simulation in lemma 15.

Define the direct connection language of a family $\left\{G_{n}\right\}$ of Boolean circuits to be the set of strings of the form $\langle n, g, y, p\rangle$ such that either (i) $g$ is an OR gate and $y$ is an input of $g$, or (ii) $g$ is an AND gate and $y$ is a left (right) input of $g$ if $p$ is $L$ (respectively, $R$ ), or (iii) $g$ is a gate name and $y$ is the type of the gate $g$. A family $\left\{G_{n}\right\}$ of Boolean circuits of size $C(n)$ is said to be uniform if the corresponding direct connection language can be recognized by a deterministic Turing machine in time $O(\log C(n))$.

The uniformity condition for monotone arithmetic circuits is defined exactly as for Boolean circuits with PLUS (MULT) gates replaced for OF (respectively, AND) gates.

Degree: The degree measure for monotone arithmetic circuits is defined anlaogous to Boolean circuits (see section 1.1). Thus, the constants have degree zero, the circuit inputs have degree one, the degree of a PLUS vertex is the maximum of the degrees of its inputs, and the degree of a MULT vertex is the sum of the degrees of its inputs.

Notations: Let $N$ denote the set of natural numbers.
$\therefore$ function $f:\{0,1\}^{*}-\lambda$ is in :Tniform Circuit SIZE,DEPTH,DEGREE $(Z(n), d(n), D(n))$ if and only if there exists a uniform family $\left\{G_{n}\right\}$ of Boolean circuits of size $O(Z(n))$, depth $O(d(n))$, and (fegren $O(D(n))$ such that for all strings $x$ of length $n . f(x)$ is the number of accepting subtrees of $G$ on inpul $x$.

The other comuting classes are defined in a similar fashion.

### 4.2 The Characterization Results

Ther following fact can be used to set up a correspondence between ] 3oolean and monotone arithmetic dicuits. The proof of his fact is a diecet consequence of the definition of an accepting subtree of a Bualabll circuil (ane serction 1.1).

Fact 11 Let $B$ be a Boolean circuit that evaluates to one an input $x$. Given $x$ as an input, the number of accepting subtrees of $B$ rooted at an OR (AND) gate $v$ is the sum (respectively, product) of the number of accepting subtrees of $B$ rooted at the inputs of $v$.

It may be noted that lemmas 12,13 , and 14 beiow are stronger statements than needed to prove the main results of this section, namely lemma 15 and theorem 17.

Lemma 12 Let $B$ be a Boolean circuit of size $Z$, depth $d$, and degree $D$. Then there exists an arithmetic circuit $A$ of size $Z$, depth $d$, and degree $D$ such that $B$ has $p$ accepting subtrees on an input $x$ on which it evaluates to one if and only if $A$ has value $p$ on input $x$.

Proof Sketch: Given a Boolean circuit $B$, let the arithmetic circuit $A$ be obtained by replacing all the OR (AND) gates of $B$ by PLUS (respectively, MULT) gates. Then the conclusion follows by using fact 11 .

Lemma 13 Let $A$ be a monotone arithmetic circuit of size $Z$, depth $d$, and degree $D$ with $n$ inputs from $\{0,1\}$. Then there exists a Boolean circuit $B$ of size $Z$, depth $d$, and degree $D$ such that $A$ has value $p$ if and only if $B$ has $p$ accepting subtrees given this input.

Proof Sketch: Given a monotone arithmetic circuit $A$, the Boolean circuit $B$ is obtained from $A$ by replacing all PLUS (MULT) gates by OR (respectively, AND) gates. The proof follows by a simple inductive argument.

The circuits in wolved in the lemmas 12 and 13 above can be made uniform thereby showing the following correspondence between monotone arithmetic circuits and Boolean circuits.

Lemma 14 For $Z(n) . D(n)=\Omega(n)$.
$=$ niform Circuit SIZE, DEP「IIIDEGREE $\left(Z^{O(1)}(n) \cdot d(n) \cdot D(n)\right)=$
Guiform Mononone Arithmetic Circuit size deptil.DEGREE( $\left.Z^{O(1)}(n), d(n) \cdot D(n)\right)$
I.emma 1.5 below establishes the correspondence between the number of accepting pathe in nomentmoministic Thring machines and the mumber of accepting subtres of Boolean circuits.

Lemma 15 for $T(11)=\Omega(11)$.

Proof: Let $M$ be a nondeterministic Turing machine that runs in time $T(n)$. By theorem 7 , there exists a uniform family $\left\{B_{n}\right\}$ of $O(T(n))$ depth bounded skew circuits that accepts the same language as $M$. The degrec of $B_{n}$ is $O(T(n))$. This is due to the fact that the degree of a depth $d$ skew circuit cannot excced $d$. Any accepting subtree of $B_{n}$, given an input on which it evaluates to one, is a completely skewed binary tree. We claim that $M$ has $p$ accepting paths on an input $x$ of length $n$ if and only if $B_{n}$ has $p$ accepting subtrees.

To simplify the proof, we will assume that $M$ is deterministic while reading its inputs, and that the immediate successor of a read configuration is an existential configuration.

Let $x$ be an input of length $n$ accepted by $M$. Then $B_{n}$ evaluates to one on $x$. We will show that there is a bijective function that maps the accepting paths in the computation tree of $M$ on input $x$ with the accepting subtrees of $B_{n}$ on input $x$.

Let $p$ be an accepting path of $M$ on input $x$. The starting vertex of $p$ is labelled by the initial configuration $c_{I}$ of $M$. Consider the following subtree $A(p)$ of $B_{n}$ on input $x$. The root of $A(p)$ is the output gate $\left[0, c_{I}\right]$ of $B_{n}$. In general, the construction proceeds as follows. For the $t$-th vertex of $p$ labelled with an existential configuration $c$, pick the corresponding gate $[t, c]$ of $B_{n}$. The configuration $d$ that immediately succeeds $c$ along $p$ is either an existential configuration or a read configuration. If $d$ is an existential configuration pick as the input of the gate $[t, c]$ its input labelled $[t+1 . d]$. Suppose $d$ is a read configuration in which $M$ reads the $i$-th input symbol and moves to an existential configuration $e(f)$ if the $i$-th input is zero (respectively, one). Consider the case when the $i$-th input symbol is zero. (The construction in the case when the $i$-th input symbol is one is analogous.) Then $d$ has the configuration $e$ as its immediate successor along $p$. Pick the gate $[t+1, d, i]$ as the input of the gate $[t, c]$, the Arid gate $[t+1, d, i, 0]$ as the input of $[t+1, d, i]$, and tlic gate $[t+2, c]$ as the input of the gate $[t+1, d . i .0]$. It is easy to see that $A(p)$ is an accepting subtree of $B_{r}$ on input $x$.

The mapping described above from accepting paths of $M$ on iuput $x$ to accepting subtrees of $B_{,,}$on input $r$ is well-defined. We will now argue that it is also a bijective function.

Suppose $\mu$ and $q$ are two distinct accepting paths of $M$ on input $x$. Let $A(p)$ and $A(q)$ be the corresponding subtrees defined by the above mapping. Now. $p$ and $q$ both have the same start verex mamely, the one labelled with the initial confignation $c_{l}$. Let the initial common segment of $p$ and $q$ have $t$ vertices. Let the $t$ th vertex be labelled by the configuration $c$. Then $c$ must be an eximmial ronfiguration. 'The corresponding gates in $-(p)$ and $A(q)$ are labelled by $[t, c]$. Since
 is dillomen from that of $[t, e]$ in $A(q)$.

Suppose $A$ is an accepting subtree of $B_{n}$ on input $x$ of length $n$. We claim that there is an accepting path $p$ of $M$ on input $x$ such that $A$ is the image of $p$ as defined by the mapping above. The path $p$ is constructed as follows. The starting vertex of $p$ is labelled with the initial configuration $c_{I}$. Let $[t, c]$ be a vertex in $A$ where $c$ corresponds to an existential configuration of $M$ on input $x$. There are two cases.

Case 1: Suppose the gate $[t+1, d]$ is included in $A$ as the input of the gate $[t, c]$. Then $d$ is an existential configuration and it is an immediate successor of the configuration $c$ of $M$. Since $A$ is an accepting subtree on input $x$, the gate $[t+1, d]$ evaluates to one on input $x$. It follows that $d$ is an accepting configuration of $M$ on input $x$. Include a vertex labelled $d$ as the immediate successor of the vertex labelled $c$ along $p$.

Case 2: Suppose the gate $[t+1, d, i]$ is included in $A$ as the input of the gate $[t, c]$. Then $d$ is a read configuration that is an immediate successor of $c$. If $[t+1, d, i, 0]([t+1, d, i, 1])$ is the input of $[t+1, d, i]$ that is included in $A$, the $i$-th input symbol must be zero (respectively, one). Consider the case when the $i$-th input symbol is zero. (The case when the $i$-th input symbol is one is analogous.) Let the input of $[t+1, d, i, 0]$ included in $A$ be the gate $[t+2, e]$. Then include the vertex labelled $d$ as the immediate successor of $c$ and the vertex labelled $e$ as the immediate successor of $d$ along $p$. It is easy to verify that $p$ is an accepting path of $M$ on input $x$ and $A$ is an image of $p$ defined by the above mapping.

Conversely, let $\left\{B_{n}\right\}$ be a uniform family of Boolean circuits of depth $T^{O(1)}(n)$ and degree $T^{O(1)}(n)$. Let $M$ be a nondeterministic Turing machine that simulates $B_{n}$ on an input $x$ of length $n$ in a depth-first fashion as in the proof of lemma 4 . The one difference here is the need to ensure that the simulation of an AND gate maintains the correspondence between the number of accepting paths of the machine and the number of accepting subtrees of the circuit. Let $C(v)$ denote the configuration of $M$ as it begins checking the gate $r$.

In simulating an AND gate $r, M$ does the following. It guesses the right input, say $r$, of $r$. werifies with the uniformity machine that the guess is correct, and pushes $c_{2}$ onto the stack. It then wnoses the lelt input. say $v_{1}$, of $t$. verifies with the uniformity machine that the guess is correct. and verifies that $r_{1}$ cvaluates to one. This will guarantee that there is a single accepting path segment from the configuration $C(v)$ to the configuration $C\left(v_{1}\right)$.

Then it follows, from the clam below, that $M$ has $p$ accepting paths on $x$ if and onty if $B_{n}$ has $p$ accopting sulateres on input $x$.
 at $r$, it han paccepting pathe rooled at ( $(v)$ if and only if there are $p$ accepting subtrees of $B_{n}$
rooted at $v$.
Proof of the claim: This is by induction on the depth $d(v)$ of the vertex $v$.
The claim is clearly true for an input vertex $v$ with value one.
Suppose $v$ is an OR gate that evaluates to one on $x$. Let $v_{1}, \ldots, v_{m}$ be its inputs. Let $1 \leq q \leq m$ of these inputs, say $v_{i 1}, v_{i 2}, \ldots, v_{i q}$ evaluate to one on input $x$. The machine $M$, in checking whether $v$ evaluates to one, existentially chooses one of these $q$ inputs. Thus, the number of accepting paths rooted at $C(v)$ is given by the sum of the number of accepting paths rooted at $C\left(v_{i 1}\right), C\left(v_{i 2}\right), \ldots, C\left(v_{i q}\right)$. By induction hypothesis, this sum is equal to the sum of the accepting subtrees rooted at $v_{i 1}, v_{i 2}, \ldots, v_{i q}$. Since this is equal to the number of accepting subtrees of $B_{n}$ rooted at $r$, the claim follows.

Suppose $v$ is an AND gate that evaluates to one on $x$. Let $v_{1}$ and $v_{2}$ be its inputs. By construction, the number of accepting paths rooted at $C(v)$ is equal to the number of accepting paths rooted at $C\left(v_{1}\right)$. That is, if $M$ begins its simulation with the gate $v$, there is a single accepting path segment from $C(v)$ to $C\left(v_{1}\right)$. Thus, the number of accepting paths rooted at $C(v)$ is the same as the number of accepting paths rooted at $C\left(v_{1}\right)$. The machine $M$, in verifying $v_{1}$, traverses an accepting subtree of $B_{n}$ rooted at $v_{1}$. It then pops the vertex $v_{2}$. Hence there is a vertex labelled $C\left(v_{2}\right)$ along every accepting path of $M$ rooted at $C\left(v_{1}\right)$. Therefore, the number of accepting paths rooted at $C\left(v_{1}\right)$ is the product of the number of accepting path segments from $C\left(v_{1}\right)$ to $C\left(v_{2}\right)$ with the number of accepting paths rooted at $C\left(v_{2}\right)$. By induction hypothesis, the number of accepting path segments from $C\left(v_{1}\right)$ to $C\left(v_{2}\right)$ is the number of accepting subtrees rooted at $v_{1}$ of $B_{n}$, and the number of accepting paths rooted at $C\left(v_{2}\right)$ is the number of accepting subtrees of $B_{n}$ rooted at $v_{2}$. It follows that the number of accepting paths rooted at $C(v)$ is the number of accepting subtrees rooted at $v$ of $B_{r}$.

By lemma 1. the tree-size of $B_{n}$ is $T^{O(1)}(n)$. Since $B_{n}$ has size at most exponential in $T^{O(1)}(n)$. it follows. as in the simulation of lemma that $M$ uses time $T^{O(1)}(n)$.

In lemma 15 above, we could have used semi-unbounded fan-in circuits instead of bounded fan-in circuits to obtain the following result:

Theorem 16 For $T(n)=\Omega(n)$.

$$
\begin{aligned}
& \text { XTIME }\left(T^{O(1)}(n)\right)=
\end{aligned}
$$

Inmmas it and is together imply the following theorem:

Theorem 17 For $T(n)=\Omega(n)$,

$$
\mathfrak{N T i m e}\left(T^{O(1)}(n)\right)=
$$

Uniform Monotone Arithmetic Circuit DEPTH, $\operatorname{DEGREE}\left(T^{O(1)}(n), T^{O(1)}(n)\right)$.

As a special case of the above theorem, we obtain the following new characterization of the important counting class $\sharp P$ :

## Corollary 18

$$
\sharp P=\text { Uniform Monotone Arithmetic Circuit DEPTH, } \operatorname{DEGREE}\left(n^{O(1)}, n^{O(1)}\right)
$$

### 4.3 Some Consequences

In this section, we will examine some consequences of the results in section 4.2.
Unique SAT: The Unique SAT problem is defined as follows [10]: Given an instance of SAT, does it have a unique solution? As another interesting corollary of theorem 17 , we can identify an arithmetic circuit value problem that is equivalent to the Unique SAT problem.

Let $M$ be a fixed uniformity machine for a family $\left\{G_{n}\right\}$ of monotone arithmetic circuits of polynomial depth and polynomial degree. Given as input $n$ and an $n$ bit vector $x$, the MCVP1 problem is to determine whether the circuit $G_{n}$ evaluates to one on input $x$.

Corollary 19 There is a $\log$ space transformation from Unique SAT to MCVP1 and vice versa. $\square$

New $N P$-Complete Problems: Theorem 17 suggests a new arithmetic circuit value problem that is complete for $N P$. Let $M$ be a fixed uniformity machine for a fanily $\left\{G_{n}\right\}$ of monotone arithmetic circuits of polynomial depth and polynomial degree. Given as input $n$ and an $n$ bit vector $x$, the MCVP problem is to determine whether the circuit $G_{n}$ evaluates to a non-zero value on input $x$.

Proposition 20 The M(V]' problem is NP-complete.

Characterizing : PSPA(E:Using Monotone Arithmetic Circuits: Ving the known char-
 can be prown using the welnigues in the proof of that lemma:

Lemma 21 For $T(n)=\Omega(\log n)$,

$$
\forall \operatorname{ATIME}\left(T^{O(1)}(n)\right)=\text { Uniform Circuit DEPTH }\left(T^{O(1)}(n)\right) \cdot \square
$$

This lemma combined with lemma 14 and the result by Ladner [9] that $\sharp P S P A C E=\sharp \operatorname{ATIME}\left(n^{O(1)}\right)$ implies the following theorem:

## Theorem 22

$$
\text { PSPACE }=\text { Uniform Monotone Arithmetic Circuit DEPTH }\left(n^{O}(1)\right)
$$

It should be noted here that Bertoni et al. [1] also characterized $¥ P S P A C E$ as the class of functions computed by polynomial time Random Access Machines with the operations of addition, integer subtraction, multiplication, and integer division.

## 5 Conclusion

This work provides a circuit framework in which some well-known open problems of complexity theory can be studied. We considered two constraints on the Boolean circuit model, namely skewness and semi-unboundedness, and used it to define nondeterministic space and time complexity classes. We also considered monotone arithmetic circuits to define counting classes based on nondeterministic time.

The known uniform Boolean circuit characterizations of classes between LOGCFL and PSPACE are summarized in table 1 (the definitions of the classes LOGCFL and $P$ in this table use log-space uniformity). It should not be too difficult to construct entries for classes above PSPACE.

As a consequence of these characterizations, we can define for each of these complexity classes a Boolean circuit value problem that is a natural complete problem for the class. For example, the following circuit value problem is $N P$-complete. I.et $M$ be a fixed uniformity machine for a family $\left\{G_{n}\right\}$ of Boolean circuits of polynomial depth and polynomial degree. Given as input $n$ and an $n$ bit wector $x$. the problem is to determine whether the circuit $G_{n}$ evaluates to one on input $x$.

Wr will conclude with a few remarks about the relevance of the semi-mbomededuess notion for questions in complexity theory. From table 1 . it can be seen that many of the well-known space and time complexity chases lave defimitions in terms of semi-unbomed fan-in circhits. 'lhus. lor
 firroil mack:

| OR fan-in | AND fan-in | SIZE | DEPTH | DEGREE | CLASS |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $n^{O(1)} /$ bounded | $n^{O(1)} /$ bounded | $n^{O(1)}$ |  | $n^{O(1)}$ | LOGCFL |
| $n^{O(1)}$ | bounded | $n^{O(1)}$ | $\log n$ |  | LOGCFL |
| $n^{O(1)}$ | $n^{O(1)}$ | $n^{O(1)}$ | $\log n$ |  | $A C^{1}$ |
| $n^{O(1)} /$ bounded | $n^{O(1)} /$ bounded | $n^{O(1)}$ |  |  | $P$ |
| $2^{n^{O(1)}}$ | bounded | $2^{n^{O(1)}}$ | $\log n$ |  | $N P$ |
| $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}}$ | $n^{O(1)}$ | $n^{O(1)}$ | $N P$ |
| $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}}$ |  | $2^{n^{O(1)}}$ | PSPACE |
| $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}} /$ bounded | $2^{n^{O(1)}}$ | $n^{O(1)}$ |  | PSPACE |

Table 1: Circuit Definitions of Complexity Classes

$$
\begin{aligned}
\text { LOGCFL } & =\text { Uniform Semi-Unbounded Fan-in Circuit SIZE, } \operatorname{DEPTH}\left(n^{O(1)}, \log n\right) \\
P & =\text { Uniforn Semi-Unbounded Fan-iu Circuit SIZE, } \operatorname{DEPTH}\left(n^{O(1)}, n^{O(1)}\right) \\
N P & =\text { Uniform Semi-Unbounded Fan-in Circuit SIZE, } \operatorname{DEPTH}\left(2^{n^{O(1)}}, \log n\right) \\
P S P A C E & =\text { Uniform Semi-Unbounded Fan-in Circuit SIZE,DEPTH }\left(2^{n^{O(1)}}, n^{O(1)}\right)
\end{aligned}
$$

One can define an analogue of the polynomial time hierarchy using semi-unbounded alternating Turing machines. Then. by theorem $9 . N P$ is the class languages accepted by polynomial time semiunbounded alternating Turing machines using $O(\log n)$ alternations. This is interesting because it shows that with the constraint of semi-mboundedness $O(\log n)$ alternations is in NP, whereas withoui this constraint. even constant alternations is not known to be in NP.

## Acknowledgements

I am grateful to Martin Tompa for useful discussions. My thanks are due to Lary Ruzzo whose
 reported here. I am atso thankfil to (ary Peterson for his comments.

## References

[1] Bertoni, A., G. Mauri, and N. Sabadini, Simulations Among Classes of Random Access Machines and Equivalence Among Numbers Succintly Represented, Annals of Discrete Mathematics 25, (1985), 65-90.
[2] Borodin, A., On Relating Time and Space to Size and Depth, SIAM Journal of Computing 6, (1977), 733-743.
[3] Borodin, A., S.A. Cook, P.W. Dymond, W.L. Ruzzo, and M. Tompa, Two Applications of Inductive Counting for Complementation Problems, SIAM Journal of Computing 18, (1989), 559-578.
[4] Cook, S.A., Deterministic CFL's are accepted simultaneously in polynomial time and log squared space, Proc. 11th Annual ACM Symposium on Theory of Computing, (1979), 338-345.
[5] Cook. S.A., A Taxonomy of Problems with Fast Parallel Algorithms, Information and Control 64, 1-3 (Jan/Feb/Mar 1985), 2-22.
[6] Dymond, P.W. and S.A. Cook, Complexity Theory of Parallel Time and Hardware, Information and Computation 80, (1989), 205-226.
[ $\overline{7}$ ] Goldschlager, L.M., The Monotone and Planar Circuit Value Problems are log space Complete for $P$, SIGACT News 9, 2, 197T, 25-29.
[8] Ladner. R. E.. The Circuit Value Problem is log space Complete for P. SIGACT News 7, 1. 1975. 18-20.
[9] Ladner. R. E.. Pulynomial Space Counting Problems, manuscript. May 1986.
[10] Papadimitriou. C. H. and M. Yannakakis. The Complexity of Facets (and some Facets of Complexity). Journal of Computer and System Sciences 28, (198-4). 244-259.
[11] Pippenger, N., On Simultaneous Resource Bounds, Proc. woth Annual Symposium on Foundations of Computer Science, Puerto Rico, 1979.
[12] ]ippenger, N. and M.J. Fischer. Relations among Complexity Measures, Journal of the Association for Computing Marhinery 26. (1979). 361-381.
 (1980). 218.235
[14] Ruzzo, W.L., On Uniform Circuit Complexity, Journal of Computer and System Sciences 22, (1981), 365-383.
[15] Skyum, S. and L.G. Valiant, A Complexity Theory Based on Boolean Algebra, Journal of the Association for Computing Machinery 32, (1985), 484-502.
[16] Stockmeyer, L. and U. Vishkin, Simulation of Parallel Random Access Machines by Circuits, SIAM Journal of Computing 13, (1984), 409-422.
[17] Valiant, L.G., The Complexity of Computing the Permanent, Theoretical Computer Science 8, (1979), 189-201.
[18] Venkateswaran, H., Properties That Characterize LOGCFL, Proc. 19th Annual ACM Symposium on Theory of Computing, (1987), 141-150.
[19] Venkateswaran, H. and M. Tompa, A New Pebble Game That Characterizes Parallel Complexity Classes, SIAM Journal of Computing 18, (1989), 533-549.


Two Dynamic Programming Algorithms for Which Interpreted Pebbling Helps<br><br><br>


#### Abstract

 irto account the ivpes of the gates of he cractis on whet the eames are piatud if simple relationship is estabished beanen the eacoded games and the corte sponcing origina games. This is useiul in stowing that the exionded games allow more cticient pebbing than the eriginal games on cerain natual crams for probloms such as comtext-frec language ocongnison and ramstive dosure of 


## i. Introdection

Pebble games have provided convenient models to study the space and time used in straingt-line implementations of circuits. In this paper, we consider the evaluation of Boolean circuits using one-person and two-person pebbling techniques. The one-person pebble game models the space used in a deterministic evaluation of circuits (see, for example, the survey by Pippenger, 1980). The two-person pebble game models the time used in an allernating evaluation of circuits (Dymond and Tompa, 1985). We consider these two games extended to take into account the types of the gates of the circuits on which the games are played. A pebbling strategy in such an extended game corresponds to an evaluation strategy that depends on the input values. We show a simple relationship between the extended games and the corresponding original games. This relationship uses the notion of an accepting subtree of a Boolean circuit on an input for which it evaluates to one. Specifically, we show that an exiended game on a Boolean circuit with an input for which it evaluates to one corresponds to the original game on an accepting subtree of the circuit on that input. A consequence of this would be that Boolean circuits that have small (say, polynomial size) accepting subtrees have efficient pebblings in the extended games. These efficient pebblings lead to the evaluation of the corresponding Boolean circuits using small space/parallel time.
We then show the following results for Boolean circuits that correspond to the Cocke-Kasami-Younger algorithm for context-free language
$0890.5401 / 9153.00$
Conarigh i: 1991 he Academic Press ine Al? regte of reproductern in any hern reserved
 of a Boolcan matrix:

1. Any one-person ! mo-person. respectives pobbling stategy an :a
 pebbles llinear time. respectively.
2. The circuits cormsponding to Warshall's wanstine wosure alow


The fomer resait shous be anatonsed with the fact but Cocke-kasami-lounger chats have polynomal size accepting subtrees (Ruzzo. 1980. The latier result shouid be contasted whit a linear lower bound on the number of peobles in the one-person pebble game on Warshalls circuits (Tompa. 1982). Thus, the extended games are (exponenlially) more powefful that the original games on the Boolean circuits corresponding to these algorithms.

Although it is easy to construct Boolean circuits for which an exponential separation between the extended and original games can be shown. the results here are interesting because the circuits considered correspond to well-known algorithms for two natural problems. Another point of interest about the results in this paper is that they show that lower bounds based on the one-person pebble game are on the space required for an oblivious evaluation oî circuits, but there may exist other small space evaluations that are not oblivious. Because of the relationship between the space in the one-person pebble game and the time in the two-person pebble game (Tompa, 1983), the same observation holds for small depth implementations of Boolean circuits.

Finally, the result that Warshalls circuits have an efficient parallel implementation is of independent interest. As far as we know, this thas not been observed before. In this context, it may be noted that Warshall's circuits have exponential degree unlike the Cocke-Kasami-Younger circuits which have polynomial degree. (Here, the degree measure refers to the algebraic degree of the formal polynomial computed by the circuit.) Thus known parallelization techniques such as the one by Valiant, Skyum, Berkowitz, and Rackoff (1983) do not seem to be applicable to Warshall's circuits.

## 2. Preliminaries

### 2.1. Boolean Circuits

Defrimions. A Boolean viruin $G_{n}$ with $a$ inputs is a finite acyclic directed graph with vertices having indegree zcro or at least two and labelled as
follows. All vertices with indegree wo wor called gatware labelled wher AND of OR. The impur of the circuit is a set of $2 n$ vertices of indegrec zern labelied as $X_{1}, \bar{X}_{1}, \bar{X}_{2}, \bar{X}_{2}, \ldots, X_{n}, \bar{X}$, , th other wertices or indegree zero ato labelled from the set 0 . 1 , Vertices with outdegree zero are calied ouphous.

The size $C\left(G_{n}\right)$ of a circuit $G_{r}$ is the number of gates in $G_{n}$. The dipht of a vertex $r$ in a circuit is the length of a longest path from any input to i. The depth of a circuit is the depth of is ouput hernex.
 Wone with no loss of gencolity as there is a wel! knowa lechnique for smulating. with a doubling of size dad no morease in depth. a Boolean circuit with negations by a Boolean circuit in which the negations appear onty at the inputs. (See. (or example. Goldscinager, 197.)

Let $x=x_{1}, \ldots, x_{n}$ be a length $n$ bit string. The calue of a vertex $:$ of $G_{n}$. on input $x$, is defined as follows. If r is an input vertex labellied $X^{\prime}$, for some $1 \leqslant i \leqslant n$, the value of $t$ is defined to be the bit $x$, If $t$ is an input veriex labelled $\bar{X}_{;}$, for some $1 \leqslant i \leqslant n$, the value of $r$ is defined to be $\bar{x}$, the complement of the bit $x_{i}$. If $l$ is a gate of type AND (OR), its value is defined to be the value of the Boolean function OR (AND) of its two inputs. The value of a circuit $G_{n}$, on an input $x$, is defined to be the value of its output gate. The evaluation of a circuit $G_{n}$, on some input $x$, consists of computing the value of the circuit on input $x$.

Circuits when considered as language acceptors will be assumed to have only one output vertex. The language $L_{n}$ accepted by such a Boolean circuit $G_{n}$ is defined as the set of all input strings of length $n$ on which $G_{n}$ evaluates to one. A family of circuits is a sequence $\left\{G_{n} \mid n=0,1,2, \ldots\right\}$, where the $n$ the circuit $G_{n}$ has $n$ inputs. The language $L$ accepted by a family $\left\{G_{n}\right\}$ of circuits is defined as follows: $L=\bigcup_{n \geq 0} L_{n}$, where $L_{n}$ is the language accepted by the $n$th member $G_{n}$ of the family.

## 3. The Pebble Games

### 3.1. The Uninterpreted One-Person Game

The one-person pebble game models a deterministic evaluation of circuits. This game, which will be referred to as the uninterpreted oneperson game, has found a wide range of applications in computer sience. The survey by Pippenger (1980) is an excellent source on this topic.

This game is played on the vertices of a directed acyclic graph $G$ according to the following rules: a pebble may be placed on a vertex iff all of its immediate predecessors have pebbles on them, and a pebble may be removed from a vertex at any time. Starting with a pebble-free graph, the goal is to pebble a certain vertex or a ste of vertices at some time.

If $G$ is a circuit computing some function then a flay of this game corresponds 10 evaluating this circuit as follows: placing a pebble corresponds to computing the value at a vertex of the circuit knowing the values of its inputs and storing it in a register. and removing a pebbic corresponds to freeing a register.

Resources. The space in this game is defined to be the maximum number of pebbles on the graph at any point in the game. and time is the number of pebble placements.

### 3.2. The Linmerpreted Two-Person Gam

A two-person pebble game to model computations by alternating Turing machines was introduced by Dymond and Tompa (1985). This game, which will be referred to as the uninterpreted wo-person game. when played on a Boolean circuil can be viewed as an alternating implementation of the circuit.

This game is played on the vertices of a directed acyclic graph $G$ by two players called the Challenger and the Pebbler according to the following rules.

The Challenger begins the game by challenger any vertex. The game now procedds in rounds with each round consisting of a pebbling move followed by a challenging move. In a pebbling move, the Pebbler picks up zero or more pebbles from vertices already pebbled and places pebbles on any nonempty set of vertices. In a challenging move, the Challenger either rechallenges the currently challenged vertex, or challenges one of the vertices that acquired a pebble in the current round.

The Challenger loses the game at a vertex $v$ if, immediately following the Challenger's move, $v$ is the currently challenged vertex and all immediate predecessors of $v$ have pebbles on them. The Challenger loses the game if it loses at some vertex $v$.

If $G$ is thought of as a circuit computing some function, then a play of this two-person game corresponds to an alternating implementation of that circuit, in the following sense. A pebble placed on a vertex $v$ by the Pebbler corresponds to existentially guessing the value computed at $r$. A move of the Challenger corresponds to universally verifying each of those guesses. pius the fact that those guesses lead to the correct value computed at the current challenged vertex.

Resources. As in the case of the one-person game, the space used is the maximum number of pebbles on the graph at any point in the game, and the time used is the number of pebble placements.

The game on a graph with $n$ inputs is said to take space $p(n)$ (time $t(n)$ ) if there is a winning strategy for the Pebbler such that, for all plays by the Challenger. the Pebbler uses at most space $p(n)$ (time $t(n))$.

The time measure in the uninterpreted wo-person pebble game is closely related to the space measure in the uninterpreied one-person pebble game. This reflects the relationship between alternating ime and deterministio space in the case of Turning machines.

Lemma 1 (Tompa. 1983). If the aninterpreted wo-person game can be played on a graph $G$ in time $T$, then the eminterpeted oneperson game can beplated on $G$ aning $I-1$ pebhles.

This lemma is useful in translating lower hounds on space in the oneperson pebble game to lower bounds on time in the two-person pebble game. In fact. we will use this result and prove only a lower bound on the number of pebbles needed for any uninterpreted one-person game on the considered circuit family. In the other direction. our upper bound arguments are for the interpreted two-person game.

### 3.3. The Imerpreted One-Person Game

We will refer to the uninterpreted one-person game modified to take into considerations the gate types of the circuit on which it is played as the interpreted one-person game. This game is essentially the same as a game known as the AND/OR pebble game. This game has been previously used to define complete problems for the class PSPACE (Lingas, 1978; Gilbert, Lengauer, and Tarjan, 1980) and the class P (Immerman, 1979; Sudborough, 1980).

This game is played on a Boolean circuit $G_{n}$ together with its input $x$ according to the following rules:

- A pebble may be placed on an input vertex if its value is one.
- A pebble may be placed on an OR gate if at least one of its immediate predecessors has a pebble on it.
- A pebble may be placed on an AND gate if all of its immediate predecessors have pebbies on them.
- A pebble may be removed from a vertex at any time.

The player wins the game if, starting from a pebble-free graph, it can place a pebble on the output gate of the circuit $G_{n}$ in a finite number of moves. It is easy to verify that the player in this game has a winning strategy on the circuit $G_{n}$ with input $x$ iff the circuit evaluates to one on input $x$.

A circuit $G_{n}$ is said to be pebbleable in the one-person interpreted game in space $p(n)$ (time $(n))$ if, for ail $x \in L$ of length $n$, there is a strategy for the player that uses at most $p(n)$ pebbies ( $n(n)$ time).

### 3.4. The Interpreted Two-Person Gamu

The uninterpreted two-person game was extended by Venkateswaran and Tompa (1989) in two ways. One extension to the game takes into account the types of the gates of the circuit on which the game is piayed The second extension incorporates duality between the two players. This extended two-person game, called the dual interproted gome, was used b! Venkateswaran and Tompa (1989) to characterize two natural paralle! complexity classes. In this paper, we will be concerned with the first mentioned extension of the uninterpreted game. namely considering the types of the gates of the circuit on which it is plaved. We will refer to this modified game as the interpreted wo-person gane.

This game is played by two playes called Player 1 and Player 0 on a Boolean circuit $G_{n}$ together with its inpu1 $\therefore$. The rules of the game. as given below, are analogous to those of the uninterpreted game with Player 0 acting as the Challenger and Player 1 acting as the Pebbler. The modiñcations show up in the winning/lsoing conditions.

Player 0 begins the game by challenging an output vertex. The game now proceeds in rounds with each round consisting of a pebbloing move followed by a challenging move. In a pebbling move, Player 1 picks up zero or more pebbles from vertices already pebbled and places pebbles on any nonempty set of vertices. In a challenging move, Player 0 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 Player 0's move, the current challenged vertex is an input with value one, or an OR gate at least one of whose immediate predecesscrs is pebbled, or an AND gate both of whose immediate predecessors are pebbled. Player 0 wins the game if Player 1 cannot win a finite number of rounds.

It is easy to verify that Player 1 in this game has a winning startegy on the circuit $G_{n}$ with input $x$ if the circuit evaluates to one on input $x$.

A circuit $G_{n}$ is said to be pebbleable in the two-person interpreted game in space $p(n)$ (time $t(n)$ ) if, for all inputs on which $G_{n}$ evaluates to one, there is a strategy for Player 1 such that, for all plays by Player 0 , Player 1 wins using at most $p(n)$ pebbles ( $t(n)$ time).

### 3.5. A Relationship Between the Uninterpreted and Interpreted Games

We now show a simple relationship between the interpreted games on a circuit given an input that it accepts, and the uninterpreted gaes on a certain subgraph of the circuit defined by that input. This uses the notion of accepting subtrees for Boolean circuits.

Accepting Subirees (Venkateswaran and Tompa, 1989). We define accepting subtrees for Boolean circuits by analogy to the notion of 6
accepting subtrees for alternating Turing machans (Ruzzo. 1980). This is done by considering the treceequitalent $T(G)$ of a circuit $G$. obtained by modifying it so that every vertex in $T(G)$. except its output. has outdegree one and $T(G)$ accents the same language as $G$. Let $: \because \in L$ be of length $a$. An arcepting subree $H$ of a circuit $G$ on inpur $x$ is a subree of $T(G)$, its treeequivalent. defined as follows:

- Hincludes the ompar gate.
- for any AND gate $r$ included in $H$. an the immediate predecessors of $r$ in $T(G)$ are included as its immediate predecessors in $H$.
- for any OR gate $v$ included in $H$, exactiy one inmediate predecessor of $t$ in $T(G)$ is included as its only immediate predecessor in $H$, and
- any vertex of indegree zero included in $H$ has value one as determined by the input $x$.

It can be shown, by a straightforward application of the definition of an accepting subtree, that a Boolean circuit $G$ evaluates to one on input $x$ if and only if there exists an accepting subtree of $G$ on input $x$.

Lemma 2 below shows a relationship between the interpreted two-person game played on a circuit given an input that it accepts, and the uninterpreted two-person game on an accepting subtree of the circuit on that input.

Lemma 2. Let $G$ be a Boolean circuit that evaiuates to one on input $x$. Then Player 1 can win in the interpreted two person game on $G$ together with $x$ using space $p$ and time $t$ if there is some accepting subiree $H$ of $G$ on input $x$ such that the Pebbler can win in the uninterpreted two-person game on $H$ with space $p$ and time $t$.

Proof. Let $G$ evaluate to one on input $x$. Let $H$ be an accepting subtree
of $G$ on which the Pebber can win the uninterpreted two-person game using space $p$ and time $t$. Consider the interpreted two-person game on $G$ with input $x$. A winning strategy for Player 1 that uses no more than $p$ with input $x$. A winning strategy for Plaver 1 that uses no more than $p$
pebtles and $t$ steps is to simulate the moves of the Pebbier in the uninterpreted game on $H$. Thus, Player 1 pebbles a gate whenever any of its copies in $H$ are pebbled, and removes the pebble from a gate chanever all of its copies in $H$ become pebble-free.

That Player 1 wins on $G$ in the same round as the Pebbler would win
in the uninterpreted game on $H$ follows from the definition of an accepting subtree and the rules of the two games, as follows. If the Challenger loses at an input of $H$ in the uninterpreted game. then the corresponding input in $G$ must evaluate to one, so Player 1 wins in the interpreted game. If the
-

Challenger loses at an OR gate $t$ of $H$ in the uninterpreted game. it must be because the child of $t$ in $H$ has a pebble on it. In this case. the gate in $G$ corresponding to the child of $r$ in $H$ is also pebbled in the interpreted game, so Player 1 wins in the interpreted game. Fina!!y. is the Chatlenger loses at an AND gate $\because$ of $H$ in the uninterpreted game. then both inputs of the corresponding AND gate in $G$ are pehbled in the interpreted game. so Player 1 wins in the interpreted game.

A relationship between the interpreted and the uninterpreted one-person games analogous to the two-person case of Lemma 2 is expressed in the lemma below. The proof of this lemma is an easy adaptation of the above proof and is imitted here.

Lemma 3. Let $G$ be a Boolean circuit that evaluates to one on input $x$. The player can win in the interpreted one-person game on $G$ together with $x$ using space $p$ and time $I$ if there ix some accepting subtree $H$ of $G$ on input $x$ whit can be pebbled in the uninterpreted one-person game with space $p$ and time $t$.

## 4. Two Algorithms for Which Interpreted Pebbling Helps

It is easy to construct examples of circuits for which the interpreted games are exponentially more powerful than the corresponding uninterpreted versions. We show in this section that Boolean circuits for two natural problems have this behavior.

The upper bounds on time to play the interpreted two-person game on the considered circuits are based on an efficient pebbling of binary trees in the uninterpreted two-person game (see Lemma 5 below). This pebbling of binary trees is a pebbling reformulation of the technique used by Ruzzo (1980) to simulate space and tree-size bounded alternating Turing machines by space and time bounded alternating Turing machines (see also Venkateswaran and Tompa, 1989). This technique is based on a treecutting lemma (see Lemma 4 below) that was first used by Lewis, Stearns, and Hartmanis (1965) to show that context-free languages are recognized by deterministic Turing machines using space $O\left(\log ^{2} n\right)$.

Lemma 4. Let $T$ be a wee with $N$ vernices, each of which has at most no children. Then there is a vertex $s$ of $T$ such that the subiree rooted at $s$ has $p$ vertices, where $N / 3 \leqslant p<2 N / 3+1$.
(For a proof of this lemma, see Lewis, Stearns, and Hartmanis (1965), or Hoperoft and Uliman (1969).)
 dildene. Then the wamerpered wo-person gane on $T$ econ be plated whit Olipethas and Ollog Ni ame
(For a prool, sce Venkateswaran and Tompa. 1989.)
We first define a Boolean circuit lamily that corresponds to the Cocke Kasami-Younger algorithm for context-free language recogntion and shw that at least a linew number of pebbles are required to play the unin. terpeted one-person game on these circuits. The contrast with the interpreted version will then follow by the observation that these circuits have polynomial size accepting subirees. Second. we consider a Boolean circuit family that corresponds to Wiarshall's algorithm for transitive closure and show that these circuits have polynomial size accepting subirees. By a result of Tompa (1982), a linear number of pebbles are required to play the uninterpreted one-person game on these circuits.

### 4.1. The Cocke-Kasami-Younger Circuirs

Let $G$ be a context-free grammar over the alphabet $\{0,1\}$ and let $G$ be in Chomsky normal form. Although the alphabet set is restricted to be $\{0,1\}$ to simplify the presentation of a circuit family that accepts the language generated by $G$, the resulting circuits have sufficiently rich structure to demonstrate the characteristics of context-free language recognition, in particular a linear lower bound on the time to play the uninterpreted game on them.

Given a sting $x=x_{1}, \ldots, x_{n}$ of length $n$, let $x_{i j}$ denote the substring $x_{i}, \ldots, x_{j}$ of $x$. The Cocke-Kasami-Younger dynamic programming algorithm decides whether $x$ is in the language generated by $G$. It does this by determining for each $i(1 \leqslant i \leqslant n)$ and for each $j$ ( $n \geqslant j \geqslant i$ ) and for each nonterminal $A$ whether $A \stackrel{*}{=} x_{i j}$. This algorithm can be described inductively as follows. (See Hopcroft and Ullman, 1979; Ruzzo, 1980.)

For $j=i, A \stackrel{*}{\Rightarrow} x_{i j}$ if and only if $A \rightarrow x_{i}$ is a production in $G$. For $j>i$, $A \stackrel{*}{\Xi} x_{i j}$ if and only if there is some production $A \rightarrow B C$ in $G$ and some integer $k, i \leqslant k<j$ such that $B \stackrel{*}{\Rightarrow} x_{i k}$ and $C \stackrel{*}{\Rightarrow} x_{k+1, j}$. Finally, the membership of $x$ in the language defined by $G$ is determined by checking whether $S \stackrel{*}{\Rightarrow} x_{1 n}$.

For a fixed grammar $G$, a Boolean circuit family $\left\{G_{n}\right\}$ that accepts the language generated by $G$ can be derived from this algorithm. The $n$th member $G_{n}$ of such a circuit family is described below

A gate in the circuit $G_{n}$ has one of the following forms:

- $[A, i, j]$, for some nonterminal $A$ and integers $i$ and $j$ such that $1 \leqslant i \leqslant j \leqslant n$. This is an OR gate that evaluates to one on input $x$ if and only if $A \stackrel{*}{\Rightarrow} x_{i j}$.
- [B. C. i.j,k]. for some integers $i$ and $j$ such that $1 \leqslant i<j \leqslant n$. for all pairs of nonterminals $B$ and $C$ for which $A \rightarrow B C$ is a production for some nonterminal $A$, and for some integer $k$ such that $i \leqslant k<j$. This is an $A N D$ gate that has iwo inputs $[B, i, k]$ and $[C, k+1, i]$, and it evaluates to one on input $x$ if and only if $B \stackrel{*}{\Rightarrow} x_{1 k}$ and $C \stackrel{*}{\Rightarrow} x_{1} \ldots$

The output gate is [S.1.n]. where $S$ is the start symbel in the grammar $G$. The inputs to a gate of the form [.A. i, i] are. for $i<j$. all gates of the form [B.C. $i, j, k$ ]. where $i \leqslant k<j$ and $A \rightarrow B C$ is a production in the grammar. The gate [A, i. i] has a single input which is one of the following:

- the constant 1 if both the productions $A \rightarrow 0$ and $A \rightarrow 1$ are in the grammar, and the constant 0 otherwise.
- $X_{i}$, the $i$ th input if $A \rightarrow 1$ is a production in the grammar,
- $\bar{X}_{i}$, the negation of the $i$ th input if $A \rightarrow 0$ is a production in the grammar.

For each context-free grammar in Chomsky normal form, there is one such Boolean circuit family that can be derived from the Cocke-KasamiYounger algorithm. The objective here is to show that there is a contextfree grammar for which the uninterpreted game on the corresponding circuits takes at least linear time.

Consider the grammar $G$ with a single nonterminal $C$, the terminal alphabet $\{0,1\}$ and the productions $C \rightarrow C C$ and $C \rightarrow 1$. It should be noted that, for each $n$, the language generated by the grammar $G$ has a single string, namely $1^{n}$. That is, the corresponding circuits in the family compute the AND function. The construction, as described above, of the $n$th member $G_{n}$ of the circuit family $\left\{G_{n}\right\}$ corresponding to this grammar is presented below. This circuit is quite similar to the graphs corresponding to some other dynamic programming algorithms for problems such as optimum binary search trees (Aho, Hopcroft, and Uliman, 1974). The general circuit described above will be referred to as a CFL-circuit to distinguish it from the circuit referred to as a DP-pyramid below.

In describing this circuit, a gate of the form $[C, i, j]$ will be denoted as $C_{i, j}$ and a gate of the form [ $C, C, i, j, k$ ] will be denoted as $C_{i, i, i}^{k}$
There are three types of vertices: OR, AND, and input vertices. The input vertices are labeled $C_{i, i}$ for $1 \leqslant i \leqslant n$. All non-input vertices in $G_{n}$ have indegree at least two. Given two vertices labeled $C_{i, k}$ and $C_{k+1, j}$ for some $i, j, k$ in the range $1 \leqslant i \leqslant k<j \leqslant n$, there is an AND verlex labeled $C_{i, j}^{k}$ with these two vertices as immediate predecessors. The $j-i$ AND vertices $C_{i, j}^{k}$ for $i \leqslant k \leqslant j-1$ form the inputs of an OR vertex labeled as $C_{i, 1}$,

This circuit will be referred to as a DP-pyramid. The DP-pyramid 10

for $n=4$ is shown in Fig. 1. Note that in these circuits the OR gates have fan-in at least two while the AND gates have fan-in two.

Theorem 6. The uninterpreted one-person game on the DP-pyramid $G_{n}$ with $n$ inputs requires $\Omega(n)$ space.
Proof. A subgraph of the given DP-pyramid will be picked and the lower bound will be proved for this subgraph. The theorem then follows by the simple fact below.

Fact 7. Let $G=(V, E)$ be a graph with bounded indegree. Let $G^{\prime}=\left(V^{\prime}, E^{\prime}\right)$ be a subgraph of $G$. Then, if any vertex of $G$ can be pebbled with $p$.pebbles in the uninterpreted one-person gae, then any vertex of $G^{\prime}$ can be pebbled using at most $p$ pebbles in the uninterpreted one-person game.
(1) A subgraph of a DP-pyramid is picked level by level as described below. Level zero consists of the input vertices. At level $s, 1 \leqslant s<n$, for all $i$ and $j$ such that $0 \leqslant i<j \leqslant n$ and $j-i=s$. retain from the original graph only the following vertices and the edges between them: $C_{i, j}, C_{i, j}^{i}, C_{i, j}^{j-1}$. This causes the deletion of $(j-i-2)$ AND vertices $C_{i, j}^{i+1}, \ldots, C_{i, j}^{j, 2}$. For
$1-i \geqslant 2$, for the AND vertex $C$. delete the edge from its immediate predecessor $C_{1, \ldots}$ Similarly, for $i-i \geqslant 2$. for the $A N D$ vertex $C_{1,1}^{1-1}$ delete the edge from its immediate predecessor $C_{\text {, }}$.. Note that at any level both $O R$ and AND vorites are included. Let the subgraph so picked he denoted as $H_{n}$. See Fig. 2 for the subgraph $H_{*}$ corresponding to the DP-pyramid $G_{*}$ shown in Fig. 1.
(2) The lower bound prod for the graph $H$ follows an argument that was frrs used by Cook 11974, io prove such a lower bound for a class of graphs called pyramid graphs. Initially. all paths from inputs to $C_{1, \text {, are }}$ pebbie-free. When $C_{1, \text {, }}$ is pebbled no paths from inputs to $C_{1, n}$ are pebblefree. Consider the first move that results in every such path having a pebble. This must involve pebbling an input vertex of a path $p$ that was pebble-free just before this move. Now, consider the set $P_{1}$ of rertices in path $p$ consisting of the inpul vertex, the AND vertex at level 1 , and the $n-2$ OR vertices above level 1 . With each vertex $x \in P_{1}$, there is associated a unique path $p_{x}$ that coincides with $p$ in the segment from $C_{1, n}$ through $x$ and disjoint from $p$ in the segment from $x$ to an input vertex. Furthermore, these $n$ paths can be so picked that they are also disjoint from each other on the segments not in $p$. It must be true at this point that not two


Fig. 2. Pyramid in a DP-pyramid.
of these $n$ disjoint paths can share a pebble. This is because. they cen have no input vertex in common. any non-input vertex that is common to them must be a vertex in the pebble-free path $p$ and the considered move involves pehbling an input vertex. Therefore. after this move at least $n$ vertices must have pebbles on them.

Corollary 8. The minterpered wo-person game on the DP-promid $G_{n}$ with $n$ inpurs recuines $\Omega(a)$ ima.

Proof. This follows from the above theorem and Lemmal.
TheOREM 9. Lel $G$ be a fixed comexifree grammar in Chomsky nomal form and let $\left\{G_{n}\right\}$ be the family of CFL-circuits that accept the language generated by G. Given an mput sming $x$ of length $n$ in the language, the interpresed two-person gane on $G_{n}$ can be played in $O(l o g n)$ time issing a $\mu$ constant number of pebbles (the plaver in the interpreted one-person game on $G_{n}$ can win using $O(\mathrm{~J} O \mathrm{~g} n)$ space $)$.

Proof. There is an accepting subtree $H$ of $G_{n}$ on input $x$ that is a binary tree of linear size. (Actually, this corresponds to a parse tree for .x.l From Lemma 2 above, Player 1 can win the interpreted two-person game in space $p$ and time $t$ if the Pebbler can win the uninterpreted two-person game on this accepting subtree within these resources. That the Pebbler can so win in the uninterpreted two-person game follows from Lemma 5. The one-person version can be proved similarly using Lemmas 1, 3, and 5.

### 4.2. The Warshall Circuits

Let $A$ be an orde $n$ Boolean matrix. Warshall's algorithm to compute the transitive closure $A^{*}$ of $A$ is given below (see, for instance, Aho, Hoperoft, and Ullman, 1974):

$$
\begin{aligned}
& \text { FOR } 1 \leqslant i, j \leqslant n, C_{i j}^{0} \leftarrow(I \text { OR } A)_{i j} \\
& \text { FOR } k \leftarrow 1 \text { TO } n \mathrm{DO} \\
& \qquad \text { FOR } 1 \leqslant i, j \leqslant n \mathrm{DO} \\
& \qquad C_{i j}^{k} \leftarrow C_{i j}^{k-1} \text { OR }\left(C_{i k}^{k-1} \text { AND } C_{k j}^{k-1}\right) ;
\end{aligned}
$$

Here $J$ is the order $n$ identity matrix.
A Boolean circuit family $\left\{G_{m}\right\}$ corresponding to this algorithm can be defined in a straightforward manner. Let $m=n^{2}$. The non-input vertices in $G_{m}$ are labeled as either $C_{i j}^{k}$ or $C_{i k}^{k}$ for $1 \leqslant i, j, k \leqslant n$. The input vertices are labeled as $C_{i j}^{0}$ for $1 \leqslant i, j \leqslant n$. The vertices labeled $C_{i,}^{\prime}$ for $1 \leqslant i, j \leqslant n$ are out-

 verses with two mputs: $C^{*}$ and $C_{2}$
 linear number of pebbles in the uminterpreded ane-pe-son pebbie game



 roned at $C^{\prime}$.

Proni. It is clear that the existence of an accepting subtree rooted at $C_{i,}^{\prime}$ on input 4 guarantees that $C_{\text {", }}^{\prime \prime}$ evaluates :o one.

In the other direcion, let $H$ denote the directed graph with vertex set $\{1.2 . . . . i n\}$ and adjacency matrix A. Let $C_{0}^{\prime \prime}$ evaluate to one on input A for some $l \leqslant i, j \leqslant n$. Then there is a simple path $P$ from vertex $i$ to veriex $j$ in the graph $H$. Let $k$ be the maximum intermediase vertex along this path for some $1 \leqslant k \leqslant n$. Then in the circuit $G_{m}$ the vertices $C_{i j}^{4}$, for all $k \leqslant g \leqslant n$. will evaluate to one. An accepting subtree rooted at $C_{i}^{\prime \prime}$ begins as a chain of vertices from $C_{i j}^{n}$ to $C_{i j}^{k}$.

Now the path $P$ can be divided into two segments $P_{1}$ and $P_{2}$ such that $P_{\text {: }}$ is a directed simple path from vertex $i$ to vertex $k$ and $P_{2}$ is a directed simple path from vertex $k$ to vertex $j$. All intermediate vertices in these two segments will be at mosi $k-1$. Therefore, the vertices $C_{i j}^{k-1}$ and $C_{k j}^{k-1}$ in the circuit $G_{m}$ will evaluate to one. Hence, the AND vertex $C_{i k j}^{k}$ will also evaluate to one. The child of $C_{i j}^{k}$ in the accepting subtree being constructed will be the vertex $C_{i k j}^{k}$ which, in turn, has as its children in this tree the two vertices $C_{i k}^{k-1}$ and $C_{k j}^{k-1}$. Note that the paths $P_{1}$ and $P_{2}$ do not share any vertices. We can repeat the argument above for the vertices $C_{i k}^{k-1}$ and $C_{k j}^{k-1}$ to obtain a binary tree that is an accepting subtree of $G_{m}$ rooted at $C_{i}^{n}$.

The accepting subtree so constructed has polynomial size since all its vertices have distinct labels.

Theorem il below now foliows as in the case of Theorem 9.
Theorem 11. Let A be an order $n$ Boolean matrix and let the (i,i)th entry of is transitive closure be one for some $1 \leqslant i, j \leqslant n$. Let $m=n^{2}$. Let $G_{\mu}$ be the Warshall circuit that computes the transitite closure of order $n$ Boolean matrices. On input $A$, with the initial challenge on $C_{i j}^{n}$ of $G_{m}$. Player 1 can win the interpreted wo-person game in Ollog n) time using a constant number of pebhles. (On input A. the plaver in the interpreted one-person game on $G_{n}$ can win using $O(\log n)$ space $)$.

## 5. Cowcimding Remarks

We have considered one-person and iwo-person pebble games on Boclean circuits that take into account the gaic types. These exicnocd games are useful in discovering new parallel implementations of sequential algorithms. The result about the Warshall circuits in this paper is an example of such a parallelization. One direction for furter research is io identify matural problems and algorithms for these probiems which can be parallelized in this mamer. It would also be interestong to identify natural circuits which are hard for interpreted pebbling.

Finally, we note that the result that the warshall circuits have polynomial size accepting subtrees shows an asymmetry between the tree-size and the degree measure for Boolean circuits. It is known that polynomial size circuits with polynomial degree have polynomial size accepting subtrees (Venkateswaran, 1987). But, as exemplified by the Warshall circuits. it is not neessary that the degree should be polynomial fro accepted subtrees to be polynomial. This is interesting because in the case of homogeneous Boolean circuits degree and accepting tree size can be seen to be polynomially related. (A Boolean circuit is homogeneous if all inputs of all OR gates in the circuit have the same degree.)

## Acknowledgments

I am greatly indebted to Larry Ruzzo and Martin Tompa for fruitul discussions concerning this material and for pointing out some of the subtleties involved in the circuits for contextfree language recognition. I am also grateful to Richard Ladner for his valuable comments My grateful thanks are also due to the two anonymous referees whose critical comments and suggestions have helped in revising this paper.

Part of this work was done at the Unjversity of Washingion, where it was supported by the National Science Foundation under Grants DCR-8301212 and DCR-8352093, at the Georgia Institute of Technology, where it was supported by the Nationa Science Foundation under Grant CCR-8711749, and ai the Indian Institute of Science, Bangalore, India.

Received May 12. 1987; final mandiscript received January 17. 1990

## References

Aho, A. V., Hopcroft, J. E.. and Ullman. I. D. (1974), "The Design and Analysis of Computer Algorithms," Addition-Wesley. Reading. MA
Cook. S. A. (1974). An observation on time-storage wade-off. J. Compur Sinum Sci. 9. 308-316
Dymond. P. W. AND TOml'A, M. (1985), Speedups of delerministic machines by synchronous parallel machines. J. Compur. Sy:rem Sci. 30. 149-161

Gilbert, J. R., Lengaüer, T. and Tarian, R. E. (1980). Pebling problem is complete in polynomial space, SLAM J. Comput. 9, 513-524.
Goluschlager, L. M. (1977). The monotone and planar circuit value problems are log space complete for P, SIGACT Nen's 9, 25-29.
Hopcroft, J. E., and Ullman, J. D. (1969), "Introducion to Languages, Automata, and Computation," Addition-Wesley, Reading, MA.
Hopcroft, J. E., and Ullman, J. D. (1979), "Introduction to Languages, Automata, and Computation." Addition-Wesley, Reading. MA.
Immerman. N. (1979). Length or predicate calculus formulas as a new complexity measure, in "Proceedings, 20th Annual IEEE Symposium on the Foundations of Compuier Science," pp. 337-347.
Lewis, P. M., Stearns. R. E., and Hartmanis, L. (1965), Memory bounds for recognition of contexi-free and context sensitive languages, in "Froceedings, 6th Annual IEEE Symposium on Switching Theory and Logic Design," pp. 191-202.
Lingas, A. (1978), A PSPACE-complete problem related to a pebble game, in "Proceedings, 5th colloqujum on automata, languages and programming." pp. 300-321.
Pippenger, N. (1980), Pebbling. in "Proceedings of the Fijth IBM Symposium on Mathematical Foundations of Computer Science," IBM Japan.
Ruzzo, W. L. (1980), Tree-size bounded alternation, J. Compul. Systent Sci. 20, 218-235.
Sudorough, I. H. (1980), The complexity of path problems in graphs and path systems of bounded bandwidth, in "Proceedings, Workshop on Graph Theoretic Concepts in Computer Science," Bonn.
Tompa, M. (1982), Two familiar transitive closure algorithms which admit no polynomial time, sublinear space implemétations, SIAM J. Comput. 11, 130-137.
Tompa, M. (1983), Manuscript on 2 -person pebble game.
Vallant, L. G., Skyym, S., Berkowitz, S., and Rackoff, C. (1983), Fast parallel computation of polynomials using few processors, SIAM J. Comput. 12, 641-644.
Venkateswaran, H. (1987), Properties that characterize LOGCFL, in "Proceedings, 19th Annual ACM Symposium on Theory of Computing," New York, pp. 141-150.
Venkateswaran, H., and Tompa, M. (1989), A new pebble game that characterizes parallel complexity classes, SIAM J. Compur. 18, 533-549.

# Circuits, Pebbling and Expressibility* 

V. Vinay ${ }^{\dagger}$<br>H. Venkateswaran ${ }^{\ddagger}$<br>C.E. Veni Madhavan ${ }^{\S}$

Technical Report GIT-CC-90-32

[^4]
## 1 Introduction

We give characterizations of nondeterministic complexity classes such as NP and PSPACE and the classes in the polynomial time hierarchy in the two-person pebble game model [VT89]. These characterizations motivate the definitions of these classes using firstorder sentences extending the results in [Im82]. It is shown that the role-switches resource in the pebble games closely model the levels of the polynomial time hierarchy. These characterizations are made possible by explicitly considering circuit-size in the pebbling characterizations and the size of the underlying universe in the first-order characterizations.

A dual interpreted game to model parallel computations was defined in [VT89]. They used this game to obtain characterizations of parallel complexity classes such as LOGCFL and $A C^{1}$. This paper carries this work further to obtain characterizations of the class NP and the classes in the polynomial time hierarchy in the game model. A resource called role-switches was used in the dual game [VT89] to capture the difference between computations in the classes LOGCFL and $A C^{1}$. Subsequently, Borodin et al. [BCDRT89] showed that constant number of role-switches do not help when the underlying circuits have polynomial size. We show that role-switches model the alternating time hierarchy more accurately and thus their collapse implies the collapse of hierarchies such as the poiynomial time hierarchy. Specifically, we show that the $k$-th level of the polynomial time hierarchy uses $k-1$ role-switches. In this respect, it is very similar to a recent result in [JK88] that shows that for all $k \geq 1$, the $k$-th level of the polynomial time hierarchy coincides with the $(k+1)$-st level of a certain alternating auxiliary pushdown hierarchy. To get our results, we generalize the dual game to consider both the size and the fan-in of the circuits on which the game is played. This makes it possible to extend the pebble game to exponential size circuits and unbounded fan-in circuits. The extended game provides a unified framework in which the earlier pebble game characterizations of the classes LOGCFL and $A C^{1}$ [VT89] and the new characterizations can be expressed.

We also give a uniform first-order sentence characterization of NP and PSPACE. These are definitions over an exponential universe. The characterization of PSPACE here when compared with the one in [Im82] shows an interesting tradeoff between the size of the underlying universe and the size of a sentence. One of the objectives of this work was to explore the relationship between two-person pebble games and ex-
pressibility using first order sentences. These results suggest that the number of variables correspond to the number of pebbles and the size of the formula corresponds to pebbling time (rounds).

This work was motivated by the semi-unbounded fan-in circuit characterization of NP in [Ve88]. The results here illustrate the importance of the notion of semi-unboundedness. Semi-unbounded fan-in circuits (exponential fan-in for OR gates and polynomial fan-in for AND gates) of constant depth characterize NP whereas unbounded fan-in circuits of constant depth characterize classes in PH. Thus semiunboundedness captures an essential difference between the computations in NP and PH. It is interesting to note that the circuit characterization of PH is one of the first uniform circuit characterizations of this very important complexity hierarchy. It may also be noted that the classes defined by constant depth semi-unbounded fan-in circuits (polynomial fan-in $O R$ gates and $\log n$ fan-in AND gates) and unbounded fan-in circuits (polynomial fan-in for both $O R$ and $A N D$ gates) at the low-level are known to be different.

One of the contributions of this paper is that it shows the robustness of all the complexity classes which have very similar definitions in the models that we have considered, namely, Boolean circuits, pebble games and logic. They isolate some modelindependent abstract properties that the computations in these classes seem to possess.

## 2 Game Characterizations

In this section, we present the characterizations of the class NP and the polynomial time hierarchy using the dual interpreted game model of [VT89]. The game characterizations use uniform Boolean circuit definitions of NP and PH.

### 2.1 Definitions and Notations

$\Sigma_{k}$-Unbounded fan-in Boolean circuits: This is a family of unbounded fan-in circuits in which the output is an unbounded fan-in OR gate and along any path, from any circuit input to the output gate, no more than $k-1$ unbounded alternations occur. (Note that the gates that do not have unbounded fan-in have constant fan-in.) For $k=1$, such a family of circuits will be called semi-unbounded fanin circuits. We will assume that any circuit in this family can be divided into $k$ distinct layers such that a
gate $v$ is in layer $p$ if and only if the maximum number of unbounded alternations along any path from the output gate to $v$ is $p-1$.

A family of $\Pi_{k}$-Unbounded fan-in Boolean circuits is defined in a dual fashion.

We will assume that the circuit families considered are all $U_{D}$-uniform [Ru81]. See the paper by Ruzzo [Ru81] for a definition of this uniformity notion.
Let SemiUnbounded USIZE,DEPTH $(Z(n), D(n))$ denote the class of languages accepted by a uniform family of semi-unbounded fan-in circuits with size $O(Z(n))$ and depth $O(D(n))$. The classes Unbounded USIZE,DEPTH ( $Z(n), D(n)$ ) are defined similarly. We will also be interested in unbounded fan-in circuit families in which the AND gates have polynomial fan-in. Let Unbounded USIZE,DEPTH,AND $(Z(n), D(n), f(n))$ denote the class of languages accepted by a uniform family of unbounded fan-in circuits with size $O(Z(n))$, depth $O(D(n))$ and, in which, all $A N D$ gates have fan-in at most $f(n)$.

### 2.2 The dual interpreted two-person pebble game

This game, introduced in [VT89], is played by two players called Player 0 and Player 1 on the vertices of a bounded fan-in Boolean circuit $G_{n}$ together with its input $x$. The objective of Player 0 (Player 1 ) is to establish that the output of the circuit evaluates to 0 (1). Thus, a pebble placement or challenge on a gate $v$ by Player 0 (Player 1 ) corresponds to asserting that $v$ evaluates to 0 (1). At any point, one of the players takes on the role of the Challenger and the other that of the Pebbler. The role of a player is automatically determined as part of the circuit information as follows. The gates in $G_{n}$ are partitioned into two sets, those of "challenge type" 0 and those of "challenge type" 1. A challenge placed on a gate of challenge type 0 (challenge type 1) causes Player 0 (Player 1) to be the Challenger in the next round. It is assumed that this additional bit per vertex is available as part of the circuit description.

A challenge by Player 0 (Player 1) will be referred to as a 0 -challenge ( 1 -challenge). Similarly, a pebble placed by Player 0 (Player 1) will be referred to as a 0 -pebble (1-pebble).

Rules: The initial challenge is on the output gate. 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 current challenged vertex is an input with value 1 , or an $O R$ gate at least one of whose immediate predecessors is 1 -pebbled, or an AND gate all of whose immediate predecessors are 1pebbled. Player 0 wins if, immediately following the Challenger's move, the current challenged vertex is an input with value 0 , or an $O R$ gate all of whose immediate predecessors are 0 -pebbled, or an AND gate at least one of whose immediate predecessors is 0 pebbled. The winner in an infinite play of the game is the player who has been the Pebbler for only finitely many rounds.
Resources: The four resources of interest in a play of this game are: space, time, rounds, and role switches.
The game on a circuit $G_{n}$ with input $x \in L$ of length $n$ is said to use space $p(n)$ (time $t(n)$, rounds $r(n)$, role switches $s(n)$ resp.) if and only if there is a strategy for Player 1 such that, for all plays by Player 0 , Player 1 wins using at most $p(n) 1$-pebbles ( $t(n)$ 1-pebble placements, $r(n)$ rounds in which Player 1 is the Pebbler, $s(n)$ role switches between pebbling and challenging roles resp.). Resources when $x \notin L$ are defined by interchanging Player 0 and Player 1. A circuit $G_{n}$ with input $x$ is said to be pebbleable in space $p(n)$, time $t(n)$, rounds $r(n)$, role switches $s(n)$ if the winner has a winning strategy using no more than $p(n)$ pebbles, $t(n)$ pebble placements, $r(n)$ alternations between the players and $s(n)$ role-switches between the pebbling and challenging roles. Note that only the winner's resources are counted.

### 2.3 Extensions to the dual interpreted game

We now consider extensions of the dual interpreted game to facilitate playing the game on Boolean circuits that have exponential size and/or unbounded fan-in.

To extend the game to exponential size circuits, we introduce a purely syntactic parameter called
weight. For our results here, the weight of a pebble is $O(\log Z(n))$, where $Z(n)$ is the size of the circuit on which the pebble game is played. This helps to distinguish between complexity classes which have otherwise the same pebbling resource characteristics.

For playing the dual interpreted game on unbounded fan-in circuits, we will first introduce a simple rule about challenge types of gates that is sufficient for the purposes of this paper.
Rule (*): Any unbounded fan-in $O R(A N D)$ gate is of challenge type 0 (1).

The modifications needed to extend the game to unbounded fan-in Boolean circuits are reflected in the way resources are counted. For this purpose, we consider two possibilities. One possibility is to use rule (**) below. This is motivated by the observation that a gate with fan-in $f$ can be regarded, for our purposes, as a bounded fan-in circuit of depth $\log f$.

Rule (**): If the game is lost at an unbounded fan-in gate, the pebbler of that round is charged $\log \log f$ rounds and time, where $f$ is the fan-in of that gate.

The other possibility is to not use this rule. In other words, the resources for playing the game on bounded fan-in Boolean circuits and unbounded fanin Boolean circuits are treated the same way. We will refer to this as the unit-cost game model.

These two variations on counting resources lead to two different pebble game characterizations of NP and PH.

### 2.4 The Characterization Results

Let $\Sigma(\Pi)-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}(p(n), r(n) . s(n), w(n))$ be the class of languages $L$ accepted by : uniform family $\left\{G_{n}\right\}$ of Boolean circuits of size $2{ }^{(w(n))}$, wherein Player 1 (Player 0) begins the game as the Pebbler, and such that $G_{n}$ is pebbleable in $p(n)$ pebbles, $r(n)$ rounds, and $s(n)$ role switches.

## Let

$\mathrm{U} \Sigma(\mathrm{U} \mathrm{\Pi})-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}(p(n), r(n), s(n), w(n))$ be the class of languages $L$ accepted by a uniform family $\left\{G_{n}\right\}$ of Boolean circuits of size $2^{O(\omega(n))}$, wherein Player 1 (Player 0) begins the game as the Pebbler, and such that $G_{n}$ is pebbleable in $p(n)$ pebbles, $r(n)$ rounds and $s(n)$ role switches in the unit-cost model.

Note: The classes are defined in terms of rounds rather than time. This seems more natural when nonconstant pebbles are used. In the case when constant
pebbles are used, it is easy to see that the number of rounds and the time differ only by a constant factor.

We drop the $\Sigma(\Pi)$ prefix if either player can begin the game as the Pebbler.

Theorem 1 below follows from corollaries $5,7,10$, 12,14 and 15 below.

```
Theorem 1.
    \(\mathrm{NP}=\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), \log n, 0, n^{O(1)}\right)\).
    2. \(\mathrm{NP}=\)
    \(\mathrm{U} \Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(n^{O(1)}, O(1), 0, n^{O(1)}\right)\).
    3. \(\Sigma_{k}^{p}=\)
    \(\Sigma\) - PB,RND, \(\mathrm{SW}, \mathrm{WT}\left(O(1), \log n, k-1, n^{O(1)}\right)\).
4. LOGCFL \(=\)
    \(\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}(O(1), O(\log n), 0, O(\log n))\).
5. \(A C^{1}=\)
    \(\Sigma\)
    PB,RND,SW,WT \((O(1), O(\log n), O(\log n), O(\log n))\).
```

The pebbling characterization of PH in theorem 1 above should be contrasted with the results in [BCDRT89] where they show that constant roleswitches may not help when polynomial size circuits are considered.

It is interesting to look at other classes defined by uniform families of exponential circuits that are not constant depth. Thus, for instance, we can define $N A C^{1}$ to be the class of languages recognized by alternating Turing machines in polynomial space and alternation depth $O(\log n)$. By a result of Cook and Ruzzo [Co85], this class is equivalent to the class of languages accepted by uniform families of unbounded fan-in circuits of size $2^{n^{\circ(2)}}$ and depth $O(\log n)$. This class which is contained in PSPACE is interesting because it contains NP and is closed under complement. A pebbling characterization of $N A C^{1}$ is given in theorem 2 below whose proof follows from corollaries 12 and 14.

## Theorem 2

$N A C^{1}=\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), \log n, \log n, n^{O(1)}\right)$.

Finally, a pebbling characterization of PSPACE is given in theorem 3 below whose proof follows from corollaries 5 and 14.

Theorem 3 PSPACE $=\mathrm{PB}, \mathrm{WT}\left(O(1), n^{O(1)}\right)$.

### 2.5 Pebbling Semi-Unbounded Fan-in Circuits

Pebble games on semi-unbounded fan-in circuits are interesting because many natural complexity classes have definitions using semi-unbounded fan-in circuits [Ve87, Ve88]:

Facts:
LOGCFL $=$ SemiUnbounded USIZE,DEPTH $\left(n^{O(1)}, \log n\right)$.
$N P=$ SemiUnbounded USIZE,DEPTH ( $\left.2^{n^{O(1)}}, \log n\right)$.
$\mathrm{NP}=$
Unbounded USIZE,DEPTH,AND ( $\left.2^{n^{\circ(1)}}, O(1), n^{O(1)}\right)$.
$\mathrm{P}=$ SemiUnbounded USIZE,DEPTH ( $n^{O(1)}, n^{O(1)}$ ).
PSPACE =
SemiUnbounded USIZE,DEPTH ( $2^{n^{\circ(1)}}, n^{O(1)}$ ).
Considering a general semi-unbounded fan-in circuit family of size $Z(n)$ and depth $D(n)$, we have the following result:

## Theorem 4

SemiUnbounded USIZE,DEPTH $(Z(n), D(n)) \subseteq$ $\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}$
$(O(1), \max (\log \log Z(n), D(n)), 0, \log Z(n))$.
Proof Sketch: We use the following definition of NP: SemiUnbounded USIZE,DEPTH ( $\left.2^{n^{\circ(1)}}, \log n\right)$. All gates in the circuit have challenge type 0 . Let the circuit evaluate to 1 on the given input. Consider a depth-first pebbling of a proof in the circuit. Since the $A N D$ gates are bounded, by Rule ( ${ }^{* *}$ ), the time taken by Player 1 to pebble the circuit would be no more than $\max (\log \log Z(n), D(n))$ using a constant number of pebbles. If the circuit evaluates to 0 , the Player 0 wins without using any resources.

Considering unbounded fan-in circuits in which the OR gates are restricted to have bounded fan-in, it is straightforward to prove a dual version of theorem 4 above. So, we have the following corollaries:

Corollary 5 1. LOGCFL $\subseteq$ $\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}(O(1), O(\log n), 0, O(\log n))$.
2. $\mathrm{NP} \subseteq$

$$
\Sigma-\overline{\mathrm{P}}_{\mathrm{B}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), O(\log n), 0, n^{O(1)}\right) . . .}
$$

3. CONP $\subseteq$

$$
\Pi-\mathrm{PB}, \mathrm{RND}^{2}, \mathrm{SW}, \mathrm{WT}\left(O(1), O(\log n), 0, n^{O(1)}\right)
$$

4. PSPACE $\subseteq$

$$
\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), n^{O(1)}, 0, n^{O(1)}\right)
$$

5. $\mathrm{P} \subseteq$
$\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), n^{O(1)}, 0, O(\log n)\right)$.

To obtain an alternative pebbleing characterization of NP, we have the following theorem:

## Theorem 6

Unbounded USIZE,DEPTH,AND $(Z(n), D(n), f(n)) \subseteq$ $\mathrm{U} \Sigma$ - PB,RND,SW,WT$(f(n), D(n), 0, \log Z(n))$.

Proof Sketch: The proof is similar to that of theorem 4 , when the following definition of NP is used: Unbounded USIZE,DEPTH,AND ( $2^{n^{\circ(1)}}, O(1), n^{O(1)}$ ), All gates in the circuit have challenge type 0 . Let the circuit evaluate to 1 on the given input. Consider a depth-first pebbling of a proof in the circuit. Player 1 can win the game using at most $f(n)$ pebbles in $D(n)$ rounds since the AND gates have fan-in at most $f(n)$. If the circuit evaluates to 0 , the Player 0 wins without using any resources.

This theorem yields the following corollary:

## Corollary 7 NP $\subseteq$

$\mathrm{U} \Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(n^{O(1)}, O(1), 0, n^{O(1)}\right)$.
To obtain a pebbling characterization of the polynomial time hierarchy, we begin with a uniform Boolean circuit characterization of the polynomial time hierarchy.

Theorem $8 \Sigma_{k}^{P}=$
$\Sigma_{k}$ - Unbounded USIZE,DEPTH ( $\left.2^{n^{\circ(1)}}, \log n\right)$.
Proof Sketch: Let $L$ be a language in $\Sigma_{2}^{P}$ and $M$ be an NP machine with an NP oracle that accepts $L$. We will assume that $M$ makes an oracle query only once along a computation path. Using the circuit characterization of NP in terms of uniform $\Sigma_{1}$ Unbounded fan-in circuits and CONP in terms of $\Pi_{1}$ Unbounded fan-in circuits, we can combine them to obtain a two-layered circuit that simulates $M$.

In the other direction, a uniform $\Sigma_{2}$-Unbounded fan-in circuit of size $2^{n^{\circ(1)}}$ and depth $O(\log n)$ can be evaluated by an NP machine $M$ with an NP oracle as follows: $M$ existentially guesses a proof in the circuit till it reaches an unbounded $A N D$ gate at which point it will simulate a CONP machine to verify that this $A N D$ gate is accepting. $\square$

The resources for playing the pebble game on the circuits defining PH is given by the following theorem:

## Theorem 9

$\Sigma_{k}$ - Unbounded USIZE,DEPTH $(Z(n), D(n)) \subseteq$ $\Sigma$ - PB,RND,SW,WT
$(O(1), \max (\log \log Z(n), D(n)), k-1, \log Z(n))$.

Proof sketch: It is clear that only odd (even) numbered layers have unbounded fan-in $\operatorname{OR}(A N D$, respectively) gates. Since all gates in the odd (even) numbered layers are assigned challenge type 0 (challenge type 1 , resp.), the game can be confined to one layer using $O(1)$ pebbles and $k-1$ role-switches. Since any one layer is a semi-unbounded fan-in circuit or its dual, the result follows.

Corollary $10 \Sigma_{k}^{P} \subseteq$
$\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), O(\log n), k-1, n^{O(1)}\right)$.

### 2.6 Pebbling Unbounded Fan-in Circuits

Considering unbounded fan-in circuits, we can prove the following analog of theorem 4:

## Theorem 11

Unbounded USIZE,DEPTH $(Z(n), D(n)) \subseteq$
玉 - PB,RND,SW,WT
$(O(1), \max (\log \log Z(n), D(n)), D(n), \log Z(n))$.
The following two corollaries of this theorem are now immediate:

Corollary 12 1. $A C^{1} \subseteq$
$\Sigma$
PB,RND,SW,WT $(O(1), O(\log n), O(\log n), O(\log n))$.
2. $N A C^{1} \subseteq$
$\Sigma$
PB,RND,SW,WT $\left(O(1), O(\log n), O(\log n), n^{O(1)}\right)$.

### 2.7 Simulating the Game hv an Alternating Turing Machi

The following theorem, which gives : :sources used by an alternating Turing machin imulate the game, generalizes theorem 11 in [V: $\}$ and can be proved by slightly modifying the proof of that theorem.

Theorem 13 If $L$ is accepted by a uniform family $\left\{G_{n}\right\}$ of bounded fan-in circuits of size $Z(n)$ such that $G_{n}$ is pebbleable in $p$ pebbles, $t$ time, and $r$ rounds in the dual game, then $L$ is accepted by an alternating Turing machine within space $O(p \cdot w(n))$, time $O(\max (t \cdot w(n), w(n) \cdot \log w(n))$ and alternations $O(\max (r, \log w(n))$. Here, w(n) is taken to be $\log Z(n)$. If, in addition, Player 1 is always the Pebbler, then $L$ is accepted within space $O(p \cdot w(n))$ and tree-size $\max \left(w^{2}(n), p^{O(r)}\right)$.

Proof Sketch: The proof is analaogous to that of theorem 11 in [VT89]. Recall now that the direct connection language of the circuits involved can be recognized in time $O(w(n))$ by a deterministic Turing machine, since the circuits are $U_{D}$-uniform. But, this can be simulated by an alternating Turing mar chine with space $O(w(n))$, time $O(w(n) \cdot \log w(n))$, alternations $O(\log w(n))$ and tree-size $O\left(w^{2}(n)\right)$.
The following corollaries now follow from known relationships.

## Corollary 141.

$\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}(O(1), O(\log n), 0, O(\log n))$
$\subseteq$ LOGCFL.
2. $\Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), O(\log n), 0, n^{O(1)}\right)$ $\subseteq \mathrm{NP}$.
3. $\mathrm{U} \Sigma-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(n^{O(1)}, O(1), 0, n^{O(1)}\right)$ $\subseteq \mathrm{NP}$.
4. $\Pi-\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), O(\log n), 0, n^{O(1)}\right)$ $\subseteq \operatorname{CONP}$.
5. $\Sigma$ PB,RND,SW,WT $(O(1), O(\log n), O(\log n), O(\log n))$ $\subseteq A C^{1}$.
6. $\Sigma$

PB,RND,SW,WT $\left(O(1), O(\log n), O(\log n), n^{O(1)}\right)$ $\subseteq N A C^{1}$.

The simulation of a $k-1$ role switch game in exponential size circuits by a $\Sigma_{k}^{P}$ machine is captured by the following corollary.

## Corollary $15 \mathrm{\Sigma}$

$\mathrm{PB}, \mathrm{RND}, \mathrm{SW}, \mathrm{WT}\left(O(1), O(\log n), k-1, n^{O(1)}\right) \subseteq \Sigma_{k}^{P}$.
Proof sketch: We show this for $k=2$. An NP machine can simulate the game until a role-switch occurs. When the role switch does occur, Player 0 becomes the Pebbler and there are no more role switches. The outcome of the game, given its current configuration, can thus be determined by an NP oracle.
Remarks: It is straightforward to give a pebbling characterization of the polynomial time hierarchy in the unit-cost game model analogous to such a characterization of the class NP (see theorem 1). Such a characterization is possible because $P H$ can be characterized as Unbounded USIZE,DEPTH ( $2^{n^{\circ(1)}}, O(1)$ ). It is also not too difficult to define uniform $A C^{\circ}$ in the pebble game model. The details will appear in the full version of this paper.

## 3 Logic Characterizations

The main result in this section is the characterization of NP using first order sentences. In [Im82], two resources on first order sentences, namely variables and size were introduced to obtain characterizations of simultaneous resource bounded classes. In [ $\operatorname{Im} 81, \operatorname{Im} 82, \operatorname{Im} 87]$, it is assumed that all variables carry no more than $\log n$ bits of information. Motivated by the results in the previous section, we introduce variables which carry $w(n) \geq \log n$ bits of information.

We also define uniformity for first order sentences by introducing the notion of a direct connection language analogous to those for Boolean circuits [Ru81]. All the symbols in the formula are indexed and since a variable may occur in more than one place, the index distinguishes them. Note that not more than $\log Z(n)$ bits are necessary to index a formula with at most $Z(n)$ symbols. Queries such as, "Is variable $v$ at position $p$ universally quantified?" can all be answered by the uniformity machine. In the case where constant number of variables are used, the syntactic uniformity from [Im82] can also be used.

### 3.1 The Characterization results

Let VAR,SIZE,WT( $V(n), Z(n), W(n))$ denote a sequence of uniform first order sentences $\left\{F_{n}\right\}$ where $F_{n}$ has $V(n)$ variables, $O(Z(n))$ symbols and the quantifiers range over a universe whose cardinality is $2^{O(W(n))}$. Let VAR,SIZE,WT $(B \forall)(V(n), Z(n), W(n))$ be defined as above except that now the universal quantifiers range over a Boolean universe.

We prove the following theorems whose proofs follow from corollaries 18,20 and 22.

## Theorem 161.

$\mathrm{NP}=\mathrm{VAR}, \mathrm{SIZE}, \mathrm{WT}(B \forall)\left(O(1), O(\log n), n^{O(1)}\right)$.
2. $\operatorname{PSPACE}=\mathrm{VAR}, \operatorname{SIZE}, \mathrm{WT}\left(O(1), n^{O(1)}, n^{O(1)}\right)$.

The characterization of PSPACE by Immerman [Im82], when phrased using the weight resource would be VAR,SIZE,WT $\left(O(1), 2^{n^{\circ(1)}}, \log n\right)$. Thus these two characterizations of PSPACE provide a weight-size tradeoff.

The characterization results above will be proved by relating first order expressibility to alternating Turing machine resources.

Theorem 17 For $W(n) \geq \log n, S(n) \geq \log n$,
$\operatorname{ASPACE}, \operatorname{TREESIZE}(S(n), Z(n)) \subseteq$
VAR,SIZE,WT $(B \forall)\left(O\left(\frac{S(n)}{W(n)}\right), \frac{S(n)}{W(n)} \cdot \log Z(n), W(n)\right)$.

Proof: The proof is adapted from the second inclusion in theorem B .1 of [Im82] with modifications needed to accomodate the weight resource. The space used by the machine is $S(n)$ and every variable contains $W(n)$ bits of information. Hence, no more than $O\left(\frac{S(n)}{W(n)}\right)$ variables are needed to code any configuration. The size of the sentences will be $O\left(\frac{S(n)}{W(n)} \cdot \log Z(n)\right)$.

Corollary 18 1. LOGCFL $\subseteq$
VAR,SIZE,WT $(B \forall)(O(1), \bar{O}(\log n), O(\log n))$.
2. $\mathrm{NP} \subseteq$

VAR,SIZE,WT $(B \forall)\left(\left(O(1), O(\log n), n^{O(1)}\right)\right.$.

To characterize PSPACE we consider the relationship between first order expressibility and time bounded alternating Turing machines. In one direction, we have the following theorem. We omit the easy proof.

Theorem 19 For $W(n) \geq \log n, S(n) \geq \log n$,
$\operatorname{ASPACE}, T \operatorname{TME}((, S)(n), T(n)) \subseteq$
VAR,SIZE,WT $\left(O\left(\frac{S(n)}{W(n)}\right), T(n), W(n)\right)$.

Corollary 20 1. $\mathrm{P} \subseteq$
VAR,SIZE,WT $\left(O(1), n^{(1)}, O(\log n)\right)$.
2. PSPACE $\subseteq$

VAR,SIZE, $\mathrm{WT}\left(O(1), n^{O(1)}, n^{O(1)}\right)$.

The containments in the other direction follow from the theorem below whose proof is omitted from this extended abstract.

Theorem 21 If $L$ is expressible by a uniform family of senetences $\left\{F_{n}\right\}$ that uses $V(n)$ variables, $T(n)$ size and $W(n)$ weight, then $L$ is accepted by an alternating Turing machine within space $O(V(n) \cdot W(n))$ and time $O(T(n) \cdot W(n))$. If, in addition, the universal quanifiers are Boolean, then $L$ is accepted by such a machine with treesize $c^{T(n)}$ for some constant $c$.

## Corollary 221.

VAR,SIZE,WT $(B \forall)(O(1), O(\log n), O(\log n))$
$\subseteq$ LOGCFL.
2. VAR,SIZE,WT $(B \forall)\left(O(1), O(\log n), n^{O(1)}\right)$ $\subseteq$ NP.
3. VAR,SIZE,WT $\left(O(1), n^{O(1)}, O(\log n)\right)$ $\subseteq P$.
4. VAR,SIZE,WT( $\left(O(1), n^{O(1)}, n^{O(1)}\right)$ $\subseteq$ PSPACE.

## 4 Open Problems

We will conclude by stating some open problems.

- Do role switches in two-person pebble games help? It is known that for certain polynomial size circuit hierarchies constant number of role switches do not help. But, our characterization of PH in terms of role switches suggest that taking weight into consideration may alter this situation. It would also be quite interesting to identify circuits for natural problems for which role switches help.
- The circuit characterization of complexity classes suggests the definition of new classes. We defined one such class $N A C^{1}$ that seemed like a good analog of $A C^{1}$. An interesting question here is to identify natural complete problems for this and other such classes. In this connection, it is worth mentioning that Chandra and Tompa [CT88] have shown that a class of short two-person games are complete for $A C^{1}$. These problems may suggest similar problems complete for $N A C^{1}$.
- Semi-unboundedness versus unboundedness: Semi-unboundedness seems like a useful concept to capture the computations in many natural complexity classes [Ve88]. An important question in this area concerns the relationship between this notion and that of unboundedness. For instance, in the uniform Boolean circuit model, this may shed light on the relationship between NP and PH.
- First order expressibility versus second order expressibility: It is well known that NP is identical with the class of second order existential formulas [Fa74]. What is the link between the first order characterization of NP in this paper and second order formulas?
- Tradeoffs between weight and size: We have exhibited a weight and size tradeoff in the first
order characterizations of PSPACE. What are some general tradeoff relations between weight and size?


## References

[BCDRT89] Borodin, A., S.A. Cook, P.W. Dymond, W.L. Ruzzo, and M. Tompa, Two Applications of Inductive Counting for Complementation Problems, SIAM Journal of Computing 18, (1989), 559-578.
[Co85] Cook, S.A., A Taxonomy of Problems with Fast Parallel Algorithms, Information and Control 64, 1-3 (Jan/Feb/Mar 1985), 2-22.
[CT88] Chandra, A.K. and M. Tompa, The complexity of short two-person games, IBM Technical report RC19495 (1988) (To appear in Discrete Applied Mathematics).
[Fa74] Fagin, R., Generalized First-order Spectra and Polynomial time Recognizable Sets, in Complexity of Computation, ed. R. Karp, SIAM-AMS Proceedings 7, (1974), 27-41.
[Im81] Immerman, N., Number of Quantifiers is better than number of tape cells, JCSS 22, (1981), 65-72.
[Im82] Immerman, N., Upper and Lower Bounds for First Order Expressibility, JCSS 25, (1982), 76-98.
[Im87] Immerman, N., Languages that capture complexity classes, SIAM J. of Computing 16, (1987), 760-778.
[JK88] Jenner, B. and B. Kersig, Characterizing the Polynomial Hierarchy by Alternating Auxiliary Pushdown Automata, Proc. STACS, (1988), LNCS 294, 118-125.
[Ru80] Ruzzo, W.L., Tree-Size Bounded Alternation, JCSS 20, (1980), 218-235.
[Ru81] Ruzzo, W.L., On Uniform Circuit Complexity, JCSS 22, (1981), 365-383.
[SV84] Stockmeyer, L. and U. Vishkin, Simulation of Parallel Random Access Machines by Circuits, SIAM Journal of Computing 13, (1984), 409-422.
[Ve88] Venkateswaran, H., Circuit definitions of nondeterministic complexity classes, Proc.

8th Annual conference on Foundations of Software Technology and Theoretical Computer Science, Pune, India, December 1988.
[VT89] Venkateswaran, H. and M. Tompa, A new pebble game that characterizes parallel complexity classes, SIAM Journal of Computing 18, 3, (June 1989), 533-549. (Also in Proc. 27th Annual Symposium on Foundations of Computer Science, Toronto, 1986.)
[Ve87] Venkateswaran, H., Properties that characterize LOGCFL, Proc. 19th Annual Symposium on Theory of Computing, New York, 1987 (To appear in Journal of Computer and System Sciences).

# Experimental Evaluation of Algorithmic Performance on Two Shared Memory Multiprocessors * 

Anand Sivasubramaniam Gautam Shah Joonwon Lee<br>Umakishore Ramachandran H.Venkateswaran<br>College of Computing<br>Georgia Institute of Technology<br>Atlanta, GA 30332. USA.


#### Abstract

The results of experimenting with three parallel algorithms on the Sequent Symmetry architecture and the BBN Butterfly architecture are reported. The main objective of this study is to understand the impediments to the efficient implementation of parallel algorithms, developed for theoretical models of parallel computation, on realistic parallel architectures. Scheduling, task granularity, and synchronization are the issues that are explored in implementing these algorithms on the two architectures. In the case of BBN Butterfly, which is a distributed shared memory architecture, data distribution in the distributed memories is also studied. The key findings are that synchronization is not a significant cost for the algorithms we studied on the two architectures; the bus is not a bottleneck for the configuration of the Sequent machine that we experimented with; and a fairly simple minded data distribution may be as good as any other on the BBN Butterfly.


## 1 Introduction

Parallel Computation provides some of the most challenging problems in computer science. The term parallel computation covers a broad spectrum of research ranging from purely theoretical models for complexity analysis of parallel algorithms, to detailed system performance issues of large problems that lend themselves to parallel implementations. Both ends of the spectrum have one thing in common, namely, to understand the performance potential of parallel computation. A computation is expressed as a task graph and the objective is to determine the speedup that is realizable for the computation. While the theoretical models are concerned with the asymptotic limits of computing in parallel, the system-oriented studies are concerned with determining the best heuristic mapping for a computation on a target architecture that would result in the best (average case) performance. Each study has its merits and de-merits. The asymptotic limits give a ceiling for maximum achievable performance for a given

[^5]algorithm based on an abstract model of parallel arclitectures. The average case results are useful for determining what is achievable in reality.

Understanding the performance of parallel computation requires a knowledge of the capabilities of the underlying parallel architecture. Further, the performance limits depend on the mapping of the problem on to the parallel architecture. Theoretical models abstract away real life limits such as the number of processors, synchronization requirements, scheduling and data distribution to derive the asymptotic limits. On the other hand, system-oriented studies are so concerned with mapping the algorithm to real architectures that it is difficult to know from the results of such studies where the parallelism inherent in the algorithm has been lost. The aim of this study is to address some of the issues in the interface between theory and architecture, from the point of view of algorithmic performance.

Parallel algorithms for certain problems theoretically guarantee a certain amount of speedup. But when these algorithms are implemented on existing architectures, the results may not agree with the expected theoretical speedup. Some inherent features in the algorithm, its implementation, and the hardware capabilities of the machine together contribute to the slow-down. The parallel algorithms usually assume a certain minimum number of processors to be available with an underlying interconnection topology between them. To implement such algorithms, we may have to make do with a limited number of processors and simulate the assumed interconnection. The language run time and the operating system may further introduce synchronization costs not inherent in the algorithm but are necessary to implement them on the parallel machine. And lastly, the hardware capabilities like synchronization primitives, memory access times and caching strategies may introduce further slow-down.

One straightforward way to understand the architectural impact on parallel computation is to implement algorithms with intrinsic parallelism on parallel architectures and interpret the results with respect to the above factors. Therefore, we have closen to perform our experiments on two parallel machines - the Sequent Symmetry and the BBN Butterfly-with entirely different architectures. The Sequent is a bus based shared memory multiprocessor ma-


[^0]:    ${ }^{1}$ This material is based upon work supported by the National Science Foundation under grant CCR-8711749.

[^1]:    ${ }^{1}$ Part of this work was supported by the National Science Foundation under grant CCR-8711749.

[^2]:    - Supported in part by SFB 303.
    'Supported in part by NSF grant CCR-8711749.
    'Supported in part by NSF grant CCR-8813283.

[^3]:    ${ }^{1}$ This material is based upon work supported by the National Science Foundation under grant CCR-8711749. A preliminary version of this paper was presented at the eighth annual conference on Foundations of Software Technology and Theoretical Computer Science held at Pune, India during 21-23 December 1988. A preliminary version also appeared as a Georgia Institute of Technology Technical Report GIT-ICS-88-09.

[^4]:    *This paper was presented at the Fifth Annual Conference on Structure in Complexity Theory held in Barcelona, Spain, 1990.
    ${ }^{\dagger}$ Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India.
    ${ }^{\prime}$ College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332-0280. Most of this work was done while this author was at the Indian Institute of Science, Bangalore, India, on leave from the Georgia Institute of Technology. This work has been funded in part NSF Grant CCR-8711749.
    ${ }^{\S}$ Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India.

[^5]:    - This work has been funded in part by NSF grants CCR8711749, CCR-8619886, and MIPS-8809268.

