Counting Auxiliary Pushdown Automata and Semi-Unbounded Arithmetic Circuits

V Vinay
Department of Computer Science and Automation
Indian Institute of Science
Bangalore-560 012
India
e-mail :vigyan!csa!vinay@shakti.ernet.in

April 22, 1991

1 Introduction

We examine various counting measures on *space bounded nondeterministic auxiliary pushdown machines.* Hitherto, counting measures on nondeterministic time bounded [Val 79,KST 89] and space bounded **[AJ** 90] machines have been studied.

In the main theorem, we show how a NAuxPDA may be simulated efficiently by a uniform family of boolean circuits, which preserve the number of accepting paths in the NAuxPDA as the number of accepting subtrees in the boolean circuit. Our techniques simulate the NAuxPDA in novel way by considering the height and reversal bounds of an AuxPDA. Reversal bounded AuxPDA have been studied previously [BH 88].

One of the highlights of this paper is an exact characterization of the important class, $\mathcal{DET}$. We show that $\mathcal{DET}$ is exactly class of functions that can be computed as the difference between the outputs of two counting logspace machines! The proof is easy but inexplicably has gone unnoticed in the complexity theory.

The main theorem have several applications in proving known results in a simple and direct way. We list some implications of our results.

- Venkateswaran [Ve 87].

  We characterize $\sharp$AuxPDA SPACE,TIME$(\log n, poly(n))$ in terms of certain arithmetic circuits. The result strengthens the semi-unbounded fan-in circuit characterization of $\mathcal{LOGCFL}$ due to Venkateswaran [Ve 87].

- Venkateswaran [Ve 88].

  Venkateswaran [Ve 88] gave a characterization of NP using semi-unbounded Boolean circuits. The arithmetic circuits corresponding to the circuit characterization of NP lead to an alternative characterization of $\#P$. This characterization was recently discovered independently by Babai and Fortnow [BF 90] in a different form. In the light of the similarity between $\mathcal{LOGCFL}$ and NP [JK 88,VC 90, VVV 90], our results may also be regarded as a polynomial analogue of certain straight line programs which capture $\sharp\mathcal{P}$ [Ve 88].

- Valiant, et al [VSBR 83].

  Valiant et al [VSBR 83] showed how arithmetic circuits of polynomial size and degree

CH3022-1/91/0000/0270/$01.00 © 1991 IEEE

d over $\{+, \star, -\}$ can be reduced to semi-unbounded arithmetic circuits of polynomial size, and depth $O(\log n + \log d)$. The resulting circuits are **P**- uniform. We give a structural proof of this result. In the interesting case of polynomial degree, we demonstrate $\mathcal{DLOG}$-uniform $\log n$ depth circuits.

- Alvarez and Jenner [AJ 90].

  They studied counting space classes. They prove that the counting and optimizing versions of $\mathcal{NLOG}$, $\sharp\mathcal{LOG}$ and $Opt\mathcal{LOG}$, are in $\mathcal{NC}^2$. We show counting versions and optimizing versions of $\mathcal{LOGCFL}$, $\sharp SAC^1$ and $OptSAC^1$, are in $\mathcal{NC}^2$. This is an improvement over their result as $\mathcal{NLOG}$ is contained in $\mathcal{LOGCFL}$. Moreover, our proof requires different techniques from theirs, as $\mathcal{LOGCFL}$ is not known to be in $\mathcal{DET}$. Also, many of their other results generalize naturally to the AuxPDA classes. Among them, we show $SpanSAC^1$ is Turing hard for $\mathcal{PH}$.

- Ladner [Lad 89].

  He introduced the notion of "natural" $\mathcal{PSPACE}$ counting class: $\natural\mathcal{PSPACE}$. It is shown there that this class corresponds to $\mathcal{FPSPACE}(poly)$. We explore the counting versions of the class **P** as NAuxPDA SPACE,TIME$(\log n, exp(n))$ [Co 71]; we shall refer to this class by #SAC". Our investigation into its "natural" counter-part, $\natural SAC^\infty$, shows that this class coincides with **3P!**

- Huynh [Huy 90].

  This paper proves ranking languages in 1UAuxPDA SPACE,TIME$(\log n, poly(n), i)$s in $\mathcal{NC}^2$. We prove this ranking function is complete for #SAC', which immediately shows that the rank function is in $\mathcal{NC}^2$. We also prove that ranking languages in 1UAuxPDA SPACE,TIME$(\log n, 2^{poly(n)}, poly(n))$ is in **3P**— making it the "largest)) known class to be P-rankable.

- Krentel [Kre 88].

  Krentel introduced the notion of OptP and studied it in great detail. We show a natural circuit characterization for the classes OptP , $OptSAC^1$ and $OptSAC^\infty$.

These apart, we introduce the notion of **P**-optimizable sets as a generalization of P-printable sets. We prove **P** is P-optimizable if and only if $\mathcal{P}=$ OptP. We show that context free languages are **P**- optimizable whereas it is unlikely that they to be P-rankable.

2 Definitions and Notations

We assume that the reader is familiar with ATMs and the definitions of the standard complexity classes such as $\mathcal{NC}^1$, $\mathcal{DLOG}$, $\mathcal{NLOG}$, $\mathcal{LOGCFL}$, $\mathcal{NC}^2$, **P** etc.

2.1 Nondeterministic Auxiliary Pushdown Automata

By an AuxPDA we mean an nondeterministic Turing machine with an additional pushdown store. The space used by the machine is corresponds to the space on the worktape only. For a more formal definition the reader is referred to [Co 71,Ru 80].

Surface Configuration: By a *surface* configu***ration,*** v, of an AuxPDA machine M on input **x,** we mean v = (q,i,α,j,z) where q is the current state of M, i is the input head position, a is the worktape contents, j is the worktape head position and z is the top of stack symbol. A surface configuration has information only about the stack top rather than the whole pushdown.
We shall often say Configuration to mean surface configuration when there is no ambiguity.

Acceptance: We make the following assumptions about the AuxPDA machine, M.

- M accepts on a unique accepting configuration, on an empty pushdown store. We assume there is a unique bottom stack marker that the machine pops to accept.
- M pushes or pops in units of $S(n)$, where $S(n)$ is the space bound of the machine M.
- M pushes or pops at every move.

Realizable **Pairs:** A pair (P,Q) is said to be *realizable pair* if

- there is a computation of M which when started on P leads to Q.
- The pushdown height at **P** and Q are identical.
- The pushdown height in any of the intermediate step never goes below the pushdown height at **P.**

Profile: A *profile* of a computation sequence, is a graph depicting the behavior of the pushdown height over time for that computation sequence.

Valley Configuration: A configuration in a profile is called a *valley configumtion* if its previous step was a pop and the next step is a push.

Slice: Suppose(P,Q) is a realizable pair. Fix a computation sequence that witnesses the realizable pair. The *slice* of (P,Q) with respect to the computation sequence is the number of configurations, Z, inclusive of Q, along the computation sequence such that (P,Z) is realizable. We shall denote this quantity by $sl(P,\mathrm{Q})$ where the computation sequence should be clear from the context.

2.2 Semi-unbounded Alternating Turing Machine

By a *Semi-Unbounded Alternating Turing Machine* we mean a ATM wherein there exists an accepting subtree with at most a constant number of universal configurations between any successive pair of existential configurations.

2.3 Circuits

We will assume the reader is familiar with the basic *Boolean Circuit* model where either all gates have bounded fan-in or unbounded fan-in. Wlog assume that the gates are of two types, {AND ,OR }.

Semi-unbounded Fan-in Circuits: A circuit family, $\{C_n\}$, is called *semi-unbounded* if, for any member of the family, the OR gates of the circuit have unbounded fan-in and the AND gates have bounded fan-in.

Arithmetic Circuits: By an *arithmetic circuit* we usually mean a circuit where the OR gates and AND gates are interpreted over some suitable (semi-) ring. The interpretations we use in this paper are **(1)** PLUS , MULT (2) MAX , CONCAT and **(3)** $\cup$, CONCAT over appropriate (semi-) rings. The notion of semi-unboundedness can be extended to a natural way to arithmetic circuits with MULT and CONCAT taking the role of AND .

*In general, we use circuits to mean either Boolean **or** Arithmetic circuits.*

All of these circuit family needs *uniformity* conditions. We use logspace uniform circuits. In only one simulation ($OptP$in terms of arithmetic circuits) do we need P-uniform circuits.

2.4 The New Classes:

We are now ready to define the new classes. For more details see **[AJ** 90]. We shall define the counting classes as operators. *All these operators may be applied on any nondeterministic (resource bounded) machines*

Counting Operator, $\sharp$: By $\sharp$ of a nondeterministic machine, we mean the function which takes the input string, x, to a natural number denoting the number of accepting paths of the machine on input x.

Opt: By **Opt** of a nondeterministic transducer, we mean the function which takes the input string, x, to another string denoting the lex-maximum

string output by the transducer over all accepting paths of the transducer on input x.

Span: By Span of a nondeterministic transducer, we mean the function which takes the input string, x, to a natural number denoting the number of distinct strings output by the transducer over all accepting paths of the transducer on input x.

Of course, any of the operator applied on a class of machines is a collection of all functions gotten by applying the operator on each machine of the class.

The counting operator may also be defined on boolean circuits. Here the function will return the number of accepting subtrees in the circuit. It is well-known that the number of such accepting subtrees is the output of the corresponding arithmetic circuit where OR gate and AND gates are replaced by PLUS gates and MULT gates respectively. We shall use this duality relationship to prove our results. We also exploit this correspondence to use a single notation; $\sharp$ on boolean circuits to correspond to the corresponding arithmetic circuit (over 0, 1 inputs).

2.5 Notation

By NAuxPDA SPACE,TIME$(S(n),T(n))$, we mean set of languages accepted by a AuxPDA that runs within space $S(n)$ and time $T(n)$.

By $\sharp$AuxPDA SPACE,TIME$(S(n),T(n))$, we mean set of functions computed by applying the $\sharp$ operator on an AuxPDA that runs within space $S(n)$ and time $T(n)$. The classes OptAuxPDA SPACE,TIME$(*,*)$ and SpanAuxPDA SPACE,TIME$(*,*)$ is similarly defined.

By $\sharp$Semi-Unbounded ASPACE,ALT$(S(n),A(n))$, we mean the set of functions computed by applying the #operator on a semi-unbounded ATM within space $S(n)$ and alternation depth $A(n)$.

By Semi-Unbounded USIZE,DEPTH$(S(n),D(n))$, we mean set of languages accepted by uniform families of semi-unbounded circuit that have size $S(n)$ and depth $D(n)$.

By $\sharp$Semi-Unbounded USIZE,ALT$(S(n),A(n))$, we mean set of functions output by computed by uniform family of semi-unbounded circuits of size $S(n)$ and alternation $D(n)$. The context will make the interpretation **of** the gate operators clear.

By $\sharp$USIZE,DEPTH$(S(n),D(n))$, we mean set of functions output by computed by uniform family of bounded circuits of size $S(n)$ and depth $D(n)$. The context will make the interpretation of the gate operators clear.

By $\sharp$Semi-Unbounded USIZE,DEPTH$(S(n),D(n))$, we mean set of functions output by computed by uniform family of semi-unbounded circuits of size $S(n)$ and depth $D(n)$. The context will make the interpretation of the gate operators clear.

Finally, additional parameter of the form $\{b(n)\}$ conveys that the number of output bits of the transducer is limited to $O(b(n))$ and parameter of the form $[b(n)]$ conveys that the number of accepting paths of the transducer is bounded by $2^{O(b(n))}$.

2.6 Abbreviations

We shall use the following interchangeably.

$\sharp SAC^k$ = $\sharp$AuxPDA SPACE,TIME$(\log n, 2^{O(\log^k n)})$.

$\sharp SAC^\infty$ = $\sharp$AuxPDA SPACE,TIME$(\log n, 2^{poly(n)})$.

Similarly for Opt and Span operators.

3 The Main Theorem

Ruzzo [Ru 80] was the first to show the close correspondence between NAuxPDA time and alternating treesize. In the same paper , using what now is known to be a standard pebbling argument on trees, he shows the relationship between alternating treesize and alternations. Venkateswaran [Ve 87] noticed that the alternations were actually semi-unbounded alternations. This was used as a

first step by Venkateswaran in showing the semi-unbounded characterization of $\mathcal{LOGCFL}$. Unfortunately, Ruzzo's simulation does not preserve the number of proofs. This is due to the fact that the pebbling argument may not result in a unique decomposition of an accepting subtree.

To design a simulation that preserves proofs, we need to decompose a proof in a unique way. We show how to do this in a simple way. In fact, we give a simulation which takes us directly from an AuxPDA to Semi-unbounded alternating Turing machine. As we shall see, the notion of a slice plays an important role in the simulation. The other containments are similar to [Ve87], with minor modifications.

We start with a few simple but important lemmas. We assume that the AuxPDA halts on all paths, either accepting or rejecting.

Lemma 3.1 *For* $T(n) \geq 2^{O(S(n))}$, $\sharp$AuxPDA SPACE,TIME$(S(n),T(n)) \subseteq$ $\sharp$AuxPDA SPACE,HT$(S(n),\log T(n))$ ∎

Lemma 3.2 *For an* NAuxPDA *running in space* $S(n)$, *any slice is bounded by* $2^{O(S(n))}$.

Proof: Otherwise the NAuxPDA would have a non-terminating path as some two configurations have to repeat. ∎

Lemma 3.3 $\sharp\mathcal{LOG} \subseteq \sharp SAC^1$.

Proof: It is actually sufficient to know an upper bound on the length of the path. It is then not difficult to modify the familiar divide and conquer argument to prove the inclusion. ∎

Lemma 3.4 *Consider a* PLUS *circuit with inputs* $x_1, x_2, \ldots, x_n$ *and output* o. *Then the output of the circuit,* $\sharp o$, *is* $\sum_{i=1}^{i=n} c_i \star x_i$, *where* c_i *is the number of paths from the input* x_i *to the output* o. *When the* PLUS *gates are replaced by* OR *gates,* $\sharp o$ *denotes the number of accepting paths in the circuit.* ∎

Theorem 3.5
$\sharp$AuxPDA SPACE,TIME, HT$(S(n),T(n),\log T(n)) \subseteq$
$\sharp$Semi-Unbounded ASPACE,ALT$(S(n), S(n)*\log T(n)) \subseteq$
$\sharp$Semi-Unbounded USIZE,ALT$(2^{O(S(n))}, S(n)*\log T(n)) \subseteq$
$\sharp$Semi-Unbounded USIZE,DEPTH$(2^{O(S(n))}, S(n)*\log T(n) \subseteq \sharp$AuxPDA SPACE,TIME, HT$(S(n), T(n)^{O(S(n))}, S(n)*\log T(n))$. *Furthermore, the* $S(n)$ *factor in* $S(n)*\log T(n)$ *may be removed when* $T(n) = 2^{O(S(n))}$ *(and hence all inclusions become equality).*

Proof:
(1) $\sharp$AuxPDA SPACE,TIME, HT$(S(n),T(n),\log T(n)) \subseteq$
$\sharp$Semi-Unbounded ASPACE,ALT$(S(n), S(n)*\log T(n))$.

Clearly, it is adequate to show $\sharp$AuxPDA SPACE,HT$(S(n),\log T(n)) \subseteq$ Semi-unbounded ASPACE,ALT$(S(n), S(n)*\log T(n))$. Fix an accepting path. Wlog assume the AuxPDA M, *pushes* **or** *pops* at every move. We will show how an ATM can verify this path uniquely. From lemma **3.2,** it is clear that any slice of the profile of this accepting path may be represented within $O(S(n))$ bits.

Initially, the ATM starts with the realizable pair (C_{in}, C_{acc}). At some intermediate step, let the pair be (P,Q). If P and Q are identical, the ATM accepts. Otherwise, it does one of two things:

1. existentially guesses a pair (P',Q') and a string b of $S(n)$ bits. It verifies $P \xrightarrow{push(b)} P'$ and Q' $\xrightarrow{pop(b)} Q$. It continues with (P',Q') if verification holds. Otherwise it rejects.
2. existentially guesses the slice, $sl(P,Q)$, between P and Q. It then guesses a configuration Z such that $sl(P,Z) = \lfloor sl(P,Q)/2 \rfloor$. The computation universally branches to $(P,Z,sl(P,Z))$ and $(Z,Q,sl(Z,Q))$. This halving process is repeated until there are no valley configurations (slice takes the value one) between any pair. In order to verify the

claim that there are no valley configurations, (1)is forced in the next iteration.

That the above simulation results in a unique decomposition of the accepting path and the simulation is semi-unbounded is easy to see.

Analysis: From lemma 3.2, any slice is bounded by $2^{O(S(n))}$. Note that (1)decreases the height of the pushdown by one, whereas **(2)** decreases the height of the pushdown by one within $O(S(n))$ alternations. On the whole, the pushdown height comes down by one within every $O(S(n))$ semi-unbounded alternations. The result follows as the height of the stack is bounded by $\log T(n)$.

We now show how to modify the construction in the case **of** $S(n)$ space and $2^{O(S(n))}$ time NAuxPDA's. As the number of reversals cannot exceed the time bound, the ATM can guess the number of reversals in a profile. With any realizable pair (P,Q) we also maintain the exact number of reversals between them; instead of pairs, we store triples, (P,Q,rev). When P = Q, the ATM accepts if **rev** = 0 but rejects if $rev > 0$. In choice (1)above, the number of reversals remain the same but the pushdown height comes down by one. We modify choice **(2)** to the following:

- guess consecutive configurations, Z_1, Z_2 and numbers r_1, r_2, r_3 such that $(r_1, r_3 < rev/2), r_1 + r_2 \geq rev/2, r_2 + r_3 \geq rev/2$ and $r_1 + r_2 + r_3 =$ **rev** $-$ **2** (make appropriate changes in boundary conditions when $Z_1 = P$ or Z_2 = Q). The ATM then guesses Z_1', Z_2', b and universally verifies $(P, Z_1, r_1), (Z_3, Q, r_3)$ and (Z_1', Z_2') where $Z_1 \xrightarrow{push(b)} Z_1'$ and $Z_2' \xrightarrow{pop(b)} Z_2$. The additional guessing is necessary to make sure that there are no valley configurations between Z_1 and Z_2.

We claim that such a Z_1 and Z_2 are unique. To see this, note that if $r_2 \geq rev/2$, Z_1 and Z_2 are unique. Otherwise, imagine a vertical line cutting the number of reversals between P and Q into two equal halves. Then Z_1 (resp. Z_2) is the nearest valley configuration to the left (resp. right) of the imaginary line away from P (resp. Q).

Analysis: Consider the potential function $\Psi = \log(rev) + h$, where ***h*** is the height of the pushdown needed to prove (P,Q) is a realizable pair. When (1) is used, the height of the pushdown comes down by one. When the modified form of **(2)** is executed, either the number of reversals come down by half (as in the case of (P, Z_1) and (Z_2, Q)) or the pushdown height comes down by one (as in the case of (Z_1, Z_2)). Either way Ψ decreases by one. As the height **of** the pushdown is $O(\log n)$ and the number of reversals is $poly(n)$, the result follows.

(2)

usemi-Unbounded ASPACE,ALT$(S(n), S(n) * \log T(n))$

$\subseteq$

usemi-Unbounded USIZE,ALT$(2^{O(S(n))}, S(n) * \log T(n))$.

The inclusion is immediate from standard techniques [Ru 81,Ve 87].

(3)

usemi-Unbounded USIZE,ALT$(2^{O(S(n))}, S(n) * \log T(n))$

$\subseteq$

#Semi-Unbounded USIZE,DEPTH$(2^{O(S(n))}, S(n) * \log T(n$

This result follows from the techniques **of** [Ve 87] in conjunction with lemmata 3.3 and 3.4. Let $\{C_n\}$ be a uniform circuit family meeting the above resource constraints. We will show how to construct a uniform family $\{G_n\}$ of semi-unbounded fan-in circuits, such that $\{C_n\}$ and {G,} have the same number of accepting subtrees on any input. The key idea is to make all inputs to an OR gate be non-OR gates. **As** the **AND** gates are semi-unbounded, they may be retained without any changes. Fix a parameter $d(n) = \lceil cS(n) \rceil$, where the size of the n-th circuit of $\{C_n\}$ is $2^{cS(n)}$ for some constant **c.**

We use the following notation. By g, g_1, g_2, h we refer to gates in C_n. These shall figure as $[g], [g_1], [g_2], [h]$in G, with the ***same gate type.*** Let the output gate of G, be $[o]$ where ***o*** is the output

gate of C_n. Given a gate $[g]$, the inputs to the gate are defined as follows.

- $[g]$ is an AND gate. Let the inputs to g be g_1 and g_2. Then the inputs to $[g]$ are $[g_1]$ and $[g_2]$.
- $[g]$ is an input gate. Then there is nothing to be done except remembering the input bit to be wired.
- $[g]$ is an OR gate. The inputs to the gate $[g]$ are all gates of the form $[g,h]$, where h is a non-OR gate. A $[g,h]$ gate is an AND gate with two inputs $[0,g,h]$ and $[h]$. The inputs to the gate $[h]$ are constructed recursively. The gate $[0,g,h]$ is the output gate of a reachability sub-circuit that verifies that there is indeed a path from g to h in C_n *using only* OR *gates,* This may be checked within depth $d(n)$ by using lemma **3.3.** This construction with lemma **3.4,** guarantees that the number of accepting paths at $[g]$ is identical to the number of accepting paths at g.

The construction is clearly logspace uniform. The depth of the resulting circuit is $O(S(n) * \log T(n))$. This proves the inclusion.

(4)
$\sharp$Semi-Unbounded USIZE,DEPTH$(2^{O(S(n))}, S(n) * \log T(n))$
$\subseteq$
$\sharp$AuxPDA SPACE,TIME, HT$(S(n), T(n)^{S(n)}, S(n) * \log T(n))$.

Use nondeterminism at an OR gate and always evaluate the left child of an AND gate first by pushing the right child of the AND gate onto the pushdown. This ensures that an accepting tree is mapped to a unique accepting path in the AuxPDA computation. ∎

4 Structural Results

We begin by noting some implications of the main theorem. The theorem tells us exactly how the results of Venkateswaran [Ve 87] have been strengthened.

Theorem 4.1 $\sharp SAC^1$ = $\sharp$Semi-Unbounded USIZE,DEPTH$(poly(n), \log n)$ ∎

The proof of the main theorem yields techniques that characterize $OptSAC^1$ and $Opt\mathcal{P}$.

Theorem 4.2 *(a)* $OptSAC'$ = $\sharp$Semi-Unbounded USIZE,DEPTH$(poly(n), \log n)$, *(b)* $opt\mathcal{P}$ = $\sharp$Semi-Unbounded USIZE,DEPTH$(2^{n^{O(1)}}, \log n)$ *where* OR *is interpreted as* MAX *and* AND *is interpreted as* CONCAT.

Proof: The proof is similar to that of the main theorem. We have to be careful about the order of the inputs to a gate. ∎

We now show how ***SpanSAC'*** may be computed with circuits.

Theorem 4.3 $SpanSAC^1 \subseteq$ $\sharp$Semi-Unbounded USIZE,DEPTH$(poly(n), \log n)$, *where* OR *is interpreted as* $\cup$ *and* AND *is interpreted* as CONCAT.

Proof: The concatenation operator is taken over sets. Actually, it is the cardinality of the output that is the span. ∎

We now prove, similar to [AJ 90] , that ***SpanSAC'*** is Turing-equivalent to $\sharp\mathcal{P}$.

Lemma 4.4 $SpanSAC' \subseteq \sharp\mathcal{P}$.

Proof: Let f be a function in ***SpanSAC'***. Let the function be witnessed by an NAuxPDA transducer N. Consider the following language L.

$L = \{x\#y \mid$ y is an output of N on input $x\}$.

Clearly, this language is in $\mathcal{LOGCFL}$ and therefore in $\mathcal{P}$. As the length of y is polynomially bounded, an NP machine guesses all y's bounded by that length and verifies the guess using the $\mathcal{P}$ machine. ∎

The following result is known

Lemma 4.5 ([AJ 90]) $\natural\mathcal{P} \subseteq P(Span\mathcal{LOG})$. ∎

We need the following result due to Toda. For a simpler proof see [RVVY 90,BF 90,Tr 90,RR 90].

Theorem 4.6 ([To 89]) $\mathcal{PH} \subseteq BP.\oplus\mathcal{P} \subseteq \mathcal{P}(\natural\mathcal{P})$.

From the observation that $Span\mathcal{LOG}$ is contained in $SpanSAC^1$, we get

Theorem4.7 $\mathcal{PH} \subseteq \mathcal{P}(Span\mathcal{LOG}) = \mathcal{P}(SpanSAC^1) = \mathcal{P}(Span\mathcal{P}) = \mathcal{P}(\natural\mathcal{P}) = \mathcal{P}(PP)$.

Proof: Follows from standard results connecting $\mathcal{PP}$ and $\natural\mathcal{P}$, lemmata 4.4 and4.5, and Toda's theorem . I

Though $SpanSAC^1$ is very hard, we nevertheless can show a restriction of the class is in $\mathcal{NC}^2$.

Corollary 4.8 1. $\natural SAC^1 \subseteq \mathcal{NC}^2$

2. OptSAC' $\subseteq \mathcal{NC}^2$.

3. $SpanSAC^1[\log n] \subseteq \mathcal{NC}^2$.

Proof: Replace all "+" and "$\star$" gates by appropriate $\mathcal{NC}^1$circuits. I

Apparently, it is unlikely that $\natural SAC^c$ontains $\natural SAC^1$(similar results hold for Opt also) as the following lemma demonstrates.

Lemma4.9 $\natural SAC^1 = \natural\mathcal{LOG}$ *iff* $\mathcal{NLOG} = \mathcal{LOGCFL}$. I

Can OptSAC' be contained in $\natural\mathcal{LOG}$? Again, it seems unlikely.

Lemma 4.10 *If* OptSAC' $\subseteq \natural\mathcal{LOG}$ then $\mathcal{ULOG} = \mathcal{LOGCFL} = \mathcal{NLOG}$.

Proof: The characteristic function of any $\mathcal{LOGCFL}$ language is in OptSAC'. By hypothesis, it is in $\natural\mathcal{LOG}$. Therefore, this fact has to be witnessed by a $\mathcal{ULOG}$ machine. I

At the other extreme, $\natural SAC^\infty$,$OptSAC^\infty$,$SpanSAC^\infty$have huge outputs whose length may exceed polynomial space. It is therefore reasonable to restrict these machines to polynomial length outputs — with the hope that such a restriction is fruitful.

We look at a "natural" restriction first considered by Ladner [Lad 89]. Nondeterminism along any path is restricted to a polynomial number of moves. This naturally forces the number of paths in the computation tree of the NAuxPDA to be bounded by an exponential in the input length. We will denote such a class by $\natural SAC^\infty$. Surprisingly, this class coincides with $3P$. Again, this result is made possible by the main theorem. That computing functions in $3P$ are equivalent to counting the number of accepting subtrees in certain boolean circuits demonstrates the richness of the counting paradigm.

Theorem 4.11 $\mathcal{FP} = \natural SAC^\infty =$ $\natural$Semi-Unbounded USIZE$(poly(n))[poly(n)]$.

Proof: (1)$\mathcal{FP} \subseteq \natural SAC^\infty$. The proof is similar to that in [Lad 89]. Consider a polynomial time computable function, f. The language defined below is then in $\mathcal{P}$.
$L = \{(x,i)$: the i^{th} bit of $f(x)$ is a 1**)**. By virtue of being in P, there is a DAuxPDA accepting it in space log n and $n^{O(1)}$ pushdown height. Let m be a polynomial bound on the length of f's output.

Start with the least significant bit . In general, if the i^{th} bit is a "1" then do one of three things.

1. accept
2. check if $(x, i+1) \in L$.
3. check if $(x, i+1) \in L$.

1

If the i^{th} bit is a "0" then do one of *(2)* **or** *(3)* above.

(2) $\sharp SAC^{\infty} \subseteq$ $\sharp$Semi-Unbounded USIZE($poly(n)$)[$poly$(n)].

It follows from the main theorem and the fact that the number accepting paths may be represented using polynomial number of bits.

(3) $\sharp$Semi-Unbounded USIZE($poly(n)$)[$poly(n)$] $\subseteq$ ***3 P.***

As the number of output bits of the arithmetic circuit is polynomially bounded, the PLUS and MULT gates may be replaced by $\mathcal{NC}^1$ circuits. **As** polynomial size circuits with polynomially many outputs is precisely $\mathcal{FP}$, the result follows. ∎

5 Arithmetic Circuits

In this section, we show how our main result can be used to convert small degree arithmetic circuits to small depth arithmetic circuits.

Valiant et al [VSBR 83] showed how circuits with small algebraic degree may be converted into circuits of small depth, without substantial increase in the size of the circuits. Their result may be restated in the following form. We state the results **for** polynomial size circuits for simplicity.

Theorem 5.1 *Let* **{C,}** *be a uniform family of arithmetic circuits of polynomial size and degree d. Then there is an equivalent P-uniform family of semi-unbounded arithmetic circuits of polynomial sire* and $O(\log n + \log d)$ ***depth.*** I

We prove a slightly weaker version of the above theore with improvements in certain cases. The proof of Valiant et al is complicated, whereas our proof is reasonably straight forward. We show how negative-numbers can be handled by AuxPDA. This idea turns out to be crucial in characterizing $\mathcal{DET}$ (see next section).

Theorem 5.2 *Let* **{C,}** *be an uniform family* of *arithmetic circuits* of *polynomial size and degree* d. *Then there is an equivalent P- uniform family of semi-unbounded arithmetic circuits of polynomial size and* $O(\log n(\log n + \log d))$ ***depth.*** *Furthermore, when the degree* of *the arithmetic circuit is polynomial, the equivalent semi-unbounded arithmetic circuit has* $O(\log n)$ ***depth and is*** $\mathcal{DLOG}$***-uniform.***

Proof: Without loss of generality, we assume that the input to the circuit is one of $0, +1$ or -1. From lemma *5.3* below, it is clear that there is a non-uniform special AuxPDA, N, with three final states $\{0, +1, -1\}$ such that the output of the circuit, say f, is identical with the difference between the number of paths leading to a $+1$ and the number of paths leading to a -1. It is a simple matter now to construct ***two*** NAuxPDA's, N_{+1} *and* N_{-1}, so that $N_{+1}(resp. N_{-1})$ accept iff the path leads to a $+1(resp. -1)$. Letting $\sharp(N_i, x)$ denote the number of accepting paths in the machine N_i, we see that $f(x) = \sharp(N_{+1}, x) - \sharp(N_{-1}, x)$. By lemma **5.4,** the running time of each of these machines is bounded by logn + logd. By combining the main theorem, and the fact that the non-uniform advice to the special AuxPDAcan be computed in ***P,*** each of these machines may be faithfully simulated by a ***P*** uniform (instead of a $\mathcal{DLOG}$- uniformity) family of arithmetic circuits of depth $O(\log n(\log n + \log d))$. The difference between the outputs of these circuits give the required answer. With a bit more effort, the output gate may be changed from a SUB gate to a PLUS gate by forcing the second machine to output a negative number without affecting the resources. The improvement for polynomial degree also follows similarly from the main theorem and *3.1.* ∎

When the algebraic degree of the polynomial is a polynomial in the length of the input, it is clear that our simulation is an improvement over theirs.

To complete the proof, we need the two lemmas below.

By a ***special*** $\sharp$AuxPDA we mean one with three

final states $\{0, +1, -1\}$. The output of such a machine is the difference between the number of paths with final state as $+1$ and the number of paths with final state as -1.

Lemma 5.3 *Let $\{C_n\}$ be a uniform family of bounded arithmetic circuits. Then there is a non-uniform special #AuxPDA such that on any input, the output of the arithmetic circuit is identical to to output of the special #AuxPDA .*

Proof: The proof is by induction on the depth of the circuit. There is nothing to prove at the input level . Assume the result holds for all circuits of depth $D-1$. We will show that it holds for circuits of depth D as well.

Consider a depth D arithmetic circuit. The advice on the tape is the algebraic degree of each gate. Let its output gate be *out*. Let its inputs be inp_1 and inp_2. By hypothesis, there exists two special #AuxPDA N_1, N_2. Let the number of paths leading to a $+1(-1)$ in N_i be $x_i(y_i)$. Then the value output by the gate inp_i is $val(inp_i) = x_i - y_i$, by induction hypothesis.

- *out* is an PLUS gate. Then the AuxPDA existentially simulates either N_1 or N_2. Then $val(out) = (x_1 - y_1) + (x_2 - y_2)$.
- *out* is a MULT gate. In this case the output, $val(out)$ should be $(x_1 - y_1) * (x_2 - y_2)$. The AuxPDA simulates the circuit in the following way. From the advice tape, it know the degree of the inputs. Wlog assume inp_2 is the heavier input. It pushes inp_2 onto the pushdown and simulates N_1. When it completes simulating N_1, the AuxPDA is in some path with a value form $\{0, +1, -1\}$ and with inp_2 on top of stack. Call the value of that path p. Pop top of stack and if $p \neq 0$ then simulate N_2. The simulation of N_2 terminates in some final state v. The AuxPDA enters the final state $p * v$. So all that has been done is to look at the parity of the final states of the two accepting paths. Now the number of paths in AuxPDA that are in state $+1(resp. -1)$ are $x_1 * x_2 + y_1 * y_2 (resp. x_1 * y_2 + x_2 * y_1)$. So the output of the AuxPDA is $x_1 * x_2 + y_1 * y_2 - x_1 * y_2 - x_2 * y_1$, which completes the induction.

This completes the proof. **I**

Lemma 5.4 *Let $\{C_n\}$ be a bounded arithmetic circuit of depth D and degree d. Then the running time T, of an AuxPDA simulating the circuit as in the above lemma is bounded by $D * d + 1$. Also, the pushdown height is bounded by $O(\log D + \log d)$*

Proof: The proof is by induction on depth. A similar lemma is proved in [Ve 88] to bound tree-size of boolean circuits. At the input level, the result is trivially true. So assume the lemma holds for all circuits of depth $D - 1$. Let T_1, T_2 be the running time of the two children inp_1, inp_2 respectively. Also, let their respective algebraic degrees be d_1, d_2.

- *out* is an PLUS gate. Then ,
  $T \leq max\{T_1, T_2\} + 1$.
  $\leq d * (D - 1) + 1$
  $\leq D * d + 1$.
- *out* is an MULT gate. Then,
  $T \leq T_1 + T_2 + 1$
  $\leq (d_1 + d_2) * (D - 1) + 3$
  $\leq d * (D - 1) + 3$
  $\leq D * d + 1$.

The induction is complete.

To complete the proof, note that only inputs to MULT gates are pushed onto the pushdown. But machine always explores the lighter child first. ∎

6 $\mathcal{DET}$ is as easy as counting paths in a DAG!

Consider the problem of matrix powering **(raising** an order n matrix to its n^{th} power, where

all interers are n bits in length). This problem is known to be complete for $\mathcal{DET}$ [Co 85]. Since matrix multiplication can be computed by semi-unbounded depth two arithmetic circuits over interegs, it is clear that matrix powering has polynomial size and $O(\log n)$ depth semi-unbounded arithmetic circuits. From lemma **5.3** we have

Theorem 6.1 *The determinant of an order n matrix over integer entries of length at most n can be computed as the difference of two $\sharp SAC^1$ functions.* I

Note that this result is not implied by theorem **5.1** due to the uniformity condition.

It is surprising that this result can be vastly improved– in fact to give an exact characterization of $\mathcal{DET}$! This result has also been independently observed by Damm [Damm **91**] via different techniques.

We will show how to do matrix powering using a *special* $\mathcal{NLOG}$ machine with three final states $\{0, +1, -1\}$. The output **of** such a machine is the difference between the number of paths with final state as $+1$ and the number of paths with final state as -1.

Lemma 6.2 *MATPOW can be simulated in such a way that the ijth entry of $A^m, m \leq n$ is the output of a special $\mathcal{NLOG}$ machine.*

Proof: The proof is by induction on the index m. Clearly there is nothing to prove when $m = 1$. So assume the result is true for $m - 1$. Noting $A^{m} = A \star A^{m-1}$, we design a special $\mathcal{NLOG}$ machine with the required properties for entry a_{ij}^m.

- existentially guess a $\mathbf{k,l} \leq \mathbf{k} \leq n$. This requires only $\log n$ bits.
- **Look** at a_{ik}. If
  - $a_{ik} = 0$, the machine halts in the final state 0.
  - $a_{ik} \geq 0$, the machine branches into $-a_{ik}$ paths and along any such path simulates the special $\mathcal{NLOG}$ machine **for** entry a_{kj}^{m-1} guaranteed to exist by induction hypothesis.
  - $a_{ik} \leq 0$, the machine branches into a_{ik} paths and simulates the special $\mathcal{NLOG}$ machine for entry a_{kj}^{m-1} guaranteed to exist by induction hypothesis. When any of the paths reach a final state, the final state is toggled from **1** to -1 and vice-versa.

Branching into strictly $|a_{ik}|$ paths and then simulating a_{kj}^{m-1} has the effect of multiplying them. The nondeterministic machine is easily seen to use $O(\log n)$ space. The induction is complete. ∎

A special $\mathcal{NLOG}$ machine may be thought of as outputting the difference of two $\sharp\mathcal{LOG}$ machine in the following manner; one of the machine accepts along a path iff it reaches a $+1$ final state, whereas the other machine accepts along a path iff it reaches a -1 state. Combining this observation with the previous lemma, we get

Theorem 6.3 *1. $\mathcal{DET} \subseteq \mathcal{DIFF}(\sharp\mathcal{LOG})$.*

2. *MATPOW over non-negative integers is complete for $\sharp\mathcal{LOG}$.* I

The following result is proved in [BDLM **91**].

Theorem 6.4 $\mathcal{LOG}^{\sharp\mathcal{LOG}} \subseteq \mathcal{DET}$. I

Putting them together, we get an exact characterization of $\mathcal{DET}$.

Theorem 6.5 $\mathcal{DET} = \mathcal{LOG}^{\sharp\mathcal{LOG}} = \mathcal{DIFF}(\sharp\mathcal{LOG})$. ∎

As a corollary, we prove the main result in [Damm 90,BDLM **91**].

Corollary 6.6 ([Damm 90,BDLM 91]) *MATPOW over Z_p is complete for $\mathcal{MOD}_p\mathcal{LOG}$, where p is fixed prime.*

Proof: Since $(f - g) \equiv (f + (p-1)g) (\mathrm{mod}\ p)$, $\mathcal{DIFF}(\mathcal{MOD}_p\mathcal{LOG}) = \mathcal{MOD}_p\mathcal{LOG}$. ∎

Remarks:

- The problem of (s,t) path in a DAG may be generalized to (s,t_1,t_2) paths in a DAG, i.e., a path from s to t_1 and s to t_2. Clearly, the special $\mathcal{NLOG}$ machine is reducible to (s,t_1,t_2) path. As a decision problem, this problem is easily seen to be $\mathcal{NLOG}$- complete. But as a counting version, they are powerful enough to capture $\mathcal{DET}$!
- The matrix product must be undertaken in a linear fashion. Note that the familiar doubling argument does not seem to work. It does not pay to be greedy!
- Note that the same proof works for iterated matrix product *(ITMATPROD)* as well.
- It is however not clear to give a direct and clean proof starting from the determinant itself.
- It is trivial to see that *iterated integer product* is in $\sharp\mathcal{LOG}$.

7 Complete Problems, Ranking and Optimizing

We present some natural functions complete for the counting classes that we have defined. These functions are (variations of) ranking functions for certain class of languages. We show ranking $\mathcal{DCFL}$'s is complete for $\sharp$AuxPDA SPACE,TIME$(\log n, poly(n))$, ranking $\mathcal{CFL}$'s is complete for SpanAuxPDA SPACE,TIME$(\log n, poly(n))$, and max-word function for ranking $\mathcal{CFL}$'s is complete for OptAuxPDA SPACE,TIME$(\log n, poly(n))$. Using these with the results from the previous sections, we prove several results on the complexity of ranking. We also note a natural complete problem for $\sharp$AuxPDA SPACE,TIME$(\log n, poly(n))$; the problem of computing the output of a polynomial size and polynomial degree arithmetic circuit over non-negative inputs.

$r_{\sharp\mathcal{DCFL}}$:
Input: An encoding of a (one way) deterministic pushdown automata and a string x.
Output: The number of strings lexicographically smaller than x in the language accepted by the pushdown automata.

$r_{\sharp\mathcal{CFL}}$:
Input: An encoding of a (one way) nondeterministic pushdown automata and a string x.
Output: The number of strings lexicographically smaller than x in the language accepted by the pushdown automata.

$m_{Opt\mathcal{CFL}}$:
Input: An encoding of a (one way) nondeterministic pushdown automata and a string x.
Output: The largest string lexicographically smaller than x in the language accepted by the pushdown automata.

Theorem 7.1 **1.** *Evaluating polynomial size and polynomial degree arithmetic circuits over non-negative integer inputs is complete for* $\sharp$AuxPDA SPACE,TIME$(\log n, poly(n))$.

2. $r_{\sharp\mathcal{DCFL}}$ *is complete for* $\sharp$AuxPDA SPACE,TIME$(\log n, poly(n))$.

3. *Ranking languages accepted by* 1NAuxPDA SPACE,TIME$(\log n, poly(n))$ *is complete for* $\sharp$AuxPDA SPACE,TIME$(\log n, poly(n))$.

4. $r_{\sharp\mathcal{CFL}}$ *is complete for* SpanAuxPDA SPACE,TIME$(\log n, poly(n))$.

5. *Ranking languages accepted by* 1NAuxPDA SPACE,TIME$(\log n, poly(n))$ *is complete for* SpanAuxPDA SPACE,TIME$(\log n, poly(n))$.

6. $m_{Opt\mathcal{CFL}}$ *is complete for* OptAuxPDA SPACE,TIME($\log n, poly(n)$).

7. *Optimizing languages accepted by* 1NAuxPDA SPACE,TIME($\log n, poly(n)$) *is complete for* OptAuxPDA SPACE,TIME($\log n, poly(n)$).

Proof: The proofs are not difficult and they are deferred to the final version of the paper. ∎

.

Theorem **7.1** when combined with Corollory **4.8** yield simple and structurally strong proofs (in the light of the completeness result) of

Theorem 7.2 ([Huy 90]) *1. Languages accepted by* 1UAuxPDA SPACE,TIME($\log n, poly(n), c$) *an be ranked within* $\mathcal{NC}^2$.

2. *Languages accepted by* 1NAuxPDA SPACE,TIME($\log n, poly(n)$) *are* $\mathcal{P}$*- rankable iff* $P = \natural\mathcal{P}$. ∎

Combining Theorem **7.1** with Theorem **4.6** and Corollary **4.8** we also get

Theorem 7.3 *1. Languages accepted by* 1NAuxPDA SPACE,TIME($\log n, poly(n)$) *are* $\mathcal{NC}^2$*- optimizable.*

2. *Sparse* $\mathcal{CFL}$*'s are rankable in* $\mathcal{NC}^2$. ∎

We now show a "larger" P- rankable set.

Theorem 7.4
1UAuxPDA SPACE,TIME($\log n, exp(n), poly(n)$) *is* P*- rankable.*

Proof: The ranking function is in $\natural$Semi-Unbounded USIZE($poly(n)$)[$poly(n)$]. Theorem **4.11** completes the proof. ∎

The results above seem to imply that in some sense, optimizing is easier than ranking. The notion of $\mathcal{P}$-optimizable sets is a generalization of the notion of $\mathcal{P}$-printability, just as $\mathcal{P}$-rankability is. It is therefore interesting to explore how these classes differ. It is a well known fact that $\mathcal{P}$is $\mathcal{P}$-rankable if and only if $\mathcal{P} = \natural\mathcal{P}$[Hem 87]. We show that it is unlikely that $\mathcal{P}$is $\mathcal{P}$-optimizable either – unless $\mathcal{P} = OptP$. Clearly, this collapse is milder, in the light of Toda's theorem.

Theorem 7.5 *P is* P*- optimizable* iff $\mathcal{P} = OptP$.

Proof: ($\subseteq$) Suppose P is P optimizable. Consider the language
$L = \{F\#assign : assign$ is a vector that satisfies the formula $F\}$.
Then L belongs to $\mathcal{P}$. Also the problem of finding the lex-maximum assignment is complete for $OptP$ [Kre 88]. The result follows.
($\supseteq$) Easy. ∎

8 Open Problems

The relationship between uniformity and ambiguity is puzzling. It is straight forward to show any language accepted by an AuxPDAis also accepted by an AuxPDAwithin height logarthmic in time. However, in the resulting machine, the number of accepting paths increase. What then the relation between uniformity and ambiguity? And between uniformity and nondeterminism?

Can Valiant et.al's [VSBR 83] result be made $\mathcal{DLOG}$-uniform? Note that in our simulation, we need the degree of each gate as an advice. So the circuit need to be evaluated once for degree. This is similiar to the case of iterated integer multiplication where it need to be evaluated once [IL 89].

Acknowledgments
This paper owes much to the influence and work of two wonderful persons, Venkateswaran and Birgit Jenner. Venkateswaran's fundamental work on semi-unboundedness and Birgit's work (with Alvarez) on counting space classes are instrumental in starting this research.

I thank Venkateswaran for initially drawing my attention to the link between $\mathcal{DET}$ and $\mathcal{LOGCFL}$, as a possible application of my results in [V 90a]. I thank Birgit Jenner for drawing my attention to [BDLM 91] and encouraging me to add section 6. I thank Peter Rossmanith for drawing my attention to an error in an earlier draft. I thank Ashok Subramaniam for pointing out certain simplifications to the proof of the main theorem and more importantly for bringing [Damm 90] to my notice. The idea in theorem 5.2 is due to a chance remark by Ravi Kannan made in a entirely different context. I thank Veni Madhavan for his guidance and encouragement.
I thank Rengarajan and Laxmi for listening to the many wrong proofs and ideas without ever losing their patience (or sanity)!

References

[AJ 90] Alvarez, C., and Jenner, B., A Very Hard Log Space Counting Problem, Proc. 5th Structure in Complexity Theory Conference,(1990), 154-168.

[BDLM 91] Buntrock, G., Damm, C., Hertrampf, U., and Meinel, C., Structure and Importance of Logspace-MOD-Classes, STACS 91, LNCS 480, 360-371.

[BF 90] Babai, L., and Fortnow, L., A Characterization of $\sharp\mathcal{P}$ by Straight Line Programs of Polynomials, with Applications to Interactive Proofs and Toda's Theorem, Proc. 31st annual FOCS Symposium.

[BH 88] Buntrock, G., and Hoene, A., On Reversal Complexity of Auxiliary Pushdown Automata, Tech. Rep 88-11, Technische Universitat, Berlin.

[Co 71] Cook,S.A., Characterizations of pushdown machines in terms of time-bounded computers, *JACM* **18** ,(1971), 4-18.

[Co 85] Cook, S.A., **A** Taxonomy of Problems with Fast Parallel Algorithms,*Information and Control* 64, (1985), 2-22.

[Damm 91] Indirect communication via Birgit Jenner.

[Damm 90] Damm, C., Problems complete for $\oplus\mathcal{LOG}$, *Information Processing Letters* 36, (1990), 247-250

[Hem 87] Hemachandra, L., The Complexity of Ranking, Proc. 2nd Structure in Complexity Theory Conference,(1987), 103-117.

[Huy 90] Huynh, D. T., The complexity of Ranking Simple Languages, *Mathematical Systems Theory* **23**,(1990),1-20.(also in 3rd Structure Conference, 1987)

[IL 89] Immerman, **N.,** and Landau, S., The Complexity of Iterated Multiplication, Proc. 4th Structure in Complexity Theory Conference,(1989), 104-111.

[JK 88] Jenner,B. and Kersig,B., Characterizing the polynomial hierarchy by alternating Auxiliary pushdown automata *RAIRO theoretical Informatics and Applications* 23, (1989), 91-99.

[KST 89] Kobler, J., Schoning, U., and Torán, J., On Counting and Approximation, *Acta Informatica* 26, (1989), 363-379.

[Kre 88] Krentel, M., The Complexity of Optimization Problems. *Journal of Computer and System Sciences* **36**,(1988),490-509.

[Lad 89] Ladner, R., Polynomial Space Counting Problems, *SIAM Journal of computing* **18**,(1989), 1087-1097.

[RVVY 90] Ravi Kannan, Venkateswaran, **H.,** Vinay, V., and Yao, A. C., A Circuit-Based Proof of Toda's Theorem, to appear *Information and Computation.*

[RR 90] Regan, K. W., Royer J. S., A Simpler Proof of $\mathcal{PH} \subseteq$ BP.$\oplus\mathcal{P}$. Draft, May 1990.

[Ru 80] Ruzzo,W.L., Tree-size bounded alternation, *JCSS* 21(1980), 218-235.

[Ru 81] Ruzzo, W. L., On uniform circuit complexity, ***Journal*** of ***Computer and System Sciences 22,*** (1981),

[To 89] Toda, **S.,** On the computational power of PP and $\oplus\mathcal{P}$, Proc. 30th annual FOCS symposium, (1989), 514-519.

[Tr 90] Toran, **J.,** Counting the Number of Solutions, Tech rep. LSI-90-17, Department de Llenguatges i sistemes informatics, Universitat Politecnica de Catalunya.

[Val 79] Valiant, L. ***G.,*** The Complexity of Computing the Permanent, ***Theoretical Computer Science 8,*** (1979), 189-201.

[VSBR 83] Valiant, L. ***G.,*** Sykum, **S.,** Berkowitz, S., and Rackoff, C., Fast Parallel Computations of Polynomials using Few Processors, ***SIAM Journal on Computing*** **12**,(1983), 641-644.

[Ve 87] Venkateswaran,H., Properties that characterize $\mathcal{LOGCFL}$, Proceedings 19^{th} Annual ACM STOC, (1987), 141-150.

[Ve 88] Venkateswaran,H., Circuit definitions and nondeterministic complexity classes, Proc. 8^{th} FST & TCS, (1988), LNCS 338, 175-192.

[VC 90] Vinay, V. and Chandru, V., The Expressibility of Nondeterministic Auxiliary Stack Automata and its relation to Tree-size Bounded Alternating Auxiliary Pushdown Automata, Proc. 10^{th} FST & TCS, (1990), LNCS 472, 104-114.

[VVV 90] Vinay,V.,Venkateswaran, H. and Veni Madhavan,C.E., Circuits, Pebbling and Expressibility, Proc. 5th Structure in Complexity Theory Conference, (1990), 223-230.

[V 90a] Vinay, V., Counting Auxiliary Pushdown Automata and Semi-unbounded Arithmetic Circuits. Tech. Rep. Indian Institute of Science.